Dissertation

zur Erlangung des Doktorgrades der Technischen Fakultät der
Albert–Ludwigs–Universität Freiburg im Breisgau
— Doktor der Naturwissenschaften —

Long Term Motion Analysis for Object Level Grouping and Nonsmooth Optimization Methods

---

Peter Ochs

March 2015

Lehrstuhl für Mustererkennung und Bildverarbeitung
Technische Fakultät, Albert–Ludwigs–Universität
Freiburg im Breisgau

Tag des Kolloquiums

06.03.2015

Dekan

Prof. Dr. Georg Lausen

Prüfungsausschuss

Prof. Dr. Matthias Teschner	(Vorsitz)
Prof. Dr. Wolfram Burgard	(Beisitz)
Prof. Dr. Thomas Brox	(1. Gutachter)
Prof. Dr. Thomas Pock	(2. Gutachter)

Zusammenfassung

Die folgende wissenschaftliche Arbeit beschäftigt sich in einem ersten Teil mit konvexen und nichtkonvexen Optimierungsmethoden für bestimmte Problemklassen. Diese Optimierungsmethoden finden insbesondere Anwendung bei der Lösung vieler grundlegender Problemstellungen im Bereich Computer Vision. Als ein Beispiel sei an dieser Stelle die Berechnung des optischen Flusses zwischen zwei Bildern genannt. Diese Verbesserungen wirken sich wiederum positiv auf zahlreiche bewegungsbasierte Ansätze aus. Der zweite Teil dieser Arbeit konzentriert sich auf solch eine Methode, die Objekte in Videos auf der Grundlage einer Bewegungsanalyse segmentiert.

Ein Kapitel zur konvexen Optimierung erweitert die Analyse von der so genannten "Heavy-ball-Methode" von glatten, stark konvexen auf nichtglatte, stark konvexe Zielfunktionen. Diese Erweiterung ist analog zur Weiterentwicklung der Gradientenabstiegsmethode zu dem "forward–backward splitting". Der zentrale Beitrag ist der Nachweis, dass sich die Optimalität der bekannten Konvergenzrate der Heavyball Methode auch verallgemeinert. Der neue Algorithmus wird "iPiasco" genannt.

Es zeigt sich sogar, dass die generelle Idee des formalen Berechnens der neuen Iterierten, auch auf nichtkonvexe Probleme mit einer speziellen Struktur anwendbar ist. Das ist die Grundlage von "iPiano", der Verallgemeinerung von iPiasco auf nichtkonvexe Probleme. Die Anforderung an die Struktur, die wir zur Entwicklung von diesem Algorithmus stellen, fordert, dass sich die Zielfunktion als eine Summe einer glatten, nichtkonvexen und einer nichtglatten, nichtkonvexen, aber "einfachen" Funktion ausdrücken lässt. Im Gegensatz zum konvexen Fall, sind Konvergenzraten im nichtkonvexen nur schwer zu finden. Meist erschließt sich nur eine sehr grobe Abschätzung. Die Problematik für nichtkonvexe Optimierungsmethoden ist, ob der Algorithmus überhaupt konvergiert. Das Schlüsselkonzept für unseren Konvergenzbeweis, der auf der Verallgemeinerung des Beweises eines verwandten Algorithmus beruht, ist die so genannte Kurdyka–Łojasiewicz (KL) Eigenschaft. Die KL Eigenschaft ist eine schwache Annahme, die die meisten Funktionen, insbesondere im Anwendungsbereich, erfüllen. Daher ist iPiano ein vielversprechender Algorithmus für eine Fülle von Problemen.

Obwohl iPiano effizient ist und auch die Konvergenz gezeigt ist, ist er auf manche Probleme aus dem Bereich Computer Vision nicht anwendbar, da ein Teil der Zielfunktion als differenzierbar vorausgesetzt sein muss. Um auch solche, schwierigen Optimierungsprobleme lösen zu können, wird der "IRconvex-Algorithmus" eingeführt, welcher zu einer Klasse von Majorisierungs-Minimierungs-Ansätzen (MM) gehört. Für einige Funktionen, die in die Problemklasse von IRconvex fallen, werden explizit konvexe, majorisierende Funktionen, die sich dann recht einfach minimieren lassen, konstruiert. Dabei treffen wir auf schon bekannte Spezialfälle, wie zum Beispiel den "iteratively reweighted (IR) least squares" (IRLS) oder den "IR ℓ_1" (IRL1) Algorithmus. Jedoch erschließen sich auch neue Algorithmen, wie der "IRTight-" oder der "IRHuber-Algorithmus". In dem allgemeinen Rahmen von MM Methoden wird die Konvergenz für all diese Algorithmen bewiesen. Wieder spielt die KL Eigenschaft eine wesentliche Rolle. In einer ausgiebigen numerischen Analyse zeigt sich, dass die IRconvex-Algorithmen besonderes geeignet sind, wenn die Zielfunktionen einen nichtdifferenzierbaren Anteil haben, und ansonsten iPiano

effizienter ist.

Die bisher beschriebenen Ergebnisse setzen einiges an mathematischem Vorwissen voraus. Um diese Arbeit diesbezüglich zu vervollständigen und unabhängig zu verfassen, werden alle diese Grundlagen sorgfältig in eigenständigen Kapiteln aufgearbeitet.

Als eine Anwendung von IRconvex wollen wir die Berechnung des optischen Flusses herausstellen, die, wie zuvor erwähnt, weitreichende positive Einflüsse auf die Bewegungssegmentierung in Videos haben kann. Obwohl interessante Aspekte ansatzweise beobachtet werden können, bedarf es noch mehr Untersuchungen, um wirklich aus den Neuerungen praktischen Nutzen ziehen zu können.

Der zweite Teil der vorliegenden Arbeit beschäftigt sich mit einer Methode zur objektorientierten Segmentierung in Videos auf der Basis einer Bewegungsanalyse. Die grundlegenden Einheiten, woraus sich diese Methode zusammensetzt, sind: Schätzung des optischen Flusses, Tracking, Bewegungsanalyse von Punkttrajektorien und Gruppieren der Trajektorien. Es werden verwandte Arbeiten besprochen und die Grenzen des Segmentierungsansatzes diskutiert. Im Wesentlichen handelt es sich um die ausschließliche Analyse des translatorischen Anteils von Bewegungen und das Segmentieren von semi-dicht berechneten Trajektorien.

Da die Bewegungsanalyse auf Paaren von Punkttrajektorien beruht, lässt sich nur der translatorische Anteil vergleichen. Drehungen und Skalierungen werden bestraft und führen leicht zu einer ungewollten Unterteilung der Segmentierung innerhalb der Objekte. Die Idee, die Beziehung von mehr als zwei Trajektorien zueinander zu berücksichtigen, erlaubt es uns, auch solche etwas komplexeren Bewegungen straffrei zu beschreiben. Das darunterliegende Modell ist das eines Hypergraphen, dessen (Hyper-)Kanten mehr als zwei Trajektorien umfassen. Diese fundamentale Änderung in der Modellierung des Problems hat zur Folge, dass wir auch die spektrale Gruppierungsmethode anpassen müssen. Die erfolgreiche Umsetzung führt allerdings zu einer deutlichen Verbesserung auf einem anerkannten Benchmark.

Die zweite Einschränkung, die wir oben genannt haben, ist das semi-dichte Tracken. Obwohl der optische Fluss dicht berechnet wird, ist er jedoch oft in der Nähe von Objektkanten unzuverlässig und anfällig für Fehler. Ein positiver Aspekt des semi-dichten Tracking ist eine vergleichsweise geringe Anzahl an Trajektorien, die auch wesentlich für die Komplexität – sie ist quadratisch – bei dem Gruppieren verantwortlich ist. Daher sieht unser Ansatz so aus, dass zuerst die Trajektorien gruppiert, und dann die Labels der Trajektorien auf alle Pixel ausgeweitet werden. Erreicht wird dies durch einen variationellen Ansatz mit einem Potts-Regularisierungsmodell. Außerdem berücksichtigt diese Methode auch Bildkanten. Sie ist auf der GPU implementiert und die Genauigkeit der Segmentierung ist, obwohl sich nun jeder Pixel für ein Label entscheiden muss, vergleichbar mit der der semi-dichten Segmentierung.

Um die Genauigkeit der vorgeschlagenen Methoden zu messen und auch gegen andere Methoden zu vergleichen, stellen wir den Freiburg Berkeley Motion Segmentation (FBMS) Datensatz vor. Die Auswertung auf diesem Datensatz ist getrennt in eine Trainingsmenge und eine Testmenge von Sequenzen. Die Auswertungsmaße beruhen auf der Idee von Precision, Recall und dem F-Maß, das die Qualität in einer vereinheitlichten Zahl widerspiegelt. Basierend auf diesem Datensatz, der den weitläufig anerkannten BMS Datensatz erweitert, erweist sich die Methodik, die in dieser Arbeit vorgestellt wird, als neuester Stand der Technik.

Abstract

This work deals with theoretical and practical aspects of convex and nonconvex optimization algorithms for several classes of problems. They are applied to several (low-level) computer vision tasks. The optical flow motion estimation problem, which is among them, is a potential source for improving motion based segmentation methods. The second, more practical part of this work focuses on such an object-level motion segmentation method for videos.

The part about convex optimization extends the analysis of the so-called "Heavy-ball method" for smooth, strongly convex objective functions to nonsmooth, strongly convex problems. The algorithm that is developed is motivated by the extension of the gradient descent method to forward–backward splitting methods. Its optimality in the sense of rates of convergence for the considered class of problems is proved. We termed the algorithm "iPiasco". It extends the optimal rate of convergence that is known from the Heavy-ball method to the generalized setting.

The iteration structure of iPiasco, i.e., the way the next iterate is computed, even generalizes to a certain class of nonconvex optimization problems. This opportunity is used to introduce "iPiano", which is an algorithm that is applicable to the sum of a nonsmooth, nonconvex function and a smooth, nonconvex function. Since convergence rates in the nonconvex setting are hard to find, the focus of this research is to proof convergence. Key for the convergence analysis is the so-called Kurdyka–Łojasiewicz (KL) property. Convergence is obtained by extending a preexisting convergence result for abstract descent methods to fit the setting of iPiano. As the KL property is a rather weak assumption—many functions emerging in computer vision are shown to have this property—iPiano is a promising algorithm for a wealth of problems.

Although iPiano is an efficient algorithm that is proved to converge, it is limited to composite functions where one part is differentiable. In order to deal with hard, nondifferentiable objective functions the "IRconvex" algorithm is introduced. It belongs to the class of majorization minimization (MM) algorithms. For several instances of problems explicit convex majorizers are constructed and minimized. This leads to well-known algorithms like the iteratively reweighted (IR) least squares (IRLS) or the IR ℓ_1 (IRL1) algorithm, but also allows for new algorithm like the IR tight convex (IRTight) or the IR Huber algorithm (IRHuber). In the unified, much more general framework of MM algorithms, convergence is proved under some mild assumption, among them the KL property. An extensive numerical analysis of IRconvex algorithms shows that it is particularly efficient for nonsmooth optimization problems, whereas iPiano should (in most cases) be preferred when a part of the function is smooth.

The part of the thesis about optimization attaches importance to a thorough introduction of basic results from convex and nonconvex analysis and a recapitulation of the most prominent convex optimization algorithms (for computer vision applications).

IRconvex is applied to the optical flow problem in computer vision. It shows some favorable properties, however further investigation is required to benefit in practice. There is potential to improve optical flow and therefore also many motion based algorithms. The second part of the thesis is devoted to motion

segmentation in videos. The basic building blocks of the method that is considered are: optical flow, point tracking, motion analysis of point trajectories, and clustering of point trajectories. After introducing the method and related work, the limitations are addressed. These are the analysis of trajectories only based the translational portion of the motion and the semi-dense tracking.

As the motion analysis of point trajectories is based on pairs of trajectories, only the translational portion of the motions can be analyzed. Rotation and scaling of objects are assigned a penalty and the objects are likely to be split. Considering higher order relationships of trajectories can be done in the model of hypergraphs, where (hyper-)edges can comprise more than two trajectories. The proposed analysis of such higher order cliques and the adaptation of spectral clustering to this setting improves the performance on a well-established benchmark.

Another issue is the semi-dense tracking of points. Although optical flow is computed densely, its reliability close to object boundaries reduces and tracking is prone to errors. Moreover the clustering of the trajectories scales quadratically. Therefore the semi-dense trajectories are clustered first, and then their label information is propagated to unlabeled areas in the video. This is achieved by a variational approach with Potts regularization. The method considers image structures, runs on the GPU, and the performance of the (now dense) motion segmentation approach is as good as the semi-dense segmentation.

The performance of the proposed motion segmentation approach is compared with other methods on our new benchmark, the Freiburg Berkeley Motion Segmentation (FBMS) dataset. The evaluation of the FBMS dataset is split into a training and a test set. The evaluation metric uses the idea of precision, recall, and F-measure, which allows for a single comparable number between methods with different focus. Based on this benchmark, which comprises the widely used BMS dataset, the framework that is proposed in this thesis is considered as state-of-the-art.

TO SABRINA AND MY FAMILY

Acknowledgements

First of all I want to thank my supervisor Thomas Brox. For me he was the best supervisor for entering the world of research in computer vision. Although he just became a professor, I had the feeling that he always knew the right way to go. I am also very thankful that I could learn from his outstanding way of writing research articles and presenting them. He gave me all the freedom I needed to work on projects that I liked most. He always supported collaborating with others and enabled several research trips to TU Graz for me, where I could learn from Thomas Pock. Thomas, I thank you a lot for accepting me as a PhD student.

I also thank Thomas Pock for his guidance and his invitation to do research with him. I still remember the situation when I asked him to teach me how to work more mathematically. It was quite hard to work through the first stack of papers and books he immediately pointed me to. I am very grateful to him for opening the door to optimization for me. I really enjoyed all the discussions. Tom, I thank you for all that and, particularly, for your enthusiasm for the subject that is really infectious.

For the joint work I want to thank all my co-authors. Here, I want to particularly highlight the contributions of Alexey Dosovitskiy. Without him Chapter 7 would have been less attractive and general. Thank you for always having a solution and a sound understanding of the details.

Moreover, all the work was only possible in a nice and balanced working atmosphere at LMB. I am grateful to all the discussions with my room mate Naveen S. Nagaraja. However, his contribution goes far beyond research. I also want to explicitly thank Benjamin Ummenhofer for several discussions, the support in GPU programming, and the solutions for (the many) compilation problems I had. I also acknowledge Thorsten Schmidt's assistance for similar issues. Beside the research discussions at LMB, I enjoyed the after-lunch table soccer a lot. Thanks to all participating players: Naveen, Benjamin U., Nikolaus, Thorsten, Margret, Maja, Benjamin D., Dominic, Philipp, and Alexey.

I also highly acknowledge the proofreading of this thesis by Frank Schmidt, Naveen, Alexey, Thomas B., and Thomas P..

I am also grateful for the technical support by Stefan Teister et al. and to Cynthia Findlay for her help with organizational questions.

Last but not least, I want to thank my family. They supported me in so many ways and pushed me to pursue my way. Although they did not understand what I am doing, they were interested, asked questions, and, most important, always believed in me. I owe ∞ thanks to my wife for accepting me as I am and endorsing me to do what I like. Sabrina, thank you for your love.

Contents

Introduction

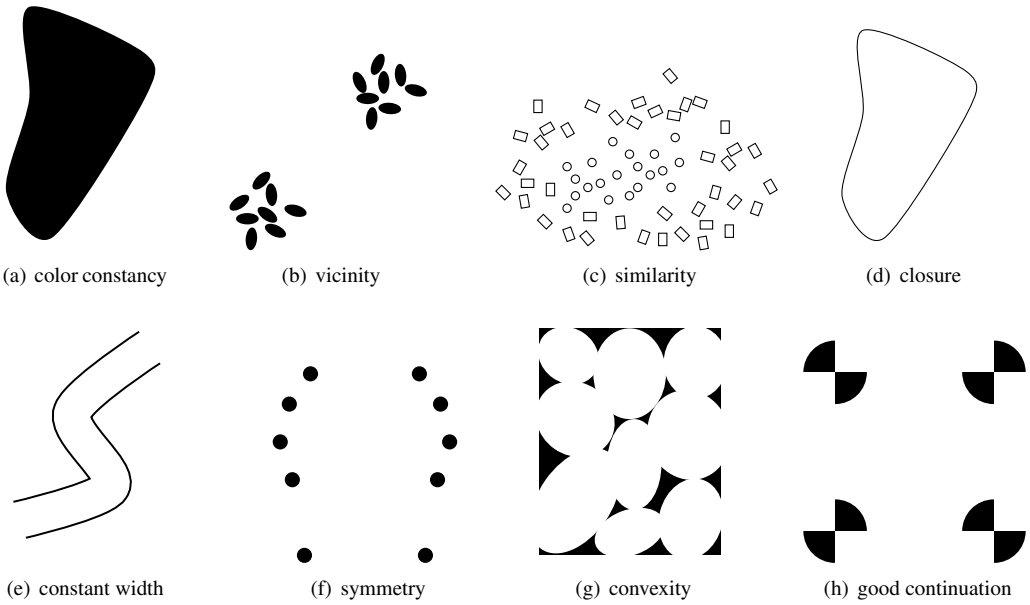
The ultimate goal of computer vision is to imitate the human visual system. This includes not only the perception, but also the analytic process. Given the computer an arbitrary image with an object, we would expect it to detect whether there is a person, an animal, an airplane, ..., or even more detailed information. Moreover, the exact position and size of the object in the image is of interest. Contemporary object detectors achieve this by learning a representation of each object to be sought. Then they verify the learned representation in the image. However, where does the machine learn it from? It learns from training examples showing the object class in different variations, (e.g. different types, poses, scales, etc.). In order to select these training examples automatically, the machine must have already learned the representation of the object class and must be able to detect it. This is like a "chicken-and-egg problem": in order to learn an object detector, the machine requires training examples where the object of interest is present, which can be obtained using the specific object detector. The simplest solution is to manually provide a set of training examples. Unfortunately, depending on the complexity of the object to be learned the number of training examples is of the order hundred or thousand—usually, the more training examples, the better.

There is great interest in generating such training examples automatically with little or without supervision. Potentially interesting for this task are unsupervised image segmentation techniques. Although a lot of research was done on image segmentation and methods improved significantly, the intrinsic ambiguity still requires some top-down knowledge to make the results usable as training examples. For example in Figure 1, how could the machine—without being provided with the knowledge from top-down—know that the white shirt and the black vest should be separated from the dark background and, at the same time, they should be grouped as a single object. In general, this is not possible.

Figure 1: In the image, separating the man from the background without any information about the human's appearance is nearly hopeless.

Prior knowledge for image segmentation or grouping is based on simple rules. Such rules are explored in the so-called Gestalt theory. According to the Helmholz principle [Low85] "Gestalts are sets of points whose (geometric regular) spatial arrangement could not occur in noise", see [DMM07]. This is still very abstract and hard to formulate as rule for grouping. There are other definitions: Gestalt theory is based on the assumption that there are active grouping laws in visual perception [Kan80, Kan97, Wer23] and aims to understand the activation of visual stimuli. Points that share attributes are grouped and represent a (new) larger object. This is a gestalt. Wertheimer formulated this in his founding paper 1923 [Wer23] as follows (translated):

> "If a number of stimuli acts simultaneously, for the human there is, in general, not a corresponding ('equally large') number of individual actualities, one and the other and the third



(a) color constancy (b) vicinity (c) similarity (d) closure

(e) constant width (f) symmetry (g) convexity (h) good continuation

Figure 2: Examples for (basic) Gestalt principles formulated by Wertheimer [Wer23] and others. (Figures are reproduced from [DMM07])

> and so on; instead there are actualities of larger range, in a determinate distinction, determinate conjunction, determinate separation. And whatever the theoretical concept may be, whether one—far away from the plain finding, for theoretical reasons—supposes, though, the sum of the '[many] sensations' as basis, there is a plain problem of facts for any concept: Are there principles for the type of so resulting 'conjunction' and 'separation'? Which ones?"

Figure 2 shows a few examples. Note, that we do not claim at all to present a complete list of Gestalt principles. This would be far beyond the scope of this thesis and our purpose. Unfortunately, none of these principles would help us group the image in Figure 1.

However, many problems of image segmentation are resolved when the object's motion is considered. Motion is much more homogeneous within objects. This phenomenon is known as the *Gestalt principle of "common fate"* [Kof35]: particles that move homogeneously are perceived as a single object; see Figure 3. Figure 4 demonstrates the difference for the example in Figure 1. Not all ambiguities can be resolved. Two people walking next to each other in the same direction are (according to the Gestalt principle of common fate) perceived as one object. Obviously, the same is true for a segmentation method that is built upon this.

Similar problems occur in the presence of articulated motion, i.e., the possibly distinct motion of individual parts of a single object. Usually, considered over a short time, legs and arms of a person move differently than the person. Nevertheless, a good visual system should be capable of grouping articulated parts as one. Tracking them over a longer time shows a common trend of motion among them, namely that of the person. The motion segmentation framework that is developed in this thesis builds on and combines the two preceding principles: grouping according to the Gestalt principle of common fate and the long term analysis of moving particles.

Therefore, an essential part is the extraction and tracking of the particles. Henceforth, the trace of such a particle is called a point trajectory. The success of generating point trajectories heavily relies

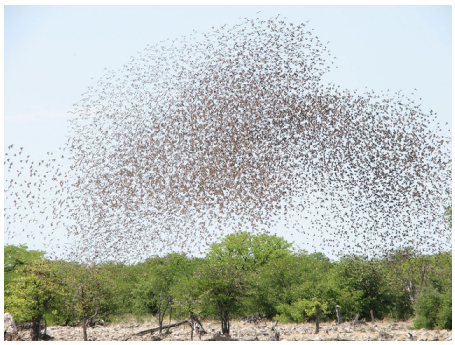


Figure 3: The flock of birds is perceived as unity [WIK]. This is explained by the Gestalt principle of "common fate".

Figure 4: The behavior of pixels throughout the video sequence provides distinct information for separating the person from the background.

on the ability to find the apparent motion between two successive frames of the video sequence—the so-called optical flow problem. As only the images are known to the computer, it can only try to find the motion between the projections of the 3D world points onto the image plane. Estimating the optical flow between images itself is a very difficult task. Despite its investigation for more than three decades, it is still not solved satisfactorily. Unsolved problems are still the estimation of large displacements, sharp object boundaries, and, particularly, problems due to occlusion, to name only a few of the major challenges.

Like other parts in the motion segmentation framework, the optical flow estimation problem is formulated in a variational framework, which is nowadays quite common for many tasks in computer vision. The rationale is the definition of an energy functional, which assigns a real value (an energy value) to each possible configuration, such that the state with lowest energy yields the desired solution. Variational approaches convince by their transparency, i.e., it is clear what can be expected from the optimal state of the energy. Clearly, in practice it is a little bit more complicated as the second part of an energy formulation is not less important: the optimal state of the energy must be determined efficiently.

A class of optimization problems that can be solved involves functions that are convex. Another class, which usually allows for more accurate models, are nonconvex functions. They are in general more difficult so solve. Therefore there is a trade-off between accuracy of modeling and ease of solving the variational problem. Unfortunately, some of the most challenging problems in optical flow estimation should be modeled with nonconvex functions, for example the occlusion problem.

As nonconvex optimization problems are in general too hard to solve, subclasses of problems are considered. The more task-specific the problem the more hope to find solutions efficiently. The development of numerical solvers for nonconvex optimization problems that are efficient on one hand and general enough for computer vision (and related fields) on the other hand are also subject of this thesis. A central question in the design and analysis of these methods is about their reliability, i.e., theoretical proofs of whether their usage leads to the desired solution (convergence analysis).

With efficient and reliable solvers for nonconvex optimization problems at hand, the next step is to achieve the practical advantages that nonconvex models promise. An interesting application is automatic occlusion detection in the optical flow problem, which could lead to far-reaching improvements in many computer vision problems. Particularly, point trajectories could be extracted more reliably, which would improve the whole motion segmentation framework and, therefore, we would achieve further progress in

the automatic learning of objects.

Outline and contributions

The thesis consists of two parts: “Optimization” and “Motion Segmentation in Videos”. The relation between both parts has been outlined in the introduction above.

Before the part on optimization begins Chapter 0 discusses a few very basic comments about images and their mathematical interpretation as functions or vectors. This discussion intends to help readers who are new to computer vision to get familiar with the terminology and conventions.

Part I contains the contributions of this thesis to the field of optimization. As it might be not clear how and where optimization problems arise in computer vision, Chapter 1 introduces the kind of optimization problems that we usually face in computer vision applications. On a more abstract level the problems can be grouped into categories according to the structure of the objective function. A rough, but important distinction, is between convex and nonconvex problems.

For many convex optimization problems there are efficient numerical algorithms. In order to introduce some of the most prominent methods some theory from convex analysis is required, which is described in Chapter 2. This chapter starts on a very basic level that enables the reader to learn all required techniques for understanding the remainder of the thesis. The reader who is familiar with convex analysis might want to skip Section 2.1 and consult this section only for definitions and notational issues. The second part of this chapter is devoted to the introduction of convex optimization algorithms. They are put into a unifying framework to highlight relations among the methods.

The subsequent Chapter 3 describes my contribution from [OBP15]. Subject of this chapter is the generalization of an algorithm that has a long tradition; It was introduced by Polyak in 1964 [Pol64] and named the “Heavy-ball method”. The generalization that we propose in this chapter (named iPiasco) is from differentiable, strongly convex functions to nonsmooth, strongly convex functions. Although the algorithm is more general, the efficiency that we prove is the same as before. Key to this contribution is the so-called proximal map.

However, as we described in the introduction above, the main motivation for the optimization part were nonconvex optimization problems. They are usually much harder than convex problems. Again we introduce all basics for understanding our contributions in later chapters. Chapter 4 introduces several important concepts from variational analysis in a compact way. A more expanded treatment of these basics is beyond the scope of the thesis. Section 4.5 deals with the Kurdyka–Łojasiewicz property of a function and a general class of functions that naturally have this property. It provides a powerful inequality for proving convergence of algorithms for nonconvex functions.

Indeed, it is used to prove an abstract convergence theorem in Chapter 5 for descent methods. This contribution to the thesis is developed as a part of [OCBP14]. It is a modification of a similar result from Attouch et al. [ABS13]. Our result seems to be better suited for algorithms that incorporate not only the last point into the update step, but the last two points.

The remaining part of the contribution in [OCBP14] is described in Chapter 6. It is an algorithm for a general class of nonconvex optimization problems with an update rule that is related to that in Chapter 3. We call the algorithm iPiano. Where [OCBP14] applies to problems that can be written as a sum of a nonsmooth, convex, simple and a smooth, nonconvex function, Chapter 6 presents a more general framework. It considers objectives that are the sum of a nonsmooth, nonconvex, simple function and a smooth, nonconvex function. Convergence of this algorithm is proved under the assumption that the objective function has the Kurdyka–Łojasiewicz property. Several experiments with computer vision

problems confirm the efficiency of the algorithm. The experiments were mainly contributed by the co-authors of [OCBP14].

In some sense Chapter 7 can be seen as a generalization of the class of problems that can be tackled with (the original) iPiano (from [OCBP14]); The nonconvex part of the objective function can also be nonsmooth. However the structure of the algorithm (IRconvex) is different to iPiano. This chapter is based on the manuscript [ODBP15]. The IRconvex algorithm is a majorization minimization algorithm. In fact this interpretation is used to prove convergence of several special instances of the algorithm in an unified framework. The proof is again based on the Kurdyka–Łojasiewicz property of the objective function. The instances of the algorithm that we propose are iteratively reweighted algorithms. These concrete constructions of the algorithm are extensively tested in numerical experiments and applied to the optical flow problem in computer vision.

Nonconvex penalties for modeling the optical flow problem have several favorable properties. In the first part we introduce algorithms that can efficiently solve such models. However improving results compared to the well established convex models is hard and requires further investigation. Nevertheless nonconvex penalties are a potential source for improving the optical flow and, therefore, also the motion segmentation framework that is presented in Part II of this thesis.

Chapter 8 introduces the state-of-the-art [BM10] for motion segmentation in videos and discusses related work. Within the scope of a journal article [OMB14], I contributed to this motion segmentation framework. The idea of this method is to analyze the motion behavior of a semi-dense set of point trajectories generated by optical flow. The estimated similarities are used for clustering the trajectories, which yields a temporally consistent, semi-dense segmentation of a video shot.

The subsequent chapters present my contributions to several conferences. Chapter 10 improves the computation of the similarities between the trajectories. The approach, which we published in [OB12], overcomes the limitation of [BM10, OMB14] that only the pairwise relationship between trajectories is explored. Pairwise considerations allow to capture the similarity of motions with respect to their translational portion. The model in Chapter 10 proposes a way to analyze more complex motions like those described by similarity transformations.

Chapter 11 focuses on various clustering tasks along the motion segmentation framework. The contribution, which is inspired by [OB11, OMB14], deals with the clustering of the trajectories according to the motion based similarities. Moreover, this chapter presents ways towards a densification of the semi-dense set of trajectories to a per pixel decision for each pixel in a video shot for one of the clusters. This approach can be considered as an interpolation of the cluster labels to all other pixels with an automatic error correction.

In order to prove that the presented way for motion segmentation is successful in many cases a pre-existing benchmark for this task was extended and introduced in [OMB14]. Chapter 9 discusses the new benchmark and evaluation metric and compares it with the previous benchmark. As this Freiburg Berkeley Motion Segmentation benchmark (FBMS) is split into a training and a test set, it allows for a clean comparison.

Chapter 12 evaluates the proposed motion segmentation method on the FBMS benchmark and compares it to several other methods.

Finally, Chapter 13 concludes the thesis and presents some ideas for future work.

Publications

Parts of this thesis have been published in international journals and have been presented at conferences.

First author publications

Journal Articles

- P. Ochs, T. Brox, and T. Pock. iPiasco: inertial proximal algorithm for strongly convex optimization. *Journal of Mathematical Imaging and Vision*, 2015.
- P. Ochs, Y. Chen, T. Brox, and T. Pock. iPiano: inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419, 2014.
- P. Ochs, A. Dosovitskiy, T. Brox, and T. Pock. On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. *SIAM Journal on Imaging Sciences*, 8(1):331–372, 2015.
- P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1187–1200, June 2014.

Conferences

- P. Ochs and T. Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *International Conference on Computer Vision (ICCV)*, 2011.
- P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- P. Ochs, A. Dosovitskiy, T. Pock, and T. Brox. An iterated ℓ_1 algorithm for nonsmooth nonconvex optimization in computer vision. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

Other publications

Journal Articles

Conferences

- N.S. Nagaraja, P. Ochs, K. Liu, and T. Brox. Hierarchy of localized random forests for video annotation. In *Annual Symposium of the German Association of Pattern Recognition (DAGM)*. Springer, LNCS, 2012.
- M. Temerinac-Ott, O. Ronneberger, P. Ochs, W. Driever, T. Brox, and H. Burkhardt. Multiview deblurring for 3-D images from light sheet based fluorescence microscopy. *IEEE Transactions on Image Processing*, 21(4):1863–1873, 2012.

Chapter 0

Preliminaries of computer vision

0.1 Representation of images

When we mathematically talk about images there are two common representations: continuous and discrete.

In the continuous formulation, a two-dimensional image[1] can be represented as a function $I\colon \Omega \to \mathcal{R}$, where Ω is usually a rectangular bounded subset of $\mathbb{R}^2$, e.g., $\Omega = [0, a] \times [0, b]$, $a, b \in \mathbb{R}_+$. The range of the image function $\mathcal{R} \subset \mathbb{R}^K$ models the color or gray values of the image ($K = 1$ for gray values and, for example, $K = 3$ for RGB-values). Images can have various properties as functions. For example they can be continuous, (weakly) differentiable, Lebesgue integrable, thus, they are from different spaces like the L_p-space or the Sobolev space. For specific tasks from image processing, it is crucial to know the underlying image model, i.e., the function space.

In the discrete formulation, images are represented as vectors in an Euclidean vector space $\mathbb{R}^N$. Both formulations, vectors in an Euclidean vector space, or functions in an infinite dimensional space, have their advantages and it is useful to always have both representations in mind. In fact, a discrete formulation can always be derived from the continuous formulation by sampling the function at discrete point. Usually, this sampling is on a regular Cartesian grid of the domain of the image function, denoted the pixel grid. Going from a discrete to a continuous formulation is possible by interpolation, however going this direction is usually avoided since additional information about the image must be invented and therefore it is not unique.

Thanks to the rich structure and properties the continuous formulation is preferably used for (theoretically) analyzing solutions and modeling optimization problems for computer vision problems. The discrete formulation is particularly important for practically solving problems. In order to apply optimization algorithms objects from the "continuous world" need to be transfered to the discrete case.

In the computer vision community the two concepts are often merged—one should always have both concepts in mind.

0.2 Discretization

Part I of this thesis deals with optimization algorithms for practical problems in image processing and

[1] For simplicity, we restrict to a two dimensional domain. It is straight forward to generalize the concept to any dimension.

computer vision. These algorithms are formulated on finite dimensional vector spaces. Therefore, we solve problems using the discrete representation of images. However, the models that are optimized are motivated in the continuous setting and, therefore, continuous objects like derivatives need to be imitated in the discrete setting. We think of our image model as a function discretized by sampling on a regular Cartesian grid with pixel width and height $h \in \mathbb{R}_+$ with N_x points in x-direction and N_y points in y-direction. Let $I\colon \Omega \to \mathbb{R}$, where $\Omega = [0, N_x h] \times [0, N_y h]$, be a gray-valued image function. The sampling defines the grid

$$\{((i - \tfrac{1}{2})h, (j - \tfrac{1}{2})h) \in \Omega |\, i = 1, \dots, N_x, j = 1, \dots, N_y\}$$

and a discrete representation $u \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_y}$ of the image via $u_{i,j} = I((i - \frac{1}{2})h, (j - \frac{1}{2})h)$. Obviously, there is a one-to-one correspondence between the discrete image defined on the grid and an Euclidean vector space $\mathbb{R}^N$ of dimension $\dim \mathbb{R}^N = N_x N_y$ by concatenating the columns of the grid into a vector, i.e., $u_{i,j} \longleftrightarrow u_{(i-1)N_x+j}$. We will work with both representation as both have their advantages, however it will always be clear from the context; If the index is a pair we assume the grid is defined as above without mentioning it.

As mentioned above, we want to imitate continuous objects on such a pixel grid. The gradient of a differentiable function can be represented in the discrete setting using a tensor–vector multiplication between a tensor $\nabla\colon \mathbb{R}^N \to \mathbb{R}^N \times \mathbb{R}^N$ and a vector (an image), where the range of ∇ is arranged as a column. The gradient operator is defined for $u \in \mathbb{R}^N$ as follows

$$\nabla := \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}, \quad \partial_x, \partial_y\colon X \to X\,,$$

$$(\partial_x u)_{i,j} := \begin{cases} \dfrac{u_{i+1,j} - u_{i,j}}{h}, & \text{if } i \in \{1, \dots, N_x - 1\}, j \in \{1, \dots, N_y\}\,, \\ 0, & \text{if } i = N_x, j \in \{1, \dots, N_y\}\,, \end{cases}$$

$$(\partial_y u)_{i,j} := \begin{cases} \dfrac{u_{i,j+1} - u_{i,j}}{h}, & \text{if } i \in \{1, \dots, N_x\}, j \in \{1, \dots, N_y - 1\}\,, \\ 0, & \text{if } i \in \{1, \dots, N_x\}, j = N_y\,, \end{cases}$$

where the implementation of the image boundary assumes that the gradient of the image function vanishes along the boundary. This kind of boundary treatment is called Neumann boundary conditions. Then, the (discrete) gradient ∇ of the (discrete) image $u \in \mathbb{R}^N$ is defined by

$$(\nabla u)_i := \begin{pmatrix} (\partial_x u)_i \\ (\partial_y u)_i \end{pmatrix}, \quad \forall i \in \{1, \dots, N\}\,,$$

where we make use of the identification $u_{i,j} \longleftrightarrow u_{(i-1)N_x+j}$. Other notations are also common, however this one seems to be appropriate for our usage. If not stated differently, we will always assume that the pixel dimensions are given by $h = 1$. For one dimensional functions (signals instead of images) we use the notation $\nabla = \partial_x$ for the discrete derivative operator.

As $\mathbb{R}^N$ and $\mathbb{R}^N \times \mathbb{R}^N$ are Euclidean vector spaces, an inner product is defined:

$$\langle u, v\rangle_{\mathbb{R}^N} := \sum_{i=1}^{N} u_i v_i \quad (u, v \in \mathbb{R}^N)\,, \quad \text{and} \quad \langle p, q\rangle_{\mathbb{R}^N \times \mathbb{R}^N} := \sum_{i=1}^{N} p_i^x q_i^x + p_i^y q_i^y \quad (p, q \in \mathbb{R}^N \times \mathbb{R}^N)\,,$$

where the superscripts in the second expression refer to the first and second coordinates of $\mathbb{R}^N \times \mathbb{R}^N$. The inner products induce a norm $\|\cdot\|_{\mathbb{R}^N} := \sqrt{\langle \cdot, \cdot\rangle_{\mathbb{R}^N}}$ and $\|\cdot\|_{\mathbb{R}^N \times \mathbb{R}^N} := \sqrt{\langle \cdot, \cdot\rangle_{\mathbb{R}^N \times \mathbb{R}^N}}$, respectively. Whenever the underlying space is intuitively known from the context, we drop the subscript at the inner product symbol.

Using the inner product, we define the adjoint operator ∇^* (or $\nabla^\top$), which emerges as the discrete divergence operator $\nabla^* = -\operatorname{div}$, by

$$\langle \nabla u, p \rangle_{\mathbb{R}^N \times \mathbb{R}^N} = - \langle u, \operatorname{div} p \rangle_{\mathbb{R}^N} \quad \forall u \in \mathbb{R}^N, p \in \mathbb{R}^N \times \mathbb{R}^N .$$

In the remainder of this section, we want to clarify some more notation and introduce some more norms that will be play a role throughout the thesis. The absolute value of $u_i \in \mathbb{R}$ and the Euclidean norm, the length, of a vector $p_i \in \mathbb{R}^2$ are denoted by

$$|u_i| := \begin{cases} u_i, & \text{if } u_i \geq 0 , \\ -u_i, & \text{if } u_i < 0 , \end{cases} \quad \text{and} \quad |p_i| := \sqrt{(p_i^x)^2 + (p_i^y)^2} .$$

We define some norms for $u \in \mathbb{R}^N$ and $p \in \mathbb{R}^N \times \mathbb{R}^N$:

$$\|u\|_2 := \left(\sum_{i=1}^N |u_i|^2 \right)^{\frac{1}{2}}, \quad \text{and} \quad \|p\|_2 := \left(\sum_{i=1}^N |p_i|^2 \right)^{\frac{1}{2}}, \qquad (\ell_2\text{-norm})$$

$$\|u\|_1 := \sum_{i=1}^N |u_i| , \quad \text{and} \quad \|p\|_1 := \sum_{i=1}^N |p_i^x| + |p_i^y| , \qquad (\ell_1\text{-norm})$$

$$\|u\|_{2,1} := \sum_{i=1}^N |u_i| , \quad \text{and} \quad \|p\|_{2,1} := \sum_{i=1}^N |p_i| . \qquad (\ell_{2,1}\text{-norm})$$

Part I

Optimization

Chapter 1

Optimization problems in computer vision

Many computer vision problems are solved in a so called variational formulation. The goal is to find a state of a function for which the variation vanishes. Such a state is for example the minimum (or maximum or saddle point) of a function. To be more precise the variational formulation is the design of a function such that the minimum corresponds to the solution of the problem at hand. The term "energy formulation" is also common. In this terminology, the state of minimal energy is sought. Problems for which minimization is equivalent to maximizing the negative function, we always focus on the minimization problem.

Solving a problem in an energy formulation consists of two steps: modeling the energy and solving it for a minimum. Unfortunately, these two steps are often complementary in the sense that better models are more difficult to optimize and simpler models are easier to solve. The modeling step is a formulation of assumptions how a good solution looks like, for example a good solution should be smooth. Deviations from the model assumptions yield a higher energy value. Therefore, the optimal solution of such a variational energy formulation is the optimal compromise between all the model assumptions. Where the modeling can be posed as a continuous or a discrete problem, i.e., the minimizer of the energy is a function or a vector, the optimization part (at least to obtain a numeric solution) must be posed as a finite dimensional problem.

As this part of the thesis focuses on developing and analyzing optimization algorithms, we consider energy functions $E\colon \mathbb{R}^N \to \mathbb{R}$ on a real Euclidean vector space and seek for a vector $\hat{u} \in \mathbb{R}^N$ that minimizes the energy. For many computer vision problems the solution vector corresponds to a (discrete) image, which itself corresponds to a sampled (continuous) image function (cf. Chapter 0). Mathematically, we want to solve for the minimizer of

$$\min_{u\in\mathbb{R}^N} E(u), \qquad \text{i.e. } \hat{u} := \arg\min_{u\in\mathbb{R}^N} E(u)\,. \tag{1.1}$$

Actually, using minimum in (1.1) instead of infimum requires some a priori consideration (see also Section 2.1.3). In general, it is not clear whether the minimum exists, i.e., whether $\min = \inf$. As this problem is rather a modeling issue than an optimization issue, we always assume that a minimizer exists.

Energy formulations have several advantages compared to ad-hoc solutions:

- The model assumptions must be clearly stated and even quantified in terms of penalties.

- There are no hidden details, if the energy is optimized properly.
- Imposing a local relationship between pixels asks for a global agreement on the model among all pixels.

Energies for computer vision problems often consists of two terms

$$E(u) := F(u) + \lambda R(u)\,, \tag{1.2}$$

the *data (fidelity) term* F and the *regularization term* or *penalizer* R, and a nonnegative *weighting parameter*. The data term penalizes the deviation from a certain data fidelity measure. The regularization term, which is also called *smoothness term*, regularizes the energy such that the solution is more plausible. Usually, it is based on prior knowledge about a general solution of the problem. "Smoothness" refers to a (locally) low variation of the image pixel values. The level of smoothness depends on the weighting parameter $\lambda \geq 0$. For larger values of λ the regularization term is more important and we expect a smoother solution, whereas small values of λ are associated with a high confidence of the solution to satisfy the model for the data term. The data term is specific to the task at hand. The smoothness term also has to be adapted to different tasks, however, the most common ones obey certain common principles.

Let us consider the example of image denoising from computer vision to make the energy modeling more concrete. In the task of image denoising the input is a noisy image $g \in \mathbb{R}^N$ and the goal is to recover a clean image. Assumptions about the noise formation process simplify the problem. We assume that there is an "original image" $h \in \mathbb{R}^N$ without noise and a noise function $n \in \mathbb{R}^N$ such that the observed noisy image is given by

$$g = h + n\,,$$

i.e. we assume an additive noise model. Under these model assumptions we want to find the image $u \in \mathbb{R}^N$ that approximates the clean image h as good as possible such that in the optimal case $u = g - n$. Unfortunately the noise n is unknown. However, it is common to assume some knowledge about the distribution of the noise. A possible data term for this problem is

$$F(u) = \|u - g\|_2^2\,. \tag{1.3}$$

The data term in (1.3) is particularly suitable when the noisy image can be modeled by an independent and identically distributed *Gaussian noise model* (normal distribution) with zero mean. We present here only an ad-hoc motivation. For a normal distribution with zero mean the probability to draw zero is the highest, low values occur frequently, and high values rarely. The quadratic data term models that. Small deviations of the sought solution from the noisy image are penalized little, whereas if the sought solution deviates a lot from the noisy image a high penalty (growing quadratically) is assigned.

On the other hand, this reasoning shows that quadratic penalizers are not good when *impulse noise* has deteriorated the image. Impulse noise or *salt and pepper noise* would lead to high penalties everywhere. A better penalty term only penalizes whether a pixel is corrupted or not. The best convex approximation to such a term is the absolute deviation. Assume that $g \in \mathbb{R}^N$ was recorded with impulse noise. Then g can be denoised with $u \in \mathbb{R}^N$ by minimizing (1.2) with

$$F(u) = \|u - g\|_1\,. \tag{1.4}$$

Figure 1.1 shows an example for image denoising with these two models of noise. The results are obtained using (1.2) with (1.3) or (1.4) and one of the regularization terms that are considered in the following.

Smoothness of a solution function can be achieved by locally penalizing the deviation from the constant function. In quantitative terms, a function with vanishing gradient magnitude is considered smooth.

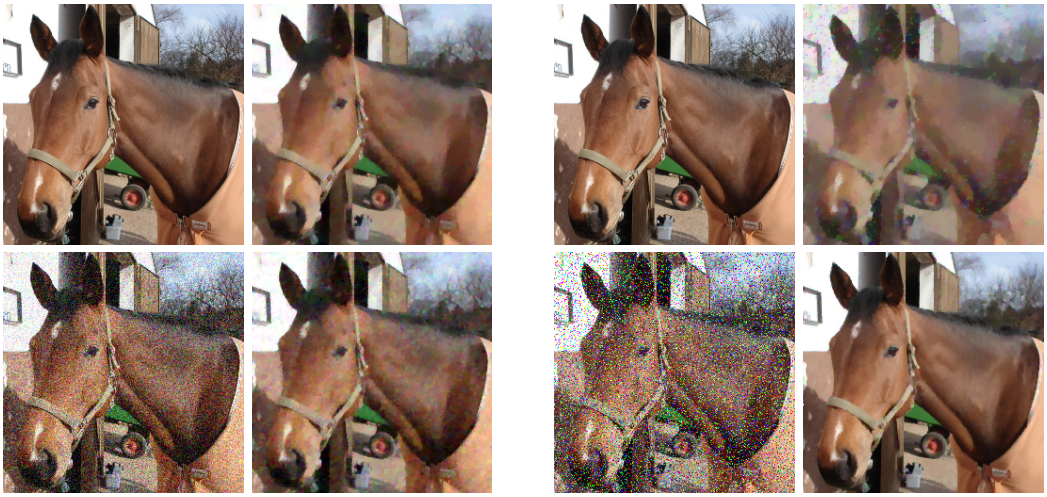


Figure 1.1: Example for image denoising. BOTH BLOCKS, TOP LEFT: Clean image. LEFT BLOCK, BOTTOM LEFT: Noisy image (Gaussian noise with standard deviation $\sigma = 0.35$). RIGHT BLOCK, BOTTOM LEFT: Noisy image (Salt and pepper noise with noise density 0.2). BOTH BLOCKS, TOP RIGHT: Minimizer of the TV-L_2 denoising model (1.3) and (1.6). BOTTOM RIGHT: Minimizer of the TV-L_1 model (1.4) and (1.6).

A solution vector is considered smooth when the discrete gradient magnitude vanishes. Cast as a regularization term for (1.2) the *Tikhonov regularization* [TA77] can be used:

$$R(u) = \|\nabla u\|_2^2 \qquad \left(= \sum_{i=1}^{N} ((\partial_x u)_i)^2 + ((\partial_y u)_i)^2 \right) . \tag{1.5}$$

Therefore, if we look for a minimizer of E in (1.2), the solution is a compromise between the data fidelity and the constant solution. As the square of the gradient magnitude of the function u is penalized, higher deviations from a constant function are assigned a significantly (quadratically growing) higher cost. Therefore, the variation of the resulting function is likely to be small, which means that the result is smooth.

Usually, images have strong edges and we want the solution to respect these edges. In order to achieve this we have to allow the result to have discontinuities, or at least, high deviations from the constant function should not be penalized more than small deviations; It could have been caused by an edge. This leads to the so called *total variation regularization* (TV):

$$R(u) = \mathrm{TV}(u) := \|\nabla u\|_{2,1} \qquad \left(= \sum_{i=1}^{N} \sqrt{((\partial_x u)_i)^2 + ((\partial_y u)_i)^2} \right) . \tag{1.6}$$

It penalizes deviations from the constant solution proportionally. The TV regularizer has many nice properties, which can be verified in the continuous domain[1]. Important properties for computer vision are: TV preserves discontinuities of functions, it measures the boundary length of a set represented by a binary function, and for differentiable functions it is the integral over the gradient magnitude.

Figure 1.2 shows results for the image denoising problem using the regularization terms (1.5) and (1.6) with different weighting parameter λ in (1.2).

[1]For doing so, we would need to introduce weak derivatives, the space of functions with bounded variation, and some more analysis tools. As this is beyond the scope of this thesis, we refer the interested reader for example to [BL11, ABM06, CCC+10, Zie89].

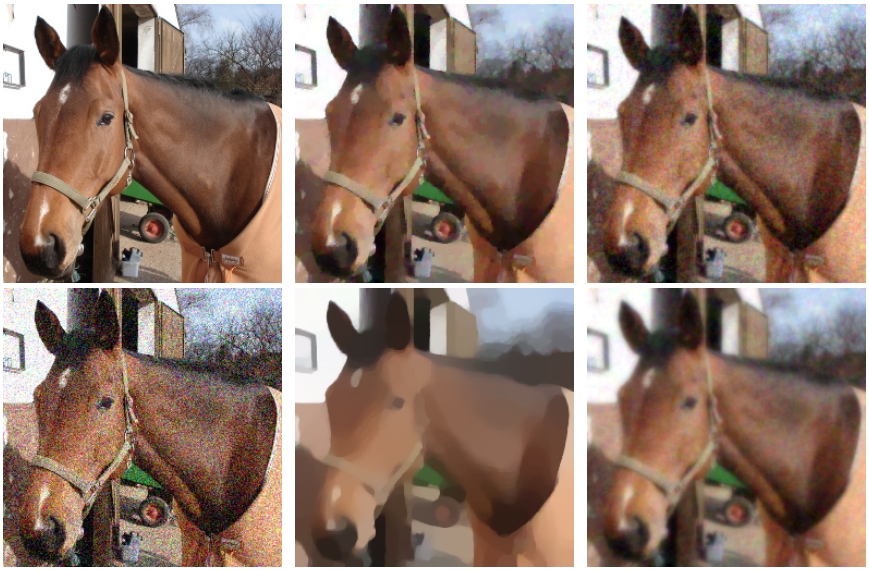


Figure 1.2: Example for image denoising with different regularization terms. LEFT COLUMN: Clean image and noisy image deteriorated by gaussian noise with standard deviation $\sigma = 0.35$. MIDDLE COLUMN: Minimizer using TV regularization. RIGHT COLUMN: Minimizer using Tikhonov regularization. The parameter λ is tuned for a good quality of the upper result.

Another property of TV is convexity, i.e., the connecting line between two points of its graph lies above the graph. Details are given in Chapter 2. For now it is enough to know that convexity is a favorable property, especially, when optimization is considered. For many classes of such problems there are efficient algorithms. Often even worst-case complexity estimates are available. Maybe the most important feature of convex functions is that any local minimizer is necessarily a global minimizer of the function. Their disadvantage is that often there is some loss regarding the accuracy of the model to the real world problem. The much more general class of nonconvex problems, on one hand, allows for better energy models, however on the other hand, comes with several problems for optimization like multiple local minima. While there has been vast progress in convex optimization—today, many nonsmooth convex optimization programs can be solved with comparable efficiency to linear programs—nonconvex optimization is still rarely applied in practice. Indeed, in a SIAM review in 1993, R. Rockafellar pointed out that: "The great watershed in optimization is not between linearity and nonlinearity, but convexity and nonconvexity". Particularly in computer vision, it is known that nonconvex regularization terms usually better model the problem than convex ones. The nonconvexity can be motivated and justified from different viewpoints, including robust statistics [BR96], nonlinear partial differential equations [PM87], and natural image statistics [HM99]. Numerous works demonstrated through experiments [BR96, RB09], that nonconvex potential functions are the right choice. However, due to the issues arising in the optimization they are still rarely used.

The nonconvexity that is mentioned here can be expressed in a formula as

$$R(u) = \sum_{i=1}^{N} \psi(|(\nabla u)_i|)\,, \tag{1.7}$$

where $\psi\colon \mathbb{R}_+ \to \mathbb{R}_+$ is a nondecreasing function like those in Figure 1.3. Obviously, when minimizing (1.2) with (1.7) the preference for a vanishing gradient magnitude is still present, however, larger deviations from this are not penalized proportionally, unlike for TV. The effect is a good compromise between overall smoothness and allowing for discontinuities. This is advantageous as locations with high varia-

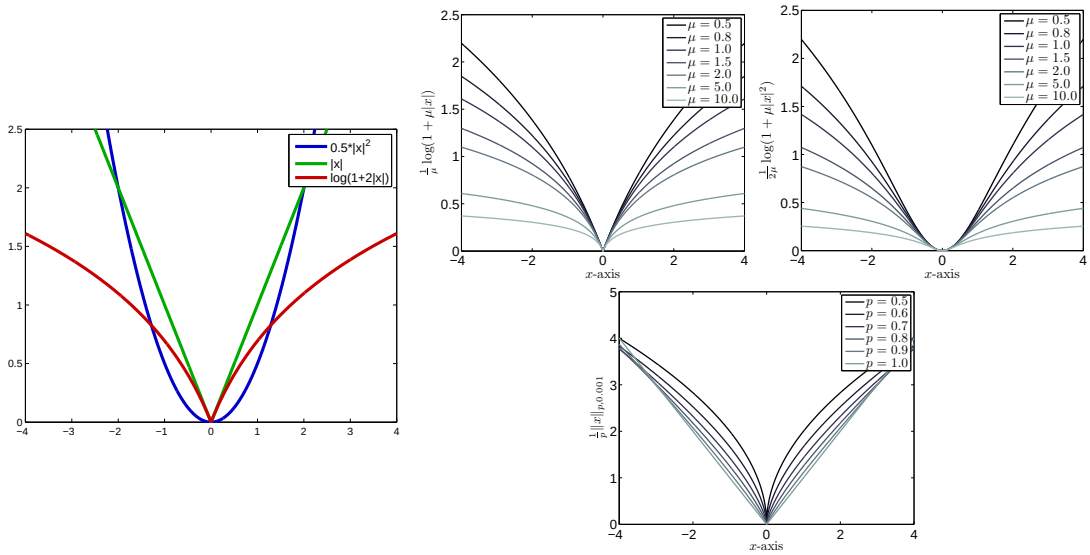


Figure 1.3: Examples for functions that are used to penalize the gradient magnitude in many regularization tasks in computer vision as in (1.7). For convex functions ψ (green and blue) the convex regularizers (1.5) and (1.6) are recovered. However, for computer vision problems there is a strong motivation to use nonconvex functions like $\log(1 + 2|x|)$ on the left or those on the right.

tions in the gradient magnitude are usually associated with edges, important image characteristics. Part I of this thesis aims on bridging the gap between modeling computer vision problems with nonconvex penalty terms and their efficient optimization.

Chapter 2

Basics of convex analysis and optimization

In general *optimization problems are unsolvable*. That is also the starting message for an introductory lecture on optimization by Yurii Nesterov [Nes04]. There is no optimization algorithm that can solve all problems. Although there are commercial optimization packages that have a wide range of applicability, many optimization problems are not solved satisfactorily. Moreover, there is always the question whether we can trust the solution; Did the algorithm really solve the problem that we wanted it to solve? Sometimes, for practical problems the solution can be validated by a plausibility check. However, if we seek for efficiency and guarantees, we should know our problem well and be aware about the mechanisms of optimization.

The foundation for this consciousness is built by the theory. Already during the time when the problem is modeled, strategies for solving it must be taken into account. Perfectly modeling the problem, but not being able to solve it, is not better than a simple model that can be solved. Important modeling aspects are well-posedness, the class of functions that is used, the solution space, and what solution can be expected. The better the structure of the problem is known, the more properties can be used during the optimization and the bigger the chance to solve the problem efficiently and accurate. Convexity of the objective function is such a property. In that case efficient optimization algorithms exist. Nevertheless, in order to decide whether a function is convex, we need to know what convexity means.

If we want to use the newest and fastest algorithms, to determine the expected efficiency, or to develop new and more efficient optimization algorithms, a strong knowledge about the underlying theory is important. In this chapter, we focus on the introduction of the basic tools from convex analysis. After Section 2.1 we are equipped with the tools to consider some algorithms that play an important role in solving convex optimization problems. They reveal several similarities that can only be understood with some theory in the background. In the next chapter, using the knowledge gained in this chapter we develop a new and efficient algorithm for a certain class of convex optimization problems.

In all what follows, we work in the Euclidean vector space $\mathbb{R}^N$ of dimension $N \in \mathbb{N}$ equipped with the standard inner product $\langle x, y\rangle := \sum_{i=1}^{N} x_i y_i$ and the Euclidean norm $\|x\| := \sqrt{\langle x, x\rangle}$ whenever $x, y \in \mathbb{R}^N$.

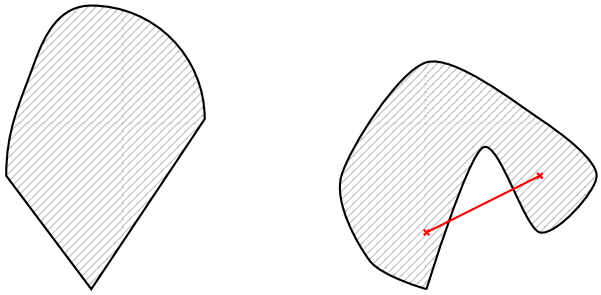


Figure 2.1: LEFT: Convex set. RIGHT: Nonconvex set. A set is convex if and only if the connecting line between each pair of points is completely contained in the set.

2.1 Convex analysis

We introduce some basic facts from convex analysis. Results are selected from Rockafellar [Roc70, RW98]; Bredies and Lorenz [BL11][1]; Bauschke and Combettes [BC11][1] and Nesterov [Nes04]. Most proofs and examples are taken from these references. Sometimes the proofs are slightly modified in order to better suit our needs. For readers who are familiar with convex analysis, this section serves as a reference for the notation that is used later on. The reader, who just wants to get an overview may skip the proofs.

2.1.1 Basics of convex sets

In order to decide whether an optimization problem can be solved with tools from convex analysis or not, the question about convexity must be answered. As the minimization of the objective function can be restricted to a certain set, convexity must be verified for the objective function and the constraints.

We start the section about convex analysis by considering the notion of convex sets. Considering convex sets is crucial for convex functions, as there is a strong relation between them. A function is convex if the set of points lying above the graph of the function is convex. Therefore, we first define convexity of sets.

Definition 2.1 (convex set)**.** *A subset $C \subset \mathbb{R}^N$ is said to be convex if $(1-\lambda)x + \lambda y \in C$ whenever $x, y \in C$ and $0 < \lambda < 1$.*

Figure 2.1 shows a discriminative example for a convex and a nonconvex set. In order to determine whether a set is convex or not, it is helpful to know how convex sets can be generated. For the construction of convex sets, the following theorem is of interest. Its proof is elementary.

Theorem 2.2 (intersection of convex sets)**.** *The intersection of an arbitrary collection of convex sets is convex, i.e., for an arbitrary index set I and convex sets $C_i \subset \mathbb{R}^N$ it holds that $\bigcap_{i\in I} C_i$ is convex.*

Proof. Let $x, y \in \bigcap_{i\in I} C_i$ lie in the intersection of all convex sets $C_i \subset \mathbb{R}^N$. By definition of convexity $z = (1-\lambda)x + \lambda y \in C_i$ for all sets C_i, thus $z \in \bigcap_{i\in I} C_i$. □

Examples for convex sets are given by half-spaces.

[1] This book considers infinite dimensional spaces, whereas we work in finite dimensions. Results from this reference are reduced to the finite dimensional setting here.

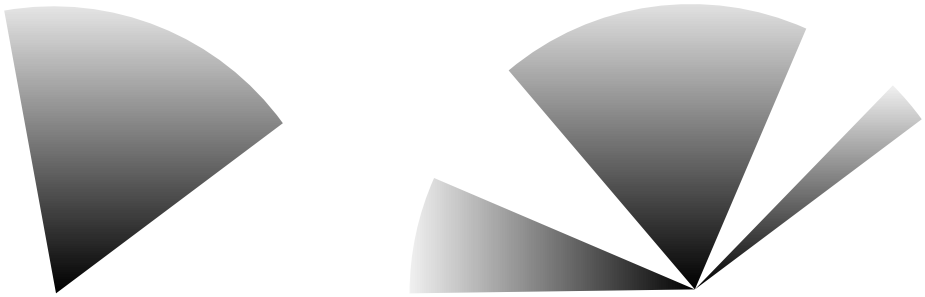


Figure 2.2: LEFT: Convex conve. RIGHT: Nonconvex cone. A cone contains for each point the ray emanating from the origin that goes through that point.

Definition 2.3 (half-space)**.** *For any non-zero $0 \neq b \in \mathbb{R}^N$ and any $\beta \in \mathbb{R}$ the set*

$$\{x \in \mathbb{R}^N |\ \langle x, b\rangle \leq \beta\} \text{ is a closed}, \qquad \text{and} \qquad \{x \in \mathbb{R}^N |\ \langle x, b\rangle < \beta\} \text{ is an open}$$

half-space.

Using the preceding theorem half-spaces may be used to construct other convex sets like a ball.

Example 2.1. The (open) unit ball $B_r(0) := \{x \in \mathbb{R}^N | \, \|x\| < r\} \subset \mathbb{R}^N$ with radius $r > 0$ is convex by Theorem 2.2 and due to

$$B_r(0) = \bigcap_{b \in \mathbb{R}^N, \|b\|=1} \{x \in \mathbb{R}^N |\ \langle x, b\rangle < r\}\,.$$

Analogously the closed unit ball $\overline{B}_r(0)$ can be constructed with closed half-spaces.

Another source of examples for convex sets is the convex cone. The general concept of cones is important for many considerations—not only in convex analysis. A cone is the union of half-lines emanating from the origin. See Figure 2.2 for an example of a convex and a nonconvex cone.

Definition 2.4 (cone, convex cone)**.** *A subset K of $\mathbb{R}^N$ is called a cone if it is closed under nonnegative scalar multiplication, i.e., $\lambda x \in K$ when $x \in K$ and $\lambda \geq 0$. The cone is a convex cone when K is a convex set.*

Theorem 2.5. *A subset of $\mathbb{R}^N$ is a convex cone if and only if it is closed under addition and nonnegative scalar multiplication.*

Proof. "⇒": Let $K \subset \mathbb{R}^N$ be a convex cone and let $x, y \in K$. By definition K is closed under nonnegative scalar multiplication. The convex combination $z = (x + y)/2$ lies in K and hence $x + y = 2z \in K$. Thus, it is also closed under addition. "⇐": If K is closed under nonnegative scalar multiplication, then K is a cone. Therefore, for $x, y \in K$ also $(1 - \lambda)x, \lambda y$ belong to K. Closedness under addition implies $(1 - \lambda)x + \lambda y \in K$, which verifies the convexity of K. □

2.1.2 Basics of convex functions

In order to work with convex functions, it is convenient to allow functions to take values on the extended real line $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. The conventions for handling ∞ are as follows. For all $\alpha \in \overline{\mathbb{R}}$ it holds that

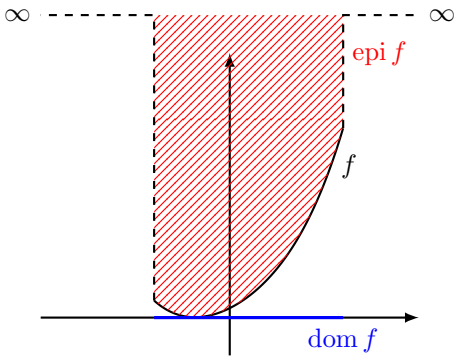


Figure 2.3: A proper convex function with marked epigraph and effective domain.

$\alpha \leq \infty$ and $\alpha < \infty$ holds if and only if $\alpha \in \mathbb{R}$. Formally, we define

$$\alpha + \infty = \infty\,, \quad \text{for all } \alpha \in \overline{\mathbb{R}}\,, \qquad \alpha \cdot \infty = \infty\,, \quad \text{for all } \alpha > 0\,, \qquad 0 \cdot \infty = 0\,.$$

Other operations like subtraction and multiplication with negative numbers are not defined. The concept of the extended real line allows us to treat convex functions defined on $\mathbb{R}^N$ and functions defined on a subset $C \subset \mathbb{R}^N$ in an unified way. However, let us first define a convex function. As mentioned earlier, we make use of a relation between a function and a suitable set associated with the function. The set under consideration is the epigraph.

Definition 2.6 (epigraph)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a function taking values on the extended real line. We denote by*

$$\operatorname{epi} f := \{(x, \alpha) \in \mathbb{R}^{N+1} \,|\, \alpha \geq f(x)\}$$

the epigraph of the function f. (Note that for $(x, \alpha) \in \operatorname{epi} f$ it holds that $\alpha < \infty$.)

Definition 2.7 (convex function)**.** *A function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is convex (on $\mathbb{R}^N$) if $\operatorname{epi} f$ is convex as a subset of $\mathbb{R}^{N+1}$.*

We will only work with convex functions defined on the whole space $\mathbb{R}^N$. The reason is that any convex function on a convex subset S of $\mathbb{R}^N$ can be extended to a convex function on the whole space by setting $f(x) = \infty$ for all $x \in \mathbb{R}^N \setminus S$. Statements that are true only on the subset where a function takes on finite values make use of the following definition.

Definition 2.8 (effective domain)**.** *For a function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ the set*

$$\operatorname{dom} f := \{x \in \mathbb{R}^N \,|\, f(x) < \infty\}$$

is called the (effective) domain of the function f.

As a function that takes the value ∞ everywhere usually requires additional technical considerations, it is often excluded from statements.

Definition 2.9 (proper function)**.** *A function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is called proper if $\operatorname{dom} f \neq \emptyset$.*

Figure 2.3 visualizes the epigraph and the effective domain of a proper convex function.

Example 2.2. (i) Consider the function $f\colon \mathbb{R} \to \overline{\mathbb{R}}$ with $f(x) = 0$ if $x \in [-1,1]$ and $f(x) = \infty$ otherwise. The function is convex since $\operatorname{epi} f = [-1,1] \times \mathbb{R}$ and the intersection of half-spaces

$$[-1,1] \times \mathbb{R} = \{x \in \mathbb{R}^2 |\, x_2 \geq 0\} \cap \{x \in \mathbb{R}^2 |\, x_1 \leq 1\} \cap \{x \in \mathbb{R}^2 |\, x_1 \geq -1\}\,, \quad x = (x_1, x_2)^\top\,,$$

is convex.

(ii) Let the function $f\colon \mathbb{R}^N \to \mathbb{R}$ be given as $f(x) = \|x\|$. The function is convex since the epigraph is a convex cone. Let $x, y \in \mathbb{R}^N$, $\alpha_x, \alpha_y \in \mathbb{R}$, and $\lambda \geq 0$. If $(x, \alpha_x) \in \operatorname{epi} f$ then $\alpha_x \geq \|x\|$ and $\lambda(x, \alpha_x) = (\lambda x, \lambda\alpha_x)$ with $\lambda\alpha_x \geq \lambda\|x\| = \|\lambda x\|$, thus $\operatorname{epi} f$ is a cone. Furthermore, the cone is closed under addition. For $(y, \alpha_y) \in \operatorname{epi} f$ it holds that $(x, \alpha_x) + (y, \alpha_y) = (x + y, \alpha_x + \alpha_y)$ and $\alpha_x + \alpha_y \geq \|x\| + \|y\| \geq \|x + y\|$ by the triangle inequality. Theorem 2.5 implies the convexity of $\operatorname{epi} f$, hence of f.

As we have seen in Example 2.2, it is cumbersome to verify convexity of a function. A first step towards simplifying this task is made by the next theorem. The analogy between a convex function and a convex set gives rise to more examples of convex functions. Theorem 2.2 tells us that the intersection of an arbitrary collection of epigraphs (of convex functions) yields a convex set. This set is also an epigraph, namely the epigraph of the function constructed as the pointwise supremum of the collection of convex functions.

Theorem 2.10 (convexity of the pointwise supremum)**.** *The pointwise supremum of an arbitrary collection of convex functions is convex.*

Proof. By Theorem 2.2 the intersection of an arbitrary collection of convex sets is a convex set. If a function is given as the pointwise supremum of convex functions, then its epigraph is the intersection of the epigraphs of those functions. □

Example 2.3. (i) Consider the collection of functions $g_x\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $g_x(p) = \langle x, p\rangle - \|x\|$ where $x \in \mathbb{R}^N$. The pointwise supremum $G\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $G(p) = \sup_{x\in\mathbb{R}^N} g_x(p)$, which is given by $G(p) = \infty$ for $\|p\| > 1$ and $G(p) = 0$ for $\|p\| \leq 1$, is convex.

(ii) Now, we consider for all $p \in \mathbb{R}^N$ with $\|p\| \leq 1$ the function $f_p\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $f_p(x) = \langle x, p\rangle - G(p)$ with G as defined in (i). Then, the pointwise supremum $F\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $F(x) = \sup_{\|p\|\leq 1} f_p(x)$ (for all $x \neq 0$ it is taken at $p = x/\|x\|$ and for $x = 0$ it is taken at any $p \in \overline{B}_1(0)$), which is given by $F(x) = \|x\|$, is convex.

Next, we introduce some results that make the verification of convexity much easier. The following inequality characterizes convex functions without the need to work with the epigraph explicitly. It serves as an useful tool at many places in convex analysis.

Theorem 2.11 (Jensen's inequality)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$. Then f is convex if and only if*

$$f(\lambda_1 x_1 + \ldots + \lambda_m x_m) \leq \lambda_1 f(x_1) + \ldots + \lambda_m f(x_m)$$

whenever all $\lambda_i \geq 0$ and $\lambda_1 + \ldots + \lambda_m = 1$.

Proof. We omit the proof as it is technical and can be found in nearly all text books covering the topic convex analysis. (See, for example, [Roc70, Thm. 4.3].) □

From Jensen's inequality we can immediately derive a theorem for the convexity of functions that are the sum of convex functions. From the opposite point of view the following theorem provides a way to often reduce the question about convexity to simpler terms. The purpose of the subsequent theorem is the same.

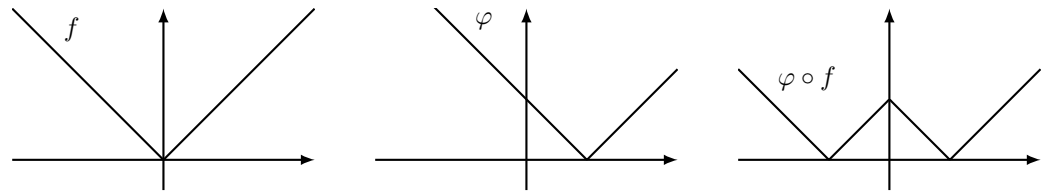


Figure 2.4: Plot of the function considered in Example 2.4.

Theorem 2.12 (convexity of linear combinations)**.** *If f_1 and f_2 are proper convex functions on $\mathbb{R}^N$ and $\lambda_1, \lambda_2 \geq 0$, then $\lambda_1 f_1 + \lambda_2 f_2$ is convex.*

Proof. We verify the statement using Theorem 2.11. For $x, y \in \mathbb{R}^N$, $\lambda \in (0,1)$ it is

$$\begin{aligned}\lambda_1 f_1((1-\lambda)x + \lambda y) + \lambda_2 f_2((1-\lambda)x + \lambda y) \\ \leq \lambda_1\left((1-\lambda)f_1(x) + \lambda f_1(y)\right) + \lambda_2\left((1-\lambda)f_2(x) + \lambda f_2(y)\right) \\ = (1-\lambda)(\lambda_1 f_1(x) + \lambda_2 f_2(x)) + \lambda(\lambda_1 f_1(y) + \lambda_2 f_2(y))\,.\end{aligned}$$

□

Theorem 2.13 (convexity of compoisitions)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a convex function, and let $\varphi\colon \mathbb{R} \to \overline{\mathbb{R}}$ be a nondecreasing, convex function. Then $\varphi \circ f$ is a convex function on $\mathbb{R}^N$ (set $\varphi(\infty) = \infty$).*

Proof. The convexity of f gives (Theorem 2.11) for $x, y \in \mathbb{R}^N$ and $\lambda \in (0,1)$

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)\,.$$

Now, we apply φ to both sides. Since φ is nondecreasing the order of the inequality is preserved. Again using Theorem 2.11 concludes the proof. □

Example 2.4. A simple example shows the necessity of φ being nondecreasing in Theorem 2.13. Let $f(x) = |x|$ and $\varphi(x) = |x-1|$. Then $h := \varphi \circ f$ is not convex because h is symmetric $h(-x) = h(x)$ and $h(\pm 1) = 0 < 1 = h(0)$. (See Figure 2.4.)

There exists a simple characterization of convexity for differentiable functions. First, we present it for the one dimensional case and then for higher dimensions. The proof for the latter is by restricting the multi-dimensional function to line segments and applying the result from the one dimensional case. Actually, convexity can be seen as a one dimensional phenomenon. A function is convex if and only if its restriction to any one dimensional line segment is convex. First, we need a lemma, which is interesting in its own right. The situation is shown in Figure 2.5.

Lemma 2.14 (slope inequality)**.** *Let $f\colon (a,b) \to \mathbb{R}$ be a real-valued function on the open interval $(a,b) \subset \mathbb{R}$. Then f is convex if and only if for all $x_0 < y < x_1$ in C it holds that*

$$\frac{f(y) - f(x_0)}{y - x_0} \leq \frac{f(x_1) - f(x_0)}{x_1 - x_0} \leq \frac{f(x_1) - f(y)}{x_1 - y}\,.$$

Moreover, for an $x \in (a,b)$ the difference quotient $\Delta_x(y) := (f(y) - f(x))/(y - x)$ is nondecreasing for $y \in (a,b) \setminus \{x\}$.

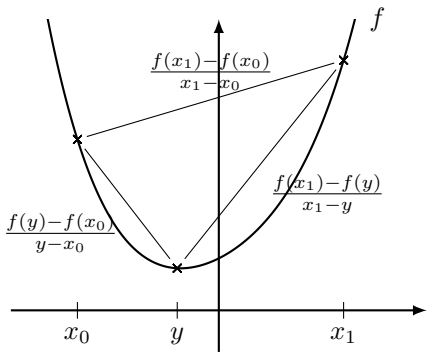


Figure 2.5: Plot of the situation in Lemma 2.14.

Proof. Thanks to Theorem 2.11 with $m = 2$ convexity is easily seen to be equivalent to

$$f(y) \leq \frac{x_1 - y}{x_1 - x_0} f(x_0) + \frac{y - x_0}{x_1 - x_0} f(x_1), \quad \text{when } x_0 < y < x_1 \text{ in } (a, b)\,.$$

Subtracting $f(x_0)$ or $f(x_1)$ from both sides yields the first or the second inequality, respectively. Nondecreasingness of $\Delta_x(y)$ is obvious by the inequalities just verified. □

Theorem 2.15 (monotonicity and convexity in one dimension)**.** *Let $f\colon (a, b) \to \mathbb{R}$ be a real-valued differentiable function on the open interval $(a, b) \subset \mathbb{R}$. Then the following conditions are equivalent on (a, b):*

(i) f is convex,

(ii) f' is nondecreasing,

(iii) $f(x) \geq f(y) + f'(y)(x - y)$ for all x and y in (a, b).

Proof. Convexity implies by the preceding Lemma 2.14 (nondecreasingness of $\Delta_x(y)$) for $x_0 < x_1$ in (a, b)

$$f'(x_0) \leq \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f(x_0) - f(x_1)}{x_0 - x_1} \leq f'(x_1)\,,$$

which is (ii). Now, assume that (ii) holds. The convex function $g_y(x) := f(x) - f(y) - f'(y)(x - y)$ satisfies $g_y'(x) \leq 0$ for $x < y$ in (a, b) and $g_y'(x) \geq 0$ for $x < y$ in (a, b), which implies that g_y attains its global minimum 0 at $y \in (a, b)$, hence (iii) holds. It remains to show that (iii) implies (i). Suppose (iii) holds and consider the affine functions $l_y(x) := f(y) + f'(y)(x - y)$ for $y \in (a, b)$. Then $f(x) = \sup_{y\in(a,b)} l_y(x)$ is a pointwise supremum over all $y \in (a, b)$ of convex functions and thus itself is convex. □

Corollary 2.16 (second derivative test in one dimension)**.** *Let $f\colon (a, b) \to \mathbb{R}$ be a twice differentiable function on the open interval $(a, b) \subset \mathbb{R}$. Then f is convex if and only if its second derivative is nonnegative on (a, b).*

Proof. Theorem 2.15 together with the equivalence of f'' being nonnegative on (a, b) and f' being nondecreasing on (a, b) proves the statement. □

Opposed to Example 2.2, where it was "hard work" to verify convexity for a relatively simple function, the criteria that we have at hand now provide more convenient tools. However they can only be applied to sufficiently smooth one dimensional functions. The generalization to multi dimensional functions immediately follows after the next example. For the generalization to nonsmooth functions we ask for the readers patience till the next subsection. If Theorems 2.12, 2.13 and 2.17 and Corollary 2.18 are used together, convexity is often a trivial question; For functions that are composite of differentiable parts sometimes some simple calculations are required.

Example 2.5. Convexity of the following examples can be verified using Theorem 2.15 and Corollary 2.16.

(i) $f\colon \mathbb{R} \to \mathbb{R}$, $f(x) = \exp(\lambda x)$ with $\lambda \in \mathbb{R}$.

(ii) $f\colon \mathbb{R} \to \overline{\mathbb{R}}$, $f(x) = -\sqrt{1-x^2}$ for $|x| < 1$ and $f(x) = \infty$ for $|x| \geq 1$.

(iii) $f\colon \mathbb{R} \to \mathbb{R}$, $f(x) = |x|^p$ with $p > 1$.

(iv) $f\colon \mathbb{R} \to \mathbb{R}$, $f(x) = \frac{1}{2}qx^2 + ax + \lambda$ with $\lambda \in \mathbb{R}$, $a \in \mathbb{R}$ and a positive constant $q \in \mathbb{R}_+$.

The following theorem generalizes Theorem 2.15 to functions with domain in $\mathbb{R}^N$.

Theorem 2.17 (monotonicity and convexity)**.** *Let $f\colon C \to \mathbb{R}$ be a real-valued differentiable function on the open set $C \subset \mathbb{R}^N$. Then the following conditions are equivalent on C:*

(i) f is convex,

(ii) $\langle \nabla f(x) - \nabla f(y), x - y\rangle \geq 0$ for all x and y in C,

(iii) $f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle$ for all x and y in C.

Proof. The proof is simple. Consider the restriction of f onto line segments and use Theorem 2.15 for the arising one dimensional function. Therefore, we omit the details here. □

Corollary 2.18 (second derivative test)**.** *Let $f\colon C \to \mathbb{R}$ be a twice differentiable function on an open convex set $C \subset \mathbb{R}^N$. Then f is convex on C if and only if its Hessian matrix*

$$H_f(x) := \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x_1, \ldots, x_N)\right)_{1 \leq i,j \leq N}, \quad x = (x_1, \ldots, x_N)^\top.$$

is positive semi-definite for every $x \in C$.

Proof. The proof works in parallel fashion to Theorem 2.17. □

Now, we are well equipped to consider a few examples of convex functions.

Example 2.6. We start with examples that can be verified as being convex using the second order derivative criterion.

(i) $f\colon \mathbb{R}^N \to \mathbb{R}$, $f(x) = \|x\|_p^p$ with $p > 1$.

(ii) $f\colon \mathbb{R}^N \to \mathbb{R}$, $f(x) = \frac{1}{2}\langle x, Qx\rangle + \langle x, a\rangle + \lambda$ with $\lambda \in \mathbb{R}$, $a \in \mathbb{R}^N$ and a positive semi-definite matrix $Q \in \mathbb{R}^{N \times N}$.

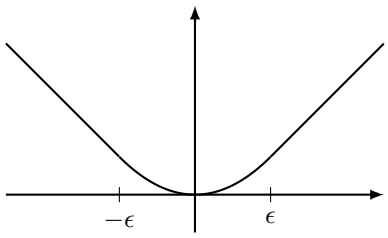


Figure 2.6: Plot of the Huber function considered in Example 2.7.

Example 2.7. An example where the second order criterion cannot be used, but Theorem 2.17 can be used is the following. Consider $f\colon \mathbb{R}^N \to \mathbb{R}$, $f(x) = \frac{1}{2\varepsilon}\|x\|^2$ for $\|x\| < \varepsilon$ and $f(x) = \|x\| - \frac{\varepsilon}{2}$ for $\|x\| \geq \varepsilon$ with $\varepsilon > 0$. This is the so-called *Huber* function (or sometimes Huber-norm). (See Figure 2.6.)

Indicator functions of convex sets are particularly interesting for optimization. They are used to treat unconstrained and constrained convex optimization problems in an unified framework. The convex constraint set can be associated with an indicator function. In order to avoid solutions outside the constraint set, indicator functions are defined different to other branches of analysis by assigning ∞ to unfeasible solutions.

Definition 2.19 (indicator function)**.** *The indicator function associated with a convex set $C \subset \mathbb{R}^N$ is defined by*

$$\delta_C(x) := \begin{cases} 0, & \text{if } x \in C\,, \\ \infty, & \text{if } x \notin C\,. \end{cases}$$

Example 2.8. In Example 2.3, we have already seen an example for an indicator function. The function $G(p)$ in Example 2.3(i) is the indicator function of the unit ball $\overline{B}_1(0)$ at the origin.

In the remainder of this subsection, we consider some topological aspects of a convex function. The next theorem proves that convexity is strongly related to continuity of a function.

Definition 2.20 (interior, closure)**.** *For any set $C \subset \mathbb{R}^N$ the interior* $\operatorname{int} C := \{x \in \mathbb{R}^N | \, \exists \varepsilon > 0 \colon x + \varepsilon\overline{B}_1 \subset C\}$ *and the closure* $\operatorname{cl} C := \bigcap\{C + \varepsilon\overline{B}_1 | \, \varepsilon > 0\}$ *are defined.*

Definition 2.21 (lower semi-continuity)**.** *A proper function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is said to be lower semi-continuous (lsc) at a point $\bar{x} \in \operatorname{dom} f$ if $f(\bar{x}) \leq \liminf_{x\to\bar{x}} f(x)$.*

Of course, there exists the analogous concept of upper semi-continuity, which requires for $\bar{x} \in \operatorname{dom} f$ that $f(\bar{x}) \geq \limsup_{x\to\bar{x}} f(x)$. A function f being upper and lower semi-continuous at a point $\bar{x} \in \operatorname{dom} f$ is continuous at point $\bar{x}$.

