Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Fakultät für **Informatik**

Institut für Technische Informatik

Lehrstuhl Industrielle Anwendungen der
Informatik und Mikrosystemtechnik (IAIM)

Automatische Gebäudemodellierung aus Laserscanning-Daten

Diplomarbeit
von

Jan Wassenberg

30.09.2006 – 29.03.2007

Hauptreferent	:	Prof. Dr.-Ing. Rüdiger Dillmann
Korreferent	:	Prof. Dr.-Ing. Uwe Hanebeck
Betreuer	:	Dr.-Ing. Hermann Groß
		Dr.-Ing. Tilo Gockel

Zusammenfassung

Wegen des zunehmenden Interesses an der Stadtmodellierung sowie der Verfügbarkeit großer Datenmengen ist eine automatisierte Gebäudeanalyse von Bedeutung. In dieser Arbeit wird ein Verfahren vorgestellt, um Laser-Daten auf Gebäudepolygone zu reduzieren. Dachflächen werden mittels Segmentierung und Hauptkomponentenanalyse berechnet. Zur Bestimmung der Gebäudegrenzen wird eine präzise und effiziente Implementierung der Hough-Transformation entwickelt. Das Verfahren wird anhand von synthetischen und realen Daten validiert; die Ergebnisse werden diskutiert.

Ich versichere hiermit, die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Die verwendeten Hilfsmittel und Quellen sind im Literaturverzeichnis vollständig aufgeführt.

Karlsruhe, den 29.03.2007

Jan Wassenberg

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Kapitel 1

Einleitung

In dieser Diplomarbeit wird eine Methode vorgestellt, mit der Gebäude in Laserscanning-Daten erkannt und als Polygone dargestellt werden können. Hierbei sind nur einfache Entfernungs-Messungen (ohne Intensitätskanal) erforderlich, sodass die Modellierung und Visualisierung eines Gebiets kostengünstig und mit gängigen Sensoren realisiert werden kann. Durch die verlustfreie Darstellung der Daten als 3D-Punktwolke kann eine hohe Genauigkeit erreicht werden.

Entwickelt werden hierzu Verfahren zur Segmentierung, Ausgleichsebenenberechnung und Linienextraktion.

1.1 Motivation und Zielsetzung

Die zunehmende Urbanisierung (in 2005 wohnten 49 % der Weltbevölkerung in Städten UN Department of Economic and Social Affairs (2005)) verstärkt das Interesse, Informationen über Stadtgebiete zu erhalten. Wegen der teils großen Ausmaße moderner Hochhäuser wären 3D-Modelle der Gebäude weitaus informativer als herkömmliche Karten oder Luftbilder. Zu den Anwendungen solcher Modelle zählen beispielsweise:

- Fremdenführung / virtuelle Stadterkundung,
- Einsatzplanung bei MOUT (Military Operations in Urban Terrain),
- Stadtplanung,
- Visualisierung im Allgemeinen.

Aus welchen Daten können solche Modelle gewonnen werden? Die stereoskopische Auswertung liefert räumliche Daten aus Luftbildern; allerdings erhält man dabei nur solche Strukturen, die aus der Kameraperspektive sichtbar sind. Eine verlässlichere Quelle ist das luftgetragende LIDAR (Light Detection And Ranging), auch ALS (Airborne Laser Scanner) genannt. Hierbei wird ein Laser-Entfernungsmesser in fast senkrechter Lage zur vermessenen Oberfläche verwendet, um die Szene abzutasten. Die gemessenen Punkte sind in Weltkoordinaten (oft UTM) gegeben, wobei heutzutage Auflösungen im

Bereich von 20 cm gängig sind. Das System ist kostengünstig und präzise. Für militärische Anwendungen oder beim Katastrophenschutz besteht ein weiterer Vorteil darin, dass es relativ unabhängig von Wetter und Helligkeit einsatzfähig ist.

Die LIDAR-Daten sollen nun analysiert werden. Da eine manuelle Verarbeitung zu zeitaufwändig und teuer ist, soll dieser Prozess automatisiert werden. In den Punktwolken sind die einzelnen Gebäude zu erkennen und über Polygone approximiert darzustellen.

Es ist aufschlussreich zu überlegen, warum Polygone überhaupt benötigt werden. Der erste Grund liegt in der Effizienz. Ein kleiner Datensatz von 100 m×100 m würde bei einer Auflösung von 20 cm bereits $2,5 \times 10^5$ Punkte enthalten. Flächen mit Punkten deckend darzustellen ist prinzipiell aufwändiger, als dafür Polygone zu verwenden. Aus dieser Sicht liegt das Primärziel also darin, die Daten zu komprimieren. Die Polygondarstellung ist weitaus handlicher als die ursprüngliche Punktdarstellung.

Polygone bieten auch weitere Vorteile: Ausreißer, Unstetigkeiten und Messfehler können korrigiert werden, sodass die wesentlichen Merkmale einer Szene leichter erkennbar werden. Schließlich kann man Texturen über die Polygone legen, was den Grad an Realismus beträchtlich steigert. Dies wäre mit einer Punktwolke nur eingeschränkt möglich.

1.2 Aufbau der Arbeit

Die vorliegende Arbeit ist wie folgt gegliedert:

2. In diesem Kapitel wird der Stand der Technik in der Gebäudemodellierung anhand einiger existierender Verfahren aufgezeigt. Dabei werden einige Grundlagen geschaffen, die zum weiteren Verständnis der Materie nötig sind.

3. Anschließend wird der Entwurf des neuen Verfahrens vorgestellt. Die angestrebten Verbesserungen und Ziele werden aufgezählt und der sich daraus ergebende Entwurf kurz geschildert.

4. Der Hauptteil besteht aus einer detaillierten Beschreibung des Verfahrens. Es erfolgen auch einige Hinweise zur Implementierung.

5. Hier werden die Ergebnisse vorgestellt, analysiert und bewertet. Neben der erzielten Genauigkeit und Performanz werden auch Schwachstellen und Möglichkeiten zu deren Beseitigung aufgezeigt. Schließlich erfolgt eine Abschätzung der Anforderungen des Verfahrens bezüglich der Datenqualität.

6. Der letzte Kapitel beinhaltet eine Zusammenfassung und Diskussion der Entwicklung und Ergebnisse. Schließlich werden einige möglichen Weiterentwicklungen aufgeführt.

Kapitel 2

Stand der Technik

2.1 Gebäudedetektion

Um in der Datenpunktwolke die Punkte zu finden, die ein Gebäude definieren, gibt es zwei Verfahren: Man kann pixelbasiert filtern oder segmentieren.

2.1.1 Pixelbasiert

Hier werden einzelne Pixel – womöglich unter Hinzunahme einer kleinen Menge benachbarter Pixel – betrachtet. Es folgt eine Aufzählung und kurze Diskussion der Punktmerkmale, die der Gebäudeklassifikation dienen können:

- Die Höhe des Punktes über dem Boden: Dies ist die naheliegenste Methode, erfordert aber die Kenntnis oder ein Modell der zu Grunde liegenden Landschaft.
- Die gemessene Rückstrahlintensität beziehungsweise Reflektivität der Oberfläche: Da diese Daten nicht immer vorliegen, sollen in dieser Arbeit nur Punktkoordinaten verwendet werden.
- Der Unterschied zwischen First Echo und Last Echo:[1] Wegen der unterschiedlichen Durchdringung des Lasers in Vegetation sind dort starke Abweichungen zu erwarten.
- Die Höhentextur: Bei Vegetation ist die zweite Ableitung der Höhen weitaus größer als bei Gebäuden.
- Die lokale Krümmung: Hierzu werden die Gradienten in vier Richtungen um den Punkt verglichen. Dachpunkte weisen im Vergleich zu Vegetationspunkte eher eine uniforme Orientierungen auf.

Diese werden in Hug (1997) vorgestellt und bewertet.

Bestimmung des Landschaftmodells

[1] Einige ALS-Systeme können das erste und letzte Echo eines Pulses aufzeichnen. Die unterschiedlichen Echos weisen auf verschiedene Oberflächenhöhen innerhalb des Laserstrahldurchmessers hin.

Wie oben besprochen, können Gebäude als größere zusammenhängende Strukturen oberhalb der Bodenhöhe erkannt werden. Zur Bestimmung des Bodens werden zwei Algorithmen kurz vorgestellt.

Elmqvist Elmqvist (2002) verwendet ein Active Contour Modell der Oberfläche, das man sich als Gummituch vorstellen kann. Durch Minimierung der Krümmung und der Entfernung zum Gelände werden Gebäude und Ähnliches entfernt. Die Wahl der Parameter ist allerdings kritisch – alle beteiligten Kräfte müssen sich in etwa entsprechen.

Briese und Pfeifer Briese u. a. (2002) führen eine hierarchische robuste Interpolation durch. Die Oberfläche wird durch gewichtete lineare Interpolation der Punkte bestimmt. Die Gewichte werden iterativ anhand einer Funktion der Entfernung zu den gemessenen Punkten aktualisiert. Beispielsweise liegen Gebäudepunkte oberhalb der anfänglichen Oberfläche und deren Gewichte werden weiter verringert. Um die Robustheit in Gebieten ohne Bodenpunkte zu erhöhen und die Konvergenz zu beschleunigen wird diese Optimierung hierarchisch auf einer Mipmap-Pyramide[2] der Daten durchgeführt.

Neben diesen oberflächenbasierten Methoden sind einige andere geläufig, die aber hier nicht relevant sind.

Mathematische Morphologie

Alternativ oder ergänzend zu anderen Methoden kann Morphologische Filterung zur Gebäudedetektion verwendet werden.

Die mathematische Morphologie Serra (1982) ist eine Methode der Bildverarbeitung basierend auf der Mengentheorie. Bilder werden mit einem 'Strukturelement'[3] bearbeitet. Die zwei grundlegenden Operationen sind Erosion und Dilatation. Die Erosion entfernt Pixel eines Merkmals wenn das Strukturelement an dieser Stelle nicht gänzlich im Inneren des Merkmals liegt. Bei der Dilatation werden Merkmale durch Hinzufügen der Punkte im Bereich des Strukturelements an jeder Stelle erweitert.

Die Relevanz für Bildverarbeitung ergibt sich mit den Operationen des Öffnens und Schließens. Beim Öffnen wird erodiert und dann dilatiert um kleine Objekte zu entfernen. Zum Schließen wird umgekehrt verfahren, was das Auffüllen kleinerer Löcher bewirkt. Somit können Gebäude von unregelmäßiger Vegetation getrennt werden.

2.1.2 Segmentierung

Bei der segmentbasierten Gebäudedetektion wird für komplette Regionen entschieden, ob sie ein Gebäude darstellen. Zunächst werden die Punkte in einen geeigneten Merkmalsraum abgebildet. Zur Bestimmung der zusammenhängenden Komponenten gibt es zwei grundsätzliche Verfahrensweisen. Entweder werden Punkte einzeln einer Region hinzugefügt ('Region Growing') oder der Raum wird aufgeteilt und Regionen teilweise

[2] Ein Mipmap der Ordnung n ist eine um 2^{-n} skalierte Version des Ursprungsbilds. Der Begriff stammt aus dem Lateinischen 'multum in parvo'. Pyramide beschreibt die aufeinandergestapelten Mipmaps.

[3] Wir stellen uns eine Schablone vor. Normalerweise wird ein rechteckiges Fenster verwendet.

erneut zusammengefasst ('Split and Merge'). Es folgt eine kurze Vorstellung relevanter Arbeiten in diesem Gebiet.

ITK (Yoo, 2004, §9) beschreibt einige Verfahren. Insbesondere ist die Watershed-Segmentierung von Interesse. Hierbei werden Merkmale durch Bestimmung der Gradienten abgegrenzt und Regionen durch 'Auffüllen' der so entstehenden 'Bassins' erzeugt. Dieses Verfahren ist relativ unempfindlich in Bezug auf die Wahl der Parameter.

Jiang und Bunke Jiang und Bunke (1994) benutzen eine einfache Beobachtung, um die Segmentierung zu beschleunigen. Der Schnitt einer Region mit einer Bildzeile muss ein verbundenes Geradenstück ergeben. Statt Pixel werden also diese Stücke als Primitive verwendet um den Datenbestand zu verringern. Alle Stücke innerhalb einer Zeile sind miteinander verkettet und können beim Region Growing leicht verbunden werden.

Rodrigues und Loke Rodrigues und Loke (2000) verwenden einen Octree[4] zur Minimierung des Rauschens. Segmentierung beziehungsweise Clustering wird erst ab einer gewissen Höhe im Baum durchgeführt. Dadurch wird Rauschen unterdrückt und die Klassen können besser getrennt werden.

2.2 Gebäudemodellierung

Nachdem Gebäude detektiert wurden, verbleibt noch, ihre Randpunkte durch Polygone zu approximieren. Im Folgenden werden einige gängige Verfahren hierzu beschrieben.

Haala und Brenner Haala und Brenner (1999) verwenden Katasterkarten, um Gebäudeumrisse zu erhalten. Allerdings sind diese nicht für alle Einsatzgebiete erhältlich. Die Gebäudeumrisse werden rekursiv in rechteckige Primitive aufgeteilt, für die anschließend Parameter eines Dachmodells geschätzt werden. Mit Methoden der CSG[5] werden benachbarte Flächen verschmolzen und unsichtbare Elemente entfernt. Fehler in der Segmentierung und Modellierung werden interaktiv behoben. Wegen der Notwendigkeit der Benutzerinteraktion ist die Verarbeitung aufwändiger.

Maas und Vosselman Maas und Vosselman (1999) stellen zwei Algorithmen vor, die auf Punktwolken arbeiten und verlustbehaftete Rasterung vermeiden. Im ersten Verfahren werden die Gebäude mittels Schwellwert und morphologischen Filter erkannt und segmentiert. Danach können die Parameter eines einfachen Giebeldachmodells mit der Methode der invarianten Momente in geschlossener Form angegeben werden. Das dadurch bestimmte Dach wird erzeugt und mit den gemessenen Daten verglichen. Bei Unstimmigkeiten kann die Dachhypothese verworfen werden oder es können Feindetails modelliert werden. Der Nachteil liegt darin, dass nur bekannte Gebäudetypen modelliert werden können. Der zweite Algorithmus verwendet die Schnittlinien der Dachebenen. Die Ebenen werden über Clustering der Parameter erhalten und anschließend mit der Methode der kleinsten Quadrate optimiert. Da alle Punkte der Ebene in die Berechnung mit eingehen, können diese präzise bestimmt werden. Der Dachrand wird wie folgt erhalten: Von einem zufälligen Punkt ausgehend konstruiert man eine Linie

[4] Ein Baum zur rekursiven 3D-Raumaufteilung.
[5] Constructive Solid Geometry

mit Winkel parallel zur Hauptachse des Daches. Punkte werden bis zu einer gewissen Entfernung zum Rand hinzugefügt. Anschließend dreht man 90 Grad nach links beziehungsweise rechts und fährt fort. Diese Methode liefert 'schöne' Konturen weil sie garantiert rechteckig sind, sie ist aber nicht robust in Hinblick auf Störungen (beispielsweise durch benachbarte Bäume). Mit dem Dachrand können die Wände konstruiert werden. Um das Dach zu erhalten werden die Schnittlinien der Dach- und Wandebenen erzeugt und benachbarte Endpunkte verschmolzen.

Elaksher und Bethel Elaksher und Bethel (2002) verfolgen einen anderen Ansatz. Punkte werden als Gebäude klassifiziert, wenn sie mindestens 5 m höher als ein Nachbarpixel sind. Ebenen werden über Clustering im Parameterraum bestimmt. Eine Besonderheit hierbei ist, dass annähernd horizontale oder symmetrische Ebenen korrigiert werden. Die Form des Daches geht aus den Verbindungen der benachbarten Kreuzungspunkte hervor. Diese punktbasierte Methode ist allerdings weniger genau als eine geometrische Lösung, weil dort mehr Punkte in die Berechnung eingehen.

Söderman und Ahlberg Söderman u. a. (2004) verwenden eine ähnliche, aber komplexere Methode. Die Daten werden zunächst gerastert. Dies verringert den Rechenaufwand, reduziert allerdings durch Quantisierung die Genauigkeit der Koordinaten. Mittels Active Contour wird das Gelände bestimmt und von den Punkthöhen abgezogen. Gebäude werden dann mithilfe eines neuronalen Netzes erkannt. Die Normalenvektoren der Punkte werden mit Least-Squares geschätzt und zu Ebenen gruppiert. Die genauen Parameter der Ebene werden durch iterative, neu gewichtete Schätzung bestimmt. Punkte, bei denen sich die Ebenen-Nachbarschaften ändern, werden als 'Topologische Punkte' definiert. Die Verbindungslinien zwischen diesen Punkten werden mit der 2D-Hough-Transformation geglättet und die Topologischen Punkte angepasst. Das Dach entsteht durch Verbinden dieser Punkte.

Rottensteiner und Briese Rottensteiner und Briese (2002) schlagen einen Algorithmus vor, der auf 'polymorphe Merkmalsextraktion' basiert. In der Vorverarbeitung wird der Boden durch hierarchische robuste Interpolation bestimmt. Nach Anwendung eines Schwellwertoperators und einer morphologischen Öffnung gehen die Gebäude durch iterative Segmentierung hervor. In einem zweiten Schritt werden verbleibende Punkte erneut segmentiert, dieses Mal mit größeren Toleranzen. Regionen mit zu hoher Höhenvarianz stellen vermutlich Vegetation dar und werden entfernt. Die Nachbarschaftsrelationen der Regionen werden über ein Voronoidiagramm bestimmt. Da die Regionen noch unbereinigt sind, ist die Qualität dieser Daten nicht gesichert. Die Gebäude werden als lediglich als Prismen dargestellt, die durch die Voronoi-Regionen begrenzt sind.

Oda und Takano Oda u. a. (2004) verfahren ähnlich. Die Segmentierung zur Gebäudeklassifikation akzeptiert geringe Höhenunterschiede in den Daten. Anschließend werden Gebäudepunkte morphologisch gefiltert. Nach Anwendung des Canny-Operators[6] werden Linien per Hough-Transformation aus den Randpunkten extrahiert. Die Endpunkte dieser Linien werden zu einem Polygon verbunden. Schließlich werden möglichst passende Texturen ausgewählt und auf die Wände projiziert.

Rottensteiner et alii Rottensteiner u. a. (2005) stellen einen neuen und vielverspre-

[6] Ein Filter zur Kantendetektion.

chenden Algorithmus vor. Die bereits erkannten Gebäudepixel werden anhand ihrer Normalenvektoren segmentiert. Um Lücken zu überwinden werden die Grenzen der Voronoi-Regionen verwendet. Mit statistischen Tests wird entschieden, ob die Trennlinien der Regionen zu verwerfen, kombinieren oder akzeptieren sind. Dadurch kann man einige Konstanten eliminieren. Allerdings ist unklar, ob mit einem stochastischen Modell die Gegebenheiten der ALS-Daten berücksichtigt werden. Anschließend werden die vorangegangen Schritte wiederholt, wobei die Polygone die segmentierten Flächen begrenzen. Dies verbessert die Ergebnisse, allerdings vermutlich auf Kosten höherer Laufzeit. Schließlich werden einzelne Eckpunkte iterativ aufeinander bewegt und die komplette Dachgeometrie wird regularisiert.

Kapitel 3

Entwurf

3.1 Angestrebte Verbesserungen

Kein existierendes Verfahren zur Gebäudemodellierung kann alle Problemstellungen abdecken. Für die Weiterentwicklung des heutigen Stands der Technik wäre es erforderlich, die folgenden Schwierigkeiten und Kompromisse zu lösen:

- Rasterung der Punkte bietet zwar Vorteile hinsichtlich der Laufzeit, verschenkt aber Genauigkeit. Weiterverarbeitung der unveränderten 3D-Koordinaten vermeidet Quantisierungsfehler. Zusätzlich gibt es die Möglichkeit mehr Informationen zu berücksichtigen. Je nach Lage des verwendeten Sensors werden nicht nur Daten aus senkrechter Blickrichtung geliefert, es sind auch manchmal Wände sichtbar. Im Gegensatz zur 2,5D-Bilddarstellung enthalten Punktwolken diese Informationen, die eventuell zur scharfen Abtrennung der Gebäudekonturen nützlich sein können.
- Verstellbare Werte, die keine direkten und reellen physikalischen Größen darstellen (wie beispielsweise die interne Energie im Active Contour-Modell), sind nur schwer verifizierbar. Sind die Parameter nachvollziehbar, können das Verfahren und seine Ergebnisse besser analysiert werden.
- Annahmen über das Rauschverhalten der gemessenen Punkte führen vermutlich zu Problemen, wenn der Sensor gewechselt wird. Wenn ein Erkennungsverfahren unabhängig von den Gegebenheiten der Datengewinnung arbeitet, ist es eher universell einsetzbar. Insbesondere sollte das Verfahren auch mit unverrauschten synthetischen Daten funktionieren.[1]
- Spezielle Modellannahmen über die zu erkennenden Gebäude vereinfachen das Problem, schränken aber auch die Anwendbarkeit des Verfahrens ein. Wenn das Modellierungsverfahren allgemein gehalten wird, können unerwartete Fälle mitunter besser abgedeckt werden.

[1] Bei unverrauschten synthetischen Daten sind die Strukturen scharf abgegrenzt, was zu Problemen bei der Verschmelzung führen kann.

- Punktbasierte Filter zur Erkennung der Dachflächengrenzen sind relativ ungenau. Wenn alle Flächenpunkte mit in die Berechnung eingehen, ist die Wahrscheinlichkeit für ein korrektes Ergebnis größer.
- Rein auf Geometrie basierende Analyseverfahren können manche Dachkonfigurationen nicht vollständig erkennen. Eine zusätzliche Betrachtung der ursprünglichen Messpunkte liefert mehr Information.
- Katasterdaten würden zur Abgrenzung der Gebäude behilflich sein, sind aber nicht für alle Einsatzgebiete erhältlich. Das Verfahren muss auch ohne diese Angaben funktionieren. Um dies sicherzustellen, sollten sie nicht vorausgesetzt werden.
- Ein Eingreifen des Benutzers kann die Ergebnisse verbessern, erhöht aber auch die Kosten der Verarbeitung. Vollständige Automatisierung ist insbesondere bei großen Datenmengen unerlässlich.

Wir werden im Folgenden sehen, inwiefern diese Vorstellungen erfüllt werden können.

3.2 Entwurfsziele

Mögliche Kriterien für den Erfolg eines Gebäudemodellierungsverfahrens sind:

Echtzeitfähigkeit Die Verarbeitungszeit des Verfahrens wird minimiert;

Genauigkeit Die Objekte sind realitätsgetreu dargestellt;

Vollständigkeit Alle relevanten Gebäude werden erkannt;

Detaillierungsgrad Auch kleine Dachstrukturen werden korrekt erfasst.

Welche Kriterien davon können erfüllt werden und wie sind die Wechselwirkungen? Vollständigkeit steht im Gegensatz zu Genauigkeit, weil hierfür womöglich Annahmen getroffen oder vorhandene Daten ‘gedehnt’ werden müssen. Auch geht bessere Qualität generell mit höherer Verarbeitungszeit einher. Wegen diesen Widersprüchen müssen Prioritäten gesetzt werden.

Echtzeitfähigkeit wird in dieser Arbeit als nicht relevant eingestuft. Für die erdachten Anwendungen ist es kein Problem, die Daten offline zu verarbeiten.[2]

Ferner ist man in den oben aufgeführten Anwendungsfällen meist bereit, auf Feindetails zu verzichten. In der ersten Ausbaustufe des entwickelten Verfahrens soll es genügen, die generelle Form des Gebäudedachs zu erkennen.

Bleibt die Frage bezüglich der Kriterien Genauigkeit und Vollständigkeit. Das Hauptkriterium ist Wiedererkennbarkeit – die wichtigsten Gebäudemerkmale müssen erkennbar sein. Es wird als unwesentlich angesehen, ob die Abweichung von den echten Gegebenheiten 10 cm oder 20 cm beträgt. Man versucht also, Gebäude vollständig zu erfassen und wird dabei geneigt sein, leichte Ungenauigkeiten in Kauf zu nehmen.

[2] Offline bedeutet, das Verfahren hat unbeschränkt viel Zeit für die Verarbeitung zur Verfügung.

3.3 Realisierungsweg

Der eingeschlagene Realisierungsweg wird kurz vorgestellt und es wird erläutert, wie die oben genannten Verbesserungen zu erzielen sind.

Die Daten werden durchgängig als 3D-Punktwolke verarbeitet. Dies erhöht die Rechenzeit und den Speicherbedarf, dafür aber auch die Auflösung und Genauigkeit. Bildbasierte Verfahren wie beispielsweise Erosion sind nicht mehr direkt anwendbar. Um ähnliche Wirkung zu erzielen, betrachtet man die Nachbarschaften der Punkte. Dies wird durch eine Raumaufteilungsdatenstruktur ermöglicht.

Das Analysieren der Gebäude in den Daten geschieht in einer fortlaufenden Bearbeitungskette. Entsprechend der Grundidee, die Daten zu komprimieren, bestimmt man zunächst Ausgleichsebenen, berechnet deren Schnittlinien und dann wiederum die Schnittpunkte. In jedem Schritt wird die Datenmenge reduziert und dafür die Abstraktionsebene erhöht.

Welchen Vorteil lässt die Verwendung von Ebenen erwarten, anstatt direkt Linien oder gar Eckpunkte aus den Daten zu extrahieren? Der Vorteil liegt darin, dass bei der Berechnung einer Ebene weitaus mehr Punkte berücksichtigt werden, sodass die Ergebnisse verlässlicher sind. Die Schnittlinien der Ausgleichsebenen und deren Kreuzungen werden dann geometrisch bestimmt.

In Abschnitt 3.1 wird erklärt, dass die geometrischen Schnittpunkte allein nicht hinreichend sein können, um alle Dachgegebenheiten zu erkennen. Deshalb vermerkt man für alle geometrische Elemente welche echten Messpunkte in der Nähe liegen. Unstetigkeiten in dieser Punktmenge weisen auf Dacheigenschaften hin, die noch nicht identifiziert wurden. An diesen Stellen werden Endpunkte eingefügt, die die Schnittlinien in Geradensegmente aufteilen.

Mit Erhalt der Schnittpunkte und den zusätzlichen Endpunkten ist das Dach bereits bestimmt, danach muss nur noch die Topologie ermittelt werden. Dies geschieht durch Verbinden dieser Punkte entlang der Schnittlinien.

Dieser Methode liegen keine Annahmen über die Beschaffung der Gebäude zu Grunde; sie sollte deswegen allgemeingültig sein.

Kapitel 4

Verfahren

Das Verfahren erkennt Gebäude in einer ALS-Punktwolke und stellt diese als Polygone dar. Die grundsätzliche Vorgehensweise besteht darin, Dachflächen und Wände zu extrahieren und daraus mittels Geometrie die Trennlinien und die Eckpunkte zu bestimmen, aus denen dann die Polygone hervorgehen. Die einzelnen Verarbeitungsschritte werden im nachfolgenden Text eingehend beschrieben.

4.1 Vorverarbeitung der Punkte

Anfangs sind lediglich die (x, y, z)-Koordinaten der gemessenen LIDAR-Punkte gegeben. In den folgenden Schritten werden zusätzliche Informationen benötigt:

- ob der Punkt am Rand des Gebäudes liegt;
- Normalenvektor der Tangentialebene und
- ein Maß für die 'Glätte' der Umgebung des Punktes.

4.1.1 Punktdatenstruktur

Diese Angaben werden in der Vorverarbeitung ermittelt und in einer Point-Datenstruktur gespeichert. Neben den Punktkoordinaten sind dort auch noch Felder für Ebenen- und Linienkennungen vorhanden. Darin kann notiert werden, zu welcher Ebene und Linie der Punkt gehört beziehungsweise in wessen Nähe er liegt.

> *Implementierungsvermerk: Größe der Datenstruktur*
>
> Durch die Kompression einiger Felder erreicht man eine runde Größe von 32 Byte pro Punkt. Eine solche Zweierpotenz erlaubt eine effiziente Indizierung insofern, dass sich Multiplikation durch eine (schnellere) Bitverschiebung realisieren lässt.

Es folgen Details zur Bestimmung der oben genannten Informationen.

4.1.2 Kennzeichnung der Randpunkte

Im späteren Verlauf sollen Gebäudewände aus den Daten gewonnen werden. Die Hough-Transformation (siehe Abschnitt A.4) kann verwendet werden um Linien aus einer Punktmenge zu extrahieren. Zur Berechnung der Wände benötigt man entsprechend die Menge der Punkte, die am Rand des Gebäudes liegen.

Für zweidimensionale Bilder sind hierfür gute Verfahren bekannt; beispielsweise sei das Filter von Canny Canny (1986) genannt. Da aber der Informationsverlust bei der Rasterung der Punkte nicht akzeptabel ist, sollen die Punkte direkt als 3D-Punktwolke weiterverarbeitet werden.

Es wird ein Algorithmus zur Gebäuderanderkennung in Punktwolken vorgeschlagen. Zunächst sammelt man alle Umgebungspunkte. Diese gehören nicht zum Dach, liegen aber in der xy-Nähe[1] eines Dachpunkts und weisen einen signifikanten Höhenabfall auf. Danach werden umgekehrt Dachpunkte als Rand markiert, falls sie diejenigen Punkte sind, die einem Umgebungspunkt am nächsten liegen (wieder ohne Berücksichtigung der Z-Komponente).

Entscheidend ist dabei, die Menge der Nachbarschaftspunkte möglichst effizient zu bestimmen. Diese Aufgabe wird sich aufgrund der Punktwolken-Darstellung noch öfters stellen. In Anhang A.2 wird hierzu eine Lösung vorgestellt, zunächst genügt es jedoch `k_nearest_xy` als bekannt vorauszusetzen.

Der Algorithmus beinhaltet also im Detail:

Algorithmus 1 : Bestimmung der Randpunkte.

