Universität Ulm
Fakultät für Mathematik und Wirtschaftswissenschaften

Kantenextraktion

Klassische Verfahren

Seminar Bildsegmentierung und Computer Vision
WS 2005/2006

Christoph Wagner

30. Januar 2006

Inhaltsverzeichnis

1 Grundlagen

1.1 Aufgabenstellung

Beim menschlichen Sehen spielen Kanten bzw. Konturen von Objekten eine wichtige Rolle. Sie ermöglichen es uns erst, Objekte, Gegenstände oder Personen zu erkennen. Die Abstraktionsfähigkeit des Gehirns geht dabei so weit, dass wir sogar in reinen Strichzeichnungen noch Menschen o.ä. wahrnehmen können. Kanten gehören damit also zu den wichtigsten Informationen, die in einem Bild enthalten sind.

Unser Ziel ist es nun, aus einem Graustufenbild Informationen über die darin enthaltenen Kanten zu extrahieren. Dazu benötigen wir eine mathematisch handhabbare Definition dieses Begriffs. Größere Bereiche im Bild, auf denen keine oder nur relativ geringe Helligkeitsänderungen vorliegen, nehmen wir als zusammenhängende Flächen war. Ist eine solche Fläche nun allerdings entlang einer erkennbaren Linie durch einen starken Kontrast von einer anderen Fläche abgegrenzt, so werden wir dort eine Begrenzung der Flächen, also einen Kante erkennen.

Das charakteristische Merkmal einer solchen Kante ist also eine lokal starke Kontraständerung über eine relativ kurze Distanz. Die Änderungsrate einer Funktion bezogen auf die Distanz ist aber gerade ihre erste Ableitung. Wenn wir also in der Ableitung der Bildfunktion nach lokalen Extremstellen suchen, dürfen wir hoffen, damit die gesuchten Kanten identifizieren zu können. Tatsächlich beruhen alle klassischen Kantendetektionsfahren in der ein oder anderen Form auf Ableitungen, weswegen sie auch häufig als *gradientenbasierte Verfahren* bezeichnet werden.

1.2 Anforderungen an Kantenfilter

Formal wollen wir zunächst einige Anforderungen festhalten, die wir an einen Kantenfilter stellen wollen. Zum einen ist dies die *Verschiebungsinvarianz*. Das bedeutet, dass die Güte der Kantendetektion nicht vom Ort der Kante abhängen soll. Mit anderen Worten, eine Verschiebung der Kante im Bild soll die Stärke, mit der sie erkannt wird, nicht verändern.

Bei der zweiten Anforderung handelt es sich um die *Isotropie*, oder auch *Rotationsinvarianz*. Sie postuliert, dass die Erkennungsgüte nicht vom Verlauf der Kante abhängen soll, d.h. eine Bilddrehung soll die Stärke, mit der eine Kante detektiert wird, nicht verändern. Ist dies nicht der Fall, spricht man von *Anisotropie* des Filters.

Wie wir noch sehen werden, ist die erste dieser beiden Anforderungen problemlos zu erfüllen, während die zweite von keinem der klassischen Verfahren vollständig erfüllt wird.

1.3 Lineare Filter

Bevor wir uns nun der eigentlich Aufgabe, nämlich der Entwicklung von Filtern zur Detektion von Kanten, widmen, wollen wir zunächst ein wichtiges Hilfsmittel dafür betrachten: lineare Filter. Wie der Name sagt, handelt es sich dabei um lineare Funktionen, die eine Bildfunktion in eine andere transformieren. Zur Entwicklung dieser Theorie ist es jedoch hilfreich, Bilder statt als Funktion von zwei Variablen als eine zweidimensionale Matrix $\mathbf{I} \in \mathbb{R}^{n \times m}$ aufzufassen.

Das Prinzip ist nun folgendes: ein Pixel, sowie eine gewisse Nachbarschaft davon (z.B. die 4er- oder 8er-Nachbarschaft) wird mit reellen Zahlen $\lambda_1, \ldots, \lambda_k$ gewichtet. Die so gewichteten Pixelwerte werden aufsummiert und der resultierende Wert an die Koordinaten des Ausgangspixels im Zielbild geschrieben. Bezeichnen wir die Elemente der Bildmatrix mit $i_{x,y}$, so ergibt sich beispielsweise unter Verwendung der 4er-Nachbarschaft der neue Wert eines Pixels wie folgt:

$$\tilde{i}_{x,y} = \lambda_1 i_{x,y} + \lambda_2 i_{x,y-1} + \lambda_3 i_{x-1,y} + \lambda_4 i_{x+1,y} \lambda_5 i_{x,y+1} \tag{1}$$

Es liegt nahe, die λ_k in dieser Gleichung nun auch zweidimensional entsprechend ihrer Position zu indizieren. Dann können wir sie der Übersichtlichkeit halber auch gleich in eine Matrix $\mathbf{H} = (\lambda_{x,y})$ schreiben. Die Filteranwendung kann man sich dann plastisch so vorstellen: für jedes Pixel in der Bildmatrix wird die Filtermarix so „darübergelegt“, dass ihr Mittelpunkt genau auf dem betrachteten Pixel liegt. Dann wird jedes Pixel der Filtermatrix mit dem darunterliegenden Bildpixel multipliziert. Die Summe dieser Werte ist dann das Filterergebnis an dieser Stelle.