Definition 2.22 (relative continuity)**.** *A function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is continuous relative to a subset $S \subset \operatorname{dom} f$ if the restriction of f to S is a continuous function.*

Theorem 2.23 (continuity of convex functions)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper lower semi-continuous convex function and let C be an open convex subset of $\operatorname{dom} f$. Then f is continuous relative to C.*

Proof. Without loss of generality we may assume that $C = \operatorname{dom} f$. Otherwise we could replace f by a function g with $\operatorname{dom} g = C$. In order to prove continuity of f we show that the level sets $U_\alpha := \{x | \, f(x) \geq \alpha\}$, $\alpha \in \mathbb{R}$, are closed, and prove that this implies upper semi-continuity of f and thus

continuity of f. The complement $U_\alpha^\mathsf{c} = \{x|\, f(x) < \alpha\}$ of U_α is open for all $\alpha \in \mathbb{R}$, since it arises as projection on $\mathbb{R}^N$ of the intersection of the open set $\{(x,\beta)|\, \beta > f(x)\}$ with the open half-space $\{(x,\beta)|\, \beta < \alpha\}$ in $\mathbb{R}^{N+1}$. Thus U_α is closed. Now, we verify the implication of upper semi-continuity of f. Let $(x^\nu)_{\nu\in\mathbb{N}}$ be sequence with $\lim_{\nu\to\infty} x^\nu = \bar{x}$ and $\lim_{\nu\to\infty} f(x^\nu) =: \beta$. For all $\alpha < \beta$ there exists $\nu_\alpha \in \mathbb{N}$ such that $f(x^\nu) > \alpha$ for all $\nu \geq \nu_\alpha$, i.e., $x^\nu \in U_\alpha$ for all $\nu \geq \nu_\alpha$. As U_α is closed, it holds that $\bar{x} \in U_\alpha$ and, moreover, $\bar{x} \in \bigcap_{\alpha<\beta} U_\alpha = U_\beta$. Therefore, $f(\bar{x}) \geq \beta$ and hence f is upper semi-continuous. □

As a particular example to which the preceding theorem can be applied is when the function f is defined on an open set, i.e. $\operatorname{dom} f = C$ for an open set C. Moreover, if $\operatorname{dom} f = \mathbb{R}^N$ then any convex function is necessarily continuous on $\mathbb{R}^N$.

Remark 2.9. In Theorem 2.23 the requirement of lower semi-continuity is not necessary. Rockafellar [RW98, Theorem 7.4] shows that any proper convex function on $\mathbb{R}^N$ is lower semi-continuous except perhaps at relative boundary points of the domain of the function. For N-dimensional domain $\operatorname{dom} f$ the relative boundary coincides with $(\operatorname{cl}\operatorname{dom} f) \setminus (\operatorname{int}\operatorname{dom} f)$.

The result about continuity can be sharpened on compact subsets of the interior of the domain of a function (Theorem 2.25). Lipschitz continuity can be proved in this setting. Lipschitz continuity is a stronger notion of continuity in the sense that the change of function values is limited.

Definition 2.24 (Lipschitz continuous)**.** *A real-valued function $f\colon S \to \mathbb{R}^M$, $M \in \mathbb{N}$, with $S \subset \mathbb{R}^N$ is called Lipschitzian (or Lipschitz continuous) relative to the set S if there exists a real number $L \geq 0$ such that*

$$\|f(y) - f(x)\| \leq L\|y - x\|\,, \quad \forall x, y \in S\,.$$

Lipschitz continuity is key for many optimization algorithms. We meet this concept again in Section 2.1.8. There, we will see that Lipschitz continuity (of the gradient of a function) allows for majorizing quadratic functions, i.e., there is a quadratic function whose epigraph is contained in that of the majorized function and the functions meet in at least one point. Many optimization methods (e.g. the Gradient method (Section 2.2.1), some splitting methods (Section 2.2.3), the Fista method (Section 2.2.5), and many others) then focus on minimizing the original function via minimizing such quadratic majorizers.

The Lipschitz continuity that is dicussed in the following theorem is based on the function values (not on the gradient of the function), i.e., $M = 1$ in Definition 2.24.

Theorem 2.25. *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper convex function and let $S \subset \operatorname{int}\operatorname{dom} f$ be a compact subset. Then f is Lipschitzian relative to S.*

Proof. For each $\varepsilon > 0$ the map $(x,u) \mapsto x + \varepsilon u$ is continuous, thus, for each $\varepsilon > 0$ the set $S + \varepsilon\overline{B}_1$ is compact. The intersection over all $\varepsilon > 0$ of the sets $(S + \varepsilon\overline{B}_1) \cap (\mathbb{R}^N \setminus \operatorname{int}\operatorname{dom} f)$ is empty, hence one of these sets must be empty. Therefore, there exists $\varepsilon > 0$ such that $S + \varepsilon\overline{B}_1 \subset \operatorname{int}\operatorname{dom} f$. By Theorem 2.23 f is continuous on $S + \varepsilon\overline{B}_1$. As $S + \varepsilon\overline{B}_1$ is closed and bounded, there exists $\alpha^+, \alpha^- \in \mathbb{R}$ such that $\alpha^- \leq f(x) \leq \alpha^+$ for all $x \in S + \varepsilon\overline{B}_1$.

Now, let $x, y \in S$ be two distinct points and let $z \in S + \varepsilon\overline{B}_1$ be an extrapolation of the line segment from x to y, i.e., let $z = y + \varepsilon(y - x)/\|y - x\| \in S + \varepsilon\overline{B}_1$. It holds that $y = (1 - \lambda)x + \lambda z$ with $\lambda = \|y - x\|/(\varepsilon + \|y - x\|)$ and by convexity of f we have $f(y) \leq f(x) + \lambda(f(z) - f(x))$. This implies $f(y) - f(x) \leq \lambda(\alpha^+ - \alpha^-) \leq (\alpha^+ - \alpha^-)/\varepsilon \cdot \|y - x\|$ for all $x, y \in S$, which concludes the Lipschitz continuity of f relative to S. □

Remark 2.10. In fact, using the same strategy as in the proof of Theorem 2.25, we can show that a proper convex function is Lipschitz continuous relative to any open bounded subset of its domain.

2.1.3 Attainment of local optima

In the next subsection, we face problems of minimizing a function, i.e. find $x \in \mathbb{R}^N$ that minimizes a function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$. The immediate question that arises when considering this problem is the existence of such a solution. This issue was mentioned earlier in (1.1). Note that this subsection is not specific to convex functions. The optimal value of a function is given by $\inf_{x\in\mathbb{R}^N} f(x)$. The question about existence is that of whether $\inf$ can be replaced by $\min$. We define:

Definition 2.26 (minimizers of a minimization problem)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a function. The set of optimal solutions to a minimization problem $\inf_{x\in\mathbb{R}^N} f$ is defined as:*

$$\arg\min_{x\in\mathbb{R}^N} f(x) := \{\bar{x} \in \mathbb{R}^N |\, f(\bar{x}) = \inf_{x\in\mathbb{R}^N} f(x) \text{ and } f(\bar{x}) < \infty\}\,.$$

Fortunately, existence of minimizers can be guaranteed under mild assumptions. A basic result is formulated in Theorem 2.28 ([RW98, Thm 1.9]). It requires another definition.

Definition 2.27 (coercivity)**.** *A function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is called coercive, if $\|x\| \to \infty$ implies $f(x) \to \infty$.*

This property is often considered in the context of existence of minimizers, however, sometimes it appears with different names. For example in [RW98] it is expressed via *level boundedness*. A function f is said to be level-bounded, if for every $\alpha \in \mathbb{R}$ the set $\mathrm{lev}_\alpha := \{x \in \mathbb{R}^N |\, f(x) \leq \alpha\}$ is bounded (possibly empty).

Theorem 2.28 (attainment of minimizers)**.** *Suppose $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is lsc, coercive (or level-bounded), and proper. Then the value $\inf_{x\in\mathbb{R}^N} f$ is finite and the set $\arg\min_{x\in\mathbb{R}^N} f(x)$ is nonempty and compact.*

Proof. As f is proper $\bar{\alpha} := \inf f$ is finite. For any $\alpha \in (\bar{\alpha}, \infty)$ the set lev_α is nonempty and bounded. Thanks to the lower semi-continuity, it is also closed ([RW98, Thm.1 1.6], cf. proof of Theorem 2.23), hence, compact. Obviously, it holds that $\mathrm{lev}_\beta \subset \mathrm{lev}_\alpha$, whenever $\beta < \alpha$. Moreover, as arbitrary intersections of nonempty and closed sets are nonempty and closed, $\bigcap_{\alpha>\bar{\alpha}} \mathrm{lev}_\alpha$ is nonempty and closed. Therefore, $\bigcap_{\alpha>\bar{\alpha}} \mathrm{lev}_\alpha = \mathrm{lev}_{\bar{\alpha}} = \arg\min f$ is compact and nonempty. As $f(x) > -\infty$ for all $x \in \mathbb{R}^N$, it follows that $\bar{\alpha} > -\infty$. □

2.1.4 Subgradient and subdifferential

The subgradient of a convex function is a generalization of the gradient of a differentiable convex function. The generalization of gradients to subgradients can be motivated geometrically. The gradient is associated with the locally best linear approximation of a differentiable function. This geometric consideration is not limited to differentiable functions. Linearly approximating a nondifferentiable convex function is also possible. However, we have to dispense with a single best approximation. In nondifferentiable points there exist many linear approximations that are tangent to the graph of the function. Therefore, the subgradient of a convex function is defined as a set-valued mapping. See Figure 2.7 for an example. The details of this figures become clear with Definition 2.31.

As a generalization of the well-known concept of differentiability—Proposition 2.32 verifies that—, we expect some useful properties for optimization. Indeed Fermat's rule, which relates stationary points of a differentiable function with the zeros of its derivative, is generalized to not necessarily differentiable convex functions in Theorem 2.34. Moreover, the same theorem shows that convexity sharpens this result, in the sense that a stationary point is always a (global) minimum. The remainder of this section is devoted to the simplification of characterizing subgradients.

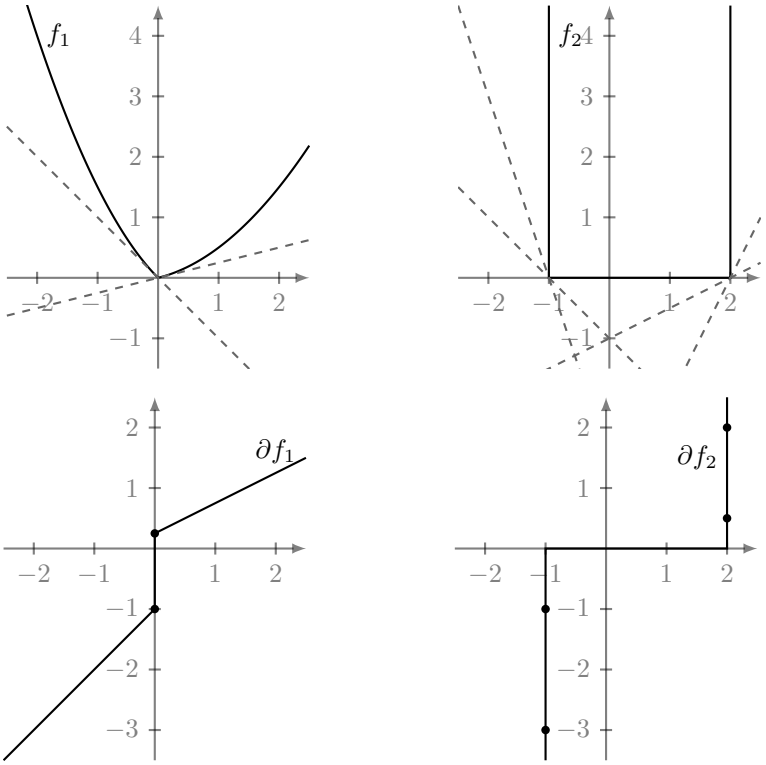


Figure 2.7: The function f_1 is not differentiable at 0 and f_2 is not differentiable at -1 and 2. The function f_2 is the indicator function of the interval $[-1, 2]$. The subdifferential generalizes differentiability for convex functions using the property of a local linear approximation. Thanks to convexity, if such tangent lines to the graph exist, they lie below the graph. The subdifferential mapping is a set-valued mapping. For each point it maps to the set of possible slopes that yield a tangent line to the graph (at this point). The lower plots visualize the respective subdifferentials of the functions in the upper plot. The slopes of the tangents that are visualized at the top are represented by a dot in the lower plot.

Definition 2.29 (set-valued mapping)**.** *A set-valued mapping $F\colon \mathbb{R}^N \rightrightarrows \mathbb{R}^M$ is a mapping that maps each $x \in \mathbb{R}^N$ to a subset of $\mathbb{R}^M$. The graph of the mapping F is given by*

$$\operatorname{Graph} F := \{(x, u) \in \mathbb{R}^N \times \mathbb{R}^M \,|\, u \in F(x)\} \subset \mathbb{R}^N \times \mathbb{R}^M .$$

For a set-valued mapping the (effective) domain is defined by

$$\operatorname{dom} F := \{x \in \mathbb{R}^N \,|\, F(x) \neq \emptyset\} \subset \mathbb{R}^N .$$

This concept generalizes that of functions. A single-valued mapping can also be considered as a set-valued mapping. Therefore, it is common to abuse notation and use $u \in F(x)$ for single-valued mappings F equivalent to $u = F(x)$. Vice versa it is also common to use $u = F(x)$ instead of $u \in F(x)$ for a singleton-valued mapping, i.e., a set-valued mapping that maps to sets that contain only a single element. As for single-valued mappings, we assume that set-valued mappings are defined on the whole space $\mathbb{R}^N$ by possibly extending them with the value $\emptyset$ (the empty set).

We introduce some operations on and with set-valued mappings.

Definition 2.30 (operations with set-valued mappings)**.** *Let $F, G\colon \mathbb{R}^N \rightrightarrows \mathbb{R}^M$, $H\colon \mathbb{R}^M \rightrightarrows \mathbb{R}^L$ be set-*

valued mappings and let $\lambda \in \mathbb{R}$. Then

$$(F+G)(x) := \{u_1 + u_2 |\, u_1 \in F(x), u_2 \in G(x)\}\,,$$
$$(\lambda F)(x) := \{\lambda u |\, u \in F(x)\}\,,$$

the composition of H and G is given as

$$(H \circ G)(x) := \{w \in H(u) |\, u \in G(x)\}\,,$$

and the inverse mapping $F^{-1}\colon \mathbb{R}^M \rightrightarrows \mathbb{R}^N$ is defined as

$$F^{-1}(u) := \{x \in \mathbb{R}^N |\, u \in F(x)\}\,.$$

Figure 2.7 suggests that tangents to the graph of a convex function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ always lie below the graph. This property is used for defining the subgradient (for convex functions). In Definition 4.12 we encounter another generalization of the subgradient to not necessarily convex functions. However, understanding the convex case first is fundamental.

Definition 2.31 (subgradient, subdifferential)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a convex function. A vector $v \in \mathbb{R}^N$ is called subgradient at $\bar{x} \in \mathbb{R}^N$, denoted by $v \in \partial f(\bar{x})$, if the following subgradient inequality holds:*

$$f(x) \geq f(\bar{x}) + \langle v, x - \bar{x}\rangle\,, \quad \text{for all } x \in \mathbb{R}^N\,. \tag{2.1}$$

The relation of $\bar{x}$ and v gives rise to define the subdifferential of f as the set-valued mapping $\partial f\colon \mathbb{R}^N \rightrightarrows \mathbb{R}^N$ by $\operatorname{Graph} \partial f := \{(x, v) \in \mathbb{R}^N \times \mathbb{R}^N |\, v \in \partial f(x)\}$.

Moreover, the function f is called subdifferentiable at $\bar{x} \in \mathbb{R}^N$ if $\partial f(\bar{x}) \neq \emptyset$. If the function is subdifferentiable at all points $\bar{x} \in \operatorname{dom} f$, then the function f is called subdifferentiable.

It is common to also call the set of subgradient vectors $\partial f(\bar{x})$ at a point $\bar{x}$ the subgradient at $\bar{x}$. The different meanings will always be clear from the context.

Example 2.11. Let $C \subset \mathbb{R}^N$ be a closed convex set and denote by δ_C the associated indicator function. According to the subgradient inequality, for $\bar{x} \in C$, it holds that

$$v \in \partial\delta_C(\bar{x}) \quad \Leftrightarrow \quad \delta_C(x) \geq \delta_C(\bar{x}) + \langle v, x - \bar{x}\rangle\,,\ \forall x \in \mathbb{R}^N \quad \Leftrightarrow \quad 0 \geq \langle v, x - \bar{x}\rangle\,,\ \forall x \in C\,.$$

Obviously, if $v_1, v_2 \in \partial\delta_C(\bar{x})$ and $\lambda \geq 0$, then $v_1 + v_2 \in \partial\delta_C(\bar{x})$ and $\lambda v_1 \in \partial\delta_C(\bar{x})$. Using Theorem 2.5, this implies that $\partial\delta_C(\bar{x})$ is a convex cone. It is the so-called *normal cone* to the set C, which can be defined by the last relation. In particular, $0 \in \partial\delta_C(\bar{x})$. Therefore, the subgradient is not empty in C. In the context of variational analysis (see Chapter 4) we consider the normal cone to an (arbitrary) set in more detail.

Example 2.12. We continue Example 2.11. Let $C = [0, \infty) \times [0, \infty) \subset \mathbb{R}^2$. It is a closed convex subset and the subdifferential is given as

$$\partial\delta_C(x) = \begin{cases} \{0\}, & \text{if } x \in \operatorname{int} C\,, \\ (-\infty, 0] \times \{0\}, & \text{if } x \in (0, \infty) \times \{0\}\,, \\ \{0\} \times (-\infty, 0], & \text{if } x \in \{0\} \times (0, \infty)\,, \\ (-\infty, 0] \times (-\infty, 0], & \text{if } x = (0, 0)^\top\,. \end{cases}$$

Let us first make sure that the subgradient of a convex function is a generalization of the gradient for a smooth convex function. Since the subgradient represents a tangent that lies below the graph of the function this fact is clear.

Proposition 2.32 (subgradient of a differentiable convex function)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper convex function and let f be differentiable at $\bar{x} \in \operatorname{dom} f$. Then $\partial f(\bar{x}) = \{\nabla f(\bar{x})\}$, and in particular*

$$f(x) \geq f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle\,, \quad \text{for all } x \in \mathbb{R}^N\,.$$

Proof. We omit the proof as Proposition 4.13 presents a more general statement. For a direct proof of this statement see, for example, [Roc70, Thm. 25.1]. □

Proposition 2.33 (existence of subgradients)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper convex function. Then $\partial f(x) \neq \emptyset$ for all $x \in \operatorname{int} \operatorname{dom} f$.*

Proof. The proof can be found in [BC11, Prop. 16.14, Cor. 16.15]. □

From the analysis of differentiable functions, we have an important characterization of minimizers, namely the derivative (the gradient) of the function necessarily vanishes at a minimum. As the subgradient is a generalization of the gradient, we can expect an analogue property for proper convex functions. In Examples 2.11 and 2.12 the minimum is the whole set C and for all $x \in C$ the subdifferential contains 0 (even at the boundary of C). The following theorem formalizes this property, which is known as *Fermat's rule*. For convex functions this property is also sufficient. The existence of minimizers was discussed in Section 2.1.3.

Theorem 2.34 (Fermat's rule)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper convex functions. Then*

$$\bar{x} \in \arg\min_{x \in \mathbb{R}^N} f(x) \qquad \Leftrightarrow \qquad 0 \in \partial f(\bar{x})\,.$$

Proof. By definition $\bar{x}$ solves $\min_{x \in \mathbb{R}^N} f(x)$ is equivalent to $f(\bar{x}) \leq f(x)$ for all $x \in \mathbb{R}^N$. This is equivalent to $f(\bar{x}) + \langle 0, x - \bar{x} \rangle \leq f(x)$, which is the subgradient inequality for the subgradient $0 \in \partial f(\bar{x})$. □

Now, we collect some properties for the subdifferential of a convex function. These will sharpen the intuition and provide tools for calculations with the subdifferential.

Proposition 2.35 (convexity of the subgradient)**.** *The subdifferential of a function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ at $\bar{x} \in \mathbb{R}^N$ is a closed convex set.*

Proof. Obviously, for any $x \in \mathbb{R}^N$ the set $\{v \in \mathbb{R}^N |\, f(x) \geq f(\bar{x}) + \langle v, x - \bar{x} \rangle\}$ is convex and closed. Theorem 2.2 and $\partial f(\bar{x}) = \bigcap_{x \in \mathbb{R}^N} \{v \in \mathbb{R}^N |\, f(x) \geq f(\bar{x}) + \langle v, x - \bar{x} \rangle\}$ imply convexity and closedness of $\partial f(\bar{x})$. □

Proposition 2.36 (calculation rules for subgradients)**.** *Let $f, g\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be two proper convex functions and let $A\colon \mathbb{R}^M \to \mathbb{R}^N$ be a linear transformation. Denote by $T_{\bar{x}}\colon \mathbb{R}^N \to \mathbb{R}^N$ the mapping that translates $x \in \mathbb{R}^N$ to the point $x + \bar{x}$. Then it holds that*

(i) $\partial(\lambda f) = \lambda \partial f$ *whenever* $\lambda > 0$,

(ii) $\partial(f \circ T_{\bar{x}})(x) = \partial f(x + \bar{x})$ *for* $\bar{x} \in \mathbb{R}^N$,

(iii) $\partial(f + g) \supset \partial f + \partial g$ *and equality holds if f is continuous in a point* $\bar{x} \in \operatorname{dom} f \cap \operatorname{dom} g$,

(iv) $\partial(f \circ A) \supset A^\top \circ \partial f \circ A$ *and equality holds if f is continuous in a point* $\bar{x} \in \operatorname{dom} f \cap \operatorname{im} A$.

Proof. These properties are also valid for proper convex functions in infinite dimensional normed real vector spaces, which is proved in [BL11, Thm. 6.51]. □

We extend Example 2.11 and show some practical examples for subgradients. They will appear again in the next section, e.g., in Example 2.16, where we consider a key building block for many convex optimization algorithms in Section 2.2.

Example 2.13 (Constrained minimization). Let $C \subset \mathbb{R}^N$ be a closed convex set and let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper convex function that is differentiable on C. Fermat's rule (Theorem 2.34) implies that $\bar{x} \in \mathbb{R}^N$ minimizes $f+\delta_C$ is equivalent to $0 \in \partial(f+\delta_C)(\bar{x})$. Using Proposition 2.36, it holds that $\partial(f+\delta_C)(\bar{x}) = \partial f(\bar{x}) + \partial\delta_C(\bar{x})$. Thanks to Proposition 2.32 it further reduces to $-\nabla f(\bar{x}) \in \partial\delta_C(\bar{x})$. Assuming the minimizer $\bar{x}$ lies in the interior of C, we have the condition $0 = \nabla f(\bar{x})$. If the solution lies on the boundary, then Fermat's rule states that the negative gradient (the direction of the steepest descent of f) is in the normal cone of the set C at $\bar{x}$ (cf. Example 2.11), which is a geometric interpretation.

Example 2.14. (i) Let $f\colon \mathbb{R}^N \to \mathbb{R}$, $x \mapsto \frac{1}{2}\|x\|^2$. Then $\partial f(x) = \{\nabla f(x)\} = \{x\}$ for all $x \in \mathbb{R}^N$.

(ii) Let $A\colon \mathbb{R}^N \to \mathbb{R}^M$ be a linear transformation, $\bar{x} \in \mathbb{R}^M$, and $f\colon \mathbb{R}^N \to \mathbb{R}$, $x \mapsto \frac{1}{2}\|Ax - \bar{x}\|^2$. Then $\nabla f(x) = A^\top Ax - A^\top \bar{x}$ for all $x \in \mathbb{R}^N$.

(iii) Let $f\colon \mathbb{R}^N \to \mathbb{R}$, $x \mapsto \|x\|_1 = \sum_{i=1}^n |x_i|$. Then $(\partial f(x))_i = \{x_i/|x_i|\}$ whenever $x_i \neq 0$ and $(\partial f(x))_i = [-1, 1]$ otherwise.

(iv) Let $f\colon \mathbb{R}^N \to \mathbb{R}$, $x \mapsto \|x\|$. Then $(\partial f(x))_i = x_i/\|x\|$ whenever $x \neq 0$ and $(\partial f(x))_i = \overline{B}_1(0)$ otherwise.

We conclude this subsection with another characterization of convexity, which is based on the subdifferential and thus valid for nonsmooth functions—in analogy to Theorem 2.15 for smooth functions. We will meet this property later again. For example it can be used to define stronger versions of convexity. Furthermore, there is a powerful theory about objects whose definition is purely based on this property: namely monotone operators (see [BC11]). We meet these objects briefly in Definition 2.50. The monotony of the subgradient mapping can be used to characterize convexity. The proof of the following theorem is a simple consequence of the subgradient inequality.

Theorem 2.37 (monotonicity of the subdifferential)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper convex function. Then, for all $x, \bar{x} \in \operatorname{dom} \partial f$ it holds that*

$$\langle v - \bar{v}, x - \bar{x}\rangle \geq 0\,, \quad \textit{for all } v \in \partial f(x), \bar{v} \in \partial f(\bar{x})\,.$$

Proof. The proof follows by summing the subgradient inequality for x and $\bar{x}$ and the subgradient inequality for swapped x and $\bar{x}$. □

Combining this results with Proposition 2.33 for convex functions f with $\operatorname{dom} f = \operatorname{int}\operatorname{dom} f$ directly implies monotonicity of the subdifferential.

2.1.5 Proximal mapping

In this section we introduce the proximity operator, which traces back to Moreau in 1965 [Mor65]. The *proximity operator* or *proximal mapping* can be seen as a generalization to the projection onto a convex set, i.e., the operator that maps $\tilde{x} \mapsto \arg\min_{x\in C} \frac{1}{2}\|x - \tilde{x}\|^2$ for a convex set C. The proximity operator is one of the key construction elements of many convex optimization algorithms. It defines the proximal

point algorithm (see Section 2.2.2), appears in the update step of forward–backward splitting methods (Section 2.2.3), in Fista (Section 2.2.5), the primal–dual algorithm in Section 2.2.6 and many other related algorithms. Moreover, it is essential for the algorithm that we propose in Chapter 3.

Definition 2.38 (proximity operator)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper, lsc, convex function and let $\lambda > 0$. The proximity operator $(\mathrm{id} + \lambda\partial f)^{-1}$ that maps from $\mathbb{R}^N$ to $\mathbb{R}^N$ is defined by*

$$(\mathrm{id} + \lambda\partial f)^{-1}(\bar{x}) := \arg\min_{x\in\mathbb{R}^N} f(x) + \frac{1}{2\lambda}\|x - \bar{x}\|^2 .$$

Lemma 2.39 (well-definedness of the proximal operator)**.** *Consider the situation of Definition 2.38. The mapping $(\mathrm{id} + \lambda\partial f)^{-1}$ is well-defined in the sense that the minimization problem yields a unique minimizer.*

Proof. Let $F(x) := f(x) + \frac{1}{2\lambda}\|x - \bar{x}\|^2$. Since f is proper $\operatorname{dom} F \neq \emptyset$. The second term ensures coercivity of F (see Definition 2.27). Applying Theorem 2.28 yields the existence of a minimizer.

In order to verify uniqueness, let $x' \neq x''$ be two minimizers of F. Then, due to the strict convexity of F, we have $F(\frac{1}{2}(x' + x'')) < \frac{1}{2}(F(x') + F(x''))$. Optimality of x' and x'' implies that $F(\frac{1}{2}(x' + x'')) < \min_{x\in\mathbb{R}^N} F(x)$, which is a contradiction. Hence $x' = x''$. □

Remark 2.15. At first glance the symbol for the proximity operator in Definition 2.38 seems to be unnatural, however, it is not just a symbol. Let us compute the minimizer of the problem in its definition by means of Fermat's rule (Theorem 2.34):

$$0 \in \partial\left(f(x) + \frac{1}{2\lambda}\|x - \bar{x}\|^2\right) = \partial f(x) + \frac{1}{\lambda}(x - \bar{x}) ,$$

where the equality uses Proposition 2.36. This can be equivalently expressed as

$$\bar{x} \in (\mathrm{id} + \lambda\partial f)(x) ,$$

which gives a meaning to the notation in Definition 2.38.

As we mentioned above, the proximity operator generalizes the projection operator onto convex sets. Consider the indicator function δ_C of a nonempty, closed, convex set $C \subset \mathbb{R}^N$ (Definition 2.19). The value of the corresponding proximity operator $(\mathrm{id} + \lambda\partial\delta_C)^{-1}$ for a point $\bar{x} \in \mathbb{R}^N$ can be estimated as the minimizer of the following equivalent problems:

$$\min_{x\in\mathbb{R}^N} \delta_C(x) + \frac{1}{2\lambda}\|x - \bar{x}\|^2 = \min_{x\in C} \frac{1}{2\lambda}\|x - \bar{x}\|^2 ,$$

where the right problem seeks for minimizing the squared distance to the convex set C. The minimizer exists and is the projection of $\bar{x}$ onto the convex set C. An important property that the proximity operator shares with the projection operator is the following. It is a Lipschitz continuous function with Lipschitz constant 1 as defined in Definition 2.24. This can also be considered as a nonexpansiveness.

Proposition 2.40 (nonexpansiveness of the proximity operator)**.** *Let the assumptions be as in Definition 2.38. The proximity operator is nonexpansive, i.e., for $x, y \in \operatorname{dom}\partial f$ holds*

$$\|(\mathrm{id} + \lambda\partial f)^{-1}(x) - (\mathrm{id} + \lambda\partial f)^{-1}(y)\| \leq \|x - y\| .$$

Proof. As $(\mathrm{id} + \lambda\partial f)^{-1}$ is a well-defined single-valued mapping, we may define $\bar{x} := (\mathrm{id} + \lambda\partial f)^{-1}(x)$ and $\bar{y} := (\mathrm{id} + \lambda\partial f)^{-1}(y)$. Then, by definition of the inverse mapping $x \in (\mathrm{id} + \lambda\partial f)(\bar{x})$ and $y \in (\mathrm{id} + \lambda\partial f)(\bar{y})$. There exist $\bar{v}_x \in \partial f(\bar{x})$ and $\bar{v}_y \in \partial f(\bar{y})$ such that $x = \bar{x} + \lambda\bar{v}_x$ and $y = \bar{y} + \lambda\bar{v}_y$, and it holds that

$$\begin{aligned}\|x - y\|^2 &= \|(\bar{x} + \lambda\bar{v}_x) - (\bar{y} + \lambda\bar{v}_y)\|^2 \\ &= \|\bar{x} - \bar{y}\|^2 + \lambda^2\|\bar{v}_x - \bar{v}_y\|^2 + 2\lambda\left\langle \bar{x} - \bar{y}, \bar{v}_x - \bar{v}_y\right\rangle \geq \|\bar{x} - \bar{y}\|^2 ,\end{aligned}$$

where the inequality holds due to Theorem 2.37. This concludes the proof. □

Before we present some more examples, we summarize some rules to simplify calculations with proximity operators.

Lemma 2.41 (calculation rules for the proximity operator)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper, convex, lsc function, $T_{\bar{x}}$ be as in Proposition 2.36, and let $\lambda > 0$.*

(i) For $\sigma \in \mathbb{R}$: $g = f + \sigma \Rightarrow (\mathrm{id} + \lambda\partial g)^{-1} = (\mathrm{id} + \lambda\partial f)^{-1}$.

(ii) For $\tau, \sigma > 0$: $g = (\tau f) \circ (\sigma\mathrm{id}) \Rightarrow (\mathrm{id} + \lambda\partial g)^{-1} = \sigma^{-1}\mathrm{id} \circ (\mathrm{id} + \lambda\tau\sigma^2\partial f)^{-1} \circ \sigma\mathrm{id}$.

(iii) For $x^0, y^0 \in \mathbb{R}^N$: $g = f \circ T_{x^0} + \left\langle y^0, \cdot\right\rangle \Rightarrow (\mathrm{id} + \lambda\partial g)^{-1} = T_{-x^0} \circ (\mathrm{id} + \lambda\partial f)^{-1} \circ T_{x^0 - \lambda y^0}$.

(iv) Let $L\colon \mathbb{R}^N \to \mathbb{R}^N$ be an isometric isomorphism. Then

$$g = f \circ L \Rightarrow (\mathrm{id} + \lambda\partial g)^{-1} = L^\top \circ (\mathrm{id} + \lambda\partial f)^{-1} \circ L .$$

(v) Let $g\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be proper, convex, lsc. Then

$$h(x, y) = f(x) + g(y) \Rightarrow (\mathrm{id} + \lambda\partial h)^{-1}(x, y) = \begin{pmatrix}(\mathrm{id} + \lambda\partial f)^{-1}(x) \\ (\mathrm{id} + \lambda\partial g)^{-1}(y)\end{pmatrix} .$$

Proof. (i) This is obvious from the definition.

(ii) The following simple equations verify the item. For $\sigma, \tau > 0$ holds that

$$\begin{aligned}(\mathrm{id} + \lambda\partial g)^{-1}(\tilde{x}) &= \operatorname*{arg\,min}_{x \in \mathbb{R}^N} \tau f(\sigma x) + \frac{1}{2\lambda}\|x - \tilde{x}\|^2 \\ &= \sigma^{-1} \cdot \operatorname*{arg\,min}_{y \in \mathbb{R}^N} f(y) + \frac{1}{2\lambda\tau}\|\sigma^{-1}y - \tilde{x}\|^2 \\ &= \sigma^{-1} \cdot \operatorname*{arg\,min}_{y \in \mathbb{R}^N} f(y) + \frac{1}{2\lambda\tau\sigma^2}\|y - (\sigma\tilde{x})\|^2 \\ &= (\sigma^{-1}\mathrm{id} \circ (\mathrm{id} + \lambda\tau\sigma^2\partial f)^{-1} \circ \sigma\mathrm{id})(\tilde{x}) .\end{aligned}$$

(iii) For $x^0, y^0 \in \mathbb{R}^N$ holds that

$$\begin{aligned}(\mathrm{id} + \lambda\partial g)^{-1}(\tilde{x}) &= \operatorname*{arg\,min}_{x \in \mathbb{R}^N} f(x + x^0) + \left\langle y^0, x\right\rangle + \frac{1}{2\lambda}\|x - \tilde{x}\|^2 \\ &= \operatorname*{arg\,min}_{x \in \mathbb{R}^N} f(x + x^0) + \frac{1}{2\lambda}\|x - (\tilde{x} - \lambda y^0)\|^2 \\ &= -x^0 + \operatorname*{arg\,min}_{y \in \mathbb{R}^N} f(y) + \frac{1}{2\lambda}\|y - (\tilde{x} + x^0 - \lambda y^0)\|^2 \\ &= (T_{-x^0} \circ (\mathrm{id} + \lambda\partial f)^{-1} \circ T_{x^0 - \lambda y^0})(\tilde{x}) .\end{aligned}$$

(iv) It holds that $L^\top L = \mathrm{id}$ and $\|Lx\| = \|x\|$. Therefore the following proves the statement.

$$\begin{aligned}(\mathrm{id} + \lambda\partial g)^{-1}(\tilde{x}) &= \arg\min_{x\in\mathbb{R}^N} f(Lx) + \frac{1}{2\lambda}\|x - \tilde{x}\|^2 \\ &= \arg\min_{x\in\operatorname{im} L^\top} f(Lx) + \frac{1}{2\lambda}\|L(x - \tilde{x})\|^2 \\ &= L^\top \cdot \arg\min_{y\in\mathbb{R}^N} f(y) + \frac{1}{2\lambda}\|y - L\tilde{x}\|^2 \\ &= (L^\top \circ (\mathrm{id} + \lambda\partial f)^{-1} \circ L)(\tilde{x})\end{aligned}$$

(v) The last statement is obvious as variables are strictly separated.

□

Finally, we present several solutions to the proximal operator of convex functions. As the evaluation of the proximal operator will play an important role for optimization algorithms, we consider terms that are particularly interesting in applications.

Example 2.16. For all examples, let $\lambda > 0$.

(i) Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $x \mapsto \frac{1}{2}\|x\|^2$, and $\tilde{x} \in \mathbb{R}^N$. Then $(\mathrm{id} + \lambda\partial f)^{-1}(\tilde{x}) = \frac{\tilde{x}}{1+\lambda}$.

(ii) Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $x \mapsto \frac{1}{2}\|Ax - y\|^2$, $A\colon \mathbb{R}^N \to \mathbb{R}^M$ a linear transformation, and $\tilde{x} \in \mathbb{R}^N$, $y \in \mathbb{R}^M$. Then $(\mathrm{id} + \lambda\partial f)^{-1}(\tilde{x})$ is given as solution of the system of equations $(\mathrm{id} + \lambda A^\top A)x = \tilde{x} + A^\top y$.

(iii) Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $x \mapsto \delta_{\overline{B}_1(0)}(x)$, and $\tilde{x} \in \mathbb{R}^N$. Then $(\mathrm{id} + \lambda\partial f)^{-1}(\tilde{x})$ is given as the orthogonal projection of $\tilde{x}$ onto the closed unit ball, i.e.,

$$(\mathrm{id} + \lambda\partial\delta_{\overline{B}_1(0)})^{-1}(\tilde{x}) = \begin{cases}\tilde{x}/\|\tilde{x}\|, & \text{if } \tilde{x} \in \mathbb{R}^N \setminus \overline{B}_1(0)\,, \\ \tilde{x}, & \text{if } \tilde{x} \in \overline{B}_1(0)\,.\end{cases}$$

(iv) Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $x \mapsto \|x\|_1$, and $\tilde{x} \in \mathbb{R}^N$. Then $(\mathrm{id} + \lambda\partial f)^{-1}(\tilde{x})$ is the *soft thresholding operator*, thanks to Lemma 2.41(v), given coordinate-wise by (for $a \in \mathbb{R}$ define $\operatorname{sign} a := a/|a|$ if $a \neq 0$ and $\operatorname{sign} 0 := 0$ otherwise)

$$(\mathrm{id} + \lambda\partial f)^{-1}(\tilde{x}_i) = \max(0, |\tilde{x}_i| - \lambda)\operatorname{sign}(\tilde{x}_i)$$

In the next subsection, we meet the celebrated Moreau identity in Theorem 2.44. It relates the proximal map of a function with the proximal map of its convex conjugate function, which we introduce in next. However, considering conjugate functions has many other advantages.

2.1.6 Legendre–Fenchel conjugate

In this subsection, we will see that convex functions reveal an interesting and powerful duality. This duality is important for the primal–dual algorithm in Section 2.2.6. It is related to the dual representation of, for example, a conic. A conic can be described as a set of points or—in the dual representation—as an envelope of tangents to the conic. In convex analysis the dual of a closed convex set may be described as intersection of half-spaces that contain the set. As usual in convex analysis, the concept is transported from convex sets to convex functions via the epigraph. We define the (convex) conjugate of a convex function and provide properties, which will make clear the descriptive duality principle just mentioned.

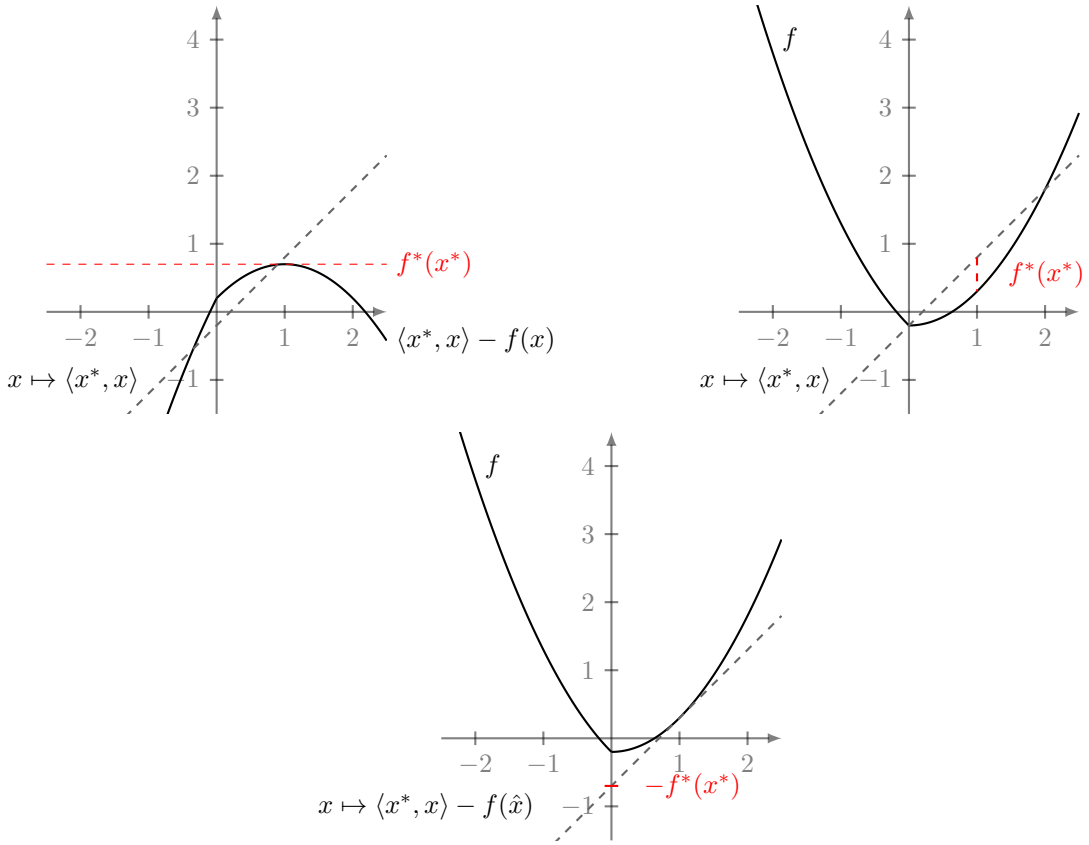


Figure 2.8: All plots visualize the value of the dual function f^* at a certain $x^* \in \mathbb{R}^N$ in a different way. TOP LEFT: Shows the function for which we seek the supremum $\langle x^*, x\rangle - f(x)$. TOP RIGHT: The value of $f^*(x^*)$ is found as the largest "gap" (with sign) between $\langle x^*, x\rangle$ and $f(x)$. BOTTOM: At the optimum $(\hat{x}, f(\hat{x}))$, the linearization with slope x^* is tangent to f at $\hat{x}$ and intersects the y-axis in $-f^*(x^*)$.

As the duality theory of convex functions is very complex and goes far beyond the scope of this thesis, we only focus on the definition, some intuition and a few examples. This is sufficient to understand the material that is considered in this thesis. The Legendre–Fenchel conjugate is not central for the development of the contribution of this thesis. For further reading we refer the interested reader to [Roc70].

Definition 2.42 (Legendre–Fenchel conjugate)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper function. We define the convex conjugate or Legendre–Fenchel conjugate by*

$$f^*(x^*) = \sup_{x \in \mathbb{R}^N} \langle x^*, x\rangle - f(x)\,.$$

Figure 2.8 shows three different ways to consider this definition. As the third point of view in Figure 2.8 suggests, the negative of the value of the dual function at x^* is determined by the intersection with supremal value of lines with slope x^* below the function f. The line that yields the supremum is necessarily tangent to the graph. Let us confirm this by a simple consideration of a proper, lsc, convex function f. The optimality condition for $\langle x^*, x\rangle - f(x)$, which is $0 \in x^* - \partial f(x)$, immediately implies $x^* \in \partial f(x)$. For a differentiable function f it is $x^* = \nabla f(x)$. This is exactly the condition for tangency.

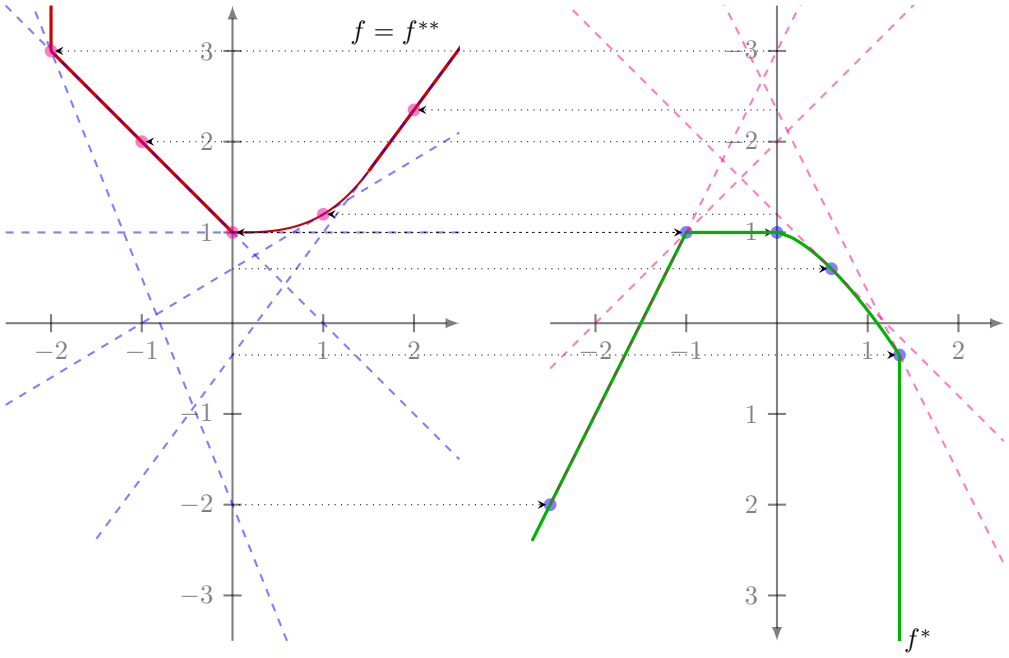


Figure 2.9: Construction of the Legendre–Fenchel conjugate of a convex function. The left plot shows the function f with some tangent lines—these are exemplary samples—to the graph of f. The intersection of a tangent with the y-axis yields the negative of the value of the dual for the respective slope. In order to cope with the sign flip, the y-axis of the right plot points downwards. The correspondences between tangents in the left plot and points in the right plot is indicated by the dotted lines and the respective big dots. In the same way, the left plot is constructed from the right one. The duality $f^{**} = f$ becomes clear.

The Legendre–Fenchel conjugate has the property that for proper, lsc, convex functions f it holds that $(f^*)^* =: f^{**} = f$ (see [Roc70, Thm. 12.2]). This means that the convex conjugate of the convex conjugate of a function coincides with the original function. Figure 2.9 confirms this fact. Moreover, it shows a geometric construction of the convex conjugate.

Before we come to some examples, we present a simplification when it comes to calculations.

Proposition 2.43 (calculation rule for the Legendre–Fenchel transform)**.** *Let g be a convex function on $\mathbb{R}^N$, and let*

$$f(x) = g(A(x - a)) + \langle x, a^* \rangle + \lambda \,,$$

where A is a one-to-one linear transformation form $\mathbb{R}^N$ onto $\mathbb{R}^N$, a and a^* are vectors in $\mathbb{R}^N$, and $\lambda \in \mathbb{R}$. Then

$$f^*(x^*) = g^*(A^{-\top}(x^* - a^*)) + \langle x^*, a \rangle + \lambda^* \,,$$

where $A^{-\top} = (A^{-1})^\top$ and $\lambda^* = -\lambda - \langle a, a^* \rangle$.

Proof. Using the substituition $y = A(x-a)$ allows us to compute f^*

$$\begin{aligned} f^*(x^*) &= \sup_{x\in\mathbb{R}^N} \langle x, x^*\rangle - g(A(x-a)) - \langle x, a^*\rangle - \lambda \\ &= \sup_{y\in\mathbb{R}^N} \left\langle A^{-1}y + a, x^*\right\rangle - g(y) - \left\langle A^{-1}y + a, a^*\right\rangle - \lambda \\ &= \sup_{y\in\mathbb{R}^N} \left\langle A^{-1}y, x^* - a^*\right\rangle - g(y) + \langle a, x^* - a^*\rangle - \lambda \\ &= \sup_{y\in\mathbb{R}^N} \left\langle y, A^{-\top}(x^* - a^*)\right\rangle - g(y) + \langle x^*, a\rangle + \lambda^* \,. \end{aligned}$$

The supremum in the last line is $g^*(A^{-\top}(x^* - a^*))$ by definition. □

Example 2.17. In order to understand the geometry of dualization best, it is important to know which functions do not change under the dualization operation.

(i) Let $f\colon \mathbb{R} \to \mathbb{R}$, $x \mapsto \frac{1}{2}x^2$. Then $f^*(x^*) = \frac{1}{2}(x^*)^2$. The supremum $\sup_{x\in\mathbb{R}} xx^* - \frac{1}{2}x^2$ is attained at $x = x^*$. Plugging-in yields the expression for f^*.

(ii) Let $f\colon \mathbb{R}^N \to \mathbb{R}$, $x \mapsto \frac{1}{2}\|x\|^2$. Then $f^*(x^*) = \frac{1}{2}\|x^*\|^2$.

Example 2.18. (i) Let $f\colon \mathbb{R}^N \to \mathbb{R}$, $x \mapsto \|x\|$. Then $f^*(x^*) = \delta_{\overline{B}_1(0)}(x^*)$. Consider the expression $g(x) := \langle x, x^*\rangle - \|x\|$. We determine for each x^* the point $x \in \mathbb{R}^N$ for which g attains its supremum. Obviously, $g(tx^*/\|x^*\|) \geq g(x)$ for all $x \in \mathbb{R}^N$ with $\|x\| = t \in \mathbb{R}_+$. Therefore, for each x^* the supremum of g is found along the one-dimensional line in direction of x^*. As $g(tx^*/\|x^*\|) = t(\|x^*\| - 1)$, it is clear that for $x^* \in \overline{B}_1(0)$ the supremum is 0 and is attained for $t = 0$. For $x^* \in \mathbb{R}^N \setminus \overline{B}_1(0)$ the supremum of g is ∞.

(ii) Consider the Huber function, i.e., let $f\colon \mathbb{R}^N \to \mathbb{R}$, $f(x) = \frac{1}{2\varepsilon}\|x\|^2$ for $\|x\| < \varepsilon$ and $f(x) = \|x\| - \frac{\varepsilon}{2}$ for $\|x\| \geq \varepsilon$ with $\varepsilon > 0$. A simple calculation shows that the convex conjugate is $f^*(x^*) = \frac{\varepsilon}{2}\|x^*\|^2 + \delta_{\overline{B}_1(0)}(x^*)$.

As promised at the end of the preceding section, we state Moreau's identity.

Theorem 2.44 (Moreau's identity)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a lsc, proper, convex function and denote by f^* its convex conjugate function. Then, it holds that*

$$x = (\mathrm{id} + \lambda\partial f)^{-1}(x) + \lambda\left(\mathrm{id} + \frac{1}{\lambda}\partial f^*\right)^{-1}\left(\frac{x}{\lambda}\right).$$

Proof. The proof can be found in [Roc70, Thm. 31.5]. □

For example, Moreau's identity allows us to evaluate the proximal map of the convex conjugate function without actually computing the conjugate function. Only the function f and its associated proximal map must be computed. For $\lambda = 1$, Moreau's identity has a particularly simple form; It reads

$$x = (\mathrm{id} + \partial f)^{-1}(x) + (\mathrm{id} + \partial f^*)^{-1}(x)\,.$$

2.1.7 Strong convexity

Based on Theorem 2.37 we want to introduce a stronger notion of convexity: strong convexity. A convex function that is additionally strongly convex has many nice properties and provides several interesting inequalities. However, we will only need a few. For example, a strongly convex function has a particularly simple lower bound. For each point there is a quadratic minorizer of the function such that they meet in that point. See Figure 2.10. Strongly convex functions play a central role in Chapter 3, where we develop a new algorithm for functions with this property.

Definition 2.45 (strong convexity)**.** *A convex function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is called strongly convex if there exists $m > 0$ such that the function $f(x) - (m/2)\|x\|^2$ is convex. The largest such parameter m is called convexity parameter or modulus.*

We denote the class of proper, strongly convex functions with convexity parameter $m > 0$ by $\mathscr{S}_m$. In order to make it easier to work with both, convex and strongly convex functions, we define the class of proper convex functions by $\mathscr{S}_0$.

Strongly convex functions allow us to sharpen the subgradient inequality.

Lemma 2.46 (quadratic subgradient inequality)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper, strongly convex function with convexity parameter $m > 0$. Then for any $\bar{x} \in \operatorname{dom} \partial f$ it holds that*

$$f(x) \geq f(\bar{x}) + \langle v, x - \bar{x}\rangle + \frac{m}{2}\|x - \bar{x}\|^2\,, \quad \textit{for all } v \in \partial f(\bar{x}) \textit{ and } x \in \operatorname{dom} f\,.$$

Proof. A simple calculation with the subgradient inequality (2.1) applied to $f(x) - (m/2)\|x\|^2$ proves the statement. □

Moreover, strong convexity is also reflected in the Lipschitz continuity of the proximal mapping.

Proposition 2.47 (Lipschitz continuity of proximal map)**.** *Let $f \in \mathscr{S}_m$, $m > 0$, $\lambda > 0$. Then $\lambda f \in \mathscr{S}_{\lambda m}$ and $(\mathrm{id} + \lambda f)^{-1}$ is $(1 + \lambda m)^{-1}$-Lipschitz continuous, i.e. for all $x, y \in \mathbb{R}^N$:*

$$\|(\mathrm{id} + \lambda f)^{-1}(x) - (\mathrm{id} + \lambda f)^{-1}(y)\| \leq (\lambda m + 1)^{-1}\|x - y\|\,. \tag{2.2}$$

Proof. For the proof we refer to [BC11, Prop. 23.11]. □

Another useful characterization in analogy to Theorem 2.37 is the following.

Theorem 2.48 (monotonicity for strongly convex functions)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper, strongly convex function. Then for all $x, \bar{x} \in \operatorname{dom} \partial f$ it holds that*

$$\langle v - \bar{v}, x - \bar{x}\rangle \geq m\|x - \bar{x}\|^2\,, \quad \textit{for all } v \in \partial f(x), \bar{v} \in \partial f(\bar{x})\,.$$

Proof. The proof follows from summing the quadratic subgradient inequality (Lemma 2.46) for x and $\bar{x}$ and the quadratic subgradient inequality for swapped x and $\bar{x}$. □

As this characterization of strong convexity is purely based on subgradients (and not on function values), it is well-suited for generalizations to more abstract concepts. An example for this are maximal monotone operators, which we define in Definition 2.50. In that context, this kind of relation is used to introduce strongly monotone operators. For details we refer to [BC11].

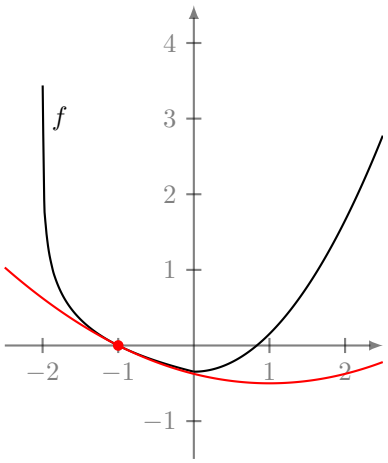


Figure 2.10: A strongly convex function f and (in red) the quadratic minorizer from Lemma 2.46 at -1.

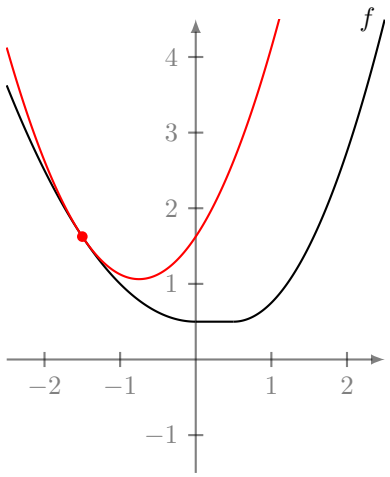


Figure 2.11: A function f with Lipschitz continuous gradient and (in red) the quadratic majorizer from Lemma 2.49 at -1.5.

2.1.8 Lipschitz continuity of the gradient of a function

Lipschitz continuity of the gradient of a function is in some way complementary to strong convexity of a function. In Lemma 2.46, we have seen that strongly convex functions have a particularly simple minorizer in each point. Functions with Lipschitz continuous gradient provide a simple majorizer. An example is shown in Figure 2.11.

Let us apply the definition of Lipschitz continuity to the gradient of a function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$. Assume that f is differentiable on a set $C \subset \operatorname{dom} f$. Then $\nabla f\colon C \to \mathbb{R}^N$ is Lipschitz continuous with Lipschitz constant $L \geq 0$, sometimes called L-Lipschitzian, if

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \text{for all } x, y \in C\,.$$

Obviously, ∇f being L-Lipschitzian implies that f is continuously differentiable on C. We denote the class of functions with this property by[2] $f \in \mathscr{C}_L^{1,1}(C)$. If the Lipschitz continuity holds on the domain of the function f we abbreviate the notation by $f \in \mathscr{C}_L^{1,1}$.

As mentioned before, functions with this property allow for quadratic majorizers, as stated in the following descent Lemma.

Lemma 2.49 (descent Lemma)**.** *Let $f\colon C \to \mathbb{R}$ have Lipschitz continuous gradient with $L > 0$ on the convex set $C \subset \mathbb{R}^N$. Then for any $\bar{x} \in C$ it holds that*

$$f(x) \leq f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x}\rangle + \frac{L}{2}\|x - \bar{x}\|^2\,, \quad \textit{for all } x \in C\,. \tag{2.3}$$

Proof. This proof is classical, but it is written in a few lines: By the mean value theorem, we have that

$$\begin{aligned} f(x) &= f(\bar{x}) + \int_0^1 \langle \nabla f(\bar{x} + t(x - \bar{x})), x - \bar{x}\rangle \, \mathrm{d}t \\ &= f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x}\rangle + \int_0^1 \langle \nabla f(\bar{x} + t(x - \bar{x})) - \nabla f(\bar{x}), x - \bar{x}\rangle \, \mathrm{d}t\,. \end{aligned}$$

[2] This notation is also used in [Nes04]. $\mathscr{C}_L^{k,m}$ denotes the class of functions that are k-times continuously differentiable and the m-th derivative is Lipschitz continuous with Lipschitz constant L.

Using the Cauchy-Schwarz inequality it follows that

$$\begin{aligned} f(x) &\leq f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle + \int_0^1 \|\nabla f(\bar{x} + t(x - \bar{x})) - \nabla f(\bar{x})\| \|x - \bar{x}\| \, \mathrm{d}t \\ &\leq f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle + \int_0^1 Lt \|x - \bar{x}\|^2 \, \mathrm{d}t \\ &= f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle + \frac{L}{2} \|x - \bar{x}\|^2 . \end{aligned}$$

□

Usually, this inequality is referred to as *(quadratic) Lipschitz upper bound of* f. Note that it does not require convexity.

This inequality is very important for many optimization algorithms. It provides a quadratic majorizer that touches the original function in one point, where they also have the same slope. As Figure 2.11 suggests, the minimizer of the quadratic majorizer yields a lower (at least not higher) function value of the original function. This idea is key for most of the algorithms considered in Section 2.2.

Before, we introduce the algorithms we recap a few facts about convergence rates.

2.1.9 Convergence rates

Let us quickly recap the different rates of convergence of an algorithm that will be important for us. It is inspired by [Nes04, Ber99]. First of all, we need to refine what quantity is to be measured. Usually, the object that converges is denoted residual (or error). For example, the residual at iteration $n \in \mathbb{N}$ can be one of the following:

$$r_n = \|\nabla f(x^n)\| , \qquad r_n = \|x^n - x^*\| , \quad \text{or} \qquad r_n = |f(x^n) - f(x^*)| ,$$

where x^* is the limit of the sequence $(x^n)_{n \in \mathbb{N}}$.

If the convergence of the sequence $(x^n)_{n \in \mathbb{N}}$ is comparable with the geometric progression of a standard sequence $(q^n)_{n \in \mathbb{N}}$ with $q \in (0, 1)$, the sequence $(x^n)_{n \in \mathbb{N}}$ converges with a *linear rate of convergence*. This is the case, if there exists $c > 0$ such that $r_n \leq c\, q^n$, which is usually abbreviated with $r_n \in \mathcal{O}(q^n)$. Another characterization is the existence of $q \in (0, 1)$ such that

$$\limsup_{n \to \infty} \frac{r_{n+1}}{r_n} \leq q .$$

The associated complexity is given by the maximal number of iterations that are required for the residual to fall below a certain bound $\varepsilon > 0$.

The convergence rate of the residual is called *superlinear* if $\limsup_{n \to \infty} \frac{r_{n+1}}{r_n} = 0$, and *sublinear*, if $\limsup_{n \to \infty} \frac{r_{n+1}}{r_n} = 1$.

Furthermore, we distinguish different rates of superlinear convergence like quadratic or cubic. For the sublinear rate, we consider convergence at the rate of $(1/n^p)_{n \in \mathbb{N}}$ with $p > 0$, i.e., $r_n \leq c/n^p$, in which case we write $r_n \in \mathcal{O}(1/n^p)$. The larger the value of p the faster the convergence.

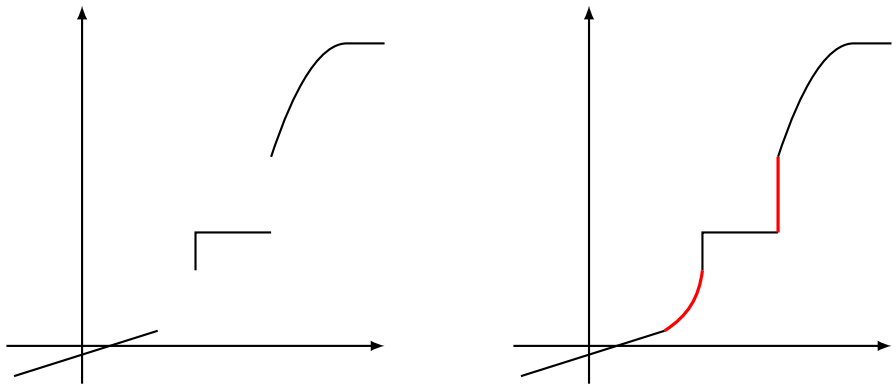


Figure 2.12: LEFT: Graph of a monotone operator. RIGHT: Graph of a maximal monotone operator. Adding the red pieces to the graph on the left yields a maximal monotone operator.

2.2 First order convex optimization

In Sections 2.2.4 to 2.2.6 we present algorithms that will be of importance throughout this thesis. In order to arrange them in the jungle of convex optimization algorithms, we give a rough overview about some important algorithmic concepts for convex optimization. The concept of maximal monotone operators has proved to be a suitable framework for that. On one hand, it is general enough to cover many algorithms and, on the other hand, it carries enough structure to develop (also practically) efficient optimization algorithms. For example finding a minimum of a convex function or finding the saddle point of a convex–concave saddle point problem comes down to the same problem, namely the problem of finding the zero of such a maximal monotone operator T, i.e., we seek[3] $x \in \mathbb{R}^N$ such that $0 \in T(x)$.

Definition 2.50 (maximal monotone operator (see [BC11, Def. 20.1 and 20.20]))**.** *(i) Let* $A \colon \mathbb{R}^N \rightrightarrows \mathbb{R}^N$*. Then* A *is monotone if*

$$\forall (x, v), (\bar{x}, \bar{v}) \in \operatorname{Graph} A : \quad \langle x - \bar{x}, v - \bar{v} \rangle \geq 0 \,.$$

(ii) A monotone operator A *is maximal monotone if there exists no monotone operator* $B \colon \mathbb{R}^N \rightrightarrows \mathbb{R}^N$ *such that* $\operatorname{Graph} B$ *properly contains* $\operatorname{Graph} A$*, i.e., for every* $(x, v) \in \mathbb{R}^N \times \mathbb{R}^N$*,*

$$(x, v) \in \operatorname{Graph} A \qquad \Leftrightarrow \qquad \forall (\bar{x}, \bar{v}) \in \operatorname{Graph} A : \quad \langle x - \bar{x}, v - \bar{v} \rangle \geq 0 \,.$$

The graphs of a monotone and a maximal monotone operator are shown in Figure 2.12. An important example is the subdifferential of a convex function. Let $f \colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be proper convex function. Then ∂f is a monotone operator. Moreover, if f is also lower semi-continuous, then ∂f is a maximal monotone operator (see for example [BC11, Thm. 20.40]). The relation is intuitive by Theorem 2.37.

Therefore, the reader who is not familiar with the concept of maximal monotone operators can think of it as ∂f with a proper, lsc, convex function f, or even as ∇f if f is additionally differentiable. We also show examples, which should make the overview understandable without prior knowledge about monotone operators.

As mentioned earlier, the proximal operator is a key concept for many practical optimization algorithms. This statement is also true when formulating them with monotone operators. However, in this context, the proximal operator is usually refered to as resolvent operator. Maximality of a monotone operator is important, as it guarantees well-definedness of the resolvent operator (cf. Lemma 2.39).

[3] Usually, this problem is considered in an infinite dimensional Hilbert space (or in even more general spaces).

2.2.1 Gradient descent

Gradient descent is a classical first-order optimization algorithm for minimizing convex and differentiable objective functions. Given a current iterate $x^n \in \mathbb{R}^N$ for $n \geq 0$ the next iterate $x^{n+1} \in \mathbb{R}^N$ is computed *explicitly* in direction of the negative gradient of the objective function at the current iterate. The negative gradient of the objective points "downhill". Therefore, for appropriate step sizes the function value reduces from x^n to x^{n+1}. The update step reads, in terms of a maximal monotone operator $T = \nabla f$ (for a convex and differentiable function f),

$$x^{n+1} = (\mathrm{id} - \lambda_n T)(x^n) \quad \text{or} \quad x^{n+1} = x^n - \lambda_n \nabla f(x^n)\,.$$

Sometimes the update step is also referred to as *forward step*. The gradient descent method converges for suitable choices of $\lambda_n > 0$. If the gradient of the objective function, ∇f, is Lipschitz continuous with Lipschitz constant $L \geq 0$, it is easy to show convergence of $(f(x^n))_{n\in\mathbb{N}}$ for $0 < \inf_n \lambda_n \leq \sup_n \lambda_n < 2/L$. See, for example, [Nes04].

Example 2.19. Let us consider a very simple example. Consider a vector $x \in \mathbb{R}^N$ and the discrete derivative operator $\nabla\colon \mathbb{R}^N \to \mathbb{R}^N$ as defined in Section 0.2. Moreover, let $z \in \mathbb{R}^N$. The goal is to find the minimum of the function $E\colon \mathbb{R}^N \to \mathbb{R}$, $E(x) = \frac{1}{2}\|x - z\|^2 + \frac{1}{2}\|\nabla x\|^2$. The gradient is easily computed as $\nabla E(x) = x - z + \nabla^\top \nabla x$ and the Lipschitz constant for ∇E is $L \leq 1 + 4$. Therefore convergence of the gradient descent method can be achieved for $\lambda \in (0, 2/L)$. We initialize with $x^0 \in \mathrm{dom}\, E$ and iterate

$$x^{n+1} = x^n - \lambda \nabla E(x^n) = x^n - \lambda(x^n - z + \nabla^\top \nabla x^n) = x^n + \lambda z - \lambda(\mathrm{id} + \nabla^\top \nabla)(x^n)\,.$$

In order to compute x^{n+1} only the current iterate x^n must be known. This is different to methods operating with backward (implicit) steps, which we consider next.

For a general maximal monotone operator the (classical) gradient descent method is not easily applicable[4]. In that case the classical algorithm for solving for $x \in \mathbb{R}^N$ with $0 \in T(x)$ is the proximal point algorithm.

2.2.2 Proximal point algorithm

The proximal point algorithm originates from the works of [Min62, Mar70] and became particularly powerful with the general convergence results for monotone operators by Rockafellar [Roc76]. Under the assumption of summable errors in the computation of the iterates, he proved weak convergence of the sequence of iterates to a zero of a monotone operator. For details we also refer the reader to the thesis of Eckstein [Eck89].

The formal update step of the proximal point algorithm for a maximal monotone operator T is

$$x^{n+1} \in (\mathrm{id} + \lambda_n T)^{-1}(x^n)\,, \tag{2.4}$$

where $(\mathrm{id} + \lambda_n T)^{-1}$ is the resolvent operator [Mor65]. The resolvent operator generalizes the proximity operator for a convex function (Section 2.1.5) to the context of monotone operators. The update step using the resolvent operator is sometimes denoted *backward step* or *implicit step*. In the special case where $T = \nabla f$ for a convex and differentiable function f the next iterate can be found by (the non-linear system of equations in the variable x)

$$x^{n+1} \quad \text{solves} \quad (\mathrm{id} + \lambda_n \nabla f)(x) = x^n\,.$$

[4]There are ways to deal with nondifferentiable functions in a kind of gradient descent framework. These methods are called subgradient descent (according to [Ber99] it first appeared in [Sho85]; see for example [Ber99, Sec. 6.3]).

Example 2.20. Let z, E, ∇ be as in Example 2.19. Then the proximal point algorithm for finding $0 = \nabla E(x)$ reads: Initialize $x^0 \in \operatorname{dom} E$ and iterate

$$x^{n+1} = x^n - \lambda \nabla E(x^{n+1}) = x^n - \lambda(x^{n+1} - z + \nabla^\top \nabla x^{n+1})$$

by solving the linear system of equations (w.r.t. x)

$$x^{n+1} \quad \text{solves} \quad (\mathrm{id} + \lambda(\mathrm{id} + \nabla^\top \nabla))(x) = x^n + \lambda z\,.$$

For this example the algorithm converges for any $\lambda > 0$. In fact even for $\lambda \to \infty$. If λ tends towards infinity (informally) the previous linear system of equation reduces to $x + \nabla^\top \nabla x = z$, which means solving the first order optimality condition of the original problem directly—for convex functions the global minimum is found.

Like in Example 2.20 solving the resolvent operator for obtaining the next iterate can be as hard as the original problem. Therefore it is interesting to find approximations to the proximal point algorithm for which the resolvent operator is easy to invert and for which the favorable properties of the proximal point algorithm still hold. A way to modify the proximal point algorithm such that it becomes applicable to many problems is by splitting methods.

2.2.3 Splitting methods

The starting point for splitting methods is the difficulty to use the proximal point algorithm in practice. The idea is to approximate the backward step from the proximal point algorithm with operations that combine the simplicity of forward (explicit) steps and the favorable convergence properties of backward (implicit) steps. A maximal monotone operator T is usually expressed as a sum of two operators A and B, i.e., $T = A + B$, where A and B are assumed to be "simple". Simple is meant in the sense that the resolvent operator is easy or efficient to compute, or a simple explicit step is performed.

Splitting algorithms for solving the problem of finding $x \in \mathbb{R}^N$ such that $0 \in (A + B)(x)$ combine forward and backward steps in different ways. It first appeared explicitly in the context of the proximal point algorithm in [LM79]. Examples are the Peaceman–Rachford splitting algorithm [PR55] and the Douglas–Rachford splitting algorithm [DR56], which show strong relations to the so-called alternating direction method of multipliers, double-backward, forward–backward, etc. There are many splitting algorithms and many relations among them. In the following, we focus on the so-called forward–backward splitting algorithms [LM79, Gol64, LP66, Bru77]

$$x^{n+1} = (\mathrm{id} + \lambda_n A)^{-1}(\mathrm{id} - \lambda_n B)(x^n)\,.$$

In [Pas79] a convergence result in the general setting of maximal monotone operators A, B, and $A + B$ is given. Under suitable conditions, he proves weak convergence of the sequence of weighted averages $z_n := \sum_{i=1}^n \lambda_i x^i / \sum_{i=1}^n \lambda_i$ to a zero of $A + B$. Convergence results for the forward–backward splitting algorithm significantly improve when B is a single-valued Lipschitz continuous operator. An example for such an operator is the gradient of a continuously differentiable function with Lipschitz continuous gradient. In this case, [Tse91, Gab83] show convergence to a zero of $A + B$ if $\lambda_n < 2/L$, where L is the Lipschitz constant of B. In the special setting where A is the subdifferential of an indicator function of a convex set—the proximal operator is equivalent to a projection—, the algorithm becomes the projected gradient descent method [LP66, Dun81, Gol64]. The forward–backward splitting algorithm is also very interesting for applications like sparse signal recovery [CW05, DDM04], image processing [RFP13], and machine learning [Sra12, DS09]. Before we step forward to generalizations of the forward–backward algorithm, we consider an example.

Example 2.21. Let z, E, ∇ be as in Example 2.19. We split $E = E_1 + E_2$ with $E_1(x) = \frac{1}{2}\|x - z\|^2$ and $E_2(x) = \frac{1}{2}\|\nabla x\|^2$. The monotone operators appearing in the update step of the forward–backward algorithm are $A = \nabla E_1$ and $B = \nabla E_2$. Then the algorithm for finding $0 = (A + B)(x)$ reads: Initialize $x^0 \in \operatorname{dom} E$ and iterate

$$x^{n+1} = x^n - \lambda(\nabla E_1(x^{n+1}) + \nabla E_2(x^n)) = x^n - \lambda(x^{n+1} - z + \nabla^\top \nabla x^n)\,,$$

which is equivalent to

$$x^{n+1} = \frac{(x^n - \lambda \nabla^\top \nabla x^n) + \lambda z}{1 + \lambda}\,.$$

For this example, convergence is achieved by choosing $\lambda \in (0, 2/L)$ where $L = 4$ is the Lipschitz constant of B.

2.2.4 Heavy-ball method

The gradient method is certainly one of the most fundamental but also one of the most simple algorithms to solve smooth convex optimization problems. In the last decades, the gradient method has been modified in many ways. One of those improvements is to consider so-called multi-step schemes [Pol64, Nes04]. It has been shown that such schemes significantly boost the performance of the plain gradient method. A particularly interesting multi-step scheme by Polyak [Pol64] is the following two-step method

$$x^{n+1} = x^n - \alpha \nabla f(x^n) + \beta(x^n - x^{n-1})\,.$$

It can be seen as a (time) discretization of its second order time continuous analogue, the so called *Heavy-ball with friction dynamical system*,

$$\ddot{x}(t) + \gamma \dot{x}(t) + \nabla f(x(t)) = 0\,,$$

which is an ordinary differential equation. It arises when Newton's law is applied to a point subject to a constant friction $\gamma > 0$ (of the velocity $\dot{x}(t)$) and a gravity potential f. This explains the naming *Heavy-ball method* and the interpretation of $\beta(x^n - x^{n-1})$ as *inertial force*. [Pol64] proves local convergence to a minimum of a function whose Hessian matrix is positive definite and bounded from above in a neighborhood of the (local) minimum. Moreover, he proves a local convergence rate of $q := (\sqrt{L} - \sqrt{l})/(\sqrt{L} + \sqrt{l})$ when setting $\alpha = 4/(\sqrt{L} + \sqrt{l})^2$ and $\beta = q^2$, where (in modern terminology) L is the (local) Lipschitz constant of the gradient and l is the (local) strong convexity parameter. Obviously, for a function that has (globally) a Lipschitz continuous gradient and is strongly convex, Polyak's result implies global convergence. Compared to the gradient descent method, which arises for $\beta = 0$, this is a significant improvement in the speed of convergence. Nowadays, it is known that the convergence rate of the Heavy-ball method is optimal [Nes04] for a first order algorithm with global convergence for all strongly convex functions with Lipschitz continuous gradient.

The conjugate gradient method reveals the same rate of convergence. In fact, the conjugate gradient method for minimizing quadratic problems can be expressed as Heavy-ball method. Therefore, it can be seen as a special case of the Heavy-ball method for quadratic problems. Nevertheless, there is an interesting difference. The conjugate gradient method does not require additional knowledge about the function parameters for such problems. They are computed online. However, the downside of the conjugate gradient method is that its nice theoretical properties do not generalize well to more general classes of functions, where the Heavy-ball method does generalize.

Like the modification of the gradient descent method to the forward–backward splitting algorithm, the Heavy-ball method can be modified in the same way. This leads us to the generalization of the projected Heavy-ball method

$$x^{n+1} = (I + \alpha\partial g)^{-1}(x^n - \alpha\nabla f(x^n) + \beta(x^n - x^{n-1}))\,, \tag{2.5}$$

where g is a (possibly nonsmooth) convex function. In Chapter 3 we investigate in detail the convergence rate of this algorithm. We show that $(x^n)_{n\in\mathbb{N}}$ converges with optimal linear rate.

In [AA01, Alv03], the Heavy-ball method was extended to maximal monotone operators. In a subsequent work [MO03], it has been applied to a forward–backward splitting algorithm, again in the general framework of maximal monotone operators. In terms of maximal monotone operators A and B, where B is single-valued and Lipschitz continuous, the update step for the proximal inertial Heavy-ball method can be written as:

$$\begin{aligned} y^n &= x^n + \beta(x^n - x^{n-1}) \\ x^{n+1} &= (\mathrm{id} + \alpha A)^{-1}(y^n - \alpha B(x^n)) \end{aligned}$$

Like for the forward–backward splitting algorithm, it can be shown that this method converges as long as $\alpha < 2/L$ and $\beta \in [0, 1)$, where L is the Lipschitz constant of B. The difference between this algorithm and the forward–backward splitting is the point y^n. For the method here B is still evaluated at the old point x^n. We do not show an example here, because the algorithm is as difficult to implement as the forward–backward splitting method.

Other modifications of the Heavy-ball method are the following. In [APR14] a time second-order dynamical system related to the Heavy-ball with friction system is studied. It is shown to be equivalent to a coupled first-order system that becomes an inertial forward–backward splitting method when it is discretized. The Heavy-ball method seems also appealing in the nonconvex setting. In [ZK93], it was generalized to smooth nonconvex functions and in [OCBP14] (see Chapter 6) to a class of structured nonsmooth nonconvex optimization problems. Modifications of the Heavy-ball method appear in the convex setting for instance in the popular accelerated gradient method of Nesterov [Nes04, Nes13] (see Section 2.2.5), where the difference is the computation of the gradient. While the Heavy-ball method uses the point from the preceding iteration, Nesterov's method computes the gradient at points that are extrapolated by the inertial force. On strongly convex functions, both methods are equally fast (up to constants), but Nesterov's accelerated gradient method converges much faster on convex functions [DT13].

2.2.5 Nesterov's method and Fista

The difference between the Heavy-ball method (2.5) and FISTA (fast iterative shrinkage–thresholding algorithm) [BT09a] is the evaluation of the gradient of the function. Where the Heavy-ball method uses the gradient at the old point, FISTA uses the gradient of the extrapolated point. The update scheme for FISTA reads

$$\begin{aligned} y^n &= x^n + \beta(x^n - x^{n-1}) \\ x^{n+1} &= (I + \alpha\partial g)^{-1}(y^n - \alpha\nabla f(y^n))\,. \end{aligned} \tag{2.6}$$

Different step size rules under different variants of this scheme were proposed [Nes83, Nes04, Tse08, BT09a]. A simpler version of this was proposed by Nesterov [Nes83] ($g \equiv 0$), which was the first method for the general class of smooth convex objective functions (with Lipschitz continuous gradient) that obeys a convergence according to $\mathcal{O}(1/n^2)$, where n is the iteration counter. For g proper convex Nesterov [Nes13] and Beck and Teboulle [BT09a] proposed algorithms with the same convergence rate.

Up to constants, the proved convergence rates for the residual of function values coincides with the lower complexity bound for this class of functions, as proved by Nesterov [Nes04]. There are also strong relations of their methods to so-called primal–dual approaches. See for example [LP14].

2.2.6 Primal–dual approach

Starting from the proximal point algorithm (2.4), we now derive the primal–dual algorithm by Chambolle and Pock [CP11, PC11]. In (2.4), we replace the step-size factors λ_n by a symmetric, positive definite matrix M, i.e.,

$$0 \in T(x^{n+1}) + M^{-1}(x^{n+1} - x^n)\,. \tag{2.7}$$

This matrix M can be considered as changing the metric that is used during the optimization. The primal–dual algorithm emerges when we set

$$T = \begin{pmatrix} \partial g(x^{n+1}) + K^* y^{n+1} \\ \partial f^*(y^{n+1}) - K x^{n+1} \end{pmatrix} \quad \text{and} \quad M = \begin{pmatrix} \mathcal{T}^{-1} & -K^* \\ -K & \Sigma^{-1} \end{pmatrix}, \tag{2.8}$$

where f, g are convex functions, f^* denotes the convex conjugate of f, K is a linear operator, K^* its adjoint operator, and $\mathcal{T}$, Σ are symmetric, positive definite matrices containing the step sizes. If the condition $\|\Sigma^{\frac{1}{2}} K \mathcal{T}^{\frac{1}{2}}\|^2 < 1$ is satisfied, then [PC11, Thm. 1] proves convergence of the algorithm (2.7), (2.8), i.e., $0 \in T(\hat{x}, \hat{y})$. Usually, $\mathcal{T}$ and Σ are chosen to be diagonal matrices, in order to keep evaluating the proximal operator simple. Plugging (2.8) into $0 \in T(\hat{x}, \hat{y})$ yields

$$0 \in \begin{pmatrix} \partial g(\hat{x}) + K^* \hat{y} \\ \partial f^*(\hat{y}) - K \hat{x} \end{pmatrix} \quad \text{or} \quad \begin{cases} -K^* \hat{y} & \in \partial g(\hat{x}) \\ K \hat{x} & \in \partial f^*(\hat{y})\,, \end{cases}$$

which is the first order optimality condition of the saddle point problem

$$\min_x \max_y \; g(x) + \langle Kx, y \rangle - f^*(y)\,.$$

Finally this saddle point problem is solved by the general primal–dual algorithm proposed in [CP11, PC11], which arises when combining (2.7) and (2.8):

$$\begin{aligned} x^{n+1} &= (\mathrm{id} + \mathcal{T}\partial g)^{-1}(x^n - \mathcal{T} K^* y^n) \\ y^{n+1} &= (\mathrm{id} + \Sigma \partial f^*)^{-1}(y^n + \Sigma K(2x^{n+1} - x^n))\,. \end{aligned} \tag{2.9}$$

Although there are several other primal–dual algorithms [EZC10, HY12a, Con13, CP12, Vũ13], we focus on (2.9) as it proved to be very efficient in the last years and will be used later in the thesis. It is particularly efficient in large scale applications from image processing.