Eingabe : dachpunkte
Ausgabe : randpunkte, umgebungspunkte

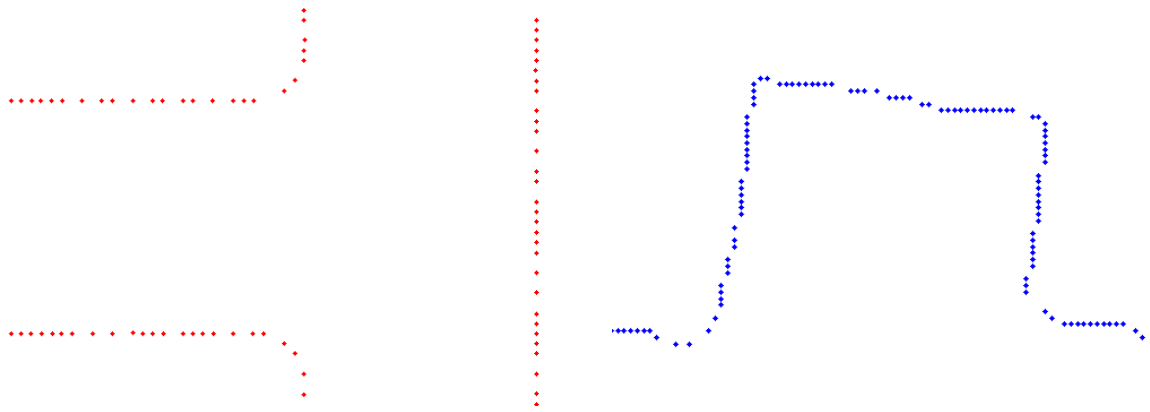
```
1  umgebungspunkte = ∅
2  für jedes p ∈ dachpunkte tue
3      nachbarn = k_nearest_xy (dachpunkte, p, K)
4      alte_anzahl = |umgebungspunkte|
5      für jedes n ∈ nachbarn tue
6          wenn n.Z ≤ p.Z −Δ dann
7              markiertEinfuegen (umgebungspunkte, n)
8      wenn alte_ anzahl ≠ |umgebungspunkte| dann
9          vorläufige_randpunkte ← vorläufige_randpunkte ∪ {p}
10 randpunkte = ∅
11 für jedes p ∈ umgebungspunkte tue
12     nächstgelegener_randpunkt = k_nearest_xy (vorläufige_randpunkte, p, 1)
13     markiertEinfuegen (randpunkte, nächstgelegener_randpunkt)
```

Es gibt hierbei einige Besonderheiten zu beachten. `markiertEinfuegen` fügt einen Punkt einer Menge hinzu, wobei der Punkt markiert wird, falls er nicht schon in der Menge vorkommt. Durch diese Datenstruktur lässt sich mit der Komplexität $O(1)$ entscheiden, ob ein Punkt zur Menge gehört und ferner bestimmen, welcher der nächste

[1] Höhenunterschiede der Punkte sollen die Nachbarschaftsrelation nicht beeinflussen.

Punkt in der Menge ist. Da diese Operationen oft benötigt werden, ist das Aufrechterhalten der redundanten Darstellung sinnvoll. Um festzustellen, ob ein vorläufiger Randpunkt derjenige ist, der einem Umgebungspunkt am nächsten liegt, müssen alle vorläufigen Randpunkte bekannt sein. Deswegen ist die separate Schleife über alle Umgebungspunkte notwendig. Schließlich sei noch angemerkt, dass das 'Abfallprodukt' umgebungspunkte später noch nützlich sein wird. Damit wird die Höhe des Gebäudefundaments errechnet.

Zusammenfassend können mit diesem Algorithmus Gebäuderänder in einer Punktwolke effizient und mit zufriedenstellender Genauigkeit erkannt werden. Die so gewonnene Randlinie ist sogar nur einen Punkt dick, was für die Hough-Transformation vorteilhaft ist. Diese Eigenschaft wird in Abbildung 4.1 ersichtlich, in der Ausschnitte der Randpunkte zweier Datensätze dargestellt sind. Bei den 'fom'-Daten ist das Aliasing (treppenartige Linien) auffällig. Lücken in den 't'-Daten werden durch das Verrauschen der Punktkoordinaten verursacht.



(a) 't' (synthetisch) (b) 'fom'

Abbildung 4.1: Randpunkte zweier Gebäude. (zeigt die ein-Punkt-Breite)

4.1.3 Berechnung der Tangentialebene

Der nächste Schritt in der Vorverarbeitung besteht darin, die Tangentialebene für jeden Punkt zu berechnen. Diese liefert den Normalenvektor und ein Maß der lokalen Krümmung, die jeweils in der Point-Datenstruktur abgelegt werden. Beide Informationen werden später vom Segmentierer verwendet.

Die Tangentialebene wird dadurch approximiert, dass eine Ausgleichsebene aus der Nachbarschaft des Punktes berechnet wird (siehe dazu Abschnitt A.1).

Dimensionierung der Nachbarschaft

Im Weiteren ist zu klären, wie die Nachbarschaft des Punktes gewählt werden soll: Die nächsten k Nachbarpunkte (k-Nearest Neighbor, KNN) oder auf eine maximale Entfernung beschränkt (Fixed-Distance Nearest Neighbor, FDNN). Im Normalfall (im Inneren einer Dachfläche) sind beide Vorgehensweisen äquivalent, da die Punktdichte uniform ist und somit die k nächsten Nachbarn näherungsweise eine δ-Umgebung

bilden. Aus dieser Überlegung wird schon ein Vorteil des KNN bemerkbar: Es ist näherungsweise unabhängig von der Punktdichte.

An Randpunkten wird die Sachlage jedoch interessant: Es kann sein, dass das KNN-Verfahren zum Beispiel Bodenpunkte der Umgebung eines Dachrinnenpunktes hinzurechnet. Somit würde der Normalenvektor des Dachpunktes etwas variieren. Dies bewirkt eine schärfere Trennung des Daches gegenüber dem Boden. Trotzdem würden die Randpunkte ins Segment übernommen werden, da die Nachbarschaft der Punkte im Inneren des Daches ausreichend groß ist, um sie auf anderem Wege mit einzuschließen.

Aufgrund der genannten Überlegungen wird im Weiteren das KNN-Verfahren eingesetzt.

Zuweisung des Normalenvektors

Es wurde ausgehend von einem Punkt die Nachbarschaft bestimmt und eine Ausgleichsebene dieser Punkte berechnet. Jetzt stellt sich die Frage, ob der daraus entstandene Normalenvektor dem Ursprungspunkt zugewiesen werden soll oder dem zentralen Punkt der Nachbarschaft.[2]

Im einfachen Fall des Dachinneren wird kein Unterschied erwartet. Da die Messpunkte annähernd gleich verteilt sind, ist der Ursprungspunkt vermutlich identisch mit dem zentralen Punkt.

Wie bereits bei der Definition der Nachbarschaft wird die Fragestellung beim Gebäuderand interessant. In Abhängigkeit davon, welche Nachbarschaftspunkte herangenommen werden, kann es sein, dass manche für den Normalenvektor ausschlaggebenden Punkte fehlen. Zum Beispiel kommt es bei ALS vor, dass Dachrandpunkte fehlen, dafür aber Punkte hinzukommen, die von Hauswandreflexionen herrühren.

Das Problem besteht also darin, dass Nachbarschaften, zum Beispiel von Randpunkten, potenziell nicht genau deren Ursprungspunkt repräsentieren. Aufgrund dieser Unsicherheit ist es sinnvoll nur dem zentralen Punkt diese Werte zuzuweisen, da dort die Aussage am ehesten gesichert ist.

Allerdings stellt sich heraus, dass zirka 30 % der Punkte keine zentralen Punkte einer Nachbarschaft sind, also nie einen Normalenvektor zugewiesen bekämen. Da sie dann auch nicht segmentiert würden, muss nachgebessert werden. Die naheliegende Lösung ist, die Informationen an den Ursprungspunkt *und* den zentralen Punkt zu übergeben; somit wird garantiert jedem Punkt ein Normalenvektor zuteil. Beim zentralen Punkt darf jedoch der vorherige Wert überschrieben werden, weil der berechnete Normalenvektor in diesem Fall für zuverlässiger gehalten wird.

4.1.4 Klassifizierung der Punkte

Als nächstes muss die Punktwolke klassifiziert werden, wobei ausschließlich von Interesse ist, ob Punkte zu einem Gebäude gehören oder nicht. Da diese Fragestellung nicht

[2]Der zentrale Punkt ist derjenige, der dem Schwerpunkt der Nachbarschaft am nächsten liegt.

Hauptziel der Arbeit ist, kann sie nicht eingehend untersucht werden. Eine einfache Methode hat sich jedoch bewährt.

Die Segmentierung bezüglich Normalenvektoren der Punkte[3] kann effektiv Gebäude und deren (stückweise) planare Flächen erkennen. Das Segmentierungsverfahren wird in Abschnitt A.3 eingehend beschrieben.

Die Gegebenheiten des Segmentierens im Zusammenhang mit der Klassifikation werden kurz vorgestellt. Gemessene Punkte der Vegetation tendieren dazu, stark unterschiedliche Normalenvektoren aufzuweisen. Hug (1997) Es entstehen dort folglich kleine Segmente, die durch ihre Größe gefiltert werden können. Flache Bodengebiete können durch niedrige Varianz bezüglich der Regressionsebene des Segments erkannt werden. Es verbleiben somit lediglich Segmente, die aus Dachebenen bestehen.

Bei (synthetischen) Daten mit sehr exakt definierten Dachflächen können diese allerdings nicht zusammengeführt werden. Selbst bei Wahl der Parameter, die die Untersegmentierung sehr begünstigen, ist der Unterschied der Normalenvektoren zu groß. Deshalb werden weiterhin in einem gesonderten Schritt alle benachbarten Dachsegmente zu einem Gesamtdach zusammengeführt.

Diese Klassifikationsregeln sind ausreichend um Gebäude in den Testdaten zu erkennen. Um verlässliche Klassifikation zu gewährleisten, müssen jedoch aufwändigere Verfahren zum Einsatz kommen.

4.1.5 Trennung der Gebäude

Für die weiteren Schritte werden die Daten der einzelnen Gebäude in Form einer Punktwolke benötigt. Der obige Klassifikationsschritt liefert bereits für jedes Gebäude eine Liste der beteiligten Punkte, diese müssen also lediglich in eine Punktwolken-Datenstruktur umorganisiert werden.

4.1.6 Speicherung der Punkte

Implementierungsvermerk: Punktwolken-Datenstruktur

Die Punkte werden durch relativ große Point-Verbünde dargestellt. Da diese oft zwischen Punktwolken den Besitzer wechseln, werden nur Verweise gespeichert. Die Punktdaten sind sequenziell abgelegt um Speicher zu sparen und die Adressierung zu beschleunigen. Beim Erstellen neuer Punktwolken können diese in eine neue, zusammenhängende Ablage bewegt werden, was die örtliche Lokalität im Speicher verbessert. Dies ist insbesondere in Hinblick auf Caching wichtig – die Rechengeschwindigkeit wird heutzutage mehr und mehr durch den vergleichsweise langsameren Speicher begrenzt.

Die PointCloud-Datenstruktur besteht also aus einer Liste von Zeigern auf Punkten sowie einer Referenz auf die Reihung der tatsächlichen Point-Instanzen.

Zwischenspeicherung der vorverarbeiteten Punkte

[3] Genauer: Normalenvektor der Tangentialebene; siehe dazu Abschnitt 4.1.3.

Die Dauer der Vorverarbeitung ist sicherlich, insbesondere bei größeren Punktmengen, nicht vernachlässigbar. Besonders wenn das gleiche Gebäude mehrfach analysiert werden soll (zum Beispiel mit veränderten Parametern), wäre es verschwenderisch die Vorverarbeitung jedes Mal zu wiederholen. Es ist von Vorteil, die Ergebnisse zwischenzuspeichern und wiederzuverwenden. Dies wird ermöglicht, indem die vorverarbeiteten Point-Datenstrukturen in einer Datei abgelegt werden. Existiert diese Datei bereits, können die fertigen Punkte direkt eingelesen werden, ansonsten werden sie neu angelegt.

Implementierungsvermerk: Überprüfung der Aktualität

Man beachte hierbei eine Gefahr: Falls Parameter geändert werden, welche die Vorverarbeitung beeinflussen, dürfen die nun veralteten Dateien nicht mehr verwendet werden. Es ist aber nicht zu erwarten, dass der Benutzer immer die betroffenen Dateien löscht. Falls das unterbleibt werden die Auswirkungen der Parameteränderung nicht sichtbar, was sehr gefährlich sein kann.

Folglich ist eine automatische Überprüfung der Aktualität vorgesehen. Für jede Datei wird der Zustand ihrer Abhängigkeiten festgehalten. Statt einer Prüfsumme des Inhalts wird der Effizienz halber der Zeitstempel und die Dateigröße betrachtet. Ändert sich einer der beiden Werte, muss die zwischengespeicherte Datei neu erzeugt werden.

Bei den vorverarbeiteten Punkten ist zunächst die Datei mit den ursprünglichen Punkten als erste Abhängigkeit angegeben. Um die Programmkonstanten mit einzubringen, legt man ein Protokoll (Textdatei) an. Darin soll vermerkt werden, welche Konstante wie geändert wurde. Neben dem nützlichen Effekt des Buchführens kann so automatisch bemerkt werden, dass die Punkte aufgrund der Parameteränderung neu vorverarbeitet werden müssen.

Zwischenspeicherung der ursprünglichen Punkte

Auch die ursprünglichen Punkte werden zwischengespeichert. Dies mag unnötig erscheinen, wenn die Ergebnisse der Vorverarbeitung bereits vorliegen. Es ist aber dann von Vorteil, wenn die Vorverarbeitung wiederholt werden muss. Dies ist insbesondere der Fall, wenn relevante Parameter geändert werden.

Die Zeitersparnis wird erreicht, indem die Koordinatentupel in Binärform abgespeichert werden (ursprünglich liegen sie als Text vor). Sie müssen also nicht mehr relativ aufwändig ins Gleitkommaformat konvertiert werden.

Der Einfachheit halber werden die Koordinaten auch hier in Point-Datenstrukturen abgelegt. Dies erlaubt ein gemeinsames Dateiformat und erübrigt separate Logik zum Einlesen. Obwohl die restlichen, erst in der Vorverarbeitung erzeugten, Felder nicht benötigt werden, spart man trotzdem noch Platz. Die Textdarstellung dreier Gleitkommazahlen braucht etwa 50 % mehr Speicherplatz als die binäre Darstellung inklusive Zusatzfelder.

Implementierungsvermerk: Transformation der Koordinaten

Die Zwischenspeicherung eröffnet die Möglichkeit, einen weiteren Vorteil zu erschließen. In der Rohform sind ALS-Daten oft georeferenziert. Die Koordinatenwerte sind mitunter größer als akkurat im 32-Bit-IEEE-754-Gleitkommaformat dargestellt werden kann. Beim Einlesen speichert man sie deswegen im größeren 64-Bit Format. Die Punkte werden anschließend in ein lokales Koordinatensys-

tem mit Ursprung $(0, 0)$ transformiert.[4] Erst dann legt man sie in der Point-Datenstruktur ab; somit wird keine Genauigkeit eingebüßt.

Man beachte, dass diese Transformation für alle Punkte der Szene durchgeführt wird. Gebäude sind zu dem Zeitpunkt noch nicht separiert, sodass diese auch weiterhin unterschiedliche Koordinatenursprünge aufweisen. Damit wird es möglich, die fertig rekonstruierten Gebäude wieder zu einer Szene zusammenzusetzen.

4.2 Berechnung der Dachebenen

Nun soll das Dach in Flächen unterteilt werden. Hierzu kann der vorhin erwähnte Segmentierer wiederverwendet werden. Dank den sogenannten 'templated policy classes' Alexandrescu (2001) sind keine Anpassungen am Code nötig. Es werden lediglich die neuen Parameter übergeben. Die wichtigste Änderung besteht darin, dass das Homogenitätskriterium wesentlich verschärft ist. Punkte werden nur zu einer Dachfläche gerechnet, falls ihr Normalenvektor den Normalenvektoren der anderen Punkte relativ ähnlich ist (zirka 12 Grad).

Implementierungsvermerk: Segmentierungsparameter

Der folgende Code-Ausschnitt veranschaulicht die Übergabe der Parameter an den Segmentierer:

```
struct PlaneTraits
{
   static const float cos_threshold;
   ..
};
const float PlaneTraits::cos_threshold = ..

template<class Traits>
class Segmenter
{
   bool isCompatible(
      const Point* pt_seed,
      const Point* pt_candidate
   ) const
   {
      ..
      if(cos < Traits::cos_threshold)
         ..
   }
};
```

[4] Da eine spätere Rücktransformation der Punkte unnötig erscheint, wird die Größe der Verschiebung nicht gespeichert.

4.2.1 Probleme

Mit der vorangegangenen Segmentierung sind die Flächen noch nicht hinreichend exakt bestimmt. Abbildung 4.2 weist auf folgende Probleme hin:

1. Ebenen werden durch Unstetigkeiten in der Dachoberfläche (zum Beispiel Dachgauben) verfälscht.
2. Benachbarte Flächen, die leicht schief zueinander liegen, werden unberechtigterweise zusammengefasst.
3. Randpunkte einer Ebene ragen bis zur Toleranzgrenze in ein anderes hinein.
4. Flächen mit 'Hindernissen' dazwischen werden nicht verschmolzen.

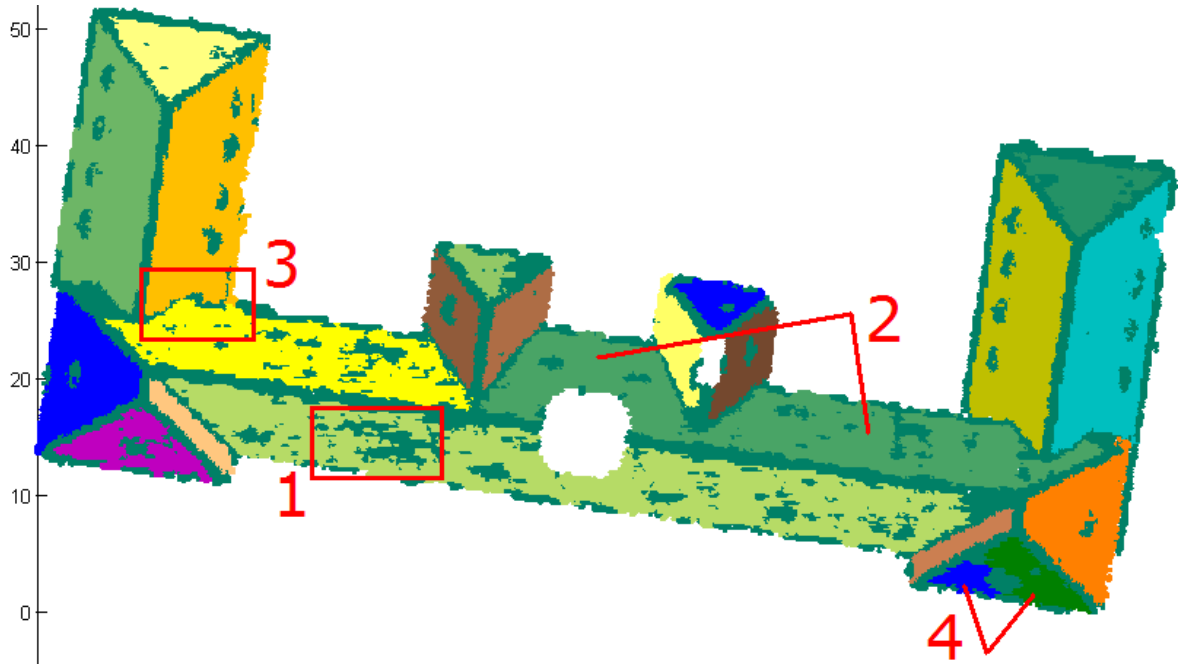


Abbildung 4.2: Probleme mit den segmentierten Dachflächen. (Punkte sind entsprechend ihrer Ebenenzugehörigkeit eingefärbt. Nummerierte Problemstellen werden im Text erläutert)

Ein Optimierungsverfahren wurde entwickelt um die Dachflächen mit hoher Präzision bestimmen zu können. Zunächst werden verbleibende Punkte segmentiert und somit neue Dachflächen erzeugt. Für jede Fläche werden Ausreißer-Punkte entfernt, die Ausgleichsebene neu berechnet und anschließend naheliegenden Punkte aufgenommen. Danach werden ähnliche Flächen verschmolzen und entartete Flächen[5] entfernt. Diese Schritte werden nachfolgend im Detail erläutert.

[5]Entartete Flächen sind solche, die scheinbar unecht oder ungültig sind.

4.2.2 Segmentierung verbleibender Punkte

In diesem Schritt werden alle momentan verbleibenden Punkte segmentiert. Anfangs sind noch keine Punkte zugeteilt. Im Zuge der Optimierung werden Flächen eventuell wieder aufgelöst. Deren verwaiste Punkte müssen erneut segmentiert werden um Löcher im Dach zu vermeiden.

Aus den segmentierten Region-Verbünden entstehen Instanzen der Plane-Datenstruktur.

Implementierungsvermerk: Dachebene-Datenstruktur

Wir betrachten die Hintergründe dieser Datenstruktur. Region ist aus Effizienzgründen recht einfach gehalten, weil in der Klassifikationsphase sehr viele (geschätzt etwa 1 % der Anzahl ursprünglicher Punkte) davon erzeugt und wieder verworfen werden. Der genaue Inhalt dieser Datenstruktur ist in Abschnitt A.3.3) nachzulesen.

Da Gebäude normalerweise aus weniger als 30 Dachflächen bestehen, ist die Effizienz dort nicht so relevant. Vielmehr ist es wichtig Erkenntnisse über die enthaltenen Punkte zu gewinnen. Dazu wird die Region in eine etwas aufwändigere Plane-Datenstruktur umgewandelt, welche folgende Elemente enthält:

- Punktwolke aller enthaltenen Punkte;
- Konvexe Hülle (2D);
- Ausgleichsebene in Hesse-Normalform;
- Achsenausgerichteter umschließender Quader;
- Minimaler umschließender Parallelepiped;
- Diverse Eigenschaften: Abmessungen, Dichte.

Die konvexe Hülle berechnet man in zwei Dimensionen; die Z-Komponenten der Punkte entfallen. Dadurch verringert sich die Größe der Hülle, insbesondere wenn eine ‘Fläche’ anfangs nicht nur aus planaren Punkten besteht. Diese Entscheidung wird Auswirkungen auf das Verschmelzen der Ebenen haben. Die Problematik wird in Abschnitt 4.2.6 angesprochen.

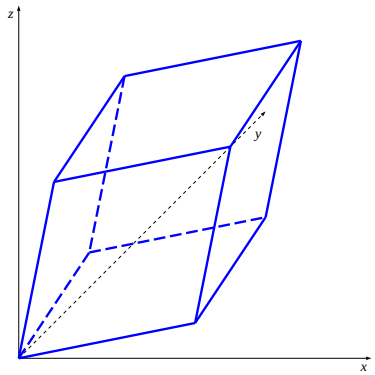


Abbildung 4.3: Prinzipskizze eines Parallelepipeds. (wird zur Begrenzung der Dachflächen verwendet)

Ein Parallelepiped ist ein schiefgezogener Quader (Abbildung 4.3). Er besteht aus sechs Parallelogrammen, die kongruent und parallel zur gegenüberliegenden Facette sind. Rowland und Weisstein Dieses Gebilde ist nützlich, weil es effizient

aber relativ genau eine Bestimmung erlaubt, ob Punkte im umschlossenen Gebiet enthalten sind. Ferner lassen sich viele der erwarteten (3D)-Flächen gut dadurch approximieren. Man berechnet es mithilfe der konvexen Hülle.

4.2.3 Verfeinerung der Ebenen

Ein weiterer Verarbeitungsschritt ist die Entfernung von Ausreißer. Hierzu werden Punkte eliminiert, die zu weit (siehe unten) von der Ausgleichsebene liegen und die Ebenen danach neu berechnet. Dies ist so lange zu wiederholen bis die Ebene ausreichend genau alle verbleibenden Punkte approximiert.

Welche Punkte entfernen?

Von zentraler Bedeutung ist, wieviel Diskrepanz toleriert wird, bevor ein Punkt als Ausreißer gilt. Man könnte einfach beispielsweise alles jenseits von 10 cm Entfernung abschneiden. Der Erfolg dessen wäre jedoch an die momentanen Gegebenheiten der Daten (Beschaffung des Gebäudes, Messgenauigkeit) gebunden. Daher empfiehlt sich eine datenunabhängige Methode.

Es wird dazu vorgeschlagen, solange das obere Quartil[6] zu entfernen, bis die maximale Entfernung den Laser-Höhenfehler unterschreitet (zirka 7 cm).

Dies erscheint als guter Kompromiss zwischen Laufzeit, Genauigkeit und Plausibilität.[7] Im Mittel ergeben sich dadurch 1,83 Iterationsschritte, womit dann 44 % der ursprünglichen Punkte in den Flächen verbleiben. Man beachte, dass dies nicht unbedingt einer Median-Filterung entspricht, da zwischen den Schritten die Ebene neu berechnet wird. Somit ist mit dieser Methode eine genauere Bestimmung der Ausgleichsebene möglich.

Alternative Methoden

Der Vergleich mit dem RANSAC (Random Sample Consensus)-Verfahren Fischler und Bolles (1981) ist interessant. Dieser wählt eine zufällige Untermenge der Punkte und berechnet daraus eine Ausgleichsebene. Falls ‘genügend’ weitere Punkte dazu passen, wird die Ebene als ausreichend gute Approximation aller Punkte akzeptiert. Ansonsten sind diese Schritte zu wiederholen, bis die Iterationshöchstgrenze erreicht ist.

Wie bei randomisierten Algorithmen üblich, liegt der Vorteil in der Laufzeit, da nicht alle Punkte betrachtet werden müssen. Allerdings ist die Wahl des Schwellwerts und die Anzahl der Iterationsschritte kritisch. Dafür muss bekannt sein, mit welcher Wahrscheinlichkeit zufällige Punkte zu einer Ebene passen.

Bei Laserdaten wird dies für problematisch gehalten, denn andere Dachmaterialien und Messbedingungen bewirken weitaus größere Differenzen im Messfehler. Laufzeit ist auch nicht das vorrangige Designziel. Es wird deshalb die etwas langsamere, aber robuste Methode bevorzugt.

[6] Die höchsten 25 % der Werte einer Verteilung.

[7] Es wäre nicht sinnvoll, eine zu strenge maximale Abweichung von beispielsweise 1 mm zu verlangen. Dies würde zu viele Punkte entfernen, weil der Messfehler weitaus höher ist.

4.2.4 Aufweitung der Flächen

Im nächsten Schritt werden die stark dezimierten Flächen wieder aufgeweitet. Man weist einen Punkt der Fläche zu, deren Ebene dem Punkt am nächsten liegt. Da die Ebenen sehr genau bestimmt wurden, werden die Flächen dementsprechend korrekt aus ihren Punkten zusammengesetzt. Allerdings sind dabei die folgenden Probleme zu beheben:

- Anschneiden anderer Flächen: Eine Ebene kann eine andere Fläche 'anschneiden', weil manche Punkte mitunter leicht näher daran liegen. Dies ist nicht weiter störend, weil es die Ebenenaufstellung nicht betrifft; es ist aber unschön und leicht vermeidbar. Falls sich die Normalenvektoren des Punktes und der Fläche unterscheiden, wird die Fläche disqualifiziert – ihr kann der Punkt nicht zugewiesen werden. Dies kann allerdings dazu führen, dass den Punkten die am zweitbesten passende Fläche zugewiesen wird. Um dies zu verhindern, wird zusätzlich eine Maximalentfernung angegeben, ab der die Punkte nicht mehr zu einer Fläche gerechnet werden dürfen.
- Isolierte Punkte: Analog können vereinzelte Punkte eventuell der falschen Fläche zugeteilt werden. Diese würden 'Inseln' in der anderen Ebene bilden.

Eine Abhilfe wäre, nur dann Punkte einer Fläche zuzuweisen, wenn sie innerhalb deren Grenzen liegen. Dies ist allerdings problematisch. Durch das Verfeinern wurden die Flächen stark dezimiert. Das Aufweiten soll die ursprüngliche Größe wiederherstellen. Dies würde aber durch eine solche Begrenzung verhindert werden.

Es liegt nahe, erst nach dem zweiten Aufweiten Grenzen einzuführen. Dies hilft aber auch nicht, weil dann die eigentlich korrekten Grenzen schon gesprengt wären. Punkte können nämlich bereits außerhalb der Fläche zugewiesen worden sein.

Schließlich erscheint es nicht praktikabel, andere Kriterien zur Flächenzugehörigkeit zu benutzen. In diesem Schritt ist noch nicht bekannt, wie die Ebenen begrenzt sein sollen. Da dieses Problem also nicht vermieden werden kann, muss man die Folgen einer falschen Zuteilung betrachten.

Falls Flächen sich gegenseitig Punkte abnehmen, werden sie tendenziell eher zusammengeführt werden. Dieser durchaus positive Nebeneffekt wird im nächsten Abschnitt ersichtlich. Allerdings werden auch die Flächengrenzen aufgeweitet. Diese dienen dazu, unechte Schnittlinien[8] zu erkennen. Sind die Grenzen verfälscht, werden ungültige Schnittlinien unter Umständen akzeptiert. Das muss verhindert werden.

Es ist möglich, falsch zugewiesene Punkte nachträglich korrekt einzuordnen. Dazu definiert man R als 'richtige' Fläche, zu der ein Punkt gehören soll, und F als 'falsche', die den Punkt bekommen hat. Auf den Punkten von F wird eine starke Erosion durchgeführt. Da die Fläche als Punktwolke gespeichert ist, muss dazu für jeden Punkt die Nachbarschaft bestimmt werden. Punkte werden F entnommen und zu R gerechnet, wenn sie nicht fast völlig von F-Punkten umgeben sind. Diese hohe Hürde soll bewirken, dass alle vereinzelten Punkte zuverlässig entfernt werden. Allerdings wird dadurch

[8] Schnittlinien zweier Ebenen (nicht Flächen!), die sich nur zufällig – womöglich noch außerhalb der Dachfläche – schneiden.

natürlich auch der Flächenrand erodiert. Deswegen (und weil nicht unmittelbar klar ist, welche andere Fläche R die tatsächlich korrekte ist) muss man überlegen, welche Flächen so bearbeitet werden sollen.