Formal entspricht dies einer diskretisierten Faltung von $\mathbf{I}$ mit einem Faltungskern $\mathbf{H}$. Wir schreiben für diese Operation $\mathbf{I} * \mathbf{H}$ und definieren sie formal wie folgt:

$$(\mathbf{I} * \mathbf{H})(x, y) := \sum_{i=-r}^{r} \sum_{j=-r}^{r} \mathbf{I}(x + i, y + j)\mathbf{H}(r + i, r + j) \tag{2}$$

Dabei gehen wir davon aus, dass $\mathbf{H} \in \mathbb{R}^{2r+1 \times 2r+1}$ eine quadratische Matrix mit ungerader Zeilen- bzw. Spaltenzahl ist. Diese Beschränkung ist natürlich nicht zwingend notwendig, aber da solche Filter in der Praxis die weitaus häufigsten sind, soll das für unsere Zwecke hier genügen. Den Wert r interpretieren wir als Radius des Filters, er gibt an, wie viele Nachbarpixel in jeder Richtung (horizontal, vertikal und diagonal) verwendet werden.

Man kann zeigen, dass solche Filter außer der Linearität noch weitere Eigenschaften haben, z.B. sind sie auch assoziativ und kommutativ. Wegen der Linearität ist außerdem sofort klar, dass solche Filter verschiebungsinvariant arbeiten. Der ersten Anforderung aus Abschnitt 1.2 ist also bereits genüge getan, wenn wir lineare Filter zu Implementierung von Kantenfiltern verwenden.

Ein Problem gibt es allerdings noch: Formel 2 ist nur für Pixel „im Inneren“ der Matrix $\mathbf{I}$ wohldefiniert, d.h. für Pixel, deren Abstand zu jedem Rand mindestens r beträgt. Für alle anderen würden wir Werte „außerhalb“ der Bildmatrix benötigen, die wir natürlich nicht kennen. Die einfachste Lösung dieses Problems ist, für alle diese Werte einfach den Wert Null anzunehmen, sie also zu ignorieren. Man kann allerdings auch den in entsprechender Richtung am nächsten gelegenen Randbildpunkt verwenden (d.h. die Matrix wird „erweitert“, indem die Randpixel r mal außen herumgelegt werden) oder man interpretiert unzulässige Koordinaten als $\mod n$ bzw. $\mod m$, d.h. beim „Herauslaufen“ aus dem Bild läuft man am gegenüberliegenden Rand wieder in das Bild hinein.

Bei der praktischen Anwendung wird außerdem der Filter häufig normalisiert, d.h. das Filterergebnis wird noch durch die Summe der Matrixeinträge (oder auch durch die Summe

der Beträge) geteilt. Alternativ kann man jedoch auch die Intensität der Filterung regulieren, indem man stattdessen einen wählbaren Vorfaktor benutzt. Der Übersichtlichkeit halber werden wir aber im Folgenden auf die Angabe der Normierungsfaktoren verzichten.

Beispiel Als Beispiel für die Anwendung der linearen Filter (und auch, weil es später noch eine Rolle spielt) soll hier der sog. *Gauß'sche Glättungsfilter* dienen, der in so gut wie allen Bildverarbeitungsprogrammen implementiert ist. Dieser beruht auf einer Diskretisierung der Dichtefunktion $f(x,y) = \frac{1}{2\pi\sigma} \exp\left(-\frac{x^2+y^2}{\sigma^2}\right)$ der zweidimensionalen Normalverteilung. Der Vorfaktor $\frac{1}{2\pi\sigma}$ wird dabei normalerweise weggelassen. Stattdessen wird so normiert, dass näherungsweise ganzzahlige Werte für die Matrixeinträge verwendet werden können, um die Anwendung im Rechner zu beschleunigen. Für eine 3×3-Filtermatrix sieht das etwa wie folgt aus:

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Der Varianzparameter σ der Normalverteilung erlaubt es dabei, die Glockenkurve schmal und steil oder eher flach und breit zu machen, entsprechend einer schwachen oder starken Glättung. Eben weil die Glockenkurve bei großen Werten für σ sehr breit wird, muss die Größe, also der Radius der Filtermatrix von diesem Wert abhängen. In der Praxis nimmt man hierfür meist den Wert 3σ.

Der Vorteil des Gaußfilters gegenüber eines normalen Mittelwertfilters (dessen Filtermatrix nur aus Einsen bestünde) liegt übrigens in der geringeren „Verwaschung“ von Konturen bei gleichzeitig guter Rauschunterdrückung.

2 Gradientenbasierte Verfahren erster Ordnung

Wir kommen nun zum eigentlichen Thema, nämlich der Kantendetektion, zurück. Wie wir bereits ganz zu Beginn überlegt haben, sind Ableitungen ein geeignetes Mittel zur Erkennung von Kanten. Da die Ableitung eine lineare Operation ist, dürfen wir also auch hoffen, einen Ableitungsfilter als linearen Filter realisieren zu können. Da wir in diesem Abschnitt nur die erste Ableitung der Bildfunktion betrachten, werden die resultierenden Verfahren auch als Verfahren erster Ordnung bezeichnet.

2.1 Herleitung

Gehen wir zunächst noch einmal davon aus, unser Bild sei eine stetige Funktion $f : \mathbb{R}^2 \to \mathbb{R}$ von zwei Variablen. Die Ableitung ist dann gegeben als der *Gradientenvektor*

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{pmatrix} \tag{3}$$

Dieser Vektor gibt, wie hinlänglich bekannt ist, in jedem Punkt die Richtung des steilsten Anstiegs der Funktion an. Die Stärke dieses Anstiegs ist definiert als der *Betrag* des Gradienten

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{4}$$

Außerdem läßt sich aus dem Gradientenvektor dessen Richtungswinkel berechnen, dieser ist nämlich genau

$$\Phi = \arctan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right) \tag{5}$$

Da die von uns betrachteten Bilder jedoch nicht stetig sind, ist für sie auch keine Ableitung definiert. Stattdessen müssen wir uns mit einer Annäherung der Ableitung über diskrete Differenzen begnügen. Dabei verwenden wir die symmetrischen diskreten Differenzen

$$\frac{\partial f}{\partial x} \approx \frac{f(x+1,y) - f(x-1,y)}{2} \tag{6}$$

bzw.