Example 2.22. Let z, E, ∇ be as in Example 2.19. We set $g(x) = \frac{1}{2}\|x - z\|^2$ and $f(\nabla x) = \frac{1}{2}\|\nabla x\|^2$ and use $f(\nabla x) = f^{**}(\nabla x) = \max_y \langle \nabla x, y \rangle - \frac{1}{2}\|y\|^2$. The step size parameter matrices $\mathcal{T}$ and Σ can be chosen as diagonal matrices with the constant diagonal τ and σ, respectively. Then, the step size restriction is given by the condition $\sigma\tau L^2 < 1$ where L is the operator norm of $K = \nabla$, which is $L = \sqrt{4}$. Then, the update step of the primal–dual algorithm reads

$$\begin{aligned} x^{n+1} &= \frac{(x^n - \tau \nabla^\top y^n) + \tau z}{1 + \tau} \\ y^{n+1} &= \frac{y^n + \sigma \nabla (2x^{n+1} - x^n)}{1 + \sigma}\,. \end{aligned}$$

Chapter 3

iPiasco: inertial proximal algorithm for strongly convex optimization

In this chapter, we present a Heavy-ball method (forward–backward splitting algorithm with additional inertial term; see Section 2.2.4) for solving a strongly convex optimization problem of the form

$$\min_{x \in \mathbb{R}^N} \; f(x) + g(x)\,,$$

in the Euclidean vector space $\mathbb{R}^N$ of dimension $N \in \mathbb{N}$. The objective function is composed of a smooth, convex function f and a nonsmooth, convex function g. Additionally, we assume that the objective function is strongly convex. On one hand, these assumptions are very restrictive, however, on the other hand, this class of objective functions can be optimized very efficiently. As we will see in Chapter 7 such problems can arise as subproblems for solving nonconvex problems. The knowledge about the structure of the problem is used for deriving a convergence rate for the proposed algorithm. It is proved to be an optimal algorithm with linear rate of convergence. For certain problems this linear rate of convergence is better than the provably optimal rate of convergence for smooth strongly convex functions.

In the general case, the rate only depends on the Lipschitz constant $L > 0$ of ∇f, the strong convexity parameter $l > 0$ of f and the strong convexity parameter $m > 0$ of g. When $g = \delta_C$ or $m = 0$, this rate coincides with the optimal rate found for the projected Heavy-ball method. Summarizing, the additional proximal step does not degrade the known convergence rates.

A particularly interesting situation arises, when g is also smooth and has Lipschitz continuous gradient ∇g with constant $M > 0$. In this case, the function $f + g$ may be split into f and g such that terms with high Lipschitz constant enter g. We demonstrate that in some situations an appropriate splitting leads to a convergence rate that is better than the optimal lower bound. This is possible thanks to the well defined structure of our optimization problem. The optimal lower bound is only valid for black-box algorithms.

A numerical experiment shows and compares the efficiency of the proposed algorithm in practice with other optimal methods. Finally, we apply iPiasco to the computer vision problems of denoising and inpainting and compare its convergence against several other methods, e.g., the projected Heavy-ball method. Large parts of this chapter are presented in [OBP15].

3.1 Related work

In [Nes04], Nesterov derives lower efficiency bounds for first-order black-box optimization on different classes of objective functions. If the objective function is nonsmooth, it is shown that the lower bound on the convergence rate is $\mathcal{O}(1/\sqrt{n})$. This bound is actually attained by the subgradient method [Sho85], which is certainly the most simple algorithm for nonsmooth optimization. In [Nes05], it was shown that if the nonsmooth objective function has a certain saddle-point structure (in contrast to black box optimization), the convergence rate can be improved via smoothing to $\mathcal{O}(1/n)$. Algorithms that achieve this rate are for example [Nes05, CP11, DDM04, Nes13, GM09, HY12b].

If the objective function is smooth (i.e. it has a Lipschitz continuous gradient) the lower efficiency bound is $\mathcal{O}(1/n^2)$. The first algorithm that achieved this rate was presented by Nesterov in [Nes83] and generalized to composite objective functions or saddle-point problems in [CP11, BT09a, Nes83, Nes13, GM09]. Finally, for the class of smooth and strongly convex functions, the lower efficiency bound is given by $\mathcal{O}(\omega^n)$, where $\omega \in (0,1)$ depends on the square root of the condition number of the objective function. Algorithms that converge with the same rate are for example [CP11, Pol64, BDF07, HL12].

Related work of forward–backward splitting algorithms and the Heavy-ball method were already discussed in Section 2.2.4.

3.2 The proposed algorithm

The objective function $h\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $h := f + g$, is assumed to be proper lower semi-continuous (lsc), extended valued and bounded from below by some value $\underline{h} > -\infty$. Moreover, let the objective be composed of a proper lsc convex (possibly nonsmooth) function $g\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ and a proper convex function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ that is twice continuously differentiable on $\operatorname{dom} g$ and has Lipschitz continuous gradient on $\operatorname{dom} g$; and let f or g be strongly convex.

We propose Algorithm 1 including the definition of the step size parameters, which emerge from the subsequent convergence analysis.

Algorithm 1. ***I**nertial **p**roxi**m**al **a**lgorithm for **s**trongly **c**onvex **o**ptimization (iPiasco)*

- *Optimization problem:* $\min_{x\in\mathbb{R}^N} h(x) = \min_{x\in\mathbb{R}^N} f(x) + g(x)$ *with*

$$f \in \mathscr{S}^{2,1}_{l,L}(\operatorname{dom} g),\ L \geq l \geq 0\,, \quad g \in \mathscr{S}_m,\ m \geq 0, \quad m + l > 0\,.$$

- *Step-size parameter: Define* $\alpha > 0$ *and* $\beta \in [0,1)$ *by*

$$\alpha = \frac{4}{(\sqrt{l+m} + \sqrt{L+m})^2 - 4m},\ \beta = \frac{(\sqrt{m+L} - \sqrt{m+l})^2}{(\sqrt{m+L} + \sqrt{m+l})^2 - 4m}$$

- *Initialization: Choose a starting point* $x^0 \in \operatorname{dom} h$ *and set* $x^{-1} = x^0$.

- *Iterations* $(n \geq 0)$*: Update*

$$x^{n+1} = (\mathrm{id} + \alpha\partial g)^{-1}(x^n - \alpha\nabla f(x^n) + \beta(x^n - x^{n-1}))\,. \tag{3.1}$$

3.2.1 Preliminaries

In order to derive the convergence result in this chapter, we need a few results beyonds those of Chapter 2. The *spectral radius* of a matrix T, which is defined as the maximal magnitude of its eigenvalues

$$\rho(T) := \max\{|\lambda| \,|\, \lambda \text{ is eigenvalue of } T\}\,,$$

is of importance. When the iteration matrix of an algorithm has only eigenvalues smaller than 1, i.e., the spectral radius is less than 1, the sequence induced by this iteration matrix converges. Moreover, the matrix norm reveals an asymptotic relationship to the spectral radius of the matrix. This is an important result from linear algebra (see, for example, [Pol87, Sec. 2.1]), which was originally proved by Gelfand [Gel41].

Lemma 3.1 (Gelfand's formula)**.** *It holds that $\rho(A) = \lim_{n\to\infty} \|A^n\|^{1/n}$, i.e., the spectral radius of A gives the asymptotic growth rate of $\|A^n\|$: For every $\varepsilon > 0$ there is $c = c(\varepsilon)$ such that $\|A^n\| \leq c(\rho(A)+\varepsilon)^n$ for all $n \in \mathbb{N}$.*

3.2.2 Convergence analysis

Let us now analyze Algorithm 1.

Lemma 3.2. *Let $a, b \in \mathbb{R}$. If*

$$(1 - \sqrt{1-a})^2 \leq b \leq (1 + \sqrt{1-a})^2\,, \tag{3.2}$$

then the matrix

$$T := \begin{pmatrix} a+b & -b \\ 1 & 0 \end{pmatrix}$$

has two complex eigenvalues with squared magnitude b.

Proof. In order to obtain the eigenvalues of T, we compute the roots of the characteristic polynomial $v^2 - (a+b)v + b$ in the variable v. The eigenvalues v_1, v_2 are

$$v_{1,2} = \frac{a+b}{2} \pm \sqrt{\frac{(a+b)^2}{4} - b}\,.$$

Now, we try to find the condition for obtaining complex roots. The discriminant of the quadratic characteristic polynomial needs to be negative. A few elementary transformation steps yield the condition $(b + (a-2))^2 \leq 4 - 4a$, which is equivalent to (3.2). Under the assumption that this inequality holds T has two complex roots with squared magnitude

$$|v_{1,2}|^2 = v_{1,2}\bar{v}_{1,2} = \left(\tfrac{a+b}{2} \pm \sqrt{\tfrac{(a+b)^2}{4} - b}\right)\left(\tfrac{a+b}{2} \mp \sqrt{\tfrac{(a+b)^2}{4} - b}\right) = b\,,$$

which concludes the proof, where $\bar{v}_{1,2}$ denotes the complex conjugate. □

Now, we are equipped to start with the analysis of the convergence rate of iPiasco as proposed in Algorithm 1. For notational convenience, we define $z^n := (x^n - x^*, x^{n-1} - x^*)^\top$.

Theorem 3.3. *Let the functions f and g with parameter l, L, and m; and the step size parameter α and β be as in Algorithm 1. Moreover, let $(x_n)_{n\in\mathbb{N}}$ be generated by Algorithm 1 and $x^* := \lim_{n\to\infty} x^n$ be the unique global optimum. Then, for every $\varepsilon > 0$ there is $c = c(\varepsilon)$ such that*

$$\|z^{n+1}\| \leq c(q+\varepsilon)^n \|z^n\| \quad \textit{for all } n \in \mathbb{N}\,, \quad \textit{where } q = \frac{\sqrt{m+L}-\sqrt{m+l}}{\sqrt{m+L}+\sqrt{m+l}}\,.$$

Proof. First, we note that $x^* \in \operatorname{dom} h$ exists and it is unique due to the properties of h. As x^* is a stationary point of h, hence $0 \in \partial h(x^*)$, x^* is a fixed point of $(\mathrm{id} + \alpha\partial g)^{-1} \circ (I - \alpha\nabla f)$ ([BC11, Proposition 25.1(iv)]), i.e.,

$$x^* = (\mathrm{id} + \alpha\partial g)^{-1}(x^* - \alpha\nabla f(x^*) + \beta(x^* - x^*))\,.$$

Combining and (6.2) with the $(1+\alpha m)^{-1}$-Lipschitz continuity of the (strongly) convex function αg from (2.2), we observe

$$\|x^{n+1} - x^*\| \leq \tilde{m}_\alpha \|(1+\beta)(x^n - x^*) - \alpha(\nabla f(x^n) - \nabla f(x^*)) - \beta(x^{n-1} - x^*)\|\,,$$

where $\tilde{m}_\alpha := (1+\alpha m)^{-1}$. As f is twice continuously differentiable and $\operatorname{dom} f$ is convex, there exists a matrix B such that the following mean value theorem holds

$$\nabla f(x^n) - \nabla f(x^*) = \left(\int_0^1 B_t\, dt\right)(x^n - x^*), \quad B_t := \nabla^2 f(x^n + t(x^* - x^n))\,,$$

where $\nabla^2 f$ denotes the second derivative of f. Plugging both results together, we conclude

$$\|z^{n+1}\| \leq \sup_{t\in[0,1]} \|A_t z^n\|, \quad \text{where } A_t := \begin{pmatrix} \tilde{m}_\alpha((1+\beta)I - \alpha B_t) & -\beta I \tilde{m}_\alpha \\ I & 0 \end{pmatrix}.$$

Now, we analyze the spectral radius of the matrix A_t. As the decisive fact is that the eigenvalues of B_t are in $[l, L]$, which is independent of t, from now on we drop the subscript t. Denoting the eigenvalues of B by λ_i, $i = 1, \ldots, N$, it is not too difficult to show (using the right permutation matrix) that A is similar to a block diagonal matrix with blocks of size 2×2 of the form

$$T_\lambda := \begin{pmatrix} \tilde{m}_\alpha(1+\beta-\alpha\lambda) & -\tilde{m}_\alpha\beta \\ 1 & 0 \end{pmatrix},$$

where from now on λ stands for one of the eigenvalues λ_i. Consequently, the spectral radius of A is given by the maximal spectral radius of T_λ for all λ.

Under the assumption that the spectral radius of A is less than 1, we can show the statement of convergence using Lemma 3.1.

It remains to show that the setting of α and β yields $\rho(A) < 1$. If we set $a = \tilde{m}_\alpha(1-\alpha\lambda)$ and $b = \tilde{m}_\alpha\beta$ in Lemma 3.2, then the right inequality in (3.2) is trivially satisfied as $b < 1$. The left inequality in (3.2) requires $\rho(T_\lambda) = \sqrt{\tilde{m}_\alpha\beta} \geq |1 - \sqrt{1+\alpha\lambda\tilde{m}_\alpha - \tilde{m}_\alpha}|$. Now, the next step is to determine α and β such that (3.2) is met and the spectral radius becomes minimal.

Since $\lambda \in [l, L]$, we have

$$(1 - \sqrt{1+\alpha\lambda\tilde{m}_\alpha - \tilde{m}_\alpha})^2 \leq \max\{(1 - \sqrt{1+\alpha\tilde{m}_\alpha l - \tilde{m}_\alpha})^2, (1 - \sqrt{1+\alpha\tilde{m}_\alpha L - \tilde{m}_\alpha})^2\}\,.$$

Hence, we determine α such that the right hand side is minimal[1]. Then, setting β such that $\tilde{m}_\alpha\beta$ equals the right hand side (using the determined α) yields the best estimate for the convergence rate $q = \sqrt{\tilde{m}_\alpha\beta}$.

[1] In general, this is analytically very challenging because $\tilde{m}_\alpha$ also depends on α.

For notational convenience, we define

$$\varrho_{l,m}(\alpha) := 1 - \sqrt{1 - \frac{1-\alpha l}{1+\alpha m}}\,.$$

Then, we have to determine $\alpha > 0$ such that $(\varrho_{l,m}(\alpha))^2 = (\varrho_{L,m}(\alpha))^2$. It can be easily seen that $\varrho_{l,m}(\alpha) = \varrho_{L,m}(\alpha)$ is solved by $\alpha = 0$, which is not feasible. The expression $\varrho_{l,m}(\alpha) = -\varrho_{L,m}(\alpha)$ is equivalent to (note that $\alpha > 0$ and $1 + \alpha m > 0$)

$$2\sqrt{1+\alpha m} = \sqrt{\alpha}(\sqrt{l+m} + \sqrt{L+m})\,.$$

As both sides are positive, squaring and solving for α proves the expression for α. Then plugging-in the computed α verifies the term for β. □

Theorem 3.3 covers several special cases, which are interesting in their own right. In the following, we state some of these special cases. The proofs directly follow from Theorem 3.3 by plugging-in the parameters and some simple and brief calculation.

First we discuss the trivial case $L = l$. This yields $q = 0$ and the problem can be solved with a single iteration. The step size parameters are $\alpha = 1/L$ and $\beta = 0$. The function f is a simple quadratic function with circular level sets; The Lipschitz upper bound is exact. Therefore, the minimization problem

$$x^{n+1} = \arg\min_x g(x) + \frac{L}{2}\|x - (x^n - \tfrac{1}{L}\nabla f(x^n))\|^2\,,$$

which is an equivalent representation of a single iteration step of iPiasco (6.2), coincides with the original problem. This formulation shows that after the quadratic function f is minimized (by the step $x^n - \frac{1}{L}\nabla f(x^n)$) evaluating the proximal term solves the whole problem.

If, additionally, $m = 0$, then a single gradient descent step with step size $1/L$ solves the problem.

In the following, we consider the special case $m = 0$ and $l \leq L$. From [Nes04], the Heavy-ball method is known as an optimal method for the class of strongly convex and two times continuously differentiable functions with Lipschitz continuous gradient. Its convergence rate coincides with the estimated lower bound from Nesterov. The next corollary reveals that adding a nonsmooth convex function g to the function f iPiasco still shows the same convergence rate as the Heavy-ball method. Hence, in this sense, iPiasco is optimal for the class of strongly convex (possibly nonsmooth) functions.

Corollary 3.4. *Let the same assumptions hold as in Theorem 3.3, but with $l > 0$ (f strongly convex) and $m = 0$ (g only convex). The quantities α, β, q in Algorithm 1 simplify to*

$$\alpha = \frac{4}{(\sqrt{L}+\sqrt{l})^2}, \quad \beta = \left(\frac{\sqrt{L}-\sqrt{l}}{\sqrt{L}+\sqrt{l}}\right)^2, \quad \textit{and} \quad q = \sqrt{\beta}\,.$$

Then, for every $\varepsilon > 0$ there is $c = c(\varepsilon)$ such that $\|z^{n+1}\| \leq c(q+\varepsilon)^n\|z^n\|$ for all $n \in \mathbb{N}$.

Remark 3.1. Denoting $Q_f := L/l$ as the ratio between the Lipschitz constant for ∇f and the strong convexity parameter for f, we can rewrite the spectral radius of the iteration matrix, denoted T here, for Corollary 3.4 as

$$\rho(T) = \frac{\sqrt{Q_f} - 1}{\sqrt{Q_f} + 1}\,,$$

which establishes a relation between the conditioning of the objetive function and the convergence rate.

Remark 3.2. The convergence rate of the Heavy-ball method is equivalent to the rate of the conjugate gradient (CG) method. An advantage of CG is that it automatically determines the parameter α and β in each iteration. On the other hand, unlike iPiasco, CG cannot handle an additional nonsmooth convex term in the objective function.

There is also an interesting new variant of Algorithm 1, where only the function g is required to be strongly convex. The convergence rate can benefit from an additional strongly convex term g.

Corollary 3.5. *Let the same assumptions hold as in Theorem 3.3, but let only g be strongly convex and f be convex, i.e., $m > 0$, $l = 0$, and $L > 0$. The quantities α, β, q in Algorithm 1 simplify to*

$$\alpha = \frac{4}{(\sqrt{m} + \sqrt{L+m})^2 - 4m}, \quad \beta = \frac{(\sqrt{m+L} - \sqrt{m})^2}{(\sqrt{m+L} + \sqrt{m})^2 - 4m},$$
$$\text{and} \quad q = \frac{\sqrt{m+L} - \sqrt{m}}{\sqrt{m+L} + \sqrt{m}}.$$

Then, for every $\varepsilon > 0$ there is $c = c(\varepsilon)$ such that $\|z^{n+1}\| \leq c(q+\varepsilon)^n \|z^n\|$ for all $n \in \mathbb{N}$.

Summarizing, we obtain optimal linear convergence for iPiasco in the case, where f is convex and two times continuously differentiable, g is convex, and any of the two functions f or g is strongly convex, additionally.

Let us discuss the results that we obtained so far more in detail. For a moment, suppose the function $g \in \mathscr{S}^{1,1}_{m,M}$ with convexity parameter $m > 0$ and Lipschitz constant $M > 0$. Then, the function h is strongly convex, continuously differentiable and has a Lipschitz continuous gradient. The lower complexity bound from [Nes04] reads

$$q(l, L, m, M) = \frac{\sqrt{L+M} - \sqrt{l+m}}{\sqrt{L+M} + \sqrt{l+m}},$$

which is increasing whenever M is increasing ($\lim_{M\to\infty} q(l, L, m, M) = 1$). As the convergence rate for iPiasco, a value in $(0, 1)$, is independent of the Lipschitz constant of ∇g, a good decomposition moves the term causing a high Lipschitz constant into the function g. It is clear that there is a certain value M for which the convergence rate of iPiasco outperforms the rate of the Heavy-ball method, see Figure 3.1. In other words: *iPiasco can beat the theoretical lower complexity bound from Nesterov [Nes04] for strongly convex and twice differentiable functions with Lipschitz continuous gradient*.

At first glance, this result seems to be an error, as it is impossible to outperform the theoretical lower complexity bound. However, this bound holds true only for black-box algorithms. Here, we seemingly very efficiently explore the composition of two functions contributing good properties to the objective. Section 3.2.3.2 shows a practical example where this fact improves the performance.

3.2.3 Numerical analysis

Now, we consider the behavior of iPiasco when it is applied to Nesterov's worst-case function for smooth and strongly convex problems [Nes04]. Let $Q_h := L_h/l_h > 1$ be the condition number for the function h with $L_h > 0$, the Lipschitz constant of ∇h, and $l_h > 0$, the convexity parameter. Then, the worst-case function from Nesterov reads

$$h_{\text{wc}}(x) := \frac{l_h(Q_h - 1)}{8}\left((x_1)^2 + \sum_{i=1}^{\infty}(x_i - x_{i+1})^2 - 2x_1\right) + \frac{l_h}{2}\|x\|_2^2, \tag{3.3}$$

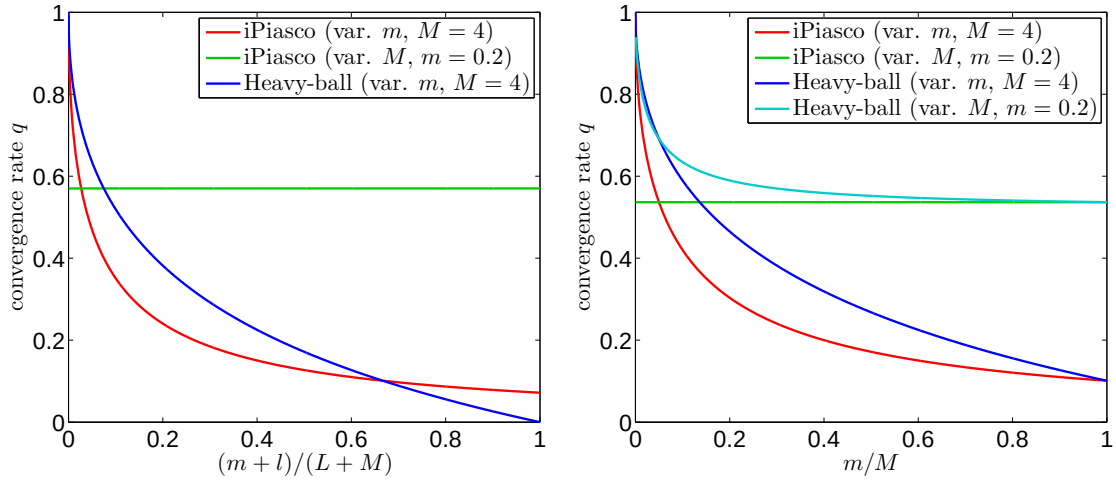


Figure 3.1: Convergence rates q for strongly convex and differentiable objectives with Lipschitz continuous gradient. Lower values of q are better. Let $f \in \mathscr{S}^{2,1}_{0,2}$, i.e., $l = 0$ and $L = 2$. iPiasco makes use of the splitting $h = f + g$, $g \in \mathscr{S}^{1,1}_{m,M}$, where the parameters M and m are adapted according to the x-axis, i.e., the (inverse) condition number of h (left plot) and the (inverse) condition number of g (right plot), respectively. Harder problems are closer to the left. In the case, where M is fixed and m tends to 0, the convergence rates tend to 1, because h tends to be not strongly convex. On the other hand, when m is fixed and M is high iPiasco is applicable, whereas the Heavy-ball method's rate tends to 1 (cyan line in the right plot), because the Lipschitz constant becomes unbounded. Observe, that the convergence rate for iPiasco is independent of M. Finally, for any condition numbers of g not equal to 0 or 1 the convergence rate of iPiasco outperforms the Heavy-ball method. This is impressive, as the Heavy-ball method coincides with the theoretical lower bound for this class of functions proved by Nesterov [Nes04] for black-box algorithms.

where we consider here $\|\cdot\|_2 = \|\cdot\|_{\ell_2}$, the norm in the space of infinite sequences. Figure 3.2 presents a comparison of iPiasco to the Heavy-ball method and the conjugate gradient method based on this function. The splitting of iPiasco is by setting $g = \frac{l_h}{2}\|x\|_2^2$ and $f = h - g$. The resulting parameters for estimating the convergence rate are $L = L_h$, $m = l_h$, and $l = 0$. The parameters for the Heavy-ball method (as a special instance of iPiasco) are $m = 0$, $L = L_h$, and $l = l_h$.

Since our algorithm can basically deal with the same class of composite objective functions as for example studied in [Nes13], iPiasco can be seen as an improvement of Nesterov's optimal gradient method for minimizing strongly convex composite objective functions.

3.2.3.1 The dual Huber–ROF model

In this subsection, we consider the dual problem of the Huber norm regularized variant of the Rudin–Osher–Fatemi model [ROF92] for image denoising. We define the decomposition

$$f(p) = \frac{1}{2}\|\nabla^\top p - \lambda u^0\|_2^2, \quad g(p) = \frac{\varepsilon}{2}\|p\|_2^2 + \delta_{\{\|p\|_\infty \leq 1\}}(p)\,, \tag{3.4}$$

where $p \in \mathbb{R}^{2N}$ is the dual vector for the ROF problem, $u^0 \in \mathbb{R}^N$ is the noisy input image, $\lambda, \varepsilon > 0$, $\delta_{\{\|p\|_\infty \leq 1\}}$ denotes the indicator function of the set $\{p \in \mathbb{R}^{2N} | \, \|p\|_\infty \leq 1\}$, and $\nabla^\top$ denotes the transposed of the gradient operator. Obviously, this model is strongly convex with modulus ε. In our decomposition, g is strongly convex and f is two times continuously differentiable. The decomposition defined above is suitable for iPiasco, which yields an optimal convergence rate for this class of problems. If we shift the quadratic penalty term $\frac{\varepsilon}{2}\|p\|_2^2$ from function g to the function f, we obtain another variant

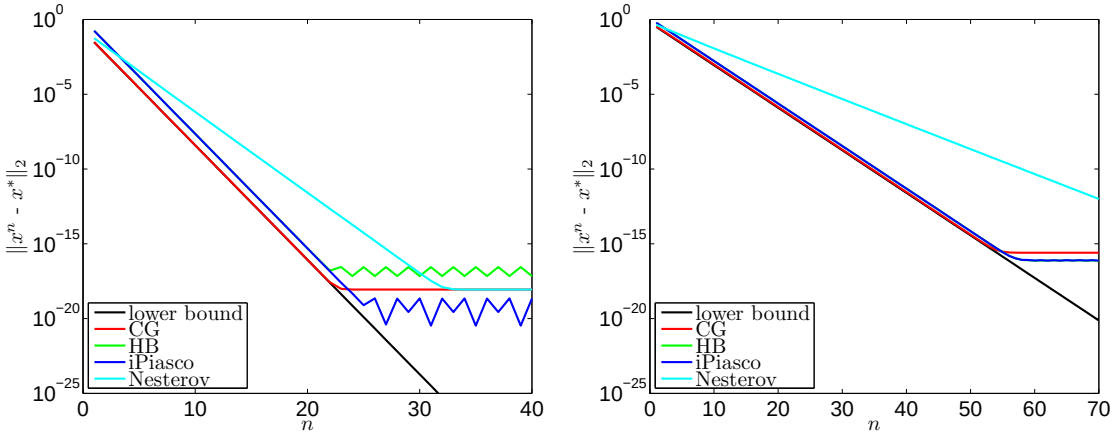


Figure 3.2: Comparison of the linear convergence rates on Nesterov's worst-case function (3.3) for smooth, strongly convex problems using a problem size of "$\infty \approx 100000$". LEFT: $l_h/L_h = 0.5$. RIGHT: $l_h/L_h = 0.1$. As predicted by theory, the conjugate gradient (CG) and the heavy ball (HB) method coincide with the theoretical lower bound. For larger values of l_h/L_h, iPiasco is a bit worse compared to CG and HB, but for smaller values of l_h/L_h (harder problems), iPiasco basically coincides with CG and HB. See also Figure 3.1. Note that Nesterov's optimal method is significantly slower.

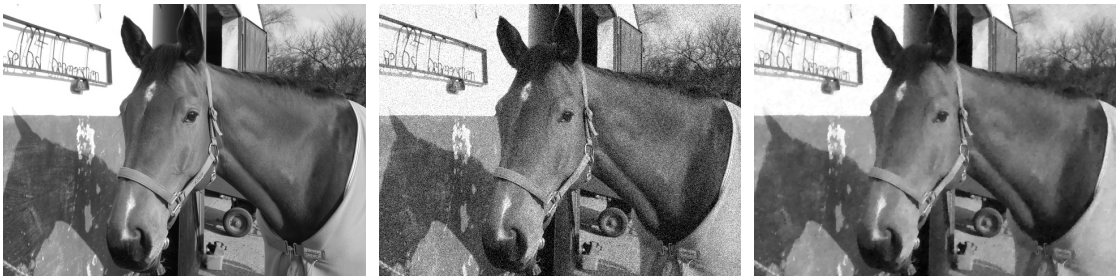


Figure 3.3: A denoising example with the dual Huber–ROF model (3.4). LEFT: Clean image. MIDDLE: Noisy image. RIGHT: Solution using (3.4). The objective function is strongly convex and the result on the right is obtained using our optimal method for this class of problems.

of iPiasco, which for the sake of discrimination is denoted iPiasco (projected). There is no best choice for decomposing the dual Huber–ROF model. This is explained by the term $\frac{\varepsilon}{2}\|p\|_2^2$: the Lipschitz constant and the convexity parameter are the same (cf. Figure 3.1).

We compare our method against the optimal methods: TwIST [BDF07] and Primal–Dual [CP14] (in this work a slightly better convergence rate for the primal–dual algorithm in [CP11] for strongly convex and smooth problems is proved). Moreover, we compare against the linearly converging Algorithm 2.2.11 from [Nes04], and the (regarding the convergence rate for this problem) suboptimal methods FISTA [BT09a], a very recent algorithm with unknown convergence rate IFB [APR14] (parameters: $a_{\text{IFB}} = b_{\text{IFB}} = L/4$ and $\lambda_{\text{IFB}} = 4/L$; L is the Lipschitz constant of ∇f), the primal–dual Algorithm 3.5 from [CDV10] (denoted Primal–Dual-CDV10, with parameters: $\gamma_{\text{CDV10}} = 2/L$ and $\lambda_{\text{CDV10}} = 0.75$), and the proximal forward–backward splitting method in [CW05] (denoted ProxFB with parameter $\gamma_{\text{ProxFB}} = 1.99/L$ and $\lambda_{\text{ProxFB}} = 0.99$). Note that the differences in the computational cost between all methods for one iteration are negligible.

In the following experiment we set $\lambda = 0.05$ and $\varepsilon = 0.1$. See Figure 3.3 for the result of this problem. The theoretical estimates for the linear convergence rates are 0.8 for iPiasco, iPiasco (proj.), and TwIST,

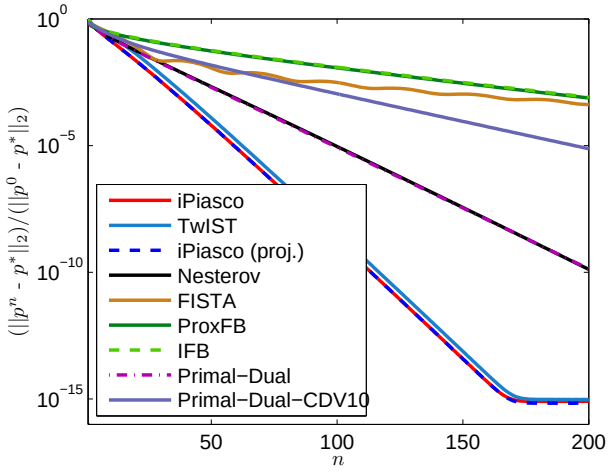


Figure 3.4: The three optimal methods iPiasco, iPiasco (proj.), and TwIST perform best on the dual Huber–ROF model (3.4).

	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
iPiasco	25	48	71	92	114	135	157
iPiasco (proj.)	25	48	71	92	114	135	157
TwIST	28	52	74	96	117	139	160
Primal-Dual	36	78	120	162	dnc	dnc	dnc
Nesterov	36	79	121	162	dnc	dnc	dnc
Primal-Dual-CDV10	58	148	dnc	dnc	dnc	dnc	dnc
FISTA	52	dnc	dnc	dnc	dnc	dnc	dnc
ProxFB	106	dnc	dnc	dnc	dnc	dnc	dnc
IFB	108	dnc	dnc	dnc	dnc	dnc	dnc

Table 3.1: Table corresponding to Figure 3.4. The first row shows error thresholds (for the normalized error). The entries in the table show the number of required iterations to fall below the respective error threshold. "dnc" means that the threshold was not reached within 200 iterations. The optimal methods iPiasco, TwIST, and iPiasco (proj.) clearly perform best.

0.943 for Nesterov's algorithm, and 0.894 for Primal–Dual [CP14]. Figure 3.4 and Table 3.1 show that the convergence of iPiasco (proj.), iPiasco, and TwIST are practically the same. This result also highlights the importance of (optimal) convergence rates. The linearly converging algorithm from Nesterov and Primal–Dual [CP14] perform equally well, but significantly worse than iPiasco. The methods FISTA, IFB, and Primal–Dual–CDV10 from [CDV10] are clearly outperformed; They have suboptimal rates of convergence.

3.2.3.2 An inpainting problem

Recently, [MW09] has shown that inpainting by a linear diffusion model is able to compete with jpeg for lossy image compression, particularly for cartoon like images. We consider an approximation of their reconstruction step defined by the decomposition

$$f(u) = \frac{1}{2}\|\nabla u\|_2^2, \quad g(u) = \frac{1}{2}\lambda\|c \cdot (u - u^0)\|_2^2 + \frac{1}{2}\varepsilon\|u\|_2^2\,, \tag{3.5}$$

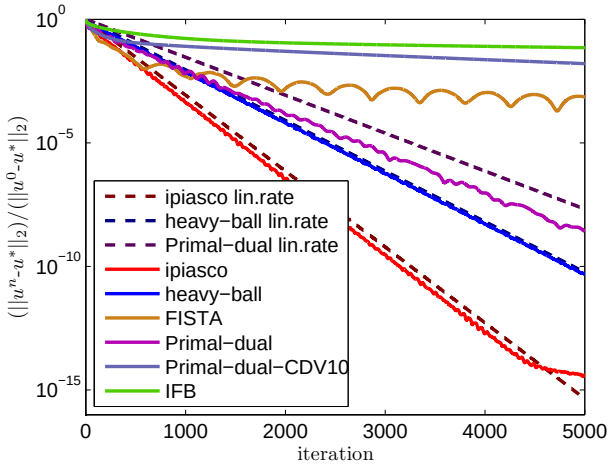


Figure 3.5: Convergence of the Heavy-ball method and iPiasco for the inpainting problem (3.5). The theoretical and practical convergence of iPiasco is much better then the convergence of the Heavy-ball method. Note that the convergence rate of the Heavy-ball method coincides with the optimal convergence rate determined by Nesterov [Nes04] for the general class of smooth strongly convex objectives. Since we exploit the structure of the inpainting problem very well, we can achieve a better convergence rate. Several other methods are clearly outperformed.

	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
iPiasco	553	1202	1850	2496	3148	3801	4459
Heavy-ball	924	1909	2880	3851	4838	dnc	dnc
Primal-dual	951	2122	3390	4595	dnc	dnc	dnc
FISTA	578	dnc	dnc	dnc	dnc	dnc	dnc
Primal-dual-CDV10	dnc	dnc	dnc	dnc	dnc	dnc	dnc
IFB	dnc	dnc	dnc	dnc	dnc	dnc	dnc
ProxFB	dnc	dnc	dnc	dnc	dnc	dnc	dnc

Table 3.2: Table corresponding to Figure 3.5. The first row shows error thresholds (for the normalized error). The entries in the table show the number of required iterations to fall below the respective error threshold. "dnc" means that the threshold was not reached within 5000 iterations. Our method iPiasco clearly outperforms all other methods.

where "$\cdot$" denotes the coordinate-wise product, $u, c, u^0 \in \mathbb{R}^N$, and $c_i = 1$ if pixel i of the original image u_i^0 was stored and $c_i = 0$ otherwise, i.e. the original pixel value u_i^0 is known only if $c_i = 1$. The linear operator implements a discrete gradient operator. The positive parameter λ allows for denoising the original pixel values, and $\varepsilon > 0$ is a small numerical number to make the problem strongly convex. This is a very difficult problem, which is related to Nesterov's worst-case function.

Using the proposed decomposition (3.5) the Lipschitz constant of the gradient of f is $L = 8$, the strong convexity parameter is $l = 0$, for g the Lipschitz constant is $M = \lambda + \varepsilon$ and the strong convexity parameter is $m = \varepsilon$. As the (inverse) condition number of g is smaller than 1, the convergence rate of iPiasco is expected to be better than the optimal rate of the Heavy-ball method (cf. Figure 3.1). In fact by setting $\varepsilon = 10^{-4}$, $\lambda = 10$ the convergence rate for the Heavy-ball method is 0.995297, whereas the rate for our iPiasco is 0.992954. The convergence of both methods is shown in Figure 3.5 and Table 3.2, together with the several other methods mentioned in the following. The inpainting result is shown in Figure 3.6. We compare also against other methods with no known or worse convergence rate. We evaluate the linearly converging primal–dual algorithm (Primal–Dual) from [CP14], FISTA [BT09a], IFB [APR14], the primal–dual algorithm (Primal–Dual–CDV10) from [CDV10], and the proximal forward–backward split-

Figure 3.6: An inpainting example [Veg], like in the reconstruction step of the image compression method [MW09]. LEFT: Original image to be stored. MIDDLE: Coordinates (black pixels) where the original gray values are stored. RIGHT: Reconstruction computed using iPiasco applied to (3.5).

ting method (ProxFB) from [CW05] with the same parameters as in the preceding subsection. Figure 3.5 does not show ProxFB as it performed almost identical to IFB.

Not only the theoretical rate of convergence for iPiasco is better than the one for the Heavy-ball method and the other methods, but also the practical convergence is significantly better. Note that increasing the parameter λ the convergence rate for the Heavy-ball becomes worse, whereas the convergence rate of iPiasco is independent of λ. For $\lambda \to \infty$ the Heavy-ball method can only be applied by explicitly handling the constraints $u_i = u_i^0$ for $c_i = 1$ as boundary conditions in the gradient operator, whereas iPiasco can be applied directly.

Chapter 4

Basics of variational analysis

The term variational analysis as it is used in the title of this chapter is not to be confused with "calculus of variation". The calculus of variation builds around the minimization of functionals in infinite dimensional spaces. Solutions to such a variational problem have the characteristic that the rate of change of the functional, the directional derivative, is zero at a solution. In simple words the goal is to find a point, for which the change of the functional is zero, when the point is moved infinitesimally (perturbation or variation of the point). In finite dimensional spaces these are points where the derivative vanishes. A big portion of the difficulties in the calculus of variation arises from constraints, which restrict the variation of a point to feasible "directions". Usually constraints are such that points can be varied along straight lines or, at least, along differentiable curves.

We allow for more complicated types of variation. Let us consider an example. In this chapter we introduce a notion of subgradients that generalizes the subgradient known from convex analysis. Like for convex functions, it is a set-valued mapping. So, if we want ask questions about the variation of subgradients, then a notion of set convergence is required. Moreover, to exploit full generality of the concepts for nonconvex functions, convergence along sequences plays an important role. In fact the definition of the (general) limiting subgradient is based on that. Its development requires a few preliminary investigations. Sometimes it is not even clear whether a sequence (if it converges) can be assigned a certain direction, which is fundamental for the corresponding theory from the calculus of variations. Subgradients are fundamental to develop the celebrated Fermat's rule for nonconvex functions, which generalizes our understanding of optimality when minimizing such functions. For these types of variation the concepts known from the calculus of variation must be revised.

The book by Rockafellar and Wets [RW98] provides a thorough introduction to this broader understanding of "variation" in finite dimensional spaces whereof this chapter reviews a few definitions and results. As variational analysis is a huge branch in mathematics—the book just mentioned covers already almost 700 pages—, we try to focus on a very compact presentation of the concepts that will be important for further reading of this thesis. For statements whose proof requires the development of more concepts we refer the interested reader to the statements in [RW98] on the spot.

The contribution of Sections 4.1 to 4.4 is a concise presentation of the theory that is required for analyzing convergence of nonconvex optimization algorithms in Chapters 5 to 7. Theorems, examples and figures[1] in this part are taken from the book by Rockafellar [RW98], enlarged by a few more details, examples and figures. Section 4.5 collects results from several research articles about the theory of Kurdyka–Łojasiewicz (KL) functions (and related concepts) in a concise manner and shows several

[1] Reproduced from the book by Rockafellar.

examples. KL functions also play a crucial role in the converge analysis considered in the subsequent chapters.

4.1 Set convergence

As mentioned above set convergence is key for the variational analysis considered here and in [RW98]. For notational convenience, we define

$$\begin{aligned}\mathcal{N}_\infty &:= \{N \subset \mathbb{N} |\, \mathbb{N} \setminus N \text{ finite}\} \leftrightarrow \{\text{subsequences of } \mathbb{N} \text{ containing all } \nu \text{ beyond some } \bar{\nu}\}\,,\\ \mathcal{N}_\infty^\# &:= \{N \subset \mathbb{N} |\, N \text{ infinite}\} \leftrightarrow \{\text{all subsequences of } \mathbb{N}\}\,.\end{aligned} \tag{4.1}$$

Loosely speaking, the first index set $\mathcal{N}_\infty$ can be considered as a collection of neighborhoods of infinity, whereas the second index set $\mathcal{N}_\infty^\#$ are the perforated neighborhoods of infinity, i.e., no matter how close to infinity the sequence is considered, some numbers can be missing. Obviously, it holds $\mathcal{N}_\infty \subset \mathcal{N}_\infty^\#$. We write $\lim_{\nu\to\infty}$ when $\nu \to \infty$ with $\nu \in \mathbb{N}$, and $\lim_{\nu \underset{N}{\to} \infty}$ when $\nu \to \infty$ where ν runs through the index set N. A simple example of $\mathcal{N}_\infty^\#$ is the index set of even (natural) numbers, which is not contained in $\mathcal{N}_\infty$. An example of $\mathcal{N}_\infty$ is the union of the set of even (natural) numbers up to 1000 and all natural numbers larger than 1000. It will be useful to have in mind that these index sets are dual in the sense that

$$\begin{aligned}\mathcal{N}_\infty &= \{N \subset \mathbb{N} |\, \forall N' \in \mathcal{N}_\infty^\#\colon N \cap N' \neq \emptyset\}\,,\\ \mathcal{N}_\infty^\# &= \{N \subset \mathbb{N} |\, \forall N' \in \mathcal{N}_\infty\colon N \cap N' \neq \emptyset\}\,.\end{aligned} \tag{4.2}$$

Theses prior considerations allow us to conveniently introduce the notion of set convergence.

Definition 4.1 (set convergence and outer and inner limit)**.** *For a sequence $(C^\nu)_{\nu\in\mathbb{N}}$ of subsets of $\mathbb{R}^N$, the outer limit is the set*

$$\limsup_{\nu\to\infty} C^\nu := \{x \in \mathbb{R}^N |\, \exists N \in \mathcal{N}_\infty^\#\colon \exists x^\nu \in C^\nu\ (\nu \in N)\colon x^\nu \underset{N}{\to} x\}\,,$$

while the inner limit is the set

$$\liminf_{\nu\to\infty} C^\nu := \{x \in \mathbb{R}^N |\, \exists N \in \mathcal{N}_\infty\colon \exists x^\nu \in C^\nu\ (\nu \in N)\colon x^\nu \underset{N}{\to} x\}\,.$$

The limit of the sequence exists if the outer and inner limit sets are equal:

$$\lim_{\nu\to\infty} C^\nu := \limsup_{\nu\to\infty} C^\nu = \liminf_{\nu\to\infty} C^\nu\,.$$

In this case, the sequence of sets C^ν is said to converge to $C := \lim_{\nu\to\infty} C^\nu$, i.e., $C^\nu \to C$. This notion of convergence is known as Painlevé–Kuratowski convergence*.*

Consider a sequence $(C^\nu)_{\nu\in\mathbb{N}}$ of subsets of $\mathbb{R}^N$ where $C^\nu \neq \emptyset$ for all ν. Then, the inner limit of $(C^\nu)_{\nu\in\mathbb{N}}$ consists of all limit points of sequences with $x^\nu \in C^\nu$ and the outer limit of all cluster points of such sequences. Therefore, $\liminf_{\nu\to\infty} C^\nu \subset \limsup_{\nu\to\infty} C^\nu$ is clear. Figure 4.1 shows one example where the set limit exists, and one where it does not exist, i.e., inner and outer limit do not coincide. Before we show another representation of inner and outer limits, we consider a few examples.

Example 4.1. Let $(C^\nu)_{\nu\in\mathbb{N}}$ be a sequence of sets.

(i) For $C^\nu := \{(-1)^\nu\}$ the limit does not exist as $\nu \to \infty$. It holds that $\liminf_{\nu\to\infty} C^\nu = \emptyset$ and $\limsup_{\nu\to\infty} C^\nu = \{-1, 1\}$.

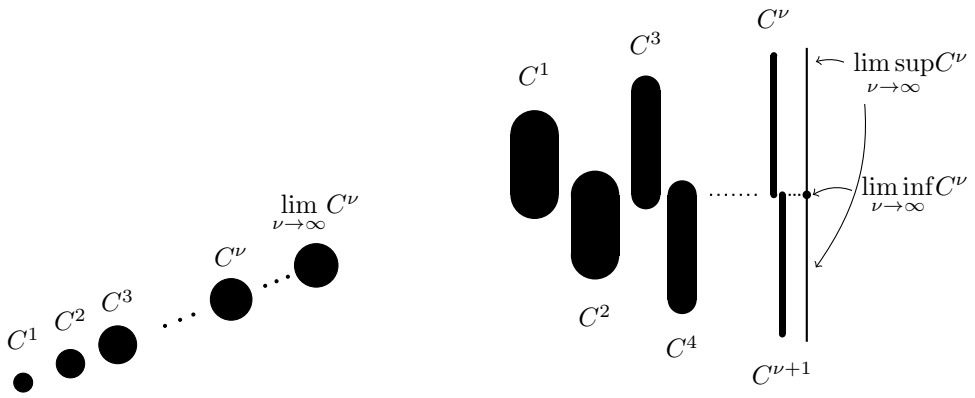


Figure 4.1: Examples of limits of sets. LEFT: Convergence of $(C^\nu)_{\nu\in\mathbb{N}}$. RIGHT: $\lim_n C_n$ does not exist. Where in the example on the left side the inner and outer limit coincide, thus the sequence of sets converges, they differ for the right example.

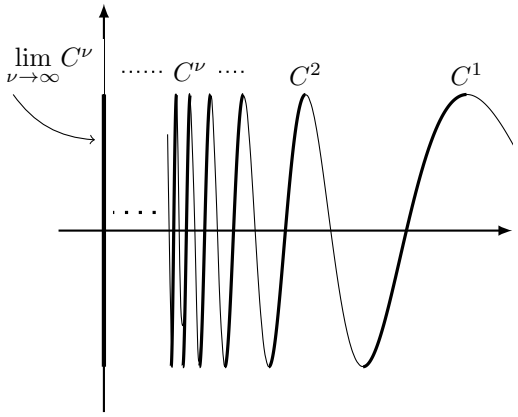


Figure 4.2: Visualization of the sets in Example 4.1(iii). The sequence of sets is given by the bold segments along the sine function. The set C^ν is a subset of the graph of the sine function covering half of a phase where the values in $(-1, 1)$ are taken exactly once. The inner and outer limit coincide, which implies convergence of the sequence of sets.

(ii) Let $C^\nu := \{(x, \sin(\nu x)) \in \mathbb{R}^2 | \, x \in \mathbb{R}\}$. Then the limit exists and $\lim_{\nu\to\infty} C^\nu = \mathbb{R} \times \{0\}$ is the coordinate axis along the first dimension.

(iii) Let $C^\nu := \{(1/x, \sin(2\pi x)) \in \mathbb{R}^2 | \, x \in (\nu - \frac{1}{4}, \nu + \frac{1}{4})\}$. See Figure 4.2 for intuition about the sequence. For this sequence the limit is $\lim_{\nu\to\infty} C^\nu = \{0\} \times [-1, 1]$. Note that the limit is a closed set, though the elements of the sequence are not closed; For $(x^\nu, y^\nu) \in C^\nu$ we have $y^\nu \in (-1, 1)$.

Example 4.2. (i) Let $(B_{\rho^\nu}(x^\nu))_{\nu\in\mathbb{N}}$ be a sequence of balls in $\mathbb{R}^N$. If $x^\nu \to x$ and $\rho^\nu \to \rho$, then $(B_{\rho^\nu}(x^\nu))_{\nu\in\mathbb{N}}$ converges to $B_\rho(x)$.

(ii) Let $A, B \subset \mathbb{R}^N$ be two nonempty sets and consider the sequence $(C^\nu)_{\nu\in\mathbb{N}}$ defined as $C^\nu = A$, when ν is even, and $C^\nu = B$, otherwise. Then $(C^\nu)_{\nu\in\mathbb{N}}$ does not converge unless A and B coincide. However the outer and inner limit can be determined as $\limsup_{\nu\to\infty} C^\nu = A \cup B$ and $\liminf_{\nu\to\infty} C^\nu = A \cap B$, respectively.

(iii) The (constant) sequence $(C^\nu)_{\nu\in\mathbb{N}}$ with $C^\nu = \mathbb{Q}^N$ (set of all vectors with rational coordinates) for all $\nu \in \mathbb{N}$ converges to $\mathbb{R}^N$.

Now, let us consider an equivalent representation of the outer and inner limit, which immediately implies some properties for sets arising in the limit.

Proposition 4.2 (alternative representation of inner and outer set limits)**.** *Let $(C^\nu)_{\nu\in\mathbb{N}}$ be a sequence of subsets of $\mathbb{R}^N$. Then, it holds that*

$$\limsup_{\nu\to\infty} C^\nu = \bigcap_{N\in\mathcal{N}_\infty} \operatorname{cl} \bigcup_{\nu\in N} C^\nu \quad \textit{and} \quad \liminf_{\nu\to\infty} C^\nu = \bigcap_{N\in\mathcal{N}_\infty^\#} \operatorname{cl} \bigcup_{\nu\in N} C^\nu \,.$$

Proof. Define $K_{\text{sup}} := \bigcap_{N\in\mathcal{N}_\infty} \operatorname{cl}\bigcup_{\nu\in N} C^\nu$ and $K'_{\text{sup}} := \bigcap_{N\in\mathcal{N}_\infty} \bigcup_{\nu\in N} C^\nu$. For $x \in K_{\text{sup}}$ we want to show that $x \in \limsup_{\nu\to\infty} C^\nu$. By definition of K_{sup} for all $N \in \mathcal{N}_\infty$ there exists a sequence $(x^n)_{n\in\mathbb{N}}$ with $x^n \in \bigcup_{\nu\in N} C^\nu$ such that $x^n \to x$. As such a sequence exists for all $N \in \mathcal{N}_\infty$, we find $N' \in \mathcal{N}_\infty^\#$ such that $N \cap N' \neq \emptyset$ for all $N \in \mathcal{N}_\infty$ (cf. (4.2)), which means that $(x^m)_{m\in N'} \subset K'_{\text{sup}}$. Moreover, this selected subsequence converges to x and there exists another subsequence (also converging to x) where each element of the sequence lies in a different element of $(C^\nu)_{\nu\in\mathbb{N}}$. Thus, we conclude $x \in \limsup_{\nu\to\infty} C^\nu$. The same idea (using (4.2)) verifies the inverse inclusion. The second equality works analogously. □

Corollary 4.3 (set limits are closed)**.** *Let $(C^\nu)_{\nu\in\mathbb{N}}$ be a sequence of subsets of $\mathbb{R}^N$. Then, the inner and outer limit $\liminf_{\nu\to\infty} C^\nu$ and $\limsup_{\nu\to\infty} C^\nu$ are closed. Moreover, in case of existence, $\lim_{\nu\to\infty} C^\nu$ is also closed.*

Proof. Obvious from Proposition 4.2. □

4.2 Tangent cone

Consider the variation of a point $\bar{x} \in \mathbb{R}^N$ as in one of the plots in Figure 4.3. Variation of a point refers to moving the point, which is only permitted within the feasible set. The point $\bar{x}$ in plot on the left and the middle of Figure 4.3 can be varied along a straight line or a continuously differentiable curve within the feasible set C, and thereby, introduces a meaning of variation in a certain direction. Those two subfigures show the classical ways of varying a point in a direction. However, the example on the right of Figure 4.3 requires a different notion of variation. In this subfigure, the set C is given by the graph of a sine function of the form $x \mapsto x^2 \sin(1/x)$, which oscillates more and more approaching 0 with magnitude scaling as x^2. There is no (continuous) path away from 0 in which the point could be moved. However, the notion of variation can be derived whenever a sequence converges to $\bar{x}$ and all points x^ν along the sequence lie in the feasible set. *Convergence relative to a set $C \subset \mathbb{R}^N$* is defined by

$$x^\nu \underset{C}{\to} \bar{x} \qquad \Leftrightarrow \qquad x^\nu \to \bar{x} \text{ and } x^\nu \in C \,.$$

A sequence $x^\nu \to \bar{x}$ *converges from the direction* $\operatorname{dir} w$, if for some sequence $\tau^\nu \searrow 0$ the vectors $(x^\nu - \bar{x})/\tau^\nu$ converge to w. Actually $\operatorname{dir} w$ is not just a symbol, it has a meaningful interpretation in the so called horizon of $\mathbb{R}^N$. For details, we refer the reader to [RW98, Chapter 3]. Here, it is enough to think of $\operatorname{dir} w$ as a vector in $\mathbb{R}^N$ without a scale with the same direction as w—the concept is related to homogeneous points in a projective space. Note the difference between w and $\operatorname{dir} w$: the vector w has a scale.

A related concept is that of a *right derivative* $\xi'_+(0) := \lim_{\tau\searrow 0} \frac{\xi(\tau)-\xi(0)}{\tau}$ of a vector-valued function $\xi\colon [0,\varepsilon] \to \mathbb{R}^N$ with $\xi(0) = \bar{x}$. The difference to the aforementioned directional convergence is that w

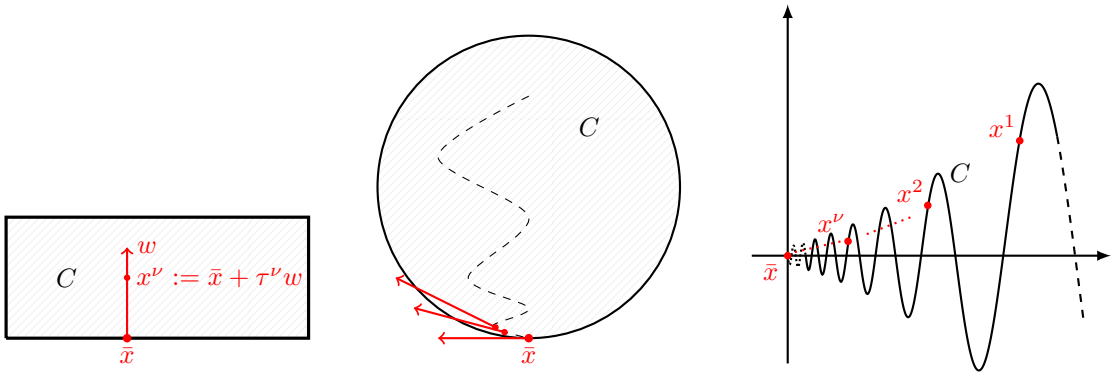


Figure 4.3: Different notions of convergence from a direction are shown. LEFT: Variation along a straight line. MIDDLE: Variation along a smooth curve. RIGHT: Variation along a converging sequence. On the left and in the middle, the classical ways of varying a point in a certain direction within the feasible set C are shown. On the right, the classical way of variation is not applicable. The weaker form of variation along a sequence is required. Each point along the sequence $(x^\nu)_{\nu\in\mathbb{N}}$ converging to $\bar{x}$ relative to C induces a sequence of difference vectors. A converging sequences $((x^\nu - \bar{x})/\tau^\nu)_{\nu\in\mathbb{N}}$ where $\tau^\nu \searrow 0$ defines a vector w and the point $\bar{x}$ can be varied in direction of w.

is a right derivative if *for every sequence* $\tau^\nu \searrow 0$ in $[0,\varepsilon]$ the sequence $(\xi(\tau^\nu) - \xi(0))/\tau^\nu$ converges to w. The definition of right derivatives does not require the function ξ to be continuous and (obviously not differentiable), except at 0.

The following definition introduces a general concept of tangent vectors, which does not rely on the restrictions of a straight line approximation or an approximation along a differentiable curve. Moreover, it shows the difference between right derivatives and directional convergence along sequences.

Definition 4.4 (tangent vector and geometric derivability)**.** *A vector $w \in \mathbb{R}^N$ is tangent to a set $C \subset \mathbb{R}^N$ at a point $\bar{x} \in C$, written $w \in T_C(\bar{x})$, if*

$$\frac{x^\nu - \bar{x}}{\tau^\nu} \to w \quad \text{for some } x^\nu \underset{C}{\to} \bar{x},\ \tau^\nu \searrow 0\,.$$

Such a tangent vector w is derivable if there actually exists $\xi\colon [0,\varepsilon] \to C$ with $\varepsilon > 0$, $\xi(0) = \bar{x}$ and $\xi'_+(0) = w$. The set C is geometrically derivable at $\bar{x}$ if every tangent vector w to C at $\bar{x}$ is derivable.

In other words, $w \in T_C(\bar{x})$ is a tangent vector if there exists a sequence $x^\nu \underset{C}{\to} \bar{x}$ that converges to $\bar{x}$ from the direction $\operatorname{dir} w$. The set of tangent vectors has some convenient properties.

Proposition 4.5 (tangent cone property)**.** *At any point $\bar{x}$ of a set $C \subset \mathbb{R}^N$, the set $T_C(\bar{x})$ of all tangent vectors is a closed cone (thus termed the* tangent cone *to C at $\bar{x}$), so is the subset of $T_C(\bar{x})$ consisting of derivable tangent vectors. Moreover, for the tangent cone holds that $T_C(\bar{x}) = \limsup_{\tau\searrow 0} \tau^{-1}(C - \bar{x})$ and the subset of derivable tangent vectors is expressible with "$\liminf$" in place of "$\limsup$".*

Proof. The cone property is trivial to verify and the closedness follows directly from the representation as outer or inner limit, respectively, (see Corollary 4.3). The expressibility as outer or inner limit is clear from the definition. □

Remark 4.3. According to Proposition 4.5, C is geometrically derivable if and only if the sets $\tau^{-1}(C - \bar{x})$ converge as $\tau \searrow 0$, i.e., $\limsup_{\tau\searrow 0} \tau^{-1}(C - \bar{x}) = \liminf_{\tau\searrow 0} \tau^{-1}(C - \bar{x})$.

Figure 4.4 shows examples for a geometrically derivable tangent cone. The following example clearly separates general tangent vectors from those that are derivable.

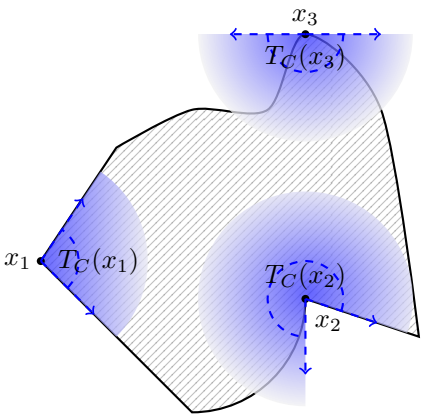


Figure 4.4: Example of a geometrically derivable set and three explicit constructions of derivable tangent cones. By Remark 4.3, for all points, the tangent cone T_C coincides with the derivable tangent cone.

Example 4.4. We consider the graph in $\mathbb{R}^2$, denoted C, of the function $x \mapsto |x| \sin(\log|x|)$ if $x \neq 0$ and $0 \mapsto 0$, whose graph similar to the one on the right of Figure 4.3. Towards 0 it oscillates more and more, however, here the magnitude of the waves scale as $|x|$. The graph is symmetric to $x = 0$.

Consider the sequence $(x^\nu)_{\nu\in\mathbb{N}}$ defined by $x^\nu = \exp(\pi/2)/\exp(2\pi\nu)$. It yields a sequence of points on the graph $((x^\nu, x^\nu))_{\nu\in\mathbb{N}}$. Defining $\tau^\nu \searrow 0$ by $\tau^\nu = x^\nu$ we obtain

$$\frac{(x^\nu, x^\nu) - (0,0)}{\tau^\nu} \to (1,1) \in T_C(\bar{x})\,.$$

Then for the tangent cone at $\bar{x} = (0,0)$ we have

$$T_C(\bar{x}) = \{(w_1, w_2) \in \mathbb{R}^2 \,|\, |w_2| \leq |w_1|\}\,.$$

The derivable cone consists only of $(0,0)$. For $\tau \searrow 0$ the magnification $\tau^{-1}(C - \bar{x}) = \tau^{-1}C$ always show the same (changing) local behavior around $\bar{x} = (0,0)$ and does not converge.

4.3 Normal cone

The counterpart of tangent vectors are normal vectors. Like for tangent vectors, different notions of convergence induce different kinds of normal vectors, and like tangent vectors, normal vectors can be of any length.

Definition 4.6 (normal vectors, normal cone)**.** *Let $C \subset \mathbb{R}^N$ and $\bar{x} \in C$. A vector v is a regular normal vector to C at $\bar{x}$, written $v \in \widehat{N}_C(\bar{x})$, if*

$$\limsup_{\substack{x \xrightarrow{C} \bar{x} \\ x \neq \bar{x}}} \frac{\langle v, x - \bar{x}\rangle}{\|x - \bar{x}\|} \leq 0\,. \tag{4.3}$$

It is a (general) normal vector to C at $\bar{x}$, written $v \in N_C(\bar{x})$, if there are sequences $x^\nu \xrightarrow{C} \bar{x}$ and $v^\nu \to v$ with $v^\nu \in \widehat{N}_C(x^\nu)$.

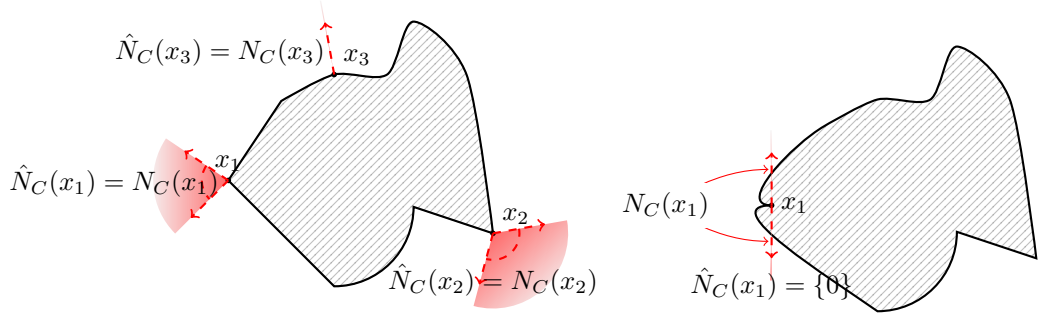


Figure 4.5: Examples for normal cones and regular normal cones. LEFT: Normal and regular normal cone coincide. RIGHT: Normal and regular normal cone differ.

Definition 4.7 (Clarke regularity)**.** *A set $C \in \mathbb{R}^N$ is regular at one of its points $\bar{x} \in C$ in the sense of Clarke if it is locally closed at $\bar{x}$ (i.e. $V \cap C$ is closed for some closed neighborhood V of $\bar{x}$) and every normal vector to C at $\bar{x}$ is a regular normal vector, i.e., $N_C(\bar{x}) = \widehat{N}_C(\bar{x})$.*

In analogy to the set of tangent vectors forming a cone, the set of normal vectors enjoys the same property, which is shown in the following proposition. It allows us to call $N_C(\bar{x})$ the *normal cone* and $\widehat{N}_C(\bar{x})$ the *regular normal cone*. In Figure 4.5 we show some examples. Moreover, the proposition yields a relation between the regular normal cone and the tangent cone. This relation coincides with the usual intuition about tangent and normal vectors at a point of a "nice" set (with smooth enough boundary).

Proposition 4.8 (normal cone property)**.** *At any point $\bar{x}$ of a set $C \subset \mathbb{R}^N$, the set $N_C(\bar{x})$ of all normal vectors is a closed cone, and so too is the set $\widehat{N}_C(\bar{x})$ of regular normal vectors, which in addition is convex and characterized by*

$$v \in \widehat{N}_C(\bar{x}) \qquad \Leftrightarrow \qquad \langle v, w \rangle \leq 0 \text{ for all } w \in T_C(\bar{x})\,. \tag{4.4}$$

Furthermore,

$$N_C(\bar{x}) = \limsup_{x \underset{C}{\to} \bar{x}} \widehat{N}_C(x) \supset \widehat{N}_C(\bar{x})\,. \tag{4.5}$$

Proof. The cone property is obvious. Closedness of $N_C(\bar{x})$ follows from (4.5) (and Corollary 4.3) and for $\widehat{N}_C(\bar{x})$ it follows from (4.4). Moreover (4.4) implies convexity of $\widehat{N}_C(\bar{x})$ using Theorem 2.2 (intersections of convex sets are convex). Since (4.5) is true by definition of the normal cone and the outer limit of a sequence of sets, it remains to verify (4.4).

"$\Rightarrow$": Let $v \in \widehat{N}_C(\bar{x})$ and $w \in T_C(\bar{x})$. Then, by definition of tangency there exist sequences $(x^\nu)_{\nu\in\mathbb{N}}$ and $(\tau^\nu)_{\nu\in\mathbb{N}}$ such that $w^\nu := (x^\nu - \bar{x})/\tau^\nu \to w$ and $\tau^\nu \searrow 0$ when $\nu \to \infty$. Due to (4.3), it holds that

$$\langle v, w \rangle = \limsup_{\nu\to\infty} \langle v, w^\nu \rangle = \limsup_{\nu\to\infty} \frac{\langle v, x^\nu - \bar{x}\rangle}{\|x^\nu - \bar{x}\|} \frac{\|x^\nu - \bar{x}\|}{\tau^\nu} \leq 0\,,$$

which is the right hand side of (4.4).

"$\Leftarrow$": Now, assume the right hand side of (4.4) holds, but $v \notin \widehat{N}_C(\bar{x})$. Due to (4.3) there exists a sequence $(x^\nu)_{\nu\in\mathbb{N}}$ with $x^\nu \underset{C}{\to} \bar{x}$ and $x^\nu \neq \bar{x}$ such that $\liminf_{\nu\to\infty} \langle v, x^\nu - \bar{x}\rangle/\|x^\nu - \bar{x}\| > 0$. As $w^\nu := (x^\nu - \bar{x})/\|x^\nu - \bar{x}\|$ has the property $\|w^\nu\| = 1$, there exists a converging subsequence (again

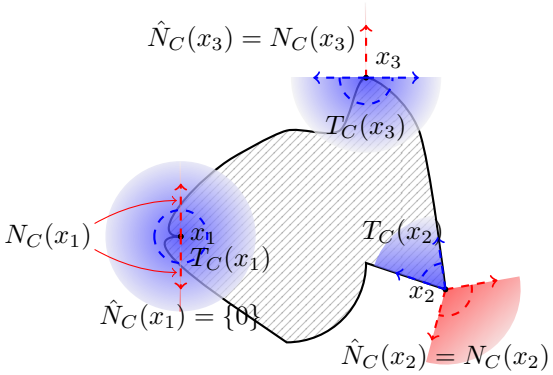


Figure 4.6: Examples for the relation between the regular normal cone and the tangent cone.

denoted by $(w^\nu)_{\nu\in\mathbb{N}}$). The limit point of such a subsequence, denoted by w, belongs by definition to $T_C(\bar{x})$. Plugging in yields the contradiction

$$0 < \liminf_{\nu\to\infty} \frac{\langle v, x^\nu - \bar{x}\rangle}{\|x^\nu - \bar{x}\|} = \lim_{\nu\to\infty} \langle v, w^\nu\rangle = \langle v, w\rangle \ .$$

□

Figure 4.6 shows an example for the relation between the regular normal cone and the tangent cone.

The closedness of normal cones, which we have seen in Proposition 4.8, implies a favorable property. It can be used to conclude if a normal vector arising as the limit of a sequence of normal vectors is a normal vector.

Proposition 4.9 (limits of normal vectors)**.** *If $x^\nu \underset{C}{\to} \bar{x}$, $v^\nu \in N_C(x^\nu)$ and $v^\nu \to v$ then $v \in N_C(\bar{x})$.*

Proof. This is obvious, since $\{(x,v)|\, v \in N_C(x)\}$ is by definition the closure of $\{(x,v)|\, v \in \widehat{N}_C(x)\}$ in $C \times \mathbb{R}^N$, and therefore, it is closed relative to $C \times \mathbb{R}^N$. □

Finally, we want to show how the normal cones and the tangent cone behave for convex sets.

Theorem 4.10 (tangents and normals to convex sets)**.** *A convex set $C \subset \mathbb{R}^N$ is geometrically derivable at any point $\bar{x} \in C$, with*

$$\begin{aligned} N_C(\bar{x}) = \widehat{N}_C(\bar{x}) &= \{v|\ \langle v, x - \bar{x}\rangle \le 0,\ \forall x \in C\}\,, \\ T_C(\bar{x}) &= \mathrm{cl}\{w|\, \exists \lambda > 0 \text{ with } \bar{x} + \lambda w \in C\}\,, \\ \mathrm{int}\, T_C(\bar{x}) &= \{w|\, \exists \lambda > 0 \text{ with } \bar{x} + \lambda w \in \mathrm{int}\, C\}\,. \end{aligned}$$

Furthermore, C is regular at $\bar{x}$ as long as C is locally closed at $\bar{x}$.

Proof. Let $K = \{w|\, \exists \lambda > 0$ with $\bar{x} + \lambda w \in C\}$. For any $x \in C$, it holds that $\bar{x} + \lambda(x - \bar{x}) \in C$ for all $\lambda \in [0,1]$. Therefore vectors $w \in K$ correspond to $\lambda(x - \bar{x})$ for some $\lambda > 0$. The vectors w are derivable tangent vectors, since they arise as right derivatives of linear functions along a line segment joining x with $\bar{x}$. Obviously, $K \subset T_C(\bar{x})$ and, by Definition 4.4, $T_C(\bar{x}) \subset \mathrm{cl}\, K$, thus $T_C(\bar{x}) = \mathrm{cl}\, K$ and C is geometrically derivable at $\bar{x}$. (4.4) verifies the expression for $\widehat{N}_C(\bar{x})$. For $v \in N_C(\bar{x})$ there are sequences $x^\nu \underset{C}{\to} \bar{x}$

and $v^\nu \to v$ with $v^\nu \in \widehat{N}_C(x^\nu)$. For any $x \in C$, it holds that $x - x^\nu \in T_C(x^\nu)$, thus $\langle v^\nu, x - x^\nu \rangle \leq 0$, and due to convergence of $(x^\nu)_{\nu\in\mathbb{N}}$ and $(v^\nu)_{\nu\in\mathbb{N}}$, we have $\langle v, x - \bar{x}\rangle = \lim_{\nu\to\infty} \langle v^\nu, x - x^\nu\rangle \leq 0$. Therefore, $v \in \widehat{N}_C(\bar{x})$, which implies that $N_C(\bar{x}) = \widehat{N}_C(\bar{x})$, as $\widehat{N}_C(\bar{x}) \subset N_C(\bar{x})$ is always true.

In order to verify the last equality, we consider $K_0 = \{w | \exists\lambda > 0 \text{ with } \bar{x} + \lambda w \in \operatorname{int} C\} \neq \emptyset$, an open subset of K. We also have $K \subset \operatorname{cl} K_0$ and conclude $K_0 = \operatorname{int} K = \operatorname{int}(\operatorname{cl} K) = \operatorname{int} T_C(\bar{x})$. □

4.4 Subgradient

The concepts introduced in this chapter so far apply to sets (in an Euclidean space). Now, we transfer them to functions. Like in convex analysis the epigraph of a function is suited for this project. Remember that the definitions *epigraph* (Definition 2.6), *effective domain* (Definition 2.8), *proper function* (Definition 2.9), *lower semi-continuity* (Definition 2.21), *relative continuity* (Definition 2.22), *Lipschitz continuity* (Definition 2.24), and *set-valued mapping* (Definition 2.29) are not specific to convex functions.

Definition 4.11 (subdifferential regularity)**.** *A function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is called subdifferentially regular at $\bar{x}$ if $f(\bar{x})$ is finite and the epigraph* $\operatorname{epi} f := \{(x,t) | x \in \operatorname{dom} f, t \geq f(x)\}$ *is Clarke regular at $(\bar{x}, f(\bar{x}))$ as a subset of $\mathbb{R}^N \times \mathbb{R}$, i.e.,* $\operatorname{epi} f$ *is locally closed and it holds $N_{\operatorname{epi} f}(\bar{x}) = \widehat{N}_{\operatorname{epi} f}(\bar{x})$.*

Due to the strong analogy between sets and function (via their epigraph), it is common to simply call the function associated with a regular epigraph *Clarke regular* or simply *regular*. Local closedness of the epigraph of a function corresponds to (local) lower semi-continuity. Note that Definition 4.11 does not require continuity. A function can be subdifferentially regular at points where it is discontinuous.

We can now introduce the most important concept for our purposes from variational analysis: the subgradient of a function. The subgradients that are defined in Definition 4.12 generalize the idea of minorizing tangents to the epigraph of convex functions by requiring the property only locally (or in the limit of certain sequences). Therefore it is a generalization of the subgradient for convex functions, which we introduced in Definition 2.31. For differentiable functions it reduces to the well-known gradient of a function (see Proposition 4.13).

Definition 4.12 (limiting subgradient, regular subgradient, horizon subgradient)**.** *For a function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ and a point $\bar{x} \in \operatorname{dom} f$*

(i) the subgradient (or limiting subgradient) is defined by

$$\partial f(\bar{x}) = \{v \in \mathbb{R}^N | \exists x^\nu \to \bar{x},\ f(x^\nu) \to f(\bar{x}),\ v^\nu \to v,\ v^\nu \in \widehat{\partial} f(x^\nu)\}\,, \tag{4.6}$$

which makes use of the regular subgradient defined by

$$\widehat{\partial} f(\bar{x}) = \left\{ v \in \mathbb{R}^N \,\middle|\, \liminf_{\substack{x\to\bar{x}\\ x\neq\bar{x}}} \frac{f(x) - f(\bar{x}) - \langle x - \bar{x}, v\rangle}{\|x - \bar{x}\|} \geq 0 \right\}.$$

(ii) The horizon subgradient is defined by

$$\partial^\infty f(\bar{x}) = \{v \in \mathbb{R}^N | \exists x^\nu \to \bar{x},\ f(x^\nu) \to f(\bar{x}),\ \exists \lambda^\nu \searrow 0 : \lambda^\nu v^\nu \to v,\ v^\nu \in \widehat{\partial} f(x^\nu)\}\,.$$

For a satisfactory theory, it is not enough to consider the regular subgradient only. For example the function $f\colon \mathbb{R} \to \mathbb{R}$, $x \mapsto -|x|$ has empty regular subgradient at 0. Of course, one could think of introducing a supergradient for these functions (which is also done), however these concepts do not satisfy all the needs of a robust calculus (see [RW98]). The limiting subgradient for the example $f(x) = -|x|$ is $\partial f(0) = \{-1, 1\}$. Before we consider the relation between subgradients and normal vectors of the epigraph, we want to convince ourselves that subgradients are generalizations of gradients for differentiable functions. In the following we abbreviate continuous differentiability with smoothness.

Proposition 4.13 (relation of gradient to subgradient)**.**

(i) If $f\colon \mathbb{R}^N \to \mathbb{R}$ is differentiable at $\bar{x}$, then $\widehat{\partial} f(\bar{x}) = \{\nabla f(\bar{x})\}$, and $\nabla f(\bar{x}) \in \partial f(\bar{x})$.

(ii) If $f\colon \mathbb{R}^N \to \mathbb{R}$ is smooth on a neighborhood of $\bar{x}$, then $\partial f(\bar{x}) = \{\nabla f(\bar{x})\}$ and $\partial^\infty f(\bar{x}) = \{0\}$.

(iii) Let $f = g + h$ where g is finite at $\bar{x}$ and h is smooth on a neighborhood of $\bar{x}$, then $\widehat{\partial} f(\bar{x}) = \widehat{\partial} g(\bar{x}) + \nabla h(\bar{x})$, $\partial f(\bar{x}) = \partial g(\bar{x}) + \nabla h(\bar{x})$, and $\partial^\infty f(\bar{x}) = \partial^\infty g(\bar{x})$.

Proof. First, remember that $\liminf$ is superadditive, i.e., for two sequences $(a_\nu)_{\nu\in\mathbb{N}}$, $(b_\nu)_{\nu\in\mathbb{N}}$ in $\mathbb{R}$ holds that $\liminf_{\nu\to\infty}(a_\nu + b_\nu) \geq \liminf_{\nu\to\infty} a_\nu + \liminf_{\nu\to\infty} b_\nu$. However, convergence of $(a_\nu)_{\nu\in\mathbb{N}}$ implies $\liminf_{\nu\to\infty}(a_\nu + b_\nu) = \lim_{\nu\to\infty} a_\nu + \liminf_{\nu\to\infty} b_\nu$.

(i) By definition $\nabla f(\bar{x}) \in \widehat{\partial} f(\bar{x})$ holds. For uniqueness let $v, v' \in \widehat{\partial} f(\bar{x})$. Using $v' := \nabla f(\bar{x})$ and the preceding comment about $\liminf$ we obtain

$$0 \leq \liminf_{\substack{x\to\bar{x}\\ x\neq\bar{x}}} \frac{f(x) - f(\bar{x}) - \langle x - \bar{x}, v\rangle}{\|x - \bar{x}\|} = \liminf_{\substack{x\to\bar{x}\\ x\neq\bar{x}}} \frac{\langle x - \bar{x}, v' - v\rangle}{\|x - \bar{x}\|}\,.$$

By considering sequences $(x^\nu_\pm)_{\nu\in\mathbb{N}}$ with $x^\nu_\pm = \bar{x} \pm \tau^\nu(v' - v)$ for some $\tau^\nu \searrow 0$ we conclude that $v = v'$.

(ii) This follows directly from the continuity of ∇f in a neighborhood of $\bar{x}$.

(iii) $\widehat{\partial} f(\bar{x}) \supset \widehat{\partial} g(\bar{x}) + \nabla h(\bar{x})$ is trivial, "$\subset$" follows by applying the inclusion "$\supset$" on $g = f + (-h)$ and noting that $\widehat{\partial} h(\bar{x})$ is a singleton (set with exactly one element). Now, we verify $\partial f(\bar{x}) \supset \partial g(\bar{x}) + \nabla h(\bar{x})$. Let $v_g \in \partial g(\bar{x})$ and $(x^\nu)_{\nu\in\mathbb{N}}$ with $x^\nu \to \bar{x}$, $g(x^\nu) \to g(\bar{x})$, and $v^\nu_g \in \widehat{\partial} g(x^\nu)$ with $v^\nu_g \to v_g$. Thanks to the smoothness of h in a neighborhood of $\bar{x}$ we have $\nabla h(x^\nu) \to \nabla h(\bar{x})$ and $h(x^\nu) \to h(\bar{x})$ as $\nu \to \infty$. As we already know that $v^\nu_g + \nabla h(x^\nu) \in \widehat{\partial} f(x^\nu)$, the inclusion is verified. The inverse inclusion "$\subset$" follows in analogue to the case of the regular subgradient. Finally, $\partial^\infty f(\bar{x}) = \partial^\infty g(\bar{x})$ is obvious from Item (ii). □

At first glance one might assume that for a differentiable function f at $\bar{x}$ also the limiting subgradient is a singleton. However, this is only true if f is continuously differentiable in a neighborhood of $\bar{x}$. The following examples demonstrates this fact.

Example 4.5. Consider the function $f\colon \mathbb{R} \to \mathbb{R}$ defined by $f(x) = x^2 \sin(1/x)$ for $x \neq 0$ and $f(0) = 0$. Then f is continuously differentiable on $\mathbb{R} \setminus \{0\}$ with $\partial f(x) = \widehat{\partial} f(x) = \{\nabla f(x)\} = \{2x \sin(1/x) - \cos(1/x)\}$ and $\partial^\infty f(x) = \{0\}$ for $x \neq 0$. Now consider $\bar{x} = 0$. The function f is differentiable at $\bar{x} = 0$ with $\widehat{\partial} f(\bar{x}) = \{\nabla f(\bar{x})\} = \{0\}$ and $\partial^\infty f(\bar{x}) = \{0\}$. However, $\partial f(\bar{x}) = [-1, 1]$.

Another example, which shows that differentiability at a single point does not imply that the horizon subgradient consists just of 0 is the following.

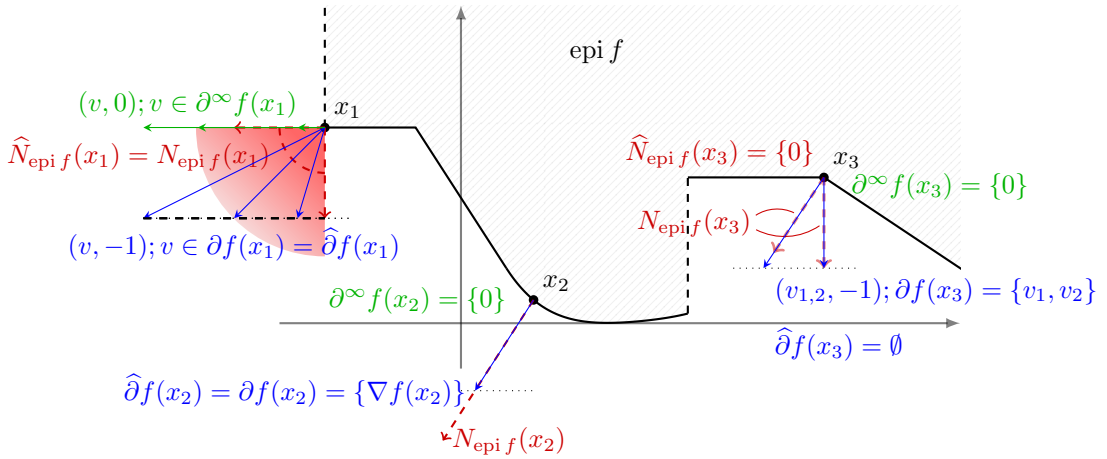


Figure 4.7: Relation between subgradients and normal cones as stated in Theorem 4.14. A function f with its epigraph is shown. At the point x_1 the epigraph of f is curved outward and, like in Figure 4.5, the normal cone and the regular normal cone coincide, thus the same holds for the regular and the limiting subgradient. As the function f is continuously differentiable at x_2, the regular and the limiting subgradient are singletons (see Proposition 4.13). At x_3 the regular normal cone is trivial (set containing only 0), so is the regular subgradient trivial (empty set). The (general) normal cone consists of vectors from exactly two directions. Theorem 4.14 shows that the limiting subgradient consists of exactly two vectors.

Example 4.6. For the function $f\colon \mathbb{R} \to \mathbb{R}$ defined by $f(x) = x^2 \sin(1/x^2)$ for $x \neq 0$ and $f(0) = 0$, it holds at $\bar{x} = 0$ that $\widehat{\partial} f(\bar{x}) = \{\nabla f(\bar{x})\} = \{0\}$, $\partial f(\bar{x}) = (-\infty, \infty)$ and $\partial^\infty f(\bar{x}) = (-\infty, \infty)$.

Next, we will furnish a relation between the subgradients in Definition 4.12 and normal vectors of the epigraph of a function. This relation will give more intuition about limiting subgradients. Moreover, it allows us to easily prove that the limiting subgradient and the regular subgradient coincide for convex functions with the usual notion of subgradients. Examples for the relation are shown in Figure 4.7.