Wann können Punkte falsch zugeteilt werden? Per Definition liegt ein falsch zugeteilter Punkt näher bei F als R. Dies passiert unter anderem wenn die Höhe des Punktes falsch gemessen wurde. Damit man von einer falschen Zuteilung sprechen kann, muss es in der näheren Umgebung Punkte geben, die zu R gehören. Es gibt also Punkte unterschiedlicher Flächen, die nicht weit voneinander entfernt liegen aber auch nicht als Unstetigkeitsstelle gekennzeichnet sind. Das ist nur möglich, wenn F und R fast parallel sind. Genau diese Flächenpaare werden wie oben beschrieben separiert.

Es stellt sich die Frage: Welches davon ist nun R beziehungsweise F?

Wegen der Greedy-Eigenschaft[9] des Aufweitens ist das falsche Zuordnen eher einseitig. Eine Fläche bekommt mehr Punkte von der anderen Fläche als ihr selber abgenommen werden. F kann also dadurch erkannt werden, dass ihre Punktdichte niedriger ist. Die falsch zugewiesenen Punkte sind nämlich nur wenige, die aber die Flächenausbreitung überproportional stark vergrößert haben.

Damit ist das Problem gelöst; alle Punkte werden der korrekten Fläche zugeordnet.

Implementierungsvermerk: Erosion

Im Abschnitt 4.2.4 wurde erwähnt, dass bei der Erosion die Nachbarschaft jedes beteiligten Punktes bestimmt werden muss. Es ist ersichtlich, dass dieser Vorgang trotz effizienter Realisierung der Nachbarschaftssuche rechenaufwändig ist. Dies stellt allerdings kein Problem dar, weil der gesamte Korrekturschritt nur selten nötig ist; siehe dazu die obige Voraussetzung.

Sollte diese Operation jedoch häufiger benötigt werden, kann man in Betracht ziehen, die Erosion an einem Bild durchzuführen. Dazu werden die Punkte in ein Raster transformiert. Die Implementierung der Erosion ist dann wegen der Möglichkeit des direkten Zugriffs auf die Nachbarn trivial. Für jede Zelle muss allerdings vermerkt werden, welche Punkte darin liegen. Somit kann das Ergebnis der Erosion von den Zellen auf die beteiligten Punkte übertragen werden.

4.2.5 Markierung der Grenzpunkte

Nach dem obigen Schritt verbleiben einige Punkte, die nicht einer Ebene zugewiesen wurden. Im einfachen Fall liegt dies an vereinzelten Höhenmessfehler und die Punkte bleiben unberücksichtigt. Eine weitere Ursache liegt jedoch in der Forderung, dass der Normalenvektor eines zuzuteilenden Punktes der Orientierung der Ebene entsprechen muss. Damit wird das ‘Anschneiden’ einer anderen Fläche verhindert (siehe vorigen Abschnitt). Allerdings sind die Normalenvektoren am Rand einer Fläche gestört, sodass solche Punkte nicht zugewiesen werden. Es bietet sich jedoch an, diesen Umstand in einen Vorteil zu verwandeln.

[9] Ein Greedy-Algorithmus verfolgt die Strategie, in jedem Schritt die momentan am besten erscheinende Entscheidung zu treffen. Im Gegensatz zur sogenannten Dynamischen Programmierung werden Entscheidungen jedoch niemals rückgängig gemacht. DeVore und Temlyakov (1996) Bellman (2003)

Später wird die Menge der Punkte benötigt, durch die die Schnittlinien verlaufen. Aus der Erstreckung dieser Punkte sollen die tatsächlichen Endpunkte der unendlich langen Linien bestimmt werden. Diese Punkte sind jedoch genau die, die am Rand der Dachflächen liegen. Damit stellen die bislang nicht zugewiesenen Punkte eine gute Annäherung an die Menge der gewünschten Grenzpunkte dar.

Es sollten jedoch noch Punkte entfernt werden, die aufgrund eines Messfehlers oder sonstiger Störungen nicht zugewiesen wurden. Ansonsten können sie dazu führen, dass Linien fälschlicherweise verlängert werden und in eine Dachfläche hineinragen. Wie können diese unerwünschten Punkte erkannt werden? Grenzpunkte liegen typischerweise dicht beisammen; die fehlerhaften Punkte im Inneren einer Fläche sind hingegen eher lückenhaft. Zur Unterscheidung dieser Punktmengen kommt ein einfaches Filter zum Einsatz. Für jeden nicht zugewiesenen Punkt wird eine relativ kleine Nachbarschaft betrachtet. Sind bereits viele der Nachbarpunkte als Grenzpunkt markiert, gilt der neue Punkt auch als solcher. Ansonsten lautet die Bedingung, dass es zwei Ebenen geben muss, zu denen viele der Nachbarschaftspunkte gehören.

Diese Bedingungen erscheinen plausibel und können relativ erfolgreich die gestörten Punkte entfernen[10]. Es wird also verhindert, dass Geradenstücke ungewollt in andere Flächen verlängert werden.

4.2.6 Verschmelzung ähnlicher Flächen

Nun werden ähnliche Flächen zusammengeführt, die möglichst das gleiche Dachstück darstellen. Dieser Schritt ist nicht unabdingbar, kann aber gelegentlich redundante Ebenen entfernen. Da nachher der Schnitt jeder Fläche mit den anderen bestimmt werden soll, vermeidet diese Zusammenführung auch redundante Schnittlinien. Außerdem lässt sich dann leichter visuell überprüfen, ob alle Flächen korrekt erzeugt wurden.

Durch das Verschmelzen ändert sich die Punktmenge. Die Ausgleichsebene wird also anschließend neu berechnet.

Kriterien für die Ähnlichkeit

Flächen sollen zusammengeführt werden, wenn sie koplanar sind, also in der gleichen Ebene liegen. Notwendig dazu ist, dass die beiden Normalenvektoren übereinstimmen. Dies wird als schnelles Ausschlusskriterium verwendet. Hinreichendes Kriterium wäre, dass die maximale Distanz zwischen den Flächen klein ist. Das lässt sich über die Entfernung der Eckpunkte zur anderen Ebene feststellen.

Man beachte, dass diese Relation nicht symmetrisch ist. Es macht einen signifikanten Unterschied, ob die Eckpunkte von E1 mit E2 verglichen werden oder umgekehrt. Dies liegt daran, dass die Punkte nicht perfekt die Fläche repräsentieren, weil sie nicht exakt vorliegen. Um sicher zu gehen, werden beide Permutationen aller Flächenpaare auf Ähnlichkeit geprüft. Da es nur wenige Flächen pro Gebäude gibt, ist der Mehraufwand vernachlässigbar.

[10] Im Detail werden die Grenzpunkte mit einer Kennung versehen, während die 'entfernten' Punkte lediglich als nicht zugewiesen markiert bleiben.

Als repräsentive Stichprobe der Berandung wird die konvexe Hülle herangenommen. Wie vorhin erwähnt, liegt diese nur in 2D vor. Bei der Berechnung werden die Z-Komponenten der Punkte ignoriert; diese lassen sich allerdings wiederherstellen. Die Hülle ist durch Indizes in der Menge der 2D-Punkte gegeben. Diese gelten auch für die ursprünglichen Punkte; somit hat man wie gewünscht eine Annäherung an die Menge der (3D) Eckpunkte.

Allerdings gibt es hierbei ein prinzipielles Problem. Falls viele der originalen Punkte, die als Hüllenpunkte ausgewählt werden, auf der gleichen Seite der Ausgleichsebene liegen, wären sie nicht mehr für die gesamte Fläche repräsentativ. Es würde ein systematischer Fehler in den Abständen zwischen Hüllenpunkten und anderer Ebene entstehen.

Dem wird auf zweierlei Arten entgegengewirkt: Erstens betrachtet man bei den Entfernungen den Median statt einer maximum-Schwelle. Dadurch werden einige nicht repräsentive Randpunkte toleriert. Zweitens ersetzt man bei jedem Punkt die ursprüngliche z-Komponente durch die z-Koordinate der Ebene an der xy-Position des Punktes. Damit ist ein akkurates Maß zur Entfernung gegeben und das obige Problem gelöst.

Disjunkte Flächen

Es stellt sich die Frage, ob koplanare aber disjunkte Flächen auch zusammengeführt werden sollen. Ein Beispiel wären die Endstücke zweier Flügel eines symmetrischen Gebäudes. Dies könnte durch die Abfrage, ob alle Hüllenpunkte der beiden Flächen paarweise weit entfernt sind, erkannt werden.

Der Vorteil des Verschmelzens wäre, redundante Schnittgeraden zu vermeiden. Nachteilig ist, dass man die Möglichkeit aufgibt, die eigentliche Dachfläche begrenzen zu können. Dies wäre aber sowieso wegen dem Arbeitsschritt "Aufweiten" unmöglich. Solche Flächen werden sich nämlich dann gegenseitig Punkte wegnehmen. Damit sind die Grenzen zerstört. Interessant ist allerdings, dass die Ausreißer dann auch das Zusammenführen begünstigen. Sie liegen nahe bei der anderen Ebene und führen dazu, dass die Ebenen eher gleich bestimmt sind.

Da der Nachteil sowieso in Kauf genommen werden muss, erlaubt man das Verschmelzen disjunkter Flächen. Dies hilft dann auch bei der visuellen Inspektion der Ergebnisse. Man beachte, dass Flächen dadurch mehrere nicht zusammenhängende Randstrecken haben können. Diese werden bei der geometrischen Analyse wieder aufgetrennt.

4.2.7 Beseitigung entarteter Flächen

Im letzten Schritt werden Flächen entfernt, die nicht mehr gültig erscheinen. Dies kann dadurch passieren, dass beim Aufweiten fast alle Punkte einer anderen Fläche zugewiesen wurden. Der Sinn des Auflösens ist dann, die eventuell verbliebenen Punkte auch der anderen Fläche zu überlassen.

Im Detail gibt es mehrere Indizien dafür, dass eine Fläche nicht korrekt erfasst wurde: Zu geringer Flächeninhalt, minimale Ausdehnung in einer Richtung des Parallelepipeds oder niedrige Punktdichte.

Bei der Dichte gibt es eine Besonderheit. Wegen der Verschmelzung disjunkter Dachflächen ist das naive Flächenmaß (auf Basis des umschließenden Quaders) inakkurat. Die Fläche kann fast beliebig überschätzt werden. Somit ist es nicht hinreichend, die Toleranz für niedrigere Dichten zu erhöhen. Alternativ kann man die Fläche durch Rasterung der Punkte und unter Berücksichtigung der 'Löcher' genauer ermitteln. Dies hätte jedoch hohe Laufzeitkosten.

Stattdessen berücksichtigt man die Dichte wie folgt: Das Verhältnis der Anzahl Punkte zur LIDAR-Punktdichte (gegeben über den Rasterabstand) muss die minimale Fläche übertreffen. Somit können auch Flächen erkannt werden, die wenige Punkte bei großer Ausdehnung aufweisen.

Wird eine Fläche aufgelöst, markiert man ihre Punkte als nicht zugeteilt. Im nächsten Iterationsschritt werden solche Punkte wieder mittels Segmentierung zu Flächen zusammengefasst. Notabene: Falls nicht genügend Punkte vorhanden sind, um die minimale Regionsgröße des Segmentierers zu erreichen, bleiben diese wenigen Punkte fernab einer Fläche. Dies ist für das Ergebnis nicht von Bedeutung, da nur die Ebene weiterverwendet wird, nicht die einzelnen Punkte.

4.2.8 Abbruchkriterien

Die Gesamtheit aller Schritte ergibt einen Optimierungsdurchlauf. Es soll weiter iteriert werden bis sich keine weitere Verbesserung einstellt. Dies wird anhand folgender Kennzahlen festgestellt: Die Anzahl nicht zugewiesener Punkte und die aufsummierte Entfernung aller Flächenpunkte zur zugehörigen Ebene.

Man beobachtet, dass der zweite Durchlauf beim 'fom'-Datensatz noch eine Verbesserung von 6 % in der Punktzahl bringt; allerdings kommt dies auf Kosten einer 10 % höheren Entfernungssumme. Dies ist vermutlich darauf zurückzuführen, dass manche 'fraglichen' Punkte noch einer Fläche zugeordnet werden. Weitere Iterationsschritte bewirken Änderungen von weniger als 1 % und können das Ergebnis sogar verschlechtern.

Diese Beobachtungen sind nun nachzuvollziehen. Beim zweiten Segmentieren werden einige Punkte neu zugewiesen, die im ersten Durchlauf verworfen wurden. Dies erklärt die unterschiedlichen Werte der Metriken. In weiteren Durchläufen sind keine großen Unterschiede zu erwarten. Die Optimierung ist deterministisch und relevante Änderungen werden nur in der Verfeinern-Phase erwartet. Diese wird jedoch so lange wie nötig wiederholt, sodass sich zwischen den Optimierungsdurchläufen nicht mehr viel ändert. Das deckt sich mit der Beobachtung.

Es werden also immer genau zwei Iterationsschritte durchgeführt und auf ein differenzierteres Abbruchkriterium verzichtet.

4.2.9 Ergebnis

Die resultierenden optimierten Dachflächen sind in Abbildung 4.4 gezeigt. Man beachte die folgenden Verbesserungen und Besonderheiten:

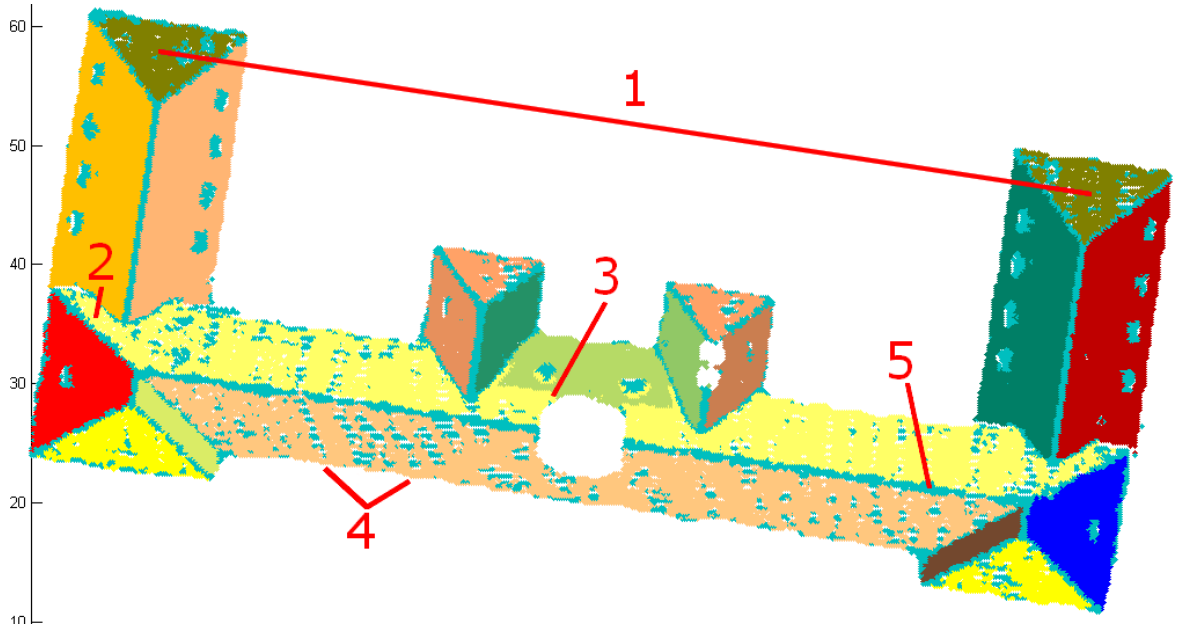


Abbildung 4.4: Optimierte Dachflächen. (Punkte sind entsprechend ihrer Ebenenzugehörigkeit eingefärbt. Nummerierte Stellen werden im Text erläutert)

1. Disjunkte aber koplanare Flächen werden zusammengelegt.
2. Die kleine Ecke wird richtigerweise der gelben Fläche zugerechnet.
3. Diese Fläche ist fast parallel zur Nachbarfläche und wurde korrekt getrennt.
4. Streifen nicht zugewiesener Punkte sind sichtbar, falls sie etwas weiter entfernt von der Ebene liegen.
5. Die Dachfirste werden als solches gekennzeichnet.

4.3 Bestimmung der Wandebenen

Nachdem die Dachflächen vorliegen, werden nun die Wände des Gebäudes bestimmt.

Zunächst ist noch zu klären, was man unter ‘Wand’ versteht. Per Definition wird diese vom Dach zumindest überdeckt, wenn nicht sogar überragt. Somit sind die eigentlichen Wände in ALS-Daten nicht sichtbar. Es verbleibt im Allgemeinen nichts anderes als anzunehmen, dass die Wände am Rand des Daches liegen.

Zur Bestimmung der Wände werden Hypothesen für deren Verlauf aufgestellt und die wahrscheinlichste ausgewählt. Nach Erzeugung der dazugehörigen Wand wird der Vorgang so lange wiederholt, bis keine der Hypothesen plausibel erscheint. Die obigen Arbeitsschritte werden nun detailliert beschrieben.

4.3.1 Bestimmung der Wandhypothesen

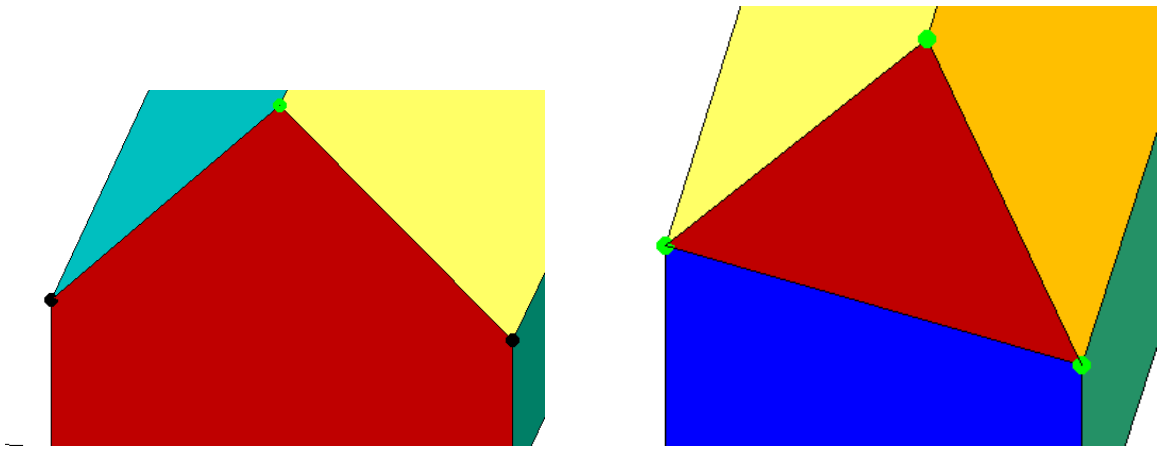
Es gibt mehrere Möglichkeiten, um Hinweise auf die Lage der Wände zu erhalten:

1. Verbindung der Eckpunkte des Daches;
2. Verwendung der konvexen Hüllen der Dachflächen;
3. Extraktion von Linien aus Randpunkten.

Diese werden im Folgenden vorgestellt und deren Vor- und Nachteile diskutiert:

Verbindung der Eckpunkte des Daches

Viele Dächer sind so beschaffen, dass einige Schnittpunkte der Flächen am Dachrand liegen. Diese können als Endpunkte der Wandlinie gesehen werden. Alle Randpunkte, die zu dieser Linie passen, gehen in die Berechnung einer Ausgleichsgerade mit ein. Allerdings ist diese Methode nicht universell einsetzbar. Es gibt Dächer, die keine geeigneten Eckpunkte aufweisen. Als Beispiele dienen die beiden Dächer in Abbildung 4.5.



(a) Giebeldach (b) Walmdach

Abbildung 4.5: Dach-Eckpunkte (grün) – ergeben die Verbindungen Wände?

Links ist ein Giebeldach gezeigt, bei dem die schwarzen Punkte erst nach Erhalt der Wände zur Verfügung stehen. Es kann nicht vom einzelnen Dacheckpunkt (grün) auf die Wände geschlossen werden. Beim rechten Walmdach entstehen hingegen alle Eckpunkte aus den Dachflächen und weisen auf eine Wand hin.

Ein anderer Nachteil dieser Methode ist, dass einige Schritte in der Dachanalyse wiederholt werden müssten, was nicht sonderlich elegant wäre. Die Eckpunkte müssten zur Erzeugung der Wände vorliegen, sind aber erst danach verfügbar.

Verwendung der konvexen Hüllen der Dachflächen

Wie anfangs erwähnt, sollen Wände als Rand des Daches verstanden werden. Am naheliegensten ist, hierfür die Randstücke der konvexen Hüllen der Dachflächen zu verwenden. Damit wäre ein Hinweis für die Hough-Transformation gegeben, dass dort eine Linie zu finden ist.

Die Qualität der Hülle ist allerdings nicht gesichert. Je nach Krümmung der Linie und Entfernungstoleranz können viele kurze Hüllenstrecken entstehen. Außerdem ist die konvexe Hülle per Definition eine Approximation der Flächengrenzen, sodass die wahren Wandverläufe dadurch nicht zu erhalten sind.

Extraktion von Linien aus Randpunkten

Die erste Möglichkeit besteht darin, Linien direkt aus der Punktwolke zu extrahieren. Dazu ist die Hough-Transformation das gängige Verfahren. Eine Wand wird dort vermutet, wo viele Randpunkte auf einer Linie liegen. Durch Transformation der Punkte in den Hough-Parameterraum werden solche Anhäufungen ersichtlich. Es werden so sukzessive Linien extrahiert und die Punkte aus der Menge der verbleibenden Randpunkte entfernt. Details zur Hough-Transformation und den Maßnahmen, die eine hohe Genauigkeit sicherstellen, werden in Abschnitt A.4 gegeben.

Die Hough-Transformation hat den generellen Nachteil, dass nicht klar ist, wann alle sinnvollen Linien extrahiert wurden. Eventuell werden unechte Linien aufgenommen oder gültige verworfen. Aufgrund der hochwertigen Realisierung der Hough-Transformation und der dazugehörigen Massnahmen zur Bereinigung der Akkumulatormatrix kann aber sichergestellt werden, dass nur sinnvolle Linien extrahiert werden.

Außerdem ist nachteilig, dass die Hough-Transformation global[11] arbeitet. Es kann vorkommen, dass Punkte verbunden werden, die eigentlich (für einen Menschen ersichtlich) nicht auf einer zusammenhängenden Linie liegen. Dies wird in einem darauffolgenden Schritt verhindert: Falls die Schnittlinie einer Wand und Dachfläche nicht in den Punktdaten erkennbar ist, wird sie verworfen. Damit ist die globale Funktionsweise auch nicht von Nachteil.

Schließlich sind erkannte Linien nicht exakt. Durch die Quantisierung der Hough-Akkumulatoren kann die Richtung nicht genau ermittelt werden. Die erreichte Genauigkeit (siehe Abschnitt A.4.5) ist aber mehr als zufriedenstellend. Es kann sogar ein weiterer Optimierungsschritt folgen. Alle Punkte, die nahe der extrahierten Linie liegen, können in die Berechnung einer Ausgleichsgerade einfließen. Damit ist eine fast beliebige hohe Genauigkeit erreichbar.

Als positive Eigenschaft der Hough-Transformation ist noch zu erwähnen, dass sie eventuell Linien aufdeckt, die mit den anderen Massnahmen nicht gefunden würden. Es werden keinerlei Annahmen über die Form des Daches getroffen. Wie ersichtlich ist, können alle Probleme der Hough-Transformation behoben werden. Die anderen möglichen Verfahren weisen hingegen mehr Nachteile als Vorteile auf. Man erhält also die Wandhypothesen durch Hough-Transformation der Randpunkte.

4.3.2 Auswahl der wahrscheinlichsten Wandhypothese

Obwohl die Hough-Transformation relativ zuverlässig ist, können trotzdem ähnliche oder unechte Wandhypothesen entstehen. Zur Vermeidung dessen wird die 'wahrscheinlichste' Wand zuerst verarbeitet und darauffolgende Duplikate ausgeschlossen. Dies

[11] Global bedeutet, dass alle Dachpunkte gleichzeitig bearbeitet werden. Über die lokalen Linien in Teilbereichen des Ganzen ist kein a-priori Wissen vorhanden.

geschieht über die Menge der verbleibenden Randpunkte. Je mehr Punkte in der Nähe einer Wandhypothese liegen, desto eher ist ihre Richtigkeit bezeugt. Sobald keiner der restlichen Hypothesen ausreichend viele Punkte entsprechen, wird das Verfahren abgebrochen.

4.3.3 Erzeugung der Ebenen

Entsprechend der obigen Reihenfolge werden Wandhypothesen akzeptiert und liegen als 2-dimensionale Linien vor. Es ist eleganter, sie – wie auch die Dachflächen – als Ebene zu behandeln. Dann kann man zur Bestimmung der Dachtraufen einfach die Schnittlinien der Ebenen heranziehen. Neben der einfachen Implementierung bietet sich ein weiterer Vorteil. Dadurch, dass alle Punkte der beiden Ebenen mit in die Berechnung eingehen, sind die Ergebnisse zuverlässiger, als wenn nur die Wandpunkte berücksichtigt würden.

Da für die Wand auch die Menge der benachbarten Punkte bekannt ist, kann die Plane-Darstellung bereits erzeugt werden. Allerdings wird die Orientierung des Normalenvektors vermutlich falsch sein. Es ist nicht gewährleistet, dass die gemessenen Punkte eine ausreichende vertikale Ausdehnung aufweisen, um die Ausgleichsebene korrekt zu bestimmen. Deshalb wird der Normalenvektor der Ebene nachträglich korrigiert. Da er parallel zum Boden und orthogonal zur Wandgerade sein soll, kann er aus dem Kreuzprodukt der Z-Achse und dem Richtungsvektor der Geraden erhalten werden.

Somit sind die Wandebenen erzeugt und werden wie Dachebenen behandelt. Sie können jedoch noch über ihre Kennzahl als Wand ausgemacht werden.

4.4 Ermittlung der Eckpunkte des Gebäudes

Im vorletzten Schritt des Verfahrens sind die Eckpunkte des Gebäudes herauszufinden. Diese definieren die Gebäudepolygone und sind also bestimmend für die Genauigkeit des Endergebnisses. Entsprechend ist dieser Vorgang etwas komplexer als erwartet. Zunächst werden die Trennlinien der Wände und Dachflächen aufgestellt und anhand der ursprünglichen LIDAR-Punkte in Geradenstücke aufgeteilt. Die so gewonnenen Endpunkte werden gruppiert, um dann die Eckpunkte zu erhalten. Es folgen Details zu den einzelnen Arbeitsschritten.

4.4.1 Erzeugung der Trennlinien

Gesucht sind die Trennlinien aller Flächen. Da deren Grenzen aber nicht exakt vorliegen, muss über die Schnittlinien der Ebenen argumentiert werden. Diese kann man geometrisch bestimmen, wobei jedoch einige Linien entstehen, die keine echte Trennlinie darstellen. Es werden also Punkte gesucht, die in der Nähe der Linie liegen. Falls keine gefunden werden, gilt die Linie als ungültig und wird entfernt. Schließlich werden

ähnliche Linien verschmolzen und alle Richtungsvektoren eine einheitliche Orientierung gegeben. Diese einzelnen Schritte werden nachfolgend erläutert.

Konstruktion der Ebenen-Schnittlinien

Man betrachtet alle Kombinationen zweier Ebenen. Über den Winkel zwischen den Normalenvektoren lässt sich feststellen, ob die Ebenen parallel oder koplanar sind. Falls nicht, existiert eine Schnittlinie, die sich mittels Kreuzprodukt der Normalenvektoren aufstellen lässt.

Sammeln nahegelegener Punkte

Für jede Linie bestimmt man die Menge der nahegelegenen Punkte und fasst sie in einer Punktwolke zusammen. Das Vorhandensein dieser Punkte deutet darauf hin, dass die Linie tatsächlich in den Daten vorhanden ist und nicht zufällig aus zwei unverbundenen Flächen entstanden ist. Außerdem werden die Punkte benötigt um die Linie in Geradenstücke aufteilen zu können. Im Detail sammelt man alle Punkte aus einer Eingabemenge, die innerhalb einer ϵ-Umgebung rund um die Linie liegen. Hierhinter verbirgt sich überraschend viel Komplexität:

Zur Bestimmung der Eingabemenge betrachten wir die drei Fälle bezüglich der Ebenen, aus denen die Schnittlinie hervorging:

- Wand mit Dach oder umgekehrt: Es sind nur Randpunkte zulässig. Da Wände vertikal sind und die Schnittlinie definitionsgemäß gänzlich in den beiden Ebenen liegt, verhindert diese Einschränkung, dass die Linie ungewollt Dachpunkte aufnimmt.
- Wand, Wand: Solche Schnittlinien sind nicht von Interesse, da die Wandpolygone unabhängig davon erzeugt werden können. Sie werden deshalb verworfen.
- Dach, Dach: Zunächst sind nur Grenzpunkte erlaubt; diese sind in Abschnitt 4.2.5 definiert. Damit verhindert man, dass die Verlängerung einer legitimen Linie fälschlicherweise Punkte einer unbeteiligten Ebene sammelt.

  Als Ausnahme gelten zwei Ebenen, die im spitzen Winkel zueinander liegen. Die Punkte entlang der Grenze werden nicht unbedingt als Grenzpunkte markiert, weil ihre Normalenvektoren laut Voraussetzung ähnlich sind. In diesem Fall erlaubt man auch das Sammeln von Punkten, die zu einer der beiden beteiligten Ebenen gehören.