$$\frac{\partial f}{\partial y} \approx \frac{f(x,y+1) - f(x,y-1)}{2} \tag{7}$$

Mithilfe der Gleichungen (6) und (7) erhalten wir also eine Schätzung des Gradientenvektors in einem beliebigen Bildpunkt (x, y). Dies können wir wie folgt als linearen Filter schreiben:

$$\mathbf{H}_x = \begin{bmatrix} 0 & 0 & 0 \\ -0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{8}$$

für die horizontale und

$$\mathbf{H}_y = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{9}$$

für die vertikale Richtung. Damit können wir nun die beiden partiellen Ableitungen der Bildfunktion durch Anwendung dieser Filter berechnen:

$$\mathbf{D}_x = \mathbf{I} * \mathbf{H}_x, \quad \text{bzw.} \quad \mathbf{D}_y = \mathbf{I} * \mathbf{H}_y \tag{10}$$

Schlußendlich berechnen wir nun daraus das Kantenbild, indem wir den Betrag des Gradienten aus den partiellen Ableitungen $\mathbf{D}_x$ und $\mathbf{D}_y$ berechnen:

$$\mathbf{E}(x,y) = \sqrt{\mathbf{D}_x^2(x,y) + \mathbf{D}_y^2(x,y)} \tag{11}$$

Abbildung 1: Ausgangsbild und horizontales Filterergebnis

Wir schreiben Gleichung (11 im Folgenden kürzer, indem wir auf die Angabe der Koordinaten verzichten und die verwendeten Operationen in Zukunft für Matrizen immer komponentenweise verstehen wollen:

$$\mathbf{E} = \sqrt{\mathbf{D}_x^2 + \mathbf{D}_y^2} \tag{12}$$

Beispiel Als einfaches Beispiel betrachten wir das Bild in Abbildung 1, das ein weißes Dreieck auf schwarzem Hintergrund zeigt, dessen Kanten mit einem Gaußfilter verwischt wurden. Das Bild verwendet 256 Graustufen, wobei Schwarz dem Wert 0 und Weiß dem Wert 255 entspricht.

Nach Anwendung des horizontalen bzw. des vertikalen Filters ergeben sich die Bilder aus Abbildung 1 bzw. 2, wobei jedoch eine kleine Besonderheit beachtet werden muss. Da der Filter nämlich offenbar auch negative Werte liefern kann, wir aber nur positive Grauwerte zugelassen haben, sind bei diesen Bildern die Wertebereiche verschoben. Hier geht die Skala von -128 für reines Schwarz über 0 (mittleres Grau) bis 127 für weiße Pixel.

Das Endergebnis des Filters, nämlich das Kantenbild, welches den des geschätzten Gradientenvektors an jedem Bildpunkt darstellt und sich aus Formel 11 bzw. 12 ergibt, ist rechts in Abbildung 2 zu sehen. Da der Betrag des Gradienten nur positive Werte annehmen kann ist hier die Interpretation der Grauwerte wieder wie im Ausgangsbild.

2.2 Prewitt- und Sobel-Operator

Ein Problem des bisher betrachteten, einfachen Gradientenoperators ist seine relativ hohe Anfälligkeit gegenüber Bildrauschen. Zufällige Bildstörungen werden nicht selten auch als Kanten erkannt. Gegen zufälliges Rauschen hilft es, das Bild zu glätten. Da die Faltung kommutativ und assoziativ ist, kann man beide Operationen auch in einer Faltungsmaske kombinieren. Sowohl der Prewitt- als auch der Sobel-Operator beruhen auf dieser Idee.

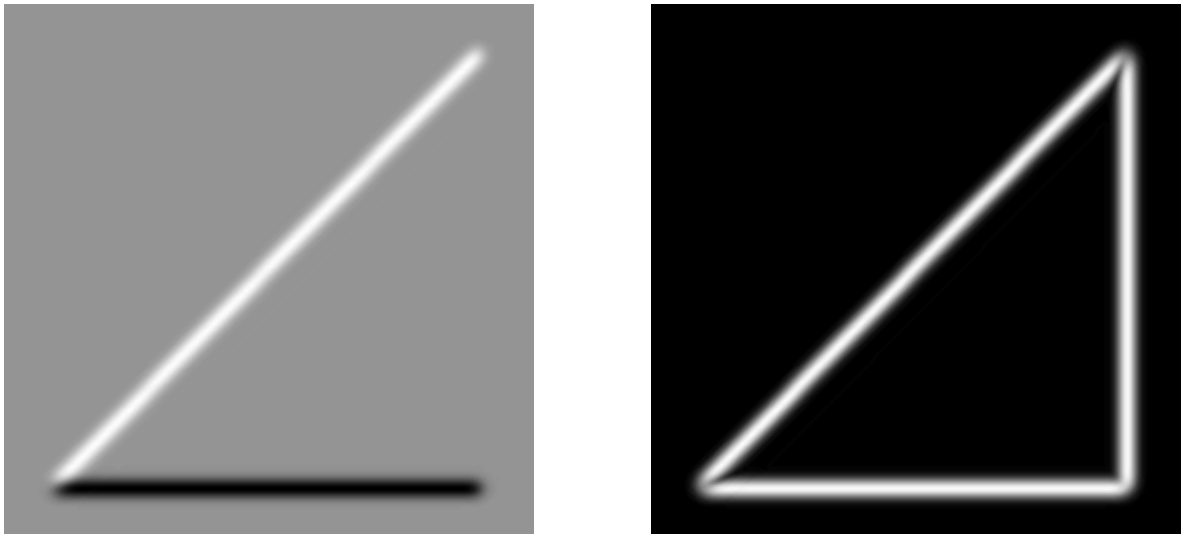

Abbildung 2: Ergebnis des vertikalen Filters und endgültiges Kantenbild

2.2.1 Der Prewitt-Operator

Beim Prewitt-Operator kommt dafür eine einfache Mittelwertberechnung über die jeweils benachbarten Zeilen bzw. Spalten zum Einsatz. Die Glättung wird orthogonal zu Filterrichtung vorgenommen, d.h. bei Berechnung des horizontalen Filters wird über die benachbarten Zeilen, beim vertikalen über die Spalten geglättet. Die Filtermasken ergeben sich also wie folgt:

$$\mathbf{H}_x^P = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{13}$$

$$\mathbf{H}_y^P = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{14}$$

2.2.2 Der Sobel-Operator

Ganz ähnlich aufgebaut ist der Sobel-Operator. Im Unterschied zum Prewitt-Operator wird jedoch hier die zentrale Filterzeile bzw. -spalte doppelt so stark gewichtet wie die übrigen Zeilen bzw. Spalten.