Theorem 4.14 (subgradients from epigraphical normals)**.** *For $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ and any point $\bar{x}$ at which f is finite, one has*

$$\begin{aligned}
\widehat{\partial} f(\bar{x}) &= \{v \in \mathbb{R}^N \,|\, (v,-1) \in \widehat{N}_{\operatorname{epi} f}(\bar{x}, f(\bar{x}))\}\,, \\
\partial f(\bar{x}) &= \{v \in \mathbb{R}^N \,|\, (v,-1) \in N_{\operatorname{epi} f}(\bar{x}, f(\bar{x}))\}\,, \\
\partial^\infty f(\bar{x}) &\subset \{v \in \mathbb{R}^N \,|\, (v,0) \in N_{\operatorname{epi} f}(\bar{x}, f(\bar{x}))\}\,.
\end{aligned}$$

The last relationship holds with equality when f is locally lsc at $\bar{x}$, and then

$$N_{\operatorname{epi} f}(\bar{x}, f(\bar{x})) = \{\lambda(v,-1) \,|\, v \in \partial f(\bar{x}), \lambda > 0\} \cup \{(v,0) \,|\, v \in \partial^\infty f(\bar{x})\}\,.$$

Proof. As the proof of this theorem would require several other concepts that play no role for the further development in this thesis, we refer the interested reader to [RW98, Thm 8.9]. □

From this theorem and the properties of normal cones, we deduce some trivial consequences.

Corollary 4.15. *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a function and let $\bar{x} \in \operatorname{dom} f$. Then $\partial f(\bar{x})$ and $\widehat{\partial} f(\bar{x})$ are closed sets and $\widehat{\partial} f(\bar{x})$ is in addition a convex set. In general, it holds that $\widehat{\partial} f(\bar{x}) \subset \partial f(\bar{x})$.*

Proof. Theorem 4.14 combined with Proposition 4.8 proves the statement. □

Corollary 4.16. *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a function that is finite and locally lsc at a point $\bar{x}$. Then $\partial f(\bar{x}) \neq \emptyset$ or $\partial^\infty f(\bar{x})$ contains a vector $v \neq 0$.*

Proof. By assumption $\operatorname{epi} f$ is locally closed at its boundary points $(\bar{x}, f(\bar{x}))$. Therefore the normal cone there cannot be just the zero cone. Then the statement follows from Theorem 4.14. □

Combining Theorem 4.14 with Definition 4.11 yields a characterization of functions where the situation with limiting and regular subgradients is particularly simple (see [RW98, Cor. 8.11]).

Corollary 4.17. *For a function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ and a point $\bar{x}$ with $f(\bar{x})$ finite and $\partial f(\bar{x}) \neq \emptyset$, local lower semi-continuity of f with $\partial f(\bar{x}) = \widehat{\partial} f(\bar{x})$ is a necessary condition for f being regular at $\bar{x}$.*

Proof. Since f is regular $\operatorname{epi} f$ is Clarke regular at $(\bar{x}, f(\bar{x}))$, which by Definition 4.7 means that $\widehat{N}_{\operatorname{epi} f}(\bar{x}, f(\bar{x})) = N_{\operatorname{epi} f}(\bar{x}, f(\bar{x}))$. Moreover $\operatorname{epi} f$ is locally closed and therefore f is locally lsc. Then the statement follows with Theorem 4.14. □

Proposition 4.18 (relation to subgradient of a convex function)**.** *For any proper, convex function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ and any point $\bar{x} \in \operatorname{dom} f$, one has*

$$\begin{aligned}\partial f(\bar{x}) &= \{v |\, f(x) \geq f(\bar{x}) + \langle v, x - \bar{x}\rangle\,,\ \forall x\} = \widehat{\partial} f(\bar{x})\,,\\ \partial^\infty f(\bar{x}) &\subset \{v |\, 0 \geq \langle v, x - \bar{x}\rangle\,,\ \forall x \in \operatorname{dom} f\} = N_{\operatorname{dom} f}(\bar{x})\,.\end{aligned}$$

The horizon subgradient inclusion is an equality when f is locally lsc at $\bar{x}$ or when $\partial f \neq \emptyset$.

Proof. Theorem 4.10 can be applied, since $\operatorname{epi} f$ is convex. Due to $N_{\operatorname{epi} f}(\bar{x}, f(\bar{x})) = \widehat{N}_{\operatorname{epi} f}(\bar{x}, f(\bar{x}))$ follows $\partial f(\bar{x}) = \widehat{\partial} f(\bar{x})$ by Theorem 4.14. By definition holds that

$$\widehat{N}_{\operatorname{epi} f}(\bar{x}, f(\bar{x})) = \{(v,\beta) |\, \langle (v,\beta), (x,\alpha) - (\bar{x}, f(\bar{x}))\rangle\,,\ \forall (x,\alpha) \in \operatorname{epi} f\}\,.$$

Plugging this into the relations in Theorem 4.14 verifies the first line. The relation of the horizon subgradient follows by Theorem 4.14 and $\{v |\, (v,0) \in N_{\operatorname{epi} f}(\bar{x}, f(\bar{x}))\} = \{v |\, v \in N_{\operatorname{dom} f}(\bar{x})\}$. □

Now, let us consider a few examples. As we know the usual gradient and the subgradient for convex function, we discuss examples of nonsmooth nonconvex functions only.

Example 4.7. The types of functions considered in this example appear in many applications from computer vision, e.g. Section 6.2.2, Section 7.5 and subsequent sections. Note that subgradients are defined only where the function takes finite values. Let $1 > \varepsilon > 0$.

(i) Let $0 < p < 1$ and $f_\varepsilon\colon \mathbb{R}^2 \to \mathbb{R}$ be given by $x = (x_1, x_2) \mapsto (\|x\|_2^2 + \varepsilon)^p = (\sqrt{x_1^2 + x_2^2} + \varepsilon)^p$. For $x \neq 0$ the function is continuously differentiable and thus the subgradient is a singleton and contains

$$\nabla f_\varepsilon(x) = \frac{p\Big(\frac{x_1}{\sqrt{x_1^2+x_2^2}}, \frac{x_2}{\sqrt{x_1^2+x_2^2}}\Big)}{(\sqrt{x_1^2 + x_2^2} + \varepsilon)^{1-p}}\,.$$

Let us considere the definition of the regular subgradient at $\bar{x} = 0$, which is the set of vectors v satisfying

$$\begin{aligned}\liminf_{\substack{x\to\bar{x}\\ x\neq\bar{x}}} \frac{f_\varepsilon(x) - f_\varepsilon(\bar{x}) - \langle x - \bar{x}, v\rangle}{\|x - \bar{x}\|_2} &= \liminf_{\substack{x\to 0\\ x\neq 0}} \frac{(\|x\|_2 + \varepsilon)^p - \varepsilon^p - \langle v, x\rangle}{\|x\|_2}\\ &= \liminf_{\substack{x\to 0\\ x\neq 0}} \frac{(\|x\|_2 + \varepsilon)^p - \varepsilon^p}{\|x\|_2} - \left\langle v, \frac{x}{\|x\|_2}\right\rangle \geq 0\,,\end{aligned} \tag{4.7}$$

where in the last expression the first term is independent of the direction from which x tends to 0 and the second is independent of the distance to 0. Consider a sequences $x^\nu \to 0$ from $\operatorname{dir} v$ given by $\tau^\nu \searrow 0$ and $x^\nu = \tau^\nu v/\|v\|$. Then (4.7) comes down to

$$\liminf_{\tau^\nu \searrow 0} \frac{(\tau^\nu + \varepsilon)^p - \varepsilon^p}{\tau^\nu} \geq \|v\| \,,$$

where the left hand side (if $\liminf = \lim$) is the directional derivative of f_ε in direction $v/\|v\|$. In the case of $\liminf$ there is a concept of subderivatives, which is defined in [RW98, Def. 8.1] and investigated further there. We mention it only in the scope of this example. Finally, we observe $\liminf_{\tau \searrow 0} \frac{\varepsilon^p - (\tau+\varepsilon)^p}{\tau} = \lim_{\tau \searrow 0} \frac{\varepsilon^p - (\tau+\varepsilon)^p}{\tau} = p\varepsilon^{p-1}$ and the above inequality is satisfied if $\|v\| \in [0, p\varepsilon^{p-1}]$, which implies

$$\widehat{\partial} f(0) = \varepsilon^{p-1} \overline{B}_1(0) \,,$$

where $\overline{B}_1(0)$ is the closed unit ball around 0 with radius 1. In order to determine the limiting subgradient we consider the limits of subgradients (here gradients) along sequences tending to 0. From the expression for $\nabla f_\varepsilon(x)$ follows that limits along such sequences are contained in $\widehat{\partial} f(0)$, which shows that $\widehat{\partial} f(0) = \partial f(0)$.

(ii) Let $f_1, f_2 \colon \mathbb{R} \to \overline{\mathbb{R}}$ be two function defined by

$$f_1(x) = \begin{cases} \log(1+x), & \text{if } x \geq -\varepsilon, \\ \infty, & \text{otherwise,} \end{cases} \quad \text{and} \quad f_2(x) = \log(1+|x|) \,.$$

The subgradients are given by

$$\partial f_1(x) = \widehat{\partial} f_1(x) = \begin{cases} \{\frac{1}{1+x}\}, & \text{if } x > -\varepsilon, \\ (-\infty, \frac{1}{1-\varepsilon}], & \text{if } x = -\varepsilon, \end{cases} \quad \text{and} \quad \partial^\infty f_1(x) = \begin{cases} \{0\}, & \text{if } x > -\varepsilon, \\ (-\infty, 0], & \text{if } x = -\varepsilon, \end{cases}$$

and

$$\partial f_2(x) = \widehat{\partial} f_2(x) = \begin{cases} \{\frac{1}{1+x}\}, & \text{if } x > 0, \\ \{-\frac{1}{1-x}\}, & \text{if } x < 0, \\ [-1, 1], & \text{if } x = 0. \end{cases} \quad \text{and} \quad \partial^\infty f_2(x) = \{0\}, \text{ for all } x \,.$$

As we already noticed in the convex setting, Lipschitz continuity is important in many cases. In the following, we make use of a weaker property, namely local Lipschitz continuity. The (local) Lipschitz constant that is associated with this continuity uniformly bounds a function in a certain neighborhood of a point.

Definition 4.19 (Lipschitz continuity and local Lipschitz continuity)**.** *A function $f \colon D \to \mathbb{R}^M$ with $D \subset \mathbb{R}^N$, $N, M \in \mathbb{N}$, is called Lipschitz continuous on $X \subset D$ if there exists $L \geq 0$ such that*

$$\|f(x) - f(x')\| \leq L\|x - x'\| \quad \textit{for all } x, x' \in X \,.$$

Then L is called the Lipschitz constant of f on X.

The function f is locally Lipschitz continuous on $X \subset D$ if for every point $x \in X$ there exists a neighborhood U of x such that f is Lipschitz continuous on $U \cap X$.

Although Lipschitz continuity is a rather weak assumption for a function, it has some useful consequences for the work with subgradients.

Proposition 4.20 (relation of horizon and limiting subgradient)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a locally lower semi-continuous (lsc) function with finite value at $\bar{x}$. Then the following statements are equivalent:*

(i) f is locally Lipschitz continuous at $\bar{x}$,

(ii) $\partial^\infty f(\bar{x}) = \{0\}$,

(iii) $\partial f : x \mapsto \partial f(x)$ is locally bounded at $\bar{x}$,

(iv) $\widehat{\partial} f : x \mapsto \widehat{\partial} f(x)$ is locally bounded at $\bar{x}$.

Proof. As the proof of this theorem would require several other concepts that play no role for the further development in this thesis, we refer the interested reader to [RW98, Thm 9.13]. □

For composite functions as they will occur later, it is advantageous to have rules how the respective subgradients are composed. We consider addition and composition of functions. The next proposition relates the subgradient of a sum of functions to the sum of the subgradients of functions.

Proposition 4.21 (addition of functions)**.** *Suppose $f = f_1 + \ldots + f_m$ for proper, lsc functions $f_i\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, and let $\bar{x} \in \operatorname{dom} f$. Then*

$$\widehat{\partial} f(\bar{x}) \supset \widehat{\partial} f_1(\bar{x}) + \ldots + \widehat{\partial} f_m(\bar{x})\,.$$

If the only combination of vectors $v_i \in \partial^\infty f_i(\bar{x})$ with $v_1 + \ldots + v_m = 0$ is $v_1 = \ldots = v_m = 0$, one also has that

$$\partial f(\bar{x}) \subset \partial f_1(\bar{x}) + \ldots + \partial f_m(\bar{x})\,.$$

If also each f_i is Clarke regular at $\bar{x}$, then f is Clarke regular at $\bar{x}$ and

$$\partial f(\bar{x}) = \partial f_1(\bar{x}) + \ldots + \partial f_m(\bar{x})\,.$$

Proof. As the proof of this theorem would require several other concepts that play no role for the further development in this thesis, we refer the interested reader to [RW98, Cor. 10.9]. □

A simple example shows that the inclusions are proper.

Example 4.8. Consider the proper, lsc functions $f_1, f_2\colon \mathbb{R} \to \mathbb{R}$ where $f_1(x) = |x|$ and $f_2(x) = -|x|$. Then $f(x) = f_1(x) + f_2(x) = 0$ and it holds that

$$\begin{array}{llll} \widehat{\partial} f_1(0) = [-1,1], & \widehat{\partial} f_2(0) = \emptyset, & \widehat{\partial} f(0) = \{0\}, & \widehat{\partial} f_1(0) + \widehat{\partial} f_2(0) = \emptyset; \\ \partial f_1(0) = [-1,1], & \partial f_2(0) = \{-1,1\}, & \partial f(0) = \{0\}, & \partial f_1(0) + \partial f_2(0) = [-2,2]\,. \end{array}$$

Proposition 4.22 (extended chain rule)**.** *Let $f(x) = F(G(x))$ for a proper, lsc function $F\colon \mathbb{R}^m \to \overline{\mathbb{R}}$ and a locally Lipschitz continuous vector-valued function $G\colon X \to \mathbb{R}^m$, $X \subset \mathbb{R}^N$. Then, for $\bar{x} \in X$ it holds*

$$\widehat{\partial} f(\bar{x}) \supset \widehat{D}^* G(\bar{x})[\widehat{\partial} F(G(\bar{x}))] = \bigcup \{\widehat{\partial} \langle y, G\rangle (\bar{x}) |\, y \in \widehat{\partial} F(G(\bar{x}))\}\,.$$

If the only vector $y \in \partial^\infty F(G(\bar{x}))$ with $0 \in \partial \langle y, G\rangle (\bar{x})$ is $y = 0$, one also has

$$\partial f(\bar{x}) \subset D^* G(\bar{x})[\partial F(G(\bar{x}))] = \bigcup \{\partial \langle y, G\rangle (\bar{x}) |\, y \in \partial F(G(\bar{x}))\}\,,$$

If in addition F is regular at $G(\bar{x})$ and $\langle y, G\rangle$ is regular at $\bar{x}$ for each $y \in \partial F(G(\bar{x}))$, then f is regular at $\bar{x}$ and $\partial f(\bar{x}) = D^* G(\bar{x})[\partial F(G(\bar{x}))]$.

Proof. We refer the interested reader to [RW98, Thm. 10.49]. □

Finally, we want to characterize an optimality condition for finding extremal points of a proper function. Like in the classical calculus (and also in convex analysis) Fermat's rule yields such a necessary first order optimality condition: the variation at extremal points vanishes. A point $\bar{x}$ that satisfies the optimality condition in Theorem 4.23 is called a *stationary point* or a *critical point*. The following theorem formalizes this optimality condition for a general function.

Theorem 4.23 (Fermat's rule generalized)**.** *If a proper function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ has a local minimum at $\bar{x}$, then $0 \in \partial f(\bar{x})$.*

Proof. As the function has a local minimum at $\bar{x}$, it satisfies $f(x) \geq f(\bar{x})$ for all x in a certain neighborhood of $\bar{x}$. By Definition 4.12 it is $0 \in \widehat{\partial} f(\bar{x})$. Corollary 4.15, which states $\widehat{\partial} f(\bar{x}) \subset \partial f(\bar{x})$, proves the statement. □

A simple consequence is of Fermat's rule is the following.

Corollary 4.24 (Fermat's rule)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper function given as $f = f_0 + g$, where f_0 is smooth, then $0 \in \partial f(\bar{x})$ corresponds to $-\nabla f_0(\bar{x}) \in \partial g(\bar{x})$.*

Proof. The statement is the combination of Theorem 4.23 with Proposition 4.13(iii). □

4.5 Nonsmooth Kurdyka–Łojasiewicz inequality

Łojasiewicz established a main ingredient for proving convergence of bounded trajectories of the gradient dynamical system on real analytic functions to a critical point [Łoj63], namely the "Łojasiewicz inequality". For a real analytic function $f\colon \mathbb{R}^N \to \mathbb{R}$ there exist some $\theta \in [\frac{1}{2}, 1)$ such that $|f - f(a)|^\theta / \|\nabla f\|$ remains bounded around the critical point $a \in \mathbb{R}^N$. Extensions of the Łojasiewicz inequality are derived, for example, in [Kur98] for smooth functions definable in an o-minimal structure (see Section 4.5.2), and for nonsmooth definable functions in [BDLS07]. In [AB09, BDL06a, BDL06b] similar results to that of Łojasiewicz are shown in the nonsmooth subanalytic setting: the "Kurdyka–Łojasiewicz inequality". Loosely speaking, it means that the limiting subgradient can be strictly separated from 0.

Simple, but important examples for such an o-minimal structure are semi-algebraic functions, which we briefly introduce in Section 4.5.1. Sometimes semi-algebraic functions are considered as the smallest nontrivial o-minimal structure. In fact, o-minimal structures are an axiomatic construction that preserves the favorable properties of the semi-algebraic structure [dD98]. Another such structure that provides interesting definable functions is constructed from globally subanalytic sets. It is important to note that, as we will see in the following subsections, these are not only exotic concepts; Most of the functions arising in optimization problems in practice are definable. In Theorem 4.35 we state the result that establishes the Kurdyka–Łojasiewicz inequality for definable functions.

Using the Kurdyka–Łojasiewicz (KL) inequality several algorithms have been shown to converge [CPR13, ABS13, AB09, ABRS10, THD09, OCBP14] even for nonconvex functions. In [ABS13] an abstract convergence theorem for descent methods with certain properties is proved. We present an extension of this in Chapter 5. Summarizing, an algorithm with a certain descent property and a relative upper bound on the (limiting) subgradient applied to a function with the KL property can be shown to converge (under some additional mild conditions to the objective) to a stationary point.

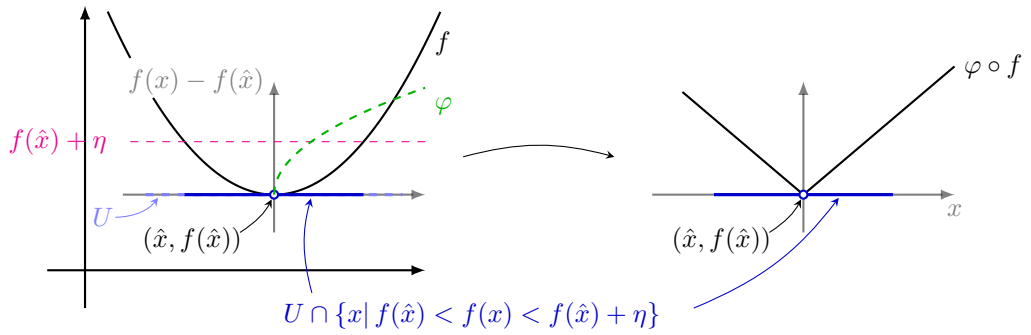


Figure 4.8: Example of the KL property for a smooth function. The composition $\varphi \circ f$ has a slope of magnitude 1 except at $\hat{x}$.

Now, we formulate the Kurdyka–Łojasiewicz property as in [ABS13] and consider examples—the ones mentioned above—in the subsequent subsections.

Definition 4.25 (Kurdyka–Łojasiewicz property)**.** *Let $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be an extended real valued function and let $\hat{x} \in \operatorname{dom} \partial f$. If there exists $\eta \in (0, \infty]$, a neighborhood U of $\hat{x}$ and a continuous concave function $\varphi\colon [0, \eta) \to \mathbb{R}_+$ such that $\varphi(0) = 0$, $\varphi \in C^1((0, \eta))$, and $\varphi'(s) > 0$ for all $s \in (0, \eta)$, and for all $x \in U \cap \{x \in \mathbb{R}^N : f(\hat{x}) < f(x) < f(\hat{x}) + \eta\}$ holds the Kurdyka–Łojasiewicz inequality*

$$\varphi'(f(x) - f(\hat{x})) \operatorname{dist}(0, \partial f(x)) \geq 1\,, \tag{4.8}$$

then the function has the Kurdyka–Łojasiewicz property at $\hat{x}$.

If, additionally, the function is lower semi-continuous and the property holds for each point in $\operatorname{dom} \partial f$, then f is called a Kurdyka–Łojasiewicz function.

It is easy to see that the Kurdyka–Łojasiewicz property is satisfied for all nonstationary points [ABRS10, Lem. 2]. For the purpose of intuition, it should be mentioned, that for smooth functions (with $f(\hat{x}) = 0$) (4.8) is equivalent to $\|\nabla(\varphi \circ f)(x)\| \geq 1$. This means, that after reparametrization via φ the gradient ∇f may be strictly separated from 0. In Figure 4.8 we show an example. For alternative interpretations, including subgradient flows, we refer the interested reader to [BDLM10].

4.5.1 Semi-algebraic sets and functions

The first chapters of [BCR98] are a good reference for the introduction of real semi-algebraic functions. We recap the definition and show a few basic results.

Definition 4.26 (semi-algebraic sets and functions)**.** *A subset S of $\mathbb{R}^N$ is a real semi-algebraic set if it is expressible as*

$$S = \bigcup_{j=1}^{p} \bigcap_{i=1}^{q} \{x \in \mathbb{R}^N | \, f_{i,j}(x) = 0, g_{i,j}(x) < 0\}\,,$$

where $f_{i,j}, g_{i,j}\colon \mathbb{R}^N \to \mathbb{R}$, $1 \leq i \leq q$, $1 \leq j \leq p$, $p, q \in \mathbb{N}$, are real polynomials.

A function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is called a semi-algebraic function if its graph $\operatorname{Graph} f$ is a semi-algebraic subset of $\mathbb{R}^{N+1}$. A set-valued mapping $F\colon \mathbb{R}^N \rightrightarrows \mathbb{R}^M$ is semi-algebraic if its graph $\operatorname{Graph} F$ is a semi-algebraic subset of $\mathbb{R}^{N+M}$.

A rather trivial result, which in view of Section 4.5.2, however, is worth mentioning, is the following.

Proposition 4.27 ([BCR98, Prop. 2.1.7])**.** *Semi-algebraic subsets of* $\mathbb{R}$ *are exactly the finite unions of points and open intervals (bounded or unbounded).*

Example 4.9. (i) Polynomials are semi-algebraic.

(ii) Let $\Omega_i \subset \mathbb{R}^N$, $i = 1, \dots, K$, be a finite partition of a set $\Omega \subset \mathbb{R}^N$ and all Ω_i be semi-algebraic. Then, a function that is defined by a polynomial on each set Ω_i is semi-algebraic.

(iii) The absolute value $|\cdot| \colon \mathbb{R} \to \mathbb{R}$ is semi-algebraic due to (ii) or, explicitly verified, since its graph can be written as

$$\operatorname{Graph} |\cdot| = (\{(x,y) | \, y - x = 0\} \cup \{(x,y) | \, y + x = 0\}) \cap \{(x,y) | \, y \geq 0\} \, .$$

(iv) The set defined by $\{(x,y) \in \mathbb{R}^2 | \, y = \exp(x)\}$ is not semi-algebraic.

The favorable property of semi-algebraicity is its stability; For example, the finite union or finite intersection of semi-algebraic sets is semi-algebraic, or the composition of semi-algebraic functions is semi-algebraic. The link between semi-algebraicity and o-minimal structures (Section 4.5.2) largely stems from the following fact (see [BCR98, Thm. 2.2.1]), which is an important result that contributes to the stability of semi-algebraicity.

Theorem 4.28 (Tarski–Seidenberg)**.** *Let* S *be a semi-algebraic subset of* $\mathbb{R}^{N+1}$ *and* $\Pi \colon \mathbb{R}^{N+1} \to \mathbb{R}^N$ *the projection on the space of the first* N *coordinates. Then* $\Pi(S)$ *is a semi-algebraic subset of* $\mathbb{R}^N$*.*

An obvious application of this Theorem is that the preimage of a semi-algebraic function is semi-algebraic, i.e., if the function $f \colon C \to \mathbb{R}$ defined on a set $C \subset \mathbb{R}^N$ is semi-algebraic, then necessarily, C is a semi-algebraic set, which is the projection of $\operatorname{Graph} f$ onto the first N coordinates. Theorem 4.28 implies some more stability results, which allow us to construct many semi-algebraic functions.

Proposition 4.29 ([BCR98, Prop. 2.2.2, Prop. 2.2.6, Prop. 2.2.7])**.** *(i) The closure and the interior of a semi-algebraic set are semi-algebraic.*

(ii) The composition $G \circ F$ *of semi-algebraic mappings* $F \colon A \rightrightarrows B$ *and* $G \colon B \rightrightarrows C$ *is semi-algebraic, where* A*,* B*,* C *are semi-algebraic sets.*

(iii) For a semi-algebraic mapping $F \colon A \rightrightarrows B$ *(with* A*,* B *semi-algebraic),* $S \subset A$ *semi-algebraic implies* $F(S)$ *semi-algebraic. If* $T \subset B$ *is semi-algebraic, then its inverse image* $F^{-1}(T)$ *is also semi-algebraic.*

Before, we step forward to the next abstraction level in the following subsection, we conclude this section with a few examples that are important for applications.

Example 4.10 ([BST14, Ex. 3–4]). (i) The sparsity measure $\|x\|_0 := \#\{i \in \{1, \dots, N\} | \, x_i \neq 0\}$, $x \in \mathbb{R}^N$, is semi-algebraic, where $\#A$ counts the number of elements in the set A.

(ii) The p-norm $\|\cdot\|_p$ with $p > 0$ is semi-algebraic for any rational number p, where $\|x\|_p^p = \sum_{i=1}^N |x_i|^p$ for $x \in \mathbb{R}^N$. It is not semi-algebraic when p is irrational.

Example 4.11. (i) Let K be a linear operator $K \colon \mathbb{R}^N \to \mathbb{R}^M$. Then for any semi-algebraic function f the composition $f \circ K$ is semi-algebraic. In particular $x \mapsto \|Kx\|_p$ is semi-algebraic.

(ii) Indicator functions of semi-algebraic sets are semi-algebraic.

4.5.2 Functions definable in an o-minimal structure

We define o-minimal structures as in [BDLS07]. Compared to the definition in [dD98] we are only interested in structures that contain semi-algebraic sets.

Definition 4.30 (o-minimal structure)**.** *Let $\mathcal{O} = \{\mathcal{O}_N\}_{N\in\mathbb{N}}$ be such that $\mathcal{O}_N$ is a collection of subsets of $\mathbb{R}^N$. The family $\mathcal{O}$ is an o-minimal structure over $\mathbb{R}$, if it satisfies the following axioms:*

(i) Each $\mathcal{O}_N$ is a boolean algebra, i.e., $\emptyset \in \mathcal{O}_N$ and for each $A, B \in \mathcal{O}_N$ also $A \cup B$, $A \cap B$ and $\mathbb{R}^N \setminus A$ belong to $\mathcal{O}_N$.

(ii) For all $A \in \mathcal{O}_N$ the Cartesian products $A \times \mathbb{R}$ and $\mathbb{R} \times A$ belong to $\mathcal{O}_{N+1}$.

(iii) For all $A \in \mathcal{O}_{N+1}$ the projection onto the first N coordinates $\Pi(A)$ belongs to $\mathcal{O}_N$.

(iv) $\mathcal{O}_N$ contains the family of algebraic subsets of $\mathbb{R}^N$, i.e., every set of the form $\{x \in \mathbb{R}^N \,|\, p(x) = 0\}$ where $p\colon \mathbb{R}^N \to \mathbb{R}$ is a polynomial function.

(v) The elements of $\mathcal{O}_1$ are exactly finite unions of points and (open) intervals.

A set $A \subset \mathbb{R}^N$ is said to be definable (in $\mathcal{O}$), if A belongs to $\mathcal{O}_N$. Analogue to the preceding subsection, the property is transferred to a set-valued mapping via its graph. A set-valued mapping $F\colon \mathbb{R}^N \rightrightarrows \mathbb{R}^M$ is said to be definable (in $\mathcal{O}$) if $\operatorname{Graph} F$ is a definable subset of $\mathbb{R}^N \times \mathbb{R}^M$.

Remark 4.12. The difference to the definition in [dD98] is that, instead of Item (iv), the following axioms are required:

(iv)′ For all $i \neq j$ in $\{1, \dots, N\}$ the set $\{(x_1, \dots, x_N)^\top \in \mathbb{R}^N \,|\, x_i = x_j\}$ is in $\mathcal{O}_N$.

(iv)″ The set $\{(x_1, x_2) \in \mathbb{R}^2 \,|\, x_1 < x_2\}$ belong to $\mathcal{O}_2$.

However, these are simple consequences of Item (iv). Item (iv)′ is included in our definition as these sets are constrained by polynomial equations. The sets in Item (iv)″ can be seen as projection onto the first two coordinates of $\{(x_1, x_2, x_3) \in \mathbb{R} \times \mathbb{R} \times (-\infty, 0) \,|\, x_3 = x_1 - x_2\}$ (constructible using Item (v)).

Obviously, semi-algebraic sets form such an o-minimal structure thanks to the theorem of Tarski–Seidenberg. Before we introduce another important example, namely that of globally subanalytic sets, let us briefly analyze some simple consequences of the axioms. Definable structures have the advantage of being stable under various operations. We state the result and prove it as in [dD98, Chap. 1, Lem. 2.3] and show a corollary, which follows immediately. Thereby, we make use of the fact that permutations of the coordinates does not alter the definability ([dD98, Chap. 1, Lem. 2.2]), and that Definition 4.30(ii) and (iii) can be applied iteratively.

Lemma 4.31. *Let $S \subset \mathbb{R}^M$ and let $f\colon S \to \mathbb{R}^N$ be a map that belongs to $\mathcal{O}$. Then we have the following properties:*

(i) The domain of the function is necessarily a definable set: $S \in \mathcal{O}_M$.

(ii) For any definable set $A \in \mathcal{O}_M$, $A \subset S$, the image of the function is definable ($f(A) \in \mathcal{O}_N$) and the restriction of f to A belongs to $\mathcal{O}$.

(iii) For a definable set $B \in \mathcal{O}_N$ the preimage is definable, i.e., $f^{-1}(B) \in \mathcal{O}_M$.

(iv) For an injective function f the inverse f^{-1} belongs to $\mathcal{O}$.

(v) If $f(S) \subset T \subset \mathbb{R}^N$ and $g\colon T \to \mathbb{R}^P$ is a second map belonging to $\mathcal{O}$, then the composition $g \circ f\colon S \to \mathbb{R}^P$ belongs to $\mathcal{O}$.

Proof. Since f is definable, its graph is a definable subset of $\mathbb{R}^M \times \mathbb{R}^N$.

(i) Use $x \in S \Leftrightarrow \exists y \in \mathbb{R}^N\colon (x,y) \in \operatorname{Graph} f$ and the projection axiom Definition 4.30(iii).

(ii) Since $y \in f(A) \Leftrightarrow \exists x \in A\colon (x,y) \in \operatorname{Graph} f$, permuting (x,y) does not change the definability, and the projection axiom proves this fact.

(iii) The projection axiom shows that $f^{-1}(B) = \{(x,y) |\, \exists y \in B\colon (x,y) \in \operatorname{Graph} f\}$ is definable.

(iv) Note that injectivity implies $(y,x) \in \operatorname{Graph} f^{-1} \Leftrightarrow (x,y) \in \operatorname{Graph} f$.

(v) Observe that $\operatorname{Graph}(g \circ f)$ is the projection of $(\operatorname{Graph} f \times \mathbb{R}^P) \cap (\mathbb{R}^M \times \operatorname{Graph} g)$ onto $\mathbb{R}^M \times \mathbb{R}^P$ and hence definable. □

Corollary 4.32. *Let $S, T \subset \mathbb{R}^M$, $S \cap T \neq \emptyset$, and let $f\colon S \to \mathbb{R}^N$, $g\colon T \to \mathbb{R}^N$ be maps that belongs to $\mathcal{O}$. Then pointwise addition and multiplication, $f + g$ and $f \cdot g$, restricted to $S \cap T$ belongs to $\mathcal{O}$.*

Although the structure of semi-algebraic function is already large and often enough for applications, there are other structures of interest. As mentioned earlier, one of these is generated by globally subanalytic functions. Let us approach this concept by a definition.

Definition 4.33 (semi-analytic, subanalytic sets and functions)**.**

(i) A subset S of $\mathbb{R}^N$ is a real semi-analytic set if for each point in $\mathbb{R}^N$ there exists a neighborhood V such that

$$S \cap V = \bigcup_{j=1}^{p} \bigcap_{i=1}^{q} \{x \in \mathbb{R}^N : f_{i,j}(x) = 0, g_{i,j}(x) < 0\},$$

where $f_{i,j}, g_{i,j}\colon V \to \mathbb{R}$, $1 \le i \le q$, $1 \le j \le p$, are real analytic functions.

(ii) A subset S of $\mathbb{R}^N$ is called subanalytic, if for each point in $\mathbb{R}^N$ there exists a neighborhood V such that $S \cap V = \{x \in \mathbb{R}^N |\, \exists y \in \mathbb{R}^M\colon (x,y) \in D\}$, where D is a bounded semi-analytic subset of $\mathbb{R}^N \times \mathbb{R}^M$ with $M \ge 1$.

As before, a function is said to have one of these properties if its graph has it. Obviously, among semi-algebraic, semi-analytic, and subanalytic sets the latter are the most general and comprises the others. However, as the following example shows, subanalyticity does not satisfy the projection axiom, and hence does not form an o-minimal structure, which does not mean that it is useless; we refer to the book [Shi93] and [BM88]. Nevertheless, there exists a subset, called *globally subanalytic sets* (to be defined soon), that induces an o-minimal structure. Though it is a proper subset, it provides enough flexibility for our purposes.

Example 4.13 (from [BDL06a]). Consider the set $A := \{(1/(n+1), n) |\, n \in \mathbb{N}\}$. The projection of A onto $\mathbb{R} \times \{0\}$ is not subanalytic at 0. The arguments to verify this comes from [dDM96, Fact 1.10]: *A subanalytic set has locally only a finite number of connected components.* Since the projected set $\{1/(n+1) |\, n \in \mathbb{N}\}$ has a infinite number of connected components, A is not subanalytic at 0.

Loosely speaking, in the preceding example the problem is the unboundedness of the set A. This problem is solved when we consider globally subanalytic sets (as in [dDM96]) where such situations are excluded. Define for $N \in \mathbb{N}$ the set $C_N := (-1,1)^N$ and τ_N by

$$\tau_N(x_1,\dots,x_N) := \left(\frac{x_1}{\sqrt{1+x_1^2}},\dots,\frac{x_N}{\sqrt{1+x_N^2}}\right) \quad \in C_N\,. \tag{4.9}$$

Definition 4.34 (globally subanalytic)**.** *A subset S of $\mathbb{R}^N$ is called globally subanalytic if its image under τ_N is a subanalytic subset of $\mathbb{R}^N$.*

Both, the globally subanalytic sets and the subanalytic sets comprise the semi-algebraic sets [Cos00]. In fact the inclusion is proper. For example the function $\exp|_{[-1,1]}\colon [-1,1] \to \mathbb{R}$, the restriction of the exponential function to the domain $[-1,1]$, is globally subanalytic (hence subanalytic), however it is not semi-algebraic. Also the inclusion of globally subanalytic sets in the subanalytic sets is proper. Where all real analytic functions are always subanalytic [dDM96, Fact 1.1], they may fail to be globally subanalytic, as the next example shows.

Example 4.14. The graph of the sine function is subanalytic (even analytic), but not globally subanalytic. While the first coordinate of

$$\tau_N(x,\sin(x)) = \left(\frac{x}{\sqrt{1+x^2}},\frac{\sin(x)}{\sqrt{1+\sin^2(x)}}\right)^\top$$

tends to 1, as x tends to ∞, the second coordinate periodically changes between $\pm 1/\sqrt{2}$. In any neighborhood of the point $(1,0)^\top$ there are infinitely many connected components, therefore the sine function is not globally subanalytic.

The advantage of globally subanalytic sets is that they generate an o-minimal structure, whereas subanalytic function do not, as demonstrated in Example 4.13. The key for defining the o-minimal structure is provided by the projection theorem (Gabrielov [Gab96]).

We want to note that there is an even "larger" structure that comprises the globally subanalytic structure. The extension that this "larger" structure introduces is the exponential function $x \mapsto \exp(x)$ (thus, by Lemma 4.31 also the logarithm) [Wil96, dD98]. Naturally, the question arises, why not to take the largest structure that is available. The more information is known about the problem, the more properties can be used, and therefore, faster algorithms can be expected for problems cast in the appropriate class.

Example 4.15. Thanks to the stability results of o-minimal structures, the following functions are easily verified to be definable in one of the o-minimal structures introduced above. Let $K\colon \mathbb{R}^N \to \mathbb{R}$ be a linear operator, let $p := p_1/p_2 \in \mathbb{Q}$, $p_1, p_2 \in \mathbb{N}$, be a positive rational number, and $\lambda \in \mathbb{R}$. Recap that finite sums of such terms are automatically definable by Corollary 4.32.

(i) $x \mapsto |Kx|^p/(1+|Kx|^p)$ is definable since $\{(x,y) \in \mathbb{R}^N \times \mathbb{R} |\, y = Kx\}$, $\{(x,y) \in \mathbb{R}^2 |\, x^{p_1} = y^{p_2}\}$, $\{(x,y) \in \mathbb{R}^2 |\, y = x/(1+x)\} = \{(x,y) \in \mathbb{R}^2 |\, (1+x)y - x = 0\}$, $\{(x,y) \in \mathbb{R}^2 |\, y = |x|\}$ are semi-algebraic sets. See also Examples 4.10 and 4.11. (This function appears in the iterations of the iPiano algorithm for the application in Section 6.2.2.1.)

(ii) $x \mapsto \min(|Kx|^p, \lambda)$ is definable in the semi-algebraic structure, as it is piecewise algebraic—with finitely many pieces. (This example includes the Mumford–Shah regularizer [MS89], i.e., a truncated quadratic function.)

(iii) $c \mapsto \frac{1}{2}\|A^{-1}Cu^0 - u^0\|_2^2$ with $C = \operatorname{diag}(c)$, where A is a positive definite matrix that linearly depends on C, is semi-algebraic, since the norm function is semi-algebraic and the graph of the inner function is defined by a (possibly large) system of polynomial equations. (This function occurs in Section 6.2.2.2.)

(iv) $x \mapsto \log(1 + |Kx|^p) + \delta_{[-\lambda,\lambda]}(Kx)$, where $\delta_{[-\lambda,\lambda]}$ is an indicator function (Definition 2.19), is definable in the globally subanalytic structure, as it is a bounded subanalytic function (actually bounded analytic). (This function occurs in Section 7.5.)

(v) $x \mapsto \log(1+|Kx|^p)$ is definable in the $\exp$-extended globally subanalytic structure. (This function also appears in Section 7.5.)

Finally, we state a crucial result for the convergence of the algorithms presented in Chapters 6 and 7. The convergence analysis of these algorithms is based on the assumption that the objective function is a KL function. The following theorem makes sure that there is a huge class of functions that naturally are KL functions. We present the theorem in the formulation of [ABRS10, Thm. 14]. The proof can be found in [BDLS07, Thm. 14].

Theorem 4.35 (Nonsmooth Kurdyka–Łojasiewicz inequality for definable functions)**.** *Any proper lower semi-continuous function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ which is definable in an* o*-minimal structure $\mathcal{O}$ has the Kurdyka–Łojasiewicz property at each point of* $\operatorname{dom} \partial f$*. Moreover the function φ in Definition 4.25 is definable in $\mathcal{O}$.*

If we consider the o-minimal structure of globally subanalytic sets, a parametrized version of the Kurdyka–Łojasiewicz inequality holds. In [Kur98, Thm. ŁI] Kurdyka shows that for globally subanalytic (differentiable) functions $\varphi(s) = s^{1-\theta}$ may be chosen with $\theta \in (0, 1)$. We formulate as in [BDLS07, Cor. 16] for nonsmooth functions.

Corollary 4.36 (Nonsmooth Kurdyka–Łojasiewicz inequality for subanalytic functions)**.** *Any proper lower semi-continuous globally subanalytic function $f\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ has the parametrized KL property at each point of* $\operatorname{dom} \partial f$*, i.e., for $\hat{x} \in \operatorname{dom} \partial f$ there exist $\rho > 0$, $\theta \in [0, 1)$, and a continuous definable function $\chi\colon \mathbb{R}_+ \to (0, \infty)$ such that*

$$|f(x) - f(\hat{x})|^\theta \leq \rho\|\xi(x)\|\,,$$

whenever $0 < |f(x) - f(\hat{x})| \leq \chi(\|x\|)$ and $\xi(x) \in \partial f(x)$ (with the convention that $0^0 = 0$).

We conclude this chapter with an counterexample. If the function is definable but not globally subanalytic, Corollary 4.36 holds not true, in general.

Example 4.16. By Theorem 4.35 the function $f(x) = \exp(-1/x^2)$ has the KL property, since it is definable in the $\exp$-extended globally subanalytic structure. However, it does not allow for a parametrized representation, since for $x \to 0$ the expression

$$\frac{|f(x)|^\theta}{|f'(x)|} = \frac{1}{2}\, x^3 \exp\left(\frac{1-\theta}{x^2}\right)$$

is unbounded for any $\theta \in [0, 1)$.

Chapter 5

An abstract convergence theorem for descent methods

The Kurdyka–Łojasiewicz (KL) property has shown to be a powerful tool in the convergence analysis of nonsmooth nonconvex optimization algorithms. As we have seen in Section 4.5, a broad class of functions, e.g. tame function (includes semi-algebraic functions), reveals this property. Therefore, particularly for applications, assuming the objective function having this property should not be considered as a severe restriction.

It was successfully applied for example in [CPR13, AB09, THD09, FGP14, OCBP14, BC14, BST14, ABS13, ABRS10] to proof convergence of algorithms for nonconvex optimization problems. The proof of convergence involving KL features seems to follow some general steps, which is raised to a more abstract level in [ABS13]. Consider the minimization problem of a proper, lower semi-continuous (lsc) function $F' \colon \mathbb{R}^N \to \overline{\mathbb{R}}$ and let $(x^n)_{n\in\mathbb{N}}$ be a sequence of iterates generated by some algorithm. Convergence of this sequence to an optimum is shown, if it satisfies the following properties ($a, b > 0$ fixed):

(H1′) (Sufficient decrease condition). For each $n \in \mathbb{N}$,

$$F'(x^{n+1}) + a\|x^{n+1} - x^n\|^2 \leq F'(x^n)\,;$$

(H2′) (Relative error condition). For each $n \in \mathbb{N}$, there exists $w^{n+1} \in \partial F'(x^{n+1})$ such that

$$\|w^{n+1}\| \leq b\|x^{n+1} - x^n\|\,;$$

(H3′) (Continuity condition). There exists a subsequence $(x^{n_j})_{j\in\mathbb{N}}$ and $\tilde{x}$ such that

$$x^{n_j} \to \tilde{x} \quad \text{and} \quad F'(x^{n_j}) \to F'(\tilde{x})\,, \quad \text{as } j \to \infty\,.$$

The convergence result that is obtained in [ABS13] can be used to easily verify convergence of several algorithms. Explicitly, it is shown for an inexact gradient descent method, inexact proximal algorithm, inexact forward–backward algorithm, and an inexact regularized Gauss-Seidel method.

Theorem 5.1 (Convergence of descent methods)**.** *Let $f \colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a proper lower semi-continuous function. Consider a sequence $(x^n)_{n\in\mathbb{N}}$ that satisfies (H1′), (H2′), and (H3′).*

If f has the Kurdyka–Łojasiewicz property at the cluster point $\tilde{x}$ specified in (H3′), then the sequence $(x^n)_{n\in\mathbb{N}}$ converges to $\bar{x} = \tilde{x}$ as n goes to infinity, and $\bar{x}$ is a critical point of f.

Moreover the sequence $(x^n)_{n\in\mathbb{N}}$ has a finite length, i.e.,

$$\sum_{n=0}^{\infty} \|x^{n+1} - x^n\| \leq \infty\,.$$

In Chapter 7, we make also use of this result to show convergence of our majorization minimization algorithm. A limiting fact of these algorithms is that the computation of the next iterate is based only on the current one; No effects of over-relaxation or inertness are used. In this chapter, we modify the abstract requirements (H1′)–(H3′) such that convergence of our Heavy-ball like algorithm (iPiano; see Chapter 6), which is a two-step method, can be shown. The analysis is close to that in [ABS13]. Large parts of this chapter are published in [OCBP14].

5.1 Inexact descent convergence result for KL functions

In the following, we prove an abstract convergence result for a sequence $(z^n)_{n\in\mathbb{N}} := (x^n, x^{n-1})_{n\in\mathbb{N}}$ in $\mathbb{R}^{2N}$, $x^n \in \mathbb{R}^N$, $x^{-1} \in \mathbb{R}^N$, satisfying certain basic conditions, $\mathbb{N} := \{0, 1, 2, \ldots\}$. For convenience we use the abbreviation $\Delta_n := \|x^n - x^{n-1}\|$ for $n \in \mathbb{N}$. We fix two positive constants $a > 0$ and $b > 0$ and consider a proper lower semi-continuous function $F\colon \mathbb{R}^{2N} \to \overline{\mathbb{R}}$. Then, the conditions we require for $(z^n)_{n\in\mathbb{N}}$ are as follows:

(H1) For each $n \in \mathbb{N}$, it holds that

$$F(z^{n+1}) + a\Delta_n^2 \leq F(z^n)\,.$$

(H2) For each $n \in \mathbb{N}$, there exists $w^{n+1} \in \partial F(z^{n+1})$ such that

$$\|w^{n+1}\| \leq \frac{b}{2}(\Delta_n + \Delta_{n+1})\,.$$

(H3) There exists a subsequence $(z^{n_j})_{j\in\mathbb{N}}$ such that

$$z^{n_j} \to \tilde{z} \quad \text{and} \quad F(z^{n_j}) \to F(\tilde{z})\,, \qquad \text{as } j \to \infty\,.$$

Remark 5.1. Our proof and the proof in [ABS13] differ mainly in the calculations that are involved; the outline is the same. There is hope to find an even more general convergence result, which comprises ours and [ABS13].

Lemma 5.2. *Let $F\colon \mathbb{R}^{2N} \to \overline{\mathbb{R}}$ be a proper lower semi-continuous function which satisfies the Kurdyka–Łojasiewicz property at some point $z^* = (x^*, x^*) \in \mathbb{R}^{2N}$. Denote by U, η and $\varphi\colon [0, \eta) \to \mathbb{R}_+$ the objects appearing in Definition 4.25 of the KL property at z^*. Let $\sigma, \rho > 0$ be such that $B(z^*, \sigma) \subset U$ with $\rho \in (0, \sigma)$, where $B(z^*, \sigma) := \{z \in \mathbb{R}^{2N} : \|z - z^*\| < \sigma\}$.*

Furthermore, let $(z^n)_{n\in\mathbb{N}} = (x^n, x^{n-1})_{n\in\mathbb{N}}$ be a sequence satisfying Conditions (H1), (H2), and

$$\forall n \in \mathbb{N}: \quad z^n \in B(z^*, \rho) \Rightarrow z^{n+1} \in B(z^*, \sigma) \text{ with } F(z^{n+1}), F(z^{n+2}) \geq F(z^*)\,. \tag{5.1}$$

Moreover, the initial point $z^0 = (x^0, x^{-1})$ is such that $F(z^*) \leq F(z^0) < F(z^*) + \eta$ and

$$\|x^* - x^0\| + \sqrt{\frac{F(z^0) - F(z^*)}{a}} + \frac{b}{a}\varphi(F(z^0) - F(z^*)) < \frac{\rho}{2}\,. \tag{5.2}$$

Then, the sequence $(z^n)_{n\in\mathbb{N}}$ *satisfies*

$$\forall n \in \mathbb{N} : z^n \in B(z^*, \rho), \quad \sum_{n=0}^{\infty} \Delta_n < \infty, \quad F(z^n) \to F(z^*), \text{ as } n \to \infty\,, \tag{5.3}$$

$(z^n)_{n\in\mathbb{N}}$ *converges to a point* $\bar{z} = (\bar{x}, \bar{x}) \in B(z^*, \sigma)$ *such that* $F(\bar{z}) \leq F(z^*)$. *If, additionally, Condition (H3) is satisfied, then* $0 \in \partial F(\bar{z})$ *and* $F(\bar{z}) = F(z^*)$.

Proof. The key points of the proof are the facts that for all $j \geq 1$:

$$z^j \in B(z^*, \rho) \qquad \text{and} \tag{5.4}$$

$$\sum_{i=1}^{j} \Delta_i \leq \frac{1}{2}(\Delta_0 - \Delta_j) + \frac{b}{a}[\varphi(F(z^1) - F(z^*)) - \varphi(F(z^{j+1}) - F(z^*)))] \tag{5.5}$$

Let us first see that $\varphi(F(z^{j+1}) - F(z^*))$ is well defined. By Condition (H1), $(F(z^n))_{n\in\mathbb{N}}$ is non-increasing, which shows that $F(z^{n+1}) \leq F(z^0) < F(z^*) + \eta$. Combining this with (5.1) implies $F(z^{n+1}) - F(z^*) \geq 0$.

As for $n \geq 1$ the set $\partial F(z^n)$ is nonempty (see Condition (H2)) every z^n belongs to $\operatorname{dom} F$. For notational convenience, we define

$$D_n^{\varphi} := \varphi(F(z^n) - F(z^*)) - \varphi(F(z^{n+1}) - F(z^*))\,.$$

Now, we want to show that for $n \geq 1$ the following holds: if $F(z^n) < F(z^*) + \eta$ and $z^n \in B(z^*, \rho)$, then

$$2\Delta_n \leq \tfrac{b}{a} D_n^{\varphi} + \tfrac{1}{2}(\Delta_n + \Delta_{n-1})\,. \tag{5.6}$$

Obviously, we can assume that $\Delta_n \neq 0$ (otherwise it is trivial), and therefore (H1) and (5.1) imply $F(z^n) > F(z^{n+1}) \geq F(z^*)$. The KL inequality shows $w^n \neq 0$, and (H2) shows $\Delta_n + \Delta_{n-1} > 0$. Since $w^n \in \partial F(z^n)$, using the KL inequality and (H2), we obtain

$$\varphi'(F(z^n) - F(z^*)) \geq \frac{1}{\|w^n\|} \geq \frac{2}{b(\Delta_{n-1} + \Delta_n)}\,.$$

As φ is concave and increasing ($\varphi' > 0$), Condition (H1) and (5.1) yield

$$D_n^{\varphi} \geq \varphi'(F(z^n) - F(z^*))(F(z^n) - F(z^{n+1})) \geq \varphi'(F(z^n) - F(z^*))a\Delta_n^2\,.$$

Combining both inequalities results in

$$(\tfrac{b}{a} D_n^{\varphi})\tfrac{1}{2}(\Delta_{n-1} + \Delta_n) \geq \Delta_n^2\,,$$

which by applying $2\sqrt{uv} \leq u + v$ establishes (5.6).

As (5.1) only implies $z^{n+1} \in B(z^*, \sigma)$, $\sigma > \rho$, we cannot use (5.6) directly for the whole sequence. However, (5.4) and (5.5) can be shown by induction on j. For $j = 0$, (5.1) yields $z^1 \in B(z^*, \sigma)$ and $F(z^1), F(z^2) \geq F(z^*)$. From Condition (H1) with $n = 1$, $F(z^2) \geq F(z^*)$ and $F(z^1) \leq F(z^0)$, we infer

$$\Delta_1 \leq \sqrt{\frac{F(z^1) - F(z^2)}{a}} \leq \sqrt{\frac{F(z^0) - F(z^*)}{a}}\,, \tag{5.7}$$

which combined with (5.2) leads to

$$\|x^* - x^1\| \leq \|x^0 - x^*\| + \Delta_1 \leq \|x^0 - x^*\| + \sqrt{\frac{F(z^0) - F(z^*)}{a}} < \frac{\rho}{2}\,,$$

and therefore $z^1 \in B(z^*, \rho)$. Direct use of (5.6) with $n = 1$ shows that (5.5) holds with $j = 1$.

Suppose (5.4) and (5.5) are satisfied for $j \geq 1$. Then, using the triangle inequality and (5.5), we have

$$\begin{aligned} \|z^* - z^{j+1}\| &\leq \|x^* - x^{j+1}\| + \|x^* - x^j\| \\ &\leq 2\|x^* - x^0\| + 2\textstyle\sum_{i=1}^{j} \Delta_i + \Delta_{j+1} \\ &\leq 2\|x^* - x^0\| + (\Delta_0 - \Delta_j) + \Delta_{j+1} \\ &\quad 2\tfrac{b}{a}[\varphi(F(z^1) - F(z^*)) - \varphi(F(z^{j+1}) - F(z^*)))] \\ &\leq 2\|x^* - x^0\| + \Delta_0 + \Delta_{j+1} + 2\tfrac{b}{a}[\varphi(F(z^0) - F(z^*))]\,, \end{aligned}$$

which shows, using $\Delta_{j+1} \leq \sqrt{\frac{1}{a}(F(z^{j+1}) - F(z^{j+2}))} \leq \sqrt{\frac{1}{a}(F(z^0) - F(z^*))}$ and (5.2), that $z^{j+1} \in B(z^*, \rho)$. As a consequence (5.6), with $n = j + 1$, can be added to (5.5) and we can conclude (5.5) with $j + 1$. This shows the desired induction on j.

Now, the finiteness of the length of the sequence $(x^n)_{n\in\mathbb{N}}$, i.e., $\sum_{i=1}^{\infty} \Delta_i < \infty$, is a consequence of the following estimation, which is implied by (5.5):

$$\sum_{i=1}^{j} \Delta_i \leq \tfrac{1}{2}\Delta_0 + \tfrac{b}{a}\varphi(F(z^1) - F(z^*)) < \infty\,.$$

Therefore, x^n converges to some $\bar{x}$ as $n \to \infty$, and z^n converges to $\bar{z} = (\bar{x}, \bar{x})$. As φ is concave, φ' is decreasing. Using this and Condition (H2) yields $w^n \to 0$ and $F(z^n) \to \zeta \geq F(z^*)$. Suppose we have $\zeta > F(z^*)$, then the KL inequality reads $\varphi'(\zeta - F(z^*))\|w^n\| \geq 1$ for all $n \geq 1$, which contradicts $w^n \to 0$.

Note that, in general, $\bar{z}$ is not a critical point of F, because the limiting subdifferential requires $F(z^n) \to F(\bar{z})$ as $n \to \infty$. When the sequence $(z^n)_{n\in\mathbb{N}}$ additionally satisfies Condition (H3), then $\tilde{z} = \bar{z}$, and $\bar{z}$ is a critical point of F, because $F(\bar{z}) = \lim_{n\to\infty} F(z^n) = F(z^*)$. □

Remark 5.2. The only difference from [ABS13] with respect to the assumptions is (5.1). In [ABS13], $z^n \in B(z^*, \rho)$ implies $F(z^{n+1}) \geq F(z^*)$, whereas we require $F(z^{n+1}) \geq F(z^*)$ and $F(z^{n+2}) \geq F(z^*)$. However, as Theorem 5.4 shows, this does not weaken the convergence result compared to [ABS13]. In fact, Corollary 5.3, which assumes $F(z^n) \geq F(z^*)$ for all $n \in \mathbb{N}$ and which is also used in [ABS13], is key in Theorem 5.4.

The next corollary and the subsequent theorem follow as in [ABS13] by replacing the calculation with our conditions.

Corollary 5.3. *Lemma 5.2 holds true if we replace* (5.1) *by*

$$\eta < a(\sigma - \rho)^2 \qquad \textit{and} \qquad F(z^n) \geq F(z^*)\ \forall n \in \mathbb{N}\,.$$

Proof. By Condition (H1), for $z^n \in B(z^*, \rho)$, we have

$$\Delta_{n+1}^2 \leq \frac{F(z^{n+1}) - F(z^{n+2})}{a} \leq \frac{\eta}{a} < (\sigma - \rho)^2\,.$$

Using the triangle inequality on $\|z^{n+1} - z^*\|$ shows that $z^{n+1} \in B(z^*, \sigma)$, which implies (5.1) and concludes the proof. □

The work that is done in Lemma 5.2 and Corollary 5.3 allows us to formulate an abstract convergence theorem for sequences satisfying Conditions (H1), (H2), and (H3). It follows, with a few modifications, as in [ABS13].

Theorem 5.4 (Convergence to a critical point)**.** *Let $F\colon \mathbb{R}^{2N} \to \overline{\mathbb{R}}$ be a proper lower semi-continuous function and $(z^n)_{n\in\mathbb{N}} = (x^n, x^{n-1})_{n\in\mathbb{N}}$ a sequence that satisfies (H1), (H2), and (H3). Moreover, let F have the Kurdyka–Łojasiewicz property at the cluster point $\tilde{x}$ specified in (H3).*

Then, the sequence $(x^n)_{n=0}^{\infty}$ has finite length, i.e., $\sum_{n=1}^{\infty} \Delta_n < \infty$, and converges to $\bar{x} = \tilde{x}$ as $n \to \infty$, where $(\bar{x}, \bar{x})$ is a critical point of F.

Proof. By Condition (H3), we have $z^{n_j} \to \bar{z} = \tilde{z}$ and $F(z^{n_j}) \to F(\bar{z})$ for a subsequence $(z^{n_j})_{n\in\mathbb{N}}$. This together with the nondecreasingness of $(F(z^n))_{n\in\mathbb{N}}$ (by Condition (H1)), imply that $F(z^n) \to F(\bar{z})$ and $F(z^n) \geq F(\bar{z})$ for all $n \in \mathbb{N}$. The KL property around $\bar{z}$ states the existence of quantities φ, U, and η as in Definition 4.25. Let $\sigma > 0$ be such that $B(\bar{z}, \sigma) \subset U$ and $\rho \in (0, \sigma)$. Shrink η such that $\eta < a(\sigma - \rho)^2$ (if necessary). As φ is continuous, there exists $n_0 \in \mathbb{N}$ such that $F(z^n) \in [F(\bar{z}), F(\bar{z}) + \eta)$ for all $n \geq n_0$ and

$$\|x^* - x^{n_0}\| + \sqrt{\frac{F(z^{n_0}) - F(z^*)}{a}} + \frac{b}{a}\varphi(F(z^{n_0}) - F(z^*)) < \frac{\rho}{2}\,.$$

Then, the sequence $(y^n)_{n\in\mathbb{N}}$ defined by $y^n = z^{n_0+n}$ satisfies the conditions in Corollary 5.3, which concludes the proof. □

Chapter 6

iPiano: inertial proximal algorithm for nonconvex optimization

The gradient method is certainly one of the most fundamental but also one of the most simple algorithms to solve smooth convex optimization problems. In the last several decades, the gradient method has been modified in many ways. One of those improvements are multistep schemes, which we discussed in Sections 2.2.4 and 2.2.5. It has been shown that such schemes significantly boost the performance of the plain gradient method. Triggered by practical problems in signal processing, image processing and machine learning, there has been an increased interest in so-called composite objective functions, where the objective function is given by the sum of a smooth function and a nonsmooth function with an easy-to-compute proximal map. This initiated the development of the proximal gradient or forward–backward method (see Section 2.2.3), which combines explicit (forward) gradient steps w.r.t. the smooth part with proximal (backward) steps w.r.t. the nonsmooth part.

In this chapter, we combine the concepts of multistep schemes and the proximal gradient method to efficiently solve a certain class of nonconvex, nonsmooth optimization problems. Although, the transfer of knowledge from convex optimization to nonconvex problems is very challenging, it aspires to find efficient algorithms for certain nonconvex problems. Therefore, we consider the subclass of nonconvex problems

$$\min_{x \in \mathbb{R}^N} \; f(x) + g(x)\,,$$

where g is *simple (possibly nonsmooth and nonconvex)* and f is a *smooth (possibly nonconvex)* function. The sum $f + g$ comprises nonsmooth, nonconvex functions. Despite the nonconvexity, the structure of f being smooth and g being simple makes the forward–backward splitting algorithm well defined (cf. [BST14, Prop. 2.2]). Additionally, an inertial force is incorporated into the design of our algorithm, which we termed *iPiano*. Informally, the update scheme of the algorithm that will be analyzed is

$$x^{n+1} \in \arg\min_x \; g(x) + \langle \nabla f(x^n), x - x^n \rangle + \frac{1}{2\alpha} \|x - (x^n + \beta(x^n - x^{n-1}))\|^2\,,$$

which, for g being convex, is equivalent to[1]

$$x^{n+1} = (I + \alpha \partial g)^{-1}(x^n - \alpha \nabla f(x^n) + \beta(x^n - x^{n-1}))\,,$$

[1] In general the set of minimizers is only contained in the proximal map, $\arg\min_x g(x) + \frac{1}{2\alpha}\|x - \tilde{x}\|^2 \subset (I + \alpha\partial g)^{-1}(\tilde{x})$ (see [RW98, Ex. 10.2]).

where α and β are the step size parameters. This update rule is a generalization of the proximal Heavy-ball method from Section 2.2.4 to nonconvex objective functions.

Setting $\beta = 0$ results in the forward–backward splitting algorithm, which has the nice property that in each iteration the function value decreases. Our convergence analysis reveals that the additional inertial term prevents our algorithm from monotonically decreasing the function values. Although this may look like a limitation on first glance, demanding monotonically decreasing function values anyway is too strict as it does not allow for provably optimal schemes. We refer to a statement of Nesterov [Nes04]: "In convex optimization the optimal methods never rely on relaxation. Firstly, for some problem classes this property is too expensive. Secondly, the schemes and efficiency estimates of optimal methods are derived from some global topological properties of convex functions"[2]. The negative side of better efficiency estimates of an algorithm is usually the convergence analysis. This is even true for convex functions. In case of nonconvex and nonsmooth functions, this problem becomes even more severe.

Previous work. This chapter generalizes the results presented [OCBP14]. Where in [OCBP14] we considered the sum of a smooth function f and a convex function g, in this chapter, we allow g to be nonconvex. The only requirement is that the associated proximal map can be solved efficiently for the global optimum. As this extension does not encounter severe challenges, we did not publish it separately (only in this thesis). The disadvantage of the generality is that the step size parameters are more restricted. In order to resolve this problem, we extend the convergence analysis in a way that more structure of g is directly reflected in better step size parameters. The analysis also covers the benefit when g is even strongly convex. Another extension of the results in [OCBP14] would be to replace the proximity function (which is the squared Euclidean distance here), with a Bregman proximity function. As a Bregman proximity function is assumed to be bounded from below and from above with a squared Euclidean distance the extension is straightforward. The full nonconvex setting with Bregman proximity function is addressed in [BCL14]. Other related work was already discussed in Section 2.2.4.

6.1 The proposed algorithm: iPiano

6.1.1 The optimization problem

We consider a structured nonsmooth, nonconvex optimization problem with a proper lower semi-continuous extended valued function $h\colon \mathbb{R}^N \to \overline{\mathbb{R}}$, $N \geq 1$:

$$\min_{x\in\mathbb{R}^N} \; h(x) = \min_{x\in\mathbb{R}^N} \; f(x) + g(x)\,. \tag{6.1}$$

The function $f\colon \mathbb{R}^N \to \mathbb{R}$ is assumed to be C^1-smooth (possibly nonconvex) with L-Lipschitz continuous gradient on $\operatorname{dom} g$, $L > 0$. Further, let the function $g\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be simple (possibly nonsmooth and nonconvex) and prox-bounded, i.e., there exists $\lambda > 0$ such that $e_\lambda g(x) := \inf_{y\in\mathbb{R}^N} g(y) + \frac{1}{2\lambda}\|y-x\|^2 > -\infty$ for some $x \in \mathbb{R}^N$. Simple refers to the fact that the associated proximal map can be solved efficiently for the global optimum. Furthermore, we require h to be coercive and bounded from below by some value $\underline{h} > -\infty$.

The proposed algorithm, which is stated in Section 6.1.2, seeks a critical point $x^* \in \operatorname{dom} h$ of h, which is characterized by the necessary first-order optimality condition $0 \in \partial h(x^*)$. In our case, this is equivalent to

$$-\nabla f(x^*) \in \partial g(x^*)\,,$$

[2]Relaxation is to be interpreted as the property of monotonically decreasing function values in this context. Topological properties should be associated with geometrical properties.

where ∂g denotes the limiting subdifferential in Definition 4.12.

In order to take advantageous of additional knowledge about g, like convexity, we extend the idea of the strong convexity modulus (Definition 2.45) to nonconvex functions.

Definition 6.1 (semi-convex)**.** *A function $g\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ is said to be semi-convex with modulus $m \in \mathbb{R}$ (or is m semi-convex), if m is the largest value such that $g(x) - \frac{m}{2}\|x\|^2$ is convex.*

Remark 6.1. We do not assume g to be m semi-convex. However, under some conditions, m semi-convexity allows us to choose larger step sizes.

Analogously to Lemma 2.46, we observe the following lemma:

Lemma 6.2. *Let $g\colon \mathbb{R}^N \to \overline{\mathbb{R}}$ be a semi-convex function with modulus $m \in \mathbb{R}$. Then, for any $\bar{x} \in \operatorname{dom} \partial g$ it holds that*

$$g(x) \geq g(\bar{x}) + \langle v, x - \bar{x}\rangle + \frac{m}{2}\|x - \bar{x}\|^2\,, \quad \text{for all } v \in \partial g(\bar{x}) \text{ and } x \in \operatorname{dom} g\,.$$

Proof. Rearrange the subgradient inequality (2.1) applied to $g(x) - (m/2)\|x\|^2$ and make use of Proposition 4.13(iii). □

6.1.2 The generic algorithm

The generic formulation of iPiano is shown in Algorithm 2. It is a forward–backward splitting algorithm incorporating an inertial force. In the forward step, α_n determines the step size in the direction of the gradient of the differentiable function f. The step in gradient direction is aggregated with the inertial force from the previous iteration weighted by β_n. Then, the backward step is a solution of the (generalized) proximity operator for the function g with the weight α_n. The prox-boundedness of g asserts that the

Algorithm 2. ***I**nertial **p**rox**i**mal **a**lgorithm for **n**onconvex **o**ptimization (iPiano)*

- *Initialization: Choose a starting point $x^0 \in \operatorname{dom} h$ and set $x^{-1} = x^0$. Moreover, define sequences of step size parameter $(\alpha_n)_{n\in\mathbb{N}}$ and $(\beta_n)_{n\in\mathbb{N}}$.*
- *Iterations $(n \geq 0)$: Update*

$$\begin{aligned} y^n &= x^n + \beta_n(x^n - x^{n-1}) \\ x^{n+1} &\in \operatorname*{arg\,min}_x\ g(x) + \langle \nabla f(x^n), x - x^n\rangle + \frac{1}{2\alpha_n}\|x - y^n\|^2 \end{aligned} \tag{6.2}$$

set of minimizers of (6.2) is nonempty and compact ([RW98, Thm. 1.25]) and makes the algorithm well-defined.

Note that the update step (6.2) is equivalent to

$$\begin{aligned} x^{n+1} &\in \operatorname*{arg\,min}_{x\in\mathbb{R}^N}\ G^n(x) \\ G^n(x) &:= g(x) + \left\langle \nabla f(x^n) - \frac{\beta_n}{\alpha_n}(x^n - x^{n-1}), x - x^n \right\rangle + \frac{1}{2\alpha_n}\|x - x^n\|^2 \end{aligned} \tag{6.3}$$

and for g being convex equivalent to

$$\begin{aligned} y^n &= x^n + \beta_n(x^n - x^{n-1}) \\ x^{n+1} &= (I + \alpha_n \partial g)^{-1}(y^n - \alpha_n \nabla f(x^n)) \,. \end{aligned} \tag{6.4}$$

In order to make the algorithm specific and convergent, the step size parameters must be chosen appropriately. What "appropriately" means will be specified in Section 6.1.3 and proved in Section 6.1.4.

6.1.3 Rules for choosing the step size

In this subsection, we propose several strategies for choosing the step sizes. This will make it easier to implement the algorithm. One may choose among the following variants of step size rules depending on the knowledge about the objective function. We introduce a flag variable $\sigma \in \{0, 1\}$ to treat the case of g being semi-convex with modulus $m \in \mathbb{R}$ ($\sigma = 1$) and g not being semi-convex ($\sigma = 0$) at the same time. Moreover, if $\sigma = 1$ (i.e. g semi-convex), we focus on the case $m < L_n$, otherwise h would be convex. This can easily be seen by applying Lemma 6.2 to g, the Descent Lemma 2.49 to f, and summing both inequalities. As in Algorithm 2, we choose $x^0 \in \operatorname{dom} h$ and set $x^{-1} = x^0$.

Constant step size scheme. The most simple step size scheme, which requires most knowledge about the objective function, is outlined in Algorithm 3. All step size parameters are chosen a priori and are constant.

Algorithm 3. ***I**nertial **p**rox**i**mal **a**lgorithm for **n**onconvex **o**ptimization with constant parameter (ciPiano)*

- *Initialization: Choose $\beta \in [0, \frac{1+\sigma}{2})$, set $\alpha < (1+\sigma-2\beta)/(L-\sigma m)$, where L is the Lipschitz constant of ∇f.*
- *Iterations ($n \geq 0$): Update x^n using (6.2) with $\alpha_n = \alpha$ and $\beta_n = \beta$.*

Remark 6.2. Observe that our law on α, β is equivalent to the law found in [ZK93] for minimizing a smooth nonconvex function. Hence, our result can be seen as an extension of their work to the presence of an additional nonsmooth simple function.

Backtracking. The case where we have limited knowledge about the objective function occurs more frequently. It can be very challenging to estimate the Lipschitz constant of ∇f beforehand. Using backtracking the Lipschitz constant can be estimated automatically. A sufficient condition that the Lipschitz constant at iteration n to $n+1$ must satisfy is

$$f(x^{n+1}) \leq f(x^n) + \left\langle \nabla f(x^n), x^{n+1} - x^n \right\rangle + \frac{L_n}{2} \|x^{n+1} - x^n\|^2 \,. \tag{6.5}$$

Although there are different strategies for determining L_n, the most common one is to define an increment variable $\eta > 1$ and to look for the minimal $L_n \in \{L_{n-1}, \eta L_{n-1}, \eta^2 L_{n-1}, \ldots\}$ satisfying (6.5). Sometimes, it is also feasible to decrease the estimated Lipschitz constant. A possible strategy is as follows: if $L_n = L_{n-1}$, then search for the minimal $L_n \in \{\eta^{-1} L_{n-1}, \eta^{-2} L_{n-1}, \ldots\}$ satisfying (6.5).

In Algorithm 4 we propose an algorithm with variable step sizes. Any strategy for estimating the Lipschitz constant may be used. When changing the Lipschitz constant from one iteration to another, all step size parameters must be adapted. The rules for adapting the step sizes will be justified during the convergence analysis in Section 6.1.4.

Algorithm 4. ***I**nertial **p**roxi**m**al **a**lgorithm for **n**onconvex **o**ptimization with backtracking (biPiano)*

- *Initialization: Choose $\delta \geq c_2 > 0$ with c_2 close to 0 (e.g. $c_2 := 10^{-6}$).*
- *Iterations ($n \geq 0$): Update x^n using (6.2), where $L_n > 0$ satisfies (6.5) and*

$$\beta_n = \frac{1+\sigma}{2}\frac{b-1}{b-\frac{1}{2}}, \qquad b := \frac{\delta + \frac{L_n - \sigma m}{2}}{c_2 + \frac{L_n - \sigma m}{2}}, \qquad \alpha_n = \frac{1+\sigma-2\beta_n}{L_n - \sigma m + 2c_2}.$$

Lazy backtracking. Algorithm 5 presents another alternative to Algorithm 2. It is related to Algorithms 3 and 4 in the following way. Algorithm 5 makes use of the Lipschitz continuity of ∇f in the sense that the Lipschitz constant is always finite. As a consequence, using backtracking with only increasing Lipschitz constants, after a finite number of iterations $n_0 \in \mathbb{N}$ the estimated Lipschitz constant will no longer change, and starting from this iteration the constant step size rules as in Algorithm 3 are applied. Using this strategy, the results that will be proved in the convergence analysis are satisfied only as soon as the Lipschitz constant is high enough and no longer changing.

Algorithm 5. *Nonmonotone **i**nertial **p**roxi**m**al **a**lgorithm for **n**onconvex **o**ptimization with backtracking (nmiPiano)*

- *Initialization: Choose $\beta \in [0, \frac{1+\sigma}{2})$, $L_{-1} > 0$, $\eta > 1$.*
- *Iterations ($n \geq 0$): Update x^n using (6.2), where $L_n \in \{L_{n-1}, \eta L_{n-1}, \eta^2 L_{n-1}, \ldots\}$ is minimal and satisfies (6.5) and $\alpha_n < (1+\sigma-2\beta)/(L_n - \sigma m)$.*

General rule of choosing the step sizes. Algorithm 6 defines the general rules that the step size parameters must satisfy. It contains the Algorithms 3 to 5 as special instances. This is easily verified for Algorithms 3 and 5. For Algorithm 4 the step size rules are derived from the proof of Lemma 6.3.

As Algorithm 6 is the most general algorithm, let us now analyze its behavior.

6.1.4 Convergence analysis

In all of what follows, let $(x^n)_{n\in\mathbb{N}}$ be the sequence generated by Algorithm 6 and with parameters satisfying the algorithm's requirements. Furthermore, for more convenient notation we abbreviate $H_\delta(x,y) := h(x) + \delta\|x-y\|^2$, $\delta \in \mathbb{R}$. Note, that for $x = y$ it is $H_\delta(x,y) = h(x)$. Moreover, we introduce a flag variable $\sigma \in \{0,1\}$ to treat the case of g being semi-convex ($\sigma = 1$) and g not being semi-convex ($\sigma = 0$) at the same time, as in the preceding section.

Let us first verify that the algorithm makes sense. We have to show that the requirements for the parameters are not contradictory, i.e., that it is possible to choose a feasible set of parameters. In the following lemma, we will only show the existence of such a parameters set. However, the proof helps us to formulate specific step size rules.

Lemma 6.3. *For all $n \geq 0$, in either case, $m \geq 2c_2 + L_n$ and $m < 2c_2 + L_n$, there exists parameters $\delta_n \geq \gamma_n$, β_n, and α_n such that, given $L_n > 0$, the requirements in Algorithm 6 are satisfied (i.e., $\alpha_n \geq c_1$, $\beta_n \geq 0$, $\delta_n \geq \gamma_n \geq c_2$, and $(\delta_n)_{n\in\mathbb{N}}$ monotonically decreasing).*

Algorithm 6. ***I**nertial **p**roxi**m**al **a**lgorithm for **n**onconvex **o**ptimization (iPiano)*

- *Initialization: Choose $c_1, c_2 > 0$ close to 0, $x^0 \in \operatorname{dom} h$ and set $x^{-1} = x^0$.*
- *Iterations ($n \geq 0$): Update x^n using (this is the same as (6.2))*

$$\begin{aligned} y^n &= x^n + \beta_n(x^n - x^{n-1}) \\ x^{n+1} &\in \arg\min_x \; g(x) + \langle \nabla f(x^n), x - x^n \rangle + \frac{1}{2\alpha_n}\|x - y^n\|^2 \end{aligned} \tag{6.6}$$

where $L_n > 0$ is the local Lipschitz constant satisfying

$$f(x^{n+1}) \leq f(x^n) + \left\langle \nabla f(x^n), x^{n+1} - x^n \right\rangle + \frac{L_n}{2}\|x^{n+1} - x^n\|^2 , \tag{6.7}$$

and $\alpha_n \geq c_1$, $\beta_n \geq 0$ are chosen such that $\delta_n \geq \gamma_n \geq c_2$, which are given,

- *if g is semi-convex with modulus $m \in \mathbb{R}$, by*

$$\gamma_n = \frac{1}{2}\left(\frac{2 - 2\beta_n}{\alpha_n} - (L_n - m)\right) \quad \text{and} \quad \delta_n = \gamma_n + \frac{\beta_n}{2\alpha_n} , \tag{6.8}$$

- *and in the general case, by*

$$\gamma_n = \frac{1}{2}\left(\frac{1 - 2\beta_n}{\alpha_n} - L_n\right) \quad \text{and} \quad \delta_n = \gamma_n + \frac{\beta_n}{2\alpha_n} , \tag{6.9}$$

and $(\delta_n)_{n\in\mathbb{N}}$ is monotonically decreasing.

Proof. By the algorithm's requirements it is $\delta_n \geq \gamma_n$. The upper bound for β_n and α_n come from rearranging $\gamma_n \geq c_2$.

First, we consider the nontrivial case $\sigma m \leq L_n + 2c_2$. We observe

$$\beta_n \leq \frac{1+\sigma}{2} - \alpha_n \frac{2c_2 + L_n - \sigma m}{2} \leq \frac{1+\sigma}{2} \quad \text{and} \quad \alpha_n \leq \frac{1 + \sigma - 2\beta_n}{2c_2 + L_n - \sigma m} . \tag{6.10}$$

The last statement follows by incorporating the descent property of δ_n. Let $\delta_{-1} \geq c_2$ be chosen initially. Then, the descent property of $(\delta_n)_{n\in\mathbb{N}}$ requires one of the equivalent statements

$$\delta_{n-1} \geq \delta_n \quad \Leftrightarrow \quad \delta_{n-1} \geq \frac{1 + \sigma - \beta_n}{2\alpha_n} - \frac{L_n - \sigma m}{2} \quad \Leftrightarrow \quad \alpha_n \geq \frac{1 + \sigma - 2\beta_n}{L_n - \sigma m + 2\delta_{n-1}} \tag{6.11}$$

to be true. The only thing that remains to show is that there exist $\alpha_n > c_1$ and $\beta_n \in [0, \frac{1+\sigma}{2})$ such that the relations in (6.10) and (6.11) are fulfilled. Consider the condition for a nonnegative gap between the upper and lower bounds for α_n:

$$\frac{1 + \sigma - 2\beta_n}{L_n - \sigma m + 2c_2} - \frac{1 + \sigma - 2\beta_n}{L_n - \sigma m + 2\delta_{n-1}} \geq 0 \quad \Leftrightarrow \quad \frac{\delta_{n-1} + \frac{L_n - \sigma m}{2}}{c_2 + \frac{L_n - \sigma m}{2}} \geq \frac{1 + \sigma - \beta_n}{1 + \sigma - 2\beta_n} .$$

Defining $b := (\delta_{n-1} + \frac{L_n - \sigma m}{2})/(c_2 + \frac{L_n - \sigma m}{2}) \geq 1$, it is easily verified that there exists $\beta_n \in [0, \frac{1+\sigma}{2})$ satisfying the equivalent condition

$$\frac{1+\sigma}{2}\frac{b-1}{b-\frac{1}{2}} \geq \beta_n . \tag{6.12}$$

As a consequence, the existence of a feasible α_n follows, and the descent property for δ_n holds.

Now, we consider the case $2c_2 + L_n - \sigma m \leq 0$, i.e., $\sigma m \geq L_n + 2c_2$, which can only arise if $\sigma = 1$. If we choose $\beta_n = \beta \in [0, \frac{1+\sigma}{2})$, the requirement $\gamma_n \geq c_2$ is satisfied for any $\alpha_n \geq c_1$. The monotone decreasingness of $(\delta_n)_{n\in\mathbb{N}}$ is fulfilled, for example, when choosing $2\delta_0 > \max_{n\in\mathbb{N}} L_n - m \geq c_2$ and setting $\alpha_n = (2-\beta)/(2\delta_{n-1} + L_n - m)$, which implies $\delta_n = \delta_{n-1}$. □

In the following proposition, we state a result which will be very useful. Although, iPiano does not imply a descent property of the function values, we construct a majorizing function that enjoys a monotonically descent property. This function reveals the connection to the Lyapunov direct method for convergence analysis as used in [ZK93].

Proposition 6.4. *(i) The sequence $(H_{\delta_n}(x^n, x^{n-1}))_{n\in\mathbb{N}}$ is monotonically decreasing and thus converging. In particular, it holds that*

$$H_{\delta_{n+1}}(x^{n+1}, x^n) \leq H_{\delta_n}(x^n, x^{n-1}) - \gamma_n \|x^n - x^{n-1}\|^2 . \tag{6.13}$$

(ii) It holds that $\sum_{n=0}^{\infty} \|x^n - x^{n-1}\|^2 < \infty$ and, thus, $\lim_{n\to\infty} \|x^n - x^{n-1}\| = 0$.

Proof. (i) Observe that by (6.3), we have $G^n(x^{n+1}) \leq G^n(x^n) = g(x^n)$. If g is semi-convex with modulus $m \in \mathbb{R}$, then, using Lemma 6.2 with $0 \in \partial G^n(x^{n+1})$, it holds that $G^n(x^{n+1}) + \frac{1}{2}(m + 1/\alpha)\|x^{n+1} - x^n\|^2 \leq G^n(x^n)$. Summarizing both inequalities using the flag variable σ (defined in the beginning of Section 6.1.4), it holds that

$$G^n(x^{n+1}) + \frac{\sigma}{2}\left(m + \frac{1}{\alpha_n}\right)\|x^{n+1} - x^n\|^2 \leq G^n(x^n) = g(x^n) .$$

This and the quadratic upper bound from the Descent Lemma 2.49 imply

$$\begin{aligned} f(x^{n+1}) + g(x^{n+1}) \leq f(x^n) + \left\langle \nabla f(x^n), x^{n+1} - x^n \right\rangle + \frac{L_n}{2}\|x^{n+1} - x^n\|^2 \\ + g(x^n) - \left\langle \nabla f(x^n) - \frac{\beta_n}{\alpha_n}(x^n - x^{n-1}), x^{n+1} - x^n \right\rangle \\ - \frac{1}{2}\left(\frac{1+\sigma}{\alpha_n} + \sigma m\right)\|x^{n+1} - x^n\|^2 . \end{aligned}$$

Using $2\langle a, b\rangle \leq \|a\|^2 + \|b\|^2$ for vectors $a, b \in \mathbb{R}^N$, we have

$$\left\langle \frac{\beta_n}{\alpha_n}(x^n - x^{n-1}), x^{n+1} - x^n \right\rangle \leq \frac{\beta_n}{2\alpha_n}\left(\|x^{n+1} - x^n\|^2 + \|x^n - x^{n-1}\|^2\right) .$$

Combining the two preceding results yields

$$h(x^{n+1}) \leq h(x^n) - \frac{1}{2}\left(\frac{1+\sigma-\beta_n}{\alpha_n} - (L_n - \sigma m)\right)\|x^{n+1} - x^n\|^2 + \frac{\beta_n}{2\alpha_n}\|x^n - x^{n-1}\|^2$$

and a simple rearrangement shows

$$h(x^{n+1}) + \delta_n\|x^{n+1} - x^n\|^2 \leq h(x^n) + \delta_n\|x^n - x^{n-1}\|^2 - \gamma_n\|x^n - x^{n-1}\|^2 ,$$

which establishes (6.13) as δ_n is monotonically decreasing. Obviously, $(H_{\delta_n}(x^n, x^{n-1}))_{n\in\mathbb{N}}$ is monotonically decreasing if and only if $\gamma_n \geq 0$, which is true by the algorithm's requirements. By assumption, h is bounded from below by some constant $\underline{h} > -\infty$, hence $(H_{\delta_n}(x^n, x^{n-1}))_{n\in\mathbb{N}}$ converges.