Beim Festlegen der maximalen zulässigen Entfernung von der Linie gibt es auch eine Besonderheit. Punkte sollten innerhalb oder in der Nähe der umschließenden Quader der Flächen liegen. Eigentlich dürfen sie auch eine gewisse Entfernung zur Linie nicht überschreiten. Reale ALS-Daten weisen jedoch beträchtliches Aliasing quer zur Flugrichtung auf. In Abbildung 4.6 wird eine horizontale Sicht auf die südliche Dachtraufe des FOM-Daches gezeigt.

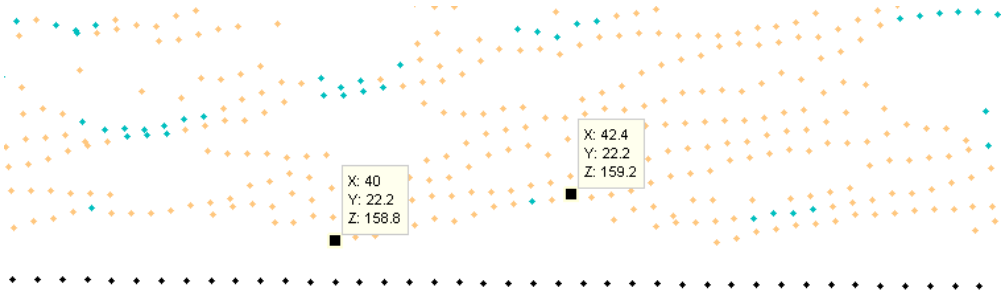


Abbildung 4.6: Horizontale Sicht auf eine ‘fom’-Dachtraufe mit Sägezahnmuster durch Aliasing quer zur Flugrichtung.

Man erkennt vertikale Unterschiede von 40 cm, die in der beschränkten LIDAR-Auflösung entlang dieser Achse begründet liegen. Dieses Problem wird durch eine höhere Toleranz in Z-Richtung gelöst. Genauer gesagt trägt der Unterschied der Z-Komponente bis zu einem gewissen Schwellwert gar nicht zur Entfernung bei. Eine Alternative wäre, die Koordinaten beispielsweise mit einem Median-Filter zu glätten. Dies reduziert jedoch überall und nicht nur in Problemstellen die Genauigkeit, sodass darauf verzichtet wird.

Mit den obigen Schritten gelingt es, alle relevanten Punkte zu erfassen, die in der Nähe der Linie liegen.

Verwerfen unechter Linien

Eine Schnittlinie ist sicherlich dann keine Trennlinie, wenn sie außerhalb der Grenzen einer der beiden Flächen liegt. Dies ist insbesondere der Fall, wenn die Linie ganz außerhalb des achsenausgerichteten umschließenden Quaders liegt. Solche Linien können bereits mit wenig Aufwand erkannt und entfernt werden.

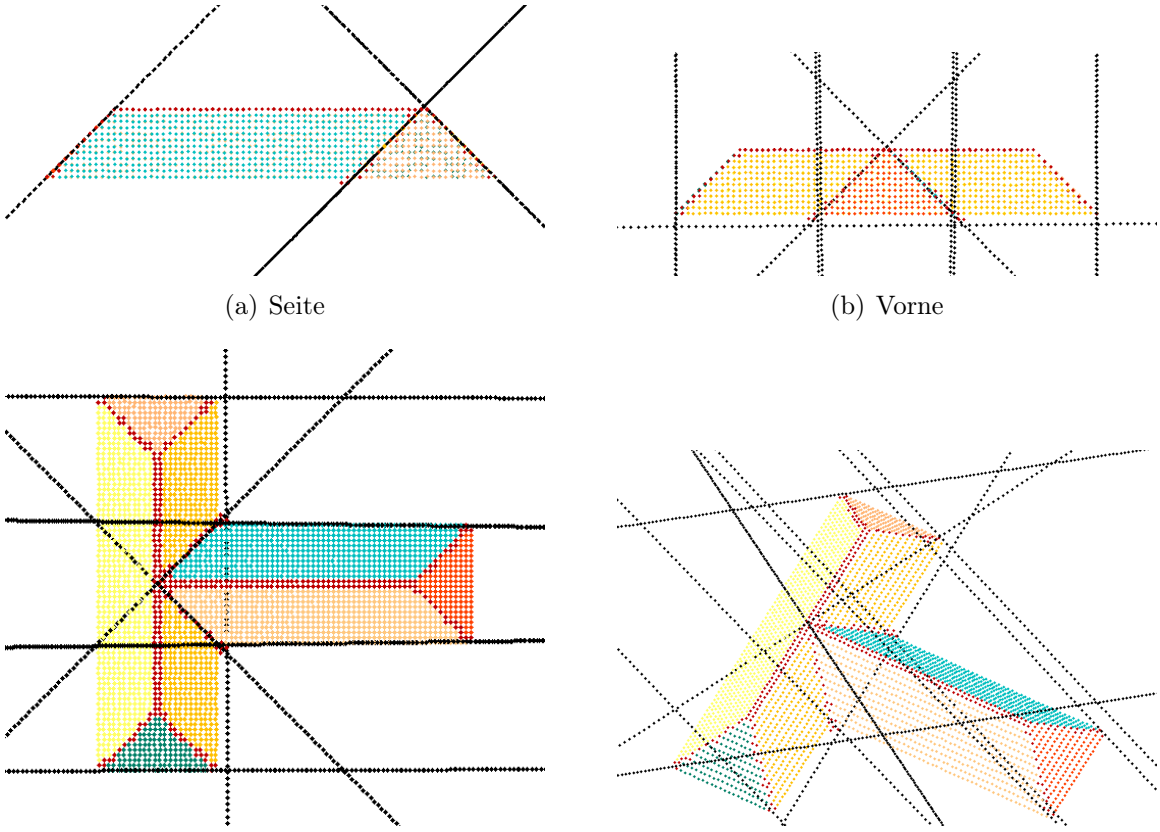
Damit sind aber noch längst nicht alle unechten Linien detektiert. Liegen keine oder nur wenige Datenpunkte in der Nähe einer Linie, so ist die Linie vermutlich nicht wirklich in den Daten vorhanden. Zusätzlich kann noch verlangt werden, dass von allen Punkten jeweils genügend zu *beiden* Ebenen gehören, aus denen die Schnittlinie entstand.

Zum Beispiel weist das T-Gebäude einige solche Linien auf. Verschiedene Ansichten davon sind in Abbildung 4.7 gezeigt.

Man beachte dabei, dass Grenzpunkte zu beiden Ebenen zählen. Dies ist wichtig, weil sie fast alle gesammelten Punkte ausmachen.

Verschmelzen ähnlicher Linien

Der Vollständigkeit halber werden Linien entfernt, die fast deckungsgleich sind. Dies sollte nicht vorkommen, weil ähnliche Ebenen bereits zusammengeführt wurden. Somit dürften die resultierenden Schnittlinien auch nicht identisch sein. Es ist aber denkbar, dass zwei nur fast ähnliche Ebenenpaare sehr wohl ‘gleiche’ Schnittlinien erzeugen. Der minimale Mehraufwand der Verschmelzung dieser Linien wird deshalb unternommen. Notwendiges Kriterium hierbei ist die Ähnlichkeit der Normalenvektoren. Hinreichend ist eine Deckungsgleichheit der Linien. Dies kann allerdings nicht über den Abstand



(a) Seite (b) Vorne

(c) Oben (d) Schräg

Abbildung 4.7: Vier Ansichten auf unechte Linien im ‘t’-Gebäude, die entfernt werden müssen.

der Linien festgestellt werden. Fast alle realen Linien würden sich wegen Rundungsfehlern im (nicht ganz) Unendlichen schneiden, sodass der berechnete Abstand null wäre. Stattdessen wird folgendes Maß verwendet. Man wählt zufällig einen Punkt aus der Menge der Linienpunkte. Der Punkt wird auf beide Geraden projiziert. Die Entfernung zwischen diesen Lotfußpunkten wird als Abstand angesehen.

Es wird so jede Permutation der Linienpaare untersucht. Sind welche ähnlich, wird die zweite Linie entfernt. Man könnte weitere Maßnahmen treffen, zum Beispiel die Geraden zusammenführen oder prüfen, welche Gerade die Andere dominiert (über den mittleren Approximationsfehler der Linienpunkte). Darauf wird allerdings verzichtet, weil wie anfangs erläutert nur selten ähnliche Linien auftreffen.

Vereinheitlichung der Richtungsvektoren

Nach den obigen Schritten verbleiben nur noch tatsächliche Trennlinien. Bei der Aufteilung der Linienstücke wird es nötig sein, dass ihre Richtungsvektoren einheitliche Orientierungen aufweisen. Die Richtung geht aus dem Kreuzprodukt der Normalenvektoren der sich schneidenden Ebenen hervor. Es genügt also die Orientierung der Normalenvektoren festzulegen; sie werden so definiert, dass sie nach außen zeigen.

Für Dachflächen ist dies bereits der Fall, weil ihre z-Komponenten zu positiven Werten gezwungen sind. Bei Wänden ist die Definition von 'außen' etwas schwieriger. Man betrachtet dazu alle Dachtrauflinien, die aus dem Schnitt einer Dachfläche mit einer Wand hervorgehen. Um unbeabsichtigte oder schiefe Schnitte zu vermeiden werden diejenigen bevorzugt, die nicht parallel zur Projektion des Normalenvektors der Dachfläche auf die xy-Ebene verlaufen. Falls der Schwerpunkt der beteiligten Dachfläche auf der falschen Seite der Linie liegt, wird der Normalenvektor der Wand umgedreht.

Da nun alle Normalenvektoren nach außen zeigen, brauchen die Richtungsvektoren der Linien nur nochmals aus den Normalenvektoren berechnet werden, um auch ihre Orientierung festzulegen.

4.4.2 Aufteilung der Geradenstücke

Die Trennlinien sind unendlich lang. Neben der Begrenzung auf das tatsächliche Gebäude bleibt noch das Problem, dass der Grundriss 'Einbuchtungen' enthalten kann. Somit würde eine Wandebene zwei disjunkte Teilstrecken der Wand beinhalten, was verhindert werden muss. Dazu spaltet man Trennlinien dort in Teilstrecken auf, wo keine realen Datenpunkte vorhanden sind. Um solche Lücken in den Linie zu erkennen, werden die dazugehörigen Punkte entlang der Linie sortiert. Diese Schritte werden im Folgenden besprochen.

Sortierung der Punkte

Anfangs sind die Punkte ungeordnet, weil sie als Punktwolke gespeichert sind. Nun werden sie anschaulich entsprechend ihrer Position auf der Linie sortiert.

Dazu projiziert man zunächst alle Punkte auf die Linie, um gleiche Ergebnisse auch bei mittelwertfrei verrauschten Koordinaten und schrägen Linien zu erhalten. Anschließend

wird eine Ebene orthogonal zur Linie aufgestellt und benutzt, um die (vorzeichenbehaftete) Entfernung der Punkte zu berechnen. Da die Ausdehnung der Punkte bezogen auf die darstellbaren Zahlen sehr klein ist, sind keine numerischen Probleme zu befürchten. Für die Sortierung ist also die Wahl des Ursprungs nicht von Bedeutung; es wird hierfür ein zufälliger Punkt auf der Linie verwendet.

Die so berechneten Entfernungen werden aufsteigend sortiert. Man beachte, dass hierdurch die Anordnung der Endpunkte der Geradenstücke festgelegt wird. Die Normalenvektoren aller Dach- und Wandebenen zeigen nach außen, was sowohl eine Richtung für die Linien vorgibt als auch die Punkte ordnen lässt. Dies wird beim Verbinden der Eckpunkte nützlich sein.

Implementierungsvermerk: Sortierschlüssel

In einer naiven Realisierung des Sortierens würden die Vergleichswerte – also Entfernung – immer wieder neu berechnet werden. Um diesen unnötigen Aufwand zu vermeiden, werden die Entfernungen nur einmal berechnet und im Punkt zwischengespeichert.

Da die Point-Datenstruktur jedoch bereits eine geschickte Größe von 32-Byte aufweist, ist es kontraproduktiv, ein weiteres Element hinzuzufügen. Stattdessen wird ein bestehendes Feld zunächst gesichert und dann überschrieben. Nach dem Sortieren werden alle Werte wiederhergestellt.

Abgrenzung der Teilstrecken

Wegen der Sortierung ist es nun möglich, die Abstände zwischen den entstehenden Teilstrecken zu bestimmen. Dazu betrachtet man einfach die Entfernung zwischen einem Punkt der Strecke und dem nächsten.

Was ist aber als Entfernung zu definieren – die Differenz der Linienparameter oder die echten Abstände der Punkte? Die Berechnung des Parameters beruht auf der Projektion der Punkte auf die Linie; möglicherweise werden dabei die echten Entfernungen unterschätzt. Man könnte zwar die Toleranz für Abstände entsprechend anpassen, aber es wird für besser gehalten, die echten Entfernungen zu berücksichtigen. Dadurch werden ungültige Linien öfter getrennt und folglich eher verworfen.

Das eigentliche Aufteilen ist unkompliziert. Man iteriert über alle Punkte. Wenn die Entfernung zwischen dem aktuellen und dem nächsten Punkt oberhalb eines Schwellwerts liegt, wird an dieser Stelle das aktuelle Geradenstück terminiert. Zuletzt wird das noch laufende Geradenstück abgeschlossen.

Beim Erzeugen eines Geradenstücks wird vermerkt:

- Länge des Stücks;
- Anzahl der Punkte;
- Koordinaten der Endpunkte;
- Abstände zu den benachbarten Geradenstücken.

Diese Eigenschaften dienen der Entscheidung, ob das Geradenstück plausibel ist und erhalten werden soll.

Entfernung unechter Teilstrecken

Wann ist ein Geradenstück ungültig und sollte folglich entfernt werden? Die häufigste Ursache dafür ist das zufällige Schneiden eines Geradenstücks mit einer anderen Linie. Dabei entstehen kurze, ungewollte Geradenstücke. Das erste Kriterium für die Ungültigkeit beruht also auf der Länge der Stücke. Der Schwellwert sollte jedoch nicht zu groß gewählt werden – ansonsten entfallen legitime, aber zerhackte Linien. Diese Überlegung ist auch die Wichtigste. Zu kurze Stücke können später immer noch verbunden werden. Es ist eher gefährlich, vermeintlich ungültige Stücke zu entfernen. Man verzichtet deshalb auf ausgefallenere Kriterien, beispielsweise die Dichte der Punkte oder das Verhältnis zwischen Länge und Abstand zum nächsten Stück.

4.4.3 Verschmelzung der Endpunkte

Nach dem Erzeugen der Geradenstücke und Entfernen der offensichtlich unechten Kandidaten werden die Endpunkte der Stücke weiter betrachtet. Im Idealfall würden sie genau an der Berandung der tatsächlichen Daten liegen und sich jeweils in einem Punkt treffen. Allerdings ist dem nicht so; man führt dies auf mehrere Ursachen zurück:

- Geringe (relativ zur obigen Idealvorstellung) LIDAR-Auflösung verhindert grundsätzlich, dass Endpunkte exakt aufeinander treffen.
- Leichte Winkelfehler in der Richtung einer Geraden bewirken, dass Endpunkte vor dem Ende angesetzt werden. Dies kommt daher, dass die Abweichung an den Enden der Gerade stärker wird, sodass Punkte dort fehlen.
- Unechte Grenzpunkte verursachen mitunter, dass Endpunkte jenseits der tatsächlichen Grenzen gelegt werden. Dies liegt daran, dass die Linie dort fälschlicherweise verlängert wurde. Auch die Massnahmen zur Bereinigung der Grenzpunkte sind nicht ausreichend, um das Problem ganz zu verhindern.

Diese Umstände haben zur Folge, dass benachbarte Endpunkte verschmolzen werden sollten. Um die Genauigkeit zu verbessern, werden auch Linienschnittpunkte berücksichtigt. Aus einer Anhäufung der End- und Schnittpunkte wird ein repräsentativer Punkt errechnet. Diese Arbeitsschritte werden nun weitergehend beschrieben.

Bestimmung relevanter Schnittpunkte

Warum sind Schnittpunkte von besonderer Bedeutung? Da in ihre Berechnung weitaus mehr Datenpunkte einfließen als bei den Endpunkten, entsprechen sie auch eher den tatsächlichen Gegebenheiten. Allerdings unterliegen diese Punkte ähnlichen Effekten wie bei den Endpunkten, sodass sie nicht alleine für eine sichere Aussage ausreichen. Man wird sie also in die Berechnung der Cluster eingehen lassen – allerdings mit sehr hohem Gewicht.

Im Anschluss kommt die Frage auf: Wenn Schnittpunkte verlässlicher sind und so stark gewichtet werden, warum sollten überhaupt noch die Endpunkte betrachtet werden? Es stellt sich heraus, dass nicht alle Dachkonfigurationen vollständig durch die Schnittpunkte der Trennlinien beschrieben werden können. Wo Schnittpunkte vorhanden sind, sollen sie verwendet werden, aber das Fehlen eines Schnittpunktes impliziert nicht, dass

sich dort kein Eckpunkt des Daches befindet. Es ist sinnvoll, beide als Grundlage für die Clusterbildung zu verwenden.

Schließlich bleibt noch zu klären, was mit 'relevanten' Schnittpunkten gemeint ist. Da es in komplexen Dächern (hier am Beispiel des FOM-Daches gezeigt) viele Flächen (bis zu 20) geben kann, ist auch die Anzahl der Linien (100) und deren Schnittpunkte (300) mitunter beträchtlich. Um die Ergebnisse leichter nachvollziehen zu können, sollen nur echte Schnittpunkte angezeigt und verwendet werden. Bedingung hierfür ist, dass beide Linien jeweils Geradenstücke hervorgebracht haben, die in der Nähe des Schnittpunktes liegen. Damit ist sichergestellt, dass der Schnittpunkt auch in der Nähe der echten Geometrie liegt.

Man kann also entscheiden, ob Schnittpunkte relevant sind. Die eigentliche Berechnung dieser Punkte wird mit Methoden der Geometrie durchgeführt.

Bildung der Cluster

Die Endpunkte und Schnittpunkte werden jeweils in Cluster eingefügt. Dazu prüft man, ob sich der Punkt im Einzugsbereich des Clusters befindet. Bezüglich der Rechenzeit ist die Anzahl der Punkte gering, also kann hierfür ein ineffizientes, aber exakt arbeitendes Kriterium angewandt werden. Für jeden Punkt des Clusters wird geprüft, ob der neue Punkt im Anziehungsradius des Clusterpunktes liegt. Die Radien sind unterschiedlich – Schnittpunkte bekommen einen stärkeren Einfluss, weil sie als sicherer angesehen werden.

Beim Eingliedern eines Punktes in einen Cluster werden drei Informationen abgelegt:

- Die Koordinaten des Punktes, inklusive Gewichtsfaktor;
- Herkunft des Geradenstücks. Insbesondere wird vermerkt, aus welchen Ebenen das Geradenstück hervorgeht (minimal zwei, maximal vier).
- Nachbarschaftsrelation. Aus den Clustern werden die Dach-Eckpunkte hervorgehen, die anschließend zu Polygonen verbunden werden. Dafür muss bekannt sein, welche der Eckpunkte zu verbinden sind. Diese Information ist in den Geradenstücken implizit enthalten, geht aber verloren, wenn nur die Eckpunkte betrachtet werden. Deswegen müssen die Verbindungen der Punkte explizit vermerkt werden, und zwar mit gerichteten Nachbarschaftsrelationen.

  Warum werden die Geradenstücke, die diese Information bereits enthielten, nicht weiterverwendet? Es stellt sich heraus, dass die Punkt-Darstellung einfacher zu handhaben ist. Betrachtet man die Relationen als Graph, wird klar, dass Verschmelzen der Knoten weniger aufwändig ist als das Bearbeiten aller betroffenen Kanten.

Berechnung des Repräsentanten

Sobald alle Schnitt- und Endpunkte auf die Cluster aufgeteilt wurden, können die Repräsentanten der Cluster berechnet werden. Alle Punkte im Cluster werden fortan nur durch diesen imaginären Punkt und die darin vermerkten Herkunfts- beziehungsweise Nachbarschaftsinformationen vertreten.

Die Berechnung der Koordinaten wird durch Schwerpunktbildung realisiert. Alle Endpunkte sind gleich gewichtet; eventuell vorhandene Schnittpunkte im Cluster werden stark bevorzugt. Es wurde erwägt, die Gewichte entsprechend der Entfernung zum Schwerpunkt anzupassen. Ein Vorteil gegenüber der einfachen Mittelwertrechnung ist jedoch nicht erkennbar, also wird hierauf verzichtet.

4.4.4 Erzeugen der Eckpunkte

Aus jedem Cluster geht genau ein Eckpunkt hervor. Bis auf die nicht mehr gespeicherten einzelnen Clusterpunkte ist die Darstellung auch identisch. Eine Trennung ist jedoch sinnvoll, weil unterschiedliche Operationen mit den Datenstrukturen durchgeführt werden.

4.4.5 Optimierung der Eckpunkte

An den resultierenden Eckpunkten besteht noch Optimierungsbedarf. Die folgenden Probleme sollten behoben werden:

- Manche äußeren Eckpunkte sind nicht verbunden;
- Große Lücken in den Daten werden nicht überbrückt;
- Eckpunkte liegen eventuell nahe beisammen.

Verbindung der äußeren Eckpunkte

Manche kurzen Wandstücke können nicht mittels Linienextraktion erkannt werden. In Abbildung 5.2 ist beispielsweise ein Gebäudevorsprung von nur 2 m sichtbar. Die Parameter der Hough-Transformation können nicht bedenkenlos angepasst werden, weil die Gefahr der Falschakzeptanz unechter Linien steigt.

Diese fehlenden Wände müssen anders erkannt werden. Wir erinnern uns, dass die Linienextraktion auf der Menge der Randpunkte durchgeführt wird. Punkte, die eine Wand definieren, werden aus dieser Menge entfernt. Fehlende Wände sind also dadurch gekennzeichnet, dass Anhäufungen verbleibender Punkte existieren.

Es ist allerdings nicht ratsam, hier eine erneute Linienextraktion durchzuführen. Auch mit veränderten Parametern sind die Aussichten, die Linie akkurat zu bestimmen, schlecht. Da die anfängliche Hough-Transformation die Punkte nicht gefunden hat, können es also nur so wenige sein, dass eine gesicherte Aussage bezüglich der Richtung der Linie nicht möglich ist.

Stattdessen werden alle Kombinationen zweier Eckpunkte aufgestellt. Die Verbindungslinie der Eckpunkte wird für eine Wand gehalten, wenn in ihrer Nähe die höchste Dichte verbliebener Randpunkte festgestellt wird. Die Eckpunkte werden sofort verbunden, anstatt zuerst eine neue Wand hinzuzufügen. Letzteres würde erfordern, den kompletten Eckpunkt-Schritt zu wiederholen, was wegen einer komplexeren Logik und höheren Laufzeit unerwünscht ist. Durch das Verbinden ist also das erste Problem zuverlässig gelöst.

Überbrückung großer Lücken

Große Lücken in den Dächern können beispielsweise durch Klimaanlagen oder Lüfter verursacht werden. In der Segmentierung würden solche Gebilde ausgelassen werden, weil sie dem restlichen Dach nicht ähneln. Folglich wäre zum Beispiel der Dachfirst in zwei Teile getrennt, was den Erfolg der Polygonerzeugung gefährden kann. Man muss also solche Eckpunkte verbinden oder verschmelzen.

Dieser Vorgang ist jedoch prinzipiell gefährlich. Das Verbinden der falschen Eckpunkte kann dazu führen, dass Polygone erzeugt werden, die das gesamte restliche Dach verdecken und somit das Gebäudemodell gänzlich unbrauchbar machen. Die sehr restriktiven Voraussetzungen lauten:

Verbindung Der aufzulösende Eckpunkt darf noch nicht mit mehr als einem Eckpunkt verbunden sein. Der Punkt, der ihn übernehmen soll, muss verbunden sein.

Lücke Die Verbindungslinie muss annähernd horizontal sein. Ferner muss sie entweder recht kurz sein oder einer Trennlinie ähnlich sein.

Herkunft Den beiden beteiligten Eckpunkten zuzüglich einem dritten benachbarten Referenz-Eckpunkt muss eine Ebene gemein sein, aus denen sie hervorgehen.

Lage Die zu füllende Lücke muss eine Verlängerung der Verbindung zum Referenz-Eckpunkt darstellen.

Sind diese Bedingungen erfüllt, werden alle Eigenschaften des aufzulösenden Eckpunktes auf den übernehmenden Eckpunkt übertragen; der Punkt wird von seinen Nachbarn getrennt und entfernt.

Hiermit ist auch das zweite Problem erfolgreich gelöst.

Verschmelzung benachbarter Eckpunkte

Es kann sein, dass Eckpunkte trotz des Clusterings nahe beisammen liegen. Angenommen, diese Punkte gehören zusammen, aber ihre Clusterpunkte liegen knapp außerhalb des Einzugsbereichs des anderen Clusters. In diesem Fall sind die Herkunftsinformationen der ursprünglichen Clusterpunkte vermutlich auf die beiden Eckpunkte aufgeteilt. Um ein korrektes Erzeugen der Polygone zu gewährleisten, müssen die Eckpunkte also verschmolzen werden.

Die einfachste Möglichkeit dazu ist, den Einflussbereich der Cluster zu erhöhen. Dies läuft aber Gefahr, auch naheliegende, unverwandte Eckpunkte zu verschmelzen. Stattdessen betrachtet man in einem separaten Schritt die Entfernung zwischen den fertigen Eckpunkten. Dadurch wird zwar der effektive Anziehungsradius der Cluster erhöht, allerdings nicht für die Punkte, die am Rand des Clusters liegen. Somit werden Ausreißer im Cluster toleriert, aber trotzdem alle zusammengehörigen Eckpunkte verschmolzen.

Die eigentliche Verschmelzung läuft wie im vorigen Absatz beschrieben ab.

4.4.6 Zusammenfassung

Die Eckpunkte entstehen aus den Kreuzungen der Schnittlinen der Wand- und Dachebenen. Um auch komplexe Dachkonfigurationen erkennen zu können, werden zusätzlich Endpunkte der Trennlinien berücksichtigt. Da die Eckpunkte bereits das endgültige Gebäudemodell vollständig bestimmen, sind sie kritisch für den Erfolg der Gebäudeanalyse. Durch Optimierungsschritte wird die Qualität dieser Punkte sichergestellt.

4.5 Erzeugung der Polygone

Polygone werden durch Zusammenknüpfen der Eckpunkte entlang ihrer Verbindungen erzeugt. Wegen ihrer unterschiedlichen Beschaffung werden Wandpolygone und Dachflächenpolygone behandelt. Details zu beiden Vorgängen folgen.

4.5.1 Wandpolygone

Wir erinnern uns, dass Wandebenen laut Abschnitt 4.3 als Ränder des Daches definiert sind. Die Ränder sind durch Verbindungen zweier Eckpunkte gegeben, die beide aus einer Wandebene hervorgingen. Zur Erzeugung der senkrecht begrenzten Wandpolygone genügt es also, diese Geradenstücke jeweils bis zum Boden zu verlängern.

Was versteht man unter Boden? Dieser kann über die Gebäudeunterkante definiert werden oder über die jeweilige Bodenhöhe in der Nähe der Dachtraufen. Wir überlegen, wann sich die beiden Methoden unterscheiden. Man kann annehmen, dass Gebäude eine durchgehende Keller- beziehungsweise Bodenebene aufweisen. Ist das Gebäude auf Flachland gebaut, so ist die erste Methode anwendbar. Zu betrachten ist noch der Fall, dass das Gebäude auf einer Schräge gebaut ist. Wenn es in einen Hang hineinragt, dürfen Wände problemlos bis zur gemeinsamen Fundamenthöhe durchgezogen werden – sie wären bei der Visualisierung durch das Gelände verdeckt. Wenn das Gebäude auf einem Hang steht, kann mit ALS-Daten leider nicht unterschieden werden, ob das Haus nicht etwa auf Stützen steht. Die Wände müssen auch bis zur tiefsten Stelle verlängert werden. Somit ist die aufwändigere zweite Methode überflüssig; es genügt, alle Wände bis zur gemeinsamen Fundamenthöhe anzusetzen.

Mit Hinblick auf einer späteren Texturierung der Wände sei angemerkt, dass die Orientierung der Wandpolygone bislang nicht definiert ist. Je nach Lage der beiden zu verbindenden Eckpunkte sind sie im Uhrzeigersinn verbunden oder umgekehrt. Um die korrekte Abbildung der Textur auf das Polygon zu ermöglichen, müsste eine einheitliche Orientierung geschaffen werden.

Dies wäre wie folgt möglich. Man findet die IntrLine-Trennlinie, in der die Dachtraufe liegt. Ihre Richtung bezüglich des Gebäudes ist wohldefiniert, weil die Normalenvektoren der Wände und Dachflächen nach außen gerichtet sind. Die korrekte Reihenfolge der beiden Eckpunkte wäre dann durch diese Richtung gegeben.

4.5.2 Dachflächen

Das Erzeugen der Dachflächenpolygone ist etwas komplexer. Trotz der sorgfältigen Bestimmung der Eckpunkte und deren Verbindungen ist es möglich, dass in seltenen Fällen Ergebnisse fehlen. Es wird ein bislang nicht vorliegender Algorithmus vorgestellt, der solche Problemstellen überwinden kann und Polygone erfolgreich schließt.

Algorithmus 2 : Erzeugung der Polygone einer Dachfläche.