$$\mathbf{H}_x^P = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{15}$$

$$\mathbf{H}_y^P = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{16}$$

Der Sobel-Operator ist dem Prewitt-Operator in Sachen Isotropie leicht überlegen. Aufgrund der einfachen Implementierung und der recht guten Ergebnisse ist er in praktisch alles Bildverarbeitungsprogrammen zu finden. Seine Isotropie-Eigenschaften disqualifizieren ihn aber laut [1] in einigen Anwendungen.

Zwar kann man die Isotropie noch verbessern, indem man die Gewichtung der zentralen Spalten bzw. Zeilen modifiziert, aber an die größte Einfluss ist verfahrensbedingt und daher nicht veränderbar. Laut obiger Quelle hat die folgende Variante des Sobel-Operators die beste Isotropie in dieser Klasse:

$$\tilde{\mathbf{H}}_x^S = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}, \quad \tilde{\mathbf{H}}_y^S = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{17}$$

2.3 Kompassoperatoren

Die grundlegende Schwäche der weiter oben betrachteten Verfahren sind ihre Präferenz für horizontal, bzw. vertikal verlaufende Kanten, die aus der Tatsache rührt, dass eben gerade nur horizontale und vertikale Schätzungen des Gradienten verwendet werden. Eine Möglichkeit, dieses Problem abzuschwächen ist daher die Verwendung von zusätzlichen Filtern zur Schätzung des Gradienten beispielsweise in Diagonalrichtung. Bei solchen Verfahren spricht man von Kompassoperatoren. Ein klassisches Beispiel ist der sog. *Kirsch-Operator*. Dieser verwendet zusätzlich Schätzungen in Diagonalrichtung, also jeweils im Abstand von 45 Grad.

$$\begin{aligned}
\mathbf{H}_0^K &= \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} & \mathbf{H}_4^K &= \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \\
\mathbf{H}_1^K &= \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} & \mathbf{H}_5^K &= \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \\
\mathbf{H}_2^K &= \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} & \mathbf{H}_6^K &= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \\
\mathbf{H}_3^K &= \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} & \mathbf{H}_7^K &= \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}
\end{aligned} \tag{18}$$

Von diesen acht Filtermatrizen unterscheiden sich je zwei offensichtlich nur durch ihr Vorzeichen. Das heißt also, dass nur die Matrizen $\mathbf{H}_0, \ldots \mathbf{H}_3$ auf das Bild angewendet werden müssen. Das Ergebnis der anderen vier Filter unterscheidet sich lediglich durch sein Vorzeichen. Es werden also folgende Zwischenbilder berechnet:

$$\begin{aligned}
&\mathbf{D}_0 = \mathbf{I} * \mathbf{H}_0, \quad \mathbf{D}_1 = \mathbf{I} * \mathbf{H}_1, \quad \mathbf{D}_2 = \mathbf{I} * \mathbf{H}_2, \quad \mathbf{D}_3 = \mathbf{I} * \mathbf{H}_3, \\
&\mathbf{D}_4 = -\mathbf{D}_0, \quad \mathbf{D}_5 = -\mathbf{D}_1, \quad \mathbf{D}_6 = -\mathbf{D}_2, \quad \mathbf{D}_7 = -\mathbf{D}_3
\end{aligned} \tag{19}$$

Das Ergebnis des Filters ist definiert als das pixelweise Maximum der acht Zwischenbilder, also

$$\mathbf{D}^K := \max_{i=0\ldots7} \mathbf{D}_i = \max_{i=0\ldots3} |\mathbf{D}_i| \tag{20}$$

Der Teilfilter, der am stärksten anspricht, bestimmt auch die Richtung der Kante.

In der Praxis hat dieses Verfahren jedoch keine große Bedeutung erlangt, da dessen Ergebnisse kaum besser sind als die des Sobel-Operators. Lediglich die recht aufwendige Wurzelberechnung kann man sich hier sparen (muss dafür aber doppelt so viele Faltungen berechnen).

3 Gradientenbasierte Verfahren zweiter Ordnung

3.1 Laplace-Filter

Bisher wurde bei allen Verfahren lediglich die erste Ableitung, also der Gradient, der Bildfunktion betrachtet. Auch haben wir bisher nicht explizit (wie eigentlich angekündigt) nach lokalen Extrempunkten dieser Ableitung gesucht. Dies soll nun nachgeholt werden. Bekanntermaßen besitzt die zweite Ableitung einer Funktion an einer Extremstelle der ersten Ableitung gerade einen Nulldurchgang. Um dies zur Kantendetektion auszunutzen, benötigen wir zunächst die Definition des klassischen Laplace-Operators aus der Analysis:

$$\nabla^2 f := \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{21}$$

Wenn also beide partiellen Ableitungen 0 werden, verschwindet der Laplace-Operator an dieser Stelle. Dort haben wir also einen Hinweis auf eine Kante. Um den Laplace-Operator als Faltungsmatrix schreiben zu können, diskretisieren wir die zweiten partiellen Ableitungen wie folgt:

$$\begin{aligned} \frac{\partial^2 f(x,y)}{\partial x^2} &\approx \frac{\partial(f(x+1,y) - f(x,y))}{\partial x} \\ &\approx f(x+1,y) - f(x,y) - (f(x,y) - f(x-1,y)) \\ &= f(x+1,y) - 2f(x,y) + f(x-1,y) \end{aligned} \tag{22}$$

