GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

XX XXX XX
YYY YYY Y

Bachelorarbeit
im Studiengang „Angewandte Informatik“

SUSAN

Ein Ansatz zur Strukturerkennung in Bildern

Julian Lüken
julian.lueken@stud.uni-goettingen.de

Institut für Numerische und Angewandte
Mathematik

Bachelor und Masterarbeiten des Zentrums für
angewandte Informatik an der
Georg-August-Universität Göttingen

10. Oktober 2019

Inhaltsverzeichnis

Kapitel 1

Einführung

Kantendetektoren sind ein wichtiges Werkzeug der Bildverarbeitung...

Kapitel 2

Mathematische Grundlagen

2.1 Analytische Grundlagen

2.2 Varianzanalyse

2.3 Bildverarbeitung

In der digitalen Bildverarbeitung für Graustufenbilder betrachten wir Abbildungen der Form

$$I : [0, H-1]_{\mathbb{Z}} \times [0, B-1]_{\mathbb{Z}} \to [0, 255]_{\mathbb{Z}}.$$

Solche Abbildungen werden im weiteren Verlauf dieser Arbeit schlichtweg als Bilder bezeichnet. H steht für die Höhe und B für die Breite des Bildes. Ferner steht 0 für schwarz und 255 für weiß.

Eines der wichtigen Instrumente der Bildverarbeitung ist die Faltung. Da die Faltung allerdings nur für Funktionen definiert wird und die Abbildung I keine Funktion ist, müssen wir die Faltung für den diskreten Fall neu definieren. Diese Definition trägt den Namen Filtermaske. Eine Filtermaske ist eine Matrix $k \in \mathbb{R}^{n \times m}$, die folgendermaßen Anwendung findet:

$$O(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} k(i, j) \cdot I(x - i + a, y - j + b),$$

wobei (a, b) der Nucleus der Filtermaske ist, I das Eingangsbild und O das neu gewonnene Bild. Wir schreiben, wie bei der Faltung,

$$O = k * I$$

Ein Beispiel für eine solche Filtermaske bietet der sogenannte Gaußsche Weichzeichner. Im stetigen Fall würde man ihn anwenden, indem man eine Funktion mit der Funktion der

Normalverteilung in zwei Dimensionen faltet. Die Funktion der Normalverteilung lautet:

$$g(x,y) := \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Im Fall von digitalen Bildern kennen wir nur diskrete Bildpunkte, also muss eine Approximation genügen. Man wähle eine Größe $n = m$ der Filtermaske und summiere für $x, y \in [-n, n]_{\mathbb{Z}}$ jeweils

$$G(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(x + \frac{i}{N}, y + \frac{j}{N})$$

für ein passendes σ und ein großes N. Die Größe der Filtermaske bestimmt sich in diesem Fall über das gewählte σ. Eine Faustregel dafür lautet, dass falls $\sqrt{\hat{x}^2 + \hat{y}^2} \geq 3\sigma$ gilt, auch $g(\hat{x}, \hat{y}) \approx 0$ gilt. Ausgehend davon können wir $n = \lfloor 3\sigma \rfloor - 1$ wählen und erhalten somit die Filtermaske

$$G = \frac{1}{273} \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix}.$$

für $\sigma = 1$, $N = 1000$ und $n = 2$. Wenden wir nun unseren Operator G auf folgendes linkes Bild an, so erhalten wir das nachstehende rechte Bild:

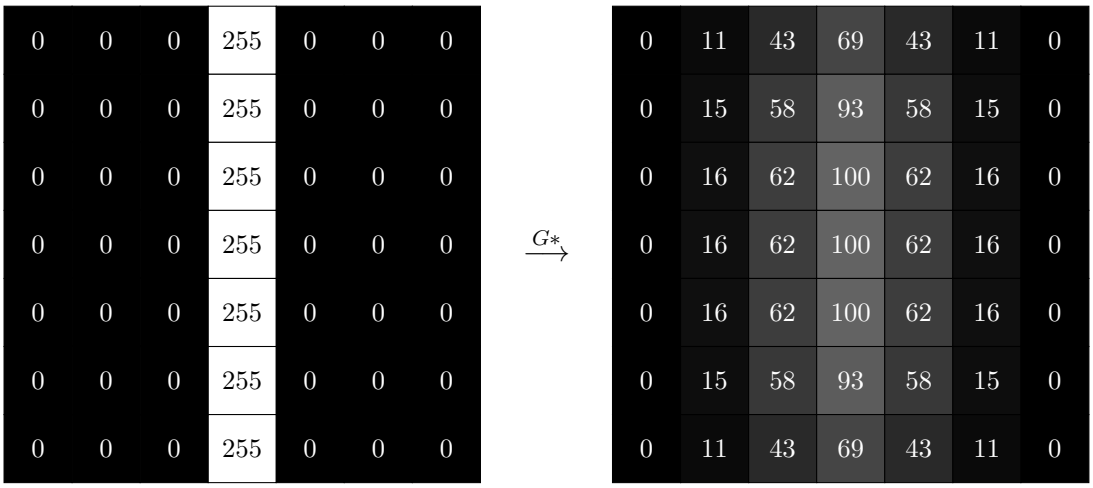

Abbildung 2.1: Die berechnete Gauß-Filtermaske in Aktion.

Kapitel 3

Der SUSAN Kantendetektor

In diesem Kapitel stellen wir den Algorithmus dar, danach erläutern einige Details zur Implementation des Algorithmus in Python 3. Im dritten Abschnitt leiten wir her, warum der SUSAN-Kantendetektor funktioniert und im letzten Teil führen wir ein paar numerische Experimente durch.

3.1 Der Algorithmus

Der SUSAN Kantendetektor aus [ref] besteht im Grunde aus drei großen Schritten:

Im ersten Schritt wird eine Maske über jedes Pixel gelegt. Mithilfe dieser Maske berechnen wir für jedes Pixel eine Kennzahl A, die im weiteren Verlauf dieser Arbeit „Antwort“ heißt. Die lokalen Maxima von A liegen genau auf den lokalen Extrema der partiellen Ableitungen in horizontaler und vertikaler Richtung unseres Eingangsbildes I, wie wir später noch zeigen.

Der zweite Schritt ist die sogenannte Non-Maximum-Suppression, bei der wir mithilfe der Richtung der im ersten Schritt berechneten Antwort die Kanten im Eingangsbild I noch genauer lokalisieren. Die Lokalisation von Kanten auf Bildern ohne Rauschen ist nach dem zweiten Schritt abgeschlossen.

Da der erste Schritt des SUSAN-Kantendetektors auf einem Schwellenwert t basiert, vernachlässigt er Intensitätsdifferenzen in I, die unterhalb von t liegen. Falsch positive und falsch negative Ergebnisse sind auf mit Rauschen behafteten Digitalbildern keine Seltenheit. Aus diesem Grund gibt es noch einen dritten Ausdünnungsschritt, in welchem räumlich isolierte Antworten entfernt werden und Kanten vervollständigt werden. Diesen Schritt behandeln wir näher in Kapitel [implementation]

3.1.1 Das SUSAN-Prinzip

Sei I ein Eingangsbild. Um jedes Pixel im Bild legen wir eine Maske gelegt. Für unseren Zweck betrachten wir lediglich die Masken der Größe 3×3 beziehungsweise 7×7, wie in

Abbildung [x]. Dabei ist das gelbe Pixel der Mittelpunkt der Maske, die orangenen Pixel

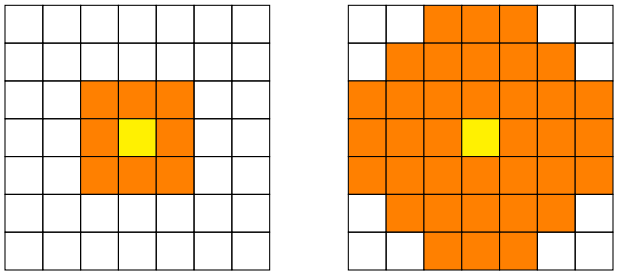

Abbildung 3.1: Die zwei Filtermasken. Links 3×3, rechts 7×7.

liegen in der Maske und die weißen Pixel außerhalb der Maske.