Eingabe : dachebene
Ausgabe : polygon

```
1  eckpunkte ← { v | hervorgegangenAus(v, dachebene) }
2  random_shuffle(eckpunkte)
3  stable_sort(eckpunkte, absteigendeNachbarzahl)
4  solange eckpunkte ≠ ∅ tue
       // neues Polygon mit dem ersten Eckpunkt beginnen
5      v ← eckpunkte [0]
6      polygon ← {v}
7      nachHinten(eckpunkte, v) // vorzeitiges Abschließen verhindern
8      solange ¬ istGeschlossen(polygon) ∧ ¬ istUngültig(polygon) tue
9          Finde w | verbunden(v,w) ∧ nichtEnthalten(polygon, v,w)
10         wenn gefunden dann
11             eckpunkte ← eckpunkte \ {w}
12             polygon ← polygon ∪ {w}
13         sonst wenn erster Behebungsversuch dann
14             eckpunktReihenfolgeUmkehren(polygon)
15         sonst wenn zweiter Versuch dann
16             wenn |eckpunkte| = 1 dann
17                 w ← eckpunkte [0]
18                 eckpunkte ← eckpunkte \ {w}
19                 polygon ← polygon ∪ {w}
20             sonst
21                 fortan auch unverbundene Nachbarn zulassen.
22 zurück polygon
```

Überbrückung fehlender Verbindungen

Wir betrachten die Folgen des Fehlens einer gewissen Anzahl Verbindungen.

N = 0 Im einfachsten Fall sind alle Eckpunkte korrekt verbunden und können ohne Weiteres durchlaufen werden.

N = 1 Wenn genau eine Kante fehlt, können die beteiligten Eckpunkte nicht verbunden werden. Der erste Behebungsversuch beinhaltet, die Reihenfolge aller bisherigen Eckpunkte umzudrehen und am neuen Endpunkt fortzufahren. Das Polygon wird somit garantiert korrekt geschlossen werden.

Warum ist dem so? Nach der Voraussetzung liegt ein Kantenzug vor, der mit einer zusätzlichen Kante geschlossen wäre. Unabhängig vom Startpunkt werden die Eckpunkte bis hin zur fehlenden Kante verbunden. Nach dem Umkehren wird beim ursprünglichen Startpunkt fortgefahren; man erreicht dann die andere Seite der fehlenden Kante. Somit sind alle Eckpunkte erfasst und das Polygon vollständig.

N = 2 Dass zwei Kanten fehlen, ist unwahrscheinlich. Auch diese Situation kann jedoch oft überwunden werden. Wenn die fehlenden Kanten aneinander angrenzen, werden alle anderen Kanten geschlossen. Falls dann genau ein Eckpunkt verbleibt, kann man die fehlenden Verbindungen ergänzen.

N > 2 Anderenfalls gibt es also mindestens drei nicht benachbarte fehlende Kanten oder mehrere übriggebliebene Eckpunkte. Diese Situation würde mangelhafte Qualität der Eckpunkte und ihrer Verbindungen bezeugen. Das Polygon wäre dann nicht mehr mit Verlass zu schließen.

Man kann jedoch folgenden Versuch unternehmen: Beim Suchen des nächsten zu verbindenden Eckpunktes werden alle verbleibenden Punkte zugelassen, auch wenn sie nicht verbunden sind. Falls sich das Polygon selbst überschneidet, soll ein anderer Punkt gewählt werden. Schließlich wäre noch denkbar, den nächstgelegenen verbleibenden Eckpunkt zu verwenden. Beide Methoden sind allerdings sehr riskant – es kann trotzdem passieren, dass die falschen Punkte verbunden werden. Da dann wirklich entartete Polygone resultieren, die womöglich die restlichen Polygone verdecken und das Gesamtmodell unbrauchbar machen, ist diese Logik deaktiviert.

Folgen fehlender Eckpunkte

Was muss geschehen, damit Eckpunkte fehlen können? Entweder sind keine Schnitt- oder Endpunkte in der Nähe vorhanden oder es wird ein Eckpunkt fälschlicherweise verschmolzen oder entfernt. Beide Voraussetzungen sind unwahrscheinlich, weil die Toleranzen für die Punkte großzügig und die Verschmelzungskriterien sehr streng ausfallen. Es wird jedoch für möglich gehalten.

Was sind nun die Folgen fehlender Eckpunkte? Kann auch in dieser schwierigen Situation ein Polygon geschlossen werden? Zunächst muss klar sein, dass kein Polygon entstehen kann, das durch diesen nicht-vorhandenen Punkt verläuft. Ansonsten kann das Polygon aber tatsächlich erfolgreich erzeugt werden – die Situation reduziert sich auf den oben besprochenen Fall zweier fehlender Eckpunkte. Ein Großteil der Fläche wird also trotzdem abgedeckt.

Trennung disjunkter Flächen

Beim Zusammenführen ähnlicher Dachebenen wird billigend in Kauf genommen, disjunkte Flächen einer gemeinsamen Ebene zuzuweisen. Wir betrachten nun, wie sich diese Tatsache beim Erzeugen der Polygone auswirkt. Zunächst werden alle Eckpunkte der beteiligten Flächen gesammelt, weil ja alle aus der ursprünglichen Ebene hervorgehen. Dann wird ein Eckpunkt gewissermaßen zufällig ausgewählt und von dort ausgehend das erste Polygon erzeugt. Die Eckpunkte der anderen Flächen bleiben hiernach

übrig und werden wiederum genauso abgearbeitet. Da Eckpunkte disjunkter Flächen nicht verbunden sind, können hierbei keine Probleme auftreten.

Ein Nachteil ist allerdings zu verzeichnen. Beim zweiten Versuch, fehlende Verbindungen zu überwinden, ist Voraussetzung für das Verbinden der letzten beiden Kanten, dass nur noch ein Eckpunkt verbleibt. Diese Fehlerbehandlungsmöglichkeit existiert also nur für das zuletzt erzeugte Polygon aus der Menge der Dachflächen, die in einer Ebene liegen. Trotz dieser Einschränkung können also disjunkte Flächen problemlos behandelt werden.

Implementierungsvermerk: Darstellung der Polygone

Die Polygone bestehen aus einer Liste der Eckpunkte. Um am Ende sicherstellen zu können, dass sie korrekt geschlossen sind, werden sie so erzeugt, dass der erste Punkt identisch mit dem letzten sein sollte.

Zur Visualisierung wird der `fill3`-Aufruf von Matlab verwendet. Der Rasterungs-Algorithmus dieser Funktion ist unbekannt; es sind aber keine Einschränkungen bezüglich der Polygoneigenschaften dokumentiert. Da die Polygone sowieso fast immer konvex oder 'gutartig' sind, wird auf eine aufwändige separate Triangulierung verzichtet.

4.6 Zusammenfassung

Mit der Erzeugung der Polygone ist die Analyse und Rekonstruktion des Daches abgeschlossen. Im Folgenden werden die Ergebnisse untersucht und bewertet.

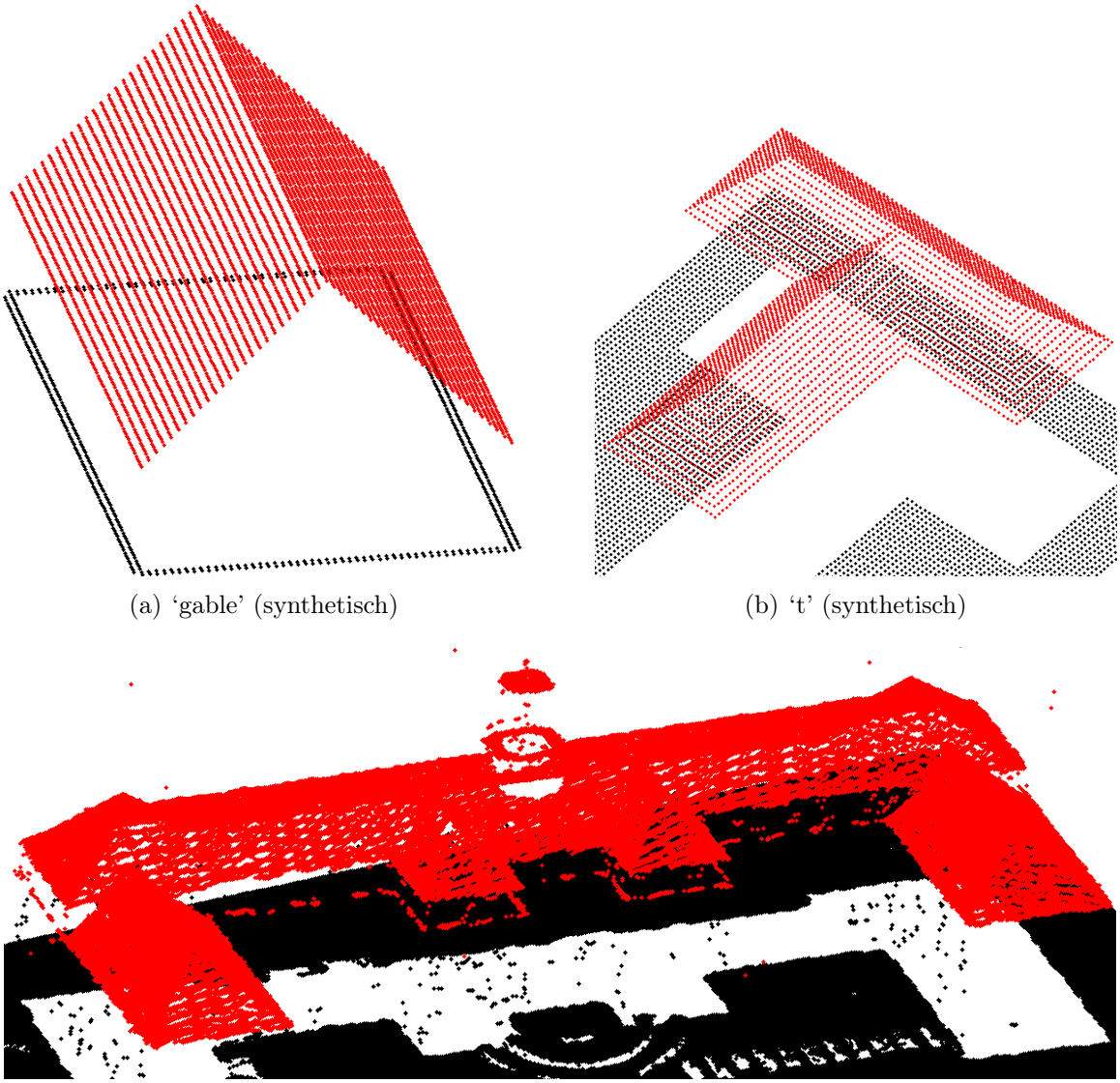
Kapitel 5

Bewertung

5.1 Datensätze

Es stehen drei Datensätze zur Verfügung, die in Abbildung 5.1 als Punktwolken dargestellt sind und im Folgenden beschrieben werden:

- ‘gable’ ist ein künstlich erzeugtes Giebeldach. Dieser einfachste Dachtyp stellt die grundsätzliche Funktionstüchtigkeit des Verfahrens unter Beweis und vereinfacht die Fehlersuche.
- ‘t’ ist ein synthetisches Gebäude, das aus zwei orthogonalen Walmdächern zusammengesetzt ist. Dieser Dachtyp bietet etwas mehr Herausforderungen, weil er weitaus mehr Dachflächen beinhaltet und Trennlinien im Inneren des Daches aufweist.
- ‘fom’ ist ein Abbild des FOM-Hauptgebäudes in Ettlingen. Hier gibt es eine Vielzahl von Problemen zu bewältigen, die in Abbildung 5.2 ersichtlich sind und nachfolgend aufgezählt werden:



(a) ‘gable’ (synthetisch)

(b) ‘t’ (synthetisch)

(c) ‘fom’ (TopoSys Falcon II)

Abbildung 5.1: Datensätze als Punktwolke dargestellt. (Punkte auf Bodenhöhe sind zur besseren Erkennbarkeit schwarz gefärbt)

(a) Gebäudesüdseite

(b) Seitenansicht

Abbildung 5.2: Herausforderungen des FOM-Gebäudes. (Nummerierte Problemstellen werden im Text erläutert)

1. Das Gebäude ist architektonisch recht komplex;
2. Die nahegelegenen Bäume schneiden den Rand an;
3. Ein zweiter Dachrand erschwert die Analyse des Daches;
4. Manche Dachlinien sind sehr kurz (2 m);
5. Der Turm verursacht eine Lücke im segmentierten Dach;
6. Einige Wände sind auch nur 2 m lang;
7. Die Dachtraufen liegen teilweise auf unterschiedlicher Höhe (Step Edge).

5.1.1 Eigenschaften

Die synthetischen Daten werden mit einer Auflösung von 20 cm erzeugt. Bei den realen TopoSys-Daten beobachtet man auch einen regelmäßigen Punktabstand von 20 cm, was aber nicht der tatsächlichen Auflösung entspricht. Die Daten sind nur in nachverarbeiteter Form erhältlich, bei der die ursprünglichen Koordinaten quantisiert wurden. Die echte Auflösung wird in Abschnitt 5.5.3 hergeleitet.

5.2 Ergebnisse

Das Verfahren wurde zunächst auf den einfacheren Daten getestet und dann für die komplexeren Daten angepasst. Es sei festgehalten, dass der Übergang zu den realen Daten zwar leichte Parameteranpassungen erforderte, aber keine grundlegenden Änderungen. Alle verschiedenartigen Datensätze können mit den gleichen Parametern bearbeitet werden, was für die Robustheit des Verfahrens spricht. Die Gebäude werden für einen Beobachter ersichtlich korrekt rekonstruiert; die Ergebnisse sind in Abbildung 5.3 dargestellt.

5.3 Erzielte Genauigkeit

> *Ich habe einen ganz einfachen Geschmack: Ich bin immer mit dem Besten zufrieden.*
> *– Oscar Wilde*

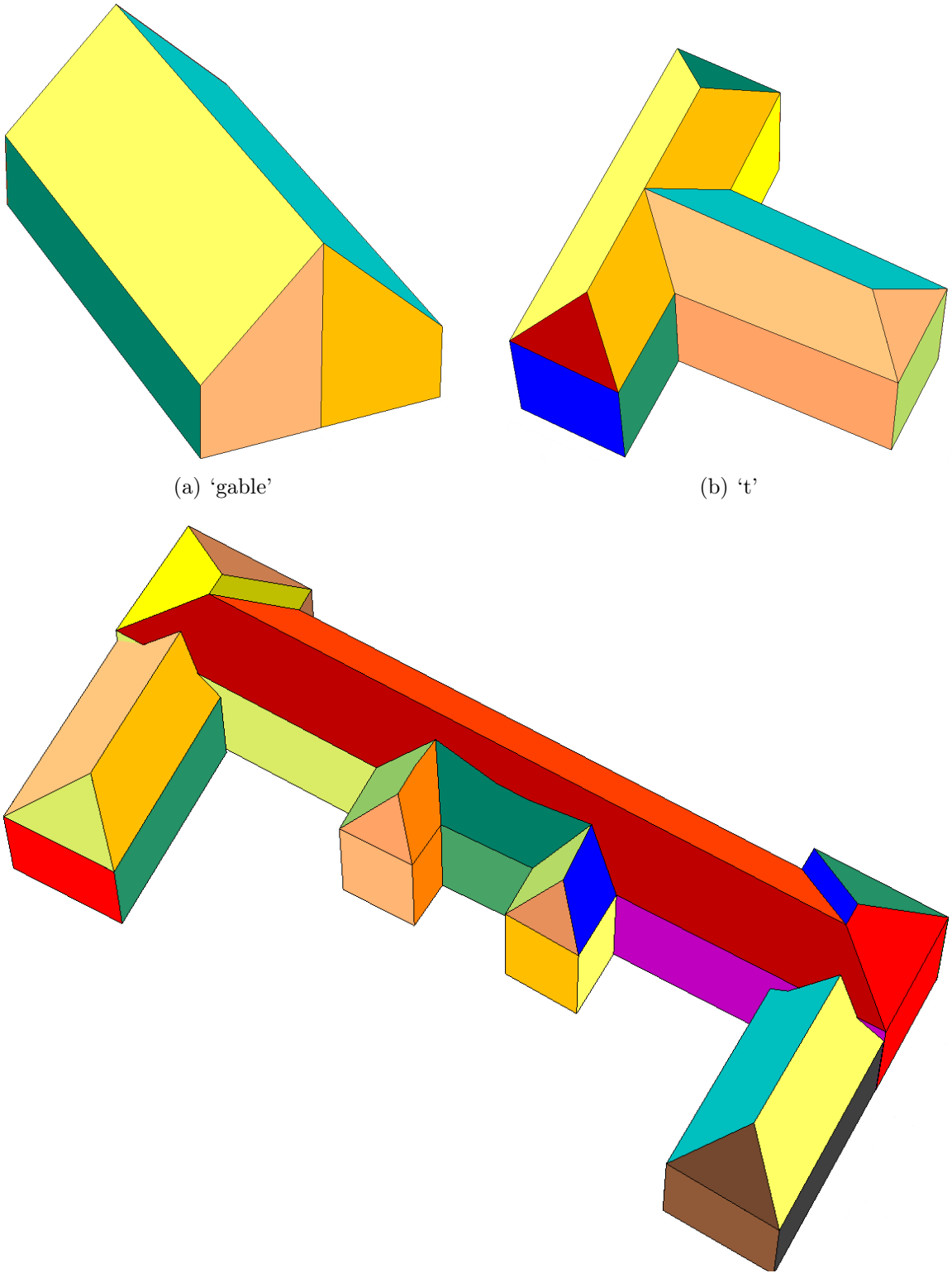
Da bei den synthetischen Daten ground truth vorliegt, kann die Genauigkeit der ermittelten Polygone direkt festgestellt werden. Die Lage- und Winkelfehler der Wände des Giebeldaches sind in Tabelle 5.1 angegeben.

Tabelle 5.1: Lage- und Winkelfehler der ermittelten ‘gable’ Wände.

Wand	Lage [Meter]	Winkel [Grad]
Nord	$\Delta y = -0,2$	0,0
Ost	$\Delta x_{\text{oben}} = -0,087, \Delta x_{\text{unten}} = 0,09$	0,5
Süd	$\Delta y = 0,6$	0,0
West	$\Delta x = 0,4$	0,0

Die Verschiebungen der Wände um ein bis drei Punkte werden durch den Segmentierer verursacht – Randpunkte gehören wegen der scharfen Trennung des Dachrandes nicht zum Gebäude-Segment. Die darauffolgenden Schritte sind alle präzise bis auf die Hough-Transformation, die den leichten Fehler in der Ostwand verursacht. Der Winkelfehler liegt jedoch innerhalb der erlaubten Toleranzen, die sogar durch Einsatz größerer Akkumulatoren verringert werden können.

Bei den realen Daten muss mangels ground truth eine andere Metrik gefunden werden. Mann kann prüfen, inwiefern der Winkel zwischen zwei für orthogonal gehaltenen Wänden dieser Vermutung entspricht. Beim ‘fom’-Datensatz werden hier exemplarisch die



(a) ‘gable’ (b) ‘t’

(c) ‘fom’

Abbildung 5.3: Rekonstruierte Gebäude. (Darstellung aus den extrahierten Flächen)

äußeren Ecken des westlichen Flügels betrachtet. Die Winkel zwischen den beteiligten Wänden liegen jeweils 3 und 1,9 Grad entfernt von 90 Grad. Man beachte, dass diese Winkel mittels eines Gebäude-Modells 'begradigt' werden könnten. An dieser Stelle geht es allerdings nur um ein Maß für die Genauigkeit der Gebäuderekonstruktion. Diese so festgestellten Fehler sind tatsächlich anhand leicht gekrümmter Kanten sichtbar, was allerdings dem Nutzen des Gebäudemodells nicht abträglich ist.

5.4 Schwachstellen und fehlerhafte Ergebnisse

5.4.1 Zum 'gable'-Gebäude

Bei diesem einfachen Gebäude gibt es keine Fehler, aber eine auffällige Stelle.

Die Endstücke des Gebäudes bestehen jeweils aus zwei halben Wänden, was an der unterschiedlichen Färbung sichtbar wird. Wir erinnern uns, dass Wände aus den verlängerten Dachtraufen hervorgehen. Wegen ihren unterschiedlichen Steigungen gelten die beiden Dachtraufen an einem Endstück als verschieden, also entstehen auch zwei Wände. Falls gewollt, könnten diese zusammengeführt werden.

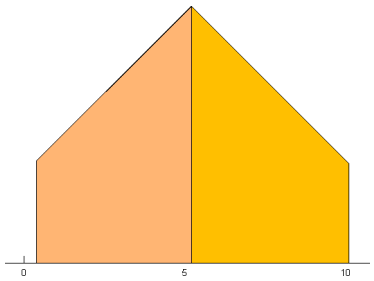


Abbildung 5.4: Zwei 'Wände' am Ende des Giebeldachs.

5.4.2 Zum 't'-Gebäude

In dem zweiten Datensatz sind zwei etwas schiefere Wände auffällig. Wie kommt es dazu? Bei genauer Betrachtung der Analyseergebnisse in Abbildung 5.5 stellt man fest, dass die Wand der grünen Fläche etwas verschoben ist. Dadurch sind die roten Schnittpunkte auch verfälscht, die dann die grünen Eckpunkte mitziehen. Da die Wand korrekt per Hough-Transformation erkannt wurde, liegt der Fehler im Erzeugen oder Verschmelzen der Linien. Die daraus folgende vertikale Verschiebung der gelben und blauen Flächen liegt jedoch in der Größenordnung der Punktauflösung, sodass sie toleriert wird.

5.4.3 Zum 'fom'-Gebäude

Hier sind die folgenden Unstimmigkeiten zu verzeichnen:

- Der linke Flügel trifft nicht genau auf den Hauptteil des Gebäudes (Abbildung 5.6(a)). Dies wird durch zu frühes Terminieren der Linie verursacht, weil kein passender Datenpunkt mehr gefunden wird. Solche Stellen könnten erkannt werden, weil dort weniger als drei Linien angrenzen. Man verzichtet jedoch auf den Versuch, diese Linien zu verlängern, weil das Dach dadurch eventuell ungewollt verändert wird.

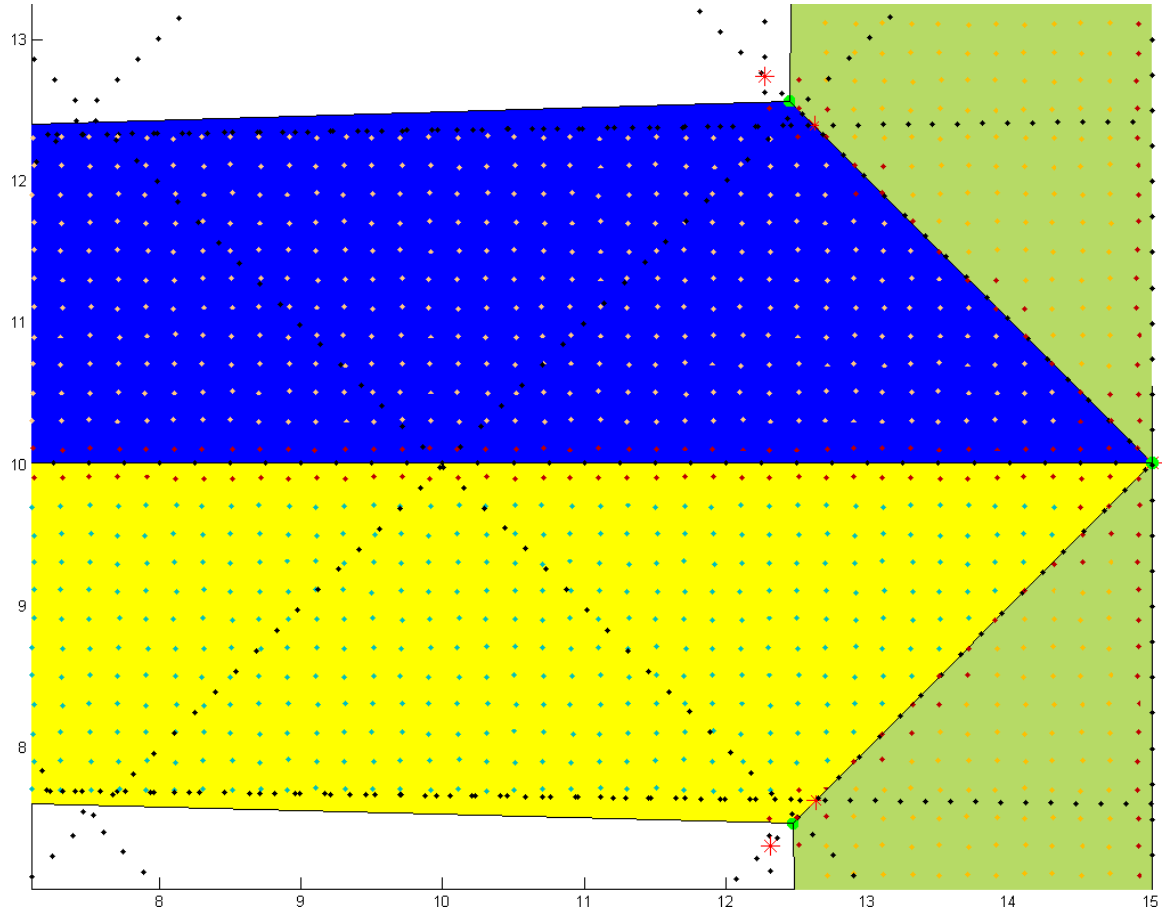


Abbildung 5.5: Verschobene und schiefe Wände beim ‘t’-Gebäude, sichtbar durch die nicht-horizontalen Grenzen der gelben und blauen Dachflächen.

- Der rechte Flügel weist genau das umgekehrte Problem auf (Abbildung 5.6(b)).
- Die Eckpunkte des Dachvorsprungs werden mit anderen Eckpunkten verschmolzen (Abbildung 5.6(c)). Zur Erläuterung: Die beiden Endpunkte in der Mitte (1, 2) liegen auf der korrekten Grenzlinie zwischen der Dachfläche und dem Vordach (3). Die Schnittpunkte (4) dieser Linie mit den benachbarten Grenzen sind korrekt erkannt worden, werden aber aufgrund ihrer Nähe zu den Anschlussstellen der Flügel (5) mit ihnen verschmolzen. Es ist unklar, wie dies im Allgemeinen verhindert werden kann, da ansonsten das Verschmelzen naheliegender Punkte nötig und erwünscht ist. Die etwas ungenaue Form der Flügel sowie die sattelförmige Grenze des Vordaches müssen also hingenommen werden.

  In diesem Bild sind übrigens Lücken sichtbar, die durch den Turm (6) und einen Lüfter (7) verursacht worden sind und erfolgreich überbrückt werden konnten.

Diese Problemstellen trüben jedoch nicht das Gesamtbild des komplexen Gebäudes, das ansonsten sehr präzise rekonstruiert werden kann.

5.5 Toleranz gegenüber Rauschen

Es wird untersucht, welche Anforderungen das Verfahren an die Daten stellt. Insbesondere ist von Interesse, wie hoch die Auflösung und die Genauigkeit sein müssen, damit noch gültige Ergebnisse produziert werden können.

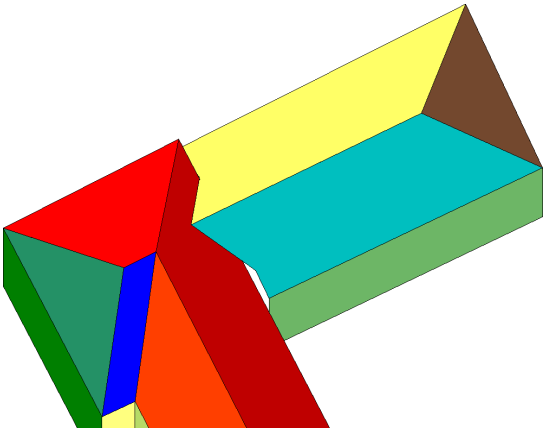
5.5.1 Datensätze

Da nur Datensätze der Auflösung 20 cm vorliegen, müssen synthetische Daten für die Untersuchungen erzeugt werden. Um den Realitätsbezug nicht zu verlieren, sollen sie den Gegebenheiten von ALS-Daten entsprechen. Insbesondere beinhaltet dies unterschiedliche Auflösungen auf allen Achsen. Die Unterprogramme zur Erzeugung synthetischer Daten unterstützen dies jedoch nicht, sodass der Effekt durch Hinzufügen von Rauschen nachgebildet wird.

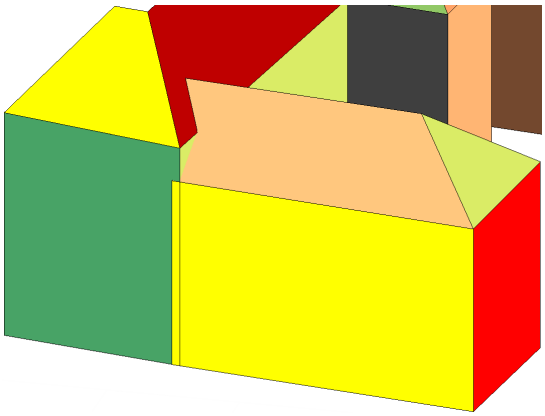
5.5.2 Anwendung des Rauschens

Den Daten des idealen Gebäudemodells wird mittelwertfreies Rauschen hinzugefügt. Um die unterschiedlichen Auflösungen nachzubilden, werden die Amplituden des Rauschens achsenweise skaliert. Bevor im nächsten Abschnitt diese Werte hergeleitet werden, sind noch die Folgen dieses Modells zu besprechen.