(ii) Summing up (6.13) from $k = 0, \dots, n$ yields (note that $H_{\delta_k}(x^0, x^{-1}) = h(x^0)$)

$$\begin{aligned}\sum_{k=0}^{n} \gamma_k \|x^k - x^{k-1}\|^2 &\leq \sum_{k=0}^{n} H_{\delta_k}(x^k, x^{k-1}) - H_{\delta_{k+1}}(x^{k+1}, x^k) \\ &= h(x^0) - H_{\delta_{n+1}}(x^{n+1}, x^n) \leq h(x^0) - \underline{h} < \infty\,.\end{aligned}$$

Letting n tend to ∞ and remembering that $\gamma_n \geq c_2 > 0$ holds implies the statement.

□

Remark 6.3. The function H_δ is a Lyapunov function for the dynamical system described by the Heavy-ball method. It corresponds to a discretized version of the kinetic energy of the Heavy-ball with friction.

In the following theorem, we state our general convergence results of Algorithm 6.

Theorem 6.5. *(i) The sequence $(h(x^n))_{n\in\mathbb{N}}$ converges.*

(ii) There exists a converging subsequence $(x^{n_j})_{j\in\mathbb{N}}$.

(iii) Any limit point $x^* := \lim_{j\to\infty} x^{n_j}$ is a critical point of (6.1) *and $h(x^{n_j}) \to h(x^*)$ as $j \to \infty$.*

Proof. (i) This follows from the Squeeze theorem as for all $n \geq 0$ it holds that

$$H_{-\delta_n}(x^n, x^{n-1}) \leq h(x^n) \leq H_{\delta_n}(x^n, x^{n-1})$$

and thanks to Proposition 6.4(i) and (ii) it holds that

$$\lim_{n\to\infty} H_{-\delta_n}(x^n, x^{n-1}) = \lim_{n\to\infty} H_{\delta_n}(x^n, x^{n-1}) - 2\delta_n\|x^n - x^{n-1}\|^2 = \lim_{n\to\infty} H_{\delta_n}(x^n, x^{n-1})\,.$$

(ii) By Proposition 6.4(i) and $H_{\delta_0}(x^0, x^{-1}) = h(x^0)$ it is clear that the whole sequence $(x^n)_{n\in\mathbb{N}}$ is contained in the level set $\{x \in \mathbb{R}^N | \, \underline{h} \leq h(x) \leq h(x^0)\}$, which is bounded thanks to the coercivity of h and $\underline{h} = \inf_{x\in\mathbb{R}^N} h(x) > -\infty$. Using the Bolzano-Weierstrass theorem, we deduce the existence of a converging subsequence $(x^{n_j})_{j\in\mathbb{N}}$.

(iii) To show that each limit point $x^* := \lim_{j\to\infty} x^{n_j}$ is a critical point of (6.1) recall that the (limiting) subgradient is closed (Corollary 4.15). For notational elegance, we abbreviate $n'_j := n_j - 1$ and $n' := n - 1$. Define

$$v^j := \frac{x^{n'_j} - x^{n'_j+1}}{\alpha_{n'_j}} - \nabla f(x^{n'_j}) + \frac{\beta_{n'_j}}{\alpha_{n'_j}}(x^{n'_j} - x^{n'_j-1}) + \nabla f(x^{n'_j+1}).$$

As $0 \in \partial G^{n'_j}(x^{n'_j+1})$ yields an element in $\partial g(x^{n'_j+1})$, which, added to $\nabla f(x^{n'_j+1})$, equals v^j, the sequence $(x^{n_j}, v^j) \in \mathrm{Graph}(\partial h) := \{(x, v) \in \mathbb{R}^N \times \mathbb{R}^N | \, v \in \partial h(x)\}$, Furthermore, it holds that $x^* = \lim_{j\to\infty} x^{n_j}$ and due to Proposition 6.4(ii), the Lipschitz continuity of ∇f, and

$$\|v^j - 0\| \leq \frac{1}{\alpha_{n'_j}}\|x^{n'_j+1} - x^{n'_j}\| + \frac{\beta_{n'_j}}{\alpha_{n'_j}}\|x^{n'_j} - x^{n'_j-1}\| + \|\nabla f(x^{n'_j+1}) - \nabla f(x^{n'_j})\|$$

it holds $\lim_{j\to\infty} v^j = 0$. It remains to show that $\lim_{j\to\infty} h(x^{n_j}) = h(x^*)$. By the closure property of the subgradient ∂h it is $(x^*, 0) \in \mathrm{Graph}(\partial h)$, which means that x^* is a critical point of h.

The continuity statement follows from (6.3),

$$g(x^{n'+1}) + \left\langle \nabla f(x^{n'}) - \frac{\beta_{n'}}{\alpha_{n'}}(x^{n'} - x^{n'-1}), x^{n'+1} - x^{n'} \right\rangle + \frac{1}{2\alpha_{n'}}\|x^{n'+1} - x^{n'}\|^2 \\ \leq g(x) + \left\langle \nabla f(x^{n'}) - \frac{\beta_{n'}}{\alpha_{n'}}(x^{n'} - x^{n'-1}), x - x^{n'} \right\rangle + \frac{1}{2\alpha_{n'}}\|x - x^{n'}\|^2 ,$$

which implies

$$g(x^{n'_j+1}) + \left\langle \nabla f(x^{n'_j}) - \frac{\beta_{n'_j}}{\alpha_{n'_j}}(x^{n'_j} - x^{n'_j-1}), x^{n'_j+1} - x \right\rangle \\ + \frac{1}{2\alpha_{n'_j}} \left(\|x^{n'_j+1} - x^{n'_j}\|^2 - \|x - x^{n'_j}\|^2 \right) \leq g(x) .$$

Proposition 6.4(ii) and boundedness of $\nabla f(x^{n'}) - \frac{\beta_{n'}}{\alpha_{n'}}(x^{n'} - x^{n'-1})$ yield $\limsup_{j\to\infty} g(x^{n_j}) \leq g(x)$. Invoking the lower semi-continuity of g yields $\lim_{j\to\infty} g(x^{n_j}) = g(x^*)$. Moreover, as f is differentiable (thus continuous), we conclude $\lim_{j\to\infty} h(x^{n_j}) = h(x^*)$.

□

Remark 6.4. The convergence properties shown in Theorem 6.5 should be the basic requirements of any algorithm. Very loosely speaking, the theorem states that the algorithm ends up in a meaningful solution. It allows us to formulate stopping conditions, e.g., the residual between successive function values.

Now, using Theorem 5.4, we can verify the convergence of the sequence $(x^n)_{n\in\mathbb{N}}$ generated by Algorithm 6. We assume that after a finite number of steps the sequence $(\delta_n)_{n\in\mathbb{N}}$ is constant and consider the sequence $(x^n)_{n\in\mathbb{N}}$ starting from this iteration (again denoted by $(x^n)_{n\in\mathbb{N}}$). For example, if δ_n is determined relative to the Lipschitz constant, then as the Lipschitz constant can be assumed constant after a finite number of iterations, δ_n is also constant starting from this iteration.

Theorem 6.6 (Convergence of iPiano to a critical point)**.** *Let $(x^n)_{n\in\mathbb{N}}$ be generated by Algorithm 6, and let $\delta_n = \delta$ and $\sigma m \leq L_n$ for all $n \in \mathbb{N}$. Then, the sequence $(x^{n+1}, x^n)_{n\in\mathbb{N}}$ satisfies (H1), (H2), and (H3) from Section 5.1 for the function $H_\delta \colon \mathbb{R}^{2N} \to \mathbb{R} \cup \{\infty\}$, $(x, y) \mapsto h(x) + \delta\|x - y\|^2$.*

Moreover, if $H_\delta(x, y)$ has the Kurdyka–Łojasiewicz property at a cluster point (x^*, x^*), then the sequence $(x^n)_{n\in\mathbb{N}}$ has finite length, $x^n \to x^*$ as $n \to \infty$, and (x^*, x^*) is a critical point of H_δ, hence x^* is a critical point of h.

Proof. First, we verify that assumptions (H1), (H2), and (H3) are satisfied. We consider the sequence $z^n = (x^n, x^{n-1})$ for all $n \in \mathbb{N}$ and the proper lower semi-continuous function $F = H_\delta$.

- Condition (H1) is proved in Proposition 6.4(a) with $a = c_2 \leq \gamma_n$.
- To prove Condition (H2), consider $w^{n+1} := (w_x^{n+1}, w_y^{n+1})^\top \in \partial H_\delta(x^{n+1}, x^n)$ with $w_x^{n+1} \in \partial g(x^{n+1}) + \nabla f(x^{n+1}) + 2\delta(x^{n+1} - x^n)$ and $w_y^{n+1} = -2\delta(x^{n+1} - x^n)$. The Lipschitz continuity of ∇f and using $0 \in G^n(x^{n+1})$ (see (6.3)) to specify an element from $\partial g(x^{n+1})$ imply

$$\begin{aligned} \|w^{n+1}\| &\leq \|w_x^{n+1}\| + \|w_y^{n+1}\| \\ &\leq \|\nabla f(x^{n+1}) - \nabla f(x^n)\| + (\tfrac{1}{\alpha_n} + 4\delta)\|x^{n+1} - x^n\| \\ &\quad + \tfrac{\beta_n}{\alpha_n}\|x^n - x^{n-1}\| \\ &\leq \tfrac{1}{\alpha_n}(\alpha_n L_n + 1 + 4\alpha_n\delta)\|x^{n+1} - x^n\| + \tfrac{1}{\alpha_n}\beta_n\|x^n - x^{n-1}\| . \end{aligned}$$

As $\alpha_n L_n \leq 1 + \sigma - 2\beta_n \leq 2$ and $\delta\alpha_n = \frac{1+\sigma}{2} - \frac{1}{2}\alpha_n(L_n - \sigma m) - \frac{1}{2}\beta_n \leq 1$, setting $b = \frac{7}{c_1}$ verifies condition (H2), i.e., $\|w^{n+1}\| \leq b(\|x^n - x^{n-1}\| + \|x^{n+1} - x^n\|)$.

- In Theorem 6.5(iii) it is proved that there exists a subsequence $(x^{n_j+1})_{j\in\mathbb{N}}$ of $(x^n)_{n\in\mathbb{N}}$ such that $\lim_{j\to\infty} h(x^{n_j+1}) = h(x^*)$. Proposition 6.4(ii) shows that $\|x^{n+1} - x^n\| \to 0$ as $n \to \infty$, hence $\lim_{j\to\infty} x^{n_j} = x^*$. As the term $\delta\|x - y\|^2$ is continuous in x and y, we deduce

$$\lim_{j\to\infty} H(x^{n_j+1}, x^{n_j}) = \lim_{j\to\infty} h(x^{n_j+1}) + \delta\|x^{n_j+1} - x^{n_j}\| = H(x^*, x^*) = h(x^*)\,.$$

Now, the abstract convergence Theorem 5.4 concludes the proof. □

The next corollary makes use of the fact that functions definable in an o-minimal structure (see Section 4.5) have the Kurdyka–Łojasiewicz property Theorem 4.35. This holds for example for semi-algebraic functions introduced in Section 4.5.1. The reader who is not familiar with the concept of o-minimal structures can simply replace "definable" with "semi-algebraic" in the following corollary.

Corollary 6.7 (Convergence of iPiano for definable functions)**.** *Let h be a definable in an o-minimal structure. Then, $H_\delta(x, y)$ is also definable. Furthermore, let $(x^n)_{n\in\mathbb{N}}$, $(\delta_n)_{n\in\mathbb{N}}$, $(x^{n+1}, x^n)_{n\in\mathbb{N}}$ be as in Theorem 6.6; and $\sigma m \leq L_n$ for all $n \in \mathbb{N}$. Then the sequence $(x^n)_{n\in\mathbb{N}}$ has finite length, $x^n \to x^*$ as $n \to \infty$, and x^* is a critical point of h.*

Proof. As h and $\delta\|x - y\|^2$ are definable, $H_\delta(x, y)$ is definable and has the KL property (Theorem 4.35). Then, Theorem 6.6 concludes the proof. □

6.1.5 Convergence rate

We prove a global $\mathcal{O}(1/\sqrt{n})$ convergence rate for $\|x^{n+1} - x^n\|$. We first define the error μ_n to be the smallest squared ℓ_2 norm of successive iterates

$$\mu_n := \min_{0\leq k\leq n} \|x^k - x^{k-1}\|^2\,.$$

Theorem 6.8. *Algorithm 6 guarantees that for all $n \geq 0$*

$$\mu_n \leq c_2^{-1}\frac{h(x^0) - \underline{h}}{n+1}\,.$$

Proof. In view of Proposition 6.4(i), and the definition of γ_n in (6.9), summing up both sides of (6.13) for $k = 0, \ldots, n$ and using that $\delta_n > 0$ from (6.9), we obtain

$$\underline{h} \leq h(x^0) - \sum_{k=0}^{n} \gamma_k\|x^k - x^{k-1}\|^2 \leq h(x^0) - (n+1)\min_{0\leq k\leq n} \gamma_k\mu_n\,.$$

As it is $\gamma_k > c_2$, a simple rearrangement concludes the proof. □

Remark 6.5. A similar result can be found in [Nes13] for the case $\beta = 0$.

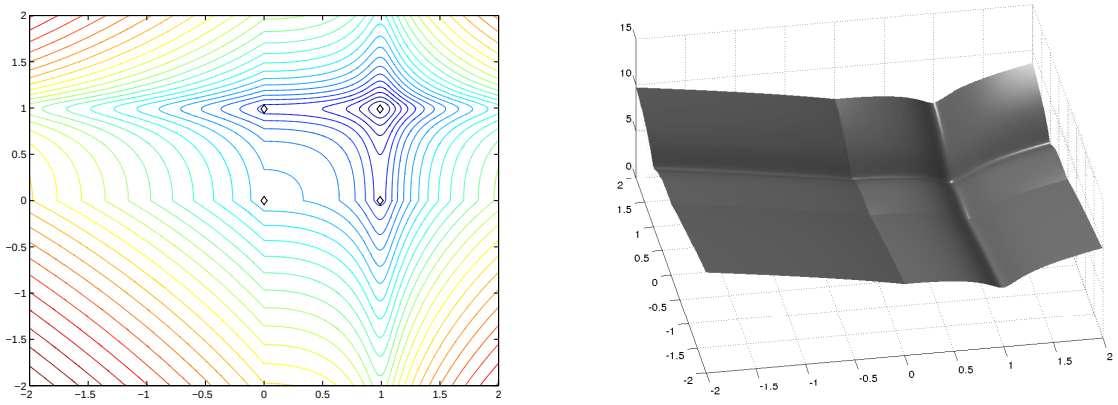


Figure 6.1: LEFT: Contour plot of $h(x)$ in (6.14). RIGHT: Energy landscape of the nonconvex function $h(x)$. The four diamonds mark stationary points of the function h.

6.2 Numerical experiments

In all of the following experiments, let $u, u^0 \in \mathbb{R}^N$ be vectors of dimension $N \in \mathbb{N}$, where N depends on the respective problem. In the case of an image N is the number of pixels. The remaining part of this chapter is published in [OCBP14].

6.2.1 Ability to overcome spurious stationary points

Let us present some of the qualitative properties of the proposed algorithm. For this, we consider minimizing the following simple problem

$$\min_{x\in\mathbb{R}^N} h(x) := f(x) + g(x)\,, \quad f(x) = \frac{1}{2}\sum_{i=1}^{N} \log(1+\mu(x_i - u_i^0)^2)\,, \quad g(x) = \lambda\|x\|_1\,, \tag{6.14}$$

where x is the unknown vector, u^0 is some given vector, and $\lambda, \mu > 0$ are some free parameters. A contour plot and the energy landscape of h in the case of $N = 2$, $\lambda = 1$, $\mu = 100$, and $u^0 = (1,1)^\top$ is depicted in Figure 6.1. It turns out that the function h has four stationary points, i.e. points $\bar{x}$, such that $0 \in \nabla f(\bar{x}) + \partial g(\bar{x})$. These points are marked by small black diamonds. Clearly the function f is nonconvex but has a Lipschitz continuous gradient with components

$$\nabla f(x)_i = \mu \frac{x_i - u_i^0}{1+\mu(x_i - u_i^0)^2}\,.$$

The Lipschitz constant of ∇f is easily computed as $L = \mu$. The function g is nonsmooth but convex and the proximal operator with respect to g is given by the well-known shrinkage operator, which we computed in Example 2.16(iv). Let us test the performance of the proposed algorithm on the example shown in Figure 6.1. We set $\alpha = 2(1-\beta)/L$. Figure 6.2 shows the results of using the iPiano algorithm for different settings of the extrapolation factor β. We observe that iPiano with $\beta = 0$ is strongly attracted by the closest stationary points while switching on the inertial term can help to overcome the spurious stationary points. The reason for this desired property is that while the gradient might vanish at some points, the inertial term $\beta(x^n - x^{n-1})$ is still strong enough to drive the sequence out of the stationary region. Clearly, there is no guarantee that iPiano always avoids spurious stationary points. iPiano is not

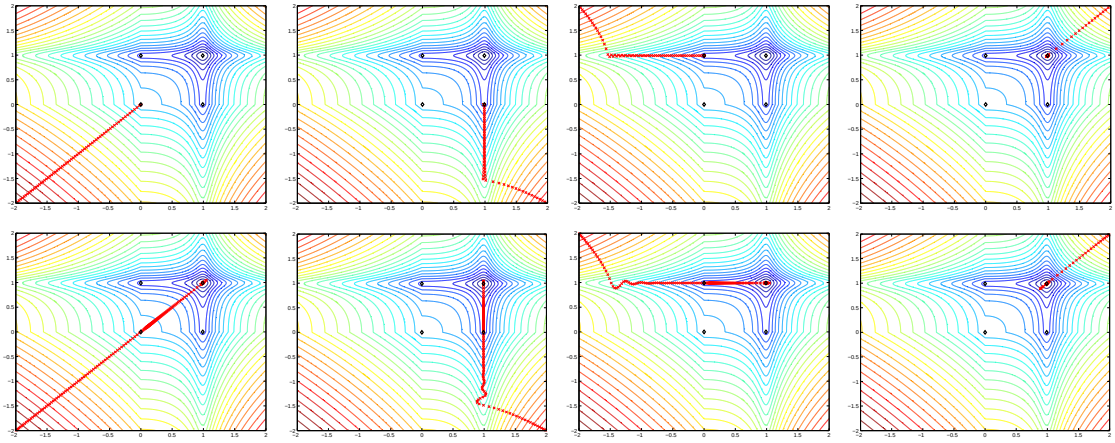


Figure 6.2: FIRST ROW, FROM LEFT TO RIGHT: Results of using iPiano for four different starting points with $\beta = 0$. SECOND ROW: Results for the same starting position as in the first row using iPiano with $\beta = 0.75$. While the algorithm without an inertial term gets stuck in unwanted local stationary points in three of four cases, the algorithm with an inertial term always succeeds in converging to the global optimum.

designed to find the global optimum. However, our numerical experiments suggest that in many cases, iPiano finds lower energies than the respective algorithm without inertial term. A similar observation about the Heavy-ball method is described in [Ber99].

6.2.2 Image processing applications

In this section, we demonstrate the applicability of the proposed algorithm to solving a class of nonconvex regularized variational models. We present examples for natural image denoising and linear diffusion based image compression. We show that iPiano can be easily adapted to all of these problems and yields state-of-the-art results.

6.2.2.1 Student-t regularized image denoising

We investigate the task of natural image denoising. For this we exploit an optimized Markov random field (MRF) model (see [CPRB13]) and make use of the iPiano algorithm to solve it. In order to evaluate the performance of iPiano, we compare it to the well-known bound constrained limited memory quasi Newton method (L-BFGS89) [LN89] [3]. As an error measure, we use the energy difference

$$\mathcal{E}^n = h^n - h^* \,, \tag{6.15}$$

where h^n is the energy of the current iteration n and h^* is the energy of the true solution. Clearly, this error measure makes sense only when different algorithms can achieve the same true energy h^* which is in general wrong for nonconvex problems. In our image denoising experiments, however, we find that all tested algorithms find the same solution, independent of the initialization. This can be explained by the fact that the learning procedure [CPRB13] also delivers models that are relatively easy to optimize, since otherwise they would have resulted in a bad training error. In order to compute a true energy h^*, we run the iPiano algorithm with $\beta = 0.8$ for enough iterations ($\sim$1000 iterations). We run all the experiments in MATLAB on a 64-bit Linux server with 2.53GHz CPUs.

[3] We make use of the implementation distributed at `http://www.cs.toronto.edu/~liam/software.shtml`.

The MRF image denoising model based on learned filters is formulated as

$$\min_{u\in\mathbb{R}^N} \sum_{i=1}^{N_f} \vartheta_i \Phi(K_i u) + g_{1,2}(u, u^0)\,, \tag{6.16}$$

where u and $u^0 \in \mathbb{R}^N$ denote the sought solution and the noisy input image respectively, Φ is the nonconvex penalty function, $\Phi(K_i u) = \sum_p \varphi((K_i u)_p)$, K_i are learned, linear operators with the corresponding weights ϑ_i, and N_f is the number of the filters. The linear operators K_i are implemented as two-dimensional convolutions of the image u with small (e.g. 7×7) filter kernels k_i, i.e. $K_i u = k_i * u$. The function $g_{1,2}$ is the data term, which depends on the respective problem. In the case of Gaussian noise, $g_{1,2}$ is given as

$$g_2(u, u^0) = \frac{\lambda}{2}\|u - u^0\|_2^2\,,$$

and for the impulse noise (e.g., salt and pepper noise), $g_{1,2}$ is given as

$$g_1(u, u^0) = \lambda\|u - u^0\|_1\,.$$

The parameter $\lambda > 0$ is used to define the tradeoff between regularization and data fitting.

We consider the following nonconvex penalty function, which is derived from the student-t distribution:

$$\varphi(t) = \log(1 + t^2)\,. \tag{6.17}$$

Concerning the filters k_i, for the ℓ_2 model (MRF-ℓ_2), we make use of the filters learned in [CPRB13] by using a bilevel learning approach. The filters are shown in Figure 6.3(a) together with the corresponding weights ϑ_i. For the MRF-ℓ_1 denoising model, we employ the same bilevel learning algorithm to train a set of optimal filters specialized for the ℓ_1 data term and input images degraded by salt and pepper noise. Since the bilevel learning algorithm requires a twice continuously differentiable model we replace the ℓ_1 norm by a smooth approximation during training. The learned filters for the MRF-ℓ_1 model together with the corresponding weights ϑ_i are shown in Figure 6.3(b).

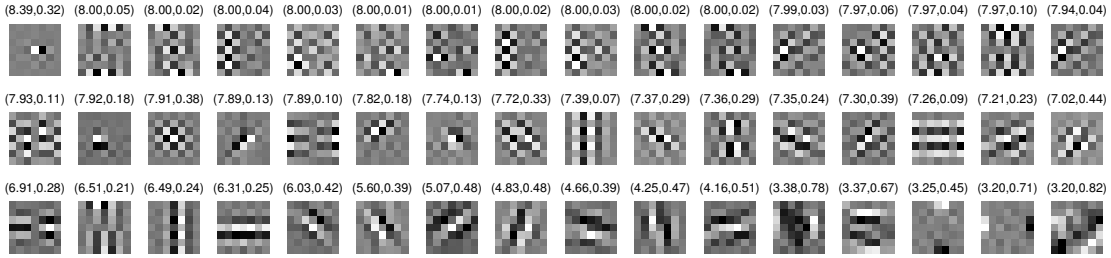
Let us now explain how to solve (6.16) using the iPiano algorithm. Casting (6.16) in the form of (6.1), we see that $f(u) = \sum_{i=1}^{N_f} \vartheta_i \Phi(K_i u)$ and $g(u) = g_{1,2}(u, u^0)$. Thus, we have

$$\nabla f(u) = \sum_{i=1}^{N_f} \vartheta_i K_i^\top \Phi'(K_i u)\,,$$

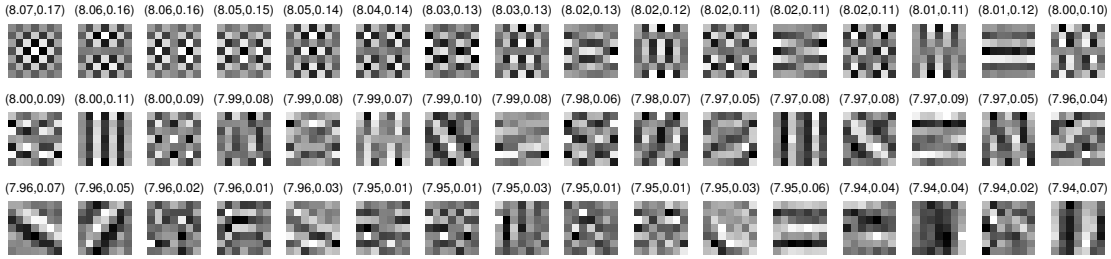
where $\Phi'(K_i u) = [\varphi'((K_i u)_1)\,, \varphi'((K_i u)_2), \ldots, \varphi'((K_i u)_p)]^\top$ and $\varphi'(t) = 2t/(1 + t^2)$. The proximal map with respect to g simply poses pointwise operations. They are easily obtained combining Example 2.16(i) and (iv) with Lemma 2.41. For the case of g_2, it is given by $u_p = (\hat{u}_p + \alpha\lambda u_p^0)/(1 + \alpha\lambda)$, and for g_1 by $u_p = \max(0, |\hat{u}_p - u_p^0| - \alpha\lambda) \cdot \operatorname{sign}(\hat{u}_p - u_p^0) + u_p^0$, for all $p = 1, \ldots, N$. Now, we can make use of our proposed algorithm to solve the nonconvex optimization problems. In order to evaluate the performance of iPiano, we compare it to L-BFGS89. To use L-BFGS89, we merely need the gradient of the objective function with respect to u. For the MRF-ℓ_2 model, calculating the gradients is straightforward. However, in the case of the MRF-ℓ_1 model, due to the nonsmooth function g, we cannot directly use L-BFGS89. Since L-BFGS89 can easily handle box constraints, we can get rid of the nonsmooth function ℓ_1 norm by introducing two box constraints.

Lemma 6.9. *The MRF-ℓ_1 model can be equivalently written as the bound-constraint problem*

$$\min_{w,v} \sum_{i=1}^{N_f} \vartheta_i \Phi(K_i(w + v)) + \lambda\, 1^\top (v - w) \quad \text{s.t.} \quad w \le u^0/2,\ v \ge u^0/2\,. \tag{6.18}$$



(a) Learned filters for the MRF-ℓ_2 model



(b) Learned filters for the MRF-ℓ_1 model

Figure 6.3: 48 learned filters of size 7×7 for two different MRF denoising models. The first number in the bracket is the weights ϑ_i, and the second one is the norm $\|k_i\|_2$ of the filters.

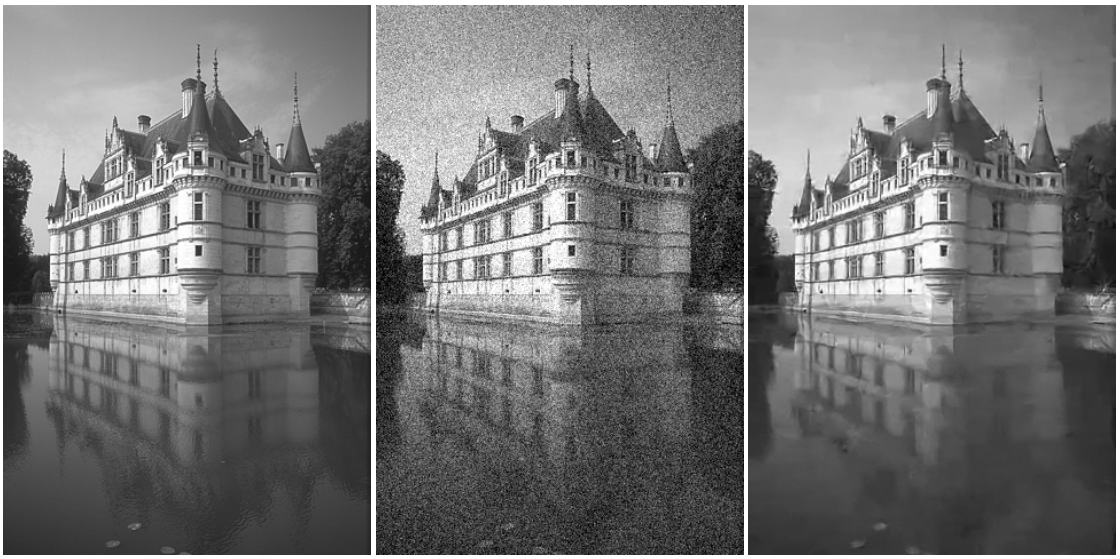


Figure 6.4: Natural image denoising by using the student-t regularized MRF model (MRF-ℓ_2). LEFT: Clean image. MIDDLE: Noisy image corrupted by additive zero mean Gaussian noise with $\sigma = 25$. RIGHT: Denoised image.

Proof. It is well-know that the ℓ_1 norm $\|u - u^0\|_1$ can be equivalently expressed as

$$\|u - u^0\|_1 = \min_t 1^\top t\,, \quad \text{s.t.} \quad t \geq u - u^0\,, \quad t \geq -u + u^0\,,$$

where $t \in \mathbb{R}^N$ and the inequalities are understood pointwise. Letting $w = (u - t)/2 \in \mathbb{R}^N$ and

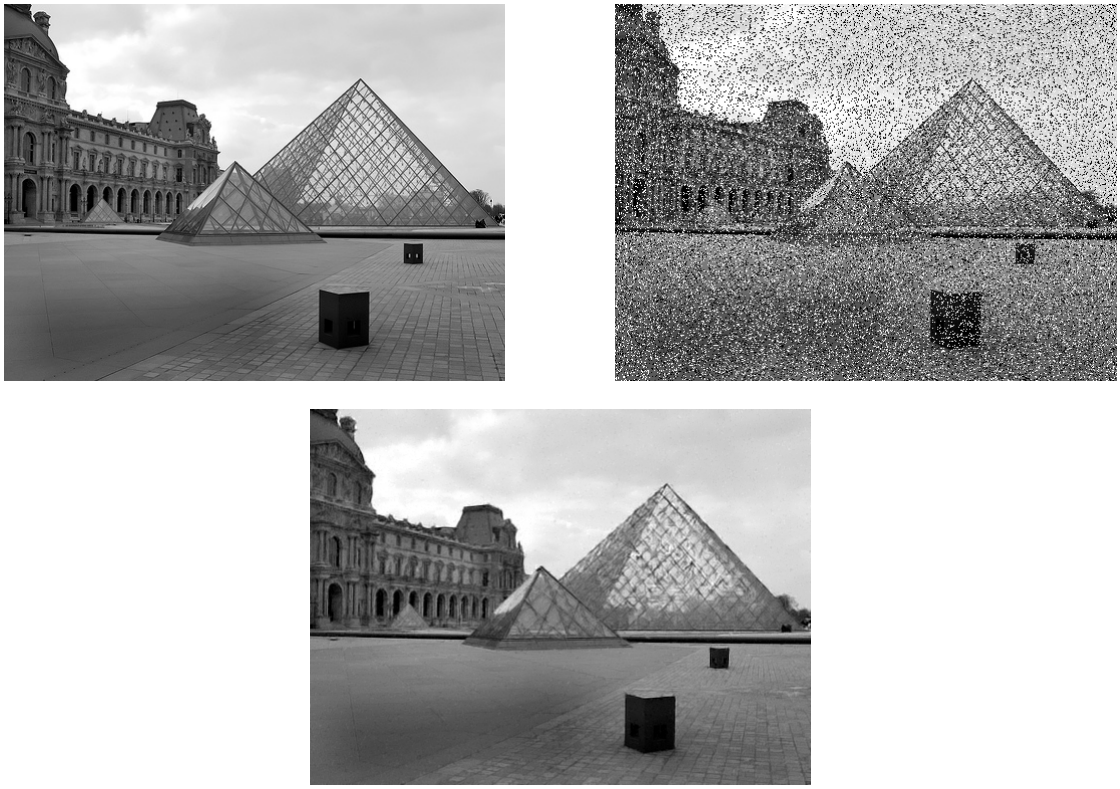


Figure 6.5: Natural image denoising in the case of impulse noise by using the MRF-ℓ_1 model. TOP LEFT: Clean image. TOP RIGHT: Noisy image corrupted by 25% salt & pepper noise. BOTTOM: Denoised image.

$v = (u + t)/2 \in \mathbb{R}^N$, we find $u = w + v$ and $t = v - w$. Substituting u and t back into (6.16) while using the above formulation of the ℓ_1 norm yields the desired transformation. □

Figures 6.4 and 6.5 respectively show a denoising example using the MRF-ℓ_2 model, and the MRF-ℓ_1 model. In both experiments, we use the iPiano version with backtracking (Algorithm 5) with the following parameter settings:

$$L_{-1} = 1,\ \eta = 1.2,\ \alpha_n = 1.99(1 - \beta)/L_n\,,$$

where β is a free parameter to be evaluated in the experiment. In order to make use of possible larger step sizes in practice, we use the following trick: when the inequality (6.5) is fulfilled, we decrease the evaluated Lipschitz constant L_n slightly by setting $L_n = L_n/1.05$.

For the MRF-ℓ_2 denoising experiments, we initialized u using the noisy image itself; however, for the MRF-ℓ_1 denoising model, we initialized u using a zero image. We found that this initialization strategy usually gives good convergence behavior for both algorithms. For both denoising examples, we run the algorithms until the error $\mathcal{E}^n$ decreases to a certain predefined threshold tol. We then record the required number of iterations and the run time. We summarize the results of the iPiano algorithm with different settings and L-BFGS89 in Tables 6.1 and 6.2. From these two tables, one can draw the common conclusion that iPiano with a proper inertial term takes significantly fewer iterations compared to the case without inertial term, and in practice $\beta \approx 0.8$ is generally a good choice.

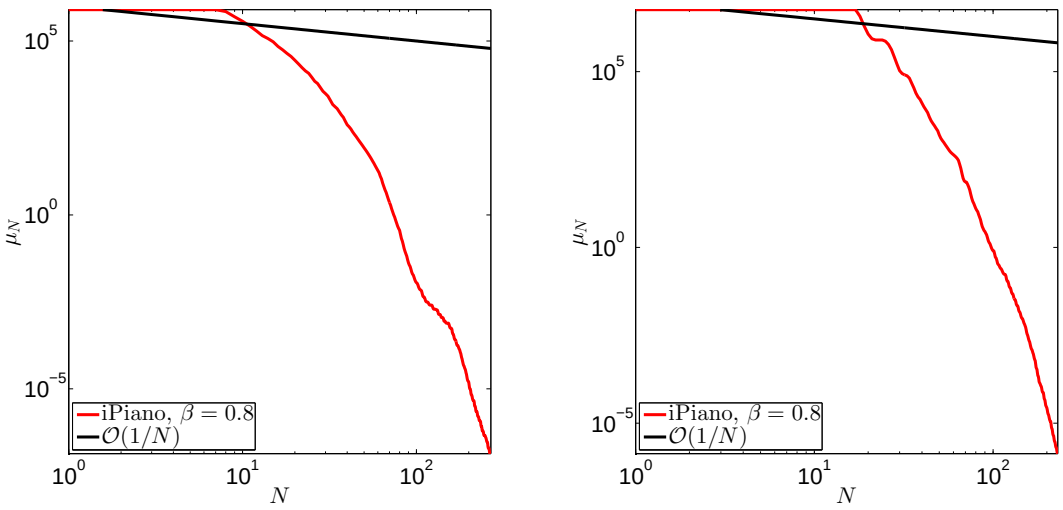


Figure 6.6: Convergence rates for the MRF-ℓ_2 and -ℓ_1 models. The figures plot the minimal residual norm μ_n. LEFT: MRF-ℓ_2 model. RIGHT: MRF-ℓ_1 model. Note that the empirical convergence rate is much faster compared to the worst case rate (see Theorem 6.8).

In Table 6.1, one can see that the iPiano algorithm with $\beta = 0.8$ takes slightly more iterations and a longer run time to reach a solution of moderate accuracy (e.g., tol $= 10^{-3}$) compared with L-BFGS89. However, for highly accurate solutions (e.g., tol $= 10^{-5}$), this gap increases. For the case of the nonsmooth MRF-ℓ_1 model, the result is just the reverse. It is shown in Table 6.2, that for reaching a moderately accurate solution, iPiano with $\beta = 0.8$ consumes significantly fewer iterations a shorter run time, and for the solution of high accuracy, it still can save much computation.

Figure 6.6 plots the error μ_n over the number of required iterations n for both the MRF-ℓ_2 and MRF-ℓ_1 models using $\beta = 0.8$. From the plots it becomes obvious that the empirical performance of the iPiano algorithm is much better compared to the worst-case convergence rate of $\mathcal{O}(1/\sqrt{n})$ as provided in Theorem 6.8.

The iPiano algorithm has an additional advantage of simplicity. The iPiano version without backtracking basically relies on matrix vector products (filter operations in the denoising examples) and simple pointwise operations. Therefore, the iPiano algorithm is well suited for a parallel implementation on GPUs which can lead to speedup factors of 20-30.

6.2.2.2 Linear diffusion based image compression

In this example we apply the iPiano algorithm to linear diffusion based image compression. Recent works [GWW$^+$08, SWB09] have shown that image compression based on linear and nonlinear diffusion can outperform the JPEG standard and even the more advanced JPEG 2000 standard, when the interpolation points are carefully chosen. Therefore, finding optimal data for interpolation is a key problem in the context of PDE-based image compression. There exist only a few prior works on this topic (see, e.g. [MHW$^+$11, HSW13]), and the very recent approach presented in [HSW13] defines the state-of-the-art.

The problem of finding optimal data for homogeneous diffusion based interpolation is formulated as

	iPiano with different β							L-BFGS89	
tol	0.00	0.20	0.40	0.60	0.80	0.95	T_1(s)	iter.	T_2(s)
10^3	260	182	116	66	56	214	34.073	43	18.465
10^2	372	256	164	94	67	257	40.199	55	22.803
10^1	505	344	222	129	79	299	47.177	66	27.054
10^0	664	451	290	168	98	342	59.133	79	32.143
10^{-1}	857	579	371	216	143	384	85.784	93	36.926
10^{-2}	1086	730	468	271	173	427	103.436	107	41.939
10^{-3}	1347	904	577	338	199	473	119.149	124	48.272
10^{-4}	1639	1097	697	415	232	524	138.416	139	53.290
10^{-5}	1949	1300	827	494	270	569	161.084	154	58.511

Table 6.1: The number of iterations and the run time necessary for reaching the corresponding error for iPiano and L-BFGS89 to solve the MRF-ℓ_2 model. T_1 is the run time of iPiano with $\beta = 0.8$ and T_2 shows the run time of L-BFGS89.

	iPiano with different β							L-BFGS89	
tol	0.00	0.20	0.40	0.60	0.80	0.95	T_1(s)	iter.	T_2(s)
10^3	390	272	174	96	64	215	43.709	223	102.383
10^2	621	403	256	145	77	260	53.143	246	112.408
10^1	847	538	341	195	96	304	65.679	265	121.303
10^0	1077	682	433	247	120	349	81.761	285	130.846
10^{-1}	1311	835	530	303	143	395	97.060	298	136.326
10^{-2}	1559	997	631	362	164	440	111.579	311	141.876
10^{-3}	1818	1169	741	424	185	485	126.272	327	148.945
10^{-4}	2086	1346	853	489	208	529	142.083	347	157.956
10^{-5}	2364	1530	968	557	233	575	159.493	372	169.674

Table 6.2: The number of iterations and the run time necessary for reaching the corresponding error for iPiano and L-BFGS89 to solve the MRF-ℓ_1 model. T_1 is the run time of iPiano with $\beta = 0.8$ and T_2 shows the run time of L-BFGS89.

the following constrained minimization problem:

$$\begin{aligned} &\min_{u,c} \frac{1}{2}\|u - u^0\|_2^2 + \lambda\|c\|_1 \\ &\text{s.t. } C(u - u^0) - (I - C)Lu = 0\,, \end{aligned} \tag{6.19}$$

where $u^0 \in \mathbb{R}^N$ denotes the ground truth image, $u \in \mathbb{R}^N$ denotes the reconstructed image, and $c \in \mathbb{R}^N$ denotes the inpainting mask, i.e. the characteristic function of the set of points that are chosen for compressing the image. Furthermore, we denote by $C = \operatorname{diag}(c) \in \mathbb{R}^{N\times N}$ the diagonal matrix with the vector c on its main diagonal, by I the identity matrix and by $L \in \mathbb{R}^{N\times N}$ the Laplacian operator. Compared to the original formulation [HSW13], we omit a very small quadratic term $\frac{\varepsilon}{2}\|c\|_2^2$, because we find it unnecessary in experiments.

Observe that if $c \in [0, 1)^N$, we can multiply the constraint equation in (6.19) from the left by $(I - C)^{-1}$ such that it becomes

$$E(c)(u - u^0) - Lu = 0\,,$$

where $E(c) = \operatorname{diag}(c_1/(1 - c_1), \ldots, c_N/(1 - c_N))$. This shows that problem (6.19) is in fact a reduced

formulation of the bilevel optimization problem

$$\begin{aligned} &\min_c \frac{1}{2}\|u(c) - u^0\|_2^2 + \lambda\|c\|_1 \\ &\text{s.t.} \quad u(c) = \arg\min_u \|Du\|_2^2 + \|E(c)^{\frac{1}{2}}(u - u^0)\|_2^2 \,, \end{aligned} \tag{6.20}$$

where D is the nabla operator and hence $-L = D^\top D$.

Problem (6.19) is nonconvex due to the nonconvexity of the equality constraint. In [HSW13], the above problem is solved by a successive primal–dual (SPD) algorithm, which successively linearizes the nonconvex constraint and solves the resulting convex problem with the first-order primal–dual algorithm [CP11]. The main drawback of SPD is, that it requires tens of thousands of inner iterations and thousands of outer iterations to reach a reasonable solution. However, as we now demonstrate, iPiano can solve this problem with higher accuracy in 1000 iterations.

Observe that we can rewrite problem (6.19) by solving u from the constraints equation, which gives $u = A^{-1}Cu^0$, where $A = C + (C - I)L$. In [MBWF11], it is shown that the A is invertible as long as at least one element of c is nonzero, which is the case for nondegenerate problems. Substituting back the above equation back into (6.19), we arrive at the following optimization problem, which now depends only on the inpainting mask c:

$$\min_c \frac{1}{2}\|A^{-1}Cu^0 - u^0\|_2^2 + \lambda\|c\|_1 \,. \tag{6.21}$$

Casting (6.21) in the form of (6.1), we have $f(c) = \frac{1}{2}\|A^{-1}Cu^0 - u^0\|_2^2$, and $g(c) = \lambda\|c\|_1$. In order to minimize the above problem using iPiano, we need to calculate the gradient of f with respect to c, which is

$$\nabla f(c) = \operatorname{diag}(-(I + L)u + u^0)(A^\top)^{-1}(u - u^0) \,. \tag{6.22}$$

Finally, we need to compute the proximal map with respect to $g(c)$ which is again given by a pointwise application of the shrinkage operator, see Example 2.16(iv).

Now, we can make use of the iPiano algorithm to solve problem (6.21). We set $\beta = 0.8$, which generally performs very well in practice. We additionally accelerate the SPD algorithm used in the previous work [HSW13] by applying the diagonal preconditioning technique [PC11], which significantly reduces the required iterations for the primal–dual algorithm in the inner loop.

Figure 6.7 shows examples of finding optimal interpolation data for the three test images. Table 6.3 summarizes the results of two different algorithms. Regarding the reconstruction quality, we make use of the mean squared error (MSE) as an error measurement to be consistent with previous work; the MSE is computed by

$$MSE(u, u^0) = \frac{1}{N}\sum_{i=1}^{N}(u_i - u_i^0)^2 \,.$$

From Table 6.3, one can see that the Successive PD algorithm requires 200×4000 iterations to converge. iPiano needs only 1000 iterations to reach a lower energy. Note that in each iteration of the iPiano algorithm, two linear systems have to be solved. In our implementation we use the MATLAB "backslash" operator, which effectively exploits the strong sparseness of the systems. A lower energy basically implies that iPiano can solve the minimization problem (6.19) better. Regarding the final compression result, usually the result of iPiano has slightly less density but slightly worse MSE. Following the work [MHW$^+$11], we also consider the so-called gray value optimization (GVO) as a post-processing step to further improve the MSE of the reconstructed images.

(a) Test image (256×256) (b) Optimized mask (c) Reconstruction

Figure 6.7: Examples of finding optimal inpainting mask for Laplace interpolation based image compression by using iPiano. FIRST ROW: Test image *trui* of size 256×256. Parameter $\lambda = 0.0036$, the optimized mask has a density of 4.98% and the MSE of the reconstructed image is 16.89. SECOND ROW: Test image *peppers* of size 256×256. Parameter $\lambda = 0.0034$, the optimized mask has a density of 4.84% and the MSE of the reconstructed image is 18.99. THIRD ROW: Test image *walter* of size 256×256. Parameter $\lambda = 0.0018$, the optimized mask has a density of 4.82% and the MSE of the reconstructed image is 8.03.

Test image	Algorithm	Iterations	Energy	Density	MSE	with GVO
trui	iPiano	1000	21.574011	4.98%	17.31	16.89
	SPD	200/4000	21.630280	5.08%	17.06	16.54
peppers	iPiano	1000	20.631985	4.84%	19.50	18.99
	SPD	200/4000	20.758777	4.93%	19.48	18.71
walter	iPiano	1000	10.246041	4.82%	8.29	8.03
	SPD	200/4000	10.278874	4.93%	8.01	7.72

Table 6.3: Summary of two algorithms for three test images.

Chapter 7

On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision

Like in Chapter 6, we consider a structured nonsmooth and nonconvex optimization problem. Here, it is of the form:

$$\min_{x \in X} F_1(x) + F_2(G(x)) \,, \tag{7.1}$$

where F_1 is a convex function, G is a coordinate-wise convex function and F_2 is nonconvex. Compared to the iPiano algorithm with convex function g, the method that is presented in this chapter is more general. Unlike the function f in the iPiano algorithm, the function F_2 is not required to have a Lipschitz continuous gradient. The additional function G that is explicitly modeled in (7.1) is suitable for computer vision problems. It is not a restriction, as the identity map is feasible.

The structure of (7.1) differs from related convex problems, for which efficient algorithms are available, only in $F_2 \circ G$ possibly being nonconvex. One would expect that such a strong analogy can be exploited. Indeed, for the algorithm we propose in this chapter we can show some favorable properties, including convergence of the function values and under some more assumptions convergence of the sequence of arguments. The numerical analysis demonstrates efficiency and robustness towards local optima. At the same time, the algorithm allows us to deal with several interesting nonconvex problems in image processing.

The proposed algorithm is in the fashion of classical majorization minimization algorithms. It generates and solves a sequence of convex optimization problems. The nonconvex part F_2 is, at each iteration, approximated by means of a majorizing convex surrogate function. Then, the resulting convex optimization problem is solved. As a matter of fact, the convex surrogate function has a structure that is amenable to efficient first-order methods for structured convex optimization. For example, in some cases, it permits the usage of the iPiasco algorithm from Chapter 3 for solving the surrogate function.

Although the convex subproblems are known to converge, it is not trivial to prove the convergence for the overall nonconvex problem. We show two convergence results. The first one establishes convergence for a subsequence of the sequence generated by our algorithm. This result is easily obtained and mainly stems from the fact that majorization minimization algorithms generate a sequence of nonincreasing function values. The second result states the convergence of the *whole* sequence. It requires a more

sophisticated analysis and some more assumptions like Lipschitz continuity of the gradient of F_2. For a subclass of Problem (7.1) satisfying these assumptions, convergence of the whole sequence of arguments to a critical point is proved. One part of the stronger regularity assumption is that the objective is a KL-function (see Section 4.5). This implies a sufficient descent property for gradient based methods also in areas where the function is flat around local optima. Our approach to proof convergence is showing that the requirements of [ABS13] (see Chapter 5) are satisfied.

In our numerical experiments the performance of the proposed algorithm is shown to be comparable to related algorithms for convex optimization, e.g., gradient descent or forward–backward splitting algorithms. Moreover, the proposed algorithm is easy to implement, which make it interesting for practical applications.

Problems like (7.1) arise frequently in image processing, computer vision or machine learning. The applicability of the iteratively reweighted ℓ_1 algorithm, which arises as a special case of the algorithm presented in this chapter, is demonstrated in [ODPB13]. Whereas the focus in that work was the diversity of applications such as denoising, deconvolution, depth map fusion, and optical flow, in this chapter we concentrate on the difference of modeling concepts in denoising and optical flow estimation. However, the concepts, how the nonconvex penalty functions are used, easily generalize to many other problems, e.g., deconvolution, depth map fusion, stereo estimation, or superresolution. Replacing convex penalty functions by nonconvex functions usually leads to better results. In particular, we analyze robust data-terms and the usage of edge-enhancing nonconvex penalizers. As a special instance, we are the first to propose a nonconvex extension of the total generalized variation regularizer [BKP10]. The total generalized variation (TGV) semi-norm is a convex penalizer that can reconstruct piecewise smooth functions. Due to the convexity of the regularizer, first- and higher-order discontinuities are only preserved but not enhanced. This may lead to over-smoothing effects in case of strong noise or weak data terms. It turns out that this effect can be partly avoided by using nonconvex penalizers in the TGV semi-norm.

Large parts of this chapter are published in [ODBP15].

7.1 Related work

Gradient descent based methods Steepest Descent, Quasi-Newton or Newton methods [Nes04, NW06, Ber99] are the classical approaches for general optimization and are also applicable in the nonconvex setting as long as the objective is smooth enough. An alternative are hill-climbing methods [TM93], annealing-type schemes [GG84], or graduated nonconvexity (GNC) [BZ87, Nik99]. However, efficiency of these methods leaves room for improvement. The worst-case complexity bound for general nonconvex problems derived in [Nes04] supports this statement. This means that there is only hope for efficient algorithms when considering nonconvex optimization problems of a specific structure.

In convex optimization problems with structure led to several efficient algorithms like Douglas-Rachford [DG64, EB92], forward–backward splitting [LM79, CW05, BT09a, Nes04], primal–dual approaches [CP11, PC11, HY12a], or augmented Lagrangian method [Ber99, Hes69, Pow69], which we discussed in Section 2.2.

While it seems to be difficult to generalize primal–dual approaches to the nonconvex setting directly [Val14, MSMC14], the augmented Lagrangian method is considered in [FW10, AFS13], the gradient projection method in [LP66, Gol64, ABS13], or a forward–backward splitting in [FM81, Sra12, Nes13, ABS13] were used for nonconvex optimization. In Chapter 6 the iPiano algorithm is introduced in the nonconvex setting. In our numerical experiments we will consider this algorithm, because several algorithms like gradient descent, projected gradient descent, the Heavy-ball method, and the forward–backward algorithm are special cases of it.

From another perspective, gradient descent method can also be interpreted as a majorization minimization (MM) algorithm [BT09b, LHY00, HL04]. The iteration step of the gradient descent method is equivalent to minimizing an isotropic quadratic upper bound—the quadratic upper bound that appears in the Descent Lemma (see e.g. Lemma 2.49). This way many algorithms can be considered as special instances of the MM algorithm. Also iPiano can be considered like that, see (6.3). Also expectation maximization algorithms are MM algorithms [FH94, FH95, JF07]. The MM principle can also be used for analytically estimating step size parameters for a given search direction (on a subspace) [CIM11, CJPT13, FCP+14]. In this context majorizers are mostly quadratic functions.

An important sub-class of the MM algorithms related to our algorithm is by Geman and Reynolds [GR92]. They rewrote a (smooth) nonconvex potential function as the infimum over a family of quadratic functions. This transformation suggests an algorithmic scheme that solves a sequence of quadratic problems, leading to the so-called iteratively reweighted least squares (IRLS) algorithm. This algorithm quickly became a standard solver and hence, it has been extended and studied in many works, see e.g. [VO98, NC07, DDFG10]. Convergence results can be found in [Idi01, AIG06].

The IRLS algorithm can only be applied if the nonconvex function can be well approximated from above with quadratic functions. However, this does not cover interesting functions such as $\log(1 + |x|)$ that are nondifferentiable at zero. Candes et al. [CWB08] tackled this problem by the so-called iteratively reweighted ℓ_1 (IRL1) algorithm. It solves a sequence of nonsmooth ℓ_1 problems and hence can be seen as nonsmooth counterpart to the IRLS algorithm. Originally, the IRL1 algorithm was proposed to improve the sparsity properties in ℓ_1 regularized compressed sensing problems.

First convergence results for the IRL1 algorithm have been obtained by Chen et al. in [CZ13] for a class of nonconvex ℓ_2-ℓ_p problems used in sparse recovery. In particular, they show that the method monotonically decreases the energy of the nonconvex problem. Unfortunately, the class of problems they considered is not suitable for typical computer vision problems, due to the absence of a linear operator that is needed in order to represent spatial regularization terms.

In [ODPB13], the convergence analysis of [CZ13] was generalized to linearly constrained optimization problems. This analysis made the algorithm and the theoretical results applicable to many computer vision problems. In this chapter, the algorithm will be generalized further and the convergence analysis will be extended a lot compared to [CZ13, ODPB13]. The convergence result is based on the analysis of the abstract descent algorithm from Chapter 5 and requires the objective function to be a Kurdyka–Łojasiewicz (KL) function. For details and references we refer to Section 4.5.

7.2 A class of optimization problems

We study a nonconvex optimization problem of a specific structure in a finite dimensional real vector space X of dimension $\dim(X) = N \in \mathbb{N}$. The standard inner product and norm are denoted $\langle \cdot, \cdot \rangle$ and $\| \cdot \|^2 := \langle \cdot, \cdot \rangle$, respectively. The optimization problem reads

$$\min_{x \in X} F(x) := \min_{x \in X} F_1(x) + F_2(G(x)) \,, \tag{7.2}$$

with a lower semi-continuous (lsc), extended real-valued, proper function $F \colon X \to \overline{\mathbb{R}}$ where $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. In addition we assume that F is bounded from below, i.e., $\inf_{x \in X} F(x) =: \underline{F} > -\infty$. We require that $F_1 \colon X \to \overline{\mathbb{R}}$ is proper, lsc, convex. Note that we explicitly allow the function F_1 to take on values at infinity, hence it can be for example the indicator function of a convex set. The function $G \colon X \to X_2$ maps from X into another finite dimensional real vector space X_2 with dimension $N_2 := \dim(X_2) \leq N$. We assume each coordinate function G_i, $i = 1, \ldots, N_2$, to be convex. The

function $F_2 \colon G(X) \to \mathbb{R}$ we assume coordinate-wise nondecreasing, i.e., $F_2(x) \leq F_2(x+\lambda e_i)$ whenever $x, x + \lambda e_i \in G(X)$ and $\lambda > 0$, where e_i is the i-th standard basis vector of X_2, $i = 1, \ldots, N_2$. We note that coordinate-wise convexity of G gives very simple structure to the set $G(X)$: it is Cartesian product of intervals, each infinite on one or both ends.

Example 7.1 (Denoising). Image denoising (see also Chapter 1) is a simple example from image processing that fits into the framework of (7.2). Given a noisy image $f \in X$ the goal of denoising is to find the image $u \in X$ such that $f = u + h$, where $h \in X$ is the noise that deteriorated the recording. Commonly, u instead of x denotes the optimization variable in image processing. The denoised image, i.e. the result, can be sought as minimizer of

$$\min_{u \in X} \lambda \|u - f\|_1 + \sum_i \log(1 + |(\nabla u)_i|)\,,$$

where $\lambda \in \mathbb{R}_+$ and $|\nabla u| \in G(X)$ denotes, as described in Section 0.2, the vector composed of coordinates $|(\nabla u)_i| := \sqrt{((\partial_x u)_i)^2 + ((\partial_y u)_i)^2}$, where $\partial_x u$ is a discrete implementation of the x-derivative of the image (considered as a function $\mathbb{R}^2 \to \mathbb{R}$). The first term measures the discrepancy between the measurements and the sought denoised image. It is called *data-term* and given by the proper, convex (nonsmooth) function $F_1(u) = \lambda \|u - f\|_1$ in (7.2). The second term, called the *regularization-term* invokes some prior knowledge about natural image statistics. The use of the nonconvex function $F_2(y) = \sum_i \log(1 + y_i)$ on $G(X)$ for the regularization-term stresses the general property of images of being smooth and having some sharp jump discontinuities. Obviously, F_2 is coordinate-wise nondecreasing on $G(X)$ (see Figure 1.3). The coordinate functions $G_i(u) = |(\nabla u)_i|$ are convex and make $F_2 \circ G$ nonsmooth.

Finding the global minimum of a nonsmooth nonconvex function, as in (7.2), is in general not feasible. We hence only aim to find *a critical point* of the function F, i.e. $x \in X : 0 \in \partial F(x)$, where ∂ denotes the limiting-subgradient (Definition 4.12). Critical points are connected to local minima of the function by *Fermat's rule* (see Theorem 4.23).

7.3 Iterative convex majorization minimization

In this chapter, we study a sub-class of majorization minimization (MM) methods that is suitable for solving the minimization problem (7.2). The idea of MM algorithms is to minimize majorizers of the function instead of the function itself. The major challenge is the construction of majorizing functions that are easier to minimize than the original function. Invoking only some weak assumptions about the structure of the optimization problem (7.2) as done above, makes it possible to design such majorizing functions.

We propose to majorize F_2 with a convex function $F_2^{x^n}$ that approximates F_2 such that $F_2^{x^n} \circ G$ is convex and meets $F_2 \circ G$ at x^n. More formally, consider the generic Method 7. Nondecreasingness of F_2^x provides convexity of the composition $F_2^x \circ G$. As the above formulation is rather abstract, we exemplify the algorithm.

Example 7.2. We consider a simplified problem of Example 7.1 and restrict $X = X_2 = \mathbb{R}$, $G(x) = |x|$:

$$\min_{x \in \mathbb{R}} F(x) = \min_{x \in \mathbb{R}} F_1(x) + F_2(x) = \min_{x \in \mathbb{R}} 2|x - 1| + \frac{1}{2} \log(1 + 25x^2)\,. \tag{7.5}$$

Figure 7.1 visualizes one update step using (7.4) at $x^n = -0.5$. For details on how to choose the surrogate function we refer to the specialized algorithms (here Algorithm 11) introduced in the following subsections.

Method 7. *Iterative convex majorization minimization method*

- *Initialization: Choose a starting point $x^0 \in X$ with $F(x^0) < \infty$ and define a suitable family of convex surrogate functions $(F_2^x)_{x \in X}$, such that for all $x \in X$ holds $F_2^x \in \mathcal{F}_{2,G}(x)$, where*

$$\mathcal{F}_{2,G}(x) := \left\{ f \colon X_2 \to \mathbb{R} \;\middle|\; \begin{array}{c} f \text{ proper, convex,} \\ f \text{ nondecreasing on } G(X), \\ f(G(x)) = F_2(G(x)), \\ \forall y \in G(X) \colon f(y) \geq F_2(y) \end{array} \right\}. \tag{7.3}$$

- *Iterations $(n \geq 0)$: Update*

$$x^{n+1} = \arg\min_{x \in X} F_1(x) + F_2^{x^n}(G(x)) \,. \tag{7.4}$$

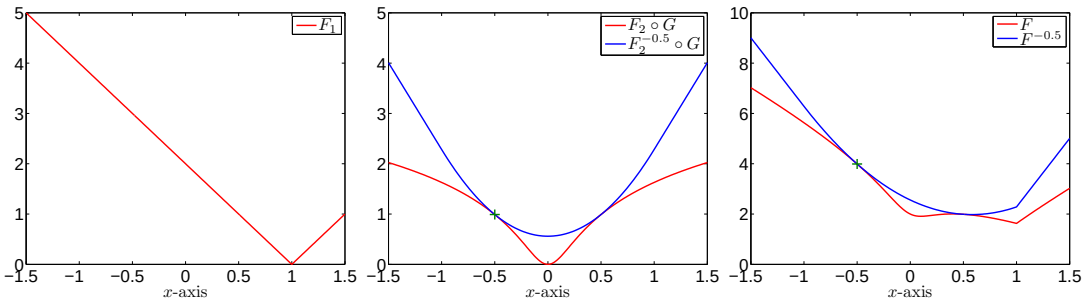


Figure 7.1: Visualization of one update step (7.4) for optimization problem (7.5) at $x^n = -0.5$. LEFT: Convex function F_1. MIDDLE: Nonconvex function $F_2 \circ G$ (red) and its convex majorizer $F_2^{-0.5} \circ G$ (blue). RIGHT: The function $F = F_1 + F_2 \circ G$ (red) and its majorizer $F^{-0.5} = F_1 + F_2^{-0.5} \circ G$ (blue). The blue function in the right plot is to be minimized to obtain x^{n+1}.

We call the Method 7 "generic", because it still requires the choice of suitable convex surrogate functions. As F_1 is already convex, the approximation $F_2^{x^n}$ of F_2 is the focus of attention. A choice of approximation follows from the following property of MM algorithms.

Proposition 7.1. *Let $(x^n)_{n \in \mathbb{N}}$ be generated by Method 7 and let for all $x \in X$ be $F_2^x \in \mathcal{F}_{2,G}(x)$. Then, the sequence $(F(x^n))_{n \in \mathbb{N}}$ monotonically decreases and converges.*

Proof. The proof directly follows from F being bounded from below by $\underline{F}$ and the definitions of x^n and $F_2^{x^n}$:

$$\underline{F} \leq F(x^{n+1}) \leq F_1(x^{n+1}) + F_2^{x^n}(G(x^{n+1})) \leq F_1(x^n) + F_2^{x^n}(G(x^n)) = F(x^n) \,.$$

The sequence $(F(x^n))_{n \in \mathbb{N}}$ decreases and is bounded from below. Hence, it converges. □

Clearly, at each iteration the value of the function F decreases at least as much as the value of the majorizing function. This suggests to use surrogate functions whose minimum is minimal. Of course, it could happen that another surrogate function with a higher minimum yields a lower value of the original function, however, there is no guarantee. Finding the optimal approximation according to the *criterion of guaranteed maximal decrease of function values* is hard. In general, a majorizer $f \in \mathcal{F}_{2,G}(x^n)$ that is not the sum of F_1 and another convex function can have a lower minimum than our approximation. However, this better majorizer may be complex to construct and difficult to optimize. Thus, we aim to fulfill the

criterion of guaranteed maximal decrease of function values in the class of surrogate functions that are sum of F_1 and another convex function. If we talk about *optimal majorizers* in the following, we mean optimality according to the guaranteed decrease of function values in this class. These are majorizers (constructed under certain conditions) with the lowest minimum.

The different algorithms that will be presented in the following section take into account the characteristics of F_2. Considering again Figure 1.3, it is obvious that there is no simple universal choice for the surrogate function. For instance, close to 0 the nondifferentiable functions in Figure 1.3 may be well approximated by the absolute value function, whereas this is a bad choice for the functions on the top right of Figure 1.3. Different functions require different construction principles of the majorizer. In this paper, we show constructions of majorizers, which address (7.2). Some of them are proved to be optimal in the sense explained above. Method 7 serves as a tool to prove their convergence in a unified framework. However, it covers many other possible constructions (thus also convergence) of majorizers that are not explicitly presented here.

7.4 Iteratively reweighted convex algorithms

As the function F_2 in the optimization problem (7.2) does not change, it may be possible to find a single convex function that, weighted appropriately, can serve as majorizer for F_2 at each step of the Method 7. This is the principle of iteratively reweighted algorithms. The construction of majorizers according to this principle is easier than for the very general Method 7 and allows for explicit algorithms. The reweighting algorithms considered in the subsequent subsections are all special cases of the IRconvex Method 8, which is an instance of Method 7.

Method 8. ***I**teratively **r**eweighted **convex** method (IRconvex)*

- *Initialization: Define a convex function $F_2^c\colon G(X) \to \mathbb{R}^{N_2'}$, $N_2' \in \mathbb{N}$, and a family of vectors $(w^x)_{x\in X}$ such that*
$$y \mapsto \langle w^x, F_2^c(y)\rangle \in \mathcal{F}_{2,G}(x), \quad x \in X\,,$$
and starting point $x^0 \in X$ with $F(x^0) < \infty$.
- *Iterations ($n \geq 0$): Update*
$$x^{n+1} = \arg\min_{x\in X} F_1(x) + \left\langle w^{x^n}, F_2^c(G(x))\right\rangle \tag{7.6}$$

Remark 7.3. As the optimization problem in (7.6) is independent of constants, $y \mapsto \langle w^x, F_2^c(y)\rangle$ may be in $\mathcal{F}_{2,G}(x)$ only after adding a constant. Formally this could be achieved by setting $\tilde{F}_2^c := (F_2^c, 1)$ and $\tilde{w}^x := (w^x, a)$, where $a \in \mathbb{R}$. Being aware of it now, subsequently, we will simply neglect the constant.

7.4.1 Iteratively reweighted ℓ_1 algorithm

Algorithm 9, the iteratively reweighted ℓ_1 algorithm, will be shown to be optimal for the optimization problem (7.2) in a certain sense, when F_2 is concave on $G(X)$. We denote here by $\bar{\partial} f := -\partial(-f)$ the *limiting-supergradient*, the analogue of the limiting-subgradient but for concave functions. The usage of $\bar{\partial}$ instead of ∂ makes the Algorithm slightly more general. For F_2 concave it is $\partial F_2(x) \subset \bar{\partial} F_2(x)$ on the interior of $G(X)$.

Algorithm 9. ***I**teratively **r**eweighted ℓ_1 algorithm (IRL1)*

- *Assumption: F_2 is concave on $G(X)$.*
- *Initialization: Define a family of vectors $(w^x)_{x\in X}$ with*

$$w^x \in \bar{\partial} F_2(y), \quad y = G(x), \quad x \in X\,,$$

and starting point $x^0 \in X$ with $F(x^0) < \infty$.

- *Iterations $(n \geq 0)$: Update*

$$x^{n+1} = \arg\min_{x\in X} F_1(x) + \left\langle w^{x^n}, G(x)\right\rangle \tag{7.7}$$

Remark 7.4. The functions $x \mapsto \left\langle w^{x^n}, G(x)\right\rangle$ may not be majorizers of F_2. However, they become majorizers when shifted by suitable constants which do not affect (7.7). The exact majorizer will be considered in Proposition 7.2.

Example 7.5. We consider the same optimization problem as in Example 7.1. Obviously, $F_2(y) = \sum_i \log(1+y_i)$ is nondecreasing and concave on $G(X) = [0;+\infty)$, $F_1(u) = \lambda\|u-f\|_1$ and $G_i(u) = |(\nabla u)_i|$ are convex. For $u^n \in X$ the vectors w^{u^n} in Algorithm 9 read $w_i^{u^n} = 1/(1+|(\nabla u^n)_i|)$ which is defined as $|(\nabla u^n)_i| := \sqrt{((\partial_x u^n)_i)^2 + ((\partial_y u^n)_i)^2}$, and the convex surrogate optimization problem in (7.7) reads

$$\min_{u\in X}\ \lambda\|u-f\|_1 + \sum_i w_i^{u^n} |(\nabla u)_i|\,.$$

Each of these subproblems is a denoising problem with total variation regularization with coordinates differently weighted.

As discussed before, in general, it is hard to construct the best surrogate function according to the criterion of guaranteed maximal decrease of function values. However, assuming that F_2 is concave on $G(X)$ it is possible and used in Algorithm 9.

Proposition 7.2. *If $\mathcal{F}_{2,G}(x^n)$ is defined as in* (7.3) *and F_2 is concave on $G(X)$ and differentiable at $G(x^n)$, then the optimal majorizer of $F_2 \circ G$ at x^n*

$$\arg\min_{f\in\mathcal{F}_{2,G}(x^n)} \left(\min_{x\in X} f(G(x))\right)$$

is given by

$$\hat{F}_2(y) = \langle \nabla F_2(G(x^n)), y - G(x^n)\rangle + F_2(G(x^n)).$$

Moreover, $F_1 + \hat{F}_2 \circ G$ is also the optimal majorizer of F among majorizers of F corresponding to majorizers of $F_2 \circ G$ from the class $\mathcal{F}_{2,G}(x^n)$.

Proof. Due to concavity of F_2, the function $\hat{F}_2$ is a majorizer of F_2. It also clearly fulfills all other conditions to belong to the class $\mathcal{F}_{2,G}(x^n)$. On the other hand, for any convex function f such that $f(G(x^n)) = F_2(G(x^n))$ and $f(y) \geq F_2(y)$ for all $y \in G(X)$ we have $f(y) \geq \hat{F}_2(y)$ for all $y \in G(X)$. Indeed, suppose there exists y^* such that $f(y^*) < \hat{F}_2(y^*)$. Then differentiability of F_2 at $G(x^n)$ implies that there exists $t^* \in (0,1)$ such that

$$t^* f(y^*) + (1-t^*) f(G(x^n)) < F_2(t^* y^* + (1-t^*) G(x^n)) \leq f(t^* y^* + (1-t^*) G(x^n)).$$

This contradicts convexity of f, hence, our supposition was not valid, and $f(x) \geq \hat{F}_2(x)$ for all $x \in X$. Therefore, $f(G(x)) \geq \hat{F}_2(G(x))$ for all $x \in X$, which immediately gives

$$\min_{x \in X} f(G(x)) \geq \min_{x \in X} \hat{F}_2(G(x)),$$

i.e. $\hat{F}_2 \circ G$ if the best majorizer of F_2. Moreover, $F_1(x) + f(G(x)) \geq F_1(x) + \hat{F}_2(G(x))$ for all $x \in X$ and hence $F_1 + \hat{F}_2 \circ G$ is also the optimal majorizer of F. □

7.4.2 Iteratively reweighted tight convex algorithm

The iteratively reweighted ℓ_1 Algorithm 9 is optimal for a certain class of functions, but not applicable to many other practically interesting cases, such as for example $F_2(|x|) = \log(1+|x|^2)$ (see Figure 1.3). The reason is that close to 0 this prototype function is strongly convex and hence not majorized by tangents. Fortunately, the structure of this function allows for simple and tight majorizers. Namely, let us consider the *class* $\mathcal{F}_{cc}$ consisting of functions $f\colon \mathbb{R}^N_+ \to \mathbb{R}$ such that:

(i) f is additively separable, i.e. $f(x_1, \ldots, x_N) = f_1(x_1) + \ldots + f_N(x_N)$,

(ii) every f_j is convex in the *convexity region* $[0,\, r_j]$ and concave in the *concavity region* $[r_j,\, +\infty)$ for some $r_j \geq 0$.

For simplicity we also suppose that there exist left and right derivatives $f'_j(r_j^-)$ and $f'_j(r_j^+)$. Then for $f \in \mathcal{F}_{cc}$ we denote $s_j = \max\left(f'_j(r_j^-), f'_j(r_j^+)\right)$ and define the following functions:

$$t_j(x_j) = \begin{cases} f_j(x_j), & \text{if } x_j \leq r_j, \\ f_j(r_j) + s_j(x_j - r_j), & \text{if } x_j > r_j. \end{cases} \tag{7.8}$$

We set $T_f(x) = (t_1(x_1), \ldots, t_N(x_N))^\top$ to be the vector of all these functions. Each t_j majorizes corresponding f_j because in the convexity region these two functions coincide, while in the concavity region t_j majorizes the tangent $f_j(r_j) + f'_j(r_j^+)(x_j - r_j)$ of the concave function f_j. Moreover, each t_j is convex by construction. We hence can plug T into Method 8, yields Algorithm 10.

Algorithm 10. ***I**teratively **r**eweighted **tight** convex algorithm (IRTight)*

- *Assumption:* $F_2 \in \mathcal{F}_{cc}$.
- *Initialization: Define a family of vectors* w^x *defined for all* $i = 1, \ldots, \dim(X_2)$ *by*

$$w_i^x = \begin{cases} 1, & y_i \leq r_i \\ \frac{(v^x)_i}{t'_i(y_i)}, & y_i > r_i \end{cases}, \quad v^x \in \bar{\partial} F_2(y)\,, \quad y = G(x), \quad x \in X$$

  and starting point $x^0 \in X$ *with* $F(x^0) < \infty$.

- *Iterations* $(n \geq 0)$*: Update*

$$x^{n+1} = \arg\min_{x \in X} F_1(x) + \left\langle w^{x^n}, T_{F_2}(G(x)) \right\rangle \tag{7.9}$$

As we already have shown, the functions t_j majorize corresponding f_j. Weighting the functions with w_i^x does not remove the majorization property. More precisely, if $v_i \in \bar{\partial} f_i(y_i^0),\ w_i = v_i \cdot (t'_i(y_i^0))^{-1}$, then $w_i t_i(y_i) + f_i(y_i^0) - w_i t_i(y_i^0) \geq f(y_i)$ for all y_i.

7.4.3 Iteratively reweighted Huber algorithm

Consider the same class of functions $\mathcal{F}_{cc}$ as in the previous subsection. In practice it is beneficial when the majorizing function has simple analytic form. This may not be the case for tight convex majorizers introduced above. However, a wide class of functions can be majorized with help of the Huber function, which is defined as:

$$h_\varepsilon(\|x\|) = \begin{cases} \frac{1}{2\varepsilon}\|x\|^2, & \text{if } \|x\| \le \varepsilon \\ \|x\| - \frac{\varepsilon}{2}, & \text{otherwise.} \end{cases} \tag{7.10}$$

We define $H_\varepsilon(y) := (h_\varepsilon(y_1), \dots, h_\varepsilon(y_K))$, which applies (7.10) coordinate-wise. Supposing that F_2 is differentiable on an open superset of $G(X)$, we can formulate Algorithm 11.

Algorithm 11. ***I**teratively **r**eweighted **Huber** algorithm (IRHuber)*

- *Assumption:* $F_2 \in \mathcal{F}_{cc}$.
- *Initialization: Define a family of vectors* w^x *defined for all* $i = 1, \dots, \dim(X_2)$ *by*

$$w_i^x = \frac{(\nabla F_2(y))_i}{h'_\varepsilon(y_i)}, \quad y = G(x), \quad x \in X$$

  and starting point $x^0 \in X$ *with* $F(x^0) < \infty$.

- *Iterations* $(n \ge 0)$*: Update*

$$x^{n+1} = \arg\min_{x \in X} F_1(x) + \left\langle w^{x^n}, H_\varepsilon(G(x)) \right\rangle \tag{7.11}$$

Example 7.6. Consider the optimization problem

$$\min_{u \in X} \lambda \|u - f\|_1 + \tfrac{1}{2} \sum_i \log(1 + |(\nabla u)_i|^2)\,,$$

where the convention for $|(\nabla u)_i|$ is as in Example 7.5. The only difference to Example 7.5 is the square in the second term. However, as mentioned already, this makes IRL1 an unsuitable choice (see Remark 7.7). The term $F_2(G(x)) = \frac{1}{2}\sum_i \log(1 + |(\nabla u)_i|^2)$ with $G(x) = |(\nabla u)_i|$ is better approximated using the Huber function, which is quadratic close to 0. Vice verse, approximating the function from Example 7.5 with the Huber function is also a bad choice.

Obviously, F_2 is smooth and belongs to the class $\mathcal{F}_{cc}$. In order to write down the surrogate function we need to calculate the derivative of F_2. For all i, it is $(\nabla F_2(y))_i = y_i/(1 + y_i^2)$. The weights are chosen such that the surrogate function has the same slope as ∇F_2 at u^n. As the Huber function has the derivative $(\nabla H_\varepsilon(y))_i = y_i/\varepsilon$, if $|y_i| \le \varepsilon$, and $(\nabla H_\varepsilon(y))_i = y_i/|y_i|$ otherwise, the weight vector w^{u^n} is inferred as

$$w_i^{u^n} = \frac{\max\{\varepsilon, |(\nabla u^n)_i|\}}{1 + |(\nabla u^n)_i|^2}\,.$$

Remark 7.7. Within our framework there are different ways to approximate the function $F_2(G(x)) = \log(1 + |x|^2)$. We consider this in 1D here. The option we used in the preceding example corresponds to setting $G(x) = |x|$, $F_2(y) = \log(1 + y^2)$ and approximating $F_2(y)$ using the Huber function. However, we could also set $G(x) = |x|^2$, $F_2(y) = \log(1 + y)$ and approximate $F_2(y)$ as in the IRL1 algorithm. Then, the (convex) surrogate function to be minimized in the IRL1 algorithm is $F_1(x) + w^{x^n} G(x) = F_1(x) + w^{x^n}|x|^2$ and the approximation is by a quadratic function and hence worse than the approximation

with the Huber function. This choice of G and F_2 corresponds to the well-known iteratively reweighted least squares algorithm (IRLS), which we will recap in Section 7.4.4.

A natural question arises: which of the two interpretations of the problem leads to better results. We argue that setting $G(x) = |x|$, $F_2(y) = \log(1 + y^2)$ is better. In this case we can approximate $F_2(y)$ by the Huber function $H_\varepsilon(y)$, and the corresponding approximation for $G(x) = |x|^2$, $F_2(y)$ would be $H_\varepsilon(\sqrt{y})$. However, $H_\varepsilon(\sqrt{y})$ in nonconvex and therefore not feasible for our iterative, convex algorithm (Method 8). This suggests to always choose F_2 and G such that G is as close as possible to $|x|$ (i.e. "as nonconvex as possible"). In this case $F_2^{x^n} \circ G$ can better approximate $F_2 \circ G$. For instance, IRHuber is a better approximation than IRLS.

As the majorization property of IRHuber is not immediately clear in this setup, we prove a general condition under which it holds and verify it for the preceding example.

Proposition 7.3. *Suppose $f\colon X \to \mathbb{R}$ and $m\colon X \to \mathbb{R}$, $X \subset \mathbb{R}$ open, are continuously differentiable nondecreasing functions and there exists a nonincreasing function $r\colon \mathbb{R} \to \mathbb{R}_+$ such that $f'(x) = r(x)\, m'(x)$. Then for every $x_0 \in \mathbb{R}$ the function $m_{x_0}(x) = r(x_0)\, m(x) + f(x_0) - r(x_0)\, m(x_0)$ majorizes the function f.*

Proof. Obviously, $f(x_0) = m_{x_0}(x_0)$ and $f'(x_0) = m'_{x_0}(x_0)$. We then have for $x > x_0$:

$$m_{x_0}(x) - f(x) = \int_{x_0}^{x} \big(m'_{x_0}(t) - f'(t)\big)\, dt = \int_{x_0}^{x} \big((r(x_0) - r(t))m'(t)\big)\, dt \geq 0.$$

Similarly, for $x < x_0$:

$$m_{x_0}(x) - f(x) = -\int_{x}^{x_0} \big(m'_{x_0}(t) - f'(t)\big)\, dt = -\int_{x_0}^{x} \big((r(x_0) - r(t))m'(t)\big)\, dt \geq 0.$$

□

We now apply this proposition to the special case $f(x) = \log(1 + \mu x^2),\ m(x) = h_\varepsilon(x)$. Since both functions are symmetric, we only consider $x \geq 0$. We then have:

$$\begin{aligned}
f'(x) &= \frac{2\mu x}{1 + \mu x^2}, \\
m'(x) = \min\left(\frac{x}{\varepsilon}, 1\right) &= \begin{cases} \frac{x}{\varepsilon}, & 0 \leq x \leq \varepsilon, \\ 1, & x > \varepsilon, \end{cases} \\
r(x) = \frac{f'(x)}{m'(x)} &= 2\mu\, \frac{\max(x, \varepsilon)}{1 + \mu x^2}, \\
r'(x) &= 2\mu \begin{cases} -\frac{2\mu\varepsilon x}{(1+\mu x^2)^2}, & 0 \leq x \leq \varepsilon, \\ \frac{1-\mu x^2}{(1+\mu x^2)^2}, & x > \varepsilon. \end{cases}
\end{aligned}$$

Obviously, r is nonincreasing as soon as $\varepsilon \geq \frac{1}{\sqrt{\mu}}$.

Algorithm 12. ***I**teratively **r**eweighted **l**east **s**quares algorithm (IRLS)*

- *Assumption:* $F_2 \in \mathcal{F}_{cc}$.
- *Initialization: Define a family of vectors* w^x *defined for all* $i = 1, \dots, \dim(X_2)$ *by*

$$w_i^x = \frac{(\nabla F_2(y))_i}{y_i}, \quad y = G(x), \quad x \in X$$

and starting point $x^0 \in X$ *with* $F(x^0) < \infty$.

- *Iterations* $(n \geq 0)$*: Update*

$$x^{n+1} = \arg\min_{x \in X} F_1(x) + \left\langle w^{x^n}, \tfrac{1}{2}(G(x))^2 \right\rangle , \tag{7.12}$$

where the square is to be understood coordinate-wise.

7.4.4 Iteratively reweighted least squares algorithm

The well-known IRLS Algorithm does also arise as a special case of Method 8. We present it in Algorithm 12 using our notation. Obviously, it is applicable at least to the same class of problems as Algorithm 11. Thus, the majorization property is clear.

Example 7.8. Consider Example 7.6. Using the IRLS algorithm the weight vector w^{u^n} is given by

$$w_i^{u^n} = \frac{1}{1 + |(\nabla u^n)_i|^2} .$$

However, the quadratic function is a worse approximation of the nonconvex norm than the Huber function. We hence expect IRHuber to outperform IRLS.