Analog verfahren wir für die zweite Koordinate. Damit ergibt sich die Filtermaske

$$\mathbf{H}^L = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{23}$$

Auch ohne Suche nach Vorzeichenwechseln liefert dieser Filter schon recht gute Ergebnisse, jedoch ist er stark rauschanfällig. Dass kann man sich auch recht leicht vorstellen, aufgrund der „punktförmigen“ Struktur des Filters, der den mittleren Bildpunkt im Gegensatz zu seinen Nachbarn recht stark gewichtet. Bei punktförmigem Rauschen ist der „glättende“ Einfluss der umgebenden Pixeln nicht groß genug, sodass solche Bildartefakte eher noch verstärkt werden. Ein Beispiel zeigt Abbildung 3 (links).

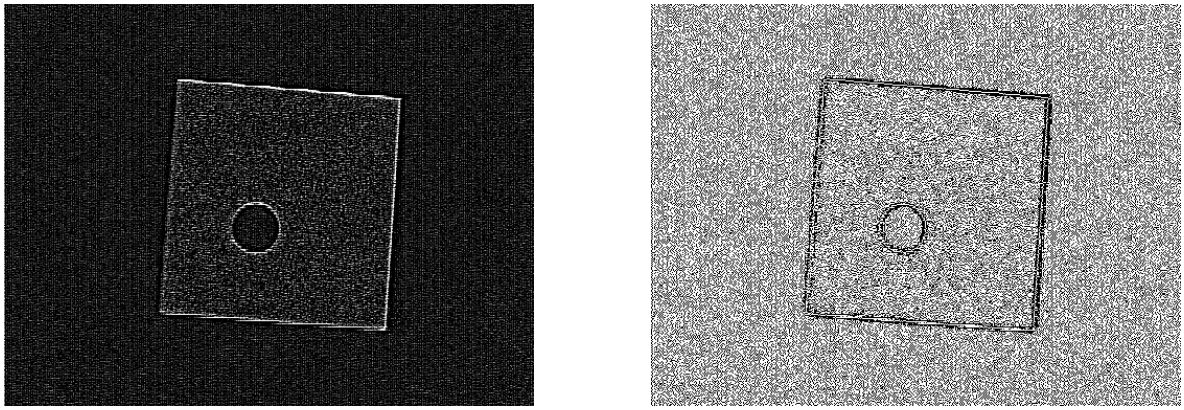

Abbildung 3: Ergebnis des Laplace-Filters (rechts mit, links ohne Suche nach Vorzeichenwechseln)

Will man die Suche nach Vorzeichenwechseln durchführen, so kann man etwa wie folgt vorgehen: Für jedes Pixel betrachtet man alle Paare von sich gegenüberliegenden Nachbarpixeln (also z.B. die Nachbarn links und rechts des Pixels). Besitzt mindestens ein Paar davon unterschiedliche Vorzeichen, markiert man das aktuelle Pixel als Kantenpixel, ansonsten als Nicht-Kantenpixel. In diesem Fall erhält man statt einem Graustufen- ein Binärbild. Wegen der hohen Rauschanfälligkeit des Laplace-Filters wird bei diesem jedoch meistens auf eine Vorzeichensuche verzichtet, da diese durch das vorhandene Rauschen die Ergebnisse eher verschlimmert als verbessert (in Abbildung 3 rechts zu sehen). Beim nachfolgend besprochenen Laplacian-of-Gaussian Verfahren wird sie dagegen häufig eingesetzt.

Eine häufig eingesetzte Variante des Laplace-Operators ist Filtermatrix (24). Sie zeichnet sich vor allem durch eine etwas verbesserte Isotropie aus und ensteht unter Hinzunahme der gemischten zweiten Ableitungen bei der Approximation.

$$\tilde{\mathbf{H}}^L = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{24}$$

3.2 Laplacian of Gaussian

Wie wir bereits gesehen haben, hilft gegen Rauschen meist eine Glättung des Bildes. Dies kann durch die Anwendung eines Gaußfilters vor der eigentlichen Kantenfilterung geschehen. Da die Anwendung von linearen Filtern jedoch assoziativ ist, kann man stattdessen auch den Laplace-Operator auf den Gaußfilter anwenden (d.h. die beiden Filtermatrizen miteinander falten) und das Bild mit der resultierenden Matrix filtern.

Zur Erzeugung von solchen Filtermasken bietet sich jedoch eine etwas andere Vorgehensweise an. Und zwar können wir, da wir die Dichte der Gaußverteilung ja kennen (und diese auch genügend oft differenzierbar ist), direkt den analytischen Laplace-Operator dieser Funktion berechnen, d.h. die Summe ihrer zweiten Ableitungen. Bezeichnen wir zunächst mit $\phi(x, y)$ die Dichte der Gaußverteilung mit Erwartungswert 0 und Varianz σ, also