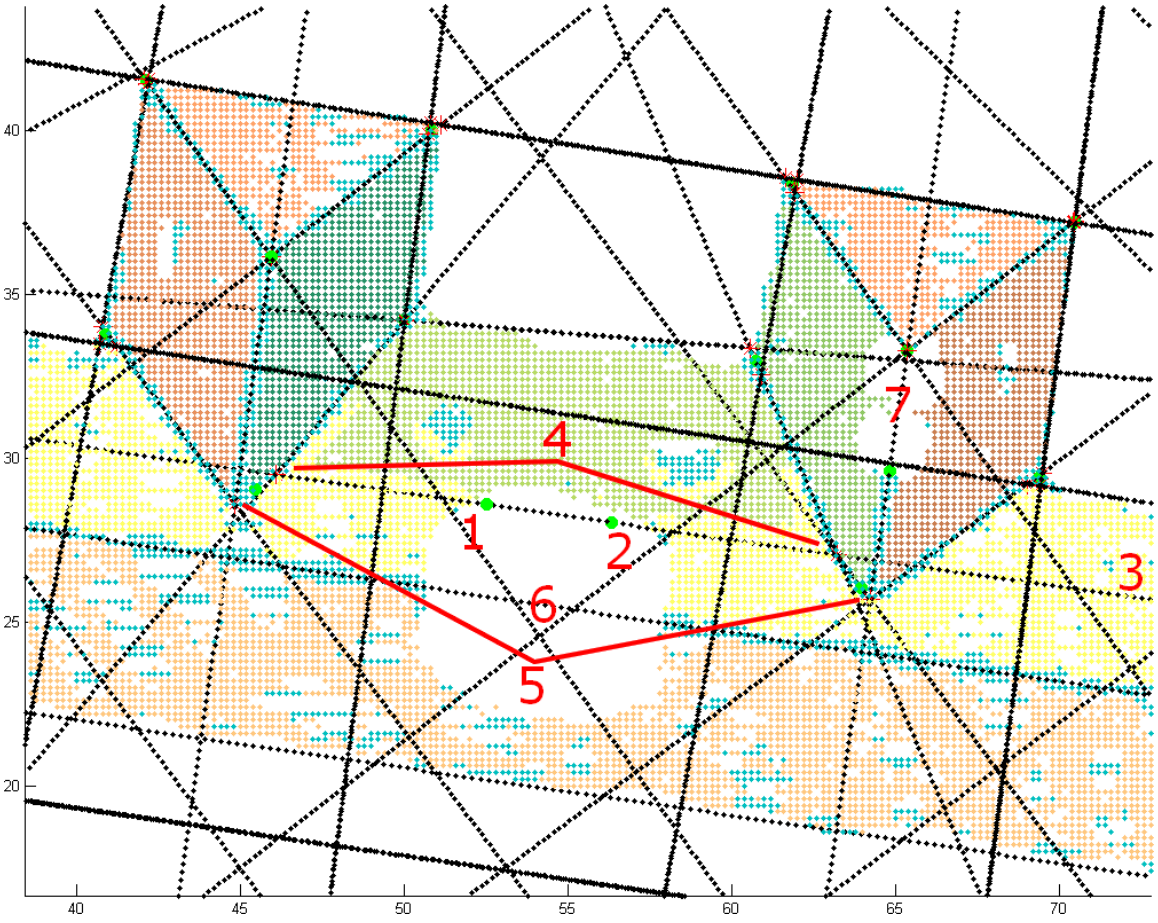
Die resultierenden verrauschten Daten sind mitunter ‘unfreundlicher’ als reale Daten und bereiten eher Schwierigkeiten. Dies rührt daher, dass alle Koordinaten unabhängig voneinander gestört werden. Um das Problem zu sehen, stellen wir uns den Versuch vor, einen tatsächlichen Dachpunkt zu erfassen. Wird beispielsweise die x-Koordinate falsch abgetastet, erhält man einen anderen Punkt, der auch entsprechend der Neigung



(a) Linker Flügel, getrennt



(b) Rechter Flügel, zu weit



(c) Vordach

Abbildung 5.6: Falsch modellierte Stellen beim FOM-Dach.

des Daches einen anderen z-Wert aufweist. Beim Nachbilden dieses Effekts mittels verrauschter Daten wäre es jedoch möglich, dass die z-Komponente zusätzlich in die falsche Richtung bewegt wird. Dementsprechend ist die Wirkung des Rauschens mit einer gegebenen Amplitude gravierender als eine Reduktion der Auflösung. Es wird also untersucht, welche Sensorauflösung zu erwarten ist; dieser Wert soll dann als obere Schranke für die Rauschamplituden dienen.

5.5.3 Herleitung der ALS-Auflösung

Als Referenz wird das gängige kommerzielle System 'TopoSys Falcon II' verwendet. Die Flugplattform ist normalerweise ein Starrflügler (Flugzeug). Um die Auflösung quer zur Flugrichtung zu verbessern, wird der Sensor elliptisch geschwenkt.

Die Auflösung entlang der Flugrichtung ist gegeben durch $v \cdot t_{sc}$; mit der Fluggeschwingkeit v=250 km/h und der Scanfrequenz $1/t_{sc}$=653 Hz erhält man 11 cm. Quer zur Flugrichtung ist die Auflösung bestimmt durch $h\frac{\theta}{N-1}$; die Flughöhe h=300 m, den Schwenkbereich θ=±7,15 Grad und die Faserreihungsanzahl N=128 ergeben 57 cm. Wehr und Lohr (1999) Die Messgenauigkeit wird mit 2 cm angegeben TopoSys GmbH.

Man beachte, dass die obigen Angaben relativ konservativ gewählt worden sind. Falls das System vom Drehflügler (Helikopter) getragen wird, ist sowohl die Flughöhe als auch die Geschwindigkeit weitaus geringer, was die Auflösungen entsprechend steigert. Da auch das vorausgesetzte System nicht mehr aktuell ist[1], sind diese Werte pessistisch.

5.5.4 Ergebnisse

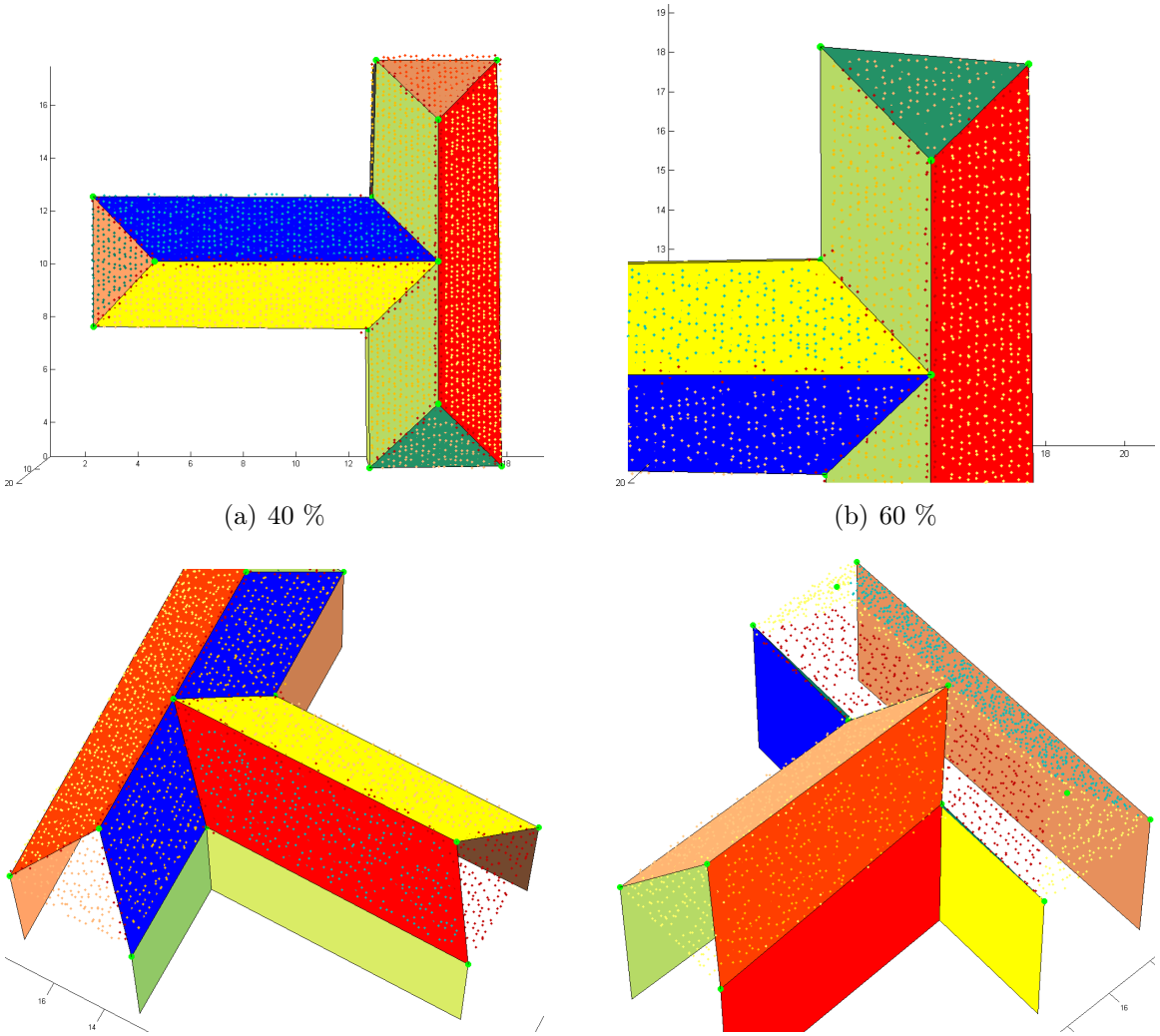
Für einige Rauschamplituden bis hin zu den oberen Schranken wurden die Gebäudepunktwolken erzeugt und mit dem unveränderten Verfahren analysiert. Die qualitativen Ergebnisse werden in Tabelle 5.2 aufgeführt.

Tabelle 5.2: Rekonstruktionserfolg beim verrauschten 't'-Datensatz.

Vorfaktor	Ergebnis	Fehler
40 %	Einwandfrei	–
60 %	Sehr gut	Eine etwas verschobene Ecke.
80 %	Befriedigend	Zwei Wände und Dachflächen fehlen.
100 %	Mangelhaft	Drei Wände und die meisten Dachflächen fehlen.

Die Ergebnisse sind in Abbildung 5.7 dargestellt, wobei das Augenmerk auf die fehlerhaften Stellen gelegt wird.

[1]Laut Angaben des Herstellers ist das System seit 2000 im Gebrauch und seit 2004 auf dem Markt. Das Nachfolgesystem Falcon III soll Ende 2006 verfügbar sein.



(a) 40 %

(b) 60 %

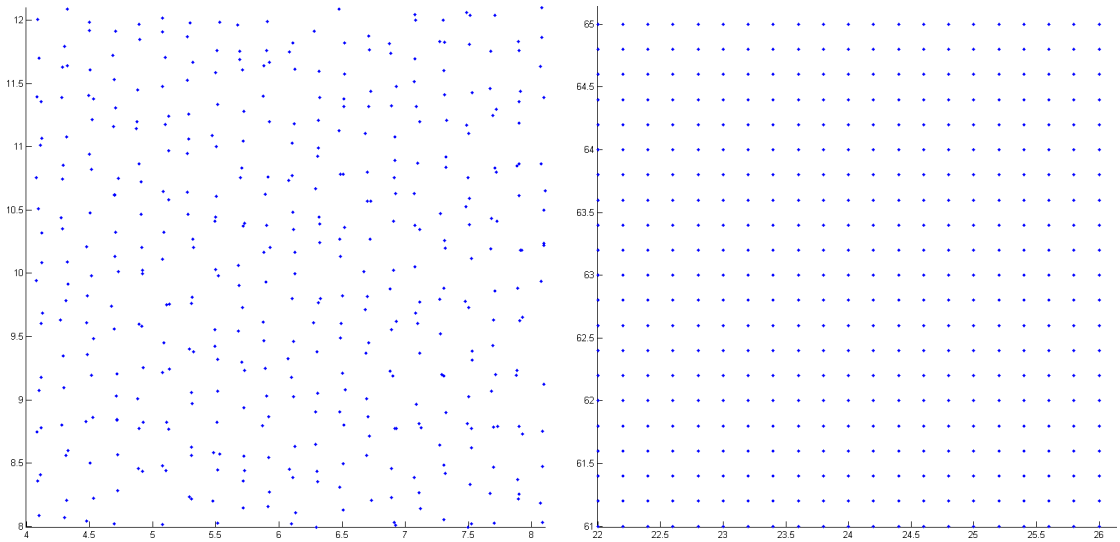
(c) 80 %

(d) 100 %

Abbildung 5.7: Ermittelte Flächen beim verrauschten ‘t’-Datensatz.

5.5.5 Bewertung

Kann man aus diesen Ergebnissen auf erwartetes Verhalten bei realen Daten schließen? Zunächst wird vermutet, dass um $0,6R$ verrauschte Daten mindestens so viele Probleme bereiten wie reale Daten mit den Koordinatenauflösungen R. Dies stützt sich auf die Beobachtung, dass benachbarte Punkte im Extremfall jeweils zueinander verschoben werden, was die effektive Wirkung des Rauschens verdoppelt. Man betrachte die lückenhafte Verteilung der verrauschten Daten (Abbildung 5.8(a)); dahingegen sind die realen Daten (Abbildung 5.8(b)) aufgrund der TopoSys-Nachbearbeitung flächendeckend uniform.



(a) synthetisch, verrauscht (b) TopoSys, nachbearbeitet

Abbildung 5.8: x, y-Punktabstand der verrauschten Daten. (zeigt größere Lücken darin)

Hinzu kommt der vorhin erwähnte Effekt, dass das Verrauschen der Koordinaten unabhängig und somit potenziell stärker wirkt. Da das Verfahren unter diesen Bedingungen trotzdem noch gute und zuverlässige Ergebnisse liefern kann, wird erwartet, dass reale Datensätze mit den anfangs hergeleiteten Auflösungen erfolgreich bearbeitet werden können.

5.6 Performanz

5.6.1 Testbedingungen

Das Programm wird mit Visual C++ 2005 (Version 8.0.50727.762) übersetzt. Für die Performanz sind die folgenden Kommandozeilenparameter relevant:

```
/O2 /Ob2 /Oity /GL /GF /FD /EHsc /MD /arch:SSE /fp:fast /c /Zi /TP
```

Die zur Verfügung stehenden Testsysteme sind in Tabelle 5.3 aufgeführt.

Tabelle 5.3: Spezifikationen der Testsysteme.

CPU	Speicher	Betriebssystem
Athlon 64 3000+ (1.800 MHz)	1 GByte	WinXP
Intel Core Duo 6400 (2.133 MHz x2)	2 GByte	WinXP x64

5.6.2 Rechenzeit

Es ist von Interesse, wie lange auf die Ergebnisse der Analyse gewartet werden muss. Diese Information wird über die Angabe der 'CPU-Zeit' im Windows Task Manager angenähert. Die erhaltene Zahl ist nur auf eine Sekunde genau, was allerdings für die gewünschte grobe Zeitabschätzung genügt.

Bei der Analyse und Rekonstruktion des FOM-Daches mit 41941 Punkten beobachtet man CPU-Zeiten von 13 und 10 Sekunden. Diese Ergebnisse stehen ungefähr im Verhältnis der CPU-Taktfrequenzen[2].

Unter Hinzunahme der Vorverarbeitung steigt die Rechenzeit für das langsamere System auf insgesamt 40 Sekunden. Dabei werden die ursprünglichen 216952 Punkte klassifiziert und aufbereitet. Wegen der Zwischenspeicherung der Ergebnisse fällt dieser Aufwand jedoch nur einmal für die gesamte Szene an, statt für jedes einzelne darin enthaltene Gebäude.

Die Verarbeitungszeit ist gering genug, dass die Vorgabe der Benutzerfreundlichkeit erfüllt ist. Wir wollen jedoch betrachten, worauf ein Großteil der Zeit entfällt.

5.6.3 Engpässe

> *20 % of lines of source code account for 80 % of run time of the program.*
> *– Jon Bentley Bentley (1985)*

Eine genaue Verrechnung der vergangenen Zeit wird mit dem zuvor implementierten und in Wassenberg (2005) beschriebenen Zeitmesser erreicht. Es stellt sich heraus, dass die Rechenzeit fast ausschließlich durch die Ermittlung der optimierten Ebenen bestimmt wird (89 %)! Darauf folgt die Segmentierung der Punkte (4,5 %) und das Erzeugen der Wände (4,2 %).

Was kostet im erstgenannten Schritt die meiste Zeit? Die Segmentierung und Berechnung der Plane-Informationen (beispielsweise die konvexe Hülle) verbrauchen 61 % dieser Zeit, während das Korrigieren falsch zugeteilter Punkte 34 % verbucht.

Welche Erkenntisse können hieraus gezogen werden? Tatsächlich sind es nur wenige Stellen, die das Gros der Rechenzeit ausmachen. Ferner stellt man fest, dass alle auf Rohdaten zugreifen. Die große Menge der ursprünglichen Punkte verursacht zwangsläufig hohe Laufzeitkosten. Später sind die Punkte beispielsweise zu Ebenen zusammengefasst, sodass die Datenmenge beträchtlich reduziert ist. Folglich ist also klar, warum die ersten Schritte der Verarbeitungskette aufwändiger sind.

[2] Auch wenn dies keinesfalls ein verlässliches Maß ist, weil die Prozessoren unterschiedliche CPI – Cycles per Instruction – aufweisen!

5.6.4 Optimierungsmaßnahmen

> *More computing sins are committed in the name of efficiency (without necessarily achieving it) than for any other single reason – including blind stupidity.*
> *– William Wulf* [3]

Man beachte, dass die Implementierung größtenteils unoptimiert ist. Echtzeitfähigkeit ist nicht das Ziel der Arbeit. Dem Autor sind jedoch einige Fallen in der Sprache C++ bewusst, die unnötige Performance-Einbußen verursachen und leicht vermieden werden können. Diese Maßnahmen wurden bereits in der Entwicklung getroffen; Beispiele hierfür sind:

- Objekte werden bevorzugt initialisiert statt zugewiesen, um unnötige Aufrufe des Default-Konstruktors zu verhindern.
- Die Parameterübergabe geschieht mittels Verweis, damit die Objekte nicht kopiert werden. Notabene: Viele Objekte sind ohnehin leichtgewichtig und können direkt übergeben werden; siehe dazu Abschnitt 5.6.4.
- Speicher wird vorreserviert, um viele kleinere Vergrößerungen eines STL-Containers zu vermeiden. Wegen der ansonsten verwendeten Strategie, die Speichergröße lediglich um 50 % auszuweiten, sind die Ersparnisse beträchtlich. Beim Einfügen von 100 Elementen können $\left\lceil \frac{\log 100}{\log 1{,}5} \right\rceil = 12$ Neuallokationen vermieden werden.

Mit diesen Maßnahmen werden generell unnötige Verlangsamungen vermieden. Um die Gesamtverarbeitungszeit weiter zu reduzieren, müsste man jedoch die oben identifizierten Engpässe vermeiden. Die hohen Laufzeitkosten dieser Programmteile erscheinen unberechtigt, sodass beträchtliche Verbesserungen erwartet werden können.

Implementierungsvermerk: Isolierung der Objekte

Zwecks Abschottung der Programm-Module bestehen einige Klassen nur aus einem referenzgezählten Zeiger auf die Implementierung. (Coplien, 1998, Seiten 4–7) Damit bleiben Interna der Klasse nach außen unsichtbar und werden nicht wie sonst über die Klassendeklaration preisgegeben. Dies reduziert Abhängigkeiten und kann die Übersetzungszeiten verringern. Das Prinzip wird mit dem folgenden Code-Ausschnitt veranschaulicht:

```
struct IntrLineSegment
{
   IntrLineSegment(IntrLine il, ..);

   IntrLine parentLine() const;
   ..

   class Impl;
   boost::shared_ptr<Impl> pimpl;
};
```

[3] Nebenbei sei wegen des Themas dieses zitierten Artikels Wulf (1972) mit Ironie festgehalten, dass die hier entwickelte Software tatsächlich goto-Anweisungen enthält. Mögen diese begründeten Einzelfälle zu den ‘gutartigen’ Verwendungen des Sprachkonstrukts zählen.

```
class IntrLineSegment::Impl
{
public:
   Impl(IntrLine parent_line, ..);
   ..
};

IntrLineSegment::IntrLineSegment(IntrLine il, ..)
: pimpl(new IntrLineSegment::Impl(il, ..))
{
}

IntrLine IntrLineSegment::parentLine() const
{
   return pimpl->m_parent_line;
}
```

5.6.5 Speicherverbrauch

Ein Nachteil der Punktwolkendarstellung liegt im Speicherverbrauch. Bei einem Bild sind die (x, y)-Koordinaten der Pixel implizit durch ihre Position gegeben; bei Punktwolken müssen sie hingegen explizit mitgespeichert werden. Wir wollen nun untersuchen, ob dieser Umstand problematisch werden kann.

Vorausgesetzt sei eine Szene von 100 m×100 m mit 20 cm Auflösung und eine Darstellung der Koordinaten als 32-Bit IEEE-754 Gleitkommazahl. Es gibt also $2,5 \times 10^5$ Punkte, die insgesamt 2 MByte mehr Speicher benötigen als ein vergleichbares Bild. Bei heutigen Speichergrößen von 1 GByte und mehr stellt dies kein Problem dar.

Neben dieser Überlegung soll auch der reale Verbrauch gemessen werden. Als Indikator dient der Windows Performance Counter “Private Bytes”. Dieser ist eine gute Annäherung an die tatsächliche Größe des Working Sets, weil er – im Gegensatz zu den Angaben des Task Managers – ausgelagerte Speicherseiten mitzählt.

Am Ende der Verarbeitung des FOM-Datensatzes stellt man einen Verbrauch von 15 MByte fest; der Maximalwert beträgt 25 MByte (diese temporäre Spitze wird vermutlich durch I/O-Puffer verursacht). Inklusive Vorverarbeitung sind es 22 MByte, maximal 63 MByte. Diese Ergebnisse bestätigen auch die Machbarkeit der Gebäudeanalyse unter Verwendung von Punktwolken.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Ergebnisse

Der Beitrag dieser Arbeit liegt in der Entwicklung eines soliden Verfahrens zur automatischen Gebäudemodellierung. Einige Komponenten sind bereits aus der Literatur bekannt, wurden aber nach Wissen des Autors noch nicht in dieser Form verwendet. Insbesondere ermöglichen kovarianzbasiertes Schätzen der Ausgleichsebenen und eine hochwertige Realisierung der Hough-Transformation präzise Rekonstruktion der Gebäude. Zu den Vorteilen gegenüber bestehenden Verfahren zählen:

Automatisierung Eingreifen des Benutzers ist nicht erforderlich.

Genügsamkeit Außer den Punktdaten sind keine weiteren Informationen[1] nötig.

Allgemeinheit Nur wenige Annahmen über Gebäude oder Daten werden getroffen.

Genauigkeit Auflösung und Präzision der ursprünglichen Daten werden beibehalten.

Als Nachteil ist lediglich der Rechenaufwand zu nennen. Die Verarbeitungszeit ist bei kleineren Szenen akzeptabel, aber für weiträumige Aufklärung etwas unpraktisch. Durch die genannten Optimierungsmaßnahmen kann aber Abhilfe geschaffen werden.

6.2 Diskussion

6.2.1 Softwaretechnik

Die entwickelte Software besteht aus 60.000 Zeilen C++. Hiervon entfallen 4K auf die externe Bibliothek ANN (Approximate Nearest Neighbors für Punktnachbarschaft), 7K auf Wildmagic (Geometrie) und 39K auf wiederverwendete Teile vorangegangener Projekte (hauptsächlich Speicherverwaltung, I/O- und Debug-Unterstützung). Die restlichen 10K Zeilen dienen der Gebäudeanalyse.

[1] Beispielsweise Gebäudeumrisse oder LIDAR-Reflektivität.

Bei dieser Komplexität sind Methoden des Software-Engineerings unerlässlich. Kapselung der Objekte verhindert Missbrauch und Abhängigkeiten. Abschottung der Module reduziert Übersetzungszeiten. Regressionstests und automatisierte Zusicherungen bringen viele Fehler zum Vorschein. Diese Vorgehensweisen werden unter anderem in Lakos (1996) beschrieben.

Schlussendlich besteht jedoch die einfachste Methode der Bewältigung von Komplexität darin, sie zu vermeiden. Software sollte so einfach wie möglich gehalten werden.

6.2.2 Erweiterungen

Im Folgenden werden einige Möglichkeiten zur Weiterentwicklung aufgezeigt. Die Qualität der gewonnenen Modelle könnte durch Annahmen über die Form des Gebäudes verbessert werden. Beispielsweise könnten Linien, die beinahe parallel oder orthogonal zur Gebäudehauptachse verlaufen, auf diese Winkel gezwungen werden. Analog könnten fast rechte Winkel auf exakt 90 Grad korrigiert werden. Dies würde unästhetische Schieflagen der Wände vermeiden.

Texturierung der Gebäudewände (das Drapieren eines Farbbildes über die Polygone) würde sehr viel zum Realismus der Szene beitragen. Hierzu existieren bereits einige Verfahren.

Neben den erwähnten Optimierungsmöglichkeiten könnte man den Algorithmus auch parallelisieren. Dies ist insbesondere sinnvoll, weil Fortschritte in der Hardwareentwicklung inzwischen eher der Simultanverarbeitung als der Rechengeschwindigkeit zugute kommen. Der größte Optimierungsbedarf besteht bei den frühen Verfahrensschritten, in denen alle Punkte verarbeitet werden müssen. Da diese Operationen bis auf gemeinsam gelesene Speicherstellen unabhängig sind, ist Parallelisierung einfach möglich.

Anhang A

Geometrie

A.1 Ausgleichsebene

In dieser Arbeit sind oftmals Ausgleichsebenen (auch Regressionsebenen genannt) von Interesse. Sie dienen der Berechnung der Tangentialebene eines Punktes sowie der Approximation der Dachflächen. Wir präzisieren die Bedeutung der Ausgleichsebene und leiten dessen Berechnung her.

A.1.1 Definition

Eine Ausgleichsebene ist diejenige, die eine Punktmenge möglichst genau approximiert. Genauer gesagt ist die mittlere quadratische Distanz der Punkte zur Ebene zu minimieren.

A.1.2 Formulierung als Eigenwertproblem

Nach Pearson (1901) entspricht der Eigenvektor zum kleinsten Eigenwert der Kovarianzmatrix nichts anderem als dem Normalenvektor einer Punktmenge.

Die Plausibilität dessen wird kurz erläutert. Eine Eigenschaft dieses Normalenvektors ist, dass er möglichst unkorreliert zur Ausdehnung der Punkte sein soll. Dies ist gerade beim Eigenvektor des kleinsten Eigenwerts der Fall (siehe Principal Component Analysis).

Ferner liegt es nahe, dass die Ausgleichsebene orthogonal zur Hauptdrehachse des Körpers liegen sollte. Diese wird über den Trägheitstensor erhalten. Der Zusammenhang zur Kovarianzmatrix ist insofern hergestellt, dass sich beide Tensoren zur Einheitsmatrix aufsummieren. Gross und Thoenessen (2006)

Die Kovarianzmatrix einer Punktmenge M definiert sich als

$$\sum_{p \in M} (p - c) \otimes (p - c) \tag{A.1}$$

$\otimes$ stellt hierbei das Tensorprodukt dar. Seien $a = (a_i), b = (b_j)$ Tensoren vom Rang 1 (Vektoren); das Ergebnis des Produkts $a \otimes b$ ist ein Tensor vom Rang 2 (Matrix). Dessen Elemente sind $M_{ij} = a_i \cdot b_j$.

A.1.3 Lösung des Eigenwertproblems

Zur Bestimmung der Eigenvektoren ist Singular Value Decomposition Klema und Laub (1980) ein gängiger Lösungsweg. Da jedoch keine Erfahrung mit BLAS oder ähnlichen Funktionssammlungen vorliegt, wurde nach einem einfacheren Weg gesucht. Das Eigenwertproblem kann insbesondere bei der symmetrischen 3×3 Kovarianzmatrix relativ einfach analytisch gelöst werden. Eigenwerte erfüllen die Gleichung

$$\det(M - \lambda E) = 0 \tag{A.2}$$

Wir definieren:

$$a = M_{11}, b = M_{12}, c = M_{13}, d = M_{22}, e = M_{23}, f = M_{33}$$

Das charakterische Polynom ist also nach der Regel von Sarrus:

$$-(a+d+f)\lambda^3 - (b^2+c^2-ad+e^2-af-df)\lambda^2 + (c^2d-2bce+ae^2+b^2f-adf)\lambda = 0 \tag{A.3}$$

Die Werte von λ, die diese Gleichung erfüllen, sind die gesuchten Eigenwerte.

Lösung kubischer Gleichungen

Wir möchten diese Gleichung möglichst in geschlossener Form lösen. Dies ist analog zur bekannten quadratischen Formel möglich. Der entsprechende Ansatz wird Cardano im Jahre 1545 zugeschrieben. Die Herleitung ist etwas komplex, sodass auf Nickalls (1993) verwiesen wird.

NB: Cardanos Formulierung erfordert das Berechnen der dritten Wurzel einer komplexen Zahl[1]. Da die komplexwertige Arithmetik in C++ recht beschränkt ist, wurde die Lösung mit trigonometrischer Substitution vereinfacht.

Es werden nun die Nullstellenterme angegeben. Das Polynom liegt in allgemeiner Form vor:

$$f(x) = x^3 + a_2x^2 + a_1x^1 + a_0 \tag{A.4}$$

Seien

$$x_N = -\frac{a_2}{3}, y_N = f(x_N), \delta = \frac{\sqrt{a_2^2 - 3a_1}}{3}, h = 2\delta^2, \theta = \frac{\arccos(\frac{-y_N}{h})}{3}$$

Dann ist

$$D := y_N^2 - h^2 \tag{A.5}$$

die sogenannte Diskriminante, welche die Art der Nullstellen beschreibt. Ist sie negativ, gibt es drei unterschiedliche reelle Nullstellen, nämlich:

$$x_0^{(n)} = x_N + 2\delta\cos(\frac{2n\pi}{3} + \theta) \quad (n \in \{0..2\}) \tag{A.6}$$

[1] dies, obwohl alle Lösungen im Reellen liegen mögen. Das Phänomen nennt sich Casus Irreducibilis.

Falls positiv, existiert genau eine reelle Nullstelle:

$$x_0^{(0)} = x_N + \sqrt[3]{\frac{-y_N + \sqrt{D}}{2}} + \sqrt[3]{\frac{-y_N - \sqrt{D}}{2}} \tag{A.7}$$

Bei Null gibt es eine oder zwei Nullstellen, und zwar:

$$x_0^{(0)} = x_N + \delta, \quad x_0^{(1)} = x_N - 2\delta \tag{A.8}$$

Damit können allgemeine kubische Gleichungen gelöst werden. Insbesondere haben wir nun die Eigenwerte als Nullstellen des charakteristischen Polynoms bestimmt.