7.4.5 Convergence analysis

Throughout the whole convergence analysis, let $(x^n)_{n \in \mathbb{N}}$ be a sequence generated by Method 7. We also always suppose that the functions F, F_1, F_2, G fulfill the conditions stated in Section 7.2. We make frequent use of the tools from variational analysis presented in Chapter 4. In addition, from now on we assume F to be coercive (Definition 2.27).

Proposition 7.4. *Let* F *be coercive, then the sequence* $(x^n)_{n \in \mathbb{N}}$ *is bounded and has at least one accumulation point.*

Proof. By Proposition 7.1, the sequence $(F(x^n))_{n \in \mathbb{N}}$ is monotonically decreasing, therefore the sequence $(x^n)_{n \in \mathbb{N}}$ is contained in the level set

$$\mathcal{L}(x^0) := \{x \in X | \, F(x) \leq F(x^0)\} .$$

From coercivity of F we conclude boundedness of the set $\mathcal{L}(x^0)$. This allows to apply the Theorem of Bolzano-Weierstraß, which gives the existence of a converging subsequence and, hence, an accumulation point. □

Additional assumptions. In order to prove convergence for the whole sequence $(x^n)_{n\in\mathbb{N}}$, two additional assumptions are required. Let us discuss them.

(i) We assume that F_2 has locally Lipschitz continuous gradient (see Definition 4.19) on a compact set B containing all the points x^n and that F_2^x have globally Lipschitz continuous gradients on B for all $x \in X$ with a common Lipschitz constant $\widetilde{L} \geq 0$. This assumption is less restrictive as it seems to be. Many nonsmooth functions may be written as a sum of a function with locally Lipschitz gradient and a convex function. Then, the convex part may be shifted to F_1. This class of functions was for example considered in [AFS13].

(ii) $F_1 + F_2^{x^n} \circ G$ must be strongly convex (see Definition 2.45) with a constant independent of n. Otherwise, the sum $F_1 + F_2^{x^n} \circ G$ can have a plateau as local minimum, i.e., there is no unique minimizer. This happens for example when $F_1(x) = |x - 1|$ and $F_2^{x^n}(G(x)) = |x|$ for all $x^n \in [0, 1]$. Our algorithm then has to choose from multiple equally good solutions and hence may not converge. One standard way to resolve this problem is to add a proximity term $c\|x - x^n\|^2$ to the convex surrogate problem (7.4) with arbitrarily small $c > 0$. This makes the surrogate problem strongly convex and makes the algorithm converge to one solution from the plateau.

A technical assumption we make from now on is that F_2 and F_2^x for all $x \in X$ are defined on open sets comprising $G(X)$ and continuously differentiable on $G(X)$. In all practical cases this is clearly fulfilled. The following properties then hold:

Lemma 7.5. *Under the aforementioned conditions, it holds for all $\bar{x} \in X$*

(i) and for all $x \in X$

$$\partial(F_2^{\bar{x}} \circ G)(x) = \partial \langle y, G\rangle (x) \text{ with } y = \nabla F_2^{\bar{x}}(x)\,,$$
$$\partial(F_2 \circ G)(x) = \partial \langle y, G\rangle (x) \text{ with } y = \nabla F_2(x)\,,$$

(ii) and for all $x \in \operatorname{dom} F_1$ and all $x \in X$

$$\partial(F_1 + F_2^{\bar{x}} \circ G)(x) = \partial F_1(x) + \partial(F_2^{\bar{x}} \circ G)(x)\,,$$
$$\partial(F_1 + F_2 \circ G)(x) = \partial F_1(x) + \partial(F_2 \circ G)(x)\,.$$

Proof. We verify the second equality for both items. The first one follows analogously.

(i) Since F_2 is continuously differentiable on an open set containing $G(X)$, for $x \in X$ it holds that $\partial^\infty F_2(G(x)) = \{0\}$ [RW98, Ex. 8.8]. Continuous differentiability also yields regularity of F_2 at $G(x)$ for $x \in X$ [RW98, Ex. 7.28]. By assumption F_2 is coordinate-wise nondecreasing, which implies that $\langle y, G\rangle (x)$ with $y = \nabla F_2(x)$ is a lsc., convex function. As a consequence, $\langle y, G\rangle (x)$ is regular at $x \in X$, which verifies the conditions for equality in Proposition 4.22. As a side product, $F_2 \circ G$ is regular for all $x \in X$.

(ii) Convexity of G implies its local Lipschitz continuity [RW98, Ex. 9.14] and, hence, also local Lipschitz continuity of $F_2 \circ G$. Therefore, $\partial^\infty(F_2 \circ G)(x) = \{0\}$ (see Proposition 4.20), which together with convexity of F_1 (hence $\partial F_1(x) = \widehat{\partial} F_1(x)$) and Clarke regularity of $F_2 \circ G$ at x (see first point in this proof) ensures $\partial F_1(x) + \partial(F_2 \circ G)(x) = \partial(F_1 + F_2 \circ G)(x)$ (see Proposition 4.21).

□

Proposition 7.6. *Let B be a bounded set containing all x^n. Let F_2 have locally Lipschitz continuous gradient on B and let F_2^x have globally Lipschitz continuous gradients on B for all $x \in X$ with a common Lipschitz constant $\widetilde{L} \geq 0$. Let also $F_1 + F_2^{x^n} \circ G$ be strongly convex with convexity parameter $\mu > 0$ for all $x^n \in X$. Then, the following holds*

(i) $F(x^{n+1}) \leq F(x^n) - \frac{\mu}{2}\|x^n - x^{n+1}\|^2$ *for all* $n \in \mathbb{N}$,

(ii) there exists $C > 0$ *such that for all* $n \in \mathbb{N}$ *there exists* $v^{n+1} \in \partial F(x^{n+1})$ *fulfilling*

$$\|v^{n+1}\| \leq C\|x^{n+1} - x^n\|\,,$$

(iii) and for any converging subsequence $(x^{n_j})_{j\in\mathbb{N}}$ *with* $\bar{x} := \lim_{j\to\infty} x^{n_j}$ *holds* $F(x^{n_j}) \to F(\bar{x})$ *as* $j \to \infty$.

Proof. (i) The strong convexity of $F_1 + F_2^{x^n} \circ G$ provides for all $v_1 \in \partial F_1(x^{n+1})$ and $v_2^n \in \partial(F_2^{x^n} \circ G)(x^{n+1})$ the inequality

$$\begin{aligned} F_1(x^{n+1}) - F_1(x^n) + F_2^{x^n}(G(x^{n+1})) - F_2^{x^n}(G(x^n))) \\ \leq \left\langle v_1, x^n - x^{n+1}\right\rangle + \left\langle v_2^n, x^n - x^{n+1}\right\rangle - \frac{\mu}{2}\|x^{n+1} - x^n\|^2\,, \end{aligned}$$

As x^{n+1} is a minimizer of (7.4) and thanks to Lemma 7.5, we can choose $v_1 + v_2^n = 0 \in \partial(F_1 + F_2^{x^n} \circ G)(x^{n+1})$. Using $F_2(G(x^{n+1})) \leq F_2^{x^n}(G(x^{n+1}))$ and $F_2(G(x^n)) = F_2^{x^n}(G(x^n))$, we conclude this part of the proof.

(ii) Local Lipschitz continuity of G (which follows from its convexity) and the gradient of F_2 provides their global Lipschitz continuity on B. We denote the corresponding Lipschitz constants by L_G and L respectively.

Using Lemma 7.5, we can select $v_1 \in \partial F_1(x^{n+1})$ and $v_2^n \in \partial(F_2^{x^n} \circ G)(x^{n+1})$ such that

$$v_1 + v_2^n = 0 \in \partial(F_1 + F_2^{x^n} \circ G)(x^{n+1}) = \partial F_1(x^{n+1}) + \partial(F_2^{x^n} \circ G)(x^{n+1})\,.$$

Then, for all $v_2 \in \partial(F_2 \circ G)(x^{n+1})$ it holds

$$\|v_1 + v_2\| = \|v_1 + v_2 - v_1 - v_2^n\| = \|v_2 - v_2^n\|\,. \tag{7.13}$$

Using the chain rule from Proposition 4.22 and Lemma 7.5, we have (define $y^n := \nabla F_2^{x^n}(G(x^{n+1}))$)

$$\partial(F_2^{x^n} \circ G)(x^{n+1}) = \partial\left\langle y^n, G\right\rangle(x) = \sum_i \partial(y_i^n G_i)(x^{n+1}) = \sum_i y_i^n \partial G_i(x^{n+1})$$

and, thus, we can decompose $v_2^n = \sum_i y_i^n \eta_i$ with $\eta_i \in \partial G_i(x^{n+1})$. We then define $v_2 := \sum_i y_i \eta_i$, where $y := \nabla F_2(G(x^{n+1}))$. The combination of both decompositions together with the Lipschitz continuity of G and [RW98, Prop. 9.24] yields

$$\|v_2 - v_2^n\| = \|\sum_i (y - y^n)_i \eta_i\| \leq L_G\|y - y^n\|\,. \tag{7.14}$$

Now, using (7.13) and (7.14), the equality $\nabla F_2(G(x^n)) = \nabla F_2^{x^n}(G(x^n))$, the Lipschitz continuity of ∇F_2 and $\nabla F_2^{x^n}$ and noting that $v_1 + v_2 \in \partial F(x^{n+1})$, the following estimation concludes this part of the proof:

$$\begin{aligned} \|v_1 + v_2\| &\leq L_G\|y - \nabla F_2(G(x^n)) + \nabla F_2^{x^n}(G(x^n)) - y^n\| \\ &\leq (L + \widetilde{L})L_G\|G(x^{n+1}) - G(x^n)\| \\ &\leq (L + \widetilde{L})L_G^2\|x^{n+1} - x^n\|\,, \end{aligned}$$

where the last transition follows from the Lipschitz continuity of G.

(iii) Let $(x^{n_j})_{j\in\mathbb{N}}$ be a converging subsequence of $(x^n)_{n\in\mathbb{N}}$. Define the sequences $(v_1^{n_j})_{j\in\mathbb{N}}$ and $(v_2^{n_j})_{j\in\mathbb{N}}$ by $0 = v_1^{n_j} + v_2^{n_j} \in \partial F_1(x^{n_j}) + \partial(F_2^{x^{n_j-1}} \circ G)(x^{n_j})$, which by Lemma 7.5 coincides with $\partial(F_1 + F_2^{x^{n_j-1}} \circ G)(x^{n_j})$. Due to the local Lipschitz continuity of $F_2^{x^{n_j-1}} \circ G$ Proposition 4.20 implies $x \mapsto \partial(F_2^{x^{n_j-1}} \circ G)(x)$ bounded, and therefore, the sequence $(v_1^{n_j})_{j\in\mathbb{N}}$ is bounded and $\lim_{j\to\infty} \langle v_1^{n_j}, \bar{x} - x^{n_j} \rangle = 0$. Using this, F lsc, F_1 convex, and $F_2 \circ G$ locally Lipschitz continuous, the following chain of inequalities concludes the proof (all limits are considered for $j \to \infty$):

$$\begin{aligned} F(\bar{x}) &\leq \liminf F(x^{n_j}) \leq \limsup F(x^{n_j}) \\ &\leq \limsup F_1(x^{n_j}) + \limsup F_2(G(x^{n_j})) \\ &= \limsup F_1(x^{n_j}) + \lim \langle v_1^{n_j}, \bar{x} - x^{n_j} \rangle + F_2(G(\bar{x})) \\ &= \limsup \left(F_1(x^{n_j}) + \langle v_1^{n_j}, \bar{x} - x^{n_j} \rangle\right) + F_2(G(\bar{x})) \\ &\leq F_1(\bar{x}) + F_2(G(\bar{x})) = F(\bar{x})\,. \end{aligned}$$

□

In [ABS13], an abstract convergence result for descent methods for semi-algebraic and tame problems is proved (see Chapter 5). The notion of semi-algebraic functions and the KL property was introduced in Section 4.5. In the following theorem, we benefit from their convergence analysis by simply proving our algorithm to satisfy their assumptions.

Theorem 7.7. *Let the assumptions be as in Proposition 7.6. Let the sequence $(x^n)_{n\in\mathbb{N}}$ be generated by Method 7. If F has the Kurdyka–Łojasiewicz property at the cluster point $x^* := \lim_{j\to\infty} x^{n_j}$, then the sequence $(x^n)_{n\in\mathbb{N}}$ converges to $x^* \in X$ as $n \to \infty$ and x^* is a critical point of F. Furthermore, the sequences $(x^n)_{n\in\mathbb{N}}$ has finite length*

$$\sum_{n=0}^{\infty} \|x^n - x^{n+1}\| < \infty\,.$$

Proof. The results of Proposition 7.4 and Proposition 7.6 are exactly the requirements of Theorem 5.1. Applying this result proves the theorem. □

7.5 Prototypes for computer vision applications

Many computer vision examples involve a linear operator in order to enforce spatial regularity of the solution. For example, this can be achieved using the gradient operator. We consider the prototype of inverse problems in computer vision

$$\min_{u\in X} \|Au - g\|_q^q + F_2(\tilde{G}(Ku))\,, \tag{7.15}$$

where $q \in \{1, 2\}$ and $K\colon X \to X$ may be any continuous linear operator (for example, gradient operator). Since in computer vision mostly the optimization variable, which often is an image, is denoted by u, we adapt this notation from now on. In the original formulation (7.2), it is $G = \tilde{G} \circ K$. Here, we further assume $\tilde{G}(0) = 0$, and $\tilde{G}(u)_i \geq 0$. A common choice for K is the gradient operator $\nabla = (\partial_x^\top, \partial_y^\top)^\top$ (∂_x is a matrix implementing forward differences in x-direction; analogue for ∂_y) and for $\tilde{G}$ the length of a vector $\tilde{G}((\partial_x u)_i, (\partial_y u)_i) = \sqrt{(\partial_x u)_i^2 + (\partial_y u)_i^2}$. In the first term of (7.15), called the *data-term*, we denote by $A\colon X \to X_1$ a continuous linear operator and by X_1 a finite dimensional real vector space. This

linear operator maps into a space, where measurements $g \in X_1$ are taken. The second term is denoted the *regularization-term*. Nonconvex regularization functions F_2 suitable for computer vision applications were already shown in Figure 1.3. The prototypes for the function $F_2 \circ G$ are

$$u \mapsto \frac{1}{p}\|\tilde{G}(Ku)\|_{p,\varepsilon}^p := \frac{1}{p}\sum_i (\tilde{G}_i(Ku)+\varepsilon)^p, \quad \varepsilon > 0, p \in (0,1] \tag{7.16}$$

$$u \mapsto \frac{1}{\mu}\log(1+\mu\tilde{G}(Ku)) := \frac{1}{\mu}\sum_i \log(1+\mu\tilde{G}_i(Ku)), \quad \mu > 0 \tag{7.17}$$

$$u \mapsto \frac{1}{2\mu}\log(1+\mu\tilde{G}(Ku)^2) := \frac{1}{2\mu}\sum_i \log(1+\mu(\tilde{G}_i(Ku))^2), \quad \mu > 0\,. \tag{7.18}$$

The first and the second functions F_2 are concave and nondecreasing on $G(X)$ but nondifferentiable, and the third is nondecreasing and differentiable. These functions clearly fulfill differentiability and Lipschitz continuity conditions required for our convergence analysis to hold. We now show that KL-property also holds:

Proposition 7.8. *Let F_2 be one of the prototypes* (7.16)*,* (7.17)*, or* (7.18)*, and let $\tilde{G}$ be semi-algebraic. Then, the function $F(u) = \|Au-g\|_q^q + F_2(\tilde{G}(Ku))$ is a KL-function.*

Proof. As $\tilde{G}$, K, and $\|Au-g\|_q^q$ are semi-algebraic (simple compositions of semi-algebraic functions), it is enough to verify that F_2 is definable in an o-minimal structure. However, thanks to the $\log$–$\exp$ structure [Wil96, dD98] (see Section 4.5), this fact is also clear for Prototypes (7.18), (7.17), and (7.16) (note that $(u+\varepsilon)^p = \exp(p\log(u+\varepsilon))$, $u \geq 0$). Then, Theorem 4.35 implies that F has the KL-property at any stationary point. □

7.5.1 Total generalized variation regularization

Opposed to TV-regularization which is used very frequently and can be seen as basic knowledge, total generalized variation (TGV) regularization was introduced only recently [BKP10]. The following introduction to TGV will be given in the continuous setting. For details we refer to [BKP10].

TGV generalizes TV based on the dual formulation incorporating the space of k-tensors

$$\mathcal{T}^k(\mathbb{R}^d) := \{\xi\colon \mathbb{R}^d \times \ldots \times \mathbb{R}^d \to \mathbb{R} : \xi \text{ is } k\text{-linear}\}$$
$$\mathrm{Sym}^k(\mathbb{R}^d) := \{\xi\colon \mathbb{R}^d \times \ldots \times \mathbb{R}^d \to \mathbb{R} : \xi \text{ is } k\text{-linear and symmetric}\}\,.$$

Let $\Omega \subset \mathbb{R}^2$ be the image domain and $u\colon \Omega \to \mathbb{R}$ be a function, then the TGV semi-norm of order $k \geq 1$ with smoothness parameter $\alpha = (\alpha_0, \ldots, \alpha_{k-1})$ is defined by

$$TGV_k^\alpha(u) := \sup\Big\{ \int_\Omega u \operatorname{div}^k \varphi \, dx \Big| \varphi \in \mathcal{C}_c^k(\Omega, \mathrm{Sym}^k(\mathbb{R}^2)), \|\operatorname{div}^l \varphi\|_\infty \leq \alpha_l, l = 0, \ldots, k-1 \Big\}\,,$$

where $\mathcal{C}_c^k(\Omega, \mathrm{Sym}^k(\mathbb{R}^2))$ denotes the space of continuously differentiable symmetric k-tensor fields with compact support in Ω, div^k the generalization of the divergence operator to these tensor fields. For $k=1$ the definition of TGV reduces to the dual formulation of the TV semi-norm.

Usually a primal formulation yields more intuition about a new concept. As in this chapter, we are only interested in $TGV_2^\alpha(u)$ we specify the order $k=2$ in the following. Applying the Legendre-Fenchel transform yields

$$TGV_2^\alpha(u) = \inf_{u_1 \in C^1(\overline{\Omega}, \mathrm{Sym}(\mathbb{R}^2))} \alpha_1\|\nabla u - u_1\|_1 + \alpha_0\|\mathcal{E}(u_1)\|_1\,,$$

where $\mathcal{E}$ denotes the symmetrized gradient operator $\mathcal{E}(u_1) = (\nabla u_1 + \nabla u_1^\top)/2$, which is a 2×2-matrix. There is also an asymmetric version of TGV defined in the primal formulation as

$$asymTGV_2^\alpha(u) = \inf_{u_1 \in C^1(\overline{\Omega}, \mathcal{T}(\mathbb{R}^2))} \alpha_1 \|\nabla u - u_1\|_1 + \alpha_0 \|\nabla u_1\|_1 \,.$$

Note, that the primal formulation of semi-norm TGV_2^α itself is stated as a minimization problem. However, when optimizing a function with TGV_2^α as a regularizer, we consider the single minimization problem in the variables u and u_1.

The main property of TGV_2^α is the ability to reconstruct piecewise affine functions without penalty. This makes TGV_2^α favorable compared to TV, which can only reconstruct piecewise constant functions. Considering the primal formulation, the intuition about the behavior of TGV_2^α can be explained as follows. Note that u_1 may be constant without increasing the norm. Then, u is allowed to be linear because ∇u may be constant (the constant of u_1) without increasing the TGV semi-norm.

7.6 Experimental analysis

7.6.1 Implementation details

The convex subproblems arising for the nonconvex optimization problems that are considered in the following can be solved efficiently, see Section 2.2. If not stated differently, we use the respective optimal algorithm from [CP11] (see also Section 2.2.6). It has proved optimal convergence rate: $O(1/e^n)$ when F_1 and F_2^* (convex conjugate of F_2) are uniformly convex, or when F_1 is uniformly convex and F_2 has Lipschitz continuous gradient, $O(1/n^2)$ when F_1 or F_2 is uniformly convex and $O(1/n)$ for the general case.

Here, we focus on the (outer) nonconvex problem. Let $(u^{n,l})$ be the sequence generated by Method 8, where the index l refers to the inner iterations for solving the convex problem, and n to the outer iterations. Proposition 7.1, which proves $(F(u^{n,0}))$ to be monotonically decreasing, provides a natural stopping criterion for the inner and outer problem. We verify every 10th inner iteration and stop as soon as

$$F(u^{n,l}) < F(u^{n,0}) \quad \text{or} \quad l > m_i, \tag{7.19}$$

where m_i is the maximal number of inner iterations. For a fixed n, let l_n be the number of iterations required to satisfy the inner stopping criterion (7.19). Then, outer iterations are stopped when

$$\frac{F(u^{n,0}) - F(u^{n+1,0})}{F(u^{0,0})} < \tau \quad \text{or} \quad \sum_{i=0}^{n} l_i > m_o, \tag{7.20}$$

where τ is a threshold defining the desired accuracy and m_o the maximal number of iterations. The difference in (7.20) is normalized by the initial function value to be invariant to a scaling of the energy.

In order to obtain a guarantee for a converging sequence of function values checking the decent property is required. However, throughout the experiments we observed that a fixed number of 10 iterations is a good choice and we can omit computing the energy.

Remark 7.9. As long as we can guarantee that the energy decreases we can expect a converging sequence of function values. However, if the subproblem is not solved exactly, the convergence properties from Theorem 7.7 are partially lost. The convergence theorem allows for inexact descent methods, i.e., it allows for some errors in the evaluation of the subproblem. However the granted quantity of the error is not addressed in the theorem. Therefore, we focus on the convergence of the energy values.

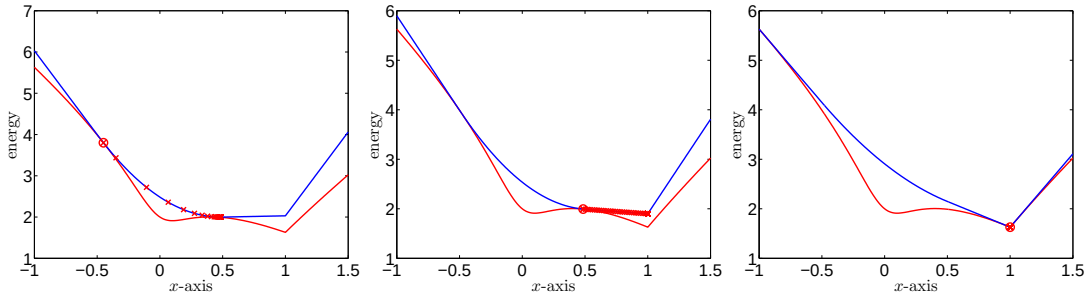


Figure 7.2: Three (outer) iteration steps of the proposed algorithm to minimize the red nonconvex function. FROM LEFT TO RIGHT: Outer iteration 1, 2, and 3. In each plot a few steps of the solver for the (inner) convex problem (minimization of the blue function) are visualized. The red circle shows the point, in which the original function is approximated. Though there is a local minimum close to the starting point, the algorithm jumps over it and finds the global optimum.

7.6.2 Competing method

We compare our algorithm against the iPiano algorithm, which we proposed in Chapter 6. We compare against iPiano, because it has proved to be very efficient and it is applicable to similar problems as considered in this chapter, namely to problems that can be decomposed as a sum of a (simple) convex function and a function with Lipschitz continuous gradient. Moreover, iPiano finds special cases in the NIPS algorithm [Sra12] when the inertial term is turned off, in the Heavy-ball method for differentiable nonconvex problems [ZK93], or in the well-known gradient projection algorithm. Therefore, actually our comparison is against several algorithms.

Assuming that our $F_2 \circ G$ has Lipschitz continuous gradient with constant $L > 0$, the update scheme of iPiano using the notation here can be written as

$$u^{n+1} = (I + \alpha \partial F_1)^{-1}(u^n - \alpha \nabla (F_2 \circ G)(u^n) + \beta(u^n - u^{n-1}))\,. \tag{7.21}$$

Due to the smoothness assumption to $F_2 \circ G$ in this algorithm, when comparing to the proposed IRL1-algorithm the nondifferentiable points must be smoothed.

7.6.3 Analysis of local minima

In this part, we experimentally study the sensitivity of our algorithm with respect to local stationary points.

7.6.3.1 A one dimensional example

Here, we show that the proposed algorithm has the ability to avoid local minima. We consider the model problem (see red function in Figure 7.2)

$$\min_{u \in \mathbb{R}} \lambda |u - f| + \frac{1}{2} \log(1 + \mu |u|^2)\,,$$

where $f = 1$, $\mu = 25$, $\lambda = 2$. As it is $F_2(|u|) = \frac{1}{2}\log(1 + \mu|u|^2)$, we use the iteratively reweighted Huber Algorithm 11 ($\varepsilon = 1$) to find a minimum of this function. Figure 7.2 shows three outer iterations of IRHuber initialized at $u^0 = -0.45$. Depending on the initialization different local optima are reached. Initializing $u^0 = \pm 0.4$ the local maximum at $u = 0.4$ is found, for $u^0 \in (-0.4, 0.4)$ the left local

$\mu = 1$	$\lambda = 0.10$	$\lambda = 0.50$	$\lambda = 1.00$	$\lambda = 2.00$	$\lambda = 5.00$
min. energy	92.39	369.69	592.51	683.67	683.67
PrimalDual-IRHuber	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
iPiano, $\beta = 0.7$	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
$\mu = 50$	$\lambda = 0.10$	$\lambda = 0.50$	$\lambda = 1.00$	$\lambda = 2.00$	$\lambda = 5.00$
min. energy	124.80	559.76	1060.89	1984.31	5097.96
PrimalDual-IRHuber	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
iPiano, $\beta = 0.7$	**1.0000**	**1.0000**	**1.0000**	**1.0000**	1.0006
$\mu = 100$	$\lambda = 0.10$	$\lambda = 0.50$	$\lambda = 1.00$	$\lambda = 2.00$	$\lambda = 5.00$
min. energy	130.96	586.67	1118.07	2101.04	5945.98
PrimalDual-IRHuber	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
iPiano, $\beta = 0.7$	**1.0000**	**1.0000**	**1.0000**	1.0002	1.0050
$\mu = 250$	$\lambda = 0.10$	$\lambda = 0.50$	$\lambda = 1.00$	$\lambda = 2.00$	$\lambda = 5.00$
min. energy	140.87	623.65	1189.83	2255.34	6986.61
PrimalDual-IRHuber	**1.0000**	1.0007	1.0007	**1.0000**	**1.0000**
iPiano, $\beta = 0.7$	**1.0000**	**1.0000**	**1.0000**	1.0014	1.0145

Table 7.1: Comparison of the final energy for our IRHuber algorithm compared to iPiano for the problem (7.22) and different parameter settings with maximal 50000 iterations. *min. energy* is the minimal final energy value among the four methods. The other values describe the multiplication factor to this minimal energy. In most experiments, IRHuber finds the lowest energy.

optimum is the solution, and initializing with $u^0 \in (-\infty, -0.4) \cup (0.4, \infty)$ the global optimum is found. Although the algorithm can also converge to a local maximum, this is rarely the case. When we initialize at $u^0 \in (-0.4, 0.4)$ the algorithm is already trapped to the left local minimum. However, different to many other method it does not necessarily converge to the nearest local minimum.

7.6.3.2 A high dimensional example

In image processing, optimization problems usually have a very high dimensionality and it is not possible to visualize the objective functions. Conclusions about whether the algorithm is attracted by local minima or whether it is "robust" against local minima can only be drawn indirectly. If the energy value (function value) corresponding to one algorithm is lower than with another, we conclude that the first algorithm has found a better local minimum. This will be shown in the following example. In this experiment, we consider the problem

$$\min_{u \in \mathbb{R}^{6305}} \lambda \|u - f\|_1 + \frac{1}{2} \sum_i \log(1 + \mu |(\nabla u)_i|^2) \tag{7.22}$$

and solve it using iPiano and our IRHuber ($\varepsilon = 1/\sqrt{\mu}$) method. For all methods we fix a maximum of 50000 iterations and use the break condition (7.20) with $\tau = 10^{-12}$. Table 7.1 confirms that our algorithm usually finds the lowest energy. The difference is more significant, the higher the values of λ and μ. For larger μ the "nonconvexity" is stronger. For small λ the optimal result is constant, i.e. $|(\nabla u)_i|$ is small everywhere and lies in the convexity region of $\log(1 + \mu y^2)$. Thus the nonconvexity of the second term is of little importance.

7.6.3.3 Robustness with respect to the initialization

We fix $\lambda = 1$ and $\mu = 1$ and solve the optimization problem

$$\min_{u \in \mathbb{R}^N} \lambda \|u - f\|_1 + \sum_i \log(1 + \mu |(\nabla u)_i|) \tag{7.23}$$

initialization	noisy image	zero	random	random with square of zero-valued pixel
initial energy	45308.479	70803.569	116368.474	114674.008
final energy	23583.466	23575.354	23576.401	23576.01

Table 7.2: Initial and final energy values for the optimization problem (7.23) with $\lambda = 1$ and $\mu = 1$. Numerically, the result values slightly differ from each other. On average the difference is very small. Visually, different initializations yield very similar results. This suggests that our algorithm is robust towards the initialization in this experiment.

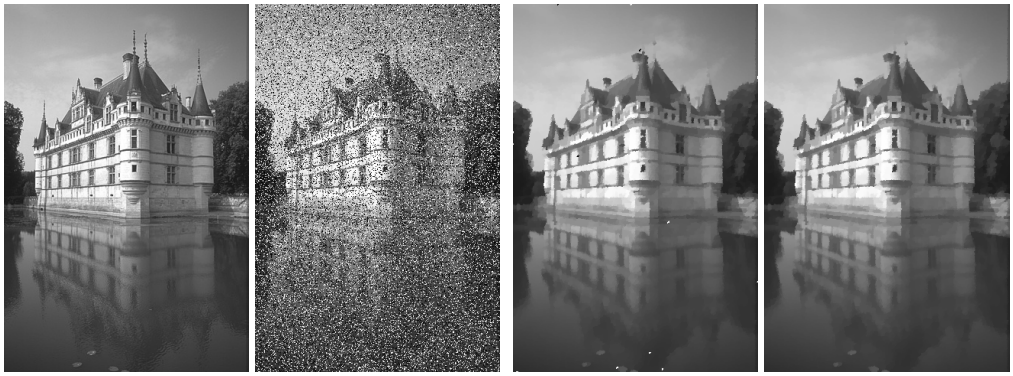


Figure 7.3: Visualization of the experiment in (7.23). LEFT PAIR: Clean and noisy image. RIGHT PAIR: Results for two different initializations, the zero image and the noisy image. As the corresponding energy values from Table 7.2 suggest, the result images are visually close to each other.

starting from different initializations u^0 using the iteratively reweighted ℓ_1 algorithm (Algorithm 9). Here $N = 154401$. The noisy input image and the ground truth are shown on the left in Figure 7.3. The energy values of the initialization and the final values are shown in Table 7.2. The energy values of the solutions slightly differ.

The maximal difference of energy values is between initializing with the noisy image and initializing with the zero image. The energy difference is approximately $d \approx 8.11$. Let us consider what it means per pixel on average and in the worst case. If we assume that this error is only caused by the data term, we can conclude

$$d = \sum_{i=1}^{N} |u_i - f_i| \geq N \min_i |u_i - f_i| \quad \Rightarrow \quad \min_i |u_i - f_i| \leq d/N \approx 5.25 \cdot 10^{-5}\,.$$

This means that the average (minimal) error per pixel is bounded by approximately $5.25 \cdot 10^{-5}$. Pixels that cause an error that is higher than the minimal one reduce the upper bound for the error for all other pixels. Considering the worst case only 8 pixel can have the maximal error of 1, which is the range of the gray values. On the other hand, if we assume the error is solely by the regularization term, it holds

$$d = \sum_{i=1}^{N} \log(1 + |y_i|) \geq N \log(1 + \min_i |y_i|) \quad \Rightarrow \quad \min_i |y_i| \leq \exp(d/N) - 1 \approx 5.25 \cdot 10^{-5}\,.$$

Therefore, the energy difference could also be caused by an average (minimal) error for the gradient of maximal $5.25 \cdot 10^{-5}$. The worst case analysis shows that only $8.11/\log(2) \approx 11$ pixel can have an error in the gradient of 1. These numbers suggest a small difference between the two solutions. The right pair

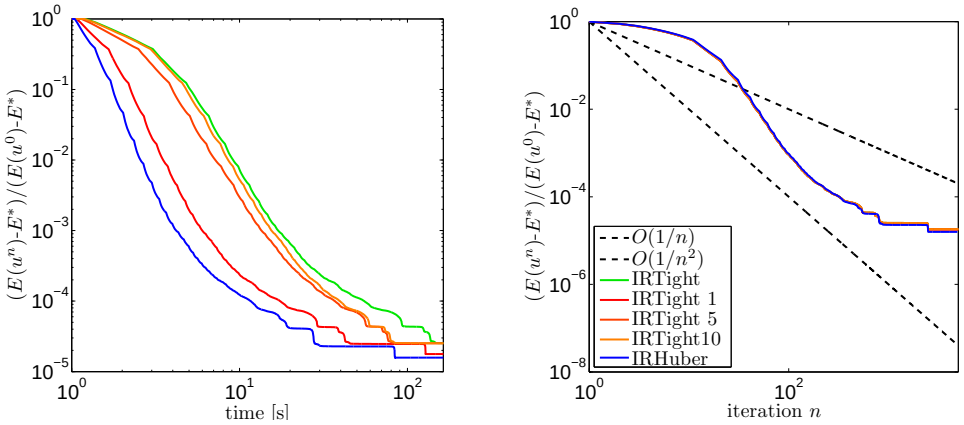


Figure 7.4: Comparison of the energy decrease for problem (7.24) between Algorithm 10 (IRTight) and Algorithm 11 (IRHuber). The legend is the same for both plots. The proximity operator arising in the convex optimization algorithm for IRTight is solved analytically or using 1, 5 or 10 Newton iterations. The proximal operator for IRHuber can be solved analytically. Therefore, in terms of runtime IRHuber is faster than IRTight, whereas in terms of iterations all methods perform equally well.

of image in Figure 7.3 visualizes this difference. In this experiment, the final results are very similar. The error seems to be better reflected by the worst case analysis as it is concentrated on a few outliers.

Unfortunately, it is hard to generalize this observation. For input images with more noise or for different parameter settings the results can differ more depending on the initialization. The main problem is that the high dimensionality makes it hard to get an intuition about local minima and maxima. In the following experiments, we always initialize with the noisy image.

7.6.4 Numerical comparison

The existence of local minima and the missing information about the global optimum for nonconvex optimization problems complicates the evaluation. For all the following experiments, we agree on the following evaluation: We use the method that achieves the lowest energy value and run it for 10^6 iterations; we use the solution u^{10^6} to define $E^* := E(u^{10^6})$. Then, we use the relative distance to this "optimal" energy value and analyze the convergence of the sequence

$$\left(\frac{E(u^n) - E^*}{E(u^0) - E^*}\right)_{n\in\mathbb{N}}.$$

Note that, if the sequence does not convergence to E^*, then the sequence can still converge to another local optimum. As we choose E^* such that it is minimal among the methods under consideration, a method that does not converge to E^* only finds a higher energy.

Iteratively reweighted Huber vs. iteratively reweighted tight convex. First, we compare IRTight (Algorithm 10) vs. IRHuber (Algorithm 11 with $\varepsilon = 1/\sqrt{\mu}$ as suggested at the end of Section 7.4.3) for the optimization problem

$$\min_{u\in\mathbb{R}^{154401}} \frac{\lambda}{2}\|u - f\|_2^2 + \frac{1}{2\mu}\sum_i \log(1 + \mu|(\nabla u)_i|^2)\,, \tag{7.24}$$

where $\lambda = 0.1$, $\mu = 800$, and $f \in \mathbb{R}^{154401}$ is the given noisy input image from Figure 7.5.

Figure 7.5: Visualization of the experiment (7.24). FROM LEFT TO RIGHT: Clean image, noisy image deteriorated by Gaussian noise, result of (7.24) with $\lambda = 0.1$, $\mu = 800$, and the result of (7.24) with $\lambda = 0.3$, $\mu = 250$.

The convergence plot of the energy is shown in Figure 7.4. IRHuber converges faster in terms of the actual computation time than IRTight. This result is explained by the simple structure of the IRHuber surrogate function. The proximity operator that arises in the convex surrogate problem for the IRHuber can be solved analytically. For IRTight solving the proximity operator requires to find the zero of a cubic polynomial in a certain interval. For IRTight $\{1,5,10\}$ this proximity operator is solved using Newton's method with a maximum of 1, 5 or 10 iterations and break condition for the maximal absolute difference of two successive iterates of 10^{-4}. Closer to the optimal value the number of iterations required by Newton's method decreases due to the initialization. The analytic solution for the proximity operator needs always the same time. In terms of iterations, all methods perform equally well.

Thanks to the simple structure of IRHuber, it is more efficient for regularization problems involving terms $\log(1 + y^2)$. Therefore, in the following experimental comparison, we consider IRHuber only. However, we should keep in mind, that if the proximity operator in IRTight is easy to solve, it is a better approximation and converges faster.

Iteratively reweighted Huber and reweighted least squares. We evaluate our algorithm in terms of speed compared to iPiano (and its special case NIPS with $\beta = 0$). We consider the problem

$$\min_{u \in \mathbb{R}^{154401}} \frac{\lambda}{2} \|u - f\|_2^2 + \frac{1}{2\mu} \sum_i \log(1 + \mu |(\nabla u)_i|^2)\,, \tag{7.25}$$

where $\lambda = 0.3$, $\mu = 250$, and $f \in \mathbb{R}^{154401}$ is the given noisy input image from Figure 7.5 (see the same figure for the result image).

The Lipschitz constant required by iPiano is set to $L = 8$. Then, $\alpha = 2(1 - \beta)/L$ is set according to the step size rules in Algorithm 5. Using Algorithm 11 and Algorithm 12, the (convex) surrogate function is strongly convex and can be solved with linear convergence rate, which is optimal for this class of problems. Algorithm *iPiasco-IRHuber* solves the primal problem using iPiasco (see Chapter 3) and *D-iPiasco-IRHuber* solves the surrogate problem in the dual formulation. Analogously, *iPiasco-IRLS* solves the primal problem arising in Algorithm 12 and *D-iPiasco-IRLS* the dual problem. *CG-IRLS* solves the primal inner problem using conjugate gradient. In this experiment we do not show the result of solving the inner problem with the optimal primal dual algorithm [CP11, Algorithm 3], because it performed worse and the constants in the estimate for the linear convergence rate are suboptimal.

Figure 7.6 analyzes the differences in convergence depending on the number of inner iterations and whether the inner problem is formulated as the primal or the dual problem. In general, we found 10 inner iterations to be a good choice for the iteratively reweighted algorithms, though the optimal choice in this particular example is 5 iterations. For IRLS, the best performance is achieved by solving the *primal* inner problem, whereas for IRHuber is is advantageous to solve the *dual* problem.

Figure 7.7 shows the comparison of the energy decrease between IRHuber and IRLS. In terms of actual computation time IRLS performs better. This is due to the split definition of the Huber function,

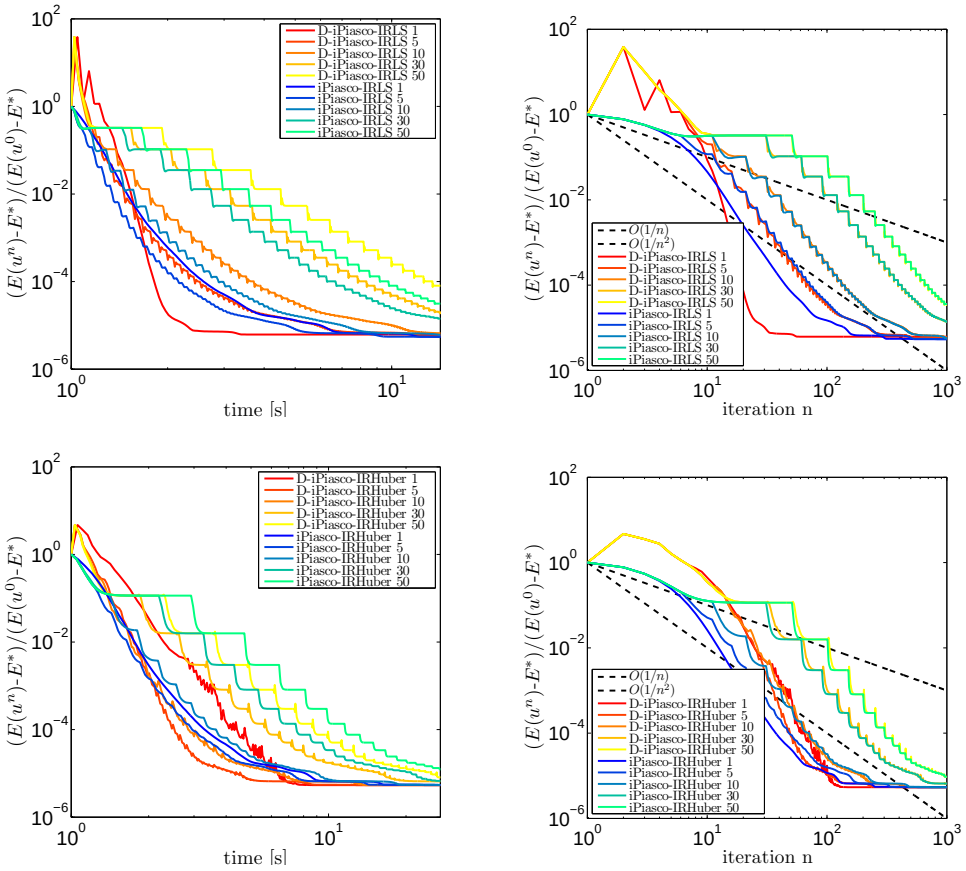


Figure 7.6: Convergence depending on the solver used for the inner problem (primal or dual) and the number of inner iterations (number in the figure's legend). In general, 5-10 iterations is a good choice. In terms of actual computation time, for IRLS using the primal of the inner problem yields the fastest convergence; for IRHuber it is the dual of the inner problem. Both convex surrogate functions are solved with algorithms that have an (optimal) linear convergence rate.

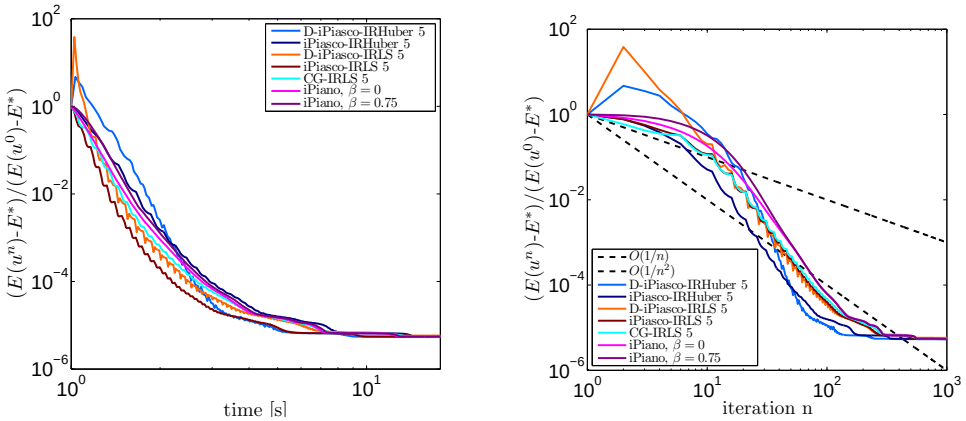


Figure 7.7: Comparison between IRLS, IRHuber, and iPiano. The iteratively reweighted algorithms perform best in this experiment. Regarding actual computation time IRLS is the fastest, whereas regarding iterations IRHuber shows the best convergence rate.

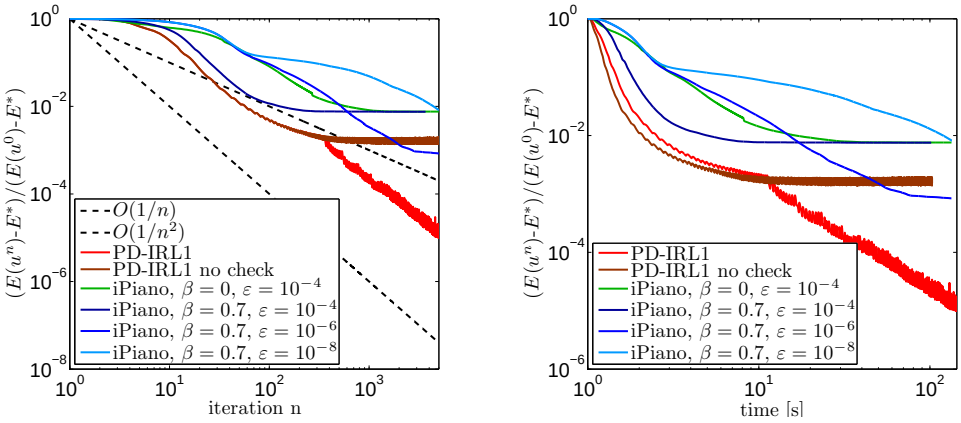


Figure 7.8: Comparison of the energy decrease for problem (7.26) between IRL1 and different parametrizations of iPiano. As our algorithm can optimize the nonregularized energy in (7.26) it achieves a lower energy. The version *PD-IRL1 no check* does not check the energy decrease but updates the weights every 10th iteration. It is the fastest in this experiment up to an accuracy of about 10^{-3}, then *PD-IRL1* takes over.

Figure 7.9: Visualization of the experiment (7.26). LEFT: Clean image. MIDDLE: Noisy image. RIGHT: Denoised image.

which additionally requires to distinguish two cases (norm less or greater than ε). As this extra computation cost does not matter in terms of the number of iterations, IRHuber converges the quickest in that case. Regarding both, number of iterations and computation time, the iteratively reweighted algorithms perform better than iPiano.

Iteratively reweighted ℓ_1 on TV-term. As mentioned before, our iteratively reweighted ℓ_1 algorithm is not well suited for problems that have a quadratic behavior around 0. The cases where the IRL1 algorithm becomes interesting is beyond the applicability of iPiano, namely for instance when $F_2(|u|) = \log(1 + |u|)$. iPiano can only be applied to a smoothed version of F_2. However, then, a different energy is minimized. We evaluate the IRL1 algorithm on the following objective:

$$\min_{u \in \mathbb{R}^{154401}} \|u - f\|_1 + \sum_i \log(1 + |(\nabla u)_i|) \tag{7.26}$$

and use $\log(1 + |(\nabla u)_i|_\varepsilon)$ for iPiano. The input f and the visual result are shown in Figure 7.9. The numeric comparison against iPiano with backtracking (Algorithm 5) is shown in Figure 7.8. On one hand, reducing the ε in the regularization of $|(\nabla u)_i|_\varepsilon$ better approximates the original problem, but on the other

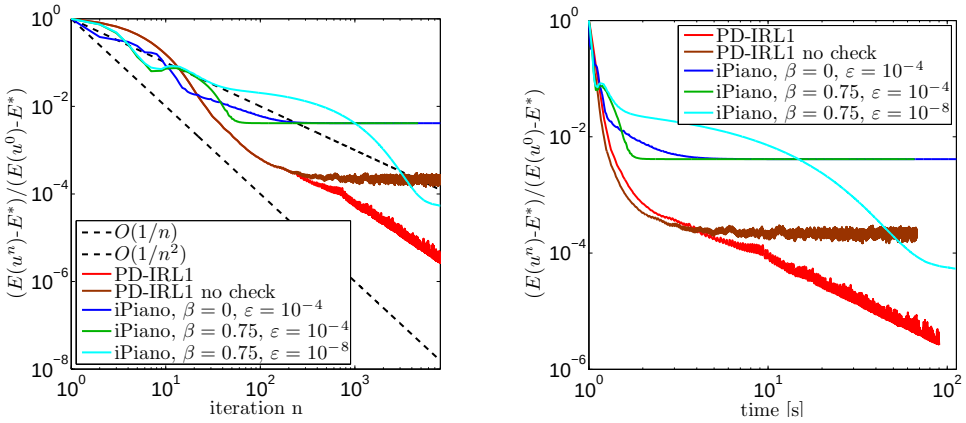


Figure 7.10: Comparison of the energy decrease for problem (7.27) between our method and different parametrizations of iPiano. As our algorithm can optimize the nonregularized energy it achieves a lower energy. Furthermore, as we solve a sequence of convex problems, we can benefit from efficient convex programming algorithm like [CP11]. The version *PD-IRL1 no check* does not check the energy decrease but updates the weights every 10th iteration. It is the fastest in this experiment.

hand, the problem is more difficult to solve for iPiano and needs many more iterations. The Lipschitz constant for iPiano depends on ε and therefore directly influences the feasible step-sizes. The method *PD-IRL1 no check* finds a worse local optimum than *PD-IRL1* and *iPiano* ($\beta = 0.7$, $\varepsilon = 10^{-8}$) with a difference of about 10^{-3} to the "optimal" one E^*. It is faster in terms of actual computation time than *PD-IRL1* , because it does not have to compute the energy. *PD-IRL1* achieves a better local optimum by doing more iterations if required. IRL1 performs better than iPiano in terms of speed and, as it optimizes the original energy achieves a higher accuracy.

Iteratively reweighted ℓ_1 on TGV-term. As a last numerical experiment we consider the total generalized variation based model

$$\min_{u\in\mathbb{R}^N, v\in\mathbb{R}^{2N}} \frac{\lambda}{2}\|u-f\|_2^2 + \left(\frac{\alpha_1}{\mu}\sum_i \log(1+\mu|(\nabla u - v)_i|) + \frac{\alpha_0}{\mu}\sum_i \log(1+\mu|(\nabla v)_i|)\right). \quad (7.27)$$

The experiment is performed on an image whose 3D-mesh is shown on the top left in Figure 7.11. Its dimension is $N = 16384$. We compare our IRL1 algorithm against the iPiano algorithm (Algorithm 5) on an ε-regularized energy for the nonconvex TGV model in terms of convergence. The model parameters are set to $\mu = 8$, $\lambda = 4$, $\alpha_1 = 0.5$, and $\alpha_0 = 1$. In Figure 7.10 the energy decrease for different methods is plotted. From the optimization viewpoint, it is well-known [BKP10, PZB11] that the TGV-regularization model is a hard problem even in the convex case. In [PZB11], the problem is efficiently solved using the primal dual algorithm [CP11], which is also used here for the convex surrogate problems. As for our algorithm a sequence of convex problems arises, we can benefit from efficient convex programming algorithms. Therefore, we observe a faster convergence for the IRL1 algorithm compared to iPiano. The difference becomes more and more significant the smaller the ε is chosen for making the TGV differentiable in 0.

7.6.5 Total generalized variation experiment

As the TGV-regularizer is developed only recently and first used in the nonconvex setting here, we perform another experiment with the energy model (7.27) where the focus is on accuracy. The right column

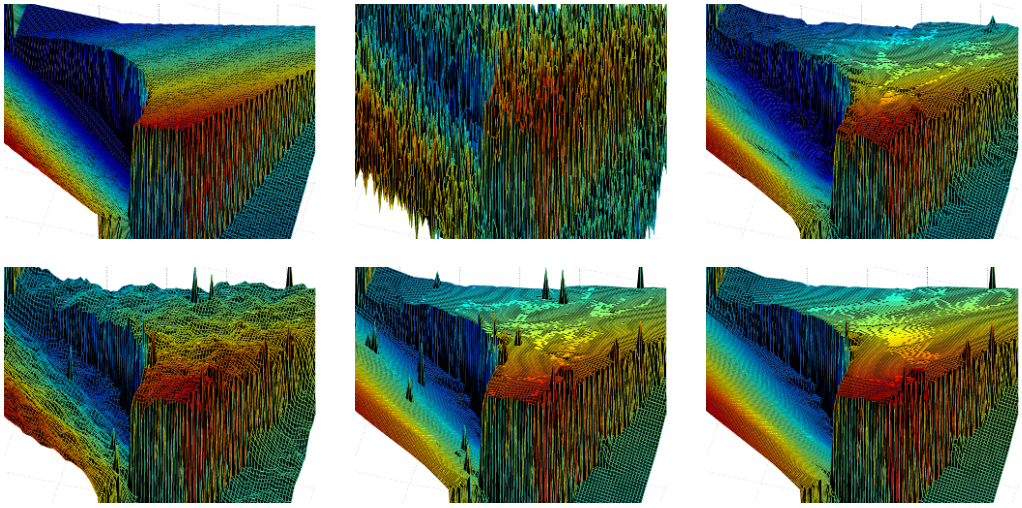


Figure 7.11: Comparison of TGV regularization using the model (7.27) using a surface representation. TOP LEFT: Clean surface. TOP MIDDLE: Noisy surface. TOP RIGHT: Solution using the convex TGV (PSNR: 35.825). BOTTOM ROW: Solutions with nonconvex log-TGV as in (7.27). BOTTOM LEFT: Solution with iPiano using $\beta = 0.75$, $\varepsilon = 10^{-4}$ (PSNR: 33.515). BOTTOM MIDDLE: Solution with Piano using $\beta = 0.75$, $\varepsilon = 10^{-8}$ (PSNR: 35.679). BOTTOM RIGHT: Solution using IRL1 (PSNR: 36.672). The comparison between the surfaces on the top right and on the bottom right shows that nonconvex penalizers are the better choice, and the comparison between the surfaces of the bottom row shows the importance of solving the nonregularized energy model. As the result with IRL1 has fewer spikes and the energy is smaller, it found a better local optimum.

of Figure 7.11 compares the convex TGV with the nonconvex TGV. For each of them, the parameters are optimized with respect to the PSNR value, which is 35.835 for the convex model and 36.672 for the nonconvex model. Parameters for the convex model are $\lambda = 1$, $\alpha_1 = 0.1$, and $\alpha_0 = 1$ and for the nonconvex model $\mu = 8$, $\lambda = 4$, $\alpha_1 = 0.5$, and $\alpha_0 = 1$.

Nonconvex norms in the regularization are a good choice when (1) sharp discontinuities are desired or (2) the properties of the regularizer (here the ability to reconstruct piecewise affine functions) are to be enforced.

On the other hand, the comparison of the results in the second row of Figure 7.11 reveals the importance of solving a nonregularized energy model. The result on the bottom right is nicely piecewise smooth. The problem of several small outliers for the other two results of the second row does not arise in the IRL1 algorithm as the first inner subproblem is the convex TGV-model which yields already a smooth result. Then, in the next iterations discontinuities are enhanced again.

7.7 Optical flow estimation with nonconvex regularizers

In this section we show application examples of nonconvex energy models for the task of optical flow estimation. The same modeling principles can be transferred directly to many other vision tasks, as demonstrated in the conference paper [ODPB13], where examples on denoising, deconvolution, depth map fusion, and optical flow estimation are shown. Here we focus on a more detailed analysis of optical flow including nonconvex data terms and nonconvex regularizers and their effects.

Optical flow describes dense correspondences between a pair of images $I(x,t)$ and $I(x,t+1)$. Modeling of variational optical flow usually consists of a regularization term and a data term. The first one models the smoothness of the flow field. The data term measures the difference between the motion

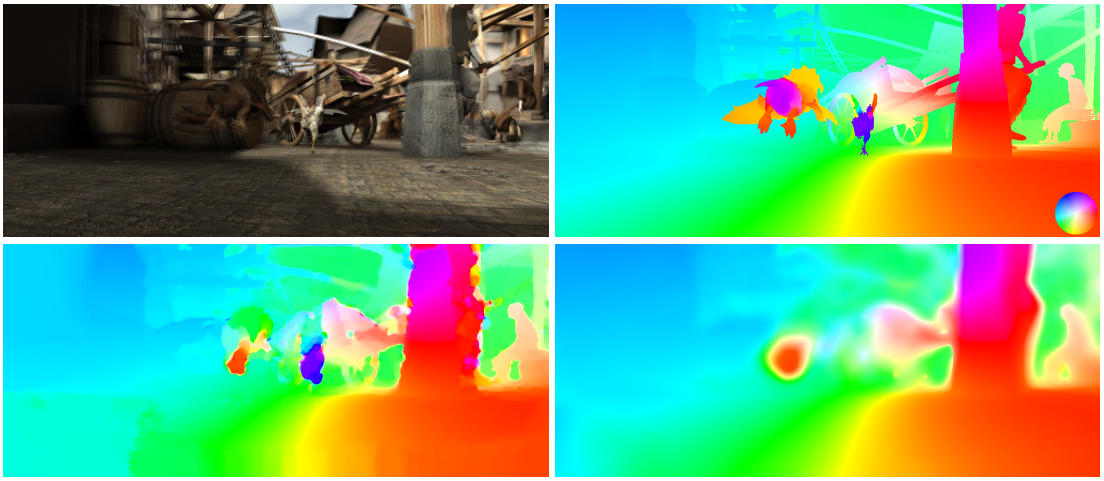


Figure 7.12: TOP LEFT: Image pair *market6_0005* to *market6_0006*. TOP RIGHT: Ground truth flow. BOTTOM LEFT: Result with LDOF [BM11] (EP: 13.18. BOTTOM RIGHT: An over-smoothed flow field (EP: 6.59. Considering only the quantitative result shows that the method from (d) is better than (c). However, the over-smoothed result in (d) is practically useless compared to (c).

compensated second frame and the image of the first frame. A simple example is the difference of gray values (brightness constancy assumption) $\|I(x + u(x), t + 1) - I(x, t)\|_2^2$, where $u = (u^1, u^2)^\top$ is the sought optical flow field. Since the unknown flow field is a variable of the generally nonconvex image, the data term is nonconvex independent of the properties of the penalty function. In practice, this kind of nonconvexity is dealt with by a Gauss-Newton scheme in combination with a continuation method [BBPW04][1].

Previous works on variational optical flow estimation always employ a convex penalty function for the data term and the regularizer. The results of the following experiments shall indicate that one can benefit from rather using nonconvex penalties. Details, such as parameter optimization, the optimal usage in conjunction with a continuation method, etc., need further investigation to compete with heavily tuned methods on standard benchmark datasets. In the following, we present three ways to apply and use the nonconvex penalties with the algorithm proposed in this chapter: a nonconvex penalty on the brightness constancy assumption, a nonconvex regularizer, and a nonconvex penalty for integrating point correspondences into variational methods.

The experiments are performed on image pairs (clean version) from the Sintel benchmark [BWSB12][2]. The standard quality measure of the Sintel benchmark is the average endpoint error. It is well-known that average errors emphasize global properties of the flow fields while details, such as sharp discontinuities, are under-represented. Figure 7.12 shows an example. This fact is disadvantageous for nonconvex regularization penalties, which are particularly good for obtaining sharp discontinuities. For this reason, we present mainly qualitative results but also report the endpoint errors.

7.7.1 Nonconvex data term: robust optical flow

Outliers in the data term, mostly caused by occlusion, are a major issue in optical flow estimation. There are two aspects of this problem: detection of occluded points and interpolation of the flow field at these

[1] It cannot be approached well with our algorithm.

[2] The Sintel benchmark provides ground truth optical flow fields for a realistically rendered video. It provides three different stages of this rendering process: *albedo*, *clean*, and *final*.

Figure 7.13: TOP LEFT: Image pair *bandage1_0011* to *bandage1_0012*. TOP RIGHT: Ground truth flow. BOTTOM LEFT: Final data term weighting. BOTTOM RIGHT: Estimated flow field using (7.28). The weighting mask is part of the optimization using our algorithm due to using a robust, nonconvex norm. It is not introduced separately. The algorithm automatically weights down the brightness constancy assumption where it is violated (darker areas on the bottom left). Each color channel is weighted individually and the corresponding color represents the weighting.

points. Nonconvex penalty functions allow for a straightforward approach to implicitly deal particularly with the first aspect. The basic assumption for estimating the optical flow is the brightness constancy (or color constancy) assumption. As it is typically not satisfied in occlusion areas, the penalty on the brightness constancy should be reduced there. This is naturally achieved by nonquadratic penalties, which implicitly weigh down the influence of points that contradict the constancy assumption. With convex penalty functions, however, the effect is often not strong enough. With nonconvex penalty functions, the influence of outliers can be reduced much more. In the limit, this approaches the algorithmic two-step treatment, where outliers are first detected explicitly and then removed completely from the estimation process.

As in occlusion areas the measured data is invalid, the optical flow field in these areas must be inferred by prior knowledge like smoothness of the flow field. In a variational formulation, which seeks for a global agreement of all constraints, the reduction of the penalty on the brightness constancy assumption automatically assigns more importance to the regularization term in these areas. It is an open question what is the best regularizer for this job. We consider regularization with total variation, which is easy to use and preserves discontinuities. It could be easily replaced by some other, more complex prior.

We consider the following energy model:

$$\min_u \sum_i |(\nabla u)_i| + \frac{\lambda}{\mu} \sum_{i,k} \log(1 + \mu |(\rho^k(u))_i|) \,, \tag{7.28}$$

where $\rho^k(u) = I_t^k + (\nabla I^k)^\top (u - u_0)$ implements the linearized brightness constancy assumption for the color channel $k \in \{1, 2, 3\}$, and $\lambda = 15$, $\mu = 5$.

We minimize the energy with the iteratively reweighted ℓ_1 algorithm. The algorithm generates an automatic weighting for the data term. The weights are small where the brightness constancy assumption does not hold, i.e., for outliers. The weighting is directly inferred from the cost function. The approach in [ARS12] models occlusions explicitly, but in the end it comes down to a similar weighting. Figure 7.13 shows an example. As expected, the weighting on the bottom left in Figure 7.13 shows a reduction of the penalty particularly in occlusion areas.

Despite the quite good detection of the occlusion region due to the nonconvex penalty, the optical flow of the arms still leaks into the background. This is due to the weak smoothness prior, which does not take the direction of the occlusion into account. Future work on smoothness priors may exploit the detected occlusion regions more effectively.

7.7.2 Nonconvex TGV regularized optical flow

Total variation is the most popular regularizer for the optical flow field. However, it penalizes also flow fields that describe rotation and scaling motion. Total generalized variation deals with this problem, as affine motion can be described without penalty. Therefore, we consider the variational model given by

$$\min_{u,v} \lambda Q(I,u) + \alpha_1 F_{2,1}(|\nabla u - v|) + \alpha_0 F_{2,0}(|\nabla v|), \tag{7.29}$$

where $Q(I,u)$ is the quadratic fitting term from [WPB10] using normalized cross correlation, and $F_{2,1}$, $F_{2,0}$ are (possibly different) nonconvex penalty functions. The model described in (7.29) can be used to enforce the TGV-properties by using nonconvex norms. This yields highly desirable sharp motion discontinuities as can be seen in the bottom row of Figure 7.14. The penalty on $|\nabla v|$ can be seen as a penalty on kinks in the flow field. Reducing the cost of sharp kinks by a nonconvex penalty often leads to the fact that sharp discontinuities in the flow field are replaced by a linear transition with two sharp kinks. Therefore, we set $F_{2,0} = \mathrm{id}$.

7.7.3 Nonconvex integration of point correspondences

Current state-of-the-art methods [XJM12, BM11, WRHS13] usually incorporate a sparse or semi-dense feature matching into the optimization procedure. In LDOF [BM11], the deviation of the estimated flow field from these feature matches is penalized. The penalty is based on the ℓ_1-norm. Nonconvex norms are more robust and can deal with more erroneous feature matches in a certain local area than the ℓ_1-norm. In the following model, we propose to use a nonconvex penalty function for the deviation from the initial feature matches:

$$\min_{u,v} \lambda \|Q(I,u)\|_1 + \frac{\beta}{p}\|u - u_{\mathrm{FM}}\|_{p,\varepsilon} + \alpha_1 \|\nabla u - v\|_1 + \alpha_0 \|\nabla v\|_1, \tag{7.30}$$

where $Q(I,u)$ is the quadratic fitting term from [WPB10] using normalized cross correlation, u_{FM} are sparse feature correspondences estimated like in [BM11], and $p \leq 1$ determines the nonconvexity of the feature matching penalizer. Figure 7.15 compares the convex vs. the nonconvex penalty term. In this formulation, we evaluate a convex energy (*cFMcTGV*) with $p = 1, \varepsilon = 0$, $\lambda = 1$ and $\beta = 2$ versus a nonconvex energy (*ncFMcTGV*) with $p = 0.5, \varepsilon = 0.001$, $\lambda = 0.8$ and $\beta = 3$. Both settings use $\alpha_1 = 0.2$ and $\alpha_0 = 1$. All parameters were optimized for two challenging image pairs of the Sintel optical flow benchmark [BWSB12]. The two image pairs are chosen complementary in the way feature matches should be used. For *market6_0005*, many feature matches (bottom of the image) should be considered as outliers, whereas for *cave2_0015* the few correct feature matches on the dragon must be used to capture the large motion. Our results show that feature matching driven optical flow methods can benefit a lot from nonconvex penalty functions.

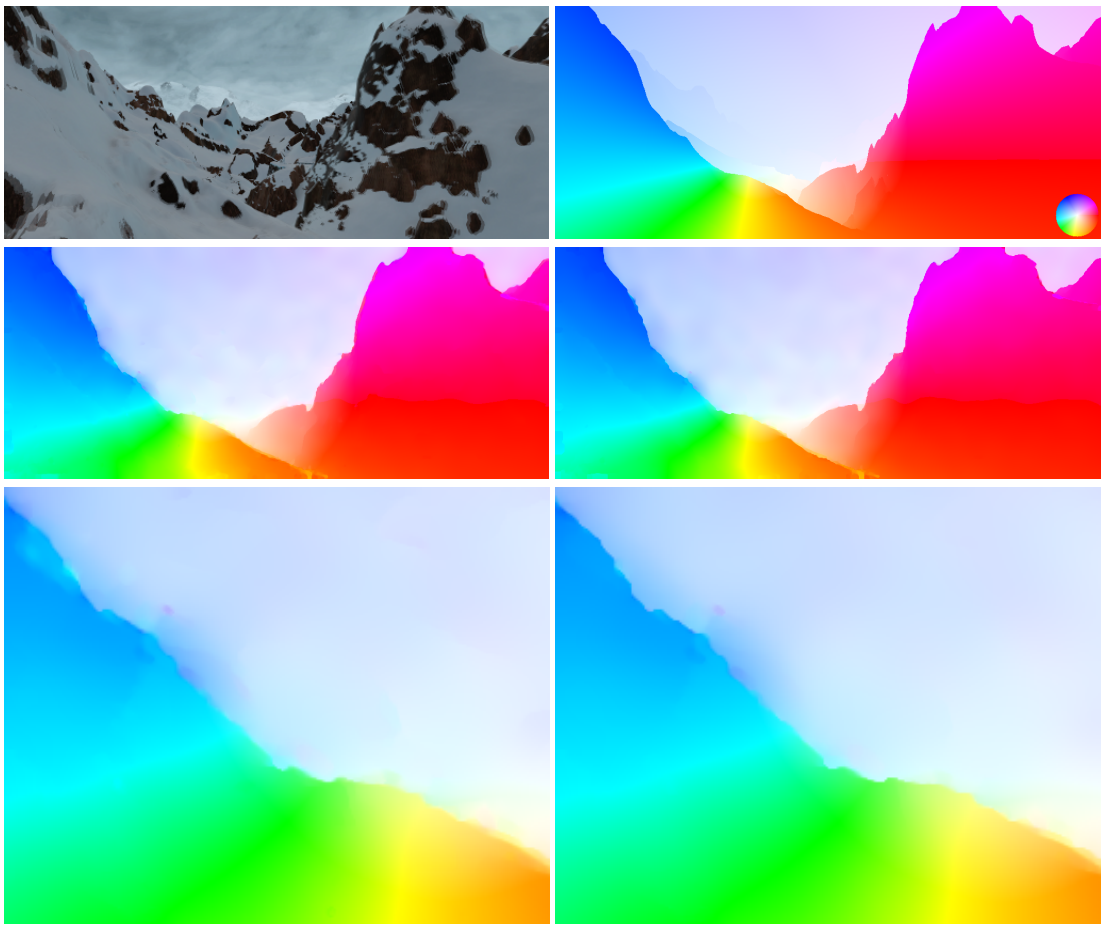


Figure 7.14: TOP LEFT: Image pair *mountain_0001* to *mountain_0002*. TOP RIGHT: Ground truth flow. MIDDLE AND BOTTOM LEFT: Estimated flow field solving (7.29) (EP: 0.156) with $\lambda = 0.3$, $\alpha_1 = 0.1$, $\alpha_0 = 1.0$, $F_{2,1} = \|\cdot\|_2$, $F_{2,0} = \|\cdot\|_2$ and a zoom. MIDDLE AND BOTTOM RIGHT: Flow field obtained with the nonconvex model (EP: 0.152) using parameters $\lambda = 0.25$, $\alpha_1 = 0.1$, $\alpha_2 = 1.0$, $F_{2,1}(|x|) = 2\log(1 + \frac{1}{2}|x|)$, $F_{2,0} = \|\cdot\|_2$ and a zoom. Using a nonconvex penalizer is beneficial and yields sharp discontinuities.

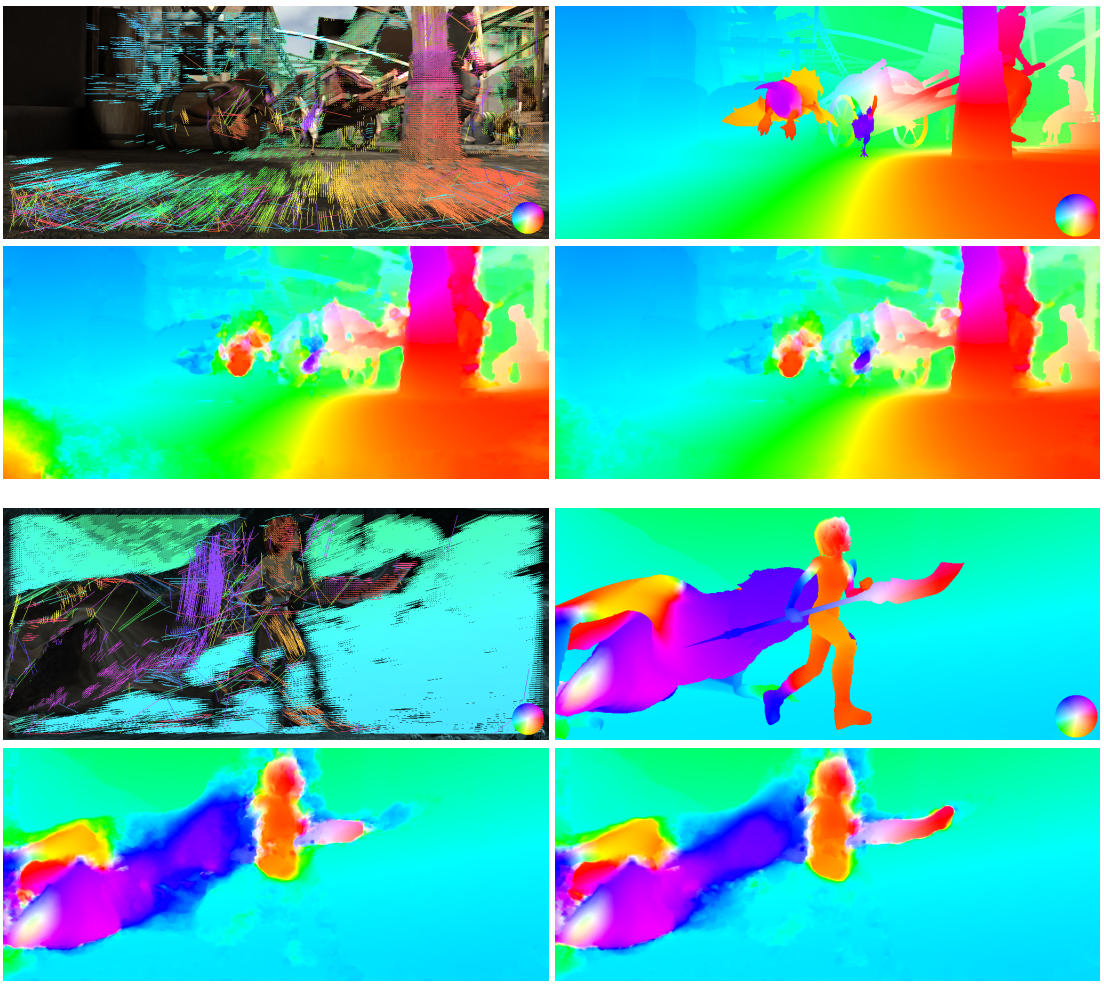


Figure 7.15: Result for two image pairs from the Sintel optical flow benchmark using the model (7.30). UPPER AND LOWER BLOCK, TOP LEFT: Feature matches for *market6_0005* and *cave2_0015*, respectively. BOTH BLOCKS, TOP RIGHT: Ground truth flow. UPPER BLOCK, BOTTOM LEFT: cFMcTGV (EP: 11.78). UPPER BLOCK, BOTTOM RIGHT: ncFMcTGV (EP: 7.77). LOWER BLOCK, BOTTOM LEFT: cFMcTGV (EP: 9.20). LOWER BLOCK, BOTTOM RIGHT: ncFMcTGV (EP: 8.79). The two image pairs are chosen because they require a complementary usage of the feature matches, in *market6_0005* matches at the bottom should be considered as outliers, whereas in *cave2_0015*, the few matches on the dragon are important. With the algorithm proposed in this chapter the usage of a robust penalty for the feature matching term is possible. Such a robust penalty is used for ncFMcTGV. In cFMcTGV a ℓ_1-penalty is used. The parameters for both methods have been optimized. ncFMcTGV can deal with the two complementary requirements of the feature correspondences much better than cFMcTGV.

Part II

Motion Segmentation

Chapter 8

Introduction to motion segmentation

Bottom-up segmentation based on color can successfully provide so-called superpixels—small, homogenous regions, which are actively used in many vision applications [AMFM11]. But what about the segmentation of a whole object or meaningful parts of an object? For example a person could wear clothes of very different color; see Figure 8.1. How can a bottom-up approach decide which of these regions must be grouped together? Top-down object priors can resolve such ambiguities, but based on which data can these priors be learned in the first place?

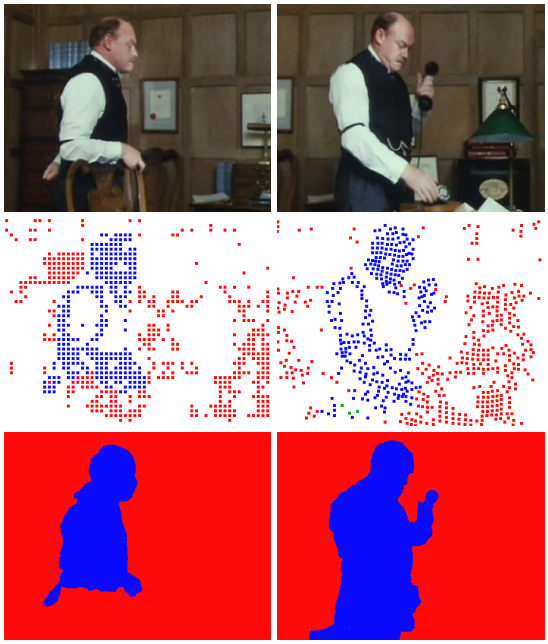


Figure 8.1: TOP ROW: Two images from a video shot. A color based segmentation would not provide object regions. MIDDLE ROW: Clustering of point trajectories indicates regions with similar motion. BOTTOM ROW: Segmentation based on these clusters provides object regions. The segmentation results are obtained with our method.

In this part of the thesis, the value of motion and the Gestalt principle of "common fate" [Kof35] is reemphasized. Motion vectors are typically more homogeneous within an object region than color and texture. Consequently, ambiguities in color based segmentation disappear as soon as objects move. Studies with formerly blind people indeed show that learning from moving objects is easier than learning from static ones [OMG+09].

In this chapter, related work and the motivation of my research is discussed. The foundation is the work of Brox and Malik [BM10], which is considered in detail in Section 8.2. Moreover, Sparse Subspace Clustering [EV13] is summarized more intensively than other lines of research in Section 8.3. As state-of-the-art method on the Hopkins155 benchmark [TV07] it appears in the evaluation in Chapter 12.

8.1 Related work

Two-frame optical flow. As a first step in the direction of motion segmentation we mention the classical approach of segmenting two-frame optical flow. While early approaches estimate the optical flow and the segmentation independently [WA94, SM98], optical flow estimation and segmentation were later considered as a joint optimization problem [CS05, AK06, BBW06, SSB12]. Obviously, object segmentation is only possible with such an approach, if the object motion is distinct and different from the background motion in *all* frames. Moreover, as pairs of frames are considered independently, the resulting segmentations are not consistent over time. In Chapter 12 such a method where the optical flow and the segmentation are computed independently is implemented as a baseline for comparison on the benchmark that is introduced in Chapter 9. [XS05, PTZ08] approach these problems by combining motion analysis with a learned appearance model. However, all these methods do not fully exploit the power of motion for object segmentation.

Importance of long term analysis. There can be long periods during which a person is as static as a pillar. Moreover, articulated objects do not move homogeneously. Arms and legs of a walking person move in opposite directions. All this causes severe problems in typical motion segmentation approaches based on two-frame optical flow. Tracking the interplay of the articulated parts over longer periods yields the missing information about the overall motion. Hence, information about what is an object should be analyzed over longer periods. Such long term analysis decreases the motion's intra-object variance relative to the inter-object variance. Moreover, motion information can be propagated to frames in which the object is mainly static. This is a basic building block of the motion segmentation method from Brox and Malik, see Section 8.2.

Superpixel based methods. A straight forward approach for obtaining a temporally consistent segmentation is based on superpixels. There are many works that produce over-segmentations and connect the emerging superpixels over time using optical flow and/or clustering methods [BT09c, GKHE10, VAPM10, LASL11, GINC11, GCS12]. This yields dense, temporally consistent segmentations, but usually they remain as over-segmentations. It is not trivial to retrieve object regions from these results. There are many challenges in tracking regions. This problem can be circumvented by interactive video segmentation methods [BS07, PMC09], which however, require significant user input.

Clearly, some kind of tracking is necessary for such long term analysis. We found that pursuing a "sparse to dense" strategy works best. There are several arguments in favor of point tracking like in [BM10]. We introduce it in Section 8.2.1. Point tracking concentrates on the reliable and stable features. Usually, superpixel are designed in a way such that homogeneous pieces are grouped. Hence, obviously, the important information for tracking lies at the superpixels' boundaries rather than in the interior. Also superpixel that incorporate texture cues suffer from similar problems. Texture is, by definition, a repetitive structure. Thus, matching of points inside a texture region is highly unstable. Although, in principle, texture also deteriorates the optical flow quality. However, this issue is much weaker than for superpixel trajectories. The image pyramid that is used in the optimization process for optical flow acts against this issue and, usually, the regularizing term propagates the actual region's motion from its boundary to the interior. Another reason against the usage of superpixel trajectories stems from occlusion and disocclusion areas. Due to these phenomena, which are present all the time, the shape of the superpixels change dramatically. Therefore, in this thesis, we focus on methods based on point tracking like [BM10, EV13].

Multi-body factorization. A dominant paradigm for clustering point trajectories has emerged from the technique of multi-body factorization [CK95, Gea98, BB91]. This approach is based on an affine camera model. The coordinates along the point trajectories are collected as columns of a matrix. This matrix is then decomposed into a matrix representing a 3D rigid body motion and a structure matrix. Trajectories that belong to the same 3D rigid body motion must lie on the same linear or affine subspace.

Subspace clustering. More recent methods directly model the (linear) dependency of the data samples [YP06, EV13]. The problem of motion segmentation is cast as the problem of segmenting samples drawn from a union of linear (or affine) subspaces. This allows defining affinities between trajectories and the use of spectral clustering. For instance in [EV13] the dependency is modeled as an optimization program where data points express themselves as linear combinations with a sparsity prior on the representatives. In Section 8.3, the approach of [EV13] is explained in detail. In [LS09] the dimensionality of the ambient space is explored and affinities are defined using angular information. A few works also explore the projective dependency among the data samples [LKSV07, SSW08]. While initially all these techniques were very sensitive to noise, more recent models have solved this problem [EV13, VTH08, LLY10, RTVM08, LLDS12, ZLPS11]. However, the main limitation remains the requirement of a dominant subset of complete trajectories. Consequently, the methodology cannot deal with strong occlusion and disocclusion, which hampers sincere long term motion analysis. This is a big issue when it comes to the analysis of real world sequences where most flexibility is gained by considering asynchronous trajectories. This fact is analyzed in the evaluation in Chapter 12.

Object segmentation by long term analysis of point trajectories. The work [BM10], Section 8.2, has overcome this drawback and is able to successfully analyze asynchronous trajectories on several real world video sequences. On the other hand, this new flexibility comes at the cost of a weaker analysis of the trajectory similarities. Instead of affine motion, [BM10] relies on a translational motion model. At first glance, this seems to be a severe restriction. However, as the considered point trajectories are much more dense than the ones previously considered the loss by the simpler motion model is compensated. Locally, every motion, even affine motions, are sticked together by translations. Nevertheless, the model is not as accurate as it could be. Chapter 10 addresses this issue and proposes a solution that improves their motion segmentation method.

As this work serves as the starting point for this thesis we ignore further details here and outline the approach more in detail in Section 8.2.2.

Other works based on point trajectories. There are few works which analyze point trajectories without the need to have a dominant subset of trajectories covering the full time line [SSZ06, BC06, CR09, FRP09]. Apart from [BC06], which analyzes trajectories but runs the clustering on a single frame basis, these methods provide temporally consistent clusters. The general idea of defining affinities between trajectories has been used in traffic scenarios already in 1997 [BMCM97].

The methodology of Brox and Malik [BM10] inspired other works. In [LASL11] the affinities between point trajectories are additionally influenced by reasoning about occlusion. In [FS11] the similarity, dissimilarity, and connectedness among trajectories is used to define affinities. In [FZS12] the (spectral) embedding of the point trajectories as in [BM10] via spectral clustering is clustered in a different way: instead of finding semantic motion clusters the goal is to is find motion discontinuities. In [FZZS12] segmentation is combined with detection. [ZYC$^+$12] transfered the knowledge about point trajectories to region trajectories, and directly obtain a dense segmentation.

Motion segmentation as means to an end. In some works, motion segmentation is not the primary goal but a way to achieve a higher level goal. Ommer et al. [OMB09] couple motion segmentation with a recognition task, in [WWR+13] tracking and detection is combined with geometric information for 3D scene modeling, [BSFC08] focuses on road scene understanding, in [WTG06] trajectory clustering is used for learning traffic models, and [SALT09] reconstruct 3D point clouds for semantic segmentation and object recognition.

8.2 Motion segmentation of Brox and Malik

The video segmentation method [BM10] is roughly composed of two steps: (1) tracking and (2) clustering; each is interesting and challenging in its own right, but is also a potential source for improvements.