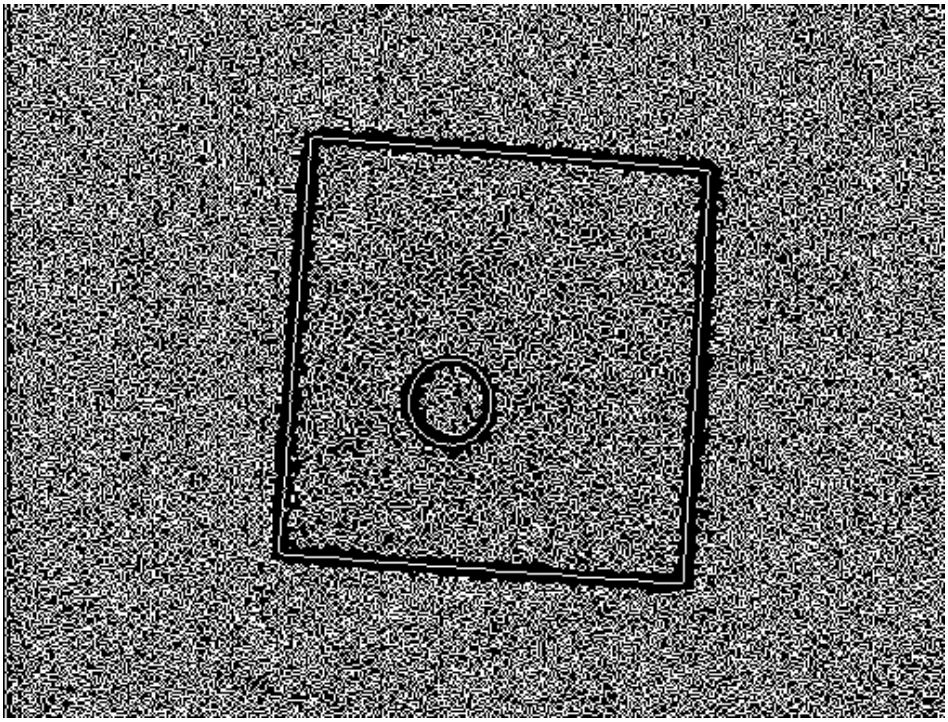

Abbildung 4: Ergebnis des LoG-Filters mit $\sigma = 1$

$$\phi(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{25}$$

Dann gilt

$$\nabla^2\phi(x, y) = \frac{1}{2\pi\sigma^4} \left(\frac{x^2 + y^2}{\sigma^2} - 2 \right) e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{26}$$

Mithilfe dieser Funktion können wir nun die Einträge für eine Filtermatrix berechnen. Dabei sollte der Radius wieder wie im Beispiel ... gewählt werden, also $r = 3\sigma$. Der Normierungsfaktor $\frac{1}{2\pi}$ wird ebenfalls wieder durch einen anderen ersetzt, der es erlaubt, die Matrixeinträge möglichst gut durch ganze Zahlen zu approximieren.

Der so resultierende lineare Kantenoperator wird nach seinem Entstehungsprinzip häufig „Laplacian of Gaussian“ gennant. In der Literatur wird häufig auch vom *Marr-Hildreth-Operator* gesprochen, nach D. Marr und E. Hildreth, die ihn in [3] zuerst eingesetzt haben.

Wie im vorigen Abschnitt erwähnt, wird dieser Filter meistens mit einer Suche nach Vorzeichenwechseln kombiniert. Abbildungen 4 und 5 zeigen das Ergebnis des LoG-Filters mit verchiedenen Werten für σ, angewendet auf das gleiche Ausgangsbild. Wie man gut sehen kann, ist der Effekt des Parameters σ ähnlich wie beim Gauß-Filter. Bei kleinen Werten wird wenig geglättet, daher werden auch filigrane Strukturen noch erkannt. Bei großen Werten wird dagegen recht stark geglättet, sodass nur noch starke Kanten als solche erkannt werden.

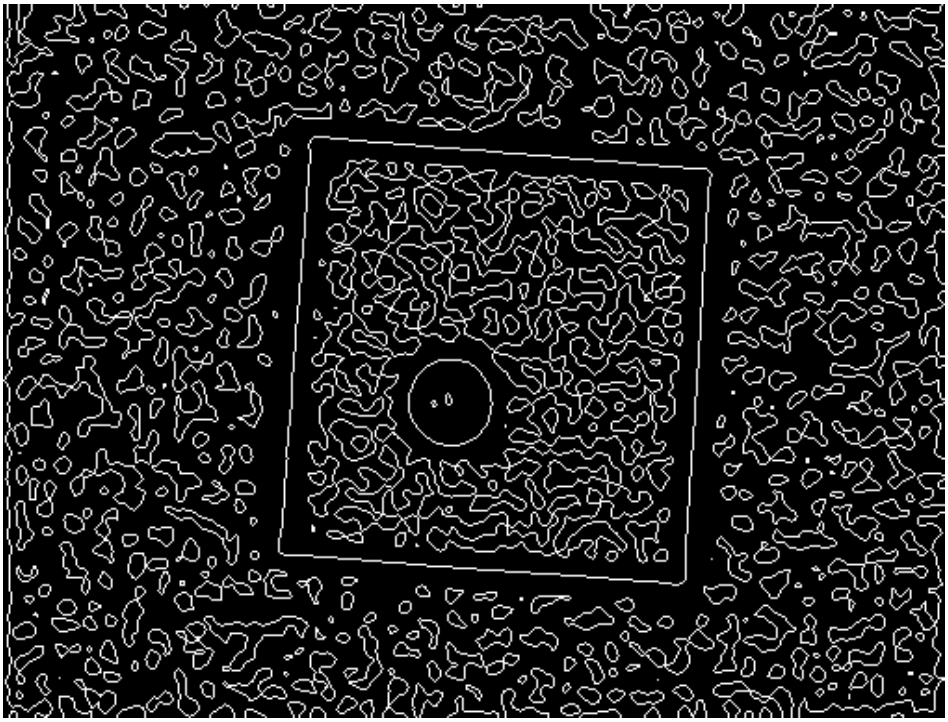

Abbildung 5: Ergebnis des LoG-Filters mit $\sigma = 3$

4 Canny-Filter

Eine grundsätzlich andere Herangehensweise an die Kantendetektion hat J. Canny in [4] benutzt. Er setzte sich zum Ziel, einen optimalen Kantendetektor zu entwickeln, wobei er die Optimalität bezüglich der folgenden drei Kriterien meinte:

1. **Erkennung:** alle tatsächlichen Kanten sollen gefunden werden, aber keine falschen
2. **Lokalisierung:** Abstand zwischen tatsächlicher und erkannter Kante soll möglichst klein sein
3. **Ansprechverhalten:** Kanten sollen nicht mehrfach erkannt werden, d.h. insbesondere soll ihre Breite nicht mehr als einen Pixel betragen

Für jedes dieser drei Kriterien definierte er ein Maß, um die Optimalität mathematisch nachvollziehbar zu machen. Der daraufhin entwickelte Algorithmus, der diese Kriterien optimal erfüllen soll, besitzt einige bemerkenswerte Unterschiede zu den bisher betrachteten. Zum einen liefert er Binär- statt Graustufenbilder (d.h. jedes Pixel kann lediglich die Werte 0=„keine Kante“ oder 1=„Kante“) annehmen. Zum anderen wird bei diesem Filter auch tatsächlich die Information über die Kanten- bzw. Gradientenrichtung verwendet, von der wir in Abschnitt 2.1, Formel (5) gesehen haben, dass wir sie aus dem geschätzen Bildgradienten berechnen können.

Der Algorithmus ist ein mehrstufiges Verfahren, welches die bisher verwendeten Verfahren mit zusätzlichen Schritten zur Nachbearbeitung kombiniert:

Abbildung 6: Ausgangsbild für Canny-Algorithmus

1. **Glättung:** zur Unterdrückung von Rauschen
2. **Kantendetektion:** Schätzung des Bildgradienten sowie seiner Richtung in jedem Pixel
3. **Unterdrückung von Nicht-Maxima:** Nur lokale Maxima der Kantenstärke werden als potenzielle Kantenpixel zugelassen
4. **Hysterese:** Unterdrückung nichtrelevanter Kanten mittels eines Zwei-Schwellenwerteverfahrens

Im Folgenden sollen diese Schritte kurz erläutert werden. Zur Illustration soll das Bild aus Abbildung 6 dienen.

4.1 Glättung und Kantendetektion

Der erste Schritt besteht einfach in einer Anwendung der bereits besprochenen Verfahren zur Bildglättung und Kantendetektion. Zuerst wird das Bild mit einem Gaußfilter geglättet, danach wird mittels des Sobel-Operators der horizontale und vertikale Gradient geschätzt und daraus die Kantenstärke und -richtung in jedem Bildpunkt berechnet. Diese Informationen (also das Kantenbild und das Richtungsbild) werden in den nächsten Schritten weiterverbeitet. Vorher jedoch müssen die geschätzten Gradientenwinkel auf 0, 45, 90 oder 135 gerundet werden, da wir nur in diesen Richtungen Nachbarpixel kennen.

Die Abbildungen 7 und 8 zeigen das resultierende Kantenbild (das nichts wesentlich neues bringt) sowie dasselbe Kantenbild, aber mit farblicher Kodierung für die (gerundeten)

Abbildung 7: Kantenbild

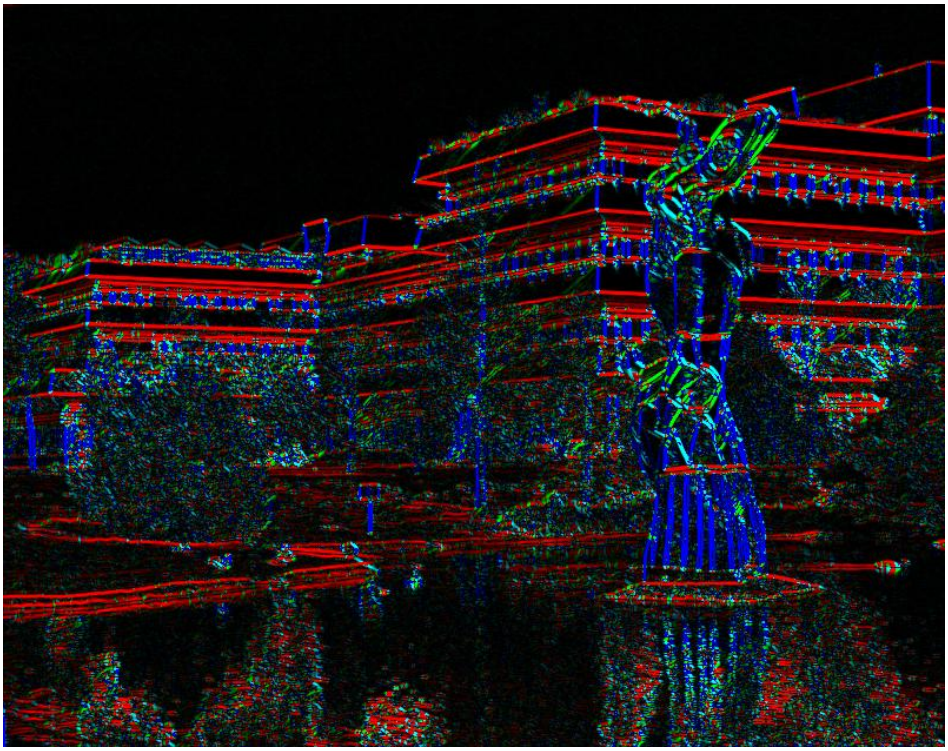

Abbildung 8: Kantenbild mit Gradientenrichtungen

Abbildung 9: Kantenbild nach Non-Maxima-Suppression

Gradientenrichtungen. Dabei sind horizontal verlaufende Kanten (also solche mit vertikalem Gradient) rot, vertikale Kanten blau, diagonal von rechts oben nach links unten verlaufende Kanten grün, und die (wenigen) restlichen türkis eingefärbt.

4.2 Unterdrückung von Nicht-Maxima

Dieser Schritt sorgt dafür, dass die Kanten genauer lokalisiert werden, was bedeutet, dass Kanten, deren Breite mehr als ein Pixel beträgt, „ausgedünnt“ werden, d.h. sie werden auf ein Pixel Breite reduziert. Dabei ist es kritisch, wo diese ein Pixel breite Kante platziert wird, um die Lokalisationsfehler zu minimieren (2. Optimalitätskriterium).