Bestimmung der Eigenvektoren

Aus einer Matrix und deren Eigenwerten können die zugehörigen Eigenvektoren bestimmt werden. Dazu würde man sie ins charakteristische Polynom einsetzen und das resultierende Gleichungssystem lösen. Dieser Ansatz ist jedoch numerisch ungeschickt, da die Berechnung auf der inexakten Lösung von (A.2) aufbaut. Es ist besser, die Eigenvektoren direkt aus der ursprünglichen Matrix zu gewinnen. Im Detail wird hier der linke Nullraum der Matrix

$$M - \lambda E$$

bestimmt; ihre Basisvektoren sind die Eigenvektoren. Dies geschieht per Gauß-Jordan Elimimation [ACM270].

A.1.4 Konstruktion der Ebene

Die Ebene wird in der Hesseschen Normalform aufgestellt. Wie anfangs gezeigt, entspricht der Normalenvektor dem bereits bestimmten Eigenvektor. Es bleibt noch, den Achsenabstand zu berechnen. Dieser ist dadurch festgelegt, dass die Ebene durch c verläuft, dem zentralen Punkt[2] der Punktmenge. Die Ebene ist also gegeben durch:

$$\langle x, EV_0 \rangle + \langle c, EV_0 \rangle = 0 \tag{A.9}$$

Der kleinste Eigenwert hat eine besondere Bedeutung. Sein Wert ist ein Maß dafür, wie gut alle Punkte zur Ausgleichsebene passen und entspricht somit näherungsweise dem Krümmungsgrad der Punktmenge. Rabbani (2006) Diese Eigenschaft wird im Abschnitt A.3.3 beim Region Growing sowie bei der Klassifikation der Punkte benutzt.

A.2 Punktnachbarschaft

Im Gegensatz zur Bilddarstellung bieten Punktwolken keine einfache Möglichkeit zur Bestimmung der Nachbarschaft eines Punktes. Diese Operation wird oft benötigt und ist durchaus zeitkritisch, weil viele Datenpunkte zu betrachten sind.

[2] der Punkt, der den gemittelten Koordinaten der Punkte am nächsten liegt.

In der Literatur spricht man hierbei vom k-nächsten-Nachbar Problem. Die Aufgabenstellung wird mit einer naiven Lösung erläutert: Man sortiert alle Punkte nach der euklidischen Entfernung zum Referenzpunkt. Die ersten k Punkte mit der geringsten Entfernung sind das gewünschte Ergebnis. Diese Realisierung hat bei N Punkten eine zeitliche Komplexität in $O(N \log N)$. Es ist jedoch viel zu aufwändig, jeden Punkt zu betrachten. Im Folgenden werden effizientere Algorithmen vorgestellt.

Eine Verbesserung wäre mit einer Quadtree[3] gegeben. Der Raum würde rekursiv in vier quadratische Unterräume aufgeteilt werden. Für die Nachbarschaftssuche brauchen danach lediglich benachbarte Zellen betrachtet werden. Nachteil ist jedoch der exponentiell anwachsende Speicherbedarf. ALS-Daten enthalten Gebiete mit geringer Messdichte. Diese entstehen auf Grund der unterschiedlichen Auflösung in Flugrichtung bedingt durch das Schwenkmuster des Scanners. Im Quadtree würden in solchen Gebieten aufgrund der starren Raumaufteilung viele Unterräume unnötig entstehen, was Speicher verschwendet. Indyk

Man kann auch den Raum nur bei Bedarf und an beliebiger Stelle aufteilen; das nennt sich BSP (Binary Space Partitioning). Somit wäre die besagte Speicherverschwendung eliminiert, allerdings eignet sich die Datenstruktur aufgrund der beliebigen Trenn-Hyperebenenorientierung nicht zur Nachbarschaftssuche.

Der KD-Tree ist eine ähnliche Methode, den Raum zu spalten; allerdings sind die Trenn-Hyperebenen achsenausgerichtet. Dadurch können die umgebenden Zellen wieder effizient gefunden werden. Jede dieser Zellen speichert einen Messpunkt und die Orientierung der Trennebene[4]. Der große Vorteil gegenüber Quadtrees ist der geringe Speicherverbrauch in $O(N)$. Nachbarschaften lassen sich sogar mit einer erwarteten Komplexität in $O(1)$ bestimmen! Bentley (1990) Ferner besteht das Problem der leeren Räume nicht mehr. Beim Aufbau des Baums wird als Trennebene diejenige gewählt, welche die meisten verbleibenden Punkte aufteilt. Als Implementierung wird die Bibliothek ANN Mount (2006) verwendet.

A.3 Segmentierer

Die Ziele und Designentscheidungen des Segmentierers werden vorgestellt und das Verfahren wird detailliert beschrieben.

A.3.1 Ziele

Ein Segmentierer teilt Punkte in Regionen ein. Es sollen hiermit sowohl Dächer erkannt als auch Dachflächen getrennt werden können. Die Kriterien und Parameter bei der Einteilung müssen also anpassbar sein.

[3] Strenggenommen handelt es sich hierbei um 3D-Daten, also müsste man von einem Octtree sprechen. Allerdings sind wegen der Beschaffenheiten von ALS-Daten kaum Punkte übereinander angeordnet. Die Daten werden als 2,5D bezeichnet. Deswegen kann man die dritte Dimension auslassen und einen Quadtree verwenden.

[4] Es genügt eine Angabe der Achsen, zum Beispiel 1 = xz-Ebene.

In beiden Anwendungsfällen wäre Untersegmentierung[5] schädlich. Es ist meist einfacher, ähnliche Regionen zu verschmelzen, als Regionen im Nachhinein aufzuspalten. Deshalb sollen Parameter eher 'scharf' gewählt sein und tendenziell eine Übersegmentierung bevorzugen.

Bezüglich der Laufzeit gibt es keine festen Vorgaben. Eine schnelle Ausführung kommt der Testbarkeit immer zugute. Allerdings geht es in dieser Arbeit nicht um eine echtzeitfähige Implementierung.

Vielmehr ist Genauigkeit von Nutzen. Die Lokalisierung der Eckpunkte des Dachs ist entscheidend für ihre akkurate Modellierung. Diese gehen letztendlich aus den Dachflächen hervor, die anfangs mit diesem Segmentierer ermittelt wurden. Allerdings wird die Lage der Flächen in einem gesonderten Schritt nochmals optimiert, sodass die Genauigkeit hier nicht alleinentscheidend ist.

Flexibilität ist also das wichtigste Ziel, gefolgt von Genauigkeit.

A.3.2 Bestehende Segmentierer

Es wurde erwägt, bereits existierende Segmentierer zu verwenden. SOMBRERO Kohlhepp (1994) liegt nicht in brauchbarer Form vor: Es ist in Occam implementiert und auf Transputer lauffähig. Dieses System ist auf Echtzeitverhalten optimiert, weil es von autonomen Robotern verwendet werden soll.

Das Toolkit ITK Yoo (2004) bietet viele Segmentierungsmöglichkeiten an. Das Paket ist jedoch so komplex und umfassend, dass man es für wirtschaftlicher gehalten hat, ein einfaches Verfahren neu zu realisieren.

Weitere wiederverwendbare und öffentlich erhältliche Segmentierer wurden leider nicht gefunden.

Eine Neuimplementierung hat den Vorteil, dass die Segmentierung genau an die Beschaffenheiten der ALS-Daten angepasst werden kann. Tóvári und Pfeifer (2005) hat einen Segmentierer für terrestrische Daten verwendet. Es wird dort erwähnt, dass ein nur auf Normalenvektoreneinheitlichkeit bedachter Segmentierer interessant wäre. Ein solcher wird nun vorgestellt.

A.3.3 Algorithmus

Das gewählte Verfahren basiert auf Rabbani (2006). Es handelt sich hierbei um einen region-growing Segmentierer. Dessen Hauptvorteile sind die einfache Realisierung und die schnelle Laufzeit. Ein erster Ansatz wird mit Algorithmus 3 angegeben.

Die erste Schleife läuft bereits in der Vorverarbeitungsphase (siehe Abschnitt 4.1). Es kann vorausgesetzt werden, dass für alle Punkte der Normalenvektor und der Grad der lokalen Krümmung berechnet wurden.

[5] Untersegmentierung bedeutet, dass zu wenig Regionen entstanden sind. Das heißt auch, dass Punkte zu Regionen zusammengefasst wurden, die eigentlich gar nicht zusammen passen.

Algorithmus 3 : Segmentierung.

Eingabe : punkte
Ausgabe : regionen

```
   // Vorverarbeitung:
 1 für jedes p ∈ punkte tue
 2     nachbarschaft ← k_nearest(p, K)
 3     c ← zentralerPunkt(nachbarschaft)
 4     e ← ausgleichsebeneBerechnen(nachbarschaft)
 5     c.normalenvektor, c.krümmung ← e.normalenvektor, e.krümmung
 6     wenn p noch kein Normalenvektor zugewiesen dann
 7         p.normalenvektor, p.krümmung ← e.normalenvektor, e.krümmung
 8 regionen ← ∅
 9 solange ∃ p | nichtSegmentiert(p) tue
10     region ← ∅
11     ursprung ← p | p.krümmung = min(krümmung)
12     nachbarschaft ← k_nearest(p, K)
13     für jedes n ∈ nachbarschaft tue
14         wenn kompatibel(n, ursprung) dann
15             region ← region ∪ {n}
16     wenn region gültig dann
17         regionen ← regionen ∪ {region}
18     sonst
19         region auflösen
20 zurück regionen
```

Segment-Datenstruktur

Segmentierte Punkte werden in einer Region-Datenstruktur gesammelt. Diese besteht aus:

- einer Liste der enthaltenen Punkte;
- eine Ausgleichsebene in Hesse-Normalform;
- einem achsenausgerichteten umschließenden Quader.

Die folgenden Operationen werden benötigt:

- Neue Region anlegen;
- Punkt in Region aufnehmen;
- Entscheiden, ob Punkt schon einer Region zugeteilt ist;
- n-ten Punkt einer Region ausgeben.

Diese lassen sich alle in $O(1)$ realisieren. Als einzige Besonderheit muss in jedem Point vermerkt werden, ob er schon zugeteilt wurde. Außerdem führt jede Region eine Liste der enthaltenen Punkte. Diese beinhaltet einen Zähler und einen Zeiger auf einen gemeinsamen Datenspeicher. Dort werden Punktindizes abgelegt; es ist bereits Platz für alle Punkte reserviert. Zum Hinzufügen muss also lediglich der Punktindex an der aktuellen Position gespeichert werden. Beim Anlegen einer neuen Region wird deren Zeiger auf das nächste Feld in der Reihung gesetzt.

Notabene: Hierbei wird angenommen, dass Regionen in einem Schub gefüllt werden, da die Punkte ansonsten nicht zusammenhängend im gemeinsam genutzen Datenspeicher liegen würden. Da der Algorithmus jedoch Regionen im Greedy-Verfahren füllt und nicht mehr weiter bearbeitet, ist die Voraussetzung gegeben.

Ähnlichkeitskriterium

Für jeden Messpunkt wären folgende Merkmale denkbar, anhand derer gruppiert werden soll:

Normalenvektor der Tangentialebene Sind diese näherungsweise gleich, entstehen stückweise glatte Regionen.

Gemessene Intensität Dies kann bei der Klassifizierung helfen – Dächer haben andere Werte als der Boden.

Höhenunterschiede Damit können womöglich Gebäude von benachbarten und hereinragenden Bäumen getrennt werden.

Verhältnis der Eigenwerte Diese weisen auf Linien/Ecken hin. Gross und Thoenessen (2006)

Welche hiervon können verwendet werden? Laut Aufgabenstellung soll hier ohne Intensitätsdaten ausgekommen werden. Um sicherzustellen, dass das Verfahren auch ohne diese funktioniert, werden diese Daten gar nicht verwendet. Das zuletzt erwähnte Merkmal ist interessant, sprengt jedoch den Rahmen dieser Arbeit. Es wird festgestellt, dass die Gebäude allein schon anhand der Oberflächenglätte zuverlässig segmentiert werden.

Weitere künstliche Einschränkungen würden bei manchen Datensätzen eventuell nicht wie gewollt funktionieren. Es ist vermutlich besser, bei einfachen Ähnlichkeitskriterien zu bleiben.

Ein Kandidat-Punkt darf also genau dann ins Segment, wenn die Normalenvektoren des Punktes und des Region-Ursprungspunktes ähnlich sind. Der Winkelunterschied soll weniger als einen Schwellwert τ betragen. Hierzu kann das Skalarprodukt im dreidimensionalen Euklidischen Vektorraum verwendet werden. Er ist definiert als

$$\langle V_1, V_2 \rangle = |V_1||V_2| \cos(\theta) \tag{A.10}$$

Man beachte, dass die Orientierung der Normalenvektoren – ob sie nach außen zeigen oder nicht – zunächst unbekannt ist. Deswegen ist das Vorzeichen des Skalarprodukts unbestimmt. Die Richtungen werden zwar später vereinheitlicht, was aber in diesem Verarbeitungsschritt noch nicht praktikabel ist. Eine einfache Möglichkeit wäre, Vektoren mit $Z > 0$ als nach außen gerichtet zu definieren. Dies versagt allerdings bei horizontalen Normalenvektoren, die tatsächlich in 2,5D ALS-Daten bei zufällig mitgemessenen Wandpunkten auftreten können. Robuste Lösungen wären möglich – (Hoppe u. a., 1992, §3.3) basiert auf Graphenoptimierung. Dieser Aufwand erscheint jedoch nicht sinnvoll.

Die obige Problematik wird durch Betragsbildung umgangen. Der Winkel θ zwischen zwei Vektoren V_1, V_2 lässt sich also berechnen als

$$\theta = \arccos(|\langle V_1, V_2 \rangle|) \tag{A.11}$$

Die betrachteten Vektoren sind die Normalenvektoren N_1, N_2 der Tangentialebenen. Somit ist das Kriterium

$$\arccos(|\langle N_1, N_2 \rangle|) \leq \tau \tag{A.12}$$

Der Schwellwert τ wird mit 20 Grad recht hoch gewählt. Dies soll bewirken, dass auch Eckpunkte möglichst zum Segment gerechnet werden. Tóvári und Pfeifer (2005)

Eine Optimierung ist möglich: Beide Seiten der Ungleichung können mit `cos` transformiert werden. Dies vermeidet die Berechnung der `acos` Funktion und beschleunigt die Segmentierung um etwa 6 %.

Überwindung der Nachbarschaftsgrenze

Offensichtlich können mit diesem einfachen Verfahren keine Segmente erzeugt werden, die über eine Nachbarschaftsgrenze hinweg verlaufen. Man könnte die Nachbarschaft so groß wählen, dass alle zu erwartenden Gebäude damit abgedeckt wären. Dies ist jedoch datenabhängig, was vermieden werden soll.

Der Ausweg ist, mehrere Ursprungspunkte für jedes Segment zu erlauben. Während über die Nachbarschaftspunkte iteriert wird, prüft man, ob die lokale Oberflächenkrümmung unterhalb eines Schwellwerts liegt; falls ja, wird er als neuer Ursprungspunkt vorgemerkt. Die geschieht in einer Priority Queue – der Punkt mit der glattesten Umgebung wird zuerst behandelt. So erhofft man sich möglichst große Segmente. Würde

man nämlich in einem etwas verrauschten (und somit nicht-glatten) Gebiet ein Segment wachsen lassen, würden eventuell keine Punkte akzeptiert.

Dieser wichtige Parameter bestimmt den Grad der Segmentierung. Ist er flexibel gewählt, kommen viele Ursprungspunkte hinzu. Da die neuen Punkte nur ähnlich zum Ursprung sein müssen (und nicht zum bisherigen Segment), werden somit viel mehr Punkte zum Segment aufgenommen. Dies führt womöglich zur Untersegmentierung, wo auch nicht ganz ähnliche Punkte fälschlicherweise zu einer Region zusammengefasst werden. Umgekehrt kann ein zu scharf gewählter Schwellwert dazu führen, dass zu viele Regionen entstehen (Übersegmentierung). Um einigermaßen datenunabhängig zu bleiben, wird der Schwellwert als n-ter perzentil berechnet. Die Wahl ist allerdings nicht so kritisch, wie man vermutet. Im Bereich 80..90 % werden gute Segmentierungsergebnisse beobachtet.

Entfernung unechter Segmente

Manche Segmente bestehen nur aus wenigen Punkten. Solche unechten Segmente werden entfernt, weil sie eine visuelle Inspektion der Ergebnisse erschweren. Davor werden ihre Punkte als nicht-besucht markiert, damit weitere Segmente eine Chance haben, diese Punkte einzubinden.

Wie anfangs erwähnt, soll das Kriterium flexibel sein, um beide Anwendungsfälle abzudecken. Die Mindestanzahl der Punkte ist variabel. Zusätzlich kann eine untere und obere Grenze der lokalen Oberflächenkrümmung angegeben werden. Dies ist bei der Klassifikation nützlich - Segmente, die vermutlich nicht ein Gebäude(teil) darstellen, können auch entfernt werden.

A.4 Hough-Transformation

A.4.1 Einführung

Die Hough-Transformation ist eine Technik, mit der beliebige Strukturen (beispielsweise Linien) in Bildern erkannt und extrahiert werden können. Man betrachte Strukturen als frei transformierbare (Rotation, Translation, Skalierung) Instanzen einer geometrischen Grundform. Es werden solche Instanzen gesucht, die möglichst viele Bildpunkte abdecken. Per Definition entsprechen diese den ‘sinnvollen’ Strukturen im Bild.

Die Lage und Beschaffenheit der Struktur-Instanz werden parametrisch dargestellt: Beispielsweise sind Kreise über ihren Mittelpunkt und Radius eindeutig definiert. Die Parameter sind typischerweise wertekontinuierlich. Sie müssen also quantisiert werden, um die Menge der zu überprüfenden Struktur-Instanzen endlich zu halten. Dazu wird der Wertebereich in gleich lange, durchnummerierte Intervalle aufgeteilt. Für jedes Tupel der so diskretisierten Parameter ist dann zu zählen, durch wieviele Punkte die entsprechende Struktur verläuft.

Um eine effiziente Berechnung zu ermöglichen, wird umgekehrt verfahren. Für jeden Punkt stellt man alle möglichen Strukturinstanzen auf, die diesen Punkt enthalten würden. Der Punkt leistet seinen Beitrag zur Strukturenerkennung, indem er eine ‘Stim-

me' für jede passende Instanz abgibt. Diese Stimmen werden in einer N-dimensionalen (entsprechend der Anzahl der Parameter) Matrix von Akkumulatoren summiert. Die Indizes in der Matrix bestimmen sich jeweils aus der Intervall-Nummer der Parameter, die die Struktur-Instanz definieren.

Es verbleibt noch, den Peak (Maximum) der Akkumulatorwerte zu bestimmen. Diese Stelle weist auf die 'stärkste' Struktur im Bild hin, weil sie die größte Anhäufung an Stimmen erhielt. Die Parameter der Struktur-Instanz sind durch die Indizes der Akkumulatorzelle gegeben. Transformiert man diese zurück in den Bildraum, so erhält man eine Approximation der Struktur. Diese ist aufgrund der Quantisierung nicht exakt, aber eine gute Annäherung.

Bisher hat man nur die stärkste Struktur im Bild extrahiert. In der Akkumulatormatrix sind jedoch Informationen über alle Strukturen enthalten. Löscht man darin den Beitrag der vorangegangenen Instanz, kann man wieder das Maximum der verbleibenden Stimmen ermitteln, und somit die nächststärkere Struktur erhalten. Es stellt sich die Frage, wie oft iteriert werden soll. Die Anzahl der Strukturen ist meist nicht von vornherein bekannt. Da Rauschen und zufällig angeordnete Punkte auch unechte Strukturen entstehen lassen, kann man so lange fortfahren bis die extrahierten Strukturen nur noch schwach ausgeprägt sind.

Damit sind alle sinnvollen Strukturen im Bild erkannt und extrahiert.

A.4.2 Vorangegangene Arbeit

Transformation

Der Ursprung der Hough-Transformation ist die Idee, Punkte auf Strukturen abzubilden und dann nach Anhäufungen von diesen zu suchen. Dies hat Hough 1960 zum Patent Hough (1962) angemeldet; als Anwendung war angegeben, Bewegungsmuster von Teilchen automatisch zu analysieren. In der dort vorgeschlagenen Fassung sind Linien die gesuchten Strukturen. Punkte werden *linear* einer Geraden zugeordnet, welche über Steigung und Achsenabschnitt definiert ist. Allerdings ist der Parameterraum unbeschränkt, weil die Steigung gegen ∞ gehen kann.

Duda und Hart Duda und Hart (1972) schlugen 1972 eine andere Parametrisierung vor, die man heute bevorzugt verwendet. Punkte werden nichtlinear in Polarkoordinaten (θ, ρ) abgebildet. Diese Darstellung ist vorteilhaft, weil die Parameter in einem beschränkten Intervall liegen. Sie wird in Abschnitt A.4.3 detaillierter vorgestellt.

Mit dem vielzitierten Artikel von Ballard Ballard (1981) wurde die Hough-Transformation in der Bildverarbeitung popularisiert. Er generalisierte das Verfahren: Mit einer geeigneten Transformationsfunktion können beliebige Strukturen extrahiert werden. Seitdem gibt es eine Vielzahl an Publikationen.

Laufzeit

Eine große Bestrebung ist, die Laufzeitkomplexität der Hough-Transformation zu verringern. O'Gorman and Clowes O'Gorman und Clowes (1973) erkannten, dass Linienstrukturen oft orthogonal zum Gradient der Punkte liegen. Somit brauchen nur θ-Werte

betrachtet und zum Akkumulator gerechnet werden, die ähnlich zur Richtung des Gradienten sind. Neben der beträchtlichen Zeitersparnis bewirkt dies auch eine Reduktion unnützer Stimmen im Akkumulator. Dadurch sind Peaks etwas sauberer zu bestimmen.

Quantisierung

Niblack und Petkovic Niblack und Petkovic (1988) untersuchten die Parameterquantisierung und haben die optimale Intervall-Größe abgeleitet. Sie wird aus der erwarteten Verteilung der Butterflies[6] berechnet. Ferner wurde vorgeschlagen, Stimmen in ρ-Richtung auf zwei benachbarte Akkumulatoren zu verteilen. Dies geschieht anteilig entsprechend dem Abstand des kontinuierlichen ρ von den diskreten Zellgrenzen. Damit sind Quantisierungsfehler etwas reduziert.

Lokalisierung der Peaks

Viele weitere Publikationen beschäftigen sich mit der Suche des genauen Peaks:

Davies Davies (1992) führt die Hough-Transformation zunächst in niedriger Auflösung durch. Danach kommt eine zweite Transformation in etwa fünffacher Auflösung. Dabei werden allerdings nur Akkumulatoren berücksichtigt, die in der Nähe des vorher ermittelten Peaks liegen. Dadurch wird der Speicherbedarf reduziert, was heute allerdings nicht mehr von größter Bedeutung ist. Schließlich wird der Peak mit einer Median-basierten Methode berechnet, welches Genauigkeit jenseits der diskreten Parameterschritte erlaubt.

Leavers und Boyce Leavers und Boyce (1987) verwenden ein sogenanntes 'Butterfly' Filter, welches den Akkumulator in θ-Richtung glättet und gleichzeitig Anhäufungen von Stimmen ausdünnt. Dies soll eine genauere Lokalisierung erreichen. Leider ist nur die Zusammenfassung dieses Artikels erhältlich.

Furukawa und Shinagawa Furukawa und Shinagawa (2003) versuchen den wahren Peak durch Benutzung aller beteiligten Butterfly-Punkte noch genauer zu ermitteln. Ein Modell zur Verteilung dieser Punkte wird aufgestellt und durch Kreuzkorrelation im Akkumulator angewandt. Der Vorteil dieser Methode ist, dass mehr Punkte mit in die Berechnung eingezogen werden als nur die in der Nachbarschaft des Peaks.

Niblack und Petkovic Niblack und Petkovic (1988) erwähnen ein Filter, welches den genauen Peak über Schwerpunktbildung innerhalb eines Fensters berechnet. Die Fenstergröße ist adaptiv bestimmt, sodass die Nachbarschaft des Peaks enthalten ist, aber Nebenmaxima ausgeschlossen bleiben. Durch diese Interpolation können ausgedehnte Peaks weitaus genauer bestimmt werden als bei der naiven Suche des maximalen Akkumulators.

Nachbearbeitung

Auch zum Thema der Verbesserung der gefundenen Peaks gibt es Arbeiten:

Rau und Chen Rau und Chen (2003) eliminieren unechte Strukturen durch eine Hauptachsenanalyse. Zunächst werden Peaks entsprechend der Gesamtstimmzahl aller Linien mit der gleichen Richtung sortiert. Es werden nur die zwei häufigsten Richtungen

[6] Sinuskurven im Akkumulator, die aus der Transformation einer Linie hervorgehen. Sie werden wegen ihrer Ähnlichkeit zur Schmetterlingsform 'Butterfly' genannt.

akzeptiert, die zusätzlich noch orthogonal zueinander liegen müssen. Damit werden ungewollte Strukturen wirkungsvoll ignoriert. Allerdings können mit diesem Verfahren nur Strukturen erkannt werden, die aus paarweise orthogonalen Segmenten bestehen.

Risse Risse (1989) entwickelte ein Verfahren, um im Akkumulator den Einfluss von bereits extrahierten Strukturen zu entfernen. Die Punkte, die mit ihrer Stimme zur 'Wahl' der Struktur beitrugen, werden erneut in den Parameterraum transformiert. An diesen Stellen werden die Akkumulatoren jedoch verringert. Damit haben diese Punkte keine Auswirkung mehr auf die restlichen zu extrahierenden Strukturen.

Akhtar und Atiquzzaman Atiquzzaman und Akhtar (1999) erklären, wie man die Länge eines extrahierten Geradensegments ermittelten kann. Alle Punkte einer Linie gehören zum gleichen θ-Parameter; nur ρ variiert. Vom Peak ausgehend werden in ρ-Richtung benachbarte Akkumulatoren ermittelt, denen mindestens ein Punkt zugeteilt wurde. Die ρ-Werte der Endpunkte dieser Akkumulatorreihe werden zurück in den Bildraum projiziert und ergeben dann die Länge der Linie.

A.4.3 Theorie

Die Hough-Transformation wurde im vorausgehenden Abschnitt in ihrer allgemeinen Form vorgestellt. Da für die vorliegende Anwendung nur Linien extrahiert werden sollen, wird die Transformation auf Linienstrukturen im $\mathbb{R}^2$ spezialisiert.

Die Eingangspunkte sind als (x, y) Koordinatenpaare gegeben. Wie oben erwähnt, sind Polarkoordinaten (θ, ρ) die heute gängige Parametrisierung einer Linie. Betrachtet man die (bei zweidimensionalen Daten eindeutig bestimmte) Orthogonale der Gerade, stellt θ den Winkel zwischen ihr und der x-Achse dar. ρ gibt die Entfernung der Linie zum Ursprung an.

Es folgt eine Überlegung zur Steigerung der Genauigkeit. Bei den extrahierten Geraden möchte man den Winkel möglichst genau erfassen. Dies ist insbesondere bei großen Gebäuden kritisch, weil die Abweichung von der wahren Linie proportional zur Länge und $\tan(\Delta\theta)$ wächst. Die Genauigkeit von ρ ist zweitrangig, weil es nur eine konstante Verschiebung verursacht. Der Winkelfehler $\Delta\theta$ kann durch eine feinere Quantisierung verringert werden, allerdings auf Kosten des Speicherverbrauchs und der Rechenzeit. Zusätzlich werden ab einem gewissen Grad (siehe Abschnitt A.4.3) Peaks verschleiert, sodass die Genauigkeit wieder sinkt. Die Anzahl diskreter Winkelwerte kann also nicht unendlich gesteigert werden. Bleibt aber der Ausweg den Wertebereich von θ zu beschränken, sodass die Intervalllänge wiederum effektiv verkürzt wird. Dazu werden die Eingangskoordinaten (x, y) in (x_m, y_m) relativ zur Mitte der Punktwolke transformiert. Linien im zweiten und dritten Quadranten sind über negative ρ dargestellt. Damit ist der Wertebereich von θ auf $[0, \pi)$ beschränkt.

Bei der 'Transformation' eines Punktes (x_m, y_m) werden alle Linien aufgestellt, die durch diesen Punkt verlaufen. Dazu iteriert man über alle diskreten Werte von θ und berechnet das zugehörige ρ:

$$\rho = x_m \cos\theta + y_m \sin\theta \tag{A.13}$$

Diese Werte liegen im Intervall $(-R, R)$, wobei R den halben Durchmesser der Punktwolke beträgt.