(1) Tracking is closely related to motion estimation. For the object being tracked the correspondence from one image to another is sought; this is the same as the motion estimation between the images: the optical flow problem. Different to common tracking algorithms, which extract a sparse set of feature points in both images and try to relate them, the method [BM10] uses the dense correspondences from an optical flow algorithm. This allows it to track a rather dense set of points. Compared to tracking patches, points are easier to track, as it is naturally "invariant" to, for example, rotation or scaling.

(2) Moreover, any object can be represented by points. In order to obtain a segmentation of moving objects the remaining step is to cluster the point tracks (point trajectories). Similarities based on motion differences are estimated between them. Then, the desired clustering is obtained by optimizing for groups with a high similarity of point trajectories.

Details are introduced in the following.

8.2.1 Point tracking

The goal of point tracking is to extract suitable points and to compute their trajectories along which they appear in a video sequence $I\colon \Omega \times [0,T] \to \mathbb{R}$. In the motion segmentation framework [BM10] point trajectories are used to obtain an unsupervised segmentation of a video. Tracking points is easier than tracking patches or objects. Nevertheless, groups of points can be used to represent objects, which is the basic idea of their motion segmentation method and thus our further development.

8.2.1.1 Large displacement optical flow tracker

The simple but successful idea to obtain high quality point trajectories is the following. Points are tracked based on a current state-of-the-art optical flow method; here we use large displacement optical flow from [BM11]. This way, we benefit from all the progress made on optical flow estimation in the 30 years since the Lucas-Kanade method [LK81] was presented, which is the basis for the KLT tracker [ST94]. In the following, we coarsely describe the most important aspects of a point tracker. For more details we refer the reader to [SBK10].

Initial points. Like in every tracker, a set of points is initialized in the first frame of a video. As we build on a dense optical flow method, in principle, we could initialize with every pixel. However, homogeneous areas can be problematic for variational optical flow. To put more emphasis on points that

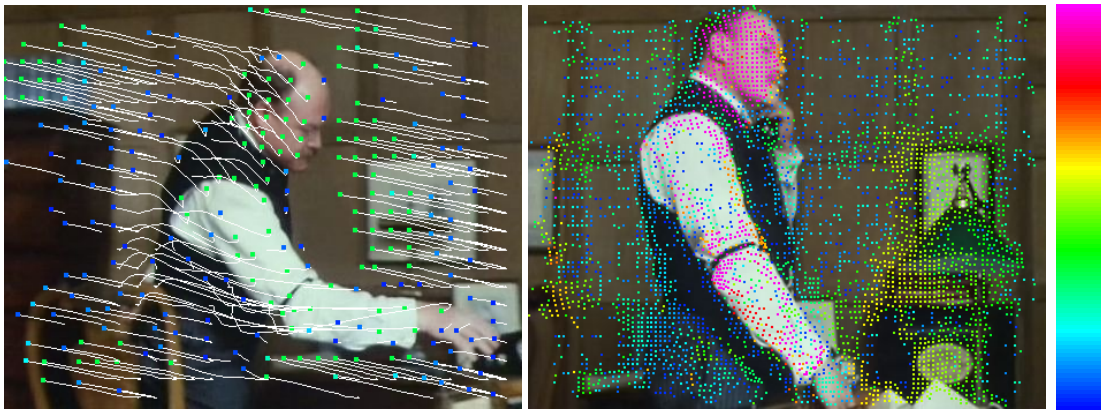


Figure 8.2: Two visualizations of the trajectories. LEFT: Current position of the tracked points together with their trajectories. The subsampling factor was 16. RIGHT: Only the current position of the tracked points is shown. The subsampling factor was 4. In both cases color shows for how long the points have been tracked as a percentage of the length of the shot (see color bar). Occlusion and disocclusion prohibit permanent tracking.

can be tracked more reliably, we remove points that do not show any structure in their vicinity based on the smaller eigenvalue of the structure tensor.

Let $I_0(x) := I(x,0) \in C^1(\Omega,\mathbb{R})$ be the first frame of the video sequence and assume reflecting boundary conditions for the image function. Then, the structure tensor is defined for $x \in \Omega$ as $S_\rho(x) := (k_\rho * (\mathrm{J}_{I_0}^\top \mathrm{J}_{I_0}))(x) \in \mathbb{R}^{2\times 2}$, where the convolution is meant entry-wise (of the $\mathbb{R}^{2\times 2}$ matrix), J_{I_0} is the Jacobian matrix of the image function I_0 at point $x \in \Omega$ and $k_\rho\colon \mathbb{R}^2 \to \mathbb{R}$ is a compactly supported Gaussian kernel with standard deviation $\rho = 1$. The criterion for excluding points $x \in \Omega$ for tracking is $\lambda_2(x) < \frac{1}{8}|\Omega|^{-1} \int_\Omega \lambda_2(x')\,dx'$, where $\lambda_2(x)$ is the smaller eigenvalue of the structure tensor at point x.

As we will see in Section 8.2.2.2, the computational complexity of the motion segmentation method is quadratic in the number of point trajectories. For efficiency reasons, we spatially subsample the initial points. Figure 8.2 shows a subsampling by factor 4 on the right and 16 on the left side. Factors larger than 12 result in loss of details as there are not enough points to cover small object parts. On the other hand, factors smaller than 4 waste computation time, as smaller objects tend to be smoothed away by the optical flow anyway.

Tracking. Each of the points $x \in \Omega$ can be tracked to the next frame $t+1$ by using the optical flow field $\mathbf{w}_t\colon \Omega \to \mathbb{R}^2$ at frame t. The point in the next frame is given by $x + \mathbf{w}_t(x)$. If $x + \mathbf{w}_t(x) \in \mathbb{R}^2 \setminus \Omega$, i.e. w_t maps x outside the image domain, the trajectory is stopped at frame t. In principle, any optical flow method can be used here, yet many of the problems we find in motion segmentation are due to shortcomings of the optical flow, e.g., large displacements, sharp discontinuities, and accuracy for the occlusion detection. Hence, it is important to use a strong method. The approach from [BM11] combines the subpixel accuracy of variational approaches with combinatorial feature matching, which allows to capture large displacements. Moreover, an efficient GPU implementation [SBK10] computes the optical flow between two 640×480 frames in less than 2 seconds. This also enables tracking in long, high resolution sequences in a reasonable time.

Occlusion detection. Tracking has to be stopped as soon as a point gets occluded. This is very important, as otherwise the point trajectory will share the motion of two different objects. Occlusion detection is a common problem, considered especially in disparity estimation, but recently has also appeared more often in conjunction with optical flow. We refer to a recent work [ARS12] and the references therein. In tracking, occlusion is usually detected by comparing the appearance of local neighborhood of a tracked point over time. In contrast, we detect occlusions by verifying the consistency of the forward and the

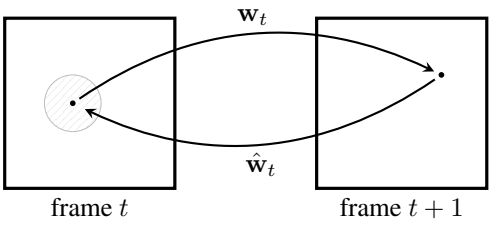


Figure 8.3: Forward–backward matching criterion. Each pixel in frame t is mapped to frame $t+1$ via the optical flow vector $\mathbf{w}_t$. The backward map $\hat{\mathbf{w}}_t$ at the subpixel position is determined by bilinear interpolation. Concatenating the two mappings should result in approximately the original position.

backward flow, as illustrated in Figure 8.3. Let $\hat{\mathbf{w}}_t \colon \Omega \to \mathbb{R}^2$ be the backward flow from frame $t+1$ to frame t and let $x, x + \mathbf{w}_t(x) \in \Omega$. Then the trajectory with coordinate x is stopped if

$$|\mathbf{w}_t(x) + \hat{\mathbf{w}}_t(x + \mathbf{w}_t(x))|^2 < 0.01(|\mathbf{w}_t(x)|^2 + |\hat{\mathbf{w}}_t(x + \mathbf{w}_t(x))|^2) + 0.5\,.$$

In a non-occlusion case, the backward flow vector points in the opposite direction of the forward flow vector. If this consistency requirement is not satisfied, the point is either getting occluded at $t+1$ or the flow was not correctly estimated. Both are good reasons to stop tracking this point at t. Since there are always some small estimation errors in the optical flow, we grant a tolerance interval.

Occlusion comes together with the opposite phenomenon: disocclusion or scaling. To fill these areas not covered by a trajectory yet, new trajectories are initialized in empty areas in each new frame using the same strategy as for the first frame.

Sometimes, due to inaccurate optical flow, the occlusion check fails. Therefore, trajectories are also stopped close to motion boundaries, which however fluctuates a little. In order to avoid drifting points from one object to the other the following stopping criterion is additionally used

$$\| \mathrm{J}_{\mathbf{w}_t}(x)\|_2^2 > 0.01|\mathbf{w}_t(x)|^2 + 0.002\,,$$

where the norm on the left hand side is the matrix norm $\| \mathrm{J}_{\mathbf{w}_t}(x)\|_2^2 = \mathrm{trace}((\mathrm{J}_{\mathbf{w}_t}^\top \mathrm{J}_{\mathbf{w}_t})(x))$.

A tracking example is shown in Figure 8.2. Some points on the man are tracked across all 75 frames. However, most trajectories are newer due to disocclusion.

8.2.2 Object segmentation by long term analysis of point trajectories

The work of Brox and Malik [BM10] is based on asynchronous point trajectories obtained with the optical flow supported point tracker [SBK10] which is outlined in Section 8.2.1. The most important advantages of this tracker are that (1) it is based on dense optical flow and, therefore, up to 100% dense point trajectories can be generated, (2) usually it provides reliable tracks covering several hundred frames of a video, and (3) it is not sensitive towards occlusions; tracks are simply stopped.

In the remainder of this section, we recap the methods presented in [BM10] and improved in [OMB14]. The notation is from [OMB14].

8.2.2.1 Analysis of point trajectories

It is clear that, as the trajectories are asynchronous, the set of considered trajectories can not be restricted to the complete ones only. Particularly, in video shots with fast motion and large occlusions this set could

be even empty. The analysis of the trajectories' motions must be defined on subsets of the video shot. Therefore, the similarities (or affinities) between trajectories are defined between all pairs of trajectories with overlapping frames, where both are visible. The transitivity of this procedure implicitly spreads the information of similarity to all trajectories; There is knowledge about the similarity of trajectories even if they live in disjoint time windows.

According to the Gestalt principle of common fate, high affinities are assigned to pairs of points that move together. Clearly, motion is not always decisive for grouping objects, e.g., two horses galloping next to each other move similarly and are considered as a single object. This reflects the basic assumption in object level motion segmentation. As soon as one of the two horses stops there is the required information needed for grouping them separately. This indicates that the actual information is found in the motion dissimilarity.

Let A and B be two trajectories with coordinates (x_t^A, y_t^A) and (x_t^B, y_t^B), at frame t. According to the previous discussion, we introduce the distance measure $d(A, B)$ for a pair of trajectories A and B in the common time window by exploiting their maximal dissimilarity, i.e., the maximal motion difference among all frames of common visibility

$$d^2(A, B) = \max_t d_t(A, B)\,, \tag{8.1}$$

and turn them into affinities via

$$w(A, B) = \exp\left(-\lambda d^2(A, B)\right)\,. \tag{8.2}$$

The scale parameter $\lambda = 0.1$ is fixed. This parameter leads us to another important issue, namely proper normalization. So far the affinity model is based on the two assumptions that the motion estimates are noise-free and all object motion is translational. Of course, both assumptions are not satisfied in real sequences, so we face the problematic question: when is a motion difference just due to noise and when is it significant enough to indicate different objects?

As this question is easier to answer if there is less noise, the first objective is to limit the noise that should be expected. On the side of optical flow, we can add some more accuracy by averaging the motion over time. This is done by approximating the derivatives $\partial_t A$ and $\partial_t B$ of two continuous spatio-temporal curves, defined by the trajectories A and B, at time t with forward-differences over $T = 5$ frames:

$$\partial_t A(t) = \frac{1}{T}(x_{t+T}^A - x_t^A, y_{t+T}^A - y_t^A)^\top \tag{8.3}$$

The same for B. If less than T common frames are available between A and B, then T is set to the number of common frames for this pair. The exact choice of T is not critical. If T is chosen too large, we might lose relevant motion differences, e.g., due to a swinging arm. At frame rates of 30fps, $T = 5$ corresponds to just 160ms, and it is unlikely that this will smooth out some significant motion detail. Values of $T = 10$ and $T = 15$ were tested without any consistent positive or negative effect on the results[1].

A second source of noise is model noise due to the assumption that all object motion is translational. Clearly, objects can undergo more complex motion. Figure 8.4 shows a failure case, where motion is dominated by scaling. Pairwise distances only allow for verification of a translational model, whereas an affine motion model would require distances computed for at least 4 trajectories at a time to verify if

[1] It is worth noting that temporal smoothing of the optical flow for the purpose of computing motion differences between trajectories is uncritical, whereas such smoothing can have very negative effects on the optical estimation process in case of camera jitter. The reason is that temporal smoothing during optical flow estimation hampers the correct matching of pixels. Such a problem does not exist when analyzing motion differences.

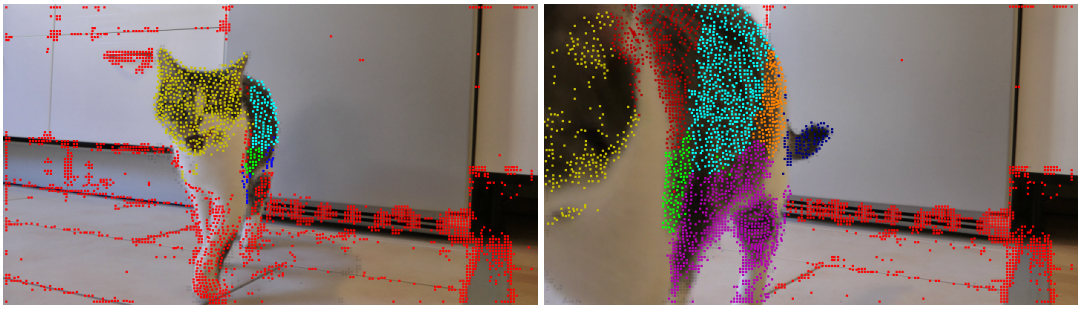


Figure 8.4: Sequence with dominant scaling motion that cannot be captured well enough by a local translational model. These situations lead to an over-segmentation of the object.

they belong to the same group. This leads to a hypergraph and will be considered in Chapter 10. Here we rather make use of the dense coverage of the image by trajectories and the fact that *locally* a higher order motion model can be approximated by a translational model. Consequently, we can limit the effect of model noise by damping the motion distance with the average spatial distance between trajectories A and B.

The distance at frame t between two trajectories A and B is defined as

$$d_t^2(A,B) = d_{\text{sp}}(A,B)\frac{|\partial_t A(t) - \partial_t B(t)|^2}{\sigma_t^2}, \tag{8.4}$$

where σ_t^2 is a locally adaptive normalization factor that deals with the fact that despite the above measures there is still some noise to be expected. The magnitude of the noise depends on the variation of the motion in the image. A larger variation indicates fast higher order motion and hence more model noise. Consequently, the distance should be normalized by the variance of the optical flow in the considered image. The intuition behind this normalization is that a motion difference of two pixels is a lot when there is hardly any motion in a scene, whereas the same motion difference is negligible in a scene with fast motion.

If there is just one object and the background, normalization by the global flow variance is sufficient, yet consider a scenario with one fast object and one slowly moving object. For the fast object, σ should be large, otherwise the object might be split into multiple regions. For the slow object, σ should be smaller to avoid that the object is merged with the background. This dilemma can be avoided by using a spatially adaptive variance estimate that is computed for each point individually in a local neighborhood. For an efficient computation of such local statistics we refer to [BC09].

8.2.2.2 Spectral clustering with spatial regularity

The pairwise affinities for n trajectories result in an $n \times n$ affinity matrix W. An (approximately) optimal partitioning of the underlying graph is obtained via spectral clustering [SM00, NJW02]. Let $D = \text{diag}\,(d_A | A = 1, \ldots, n)$ be the $n \times n$ diagonal matrix with entries $d_A = \sum_B w(A,B)$. The eigen-decomposition of the normalized graph Laplacian reads

$$V^\top \Lambda V = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}. \tag{8.5}$$

We keep the eigenvectors $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_m$ corresponding to the $m+1$ smallest eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_m$ according to the threshold $\max_i \lambda_i < 0.2$. The trivial solution $\lambda_0 = 0$ with the constant eigenvector

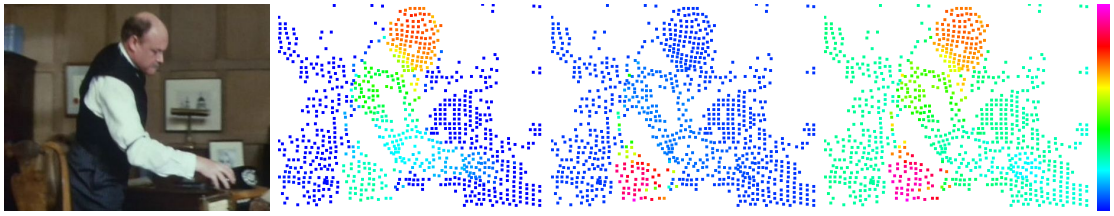


Figure 8.5: FROM LEFT TO RIGHT: Input frame from a video shot and the first 3 eigenvectors with range of values represented by the colorbar. Clearly, the eigenvectors are not piecewise constant but show smooth transitions within the object regions. However, discontinuities in the eigenvectors correspond to object boundaries very well. This information needs to be exploited in the final clustering procedure.

$\mathbf{v}_0$ is omitted. Since the number of objects is expected to be significantly smaller than the number of trajectories, i.e., $m \ll n$, the eigenvectors and eigenvalues can be efficiently computed using the Lanczos method in $\mathcal{O}(n^2)$. For further computations we normalize the eigenvectors' range to $[0, 1]$.

We also determine the number of clusters automatically (model selection). In the ideal case, i.e., clearly distinguishable translational motion and very few tracking errors, we obtain m piecewise constant eigenvectors and clusters are easily obtained with k-means clustering. There is a large number of model selection criteria in the literature, such as BiC or AiC, to automatically choose K in k-means clustering. As long as sufficiently many eigenvectors are computed, which is usually guaranteed with our conservative threshold on λ, such model selection will find a good number of clusters.

However, often the eigenvectors are not piecewise constant, as shown in Figure 8.5. Standard k-means clustering is not suited for this setting as smooth transitions in the eigenvectors get approximated by multiple constant functions and, thus, leads to an over-segmentation. This has a strong negative effect on the correct choice of the number of clusters K.

As a remedy, we suggest minimizing an energy function that comprises a spatial regularity term. This regularity term not only prefers spatially compact clusters, it also acts as a criterion for model selection. Moreover, it takes edges in the eigenvectors into account. Let v_i^A denote the Ath component of the ith eigenvector and $\mathbf{v}^A$ the vector composed of the Ath components of all m eigenvectors. Index A corresponds to a distinct trajectory. Let $\mathcal{N}(A)$ be the symmetrized set of 12 neighboring trajectories based on the average spatial distance of trajectories. We seek to choose the total number of clusters K and the assignments $\pi^A \in \{1, ..., K\}$ such that the following energy is minimized:

$$E(\pi, K) := \sum_A \sum_{k=1}^{K} \delta_{\pi^A,k} |\mathbf{v}^A - \mu_k|_\lambda^2 + \nu \sum_A \sum_{B \in \mathcal{N}(A)} \frac{1 - \delta_{\pi^A,\pi^B}}{|\mathbf{v}^A - \mathbf{v}^B|} \,. \tag{8.6}$$

Increasing the parameter $\nu > 0$ puts more weight on compact clusters and less over-segmentation. The first term is the unary cost, where μ_k denotes the centroid of cluster k. The norm $|\cdot|_\lambda$ is defined as

$$|\mathbf{v}^A - \mu|_\lambda^2 := \sum_i (v_i^A - \mu_i)^2 / \lambda_i \,, \tag{8.7}$$

i.e., each eigenvector is weighted by the inverse of the square root of its corresponding eigenvalue. This weighting is common in spectral clustering as eigenvectors that separate more distinct clusters correspond to smaller eigenvalues [BM98].

Clearly, if we do not add a penalty for additional clusters, each trajectory will be assigned its own cluster. The second term in (8.6) serves as a regularizer by penalizing the spatial boundaries between clusters. The δ_{π^A,π^B} is the Kronecker delta, which is 1 if the trajectories A and B are assigned to the

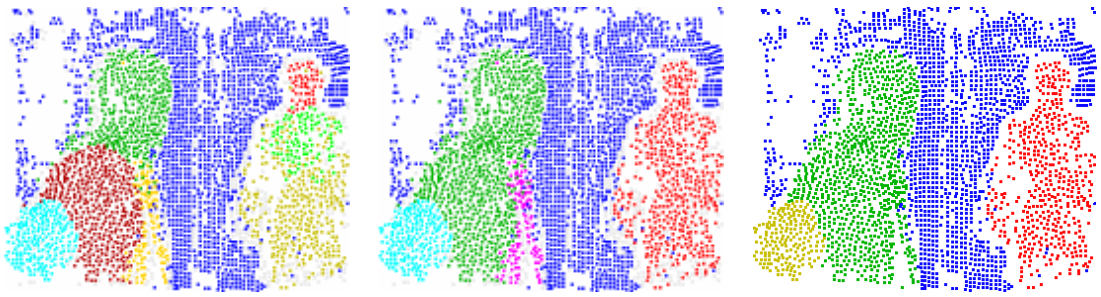


Figure 8.6: Optimization procedure for minimizing (8.6) from [BM10]. LEFT: Best k-means proposal obtained for $K = 9$. Over-segmentation due to smooth transitions in eigenvectors. CENTER: Remaining 5 clusters after choosing the best merging proposals. RIGHT: Final segmentation after merging using affine motion models. Another cluster boundary that was due to the fast 3D rotation of the left person has been removed. The only remaining clusters are the background, the two persons, and the articulated arm of the left person.

same cluster, and 0, otherwise. The penalty is weighted by the inverse differences of the eigenvectors along these boundaries. Consequently, cutting a smooth transition in the eigenvectors will induce much higher cost than cutting along a strong discontinuity. This avoids splitting clusters at arbitrary locations due to smooth transitions in the eigenvectors. The parameter ν steers the tradeoff between the two terms.

Minimizing (8.6) is problematic due to many local minima. In [BM10], (8.6) is minimized using a heuristic k-means clustering. In [OMB14], we replace this heuristic optimization by iteratively solving multi-label Markov Random fields. This method will be detailed in Section 11.3.

In [BM10], multiple runs of k-means or hierarchical 2-means clustering are executed with random initialization. This generates several hypotheses for minimizing the first term in (8.6). The second term enters the optimization by merging the hypotheses such that the energy decreases. After merging, a gradient descent step is run to correct a few erroneous decisions. The whole optimization procedure is executed for $K \in \{1, ..., 2m\}$ clusters and the result with the smallest energy is used as "minimizer" of (8.6). For details about how often k-means or 2-means is executed for each K we refer to [BM10].

Finally, in [BM10] and [OMB14] a postprocessing step is made. Clusters are merged according to the mutual fit of their affine motion models estimated via least squares in each frame. The average fit per frame is considered and merged greedily until a threshold is reached. This postprocessing step is not absolutely necessary, but corrects a few over-segmentation errors. Figure 8.6 from [BM10] visualizes the optimization procedure.

8.3 Subspace clustering

Among subspace clustering methods [EV13] is currently considered as the state-of-the-art. As we want to compare against this work, we discuss it in more detail here. In the following, first, we derive the abstract optimization problem of sparse subspace clustering, then, we cast the motion segmentation problem as such.

In the subspace clustering problem a collection of N data points $x_j \in \mathbb{R}^D$, $j = 1, \ldots, N$, is modeled by a union of (affine) subspaces S_i, $i = 1, \ldots, K$ of dimension $\dim S_i = d_i$, $0 < d_i < D$. The difficulty is that not only the subspaces are unknown, but also the assignments of data points to these subspaces are unknown. There are several approaches to that problem [YP06, EV13, VTH08, LLY10, RTVM08, LLDS12, ZLPS11, LS09] (see Section 8.1 for a more detailed distinction), however we focus on Sparse Subspace Clustering (SSC), since practically it has proved to be among the best performing methods.

8.3.1 Sparse subspace clustering

Although we assume that the bases of the subspaces are unknown, there is a way to model the subspaces. SSC builds on the fact that the data points themselves describe their subspaces. Each data point x_j, $j = 1, \dots, N$, can be written as a linear combination of all other data points

$$x_j = \sum_{i=1}^{N} z_{ij} x_i, \quad z_{jj} = 0, \quad Z = (z_{i,j})_{i,j} \in \mathbb{R}^{N \times N}. \tag{8.8}$$

For all points x_i on a different subspace than x_j the corresponding weight z_{ij} is zero. Obviously, it makes sense to avoid x_j being represented by x_j, thus $z_{jj} = 0$ is required, i.e. $\operatorname{diag} Z = 0$. In Matrix notation: let $X \in \mathbb{R}^{D \times N}$ collect data points $X = (x_1, \dots, x_N)$ and $Z \in \mathbb{R}^{N \times N}$ the coefficients of the linear combinations. Then (8.8) is equivalent to

$$X = XZ, \quad \operatorname{diag} Z = 0\,.$$

In [EV13], the property of X being representable by X is called *self-expressiveness*.

SSC seeks for a sparse representation within the (potentially) huge solution space. It is clear, that if the subspaces comprise data points in general direction, then there exist sparse coefficient vectors Z_{*j} (j-th column of Z) representing a data point x_j with only points from the same subspace. According to [EV13], this means that there is a *subspace-sparse* representation. Such a subspace-sparse representation can be obtained by solving the following optimization problem

$$\min_{Z} \|Z\|_1, \quad s.t.\ X = XZ,\ \operatorname{diag} Z = 0 \text{ and } Z\mathbf{1} = \mathbf{1}\,,$$

where the last term $Z\mathbf{1} = \mathbf{1}$ models unit row-sum of Z (i.e. the entries in a row sum up to 1), which accounts affine subspaces. Actually, one would like to minimize the ℓ_0-norm $\|Z\|_0$, however this problem is NP-hard [AK98]. Therefore, the ℓ_1-norm, the tightest convex relaxation of the ℓ_0-norm is used. The ℓ_1-norm also has the capability to find a sparse representation. Although the sparsity restricts the solution space significantly there is still a lot of ambiguity.

Note that the derivation so far assumes clean data and no noise. In [EV13] it is shown in detail how to cope with gaussian noise and outliers. We introduce this noise handling terms ad hoc here, since it is intuitive. The final optimization problem reads

$$\begin{aligned} \min_{C,Z,E} \|Z\|_1 + \frac{\lambda_C}{2}\|C\|_2^2 + \lambda_E \|E\|_1 \\ s.t.\ X = XZ + E + C,\ \operatorname{diag} Z = 0 \text{ and } Z\mathbf{1} = \mathbf{1}\,. \end{aligned} \tag{8.9}$$

The constraint $X = XZ + E + C$ grants a certain error for the representation of X. The penalty $\|C\|_2^2$ models gaussian noise in the data samples and the term $\|E\|_1$ allows for a sparse set of outliers. The optimization problem is still convex and can be solved efficiently. For example in [EV13], it is solved using the alternating direction method of multipliers. For details we refer to [EV13].

For motion segmentation the interesting part of the optimization problem is the representation matrix Z. Entries with high value can be interpreted as stronger dependency of the respective data samples. Therefore, the coefficient matrix Z can be used as affinity matrix. Spectral clustering on the symmetrized matrix $|Z| + |Z^\top|$ yields the final assignment of data points to a subspace, where the absolute value here is meant per entry.

8.3.2 Motion segmentation using sparse subspace clustering

Motion segmentation is considered as the task of segmenting a video shot of rigidly moving objects [EV13]. Usually, the video shot is represented by a set of $N \in \mathbb{N}$ trajectories, which mainly contains trajectories of the same length $F \in \mathbb{N}$. The coordinates of the trajectory are collected as columns of a matrix $X = (x_1, \dots, x_N) \in \mathbb{R}^{2F \times N}$, i.e., $x_i \in \mathbb{R}^{2F}$ is the column vector that concatenates the (x, y)-coordinates of trajectory $i \in \{1, \dots, N\}$ along F frames of the video. The matrix X is the $\mathbb{R}^{D \times N}$ matrix of data samples or feature vectors from Section 8.3.1.

As an affine camera model is assumed, the trajectories lie on (at most) three dimensional affine subspaces of $\mathbb{R}^{2F}$. The goal is to assign these feature vectors, i.e. the trajectories, to different subspaces according to the underlying motion. In practice, there is always noise. Therefore, the optimization problem (8.9) is solved and a sparse representation matrix Z is obtained. As mentioned in Subsection 8.3.1, in [EV13] spectral clustering is used to reveal the subspace assignments, which means the assignments of each trajectory to a (dominant) motion model. For implementation details and a appropriate normalizations for practical applications, we refer to [EV13].

Chapter 9

Motion segmentation benchmark

The problem of motion based object segmentation lacked a reasonably sized general benchmark. Before the Berkeley Motion Segmentation (BMS) benchmark was published in [BM10] the only available dataset was Hopkins 155 [TV07]. Unfortunately, both benchmarks come with obvious limitations.

Hopkins 155 is tailored to subspace clustering methods. In the beginning those methods only worked in a perfect environment, i.e., there are no outlying or noisy trajectories and they are all complete. Although now Hopkins 155 allows for synthetically added corruptions, the provided scenario is not realistic enough. This is particularly true for the checkerboard sequences from this benchmark. Additionally, the requirement of complete trajectories hides the challenge of occlusion detection and handling. We will confirm this statement in Chapter 12, when we evaluate the state-of-the-art method (SSC) of Hopkins 155 on the extension of the BMS benchmark that is proposed in this chapter. Some of the more realistic sequences from Hopkins 155 are used in the BMS benchmark, however, without providing manually repaired trajectories.

The limitations of the BMS benchmark are clearly the number of sequences, the number of ground truths, and the evaluation metric, which gives some freedom in the interpretation of the results. The BMS dataset is dominated by moving cars and persons under more or less clean camera motion and moderate occlusions. More details are discussed throughout this chapter. Large parts of this chapter are published in [OMB14].

Since motion segmentation is among the *hot topics* in computer vision, progress is made quickly. Although, the BMS benchmark cannot be considered as solved yet, a larger dataset with more variation among the sequences and other challenges is needed. In this chapter, we present our extension of the BMS dataset to the Freiburg Berkeley Motion Segmentation (FBMS) dataset, recapitulate the evaluation metric introduced in [BM10], and propose our new evaluation metric which allows to judge the quantitative quality of a method with a single number.

9.1 The Freiburg Berkeley Motion Segmentation dataset (FBMS)

The BMS benchmark introduced in [BM10] is composed of 26 video sequences. Among them are shots from detective stories and 12 sequences from Hopkins 155. Several frames of each shot come with pixel-accurate ground truth segmentation of moving objects. The ground truth annotation is consistent over time, i.e., the label associated with an objects is the same in all frames where a ground truth segmentation is provided. Due to the progress in motion segmentation new challenges are required. In order to avoid

stagnation in a field a benchmark must not be too simple, but also not too difficult. We extended the BMS dataset by adding 33 sequences. The new sequences show more variation in

- the image resolution,
- the number of moving objects,
- the smoothness of the camera motion,
- the motion complexity (more non-translational motion), and
- occlusion, disocclusion of the objects.

Every 20th frame comes with ground truth, adding a total of 516 annotated frames to the benchmark. The full dataset with 59 sequences and 720 annotated frames is publicly available at [FBM].

9.2 Evaluation metric provided in [BM10]

The evaluation tool introduced in [BM10] analyzes the (1) density, (2) the overall clustering error, (3) the average clustering error, (4) the over-segmentation error, and (5) the number of extracted objects. These numbers are computed for each video shot and averaged over all sequences.

(1) The *density* describes the percentage of points where a label has been estimated.

(2) The *overall clustering error* measures the percentage of wrongly labeled pixels. Thereby, the assignments of estimated clusters and ground truth regions is optimized. A single ground truth region may be assigned multiple times. In this case, a cost in terms of over-segmentation error is paid.

(3) The *average clustering error* is very similar to the overall clustering error, but on a region basis instead of a per-pixel basis. The labeling error is computed for each ground truth region individually, and then, averaged across all regions.

(4) The *over-segmentation error* counts the number of merging steps of the estimated clusters to fit the ground truth regions.

(5) The *object count* is the number of extracted objects measured by the number of regions covered with less than 10% error. Background does not count as an object and is therefore subtracted for each sequence.

Let us briefly discuss the weaknesses of these quantities.

The main issue with this evaluation method becomes obvious, when two methods with different objectives are to be compared. If one method produces a low clustering error and a high over-segmentation error and the other method a higher clustering error but a lower over-segmentation error, it is not clear which method should be rated as better. A second issue is the density. A method that densely labels the background but does not try to label objects, easily achieves a high density and a low clustering error. However, little information is extracted. Such a method should not be considered a good method. This evaluation metric clearly has the disadvantage that all 5 criteria have to be considered. Thereby, the weighting of these criteria is a matter of taste and methods are not directly comparable.

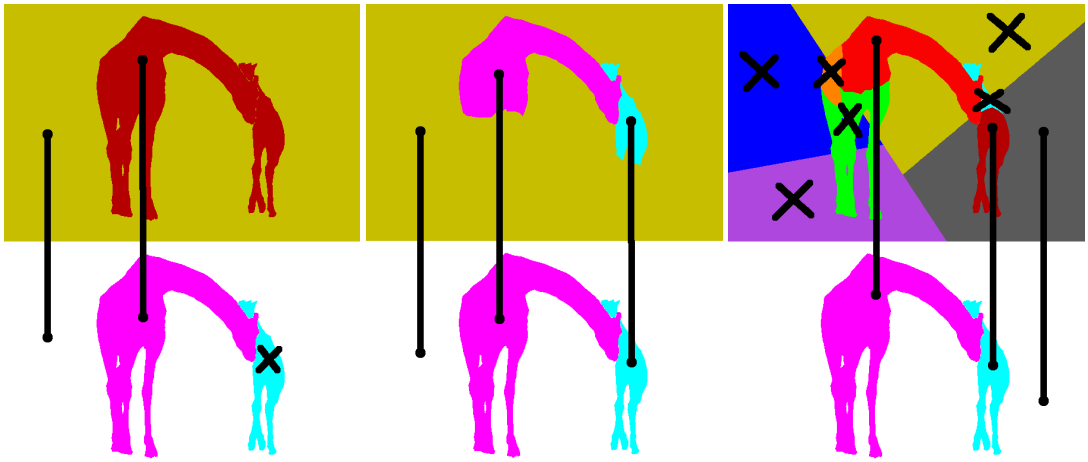


Figure 9.1: Illustration of the evaluation metric. FROM LEFT TO RIGHT: An under-segmentation, an unprecise segmentation, and an over-segmentation. The ground truth is shown in the bottom row. The black lines show the cluster–region assignments and crosses indicates clusters or regions that have not been assigned. Average precision, recall, and F-measure are $(93.68\%, 66.67\%, 77.9\%)$, $(98.22\%, 80.31\%, 88.36\%)$, and $(100.0\%, 56.02\%, 71.81\%)$.

Over-fitting to a certain dataset is always an issue, especially when a first state of saturation is reached. Intensive parameter tuning of a method on the BMS dataset can weaken the performance when other videos are considered. This situation can hamper the progress in the field of motion segmentation in videos. An appropriate splitting into a training set for parameter tuning and a test set for evaluation is desirable. However, the BMS dataset is too small. Our extension comes with several new sequences and allows for such a splitting of the dataset.

Next, the improved evaluation methodology presented in [OMB14] is explained. It unifies the evaluation in a single representative number for the clustering quality and one for the density and thus allows to compare methods that yield rather different results.

9.3 A new evaluation metric

The size of the new dataset allows for splitting it into a training set and a test set. We provide a fixed split into a roughly equal number of sequences in both sets. The split was chosen such that typical challenges appear in both sets.

We introduce an *average region density*, which is the average percentage coverage over all ground truth regions by labels. For a dense method the density is 100%; sparse trajectory clustering leads to lower densities depending on the spacing of the trajectories. By averaging over region densities rather than on a per-pixel basis, we penalize uneven spatial coverage.

To compare segmentations with a different number of output regions, a metric must reflect the tradeoff between accuracy (usually maximized by increasing the number of regions) and a good coverage of the ground truth. In detection tasks, precision and recall have proven valuable to capture a similar tradeoff between false positives and misses. Here we provide a definition of precision and recall for segmentation. Let C be the set of pixels[1] labeled by the computer algorithm and $c_i \subset C$ the subset assigned to cluster i; $g_j \subset C$ be the corresponding subset of a ground truth region j, and let $|\cdot|$ denote the size of the set.

[1] The set of pixels includes all frames with ground truth annotation.

The sets g_j only contain those pixels of a ground truth region that have been labeled by the evaluated computer algorithm. This allows a fair comparison of sparse and dense results on the basis of accuracy, whereas the density is measured by the above density measure. *Precision* is defined as

$$P_{i,j} := \frac{|c_i \cap g_j|}{|c_i|}, \tag{9.1}$$

the ground truth fraction of a cluster, and *recall* as

$$R_{i,j} := \frac{|c_i \cap g_j|}{|g_j|}, \tag{9.2}$$

the fraction of a ground truth region covered by the cluster. They are defined for each pair of cluster i and ground truth region j. The best assignment of clusters to ground truth regions is found by the Hungarian method[2], a one-to-one matching algorithm, where we maximize the *F-measure*

$$F_{i,j} := \frac{2P_{i,j}R_{i,j}}{P_{i,j} + R_{i,j}} \tag{9.3}$$

over all assignments. In case there are fewer clusters than ground truth regions, we introduce empty clusters. According to (9.2) their recall is $R = 0$, and we define $P = 1$. Unassigned clusters are ignored. Like in a typical detection setting, precision measures the percentage of correctly assigned pixels, and recall measures the covered fraction of the ground truth. However, since regions in a segmentation are disjoint, a lower recall usually does not increase precision, as in a typical detection setting. Figure 9.1 illustrates the effect of certain error classes on the metrics. Both an under-segmentation (leftmost example) and an over-segmentation (rightmost example) lead to a reduction in recall. In the first case $R = 0$ because one object is missed and in the second case because the assigned cluster covers only a small part of the ground truth region. Recall is mainly affected by a bad model selection. Precision is mostly affected by inaccurate clusters that overlap with multiple ground truth regions. The F-measure combines precision and recall and allows the comparison of approaches that yield different number of clusters, whereas with the evaluation metric in [BM10], it was unclear whether to prefer a result with lower pixel error or one with a lower over-segmentation error. It is important to note that averages over precision and recall values are computed on a per region basis rather than on a per pixel basis. The latter would put too much weight on large background regions. The average F-measure always refers to the harmonic mean of the average precision and average recall rather than the average of single region F-measures.

Finally, we report the total number of *extracted objects*, which we define as clusters with F-measure $\geq 75\%$. This quantity is to indicate the number of objects that can be extracted with a certain accuracy from a dataset. One region is subtracted per sequence to account for the background, i.e., at least two regions must satisfy the above criterion to increase the counter. We refer to Figure 9.2 and Figure 9.3 to justify the threshold of 75%, which is in any case disputable and could be adapted to the quality requirements of an application.

[2]The complexity of the rectangular Hungarian method is $\mathcal{O}(mn^2)$. Since the number of ground truth regions is fixed and limited, we take n as the number of ground truth regions and m as the number of clusters.

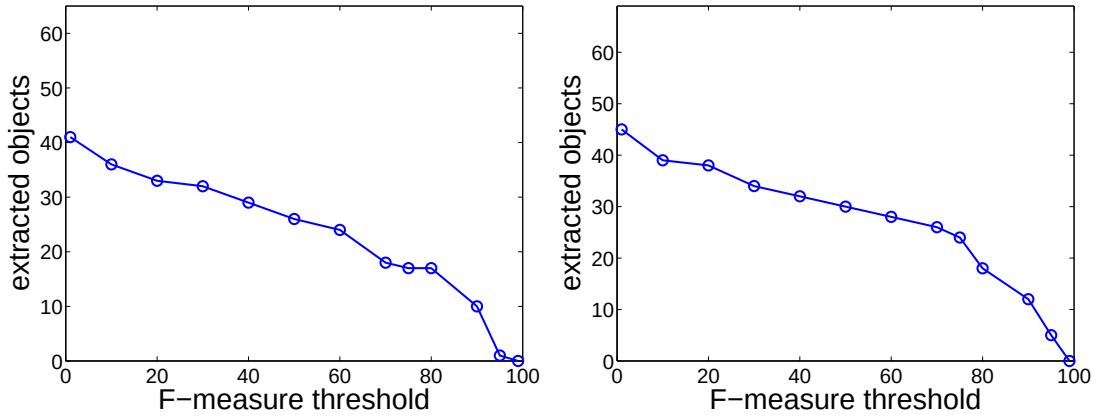


Figure 9.2: Justification for the F-measure threshold of 75% with the method in [OMB14]. LEFT: Training set. RIGHT: Test set. Results are obtained with the sparse method and trajectory sampling 8. With a larger threshold, only very few objects pass the criterion, with a smaller threshold, regions are allowed to be too inaccurate for being considered a meaningful object region; see also Figure 9.3.

$(96.4\%, 90.2\%, 93.2\%)$ $(92.3\%, 63.9\%, 75.6\%)$

$(99.3\%, 36.6\%, 53.5\%)$ $(99.2\%, 28.1\%, 43.8\%)$

Figure 9.3: Qualitative justification for the F-measure threshold of 75%. Result for the first 10 frames and evaluation only on the first frame. FROM LEFT TO RIGHT, TOP TO BOTTOM: Dense result based on a trajectory sampling of 4 overlayed to the first frame of the sequence goats01, lion01, meerkats01, and cats02 with precision, recall, F-measure of the yellow region assignment. The F-measure for the lion is just above the threshold. Only the heads of the meerkat and cat are covered. Consequently, their F-measures are well below the threshold.

Chapter 10

Higher order motion models and spectral clustering

The pairwise analysis of point trajectories in [BM10] is one among the main drawbacks. It work only because locally every motion can be approximated by translations. Therefore, the method relies on a rather dense sampling of trajectories when considering such motions. However, this inaccuracy in the modeling step of the method introduces a systematic error. Motions like scaling or rotation are always penalized, i.e., trajectories obeying such a motion do not naturally fall into the same cluster. When the camera is zooming into the scene, trajectories show a divergent motion and a comparison of the translational portion implies a splitting of the background. Visually, the difference can be inspected best in the eigenvectors used in the spectral clustering. See Figure 10.1

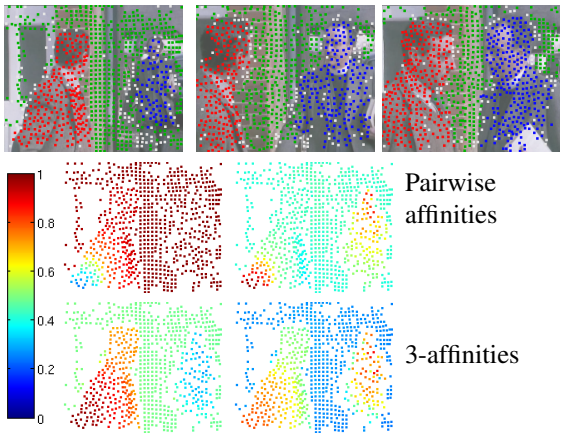


Figure 10.1: TOP ROW: A segmentation of point trajectories in a video shot using [OB12], which is presented in this chapter. SECOND AND THIRD ROW: Visualization of eigenvectors obtained using spectral clustering. In the second row the underlying affinity matrix compares the translational part of the motion [BM10], and the third row also compares rotation and scaling [OB12]. The eigenvectors of [OB12] are more piecewise constant than those of [BM10], which shows the improved modeling of the motion in [OB12].

Multi-body factorization methods, such as [CK95, YP06, RTVM08, EV13], can incorporate (affine) higher order motion models easily, but they require all trajectories to be roughly of the same length. As motion across many frames leads to significant occlusion/disocclusion phenomena, the factorization framework is restricted to much shorter shots.

In this chapter, we are aiming for the best of both models: we keep the flexible comparison of trajectories of various length, and we allow for higher order motion models that do not penalize rotation and scaling anymore. Our approach can work with motion models of arbitrary order, though in our experiments we have focused on in-plane rotation and scaling, i.e., 2D similarity transformations (without reflections). To uniquely determine the parameters of such a model, two motion vectors are required.

Obviously pairwise affinities are not applicable: a pair of motion vectors always fits a similarity transformation perfectly. A third motion vector is required to decide on the compatibility of a triplet. Rather than pairwise affinities we obtain tertiary affinities. The affinity matrix becomes an affinity tensor and the underlying graph a hypergraph.

One way to cluster the affinity tensor is by approximation with an ordinary weighted graph. Usually, reduction from a hypergraph to an ordinary graph can be interpreted as a projection. The projections applied in practice end up in setting the edge weights to the sum over all corresponding hyperedge weights. We argue that at least in the case of motion segmentation this is not a good projection, because many triplets in the hyperedge set will comprise multiple objects though the considered pair just covers a single object. This can be particularly problematic in the case of small objects. For this reason, we propose a regularized maximum projection which just requires one of the triplets to cover the same object. Since hypergraphs come with a larger computational complexity than ordinary graphs, we also present a subsampling strategy that leads to acceptable computation times while not losing the effect of the higher order model. Once the affinities for the ordinary graph are computed, we apply spectral clustering with spatial regularity as proposed in Section 8.2.2.2. Large parts of this chapter are published in [OB12].

10.1 Related work

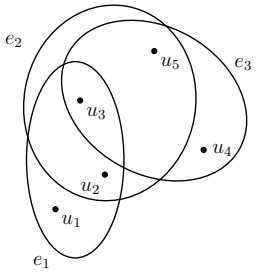
As opposed to VLSI design, where hypergraph partitioning has been used for decades [AK95], hypergraphs have appeared in computer vision only recently [OB12, HLM09, ALZ$^+$05]. One of the first papers is the one by Agarwal et al. [ALZ$^+$05], who analyzed a set of existing hypergraph partitioning methods and showed their application to illumination invariant clustering of faces. Hypergraphs have also become popular in conjunction with inference in higher order MRFs [KKT07, Ish09]. Like here, the goal of using hypergraphs is to consider larger cliques. However, due to the Markov property, vertices are only locally connected by hyperedges, which is in contrast to our method, where hyperedges cover the graph globally. Hypergraph clustering also appeared in bioinformatics [KHT09, THK09] and information retrieval [GKR00, BTC$^+$10].

For clustering a hypergraph, there are basically two different approaches. (1) tensor methods that generalize matrix spectral clustering [SZH06, RP09, LS12] and (2) approximating a hypergraph with an ordinary weighted graph [Gov05, CL09, ZHS07, ALZ$^+$05]. While methods of type (1) are only applicable to k-uniform hypergraphs, i.e., hypergraphs where each edge contains exactly k vertices, methods of type (2) are only approximations to the hypergraph spectral clustering. In [HSJR13], the differences between both methods in terms of the cut propery is discussed. Based on the Lovasz extension of the hypergraph cut the total variation and other regularization functionals on a hypergraph are introduced, which are important for semi-supervised learning methods.

[ALZ$^+$05] needs to be mentioned separately again as a method of type (2). They discuss general approximation (with a p-norm) of the hypergraph clustering, which also comprises, in a limiting case ($p \to \infty$), the maximum projection that we propose in this chapter. Previously, works are mainly concerned with a general hypergraph formulation and consider applications as a proof of concept. In contrast, our method comes with a specific application that requires hypergraphs, and we argue why the maximum projection fits this application much better than the common average projection.

10.2 Hypergraph modeling

A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ consists of a *vertex* set V and a *hyperedge* set $\mathcal{E}$ of subsets of V. The number of vertices and hyperedges is finite and given by the *cardinality* $|\cdot|$ of the respective set. A vertex $v \in V$



$V \setminus \mathcal{E}$	e_1	e_2	e_3
u_1	1	0	0
u_2	1	1	0
u_3	1	1	1
u_4	0	0	1
u_5	0	1	1

Figure 10.2: Example of a hypergraph (left) and its incidence relationships h (right), e.g., $h(u_1, e_1) = 1$ and $h(u_4, e_2) = 0$. A hypergraph describes the relation among an arbitrary number of vertices, here triples.

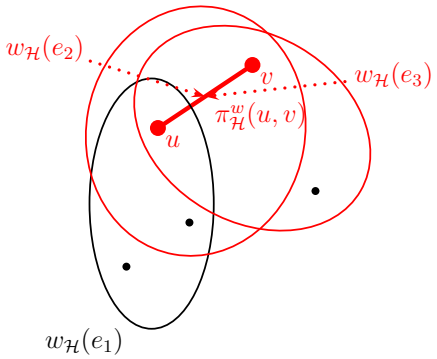


Figure 10.3: A hypergraph which shows the hyperedge weights (ellipses) that are projected to the edge (line) between the vertices u and v. The hyperedges incident with both vertices contribute to the weight of the projected edge.

and a hyperedge $e \in \mathcal{E}$ are called *incident* if $v \in e$. We represent this relationship by an indicator function $h\colon V \times \mathcal{E} \to \{0, 1\}$. Figure 10.2 shows an example.

We focus on *undirected*, *weighted*, k*-uniform* hypergraphs, i.e., the ordering of vertices in a hyperedge does not matter, each hyperedge e is assigned a weight $w_{\mathcal{H}}(e) \in \mathbb{R}_0^+$ and the number of vertices in a hyperedge (the *degree*) $\delta_{\mathcal{H}}(e) := |e| = \sum_{v \in V} h(v, e) = k$ is constant. In the special case of $k = 2$, hypergraphs become ordinary graphs. As the weights of 2-hypergraphs are represented by an $\mathbb{R}^{|V|^2} := \mathbb{R}^{|V| \times |V|}$ affinity matrix, weights of k-uniform hypergraphs are represented by a $\mathbb{R}^{|V|^k} := \mathbb{R}^{|V| \times \ldots \times |V|}$ affinity tensor. For undirected hypergraphs the affinity tensor is a symmetric k-order tensor with non-negative entries, where *symmetry* means that the entries of all index permutations are the same.

10.2.1 Projecting the hypergraph to its primal graph

To partition a hypergraph via spectral clustering, we project it to an ordinary graph. Therefore, we define the *primal graph* that consists of the same vertex set as the hypergraph but its edge set connects each pair of vertices incident with the same hyperedge.

Define the *projection operator* $\pi^w_{\mathcal{H}}\colon V \times V \to \mathbb{R}_0^+$, which assigns to each pair of vertices a weight by projecting the weights of all hyperedges incident with both of the vertices; see Figure 10.3. Appropriate projection operators are positive, symmetric, and monotone [ALZ+05].

Based on such a projection operator, we can write the projection from an affinity tensor to an affinity matrix as

$$\begin{aligned} \Pi_{\mathcal{H}}\colon \mathbb{R}^{|V|^k} &\to \mathbb{R}^{|V| \times |V|}, \\ w_{\mathcal{H}} &\mapsto \Pi_{\mathcal{H}}(w_{\mathcal{H}}) := (\pi^w_{\mathcal{H}}(u, v))_{u,v \in V}. \end{aligned} \tag{10.1}$$

The common projection is via summation:

$$\pi^w_{\mathcal{H}}(u, v) = \sum_{e \in \mathcal{E}} \frac{w_{\mathcal{H}}(e)}{\delta_{\mathcal{H}}(e)} h(u, e) h(v, e). \tag{10.2}$$

It has been proposed, for instance, in [ALZ+05, ZHS07]. Let $H := (h(u, e))_{u \in V, e \in \mathcal{E}} \in \mathbb{R}^{|V| \times |\mathcal{E}|}$ be the incidence matrix and $D_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ the hyperedge degree diagonal matrix. Then (10.2) can be written

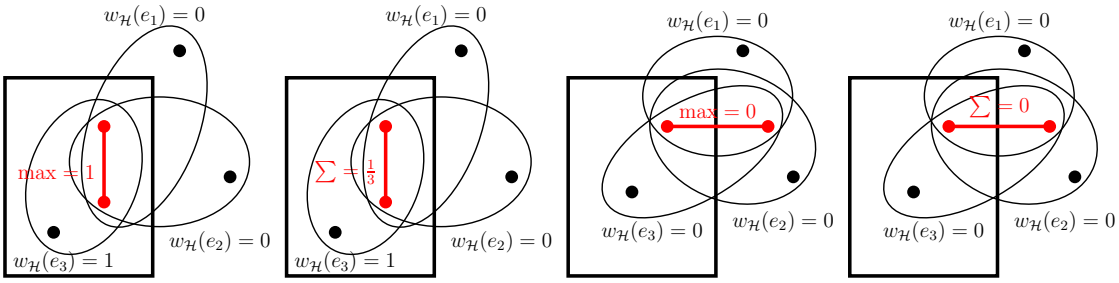


Figure 10.4: LEFT PAIR: max-projection vs. $\sum$-projection if the edge covers a single object. One additional trajectory on the object indicates that the motion model is consistent. The $\sum$-projection yields the average affinity of all hyperedges, which is way too low. RIGHT PAIR: max-projection vs. $\sum$-projection if the edge covers two different objects. Both projections yield the optimal affinity.

as $HWD_{\mathcal{E}}^{-1}H^{\top}$, which is the affinity matrix in [ZHS07]. This means that the hypergraph Laplacian in [ZHS07] fits in this projection framework as a special case [ABB06].

[HSJR13] show shown that construction an ordinary graph with these weights implies that each cut has the same value as the corresponding "true" hypergraph cut. For the proof and more details, we refer to [HSJR13].

10.2.2 max-affinity projections for motion segmentation

In [ALZ+05] a class of functions parameterized by $p \in \mathbb{R}^+$ is proposed. For $p = 1$ this corresponds to the $\sum$-projection in (10.2). In the following, we will argue that a larger p, in particular $p \to \infty$ that corresponds to the maximum operator is more appropriate for motion segmentation.

In motion segmentation, the hypergraph's vertices are point trajectories and the hyperedges are k-tuples of these trajectories. For each k-tuple of trajectories we define an affinity based on their motion similarity; details will follow in Section 10.3. According to the definition in (10.1), the max-projection reads

$$\pi_{\mathcal{H}}^{w}(u,v) = \underset{\substack{w_{\mathcal{H}}(e) \\ u,v \in e \in \mathcal{E}}}{\operatorname{argmax}} w_{\mathcal{H}}(e)\, l(e), \tag{10.3}$$

where we include a weight $l(e)$ that is the number of common frames of all trajectories in e. This weighting treats longer, strongly overlapping trajectories as more reliable.

In Figure 10.4 let us analyze two important cases in the motion segmentation scenario: (1) both vertices of a pairwise edge lie within a single object; (2) they cover two different objects. For simplicity, we focus on hyperedges of degree 3. The first case reveals the advantage of projecting with the max-operator: a single third vertex in the object is sufficient for a high affinity, whereas the $\sum$-projection leads to a bad compromise.

In the second case both vertices of the pairwise edge belong to different objects and we want the affinity to be low. The motion model suggested by the two vertices is not compatible with one of two object motions and so there is no hyperedge with a high affinity for this pair. Both projections lead to the correct affinity.

These considerations are only valid in the noise-free case and if the motion subspaces do not overlap. In contrast to the sum, the maximum operator is unstable. A single outlier or a non-empty cut of the two motion subspaces will spoil the result in case (2). Both problems exist in practice. For this reason, we must regularize the max-operator by adding the following condition: if the projection yields a high

affinity, but 90% of the considered hyperedges have 0 affinity, we assign 0 affinity to the pairwise edge. This condition makes the use of the $\max$-operator stable enough for our motion segmentation task. Other types of regularization are conceivable as well, e.g., an L_p-norm with a finite, sufficiently large p.

10.3 Computing hyperedge affinities

Let us briefly recap the motion segmentation situation required here. Let c_i, $i = 1, \ldots, n$, be n trajectories in a video with M frames. Most trajectories do not cover all M frames due to occlusion/disocclusion. If a trajectory c_i exists at a frame $t \in \{1, \ldots, M\}$, it comes with pixel coordinates $(x(t), y(t))^\top$ at this frame. Consider a k-tuple e of trajectories $c_{i_1}, \ldots, c_{i_k}$. For each such tuple we compute a distance $d\colon \mathcal{E} \to \mathbb{R}_0^+$, which is converted into an affinity via

$$w_{\mathcal{H}}(e) := \exp\left(-\lambda d(e)\right) \tag{10.4}$$

with $\lambda = 0.1$.

10.3.1 Computing 3-distances

In our experiments we focus on 3-tuples. Like in [BM10], we compose the distance d for each triplet (c_i, c_j, c_k), $i, j, k \in \{1, \ldots, n\}$ of the distances $d^{(t)}$ in all *common frames* t, i.e., frames in which all three trajectories are visible. For a fixed $t \in \{1, \ldots, M\}$ we estimate the error of the underlying motion model according to the change from frame t to $t' := t + 8$. If less than 8 common frames are available we restrict the computation to the common frames.

The incentive of considering three trajectories at a time is to have Euclidean translations, rotations, and scalings without penalty. Formally, these movements can be described by the group of special similarity transformations $\mathrm{SSim}(2)$. In contrast to the group of similarity transformations $\mathrm{Sim}(2)$ it excludes reflections.

Let c_i, c_j be two of three trajectories. The motion model $\mathcal{T}_{i,j}(t) \in \mathrm{SSim}(2)$ is described by a scaling parameter s, a rotation matrix R_α, and the translation vector $\mathbf{v} := (v_1, v_2)^\top$. These parameters can be computed uniquely from the coordinates at frames t and t' as follows:

$$\begin{aligned} s &= \tfrac{\|c_i(t') - c_j(t')\|}{\|c_i(t) - c_j(t)\|} \\ \alpha &= \arccos\left(\tfrac{\left(c_i(t') - c_j(t')\right)^\top (c_i(t) - c_j(t))}{\|c_i(t') - c_j(t')\| \, \|c_i(t) - c_j(t)\|}\right) \\ \mathbf{v} &= \tfrac{1}{2}\left(\left(c_i(t') + c_j(t')\right) - sR_\alpha\left(c_i(t) + c_j(t)\right)\right). \end{aligned} \tag{10.5}$$

We can test how well the third trajectory c_k fits this transformation by the ℓ_2-distance $\|\mathcal{T}_{i,j} c_k(t) - c_k(t')\|$.

Obviously, the motion model could be computed also from c_i, c_k or from c_j, c_k. As illustrated on the right in Figure 10.5 the choice of the pair has a large impact on the distance. One could also consider estimating the optimum model from all three trajectories in the least-squares sense, as shown on the left in Figure 10.5. However, this reduces the distance of an incompatible triplet such that it cannot be distinguished from noise in a compatible one anymore. For this reason, we pick the maximum distance among all 2-tuples $(u, v) \in \{(i, j), (i, k), (j, k)\}$ with the third trajectory c_w, $w \in \{i, j, k\} \setminus \{u, v\}$:

$$d^{(t)}(i, j, k) := \max d^{(t)}_{\text{ratio}}(u, v) \, \|\mathcal{T}_{u,v} c_w(t) - c_w(t')\|. \tag{10.6}$$

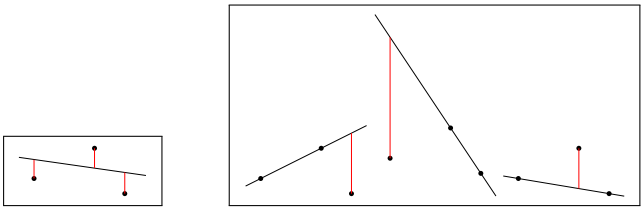


Figure 10.5: LEFT: Fitting a linear model to three points via least squares. Even though the points do not fit a line, the total error (in red) is small. RIGHT: A line is fit to all pairs of points and the error is measured based on how well the third point fits the line. The maximum discrepancy among all tuples clearly indicates that the points do not fit a common model.

If the three trajectories do *not* fit the same model, it will be very large leading to a clear separation of motion clusters. The weight

$$d^{(t)}_{\text{ratio}}(i,j) := \left(\frac{1}{2}\left(\frac{\|c_i(t)-c_j(t)\|}{\|c_i(t)-c_k(t)\|} + \frac{\|c_i(t)-c_j(t)\|}{\|c_j(t)-c_k(t)\|}\right)\right)^{\frac{1}{4}}. \tag{10.7}$$

is introduced to avoid numerical problems. When a motion model $\mathcal{T}_{i,j}(t)$ is estimated based on very nearby trajectories, small numerical errors can cause large errors at the location of distant trajectories.

Like in [BM10] we normalize distances with the optical flow variance in the image and weight them by the spatial distance of trajectories. For the final distance of the 3-tuple we take the maximum distance over all common frames $d(i,j,k) = \max_t d^{(t)}(i,j,k)$.

10.3.2 Higher order affinities

The above framework extends easily to more general motion models. For instance, distances based on an affine motion model could be computed from 4-tuples of trajectories. While this fits 3D rigid motions even better, it also further increases the computational costs. With 3-affinities we have cubic complexity $\mathcal{O}(3n^3)$. The complexity of a k-tuple is $\mathcal{O}(n^k k)$, where the factor k is due to the maximum in (10.6). In the next section, we present a sampling strategy that reduces the complexity of 3-affinities to $\mathcal{O}(n^2)$ without significant degradation compared to the full model. While k-affinities with $k > 3$ are an interesting future option, they also come with other issues. For example, as 3D rotations lead to self-occlusion, trajectories are usually too short for a more complex model to show advantages. This is why we consider only 3-affinities in the remainder of this chapter.

10.4 Sampling strategy

We propose a combination of deterministic and random sampling to reduce the number of hyperedges to be considered, which makes the approach tractable in case of large numbers of trajectories. For each pairwise edge, we sample hyperedges comprising both vertices of the edge and one additional vertex. For both vertices in each pairwise edge, we take the 12 spatially nearest neighbours and, additional, 30 vertices randomly as third vertex. It is worth noting that we sample only over the third vertex, whereas the graph is spanned globally over all trajectories, which is in contrast to MRF approaches. The mixture of deterministic and random sampling ensures finding enough relevant triplets.

We verified this in a synthetic experiment shown in Figure 10.6. The results for the full hypergraph, where all possible triplets are considered, and two sampling strategies are reported in Table 10.1.

The evaluation shows that sampling degrades the accuracy only a little when some objects are much smaller than others. In some special cases, sampling is even advantageous. If the smaller objects cover

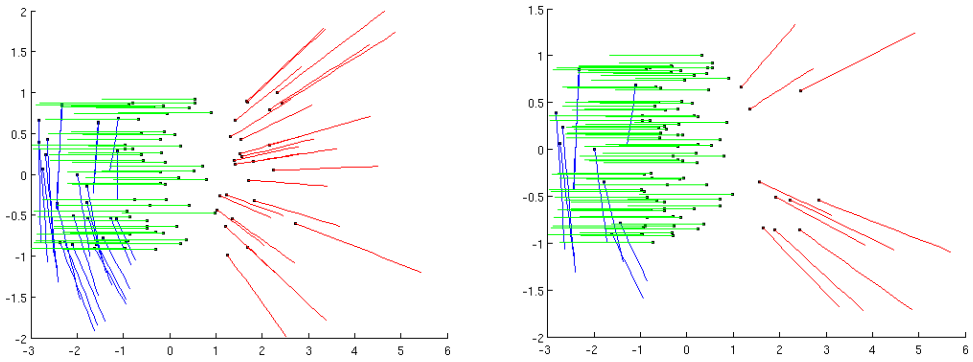


Figure 10.6: Two synthetic experiments with 100 trajectories and 3 regions of different motion and size (rotation, translation, scaling). In the experiment we reduced the size of the left and right region successively to 25, 20, . . ., 5, 4, and 3 trajectories simulating the effect of small objects and a large background.

	25	20	15	10	5	4	3
full	100	100	100	84	96	100	99
rnd	100	100	100	99.6	97.8	94.8	96
rnd+nn	100	100	100	100	97.2	97.6	96.3

Table 10.1: Results of the experiment in Figure 10.6. The number of trajectories in the left and the right cluster is given in the first row. The table shows the percentage of correctly clustered trajectories for the full graph (full), random sampling (rnd) with 12 samples, and a combination (rnd+nn) of 4 random samples and 4 nearest neighbors at a time. The results on rnd and rnd+nn are averaged over 10 evaluations. Sampling does not affect the accuracy much, even for small objects, but reduces the complexity considerably.

exactly 10% of all trajectories, 90% of all triplets comprise a vertex outside the object and, thus, yield 0 affinity. Since our regularized max-projection treats this situation erroneously as outlier, it assigns 0 affinity to the pairwise edge in case of the full graph. For the random sampling, it is sufficient to have 2 ($> \frac{12}{10}$) triplets covering the object. Since only a small subset of the pairwise edges needs high affinities, sampling such a subset is very likely. Of course this probability decreases as the object shrinks. In order to improve the performance in object segmentation (cf. Table 10.2), where certain compactness assumptions hold, we complement the random sampling by a deterministic nearest neighbor sampling.

10.5 Evaluation

We compare to multi-body factorization [RTVM08] and the translational motion model from [BM10] on the benchmark dataset introduced in [BM10]. We used the evaluation code provided with the benchmark. Results are shown in Table 10.2. The benchmark and the evaluation method are reviewed in Section 9.2, where we introduce an extension to it. The evaluation on the larger dataset is postponed to Chapter 12, where several other methods are included in the evaluation.

We use the same tracker as in [BM10] which allows to adjust the density. We run the tracker with 4- and 8-spacing, meaning that, the tracker samples only every 4th or 8th pixel, respectively, in each direction.

Performance differences become clearly visible when considering all frames. As already shown by [BM10], multi-body factorization cannot deal with large occlusions, and this shows in the numbers. Comparing the pairwise affinities [BM10] to our higher order model reveals a significant improvement of 50% in the overall error. The higher quality also shows in more objects being extracted correctly (defined

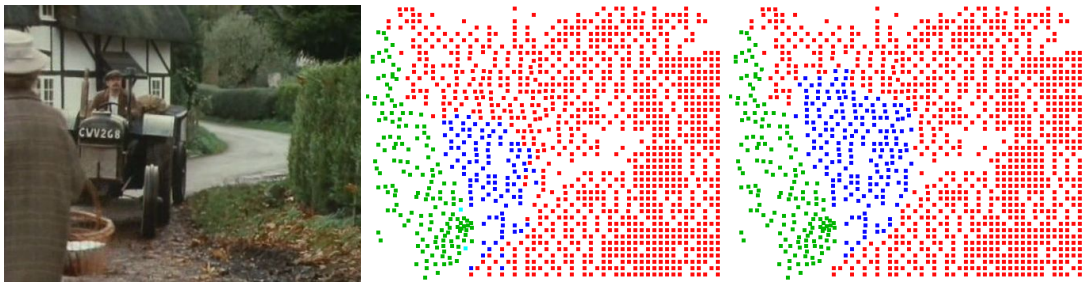


Figure 10.7: FROM LEFT TO RIGHT: Frame 46 of marple8, result with pairwise affinities, triplet result.

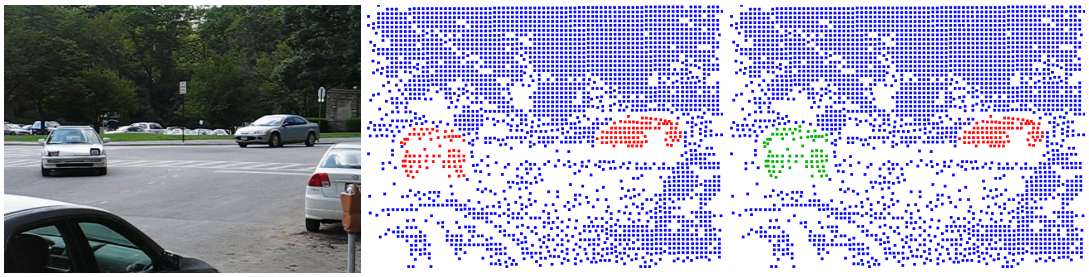


Figure 10.8: FROM LEFT TO RIGHT: Frame 21 of cars5, result with pairwise affinities, triplet result.

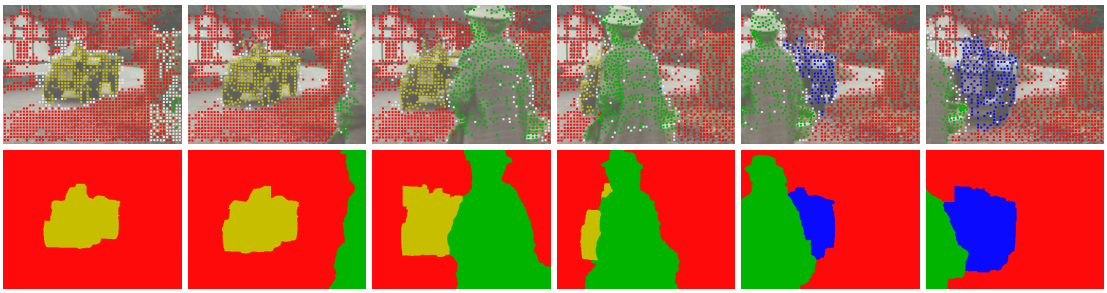


Figure 10.9: TOP ROW: Result on *marple8* obtained with the proposed method and a spatial subsampling of 8. BOTTOM ROW: Dense interpolation of this result using [OB11]; see Chapter 11.

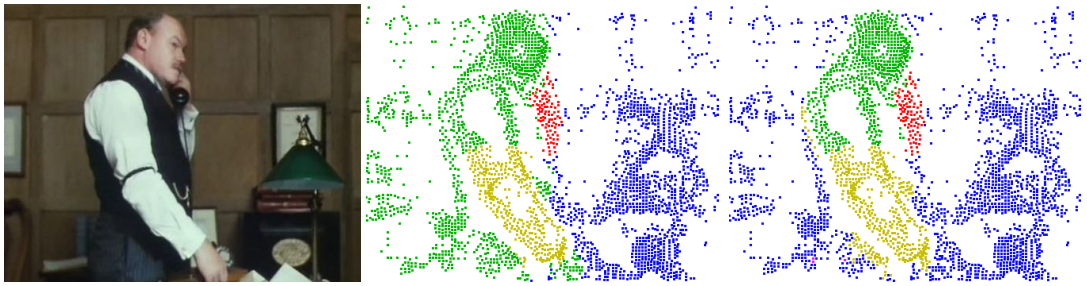


Figure 10.10: FROM LEFT TO RIGHT: Frame 65 of marple13, result with $\sum$-projection, our result (max-projection).

	Density	overall error	average error	over-segmentation	extracted objects
ALC corrupted [RTVM08]	0.99%	5.32%	52.76%	0.10	15
ALC incomplete* [RTVM08]	3.29%	14.93%	43.14%	18.80	5
pairwise affinities [BM10]	3.31%	6.52%	27.29%	2.12	29
pairwise affinities* [BM10]	3.28%	6.59%	26.69%	1.44	27
$\sum$-projection* (rnd+nn)	3.22%	5.36%	20.08%	2.08	31
max-projection* (rnd+nn)	**3.22%**	**4.48%**	**22.34%**	**1.84**	**31**
pairwise affinities [BM10]	0.79%	6.78%	25.48%	1.73	30
$\sum$-projection (rnd+nn)	0.78%	6.12%	22.71%	2.31	30
max-projection (rnd)	0.77%	7.32%	32.95%	1.92	22
max-projection (rnd+nn)	**0.78%**	**4.33%**	**21.96%**	**1.58**	**34**

Table 10.2: Evaluation on the Berkeley motion segmentation benchmark (see Chapter 9) using all available frame. Entries marked with "*" are evaluated without the sequence *marple7*. The upper part of the table reports on a 4-spacing, the lower part on an 8-spacing. (rnd) refers to a purely randomized sampling using 54 samples. (rnd+nn) uses 30 random samples and for each vertex the 12 spatial nearest neighbors.

by [BM10] as regions with less than 10% error minus the background region). Figures 10.7 and 10.8 show a qualitative comparison.

We also compared the proposed max-projection to the $\sum$-projection that is used by previous works on hypergraph partitioning, e.g., in [ALZ+05]. The $\sum$-projection performs better than just the pairwise affinities, but only the max-projection can fully exploit the higher order affinities. In fact, most of the increase in performance is not due to the use of hypergraphs as such, but due to hypergraphs in conjunction with the right projection method. Figure 10.10 shows an advantage of our max-projection over the $\sum$-projection with respect to the leakage problem explained in Figure 10.4.

Figure 10.9 shows our result for marple8 and a dense segmentation obtained using the code from [OB11]; This method is presented in Section 11.4.2.

Thanks to the sampling, the computational complexity is still in $\mathcal{O}(n^2)$, yet practical computation times increase over the translational model. While clustering the whole sequence of *cars1* with 4850 trajectories runs in 50s on a single core with pairwise affinities, 3-affinities require 48 minutes. The computation of affinities can be perfectly parallelized on the GPU though. With an expected speedup of about $50\times$ this would result in approximately 3s per frame, which is well tractable also in a large scale task.

Other real world examples. The Berkeley motion segmentation benchmark focuses mainly on translational motion. Hence, we collected some additional videos with more complex motion and compare our method to [BM10].

The advantage of higher order motion models becomes very clear in Figure 10.11. It is not possible to describe the rotation of the windmill sails with pairwise affinities. Consequently, trajectories are either treated as outliers and are removed completely, or they lead to over-segmentation. The higher order model can handle this example easily. The same effect can be seen in Figure 10.12. Although the 3D rotation of the monkey is not without penalty in our 3-affinity model, the penalty with pairwise affinities is even larger. Figure 10.13 shows a very difficult sequence with camera zoom. Some ducks move in a very similar manner and there is strong articulation. Pairwise affinities lead to a single cluster. In contrast, 3-affinities can capture the moving ducks quite well.

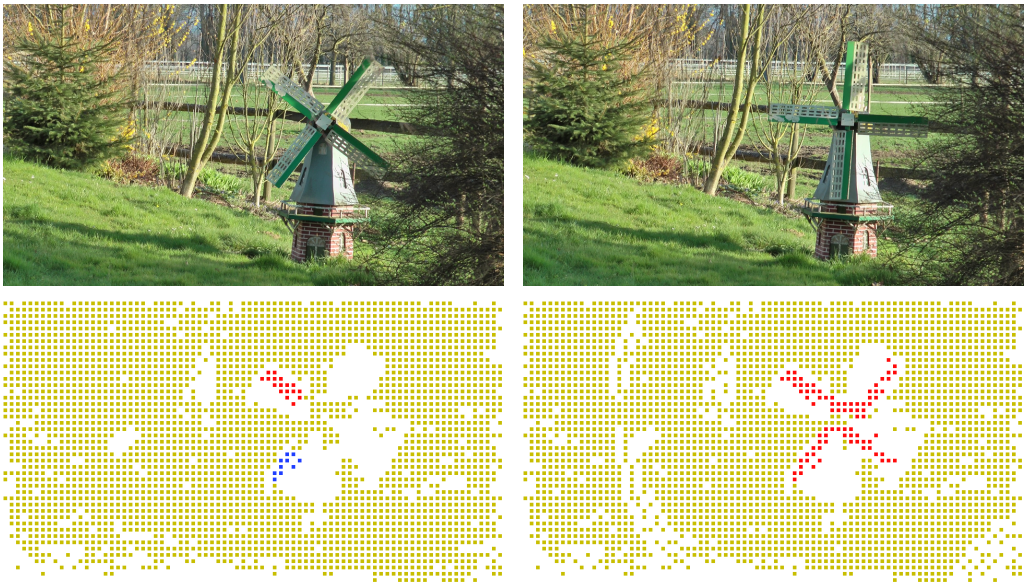


Figure 10.11: UPPER ROW: Windmill with rotating sails. BOTTOM LEFT: Result with pairwise affinities [BM10]. BOTTOM RIGHT: Our result with 3-affinities.

Figure 10.12: FIRST ROW: Monkey that lies in the beginning, stands up, and turns to the other side. SECOND ROW: Result with pairwise affinities [BM10] for the first and last frame. THIRD ROW: Our result with 3-affinities. This examples shows that 3-affinities also better fit 3D rotations than pairwise affinities, though not explicitly modeled.

Figure 10.13: FIRST AND SECOND ROW: A family of ducks passing the camera. While they are moving away, the camera starts zooming on them. Finally the mother vanishes behind a green box. Pairwise affinities cannot model the zooming motion of the camera properly. This results in a single cluster comprising the ducks and the background. OTHER ROWS: Our 3-affinities are invariant to scaling. The ducks can be separated from the background. The method even separates most of the single ducks.

Chapter 11

Clustering the affinity graph

Figure 11.1: Clustering of the affinity graph and propagating the segmentation to each pixel.

While Chapter 10 focuses on the improvement of the similarity measure between trajectories, this chapter deals with the actual clustering of trajectories. The starting point is an affinity graph describing the relations of the sparse set of trajectories like in [BM10] or Chapter 10. The goal of this chapter is a dense segmentation of a video sequence. We proceed in two steps: (a) clustering point trajectories and (b) propagating the segmentation frame-by-frame to each pixel.

Although tracking could be made dense by densely sampling and propagating trajectories, fundamental problems avoid the success of this approach. Optical flow is not reliable in homogeneous areas and, even more severe, computing spectral clustering on a dense set of trajectories is too slow.

(a) The clustering of point trajectories is considered as the problem of clustering an undirected weighted graph, where the nodes represent the trajectories and edges are weighted by the estimated affinities. Considering the noise-free case, i.e., the motion analysis yields a perfect fit for trajectories on the same object and 0 for edges between different objects, spectral clustering would generate piecewise constant eigenvectors. This situation would make the clustering trivial as the objects could be obtained by thresholding the eigenvectors. In practice there are always noise and estimation errors. This results in eigenvectors being piecewise smooth instead of piecewise constant and more advanced clustering methods are required. The image in the center of Figure 11.1 shows clustered point trajectories represented in a certain frame of the video sequence.

(b) The second step is that of going from the image in the center of Figure 11.1 to the right one. This interpolation task seems to be easy at first glance. Humans can easily do it. However, a closer look at the labeled trajectories reveals several challenges. The point trajectories do not cover some of the most

critical areas, especially near object boundaries. Those trajectories that exist near object boundaries are often assigned wrong labels because the underlying optical flow is imprecise in occlusion areas. Finally, in large homogeneous areas there is hardly any label information. The segmentation approach presented in this chapter will exploit cues that are complementary to motion, namely color and texture, to propagate the label information to the missing areas.

After discussing some related work, we introduce a general model for clustering and segmentation. Although it is stated in and motivated from continuous quantities, Section 11.3 shows that a direct translation into discrete terms yields a model for clustering trajectories. Trajectories are assigned labels based on the eigenvectors arising from spectral clustering. Considering the clustered trajectories in each frame and seeking for assigning a label to each pixel is subject of Section 11.4, where two different approaches are proposed. Large parts of this chapter are published in [OB11, OMB14].

11.1 Related work

The difficulty in the setting of trajectory clustering is that there are only pairwise relationships. This problem is different to semi-supervised learning where some labels for some nodes are known, i.e., there is an unary term besides the pairwise term. Nevertheless, the problem also requires some regularization in order to avoid a trivial assignment of labels. This leads to the so-called balanced cuts like normalized cut, minimum cut, or Cheeger cut. The regularization is imposed by certain ratios between the clusters [DH73, PSL90, HK91, SM00]. Unfortunately the balanced k-cut problem is NP-hard [SM00] and therefore only certain relaxations are solved. We refer the interested reader to [RMH14] and references therein, which is a reference that shows recent advances for continuous relaxations of balanced cuts.

One of the most famous and frequently used relaxations is spectral clustering [SM00], which relaxes the normalized cut. A good tutorial can be found in [vL07] and a standard reference for advanced details is [Chu97]. Spectral clustering seeks for the eigenvectors of the graph Laplacian, which is constructed from the affinities between the trajectories in our context. These eigenvectors provide a feature for each node in a graph. Based on these features unary potentials can be derived. In standard spectral clustering, k-means clustering on these features is used. However, as the number of clusters must be known a priori, we need a more advanced clustering method. [BM10] proposes to minimize a certain energy that serves as objective for the selection of the right number of clusters, which we have seen in Section 8.2.2.2. For minimizing this energy, which also appears in this chapter, a hierarchical k-means clustering technique is proposed in [BM10]. However, at this stage many other methods could be used to cluster the feature vectors, e.g. agglomerative and divisive clustering algorithms [JD88], Markov Random Fields (MRFs) [GG84], or variational methods [MS89, BZ87]. In this chapter, we consider a model based on MRFs. Indeed, if we fix a certain number of labels the problem that is considered for clustering trajectories in this chapter becomes a multi-label MRF, which we solve using Fast-PD [KT07, KTP08].

The problem of spreading out the label information from the sparse set of trajectories is related to interactive segmentation, where the user draws a few scribbles into the image and the approach propagates these labels to the non-marked areas. Several techniques based on graph cuts [BFL06], random walks [Gra06], and intermediate settings [SG07] have been proposed. The latest techniques are built upon variational convex relaxation methods [UPT$^+$08, PCCB09, LBS09, NBRC10], which avoid the discretization artifacts typical for classical MRFs defined on a graph.

Interactive segmentation is different to the problem considered here in two ways. First, the labels are generated by an unsupervised approach and are likely to be erroneous, whereas interactive segmentation relies on the correctness of the user's input. This means, we do not have an interpolation but an approximation problem. For this reason, we do not follow the typical approach of estimating appearance statistics

from the annotated areas combined with a typical region based segmentation. Second, user annotation provides dense finite annotation areas, whereas trajectory labels constitute single points spread over the image. It is not immediate that a variational model acting on such infinitesimally small labels makes sense in the continuous limit.

Our model is also related to image compression with anisotropic diffusion [GWW$^+$08], where only a small set of pixel values is kept and the original image is sought to be restored by running a diffusion process on this sparse representation.

On the task of dense motion segmentation, there are many recent works that produce over-segmentations using superpixels, label propagation by optical flow, or other clustering methods [BT09a, GKHE10, VAPM10, LASL11]. These over-segmentations do not provide object regions. User interactive video segmentation methods can avoid over-segmentation, but are no longer unsupervised [BS07, PMC09].