Bei einer Kante von mehr als einem Pixel Breite werden wir nun den tatsächlichen Verlauf gerade dort vermuten, wo der Betrag des Gradienten am größten ist. D.h. wir suchen lokale Maxima in Gradientenrichtung (dafür benötigen wir offensichtlich die Ergebnisse aus dem ersten Schritt). Alle Pixel, die solche Maxima darstellen, werden beibehalten, alle anderen Pixel setzen wir auf 0, wir unterdrücken also alle Pixel, die keine lokalen Maxima sind. Das Ergebnis dieser Prozedur am Beispielbild zeigt die Abbildung 9.

4.3 Hysterese

Nach dem vorherigen Schritt haben wir ein Bild erhalten, auf dem alle Kanten lediglich nur noch ein Pixel breit ist. Diese besitzen aber z.T. noch unterschiedliche Helligkeiten. Außerdem ist noch keine „Relevanzbewertung“ der Kanten vorgenommen worden, es ist also

Abbildung 10: Ergebnis des Canny-Filters

wahrscheinlich, dass sich noch irrelevante, also falsche Kanten, im Bild befinden. Als falsche Kanten bezeichnen wir dabei solche, deren Grauwerte unter einem bestimmten (von uns zu wählenden) Schwellenwert liegen.

Es wäre also nun eine Möglichkeit, einfach einen solchen Schwellenwert vorzugeben und alle Pixel mit niedrigeren Werten auf Schwarz, den Rest auf Weiß zu setzen. Damit würden aber viele relevanten Kanten „zerissen“ werden, nämlich solche, in deren Verlauf zwar meist recht hohe Grauwerte auftreten, mitunter aber auch solche, die unter dem Schwellenwert liegen. An solchen Stellen würde eine solche Kante also irrtümlicherweise in zwei seperate Teilstücke zerlegt werden.

Canny setzte daher auf eine *Hysterese* genanntes Verfahren mit zwei Schwellenwerten, die wir als T_1 und T_2 bezeichnen wollen, wobei $T_1 \leq T_2$ gelten möge. Alle Pixel, deren Werte unterhalb von T_1 liegen, werden direkt auf Schwarz gesetzt, also nicht weiter betrachtet. Alle Pixel mit Werten über T_2 werden dagegen unmittelbar als Kantenpixel markiert. Mit den restlichen Pixeln wird nun wie folgt verfahren: ist mindestens eines der Nachbarpixel in Gradientenrichtung ebenfalls ein Kantenpixel, so wird das aktuelle Pixel auch als Kantenpixel markiert. Sind dagegen beide Nachbarn in Gradientenrichtung keine Kantenpixel, wird auch das aktuelle nicht als solches markiert. Das Endergebnis dieses Filters zeigt Abbildung 10.

5 Fazit

Alle hier besprochenen Kantendetektionverfahren haben ihre Vor- und Nachteile. Während Prewitt- bzw. Sobel-Operator äußerst einfach zu implementieren sind, läßt leider ihre Isotropie zu wünschen übrig. Der Laplace-Operator dagegen ist zwar nicht so stark richtungsabhängig, dafür aber wesentlich rauschempfindlicher. Mit dem Übergang zum Laplacian-of-Gaussian-Filter handelt man sich, sofern man mit großen σ Werten arbeiten muss, dafür einen nicht unbeträchtlichen Rechenaufwand ein, da die Größe der Filtermatrix sehr groß sein kann. Außerdem erzeugt dieser Filter u.U. sehr viele falsche Kanten, wie wir im Beispiel gesehen haben. Der Canny-Filter ist von den betrachteten Verfahren sicherlich der flexibelste und liefert für die meisten Anwendungen brauchbare Ergebnisse, ist dafür jedoch recht schwierig zu implementieren. Außerdem ist das Ergebnis sehr empfindlich gegenüber der Wahl der Schwellenwerte. Bei der Anwendung ist daher häufig längeres Ausprobieren erforderlich. Für dieses Problem gibt es jedoch Ansätze zu automatischen Berechnung der Schwellenwerte.

Die mithilfe dieser Verfahren gewonnenen Informationen über Kanten kann man zur Erkennung von Objekten heranziehen. Dazu ist meistens die Nachverfolgung der Kanten nötig. Ein Problem dabei, dass auch der Canny-Filter nicht vollständig lösen kann, tritt auf, wenn eine Kante in einem Bereich schwachen Kontrasts „versackt“. In solchen Fällen kann man u.U. nur mit gewissen Vorinformationen über die zu erkennenden Objekte geschlossene Kantenzüge detektieren.

Ein weiteres Problem, mit dem wir uns gar nicht beschäftigt haben, ist die Anwendung solcher Verfahren auf Mehrkanalbilder, also z.B. Farbbilder. Da verschiedene Kanäle u.U. widersprüchliche Informationen über Kanten liefern können, ist es also in den meisten Fällen keine gute Idee, die betrachteten Algorithmen separat auf jeden einzelnen Kanal anzuwenden und dann zu einem Ergebnis zu kombinieren. Vielmehr benötigt man ganz andere Maße zur Berechnung der Stärke einer Kante.

Literatur

[1] B. Jähne, *Digitale Bildverarbeitung*, Kapitel 11, Springer, 2002

[2] W. Burger, M. J. Burge, *Digitale Bildverarbeitung*, Kapitel 7, Springer, 2005

[3] D. Marr, E. Hildreth, *Theory of Edge Detection*, Proceedings of the Royal Society of London, Series B, Vol. 207:187-217 (1980)

[4] J. Canny, *A Computational Approach To Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8:679-714 (1986)

[5] Canny-Filter online (eigene Bilder hochladbar)
`http://matlabserver.cs.rug.nl/cannyedgedetectionweb/web/index.html`