Die oben ermittelten Werte sind noch zu quantisieren. Die Anzahl der diskreten Werte ist von großer Bedeutung für die Genauigkeit der extrahierten Linien. Für den Winkel θ kann die Anzahl der Intervalle entsprechend der gewünschten Genauigkeit gewählt werden. Etwa 500 diskrete Werte sind vorgesehen, damit bei Linienlänge $L = 30$ m der maximaler Fehler noch hinnehmbare $L \tan(2\Delta\theta) \approx 12$ cm beträgt. Daraus ergibt sich

$$\Delta\theta = \pi/500 \approx 6,13 \times 10^{-3} \tag{A.14}$$

Bei ρ impliziert eine feinere Auflösung grundsätzlich auch einen kleineren Quantisierungsfehler. Es ist aber noch zu berücksichtigen, dass Peaks über mehrere Intervalle aufgeteilt werden und die Genauigkeit bei zu kleinem $\Delta\rho$ deswegen wiederum sinkt. Nach van Veen und Groen (1981) beträgt die maximal zu erwartende Ausdehnung eines Peaks (in Anzahl der Intervalle in ρ-Richtung)

$$s_\rho = \left\lfloor \frac{L \sin(\Delta\theta/2) + 2b \cos(\Delta\theta/2)}{\Delta\rho} \right\rfloor + 2 \tag{A.15}$$

wobei L und $2b$ die Länge und Breite der erwarteten Linien darstellen und $\Delta\theta, \Delta\rho$ die Quantisierung angeben. Niblack und Petkovic (1988) empfiehlt $s_\rho \approx 2$ Intervalle. Dies soll noch hohe Genauigkeit für ρ erlauben, aber das Signal-Rauschverhältnis der Akkumulatoren ausreichend groß halten. Gl. A.15 kann approximiert werden als

$$s_\rho = \frac{L \sin(\Delta\theta/2) + 2b \cos(\Delta\theta/2)}{\Delta\rho} + 1 \tag{A.16}$$

Somit ergibt sich

$$\Delta\rho = \frac{L \sin(\Delta\theta/2) + 2b \cos(\Delta\theta/2)}{1} \approx 0,492 \tag{A.17}$$

A.4.4 Implementierung

Es sei erwähnt, warum noch eine Implementierung der sehr wohl häufig verwendeten Hough-Transformation nötig ist. Die verfügbaren Quellen sind alle – zum Teil stark – auf Bilder spezialisiert. In dieser Anwendung liegen die Daten aber als Punktwolke vor. Rasterung würde Information und Genauigkeit verschenken. Eine bestehende Implementierung müsste also auf Punktwolken umgerüstet werden, was recht aufwendig erschien. Außerdem berücksichtigen vorliegende Quellen[7] mitunter nur wenige der oben aufgeführten Erkenntnisse aus der Literatur. Es wurde deshalb eine neue Implementierung erstellt, die Punktwolken verarbeiten kann und hohe Genauigkeit anstrebt.

Oben wurden einige Neuerungen vorgestellt, die Verbesserungen gegenüber dem naiven Algorithmus bieten. Es wird kurz aufgezählt, welche Verwendung fanden.

[7] `http://prdownloads.sourceforge.net/itk/InsightToolkit-3.0.0.zip?download!itkHoughTransform2DLinesImageFilter.h`, `http://www.iamg.org/documents/oldftp/VOL24/v24-10-3.tar.Z!hough_transform.c`, `http://www.bmva.ac.uk/meetings/meetings/02/23Jan02/TRAYNER_hough_demo.zip!hough_xform.c`, `http://vision.eng.shu.ac.uk/jan/mimas/mimas-1.4.tar.bz2!mm_find_line.hh`.

- Quantisierungsschritte werden optimal gewählt.
- In ρ-Richtung werden benachbarte Akkumulatoren anteilig hochgezählt.
- Peaks werden wahlweise per Butterfly-Filter oder mittels Rectangle Filter lokalisiert.
- Extrahierte Linien werden durch Rücktransformation gänzlich aus dem Akkumulator entfernt.

Zur Implementierung sind einige Anmerkungen und Erläuterungen zu treffen.

Quantisierung

Quantisierung erfolgt durch Division mit der Intervalllänge und Runden mit der Vorschrift $\lfloor x \rfloor$. Diese Methode hat zwar einen leicht höheren maximalen Fehler ($\Delta\rho - \epsilon$ statt $\Delta\rho/2$), ist aber eine monotone Abbildung. Dadurch vereinfacht sich das Ermitteln der diskreten Rho-Grenzen (siehe unten).

Zur Steigerung der Genauigkeit werden die Akkumulatoren, die in ρ-Richtung beidseits des kontinuierlichen Wertes liegen, anteilig erhöht. Hierzu berechnet man für beide den Abstand zwischen ρ und den jeweiligen diskreten Grenzen. Diese werden als Gewichte für den Inkrementwert (üblicherweise 1) verwendet.

Notabene: Bei der Berechnung der Grenzen ist es unerlässlich, den Rundungsmodus exakt zu spezifizieren. Der IEEE-754 Gleitkomma-Rundungsmodus des Prozessors kann nicht vorausgesetzt werden, weil er systemspezifisch ist. Die obige Software-Rundungslösung funktioniert jedoch universell.

Effiziente Akkumulatormatrix

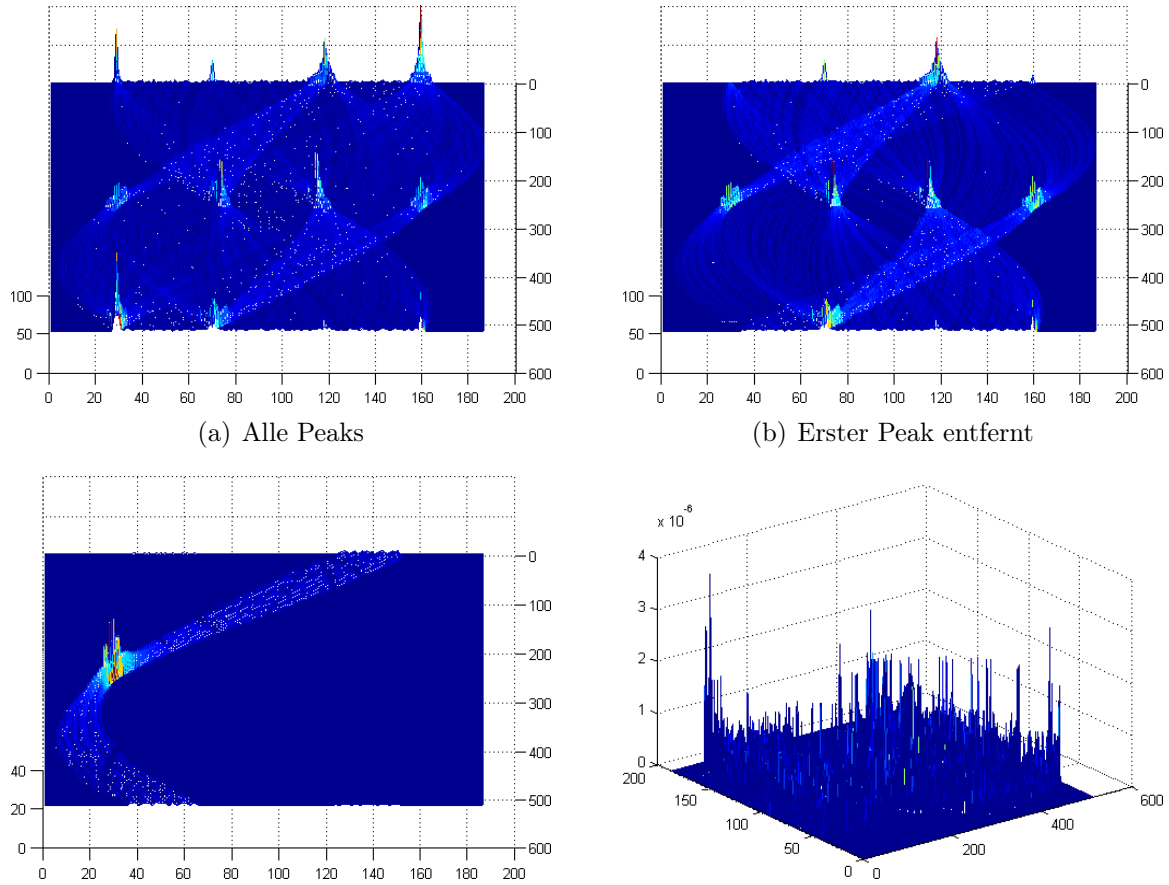
Für die Akkumulatoren stellt sich die Frage, nach welchem der Parameterindizes sie im Speicher angeordnet werden sollen. Die Anzahl der θ-Indizes ist konstant; bei ρ muss dies erst aus der Ausdehnung der Punktwolke errechnet werden (zunächst ist nur $\Delta\rho$ bekannt). Zur effizienten Adressierung bietet es sich an, nach θ anzuordnen und NUM_THETA_BINS als eine Zweierpotenz zu wählen. Damit reduziert sich ein Matrixzugriff auf eine Bitverschiebung um einen konstanten Wert, was im Allgemeinen weitaus schneller ist als die sonst erforderliche Multiplikation mit einer Variablen.

Entfernen der Peaks

Nach Risse (1989) kann der Einfluss einer bereits extrahierten Struktur durch eine Rücktransformation entfernt werden. Punkte, die für die Struktur stimmten, veranlassen nun das Herunterzählen der entsprechenden Akkumulatoren. Es kann allerdings sein, dass falsche Peaks noch vereinzelt übrig bleiben. Dies wäre zum Beispiel möglich, wenn nicht alle eigentlich dazugehörigen Punkte zur extrahierten Struktur passen. Um das auszuschließen, werden zusätzlich alle Akkumulatoren in einem Bereich um den Peak komplett gelöscht.

Der Erfolg dieser Methode wird in Abbildung A.1 ersichtlich. Oben links sind die Akkumulatoren nach der Transformation der ‘t’-Daten gezeigt; oben rechts wurde der größte Peak entfernt. Man beachte die Periodizität des Bildes an den oberen und unteren Rändern aufgrund der Definition des Winkels θ. Unten links ist der letzte Peak

sichtbar, wobei alle anderen Peaks sauber entfernt wurden. Am Ende verbleiben nur Reste der Größenordnung 10^{-6}.



(a) Alle Peaks

(b) Erster Peak entfernt

(c) Letzter Peak

(d) Restposten

Abbildung A.1: Hough-Akkumulatoren nach Transformation des ‘t’-Gebäuderandes und nach Entfernen der Peaks.

A.4.5 Ergebnisse

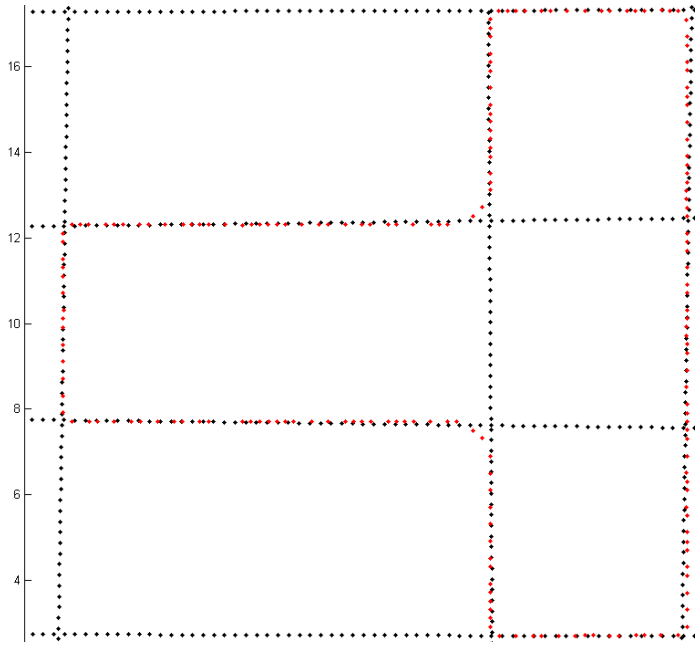
Die Genauigkeit der Geradenextraktion mittels Hough-Transformation wurde untersucht. Im ersten Test wurden 100 Punkte auf 20 m der x-Achse generiert. Die Positionierungs- und Winkelfehler lagen in der Größenordnung 10^{-7}. Unter den realistischeren Bedingungen eines zufälligen schiefen Winkels und einer Länge von 5 m wurde ein Winkelfehler von 0,08 Grad und ein Positionierungsfehler von etwa 3 mm festgestellt. Diese Werte liegen deutlich besser als gefordert.

Die Ergebnisse der Linienextraktion aus den Kantenpunkten der ‘t’ und ‘fom’ Datensätzen sind in Abbildung A.2 dargestellt. Die Linien sind als schwarze Punkte gezeichnet und der Gebäuderand rot dargestellt. Bei den synthetischen Daten sind keine Probleme feststellbar. Die realen Daten bereiten eher Schwierigkeiten. Wegen Aliasing der

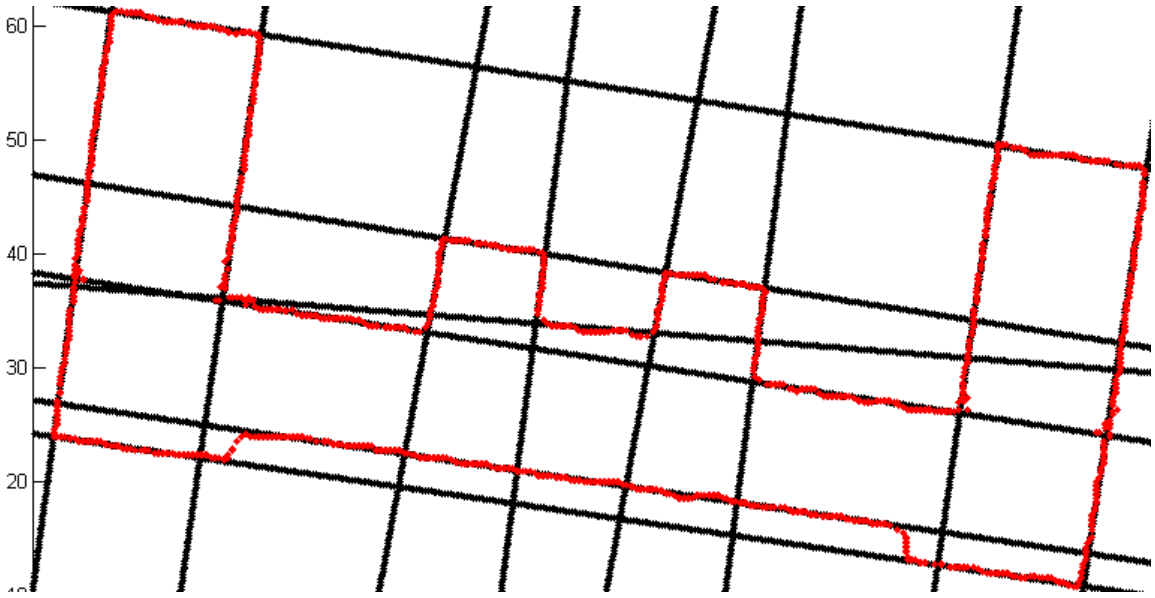
mittleren Punkte ist die ermittelte Linie dort etwas schief. Die kurzen Linien des unteren Gebäudevorsprungs können nicht extrahiert werden. Diese Problematik wurde in Abschnitt 4.4.5 besprochen.

A.4.6 Diskussion

Ein großer Vorteil der Hough-Transformation besteht darin, dass Linien auf Grund der Berücksichtigung vieler Punkte zuverlässig und genau extrahiert werden können. Desweiteren hat das Vorhandensein oder Fehlen von Rauschen keinen großen Einfluss. Es kann bis zu einem gewissen Grad gut toleriert werden, weil viele Punkte noch für die Strukturen stimmen. Dafür sind die Rechenzeit und der Speicherverbrauch beträchtlich. Bei kleiner Dimension des Parameterraums (zum Beispiel bei der Linienerkennung) ist dies allerdings noch unproblematisch. Ferner ist die richtige Quantisierung der Parameter entscheidend zum Erfolg. Gute Strategien sind aber hierfür bekannt, weil die Hough-Transformation sehr ausführlich untersucht wurde. Die Hough-Transformation ist also gut geeignet zur Linienextraktion.



(a) ‘t’ (synthetisch)



(b) ‘fom’

Abbildung A.2: Per Hough-Transformation extrahierten Linien (schwarz) sowie die ursprünglichen Randpunkte (rot).

Anhang B

Format der Dateien

Zwecks einer kompakten und eindeutigen Darstellung werden Textfelder hier in der printf-Syntax angegeben; siehe dazu die printf-Spezifikation. IEEE und The Open Group (2004)

Punkte können aus einer einfachen Szenenbeschreibungssprache erzeugt werden. Das Format lautet '`%s %f %f %f %f %f %f %f`' mit den Feldern Dachbezeichnung, x,y,z, Breite, Länge, Wandhöhe, Dachgeschosshöhe. Zulässige Dachbezeichnungen sind `flat`, `desk`, `gable`, `hipped`, `pyramid`; alle anderen Angaben sind in Meter. Jede gültige Zeile erzeugt ein Dach in der resultierenden Szene.

Punktkoordinaten werden im Format '`%f %f %f`' mit den Feldern x,y,z eingelesen. Pro gültige Zeile entsteht ein Punkt.

Der Effizienz halber werden die eingelesenen Punkte in binäre Darstellung konvertiert. Solche Dateien beginnen mit der FourCC[1] '`PTS5`' weil fünf Änderungen des Formats unternommen wurden. Es folgt eine 32-Bit Zahl der nachfolgenden Punkte in der Bytereihenfolge des schreibenden Systems[2]. Falls die Dateien auf ein System mit der anderen Bytereihenfolge übertragen werden, kann man den Konflikt erkennen und die Dateien als ungültig markieren; siehe dazu den nächsten Absatz. Notabene: Diese Zahl muss nicht zwingend gespeichert werden – sie kann aus der Dateigröße abgeleitet werden. Sie explizit anzugeben ermöglicht allerdings mehrere Datenströme in einer Datei unterzubringen.

Um sicherzustellen, dass die zwischengespeicherten Punkte auch aktuell sind und nicht auf veralteten Daten beruhen, werden die Abhängigkeiten geprüft. Für jede Datei, die Einfluss auf die verarbeiteten Punkte hat, speichert man einen Abbild ihres Zustands. Dies geschieht im Format '`%s %d %d %d`' mit den Feldern Dateiname, Größe, Zeitstempel (Sekunden seit 1970) und der Angabe, ob die Daten im little-endian System gespeichert sind. Falls beim Lesen festgestellt wird, dass der aktuelle Zustand von den

[1] Four Character Code - vier Textzeichen, die das Format der Datei und womöglich auch die Version angeben.

[2] CPU-Designer konnten sich nicht einigen, ob zum Beispiel die Zahl 0x1234 byteweise im Speicher als 0x34 0x12 abgelegt wird oder 0x12 0x34. Das erste System ist 'little-endian' und ist wegen der Verwendung in IA-32 sehr verbreitet. Beim Einlesen gespeicherter Zahlen im jeweils anderen Format müssen sie konvertiert werden.

gespeicherten Werten abweicht, darf die zwischengespeicherte Datei nicht verwendet werden.

Bei der Vorverarbeitung wird für jedes erkannte Gebäude ein Verzeichnis '`building%04d`' angelegt. Darin werden Punktwolken des Gebäudes, der Randpunkte sowie der Umgebungspunkte im obigen Binärformat abgelegt.

Die fertigen Gebäudepolygone werden auch dort gespeichert. Die Polygone sind als Liste der Eckpunkte angegeben, die im gleichen Format wie die ursprünglichen Punkte angegeben werden.

Literaturverzeichnis

[Alexandrescu 2001] Alexandrescu, A.: *Modern C++ Design*. Addison-Wesley, 2001 (C++ In Depth)

[Atiquzzaman und Akhtar 1999] Atiquzzaman, M. ; Akhtar, M.: *Determination Of End Points And Length Of A Straight Line Using The Hough Transform*. Januar 24 1999. – URL `http://citeseer.ist.psu.edu/152201.html;http://www.engr.udayton.edu/faculty/matiquzz/papers/was4-mva-cam.ps`

[Ballard 1981] Ballard, D.: Generalizing the Hough transform to detect arbitrary shapes. In: *Pattern Recognition* 13 (1981), S. 111–122

[Bellman 2003] Bellman, R.: *Dynamic Programming*. Courier Dover Publications, 2003. – ISBN 0-486-42809-5

[Bentley 1985] Bentley, J.: Programming Pearls: Tricks of the Trade. In: *Communications of the ACM* 28 (1985), Februar, Nr. 2, S. 138–141

[Bentley 1990] Bentley, J.: *K*-d trees for semidynamic point sets. (1990). – URL `http://portal.acm.org/citation.cfm?id=98564\&coll=portal\&dl=ACM`

[Briese u. a. 2002] Briese, C. ; Pfeifer, N. ; Dorninger, P.: Applications of the Robust Interpolation for DTM Determination. In: *Photogrammetric Computer Vision*, 2002, S. A: 55

[Canny 1986] Canny, J.: A computational approach to edge detection. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 8 (1986), Nr. 6, S. 679–698. – ISSN 0162-8828

[Coplien 1998] Coplien, J.: C++ Idioms / Bell Labs. URL `http://www.laputan.org/pub/sag/coplien-idioms.pdf`, 1998 (1). – Forschungsbericht. -

[Davies 1992] Davies, E.: Simple two-stage method for the accurate location of Hough transform peaks. In: *Computers and Digital Techniques* 139 (1992), Nr. 3, S. 242–248. – ISSN 1350-2387

[DeVore und Temlyakov 1996] DeVore, R. ; Temlyakov, V.: Some remarks on greedy algorithms. In: *Advances in Computational Mathematics* 5 (1996), Dezember, Nr. 1, S. 173–187. – ISSN 1572-9044

[Duda und Hart 1972] Duda, R. ; Hart, P.: Use of the Hough Transformation to Detect Lines and Curves in Pictures. In: *Commun. ACM* 15 (1972), Nr. 1, S. 11–15

[Elaksher und Bethel 2002] Elaksher, A. ; Bethel, J.: Reconstructing 3D Buildings from LIDAR Data. In: *Photogrammetric Computer Vision*, 2002, S. A: 102

[Elmqvist 2002] Elmqvist, M.: Ground Surface Estimation from Airborne Laser Scanner Data Using Active Shape Models. In: *Photogrammetric Computer Vision*, 2002, S. A: 114

[Fischler und Bolles 1981] Fischler, M. ; Bolles, R.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In: *Commun. ACM* 24 (1981), Nr. 6, S. 381–395

[Furukawa und Shinagawa 2003] Furukawa, Y. ; Shinagawa, Y.: Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space. In: *Computer Vision and Image Understanding* 92 (2003), Nr. 1, S. 1–25. – URL `http://dx.doi.org/10.1016/j.cviu.2003.07.002`

[Gross und Thoenessen 2006] Gross, H. ; Thoenessen, U.: Extraction of Lines from Laser Point Clouds. In: *Symposium of ISPRS Commission III PCV '06* Bd. XXXVI, 2006, S. ??–??. – ISSN 1682-1750

[Haala und Brenner 1999] Haala, N. ; Brenner, C.: Extraction of buildings and trees in urban environments. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (1999), Juli, Nr. 2-3, S. 130–137

[Hoppe u. a. 1992] Hoppe, H. ; DeRose, T. ; Duchamp, T. ; McDonald, J. ; Stuetzle, W.: Surface reconstruction from unorganized points. In: *Computer Graphics (SIGGRAPH '92 Proceedings)* Bd. 26, Juli 1992, S. 71–78

[Hough 1962] Hough, P.: *A Method and Means for Recognizing Complex Patterns.* U.S. Patent No. 3,069,654. 1962

[Hug 1997] Hug, C.: Extracting artificial surface objects from airborne laser scanner data. In: Gruen, A. (Hrsg.) ; Baltsavias, E. (Hrsg.) ; Henricsson, O. (Hrsg.): *Automatic Extraction of Man-Made Objects from Aerial and Space Images.* Basel : Birkhäuser Verlag, 1997, S. 203–212

[IEEE und The Open Group 2004] IEEE ; The Open Group: *Base Specifications Issue 6.* 2004. – URL `http://www.opengroup.org/onlinepubs/009695399/functions/printf.html`

[Indyk] Indyk, P.: *Algorithms for Nearest Neighbor Search.* – URL `http://dimacs.rutgers.edu/Workshops/MiningTutorial/pindyk-slides.ppt`

[Jiang und Bunke 1994] Jiang, X. ; Bunke, H.: Fast segmentation of range images into planar regions by scan line grouping. In: *Machine Vision and Applications 7* (1994), S. 115–122

[Klema und Laub 1980] Klema, V. ; Laub, A.: The Singular Value Decomposition: Its Computation and Some Applications. In: *IEEE Tran. Automat. Contr.* AC-25 (1980), S. 164–176

[Kohlhepp 1994] Kohlhepp, P.: Reconstruction and Recognition of Boundary Representations from Range Images in SOMBRERO. In: *ICIP (1)*, 1994, S. 496–500

[Lakos 1996] Lakos, J.: *Large Scale C++ Software Design.* Addison Wesley, 1996. – ISBN 0-201-63362-0

[Leavers und Boyce 1987] Leavers, V. ; Boyce, J.: The radon transform and its application to shape parameterization in machine vision. In: *Image and Vision Computing* 2 (1987), Nr. 5, S. 161–166

[Maas und Vosselman 1999] Maas, H. ; Vosselman, G.: Two algorithms for extracting building models from raw laser altimetry data. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (1999), Juli, Nr. 2-3, S. 153–163

[Mount 2006] Mount, D.: *ANN Programming Manual*. 2006. – URL `http://www.cs.umd.edu/~mount/ANN/Files/1.1.1/ANNmanual_1.1.1.pdf`

[Niblack und Petkovic 1988] Niblack, W. ; Petkovic, D.: On Improving the Accuracy of The Hough Transform: Theory, Simulations, and Experiments. In: *IEEE Computer Vision and Pattern Recognition or CVPR*, URL `http://ieeexplore.ieee.org/iel2/206/5032/00196293.pdf?tp=&isnumber=&arnumber=196293`, 1988, S. 574–579

[Nickalls 1993] Nickalls, R.: A new approach to solving the cubic - Cardano's solution Revealed. In: *The Mathematical Gazette* 77 (1993), S. 354–359. – URL `http://www.m-a.org.uk/docs/library/2059.pdf`

[Oda u. a. 2004] Oda, K. ; Takano, T. ; Doihara, T. ; Shibasaki, R.: *Automatic Building Extraction and 3-d City Modeling from Lidar Data Based on Hough Transformation*. XXth ISPRS Congress. Juli 2004. – URL `http://www.isprs.org/istanbul2004/comm3/papers/280.pdf`

[O'Gorman und Clowes 1973] O'Gorman, F. ; Clowes, M.: Finding Picture Edges through Collinearity of Feature Points. In: *IJCAI*, 1973, S. 543–555

[Pearson 1901] Pearson, K.: On Lines and Planes of Closest Fit to Systems of Points in Space. In: *Philosophical Magazine* (1901), S. 559–572. – URL `http://pbil.univ-lyon1.fr/R/pearson1901.pdf`

[Rabbani 2006] Rabbani, T.: *Automatic Reconstruction of Industrial Installations using Point Clouds and Images*, TU Delft, Dissertation, 2006. – URL `http://www.ncg.knaw.nl/Publicaties/Geodesy/pdf/62Rabbani.pdf`

[Rau und Chen 2003] Rau, J. ; Chen, L.: Fast Straight Lines Detection Using Hough Transform with Principal Axis Analysis. In: *CSPRS* 8 (2003), Nr. 1, S. 15–34. – URL `http://dpl.csrsr.ncu.edu.tw/Journalpapers/J2003_CSPRS_HoughTransform.pdf`

[Risse 1989] Risse, T.: Hough Transform for Line Recognition: Complexity of Evidence Accumulation and Cluster Detection. In: *CVGIP: Image Understanding* 46 (1989), Juni, Nr. 3, S. 327–345. – URL `http://www.sciencedirect.com/science/article/B7GXG-4D8FSNB-7C/2/085f7c96259551e18caf91de4db998f5`

[Rodrigues und Loke 2000] Rodrigues, J. ; Loke, R.: Fast segmentation of 3D data using an octree. URL `http://w3.ualg.pt/~loke/../isacs/reports/porto0a.ps.gz`, Mai 2000 (1). – Forschungsbericht. -

[Rottensteiner und Briese 2002] Rottensteiner, F. ; Briese, C.: A New Method

for Building Extraction in Urban Areas from High-Resolution LIDAR Data. In: *Photogrammetric Computer Vision*, 2002, S. A: 295

[Rottensteiner u. a. 2005] Rottensteiner, F. ; Trinder, J. ; Clode, S. ; Kubik, K.: Automated Delineation of Roof Planes from LIDAR Data. In: *Proceedings of the ISPRS Workshop Laser scanning 2005* Bd. XXXVI, 2005, S. 221–226. – ISSN 1682-1777

[Rowland und Weisstein] Rowland, T. ; Weisstein, E.: *Parallelepiped*. – URL `http://mathworld.wolfram.com/Parallelepiped.html`

[Söderman u. a. 2004] Söderman, U. ; Ahlberg, S. ; Persson, Å. ; Elmqvist, M.: *Towards Rapid 3D Modelling of Urban Areas*. In on-line documentation of SimSafe2004: Improving Public Safety through Modelling and Simulation. 2004. – URL `http://www.sne.foi.se/documents/simsafe09.pdf`

[Serra 1982] Serra, J.: *Image Analysis and Mathematical Morphology*. Book, 1982

[TopoSys GmbH] TopoSys GmbH: *Falcon II Specification*. – URL `http://www.toposys.com/toposys-en/lidar-systems/falcon-technical-data.php?print=true`

[Tóvári und Pfeifer 2005] Tóvári, D. ; Pfeifer, N.: Segmentation-based Robust Interpolation - a New Approach to Laser Data Filtering. (2005). – URL `http://www.commission3.isprs.org/laserscanning2005/papers/079.pdf`

[UN Department of Economic and Social Affairs 2005] UN Department of Economic and Social Affairs: *World Urbanization Prospects: The 2005 Revision*. 2005. – URL `http://www.un.org/esa/population/publications/WUP2005/2005wup.htm`

[van Veen und Groen 1981] Veen, T. van ; Groen, F.: Discretization errors in the Hough transform. In: *Pattern Recognition* 14 (1981), Nr. 1-6, S. 137–145. – URL `http://dx.doi.org/10.1016/0031-3203(81)90055-8`

[Wassenberg 2005] Wassenberg, J.: *Timing Pitfalls and Solutions*. 2005. – URL `http://www.gamedev.net/reference/programming/features/timing/`

[Wehr und Lohr 1999] Wehr, A. ; Lohr, U.: Airborne laser scanning: An introduction and overview. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (1999), Juli, Nr. 2-3, S. 68–82

[Wulf 1972] Wulf, W.: A Case Against the GOTO. In: *SIGPLAN Notices* 7 (1972), November, Nr. 11, S. 63–69

[Yoo 2004] Yoo, T.: *Insight into Images - Principles and Practice for Segmentation, Registration, and Image Analysis*. A K Peters, 2004. – ISBN 1-56881-217-5