11.2 Multi-label segmentation

Let us first describe the underlying fundamental multi-label segmentation model for all methods in this chapter: the Potts model. The presentation is inspired by [CCP12], however, we do not seek for the most general formulation. The task of finding such a multi-label segmentation is a minimal partition problem. The objective is to find a partitioning of the image domain $\Omega \subset \mathbb{R}^2$ (assume open and bounded) into disjoint regions $E_1, \dots, E_K \subset \Omega$, such that the region interface length Per and the cost for a weighting function $f \in L^2(\Omega, \mathbb{R}^K_+)$, i.e. $f = (f_1, \dots, f_K) \colon \Omega \to \mathbb{R}^K_+$ is Lebesgue square-integrable, is minimized. For a precise definition of Per, we refer the interested reader to [CCP12, Sec. 2.2]. The generic Potts energy reads

$$\begin{aligned} &\min_{E_1,\dots,E_K} \frac{1}{2} \sum_{k=1}^{K} \mathrm{Per}(E_k; \Omega) + \sum_{k=1}^{K} \int_{\mathrm{E_k}} f_k(x)\, dx \\ &\text{s.t.} \bigcup_{k=1}^{K} E_k = \Omega, \quad E_k \cap E_{k'} = \varnothing,\ \forall k \neq k'. \end{aligned} \tag{11.1}$$

The constraints are meant up to Lebesgue-negligible sets. Unfortunately, the minimization of this energy is NP-hard. However, there are relaxations of the energy, which lead to a tractable convex problem. A detailed discussion about different relaxations for the minimal partition problem can be found in [CCP12]. Here, we only briefly derive the most common relaxation that allows for efficient optimization. In order to do so for each $k = 1, \dots, K$ the set E_k is represented by its characteristic function $u_k = \chi_{E_k}$, which is 1 inside E_k and 0 outside. These characteristic functions are collected in a single label function $u = (u_1, \dots, u_K) \colon \Omega \to \{0,1\}^K$. We assume that $u \in BV(\Omega, \{0,1\}^K)$ lies in the space of functions with finite bounded variation and $\sum_{k=1}^K u_k(x) = 1$ holds for almost every (a.e.) $x \in \Omega$. These functions allow for measuring the perimeter (or total interface) using the total variation. Recalling that the total variation (TV) of a characteristic function $\mathrm{TV}(u_k)$ is a representation for the contour length of region E_k we obtain the following classical relaxation of the Potts energy

$$\begin{aligned} &\min_{u \in L^2(\Omega, \mathbb{R}^K_+)} \mathrm{TV}(u) + \sum_{k=1}^{K} \int_{\Omega} u_k(x) f_k(x)\, dx \\ &\text{s.t. } u_k(x) \in \{0,1\}, \quad \forall k, \quad \sum_{k=1}^{K} u_k(x) = 1, \quad \text{a.e. } x \in \Omega\,, \end{aligned} \tag{11.2}$$

where $\mathrm{TV}(u)$ is an appropriate generalization of TV to multi-valued functions (see [CCP12]) that is set to ∞, if $u \in L^2(\Omega, \mathbb{R}_+^K) \setminus BV(\Omega, \{0,1\}^K)$ or if not $\sum_{k=1}^K u_k(x) = 1$ (for almost every $x \in \Omega$). We use $\mathrm{TV}(u) = \sum_k \mathrm{TV}(u_k) = \sum_k \int_\Omega |Du_k|$, where Du_k is the distributional derivative. For $u_k \in C^1(\Omega, \mathbb{R}_+)$, the total variation simplifies to $\int_\Omega |\nabla u_k(x)|\, dx$.

Problem (11.2) is still very difficult and in fact nonconvex. However, a simple relaxation of the range of u_k to its convex envelope $[0,1]$ copes with the remaining difficulties. Now, the relaxed, convex energy can be solved efficiently and, as it is convex, a global optimum u^* is found. Backprojecting the minimizer u^* at each point $x \in \Omega$ to the set $\{0,1\}^K$ under the constraint that only one label is assigned to 1 via

$$\hat{u}_k^*(x) = \begin{cases} 1, & \text{if } k = \mathrm{argmax}_{k'}\{u_{k'}^*(x) | k' = 1, ..., K\} \\ 0, & \text{otherwise,} \end{cases} \tag{11.3}$$

yields an approximate solution $\hat{u}^*$ to the original Potts energy. For the two-label case, the thresholding theorem [CEN06] ensures global optimality for the original energy. In the general multi-label case, the integer solution is not necessarily a global optimum, but there is a computable upper bound [CPKC11].

As a special case of the Potts model there is the piecewise constant Mumford–Shah model[1] [MS85]. It arises when setting

$$f_k(x) = \lambda(I(x) - c_k)^2, \tag{11.4}$$

where $c_k \in \mathbb{R}_+$ is the mean value of region E_k, $\lambda > 0$ weights between penalizing the contour length of each region and the data fidelity given by f, and $I(x)$ is the intensity gray value at pixel $x \in \Omega$.

11.3 Trajectory segmentation

In this section we consider the segmentation or clustering of trajectories. In this case the domain is discrete and the previous Section 11.2 has to be interpreted in the discrete setting. It becomes a multi-label MRF problem on an undirected weighted graph. Although, essentially, the model is the same as in [BM10], which we recapitulated in Section 8.2.2.2, the strategy for solving it is different. For the reader's convenience, we briefly define the model here again and shed light on the relation to Section 11.2.

Let us describe formally the undirected, weighted graph. The nodes represent trajectories and the edge set connects each node A with 12 nodes of the symmetrized set of its spatially nearest neighbors $A' \in \mathcal{N}(A)$. Furthermore, each trajectory or node is associated a feature $\mathbf{v}^A$ that is given by the Ath components of the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$, i.e., $\mathbf{v}^A = (v_1^A, \ldots, v_m^A)^\top \in \mathbb{R}^m$ and v_i^A denotes the Ath component of the ith eigenvector. The eigenvalues corresponding to those eigenvectors are $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$. As already described in (8.5) in the noise free case these eigenvectors would be piecewise constant. However, in practice the eigenvectors are piecewise smooth as shown in Figure 8.5. We seek to choose the assignments $\pi^A \in \{1, ..., K\}$ such that the cartoon Mumford–Shah-like energy

$$E(\pi, K) := \sum_A \sum_{k=1}^K \delta_{\pi^A, k} |\mathbf{v}^A - \mu_k|_\lambda^2 + \nu \sum_A \sum_{B \in \mathcal{N}(A)} \frac{1 - \delta_{\pi^A, \pi^B}}{|\mathbf{v}^A - \mathbf{v}^B|} \tag{11.5}$$

is minimized, where $\nu > 0$, $\mu_k \in \mathbb{R}^m$ denotes the centroid of cluster k and the norm $|\cdot|_\lambda$ is defined, for $\mu \in \mathbb{R}^m$, as

$$|\mathbf{v}^A - \mu|_\lambda^2 := \sum_i (v_i^A - \mu_i)^2 / \lambda_i \,. \tag{11.6}$$

[1] Sometimes called the cartoon limit of the Mumford–Shah model.

Increasing the parameter $\nu > 0$ puts more weight on compact clusters and less over-segmentation. (11.5) is the same minimization problem as in (8.6).

Minimizing (11.5) is difficult. As the centroids are unknown the energy is nonconvex and has many local minima. Therefore, the solution depends on the initialization. We found that the following deterministic initialization strategy works well. We start with an equidistant initial labeling, i.e., for $k \in \{1, \dots, K\}$ all trajectories A_i with index $(k-1)\frac{n}{K} \leq i \leq k\frac{n}{K}$ are initially assigned $\pi^{A_i} = k$, where n is the number of trajectories. In each iteration we update the centroids μ_k and optimize the label assignments. If the centroid is fixed, the problem of finding the assignments can be computed efficiently, e.g., using FastPD [KT07, KTP08].

Let us consider the similarity of the energy (11.5) and the general multi-label segmentation framework in (11.1) when the centroids μ_k are fixed. The integration over the image domain Ω can be transformed into the summation over all nodes A of the graph by considering the integration with respect to the counting measure of the discrete domain of the graph (the nodes). Setting $f_k(A) = |\mathbf{v}^A - \mu_k|^2_\lambda$ relates the data-term of (11.5) to the one in (11.1). Considering the regularization term in (11.5) the similarity to (11.1) comes from the relation of the TV-norm for binary images and the perimeter of the set represented with the binary values. On a discrete domain the TV-norm is the summation over all edges and measuring the term $1-\delta_{\pi^A,\pi^B}$ with 1, if trajectory A and B are assigned to different clusters, and 0 otherwise. While the TV-norm in the continuous domain can measure the length of diagonal "edges" in the 2D-domain with $\sqrt{2}$, this is not possible for a graph, where all edges are measured as 1. This is the metrication error that is usually observed for discrete optimization methods. The weighting of the edges with $1/|\mathbf{v}^A - \mathbf{v}^B|$ can be considered as a different metric for measuring the perimeter of the respective region, i.e., the interface between trajectories A with $\pi^A = k$ and trajectories B with $\pi^B = j$, $j \neq k$. If the feature $\mathbf{v}^A$ of a trajectory is very different to the feature $\mathbf{v}^B$ the weight for measuring the perimeter is small, which attracts region boundaries. Finally, the constraints in (11.1) are automatically fulfilled because each trajectory is assigned a single cluster.

11.4 From sparse to dense labels

11.4.1 The Potts model

We cast the problem of making the sparse set of labels dense as optimization of a Potts model (11.1), or to be more precise the relaxed version (11.2). In the setting here, the weighting function f in the data term is determined by the semi-sparse set of given labels. Defining the set L_k of coordinates $x \in \Omega$ occupied by a trajectory with label k, let $f_k(x) = 0$ if $x \in L_k$ and $f_k(x) = \alpha$ otherwise. Thereby, $\alpha \in \mathbb{R}^+$ is a positive weight, penalizing region $E_{k'}$ to contain $x \in L_k$, where $k \neq k'$. If $\alpha \to \infty$, regions are forced to enclose only points of a single label. As we can expect some erroneous trajectory labels, smaller values for α are used and labels can be corrected, as demonstrated in Figure 11.2. The α values may depend on the confidence in correctness of trajectory labels and on the cardinality of $\bigcup_{k=1}^K L_k$, i.e., the tracking subsampling factors 4, 8, or 16. Accordingly we set α to 200, 500, or 1000.

So far, we neglected the problem that (11.2) is defined using integrals and a pointwise characterization on a Lebesgue-negligible set does not influence the value of the integral. In order to cope with this issue, we assume that labels are given on a certain set of unit Lebesgue measure around $x \in L_k$.

Ideally, jumps in the label function should be located at image edges, i.e., where the gradient of the (sufficiently smooth) image function $I\colon \Omega \to \mathbb{R}^3$ is high. Therefore, the TV-term in (11.2) is replaced by

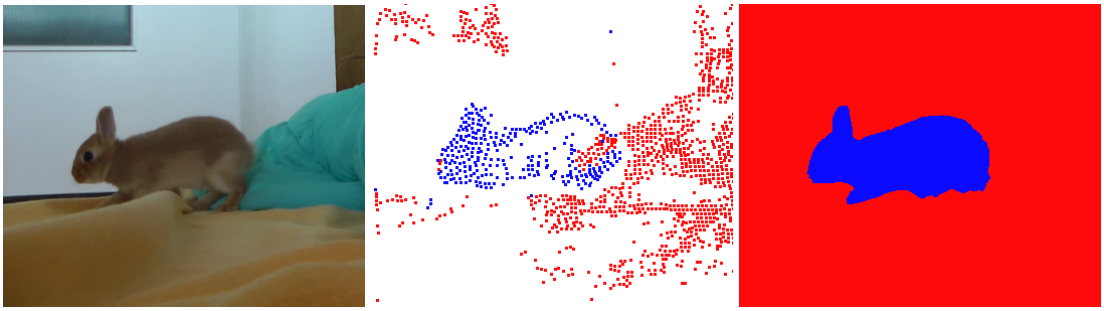


Figure 11.2: Labels given by trajectories are penalized by a certain weight, and, thus, can be corrected in the variational optimization. LEFT: Input image. MIDDLE: Sparse point trajectory labels. RIGHT: Dense segmentation via minimization of a Potts model.

the image-driven weighted TV regularization

$$\mathrm{TV}_g(u) = \sum_{k=1}^{K} \int_\Omega g\,|Du_k| \overset{u_k \in C^1(\Omega,\mathbb{R}_+)}{=} \sum_{k=1}^{K} \int_\Omega g(x)|\nabla u_k(x)|\,dx\,, \tag{11.7}$$

where $g(x) = 1/\sqrt{\sum_{i=1}^{3} |\nabla I_i(x)|^2 + \varepsilon^2}$. The parameter $\varepsilon = 10^{-3}$ serves as a stabilizing parameter for homogeneous areas. This regularizer can be interpreted as ordinary total variation measure in the metric induced by the image as a Riemannian manifold. Numerically we solve this optimization problem using the primal–dual algorithm from Section 2.2.6.

In order to obtain the desired image partitioning, the minimizer u^* of the convex energy (11.2) (with (11.7)) is reprojected to the discrete label space $\{0,1\}^K$ according to (11.3).

11.4.2 A hierarchical approach

Solving the relaxed Potts model (11.2) using a gradient based approach requires the spreading of label information from the sources given by the labeled trajectories to the pixels without label. Particularly in homogeneous regions, where only few pixels are labeled the label information must be transported over long distances. Also noise and unimportant structure hamper the spreading and highly influence the convergence. To overcome this problem, we propose a hierarchical model. The finest level in this hierarchy corresponds to the Potts model from Section 11.4.1. Additional levels make use of superpixels obtained with the approach from [AMFM11]. The representation on the left in Figure 11.3 illustrates the continuous hierarchy. The right part of Figure 11.3 shows the corresponding discrete graph structure for helping readers who prefer to think in discrete terms.

Each level $i = 0, ..., N$, in our variational model represents a sufficiently smooth, Lebesgue integrable function that is partitioned into M^i superpixels $\Omega_m^i, m = 1, ..., M^i$. For $i = 0$ we have the functions u^0 and I^0 as defined for the single-level model. For $i > 0$ we have the corresponding piecewise constant functions u^i and I^i, where $I^i(x) = \frac{1}{|\Omega_m^i|} \int_{\Omega_m^i} I^0(x')dx' \in \mathbb{R}^3$ takes the mean color of the corresponding superpixel Ω_m^i. The idea behind these additional auxiliary functions at coarser levels is to define a propagation process that better adapts to the image structures at multiple scales.

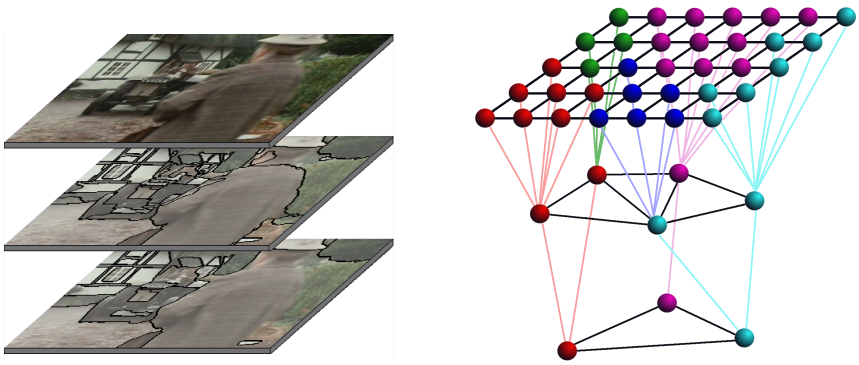


Figure 11.3: Illustration of the multi-level model. LEFT: Continuous model, where each level is a continuous function. Coarser levels are piecewise constant according to their superpixel partitioning. RIGHT: Corresponding discrete graph structure in terms of pixels/superpixels showing the linkage between levels. Our model is in fact *not* a graph, as each level is continuous.

We extend the single-level energy from (11.2) accordingly:

$$\begin{aligned} \min_{u \in L^2(\Omega, \mathbb{R}^{N+1\times K})} E(u) = \min_u & \sum_{k=1}^{K} \int_\Omega u_k^0(x) f_k(x) \rho(x)\, dx + \sum_{i=0}^{N} \mathrm{TV}_{g^i}(u^i) \\ & + \sum_{i=1}^{N} \int_\Omega \bar{g}^i(x) \sum_{k=1}^{K} |u_k^i(x) - u_k^{i-1}(x)|\, dx \\ \text{s.t. } u_k(x) \geq 0, \quad \forall k, \quad & \sum_{k=1}^{K} u_k(x) = 1, \quad \text{a.e. } x \in \Omega\,, \end{aligned} \tag{11.8}$$

where $u := (u_1^0, ..., u_K^0, u_1^1, ..., u_K^1, ..., u_1^N, ..., u_K^N) \in L^2(\Omega, \mathbb{R}^{N+1\times K})$ denotes the label function of the whole hierarchy. The first term is identical to the single-level model, except for an additional weighting function $\rho\colon \Omega \to \mathbb{R}$, which will be explained later. Label sources exist only in the finest level. They propagate their information to the coarser levels via the third term, which connects successive levels. The level diffusivity functions $\bar{g}^i$ have the same meaning as the spatial diffusivities g^i, but are defined based on the color distance between levels

$$\bar{g}^i(x) := \frac{1}{\sqrt{\sum_{j=1}^{3} |I_j^i(x) - I_j^{i-1}(x)|^2 + \varepsilon^2}}, \quad \varepsilon = 0.001 \tag{11.9}$$

rather than the image gradient. The second term in (11.8) is a straightforward extension of the corresponding term in the single-level model.

What is the effect of the additional levels? The superpixels at coarser levels lead to regions with constant values I^i. Consequently, $\nabla I^i(x) = 0$ within a superpixel, which leads to infinite diffusivities[2] g^k. In other words, within a superpixel, label information is propagated with infinite speed across the whole superpixel. Thanks to the connections between levels, this also affects points on the next finer level. Rather than traveling the long distance on the fine level hindered by noisy pixels and weak structures, information can take a shortcut via a coarser level where this noise has been removed.

The hierarchy comes with the great advantage that we need not choose the "correct" noise level, which just might not exist globally. Instead, we consider multiple levels from the superpixel hierarchy

[2]These diffusivities are only made finite for numerical reasons by means of the regularizing constant ε.

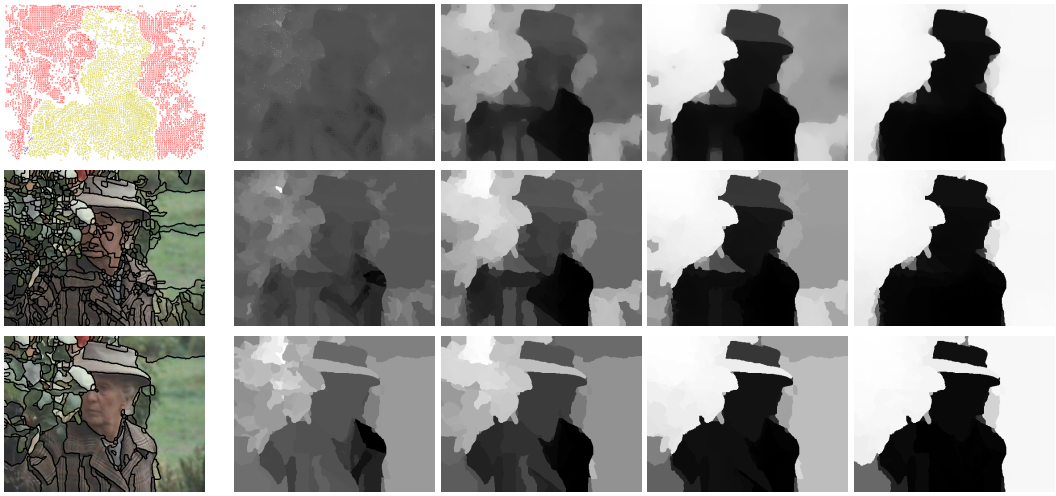


Figure 11.4: Evolution of the label functions u_1^k on all three levels simultaneously. Intermediate states after 30, 300, 3000, and 30000 iterations with the SOR method are shown. For this visualization we did not use the cascadic multigrid strategy, which requires far fewer iterations to converge.

and integrate them all into our model. In theory, it is advantageous to have as many levels as possible. For computational reasons, however, it is wise to focus on a small number of levels. Our experiments indicate that three levels are already sufficient to benefit from the hierarchical model (see Figure 11.6). Figure 11.4 visualizes the propagation of label sources throughout such a 3-level hierarchy.

Since we formulated the hierarchy as a continuous, variational model rather than a common discrete, graphical model, we have the advantage that we do not suffer from discretization artifacts. This is shown in Figure 11.5, where we compare our continuous model to an implementation based on the graph structure. Even though, in the implementation the model is solved on discrete pixel data, this discretization is *consistent*, i.e., the discretization error decreases as the image resolution increases. Moreover, a rotation of the grid does not change the outcome. These natural properties are missing in discrete models.

Figure 11.5: Difference between a discrete (top) and a continuous model (bottom). The discrete model shows block artifacts since its discretization error does not converge to 0 for finer grids sizes.

In (11.8) we have introduced a weighting function ρ that allows for weighting certain trajectories higher. The incentive is that the approach in [BM10] tends to produce wrong labels close to object boundaries due to inaccuracies of the optical flow in such areas. Hence, it makes sense to increase the influence of labels that are far away from object boundaries, whereas labels close to these boundaries should get less influence. Since we do not yet know the object boundaries, the distance is approximated by the Euclidean distance to the superpixel boundaries $\partial\Omega_m$ at the coarsest level, which can be computed very efficiently [FH04]. Based on these distances we define

$$\rho(x) := \frac{dist(x, \partial\Omega_m)}{\frac{1}{|\Omega_m|}\sum_{x\in\Omega_m} dist(x, \partial\Omega_m)} \tag{11.10}$$

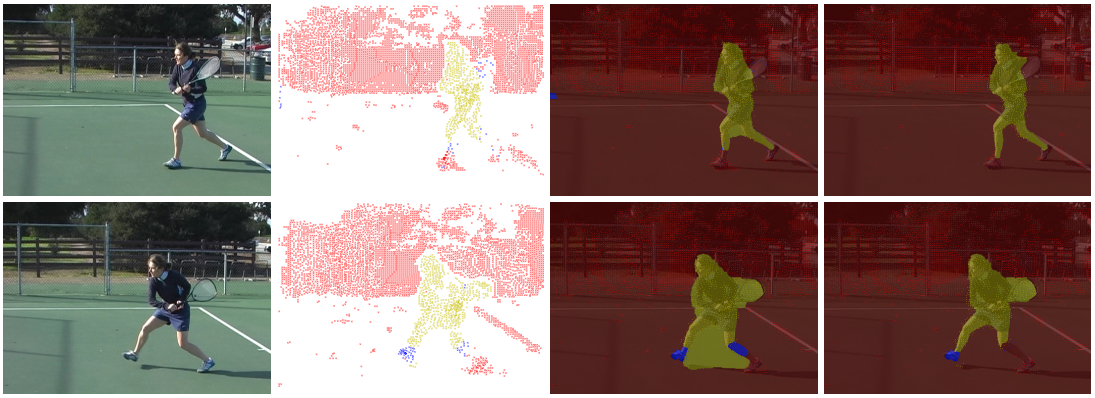


Figure 11.6: LEFT COLUMN: Two frames from the tennis sequence. SECOND COLUMN: Sparse labels [BM10]. THIRD COLUMN: Single-level model. Since there is no label information in the area between the legs and no direct connection to other parts of the tennis court, the single-level model can only interpolate the labels on the legs. RIGHT COLUMN: Thanks to the better information flow inside superpixels and nonlocal diffusivities, the multi-level model can handle this hard case correctly. Remaining problems are due to incorrect motion clustering of the feet in [BM10] and must be approached there.

with $x \in \Omega_m$. This includes a normalization of the distance by the size and shape of the superpixel. In large homogenous regions, where optical flow estimation is most problematic, ρ increases slowly with the distance. In small superpixels, indicating textured areas, even points close to the boundary are assigned large weights.

An example where the hierarchical approach has a strong positive effect is shown in Figure 11.6.

Chapter 12

Evaluation

In this chapter we evaluate the methods that were introduced in Chapters 8, 10 and 11 and compare them with other established methods on the FBMS-59 benchmark presented in Chapter 9. Large parts of this chapter are published in [OMB14].

12.1 Experimental setup

In order to demonstrate that there are challenges in the field of motion segmentation that cannot be properly handled by any previous methodology, we evaluated the approaches from [OB12] and [OMB14], together with the factorization method in [RTVM08] which can deal with incomplete trajectories (ALC), the current state-of-the-art subspace clustering method: sparse subspace clustering [EV13] (SSC_1 and SSC), and a naive baseline method based on two-frame optical flow (Naive). SSC is the standard SSC and SSC_1 is an embedding of SSC into our motion segmentation framework where the only difference to our method is the computation of the affinity matrix. The approaches from [OB12] and [OMB14] are:

- the sparse trajectory clustering based on pairwise affinities MoSegSparse as in [OMB14] (see Section 8.2.2 with the improvements in Section 11.3),
- the sparse trajectory clustering based on higher order affinities MoSegHO as in [OB12] (see Chapter 10)
- the corresponding dense results MoSegDense,
- and MoSegHODense by applying Section 11.4.1.

For ALC and SSC the correct number of labels is provided. Whereas SSC yields a segmentation with exactly this number of labels, ALC uses this number just as a prior. SSC_1 uses the model selection strategy from our framework. For all these methods, the same trajectories with an 8 sampling were used as input. In case of ALC, we randomly subsampled these trajectories by another factor 8 because the method is very slow. The trajectories for SSC_1 and SSC consist of the subset of complete trajectories only.

The baseline method Naive was set up as follows: in the first frame, K reference flow vectors are chosen randomly, where K is set to the ground truth number of objects. All other flow vectors are

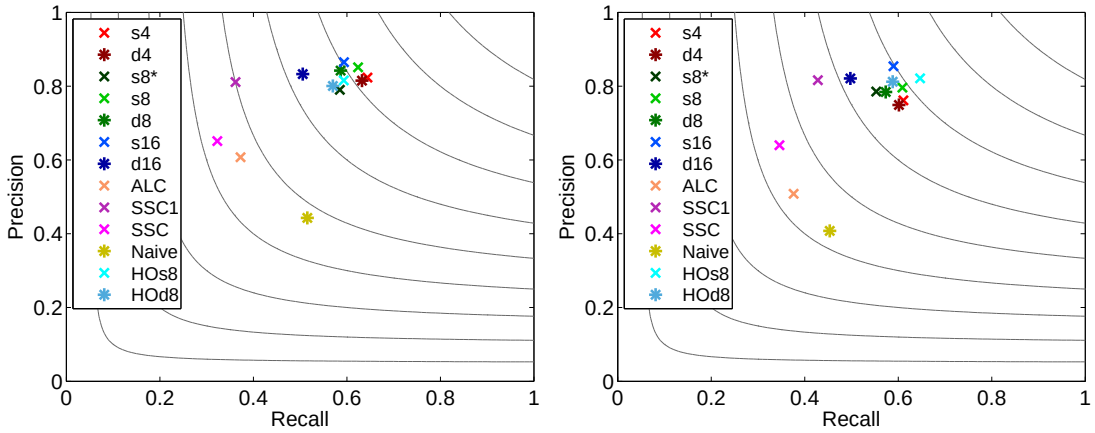


Figure 12.1: Precision–recall graph for the evaluation on all sequences; see also Tab. 12.1. LEFT: Training set. RIGHT: Test set. 's' denotes the sparse clustering result, 'd' the dense segmentation. The number indicates the subsampling rate of the trajectories. The proposed approach performs significantly better than previous approaches. Embedding affinities by SSC into the presented approach improves over the traditional SSC framework. This shows that the definition of affinities and the model selection framework both contribute to the performance.

assigned to the closest reference vector based on the Euclidean distance. In all subsequent frames, the random flow vectors are replaced by the mean flow vector from the previous frame's segmentation with some inertia to account for noise.

For ALC we used the default parameters that came with the code. For all other methods we coarsely optimized the parameters on the training set by a manual search before running the final version on the test set.

Figure 12.1 and Table 12.1 show the results on the FBMS-59 dataset using the new evaluation method proposed in Chapter 9. The performance on the test and training set is quite similar. This shows that over-fitting is not an issue for the evaluated methods. A separate training and test set of reasonable size should avoid methods that over-fit also in the long run. By using the dataset, researchers must agree on running their method on the test set only once and on not using it for parameter optimization.

12.2 Comparison

Comparison to previous methods and the baseline. The results in Table 12.1 and Figure 12.1 show that the presented framework (MoSegSparse, MoSegDense, MoSegHO, MoSegHODense) clearly outperforms all other methods. The main reason is that SSC_1, SSC, and ALC cannot handle occlusions. SSC_1 and SSC work only on the subset of complete trajectories, which reduces the density but also the F-measure considerably. Many objects are missed completely because they are not covered by *any* complete trajectory. While ALC can deal with incomplete trajectories, it fails when trajectories have little overlap. In contrast to SSC_1 and SSC, the density stays high but the F-measure is not better. Also the baseline method based on two-frame optical flow performs poorly, especially on longer sequences. It cannot separate an object from the background if there is no clear motion difference in some frames or in the case of articulated motion.

Comparison sparse vs. dense results. The presented dense segmentation inherits the temporal consistency from the long term analysis of sparse trajectories. A comparison to the sparse result shows that

Training set (29 sequences)

	D	P	R	F	F $\geq 75\%$
	all frames				
MoSegSparse (4)	3.71%	82.33%	64.26%	72.27%	17/65
MoSegDense (4)	100.0%	81.50%	63.23%	71.21%	16/65
MoSegSparse (8)	0.87%	85.10%	62.40%	72.0%	17/65
MoSegSparse* (8)	0.87%	79.02%	58.49%	67.22%	14/65
MoSegDense (8)	100.0%	84.21%	58.67%	69.16%	15/65
MoSegHO (8)	0.83%	81.55%	59.33%	68.68%	16/65
MoSegHODense (8)	100.0%	80.0%	56.99%	66.58%	13/65
MoSegSparse (16)	0.20%	86.51%	59.39%	70.43%	15/65
MoSegDense (16)	100.0%	83.27%	50.56%	62.91%	8/65
ALC	0.09%	60.73%	37.24%	46.18%	0/65
SSC_1	0.17%	81.11%	36.17%	50.03%	7/65
SSC	0.17%	65.12%	32.29%	43.17%	5/65
Naive	100.0%	44.29%	51.54%	47.64%	2/65

Test set (30 sequences)

	D	P	R	F	F $\geq 75\%$
	all frames				
MoSegSparse (4)	3.95%	76.15%	61.11%	67.81%	22/69
MoSegDense (4)	100.0%	74.91%	60.14%	66.72%	20/69
MoSegSparse (8)	0.92%	79.61%	60.91%	69.02%	24/69
MoSegSparse* (8)	0.92%	78.54%	55.29%	64.90%	16/69
MoSegDense (8)	100.0%	78.42%	57.32%	66.23%	17/69
MoSegHO (8)	0.9%	82.11%	64.67%	72.35%	27/65
MoSegHODense (8)	100.0%	81.19%	58.83%	68.22%	19/65
MoSegSparse (16)	0.22%	85.41%	58.98%	69.77%	20/69
MoSegDense (16)	100.0%	82.08%	49.78%	61.97%	9/69
ALC	0.09%	50.83%	37.62%	43.24%	0/69
SSC_1	0.17%	81.62%	42.80%	56.16%	11/69
SSC	0.17%	63.98%	34.61%	44.92%	3/69
Naive	100.0%	40.77%	45.35%	42.94%	1/69

Table 12.1: Results on training (upper block) and test set (lower block). Acronyms are **D**: average region density, **P**: average precision, **R**: average recall, **F**: F-measure and **F** $\geq 75\%$: extracted objects. Numbers are given for the sparse clustering and the dense segmentation with trajectory sampling rate given in parentheses. Details for ALC, SSC_1, SSC and Naive are discussed in the text.

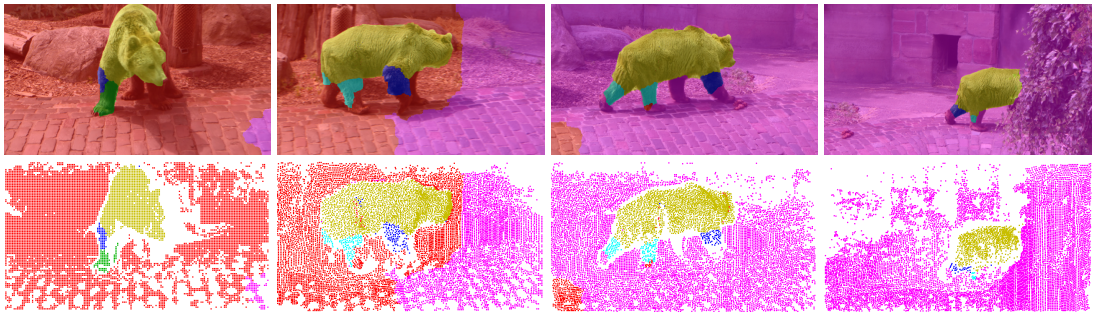


Figure 12.2: Example of an articulated object from the benchmark dataset video shot bear02 with 458 frames and 24 annotated images. TOP ROW: Dense segmentation obtained with the variational model from the point trajectories in the bottom row overlayed with the input images (precision: 98.1%, recall: 74.7%, F-measure: 84.8%). BOTTOM ROW: Clustered point trajectories (precision: 98.9%, recall: 78.4%, F-measure: 87.5%). Clearly, articulated motion leads to an over-segmentation of the object, yet the clusters could also indicate reasonable object parts.

filling the gaps between the trajectories comes with a small loss in performance. Only with a very coarse subsampling of 16, performance drops significantly. Dense segmentation is clearly a harder task, as can be seen in Figure 12.3. While the sparse segmentation just omits the difficult leg area of the cat, the dense segmentation is forced to decide for a label.

Model selection. We also evaluated the effect of model selection. On first glance it is surprising that the results with automatic model selection (SSC_1) are consistently better than those where the correct number of objects is provided (SSC). We believe that this is due to imperfect trajectories that do not allow the detection of all ground truth objects. If affinities propose a bad segmentation, then the constraint to yield a given number of clusters can be counterproductive, while the automatic model selection adapts to such situations. We also ran a variant of our method (marked with *) where we replaced the optimization over K by the correct number of clusters. This number provides only an upper bound, since the regularization could still remove some of the K clusters. This version performed worse than the version with automatic model selection.

Effect of the sampling rate of trajectories. The density of trajectories has a similar effect on model selection as the regularization parameter ν: sparser trajectories lead to fewer clusters. This is because smaller object parts are no longer covered by sufficiently many trajectories to support a separate cluster. This reduces the over-segmentation of articulated objects like the bear in Figure 12.2. On the other hand, smaller objects will be missed if trajectories are too sparse. We found that decreasing the subsampling from 4 to 2 does not help in capturing smaller objects anymore. A deeper analysis indicates that the optical flow is the limiting factor. Since for smaller objects the region area over the contour length gets smaller, misplaced discontinuities in the optical flow have a strong effect and hamper motion segmentation.

Pairwise vs. higher order affinities The motion model with higher order affinities MoSegHO can capture similarity transformations. For sequences where such general motion is present, we observe better results with MoSegHO than with the pairwise model MoSegSparse. A particularly interesting phenomenon is camera zooming. It can be described as a scaling motion. In Chapter 10 examples are shown where the higher order affinities have a high impact in the accuracy. The downside of more general motion model is the overlap of the subspaces of motions. For example a (discrete) rotation motion can have points that can be confused with another translational motion. Such effects make the pairwise model perform better on some sequences than the higher order model.

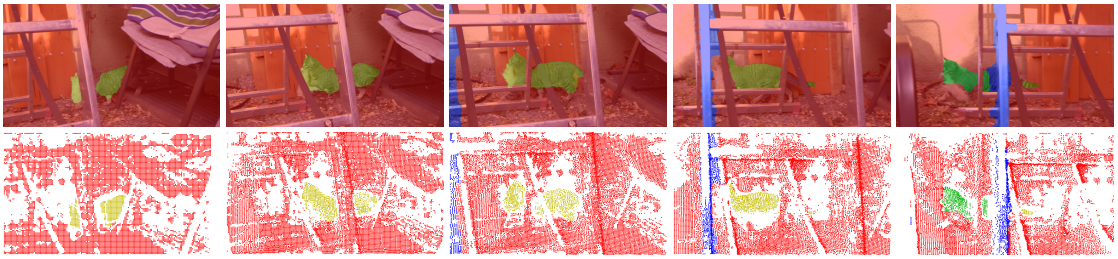


Figure 12.3: Example of a partially occluded object from the video shot cats05 with 87 frames and 6 annotated images. TOP ROW: Dense segmentation obtained with the variational model from the point trajectories in the bottom row overlayed with the input images (precision: 64.0%, recall: 52.8%, F-measure: 57.8%). BOTTOM ROW: Clustered point trajectories (precision: 65.0%, recall: 54.2%, F-measure: 59.1%). Although the trajectories on the cat are short due to occlusions, there is enough temporal overlap to assign trajectories on the cat to the same cluster across most of the obstacles.

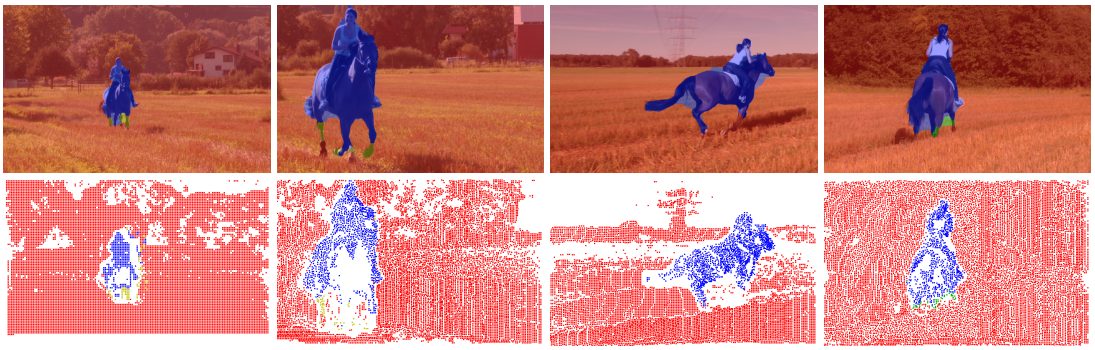


Figure 12.4: Example of a sequence with large perspective background motion from the video shot horses01 with 500 frames and 26 annotated images. TOP ROW: Dense segmentation obtained with the variational model from the point trajectories in the bottom row overlayed with the input images (precision: 94.8%, recall: 93.4%, F-measure: 94.1%). BOTTOM ROW: Clustered point trajectories (precision: 95.1%, recall: 95.3%, F-measure: 95.2%). The horse is seen from different viewpoints and the background changes completely. Nonetheless, there are two temporally consistent clusters, one for the horse and one for the background.

Computational cost. Trajectory subsampling has a positive effect on the computational cost for our methods. For subsampling rates of 4, 8, and 16, the average computation times of the clustering for MoSegSparse are 6s, 800ms, and 600ms per frame, respectively. Although the complexity of computing the affinities for MoSegHO is actually cubic, it can be reduced to quadratic complexity by a clever sampling strategy (see Section 10.3.2). However, the constant factor in the complexity estimate depends on the number of neighbors sampled for each pairwise edge, which is 54. Computation of the forward and backward flow takes on average 20s per frame on the CPU or 2s per frame on the GPU. The dense segmentation on average adds 1s per frame on the GPU. Such low computation times allow the application to scale to large video datasets on commodity hardware.

Articulated objects. The qualitative results show that the method is applicable to quite a general set of sequences and can deal with many challenges. Figure 12.2 highlights the possibility to deal with articulated motion. Strong articulation usually leads to an over-segmentation of the object, as the articulated parts are assigned to separate clusters. This can be a desired effect. The current parameter setting for the dense segmentation tends to smooth out these smaller parts, but these can be modified if necessary. A limitation of the method can be observed for feet that stay on the ground for a long time and then get occluded before they move. These limbs are assigned to the background because a true long term analysis of their motion is hampered by the occlusion.

Partial occlusion. In contrast, partial occlusion of larger objects usually is not a problem, as shown in Figure 12.3. Although the cat is occluded several times, the visible parts show sufficiently similar motion to keep the whole object in the same cluster. Only at the very end of the sequence the overlap of trajectories is too small and the cluster gets split temporally. Strong occlusion due to changing viewpoint can also be handled, as shown in Figure 12.4. The viewpoint changes by almost 180 degree, i.e., hardly any part of the horse in the first frame is visible in the last one. Also the background changes completely. In contrast to many methods from literature, the proposed way to define affinities can handle this case easily.

Chapter 13

Conclusion and future work

In Part I, we introduced and analyzed the convergence of several new algorithms for convex and nonconvex optimization. For a certain class of nonsmooth and strongly convex functions iPiasco was proposed in Chapter 3, which we have shown to have an optimal rate of convergence. Nonconvex algorithms were proposed in Chapters 6 and 7. We proved their convergence. In several computer vision problems their efficiency was verified experimentally. For example, they allow to efficiently optimize the optical flow problem in the presence of nonconvex penalty terms. Due to the wide range of applicability of optical flow, potential improvements of it also affect higher level computer vision tasks like the motion segmentation method that we considered in Part II. We proposed a motion based object level video segmentation method. It is the current state-of-the-art and outperforms other methods in several aspects. In Chapter 10 the model accuracy for the motion analysis between trajectories was increased. Moreover, the clustering of the eigenvectors associated with the trajectories via spectral clustering was improved in Chapter 11. Finally, in Chapter 11, we presented a way to assign object labels to each pixel in a video shot by propagating the information from the labeled trajectories. Chapter 12 shows that our improvements outperform several other methods on our new benchmark that was presented in Chapter 9.

In the following, let us discuss the contributions and related future work.

iPiasco. iPiasco can be seen as proximal Heavy-ball method. It incorporates some inertial force into the current (forward) gradient step and applies a backward step. The backward step, which is equivalent to solving the proximal map for a convex function, allows for nonsmooth objective functions. Regarding the convergence rate, we prove that the additional nonsmooth convex term in the objective does not influence the convergence; It is optimal like for the Heavy-ball method. If we invoke some more structure of the objective function, then iPiasco can perform better than the best first-order black-box optimization algorithm that solves all smooth strongly convex functions equally well.

iPiasco in infinite dimensional setting. As future work, it would be interesting to understand the requirement of twice differentiability for the smooth term of the objective function in the proof of convergence in more detail. Possibly there are ways to circumvent this requirement and to relax it to functions with Lipschitz continuous gradient. In fact, this issue requires the objective function to be defined in a finite dimensional space. Apart from that, the proof of the optimal convergence rate seems to be suitable for generalization to infinite dimensional Hilbert spaces.

Unified abstract convergence theorem. In Chapter 5 we have proved an abstract theorem for convergence of (inertial) descent methods for nonconvex objectives. This work was motivated by an analogous

result for abstract descent methods. As the actual technique of the proof is the same and the understanding of proving convergence using the Kurdyka–Łojasiewicz property recently improved significantly [BST14], I believe that soon there will be a convergence theorem that unifies both.

iPiano. iPiano is formally also a proximal Heavy-ball method. It applies to functions that can be written as a sum of a smooth, nonconvex function and a nonsmooth, nonconvex, simple function. Compared to the publication of iPiano in [OCBP14], where the nonsmooth part is required to be convex and simple, the analysis in this thesis allows for nonconvex simple functions. The convergence analysis is made such that the more structure is known the larger the step sizes that can be used. iPiano is proved to be efficient for computer vision tasks like denoising or image compression.

iPiano for bilevel optimization problems. The problem of image compression arises from a bilevel optimization problem. The structure of this problem is particularly simple, as it allows us to explicitly solve for the solution of the lower level problem. However, there are situations where this is not possible. Nevertheless, for some problems techniques like implicit differentiation can be used to obtain a descent direction on the loss function of the higher level problem. iPiano is a promising algorithm for such problems. In some examples, we have already seen that it is comparable with L-BFGS.

Convergence rates. As we have also seen in the experiments for iPiano, the (simple) convergence rate that we found is quite loose. iPiano performs usually much better than the rate $O(1/\sqrt{N})$ on the residual of the iterates. It would be interesting to find better convergence rates, which seems to be particularly difficult in the nonconvex setting. The issue about convergence rates is also true for IRconvex.

IRconvex. The iteratively reweighted convex algorithms (IRconvex) apply to the most general class of problems considered in this thesis. They arise as a special instance of the principle of majorization minimization for which we proved convergence. Key for majorization minimization methods is the construction of suitable majorizers of the objective function that are easier to optimize than the original problem. A relatively simple class of majorizers can be constructed by fixing a simple function and multiplying it with a weight in order to transform it into a majorizer. IRconvex arises in this way. Examples are the well known iteratively reweighted least squares algorithm or the iteratively reweighted ℓ_1-algorithm, which we considered among others more in detail. In many experiments the efficiency of the reweighting algorithms is shown. As an application in computer vision we considered the optical flow problem. However, the principles of optical flow can easily be transferred to many other problems like denoising, deconvolution, super-resolution, etc..

Optical flow with nonconvex penalty terms. Optical flow estimation is a difficult nonconvex problem even for convex penalty functions. In theory, it is known that nonconvex penalty terms, which are considered to be more robust, are the better choice than convex ones. However, their optimization is challenging. Using IRconvex the optimization with nonconvex penalty terms is easier. Unfortunately, we could only show the potential improvement when using nonconvex penalty terms. In order to incorporate these techniques into current state-of-the-art algorithms for optical flow further investigation is required. The intrinsic nonconvexity of the data-term (correspondence term) in optical flow makes it difficult to benefit from the theoretical advantages that come from nonconvex penalty terms.

Higher order motion models. In the experiments on higher order motion models a benefit in the presence of complex motions is shown. The main disadvantage of using higher order affinities is the computational complexity of evaluating triplets of trajectories. It is quite slow compared to the pairwise analysis, though our proposed sampling strategy already reduces the runtime significantly.

Evaluating triplets on the GPU. It is hard to reduce the complexity of estimating the affinities between triplets, however the computations are independent. Transferring the data to the GPU and processing the triplets in parallel would speed up the computation significantly. On contemporary GPUs a speed up factor of about 50 can be expected. This would make the approach applicable also to large datasets.

Dense motion segmentation. We proposed two models to propagate the label information from the semi-dense set of trajectories to unlabeled areas. The segmentation model is essentially a Potts segmentation model, where the labels given at the trajectories serve as sources for a label, and compact regions (weighted with the magnitude of the image gradient) are sought. The hierarchical model uses more visual cues of the underlying image, like texture, than the single level model. However, on average the performance of the single level model, which can be solved faster, is only little worse than the hierarchical model. On large datasets like the FBMS that we proposed, the single level model seems to be sufficient.

More advanced segmentation models. In recent years, the Potts segmentation model was improved in several ways that are also interesting for video segmentation. Cues that could help our motion segmentation method are for example the proportion priors introduced in [NSC13], spatially varying distributions [NC13], or moment constraints like in [KC11]. Although these models focus on segmenting images given user information, the transfer to unsupervised label information from clustering point trajectories is conceptually straightforward. However, in order to benefit in practice from such models an experimental investigation is required.

Long term motion segmentation. The motion segmentation method that we investigate in this thesis is founded in the work of Brox and Malik [BM10]. By reasoning about motion similarities of trajectories in a video sequence over a long time a robust and consistent segmentation of objects is achieved. The long term aspect even allows to cluster objects that are static for a certain time. The method naturally deals with a moderate amount of (partial) occlusion by a transitivity property of the affinities. Spectral clustering coupled with a model selection method allows to automatically determine the number of objects that are present in a video shot. In experiments we have also seen that the method performs better with model selection than with the right number of clusters given as a hard constraint. Small errors can be better coped with, if some additional clusters may be generated. This is particularly true for articulated objects where the number of clusters depends on the goal of the user.

Extracting articulated parts. The issue about articulated objects is interesting. It could be desirable to extract individual (articulated) parts as separate clusters. Currently the parameters of the motion segmentation method are such that fewer clusters, i.e., whole objects, are usually preferred. One could think of a hierarchical procedure, which extracts the whole object first, and then tries to partition it into its articulated parts. This would give valuable information about an object and, for example, is likely to be beneficial for object detectors trained with this information.

Track repair and continuation. As mentioned above, the motion segmentation method that we considered can deal only with partial occlusions. Therefore it is not a secret that legs usually cause problems. The leg of a walking person that is further away from the camera often gets occluded completely and trajectories have to be stopped. This results in leg areas being not clustered at all or to be merged with the background. As the period of occlusion is rather short and the occluded trajectories move only little, there is hope to continue trajectories through occlusion and to detect them in the disocclusion area again. If trajectories can be extended, there is no need to generate new trajectories at those points and, therefore, also the runtime of the method could be affected positively.

Bibliography

[AA01] F. Alvarez and H. Attouch. An inertial proximal method for maximal monotone operators via discretization of a nonlinear oscillator with damping. *Set-Valued and Variational Analysis*, 9(1-2):3–11, 2001.

[AB09] H. Attouch and J. Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116(1):5–16, June 2009.

[ABB06] S. Agarwal, K. Branson, and S. Belongie. Higher order learning with graphs. In *International Conference on Machine Learning (ICML)*, pages 17–24, New York, NY, USA, 2006. ACM.

[ABM06] H. Attouch, G. Buttazzo, and G. Michaille. *Variational analysis in Sobolev and BV spaces: applications to PDEs and optimization*. SIAM Journal on Applied Mathematics, Philadelphia, 2006.

[ABRS10] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, May 2010.

[ABS13] H. Attouch, J. Bolte, and B. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

[AFS13] M. Artina, M. Fornasier, and F. Solombrino. Linearly constrained nonsmooth and nonconvex minimization. *SIAM Journal on Optimization*, 23(3):1904–1937, 2013.

[AIG06] M. Allain, J. Idier, and Y. Goussard. On global and local convergence of half-quadratic algorithms. *IEEE Transactions on Image Processing*, 15(5):1130–1142, May 2006.

[AK95] C.J. Alpert and A.B. Kahng. Recent directions in netlist partitioning: a survey. *Integration: The VLSI Journal*, 19(1-2):1–81, 1995.

[AK98] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1-2):237–260, December 1998.

[AK06] T. Amiaz and N. Kiryati. Piecewise-smooth dense optical flow via level sets. *International Journal of Computer Vision*, 68(2):111–124, 2006.

[Alv03] F. Alvarez. Weak convergence of a relaxed and inertial hybrid projection-proximal point algorithm for maximal monotone operators in Hilbert space. *SIAM Journal on Optimization*, 14(3):773–782, 2003.

[ALZ+05] S. Agarwal, J. Lim, L. Zelnik–Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society.

[AMFM11] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[APR14] H. Attouch, J. Peypouquet, and P. Redont. A dynamical approach to an inertial forward–backward algorithm for convex minimization. *SIAM Journal on Optimization*, 24(1):232–256, 2014.

[ARS12] A. Ayvaci, M. Raptis, and S. Soatto. Sparse occlusion detection with optical flow. *International Journal of Computer Vision*, 97(3):322–338, 2012.

[BB91] T.E. Boult and L.G. Brown. Factorization-based segmentation of motions. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 179–186, 1991.

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warpin g. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision (ECCV)*, volume 3024 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin Heidelberg, 2004.

[BBW06] T. Brox, A. Bruhn, and J. Weickert. Variational motion segmentation with level sets. In *European Conference on Computer Vision (ECCV)*, volume 3951 of *Lecture Notes in Computer Science*, pages 471–483. Springer, 2006.

[BC06] G.J. Brostow and R. Cipolla. Unsupervised bayesian detection of independent motion in crowds. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 594–601, Washington, DC, USA, 2006. IEEE Computer Society.

[BC09] T. Brox and D. Cremers. On local region models and the statistical interpretation of the piecewise smooth Mumford–Shah functional. *International Journal of Computer Vision*, 84(2):184–193, 2009.

[BC11] H.H. Bauschke and P.L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.

[BC14] R.I. Bot and E.R. Csetnek. An inertial Tseng's type proximal algorithm for nonsmooth and nonconvex optimization problems. *ArXiv e-prints*, 1406.0724, June 2014.

[BCL14] R.I. Bot, E.R. Csetnek, and S. László. An inertial forward–backward algorithm for the minimization of the sum of two nonconvex functions. *ArXiv e-prints*, 1410.0641, 2014.

[BCR98] J. Bochnak, M. Coste, and M.-F. Roy. *Real algebraic geometry*. Springer, 1998.

[BDF07] J.M. Bioucas-Dias and M. Figueiredo. A new TwIST: two-step iterative shrinkage-thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, 2007.

[BDL06a] J. Bolte, A. Daniilidis, and A. Lewis. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, December 2006.

[BDL06b] J. Bolte, A. Daniilidis, and A. Lewis. A nonsmooth Morse–Sard theorem for subanalytic functions. *Journal of Mathematical Analysis and Applications*, 321(2):729–740, 2006.

[BDLM10] J. Bolte, A. Daniilidis, A. Ley, and L. Mazet. Characterizations of Łojasiewicz inequalities: subgradient flows, talweg, convexity. *Transactions of the American Mathematical Society*, 362:3319–3363, 2010.

[BDLS07] J. Bolte, A. Daniilidis, A.S. Lewis, and M. Shiota. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2):556–572, 2007.

[Ber99] D.P. Bertsekas. *Nonlinear programming*. Athena Scientific, 2nd edition, September 1999.

[BFL06] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[BKP10] K. Bredies, K. Kunisch, and T. Pock. Total generalized variation. *SIAM Journal on Applied Mathematics*, 3(3):492–526, September 2010.

[BL11] K. Bredies and D. Lorenz. *Mathematische Bildverarbeitung – Einführung in Grundlagen und moderne Theorie*. Vieweg+Teubner, 2011.

[BM88] E. Bierstone and P. Milman. Semianalytic and subanalytic sets. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 67(1):5–42, 1988.

[BM98] S. Belongie and J. Malik. Finding boundaries in natural images: a new method using point descriptors and area completion. In *European Conference on Computer Vision (ECCV)*, volume 1406 of *Lecture Notes in Computer Science*, pages 751–766. Springer, 1998.

[BM10] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *European Conference on Computer Vision (ECCV)*, volume 6315 of *Lecture Notes in Computer Science*, pages 282–295. Springer Berlin Heidelberg, 2010.

[BM11] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[BMCM97] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 495–501, Washington, DC, USA, 1997. IEEE Computer Society.

[BR96] M.J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91, 1996.

[Bru77] R. Bruck. On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in Hilbert space. *Journal of Mathematical Analysis and Applications*, 61(1):159–164, 1977.

[BS07] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *International Conference on Computer Vision (ICCV)*, 2007.

[BSFC08] G. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In A. Zisserman D. Forsyth, P. Torr, editor, *European Conference on Computer Vision (ECCV)*, 5302, pages 44–57, Berlin, Heidelberg, 2008. Springer-Verlag.

[BST14] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[BT09a] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Applied Mathematics*, 2(1):183–202, March 2009.

[BT09b] A. Beck and M. Teboulle. Gradient-based algorithms with applications to signal-recovery problems. In *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, 2009.

[BT09c] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *International Conference on Computer Vision (ICCV)*, pages 833–840, 2009.

[BTC+10] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, Z. Lijun, and X. He. Music recommendation by unified hypergraph: combining social media information and music content. In A.D. Bimbo, S.-F. Chang, and A.W.M. Smeulders, editors, *ACM Multimedia*, pages 391–400. ACM, 2010.

[BWSB12] D.J. Butler, J. Wulff, G.B. Stanley, and M.J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *European Conference on Computer Vision (ECCV)*, volume 7577 of *Lecture Notes in Computer Science*, pages 611–625. Springer-Verlag, October 2012.

[BZ87] A. Blake and A. Zisserman. *Visual reconstruction*. MIT Press, Cambridge, MA, 1987.

[CCC+10] V. Caselles, A. Chambolle, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. *Theoretical Foundations and Numerical Methods for Sparse Recovery*, 9:263–340, 2010.

[CCP12] A. Chambolle, D. Cremers, and T. Pock. A convex approach to minimal partitions. *SIAM Journal on Applied Mathematics*, 5(4):1113–1158, 2012.

[CDV10] P.L. Combettes, D. Dũng, and B.C. Vũ. Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, 18(3-4):373–404, 2010.

[CEN06] T. Chan, S. Esedoḡlu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics*, 66(5):1632–1648, 2006.

[Chu97] F.R.K. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[CIM11] E. Chouzenoux, J. Idier, and S. Moussaoui. A majorize–minimize strategy for subspace optimization applied to image restoration. *IEEE Transactions on Image Processing*, 20(6):1517–1528, June 2011.

[CJPT13] E. Chouzenoux, A. Jezierska, J. Pesquet, and H. Talbot. A majorize–minimize subspace approach for ℓ_2-ℓ_0 image regularization. *SIAM Journal on Imaging Sciences*, 6(1):563–591, January 2013.

[CK95] J. Costeira and T. Kanande. A multi-body factorization method for motion analysis. In *International Conference on Computer Vision (ICCV)*, pages 1071–1076, 1995.

[CL09] G. Chen and G. Lerman. Spectral curvature clustering SCC. *International Journal of Computer Vision*, 81:317–330, 2009.

[Con13] L. Condat. A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. *Journal of Optimization Theory and Applications*, 158(2):460–479, 2013.

[Cos00] M. Coste. *An introduction to o-minimal geometry*. Dottorato di ricerca in matematica / Università di Pisa, Dipartimento di Matematica. Instituti Editoriali e Poligrafici Internazionali, 2000.

[CP11] A. Chambolle and T. Pock. A first-order primal–dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[CP12] P.L. Combettes and J.-C. Pesquet. Primal–dual splitting algorithm for solving inclusions with mixtures of composite, Lipschitzian, and parallel-sum type monotone operators. *Set-Valued and Variational Analysis*, 20(2):307–330, 2012.

[CP14] A. Chambolle and T. Pock. On the ergodic convergence rates of a first-order primal–dual algorithm. *Technical Report*, 2014. To appear.

[CPKC11] D. Cremers, T. Pock, K. Kolev, and A. Chambolle. Convex relaxation techniques for segmentation, stereo and multiview reconstruction. In *Advances in Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

[CPR13] E. Chouzenoux, J.-C. Pesquet, and A. Repetti. Variable metric forward–backward algorithm for minimizing the sum of a differentiable function and a convex function. *Journal of Optimization Theory and Applications*, November 2013.

[CPRB13] Y.J. Chen, T. Pock, R. Ranftl, and H. Bischof. Revisiting loss-specific training of filter-based MRFs for image restoration. In *German Conference on Pattern Recognition (GCPR)*, 2013.

[CR09] A. Cheriyadat and R. Radke. Non-negative matrix factorization of partial track data for motion segmentation. In *International Conference on Computer Vision (ICCV)*, 2009.

[CS05] D. Cremers and S. Soatto. Motion competition: a variational framework for piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, May 2005.

[CW05] P.L. Combettes and V.R. Wajs. Signal recovery by proximal forward–backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[CWB08] E.J. Candes, M.B. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 2008.

[CZ13] X. Chen and W. Zhou. Convergence of the reweighted ℓ_1 minimization algorithm for ℓ_2-ℓ_p minimization. *Computational Optimization and Applications*, pages 1–15, 2013.

[dD98] L. Van den Dries. *Tame topology and o-minimal Structures*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1998.

[DDFG10] I. Daubechies, R. Devore, M. Fornasier, and C. S. Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.

[dDM96] L. Van den Dries and C. Miller. Geometric categories and o-minimal structures. *Duke Mathematical Journal*, 84(2):497–540, 08 1996.

[DDM04] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[DG64] J. Douglas and J.E. Gunn. A general formulation of alternating direction methods. *Numerische Mathematik*, 6(1):428–453, 1964.

[DH73] W.E. Donath and A.J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, September 1973.

[DMM07] A. Desolneux, L. Moisan, and J.-M. Morel. *From Gestalt Theory to image analysis: a probabilistic approach*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[DR56] J. Douglas and H.H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transaction of the American Mathematical Society*, 82:421–489, 1956.

[DS09] J. Duchi and Y. Singer. Efficient online and batch learning using forward–backward splitting. In H. Liu, H. Motoda, R. Setiono, and Z. Zhao, editors, *International Workshop on Feature Selection in Data Mining*, volume 10 of *Journal of Machine Learning Research - Workshop and Conference Proceedings*, pages 2899–2934, December 2009.

[DT13] Y. Drori and M. Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, pages 1–32, 2013.

[Dun81] J.C. Dunn. Global and asymptotic convergence rate estimates for a class of projected gradient processes. *SIAM Journal on Control and Optimization*, 19(3):368–400, May 1981.

[EB92] J. Eckstein and D.P. Bertsekas. On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(3):293–318, June 1992.

[Eck89] J. Eckstein. *Splitting methods for monotone operators with applications to parallel optimization*. Technical report. Center for Intelligent Control Systems, M.I.T., 1989.

[EV13] E. Elhamifar and R. Vidal. Sparse subspace clustering: algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[EZC10] E. Esser, X. Zhang, and T.F. Chan. A general framework for a class of first order primal–dual algorithms for convex optimization in imaging science. *SIAM Journal on Imaging Sciences*, 3(4):1015–1046, December 2010.

[FBM] Freiburg Berkeley Motion Segmentation Dataset (FBMS). `http://lmb.informatik.uni-freiburg.de/resources/datasets/FBMS_Trainingset.tar.gz` and `http://lmb.informatik.uni-freiburg.de/resources/datasets/FBMS_Testset.tar.gz`.

[FCP+14] A. Florescu, E. Chouzenoux, J.-C. Pesquet, P. Ciuciu, and S. Ciochina. A majorize–minimize memory gradient method for complex-valued inverse problems. *Signal Processing*, 103:285–295, October 2014.

[FGP14] P. Frankel, G. Garrigos, and J. Peypouquet. Splitting methods with variable metric for KL functions. *ArXiv e-prints*, 1405.1357, May 2014.

[FH94] J.A. Fessler and A.O. Hero. Space-alternating generalized expectation–maximization algorithm. *IEEE Transactions on Signal Processing*, 42(10):2664–2677, October 1994.

[FH95] J.A. Fessler and A.O. Hero. Penalized maximum-likelihood image reconstruction using space-alternating generalized EM algorithms. *IEEE Transactions on Image Processing*, 4(10):1417–1429, October 1995.

[FH04] P.F. Felzenszwalb and D.P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Computer Science Department, Cornell University, September 2004.

[FM81] M. Fukushima and H. Mine. A generalized proximal point algorithm for certain non-convex minimization problems. *International Journal of Systems Science*, 12(8):989–1000, 1981.

[FRP09] M. Fradet, P. Robert, and P. Pérez. Clustering point trajectories with various life-spans. In *European Conference on Visual Media Production*, 2009.

[FS11] K. Fragkiadaki and J. Shi. Detection free tracking: exploiting motion and topology for segmenting and tracking under entanglement. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[FW10] M. Fornasier and R. Ward. Iterative thresholding meets free-discontinuity problems. *Foundations of Computational Mathematics*, 10(5):527–567, 2010.

[FZS12] K. Fragkiadaki, G. Zhang, and J. Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, Rhode Island, USA, 2012.

[FZZS12] K. Fragkiadaki, W. Zhang, G. Zhang, and J. Shi. Two granularity tracking: mediating trajectory and detection graphs for tracking under occlusions. In *European Conference on Computer Vision (ECCV)*, 2012.

[Gab83] D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary Value Problem*, pages 299–340, North-Holland, Amsterdam, 1983.

[Gab96] A. Gabrielov. Complements of subanalytic sets and existential formulas for analytic functions. *Inventiones mathematicae*, 125(1):1–12, 1996.

[GCS12] F. Galasso, R. Cipolla, and B. Schiele. Video segmentation with superpixels. In *Asian Conference on Computer Vision (ACCV)*, 2012.

[Gea98] C.W. Gear. Multibody grouping from motion images. *International Journal of Computer Vision*, 29(2):133–150, 1998.

[Gel41] I. Gelfand. Normierte Ringe. *Rec. Math. [Mat. Sbornik] N.S.*, 9(51):3–24, 1941.

[GG84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[GINC11] F. Galasso, M. Iwasaki, K. Nobori, and R. Cipolla. Spatio-temporal clustering of probabilistic region trajectories. In *International Conference on Computer Vision (ICCV)*, 2011.

[GKHE10] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[GKR00] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: an approach based on dynamical systems. *VLDB Journal: Very Large Data Bases*, 8(3–4):222–236, 2000.

[GM09] D. Goldfarb and S. Ma. Fast multiple splitting algorithms for convex optimization. *ArXiv e-prints*, 0912.4570, December 2009.

[Gol64] A.A. Goldstein. Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, 70:709–710, 1964.

[Gov05] V. Govindu. A tensor decomposition for geometric grouping and segmentation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1150–1157, 2005.

[GR92] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:367–383, 1992.

[Gra06] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[GWW+08] I. Galic, J. Weickert, M. Welk, A. Bruhn, A.G. Belyaev, and H.-P. Seidel. Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 31(2-3):255–269, 2008.

[Hes69] M.R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[HK91] L. Hagen and A. Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *IEEE International Conference on Computer-Aided Design*, pages 10–13, November 1991.

[HL04] D. R. Hunter and K. Lange. A tutorial on MM algorithms. *Journal of the American Statistical Association*, 58(1):30–37, 2004.

[HL12] M. Hong and Z.-Q. Luo. On the linear convergence of the alternating direction method of multipliers. *ArXiv e-prints*, 1208.3922, August 2012.

[HLM09] Y. Huang, Q. Liu, and D. Metaxas. Video object segmentation by hypergraph cut. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1738–1745, June 2009.

[HM99] J. Huang and D. Mumford. Statistics of natural images and models. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 541–547, Fort Collins, CO, USA, 1999.

[HSJR13] M. Hein, S. Setzer, L. Jost, and S.S. Rangapuram. The total variation on hypergraphs - learning on hypergraphs revisited. In C.J.C. Burges, L. Bottou, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 2427–2435, 2013.

[HSW13] L. Hoeltgen, S. Setzer, and J. Weickert. An optimal control approach to find sparse data for Laplace interpolation. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 151–164, 2013.

[HY12a] B. He and X. Yuan. Convergence analysis of primal–dual algorithms for a saddle-point problem: from contraction perspective. *SIAM Journal on Imaging Sciences*, 5(1):119–149, 2012.

[HY12b] B. He and X. Yuan. On the $O(1/n)$ convergence rate of the Douglas–Rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[Idi01] J. Idier. Convex half-quadratic criteria and interacting auxiliary variables for image restoration. *IEEE Transactions on Image Processing*, 10(7):1001–1009, July 2001.

[Ish09] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2993 –3000, June 2009.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[JF07] M.W. Jacobson and J.A. Fessler. An expanded theoretical treatment of iteration dependent majorize–minimize algorithms. *IEEE Transactions on Signal Processing*, 16(10):2411–2422, October 2007.

[Kan80] G. Kanizsa. *Grammatica del Vedere*. Società editrice Il Mulino, Bologna, 1980.

[Kan97] G. Kanizsa. *La Grammaire du Voir*. Diderot, 1997.

[KC11] M. Klodt and D. Cremers. A convex framework for image segmentation with moment constraints. In *International Conference on Computer Vision (ICCV)*, 2011.

[KHT09] S. Klamt, U.-U. Haus, and F. Theis. Hypergraphs and cellular networks. *PLoS Comput Biol*, 5(5), May 2009.

[KKT07] P. Kohli, M.P. Kumar, and P. Torr. P3 & beyond: solving energies with higher-order cliques. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[Kof35] K. Koffka. *Principles of Gestalt Psychology*. Hartcourt Brace Jovanovich, New York, 1935.

[KT07] N. Komodakis and G. Tziritas. Approximate labeling via graph-cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453, 2007.

[KTP08] N. Komodakis, G. Tziritas, and N. Paragios. Performance vs computational efficiency for optimizing single and dynamic MRFs: setting the state of the art with primal–dual strategies. *Computer Vision and Image Understanding*, 112(1):14–29, October 2008.

[Kur98] K. Kurdyka. On gradients of functions definable in o-minimal structures. *Annales de l'institut Fourier*, 48(3):769–783, 1998.

[LASL11] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: spatio-temporal video segmentation with long-range motion cues. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[LBS09] J. Lellmann, F. Becker, and C. Schnörr. Convex optimization for multi-class image labeling with a novel family of total variation based regularizers. In *International Conference on Computer Vision (ICCV)*, pages 646–653, 2009.

[LHY00] K. Lange, D.R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1), 2000.

[LK81] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Seventh International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, Canada, August 1981.

[LKSV07] T. Li, V. Kallem, D. Singaraju, and R. Vidal. Projective factorization of multiple rigid-body motions. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

[LLDS12] R. Liu, Z. Lin, F. De la Torre, and Z. Su. Fixed-rank representation for unsupervised visual learning. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[LLY10] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning (ICML)*, pages 663–670. Omnipress, 2010.

[LM79] P.L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Applied Mathematics*, 16(6):964–979, 1979.

[LN89] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[Łoj63] S. Łojasiewicz. Une propriété topologique des sous-ensembles analytiques réels. In *Les Équations aux Dérivées Partielles*, pages 87–89, Paris, 1963. Éditions du centre National de la Recherche Scientifique.

[Low85] D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, USA, 1985.

[LP66] E.S. Levitin and B.T. Polyak. Constrained minimization problems. *USSR Computational Mathematics and Mathematical Physics*, 6:1–50, 1966.

[LP14] D. Lorenz and T. Pock. An inertial forward–backward algorithm for monotone inclusions. *ArXiv e-prints*, 1403.3522, March 2014.

[LS09] F. Lauer and C. Schnörr. Spectral clustering of linear subspaces for motion segmentation. In *International Conference on Computer Vision (ICCV)*, 2009.

[LS12] M. Leordeanu and C. Sminchisescu. Efficient hypergraph clustering. In N.D. Lawrence and M. Girolami, editors, *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 22 of *Journal of Machine Learning Research - Workshop and Conference Proceedings*, pages 676–684, 2012.

[Mar70] B. Martinet. Brève communication. régularisation d'inéquations variationnelles par approximations successives. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 4(R3):154–158, 1970.

[MBWF11] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer. Edge-based compression of cartoon-like images with homogeneous diffusion. *Pattern Recognition*, 44(9):1859–1873, 2011.

[MHW+11] M. Mainberger, S. Hoffmann, J. Weickert, C.H. Tang, D. Johannsen, F. Neumann, and B. Doerr. Optimising spatial and tonal data for homogeneous diffusion inpainting. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 26–37, 2011.

[Min62] G.J. Minty. Monotone (nonlinear) operators in Hilbert space. *Duke Mathematical Journal*, 29(3):341–346, 09 1962.

[MO03] A. Moudafi and M. Oliny. Convergence of a splitting inertial proximal method for monotone operators. *Journal of Computational and Applied Mathematics*, 155:447–454, 2003.

[Mor65] J.J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la S. M. F.*, 93:273–299, 1965.

[MS85] D. Mumford and J. Shah. Boundary detection by minimizing functionals, I. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 22–26, San Francisco, CA, June 1985. IEEE Computer Society Press.

[MS89] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42:577–685, 1989.

[MSMC14] T. Möllenhoff, E. Strekalovskiy, M. Moeller, and D. Cremers. The primal–dual hybrid gradient method for semiconvex splittings. *ArXiv e-prints*, 1407.1723, 2014.

[MW09] M. Mainberger and J. Weickert. Edge-based image compression with homogeneous diffusion. In X. Jiang and N. Petkov, editors, *Computer Analysis of Images and Patterns*, volume 5702 of *Lecture Notes in Computer Science*, pages 476–483. Springer Berlin Heidelberg, 2009.

[NBRC10] C. Nieuwenhuis, B. Berkels, M. Rumpf, and D. Cremers. Interactive motion segmentation. In *Annual Symposium of the German Association of Pattern Recognition (DAGM)*, volume 6376 of *Lecture Notes in Computer Science*, pages 483–492. Springer, 2010.

[NC07] M. Nikolova and R.H. Chan. The equivalence of half-quadratic minimization and the gradient linearization iteration. *IEEE Transactions on Image Processing*, 16(6):1623–1627, 2007.

[NC13] C. Nieuwenhuis and D. Cremers. Spatially varying color distributions for interactive multilabel segmentation. *Pattern Analysis and Machine Intelligence*, 35(5):1234–1247, 2013.

[Nes83] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

[Nes04] Y. Nesterov. *Introductory lectures on convex optimization: a basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004.

[Nes05] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, May 2005.

[Nes13] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[Nik99] M. Nikolova. Markovian reconstruction using a GNC approach. *IEEE Transactions on Image Processing*, 8(9):1204–1220, 1999.

[NJW02] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[NOLB12] N.S. Nagaraja, P. Ochs, K. Liu, and T. Brox. Hierarchy of localized random forests for video annotation. In *Annual Symposium of the German Association of Pattern Recognition (DAGM)*. Springer, LNCS, 2012.

[NSC13] C. Nieuwenhuis, E. Strekalovskiy, and D. Cremers. Proportion priors for image sequence segmentation. In *International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[NW06] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[OB11] P. Ochs and T. Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *International Conference on Computer Vision (ICCV)*, pages 1583–1590, November 2011.

[OB12] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 614–621, 2012.

[OBP15] P. Ochs, T. Brox, and T. Pock. iPiasco: inertial proximal algorithm for strongly convex optimization. *Journal of Mathematical Imaging and Vision*, 2015.

[OCBP14] P. Ochs, Y. Chen, T. Brox, and T. Pock. iPiano: inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419, 2014.

[ODBP15] P. Ochs, A. Dosovitskiy, T. Brox, and T. Pock. On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. *SIAM Journal on Imaging Sciences*, 8(1):331–372, 2015.

[ODPB13] P. Ochs, A. Dosovitskiy, T. Pock, and T. Brox. An iterated ℓ_1 algorithm for nonsmooth nonconvex optimization in computer vision. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1759–1766, June 2013.

[OMB09] B. Ommer, T. Mader, and J. Buhmann. Seeing the objects behind the dots: recognition in videos from a moving camera. *International Journal of Computer Vision*, 83(1):57–71, 2009.

[OMB14] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1187–1200, June 2014.

[OMG+09] Y. Ostrovsky, E. Meyers, S. Ganesh, U. Mathur, and P. Sinha. Visual parsing after recovery from blindness. *Psychological Science*, 20(12):1484–1491, 2009.

[Pas79] G.B. Passty. Ergodic convergence to a zero of the sum of monotone operators in Hilbert space. *Journal of Mathematical Analysis and Applications*, 72(2):383–390, 1979.

[PC11] T. Pock and A. Chambolle. Diagonal preconditioning for first order primal–dual algorithms in convex optimization. In *International Conference on Computer Vision (ICCV)*, 2011.

[PCCB09] T. Pock, A. Chambolle, D. Cremers, and H. Bischof. A convex relaxation approach for computing minimal partitions. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[PM87] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. In *Proc. IEEE Computer Society Workshop on Computer Vision*, pages 16–22, Miami Beach, FL, November 1987.

[PMC09] B.L. Price, B.S. Morse, and S. Cohen. LIVEcut: learning-based interactive video segmentation by evaluation of multiple propagated cues. In *International Conference on Computer Vision (ICCV)*, 2009.

[Pol64] B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[Pol87] B.T. Polyak. *Introduction to optimization*. Optimization Software, 1987.

[Pow69] M.J.D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, 1969.

[PR55] D.W. Peaceman and H.H. Rachford. The numerical solution of parabolic and elliptic differential equations. *SIAM Journal on Applied Mathematics*, 3(1), 1955.

[PSL90] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, July 1990.

[PTZ08] M. Pawan Kumar, P. Torr, and A. Zisserman. Learning layered motion segmentations of video. *International Journal of Computer Vision*, 76:301–319, 2008.

[PZB11] T. Pock, L. Zebedin, and H. Bischof. TGV-fusion. In A. Salomaa C.S. Calude, G. Rozenberg, editor, *Rainbow of Computer Science*, volume 6570 of *Lecture Notes in Computer Science*, pages 245–258. Springer Berlin Heidelberg, 2011.

[RB09] S. Roth and M.J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2009.

[RFP13] H. Raguet, J. Fadili, and G. Peyré. A generalized forward–backward splitting. *SIAM Journal on Imaging Sciences*, 6(3):1199–1226, 2013.

[RMH14] S.S. Rangapuram, P.K. Mudrakarta, and M. Hein. Tight continuous relaxation of the balanced k-cut problem. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[Roc70] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.

[Roc76] R.T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Applied Mathematics*, 14(5), 1976.

[ROF92] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[RP09] B.S. Rota and M. Pelillo. A game-theoretic approach to hypergraph clustering. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1571–1579. Curran Associates, Inc., 2009.

[RTVM08] S.R. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[RW98] R.T. Rockafellar and R.J.B. Wets. *Variational Analysis*, volume 317 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, 1998.

[SALT09] P. Sturgess, K. Alahari, L. Ladicky, and P. Torr. Combining appearance and structure from motion features for road scene understanding. In *British Machine Vision Conference*. British Machine Vision Association, 2009.

[SBK10] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science. Springer, 2010.

[SG07] A. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *International Conference on Computer Vision (ICCV)*, 2007.

[Shi93] M. Shiota. *Geometry of subanalytic and semialgebraic sets*. Kyōyōbu: Preprint series. Department of Math., College of General Education, Univ., 1993.

[Sho85] N.Z. Shor. *Minimization methods for non-differentiable functions*. Springer series in computational mathematics. Springer-Verlag, 1985.

[SM98] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *International Conference on Computer Vision (ICCV)*, pages 1154–1160, Bombay, India, January 1998.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[Sra12] S. Sra. Scalable nonconvex inexact proximal splitting. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 530–538. Curran Associates, Inc., 2012.

[SSB12] D. Sun, E.B. Sudderth, and M. Black. Layered segmentation and optical flow estimation over time. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[SSW08] K. Schindler, D. Suter, and H. Wang. A model-selection framework for multibody structure-and-motion of image sequences. *International Journal of Computer Vision*, 79(2):159–177, 2008.

[SSZ06] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *International Journal of Computer Vision*, 67(2):189–210, 2006.

[ST94] J. Shi and C. Tomasi. Good features to track. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[SWB09] C. Schmaltz, J. Weickert, and A. Bruhn. Beating the quality of JPEG 2000 with anisotropic diffusion. In *Annual Symposium of the German Association of Pattern Recognition (DAGM)*, pages 452–461, 2009.

[SZH06] A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *European Conference on Computer Vision (ECCV)*, pages 595–608, 2006.

[TA77] A.N. Tikhonov and V.Y. Arsenin. *Solutions of ill–posed problems*. Wiley, Washington, DC, 1977.

[THD09] H.A. Le Thi, V.N. Huynh, and T. Pham Dinh. Convergence analysis of DC algorithm for DC programming with subanalytic data. Technical report, Annals of Operations Research, INSA-Rouen, 2009.

[THK09] Z. Tian, T. Hwang, and R. Kuang. A hypergraph-based learning algorithm for classifying gene expression and arrayCGH data with prior knowledge. *Bioinformatics*, 25(21):2831–2838, 2009.

[TM93] E.-G. Talbi and T. Muntean. Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to the mapping problem. In *International Conference on System Sciences*, volume ii, pages 565–573 vol.2, 1993.

[TORO+12] M. Temerinac-Ott, O. Ronneberger, P. Ochs, W. Driever, T. Brox, and H. Burkhardt. Multiview deblurring for 3-D images from light sheet based fluorescence microscopy. *IEEE Transactions on Image Processing*, 21(4):1863–1873, 2012.

[Tse91] P. Tseng. Applications of splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, 29(1):119–138, January 1991.

[Tse08] P. Tseng. Accelerated proximal gradient methods for convex–concave optimization. *submitted to SIAM Journal on Optimization*, 2008.

[TV07] R. Tron and R. Vidal. A benchmark for the comparison of 3-D motion segmentation algorithms. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[UPT+08] M. Unger, T. Pock, W. Trobin, D. Cremers, and H. Bischof. TVSeg - interactive total variation based image segmentation. In *British Machine Vision Conference*, 2008.

[Val14] T. Valkonen. A primal–dual hybrid gradient method for nonlinear operators with applications to MRI. *Inverse Problems*, 30(5), 2014.

[VAPM10] A. Vazquez-Reina, S. Avidan, H. Pfister, and E. Miller. Multiple hypothesis video segmentation from superpixel flows. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science. Springer, 2010.

[Veg] Vegeta from Dragon Ball Z. `http://1.bp.blogspot.com/-g3wpzDWE0QI/UbycqlJWjDI/AAAAAAAAAbU/ty0EZl0kqXw/s1600/VEGETA\%2B1.jpg`.

[vL07] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[VO98] C.R. Vogel and M.E. Oman. Fast, robust total variation-based reconstruction of noisy, blurred images. *IEEE Transactions on Image Processing*, 7(6):813–824, 1998.

[VTH08] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using power factorization and GPCA. *International Journal of Computer Vision*, 79(1):85–105, 2008.

[Vũ13] B. Vũ. A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Advances in Computational Mathematics*, 38(3):667–681, 2013.

[WA94] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[Wer23] M. Wertheimer. Untersuchungen zur Lehre von der Gestalt. II. *Psychologische Forschung*, 4(1):301–350, 1923.

[WIK] Gestalt principle of common fate. `http://en.wikipedia.org/wiki/Principles_of_grouping`.

[Wil96] A.J. Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted pfaffian functions and the exponential function. *Journal of the American Mathematical Society*, 9(4):1051–1094, 1996.

[WPB10] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[WRHS13] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: large displacement optical flow with deep matching. In *International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[WTG06] X. Wang, K. Tieu, and E. Grimson. Learning semantic scene models by trajectory analysis. In *European Conference on Computer Vision (ECCV)*, volume 3953 of *Lecture Notes in Computer Science*, pages 110–123. Springer, 2006.

[WWR+13] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele. Monocular visual scene understanding: understanding multi-object traffic scenes. *Pattern Analysis and Machine Intelligence*, 35(4):882–897, 2013.

[XJM12] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012.

[XS05] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1644–1659, 2005.

[YP06] J. Yan and M. Pollefeys. A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision (ECCV)*, volume 3954 of *Lecture Notes in Computer Science*, pages 94–106. Springer, 2006.

[ZHS07] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: clustering, classification, and embedding. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[Zie89] W. P. Ziemer. *Weakly Differentiable Functions*. Springer, New York, 1989.

[ZK93] S.K. Zavriev and F.V. Kostyuk. Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341, 1993.

[ZLPS11] L. Zappella, X. Lladó, E. Provenzi, and J. Salvi. Enhanced local subspace affinity for feature-based motion segmentation. *Pattern Recognition*, 44(2):454–470, 2011.

[ZYC+12] G. Zhang, Z. Yuan, D. Chen, Y. Liu, and N. Zheng. Video object segmentation by clustering region trajectories. In *International Conference on Pattern Recognition*, 2012.