Das SUSAN-Prinzip funktioniert für die Anzahl von orangenen Pixeln g wie folgt: Für jedes Pixel r_0 in I (wobei $I(r_0)$ der Grauwert am Pixel r_0 ist), berechne die Antwort

$$A(r_0) = \max\{0, g - n(r_0)\}.$$

Dabei ist n definiert als

$$n(r_0) = \sum_r c_t(r, r_0),$$

wobei r die Pixel in der respektiven Maske sind und

$$c_t(r, r_0) = \exp\Big(- \Big(\frac{I(r) - I(r_0)}{t} \Big)^6 \Big)$$

eine Vergleichsfunktion für zwei Pixel ist. Statt der obigen Vergleichsfunktion kann auch die Näherung

$$c_t(r, r_0) \approx \begin{cases} 1 & \text{falls } |I(r) - I(r_0)| \leq t \\ 0 & \text{sonst} \end{cases}$$

verwendet werden. Als USAN bezeichnen wir genau diejenigen Pixel r aus der Maske, für die $c_t(r, r_0) > 0$. Jedes Pixel hat also seine eigene USAN.

Nehmen wir nun zum Beispiel ein Eingangsbild wie in Abbildung [x]. Hier wurde lediglich das SUSAN-Prinzip angewandt und danach wurde jeder Graustufenwert mit dem Faktor $\frac{255}{\max\{A(i,j)\}}$ multipliziert, um den Effekt hervorzuheben. Zunächst einmal finden wir, dass $A(i, j)$ genau da am stärksten ist, wo unser Eingangsbild eine Kante hat. Auch an den Rändern des Bilds findet das SUSAN-Prinzip eine Kante. Es besteht lediglich das Problem, dass um die Kante herum eine kleine Antwort dort ist, wo im originalen Bild nur die Nähe zu einer Kante besteht. Wir schließen daraus, dass die Lokalisation der Kanten zwar durchaus funktioniert, aber verbesserungswürdig ist.

255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255
255	255	255	255	255	0	255	255	255	255	255

147	40	40	40	147	255	147	40	40	40	147
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
40	0	0	0	40	201	40	0	0	0	40
147	40	40	40	147	255	147	40	40	40	147

Abbildung 3.2: Das SUSAN Prinzip (links Eingang, rechts Antwort)

3.1.2 Non-Maximum-Suppression

Das oben genannte Prinzip findet Kanten innerhalb von Bildern mit nur bedürftiger Genauigkeit. Im Bereich der eigentlichen Kante stellen wir fest, dass die Antwort A ungleich 0 ist. Um die Kante genauer zu lokalisieren, verwenden wir das Prinzip der Non-Maximum-Suppression, bei der nur die maximale Antwort entlang einer Kante erhalten bleibt. Zu diesem Zweck berechnen wir die Richtung eines jeden Pixels $r_0 = (x_0, y_0)$, für welches $A(x_0, y_0) \neq 0$ gilt, durch eine Fallunterscheidung. Wir unterscheiden dabei verschiedene Arten von Kanten. Das Prinzip der Fallunterscheidung besticht durch seine Einfachheit: Befinden sich in der USAN mehr Pixel als unsere Maske

1. **Inter-Pixel:**

   Falls die Größe der USAN die des Maskendurchmessers übersteigt und die Distanz zwischen $\mathrm{COG}(r_0)$ und r_0 größer als 1 Pixel ist, so ist die Richtung $D(r_0)$ gegeben durch

   $$D(r_0) = \begin{cases} \arctan\left(\frac{x_0 - \mathrm{COG}(x_0)}{y_0 - \mathrm{COG}(y_0)}\right) & \text{falls } \mathrm{COG}(y_0) \neq y_0 \\ \frac{\pi}{2} & \text{sonst} \end{cases},$$

   wobei

   $$\mathrm{COG}(r_0) := \frac{\sum_r r\, c(r, r_0)}{\sum_r c(r, r_0)}.$$

2. **Intra-Pixel:**

Andernfalls müssen wir die zweiten Momente der USAN folgendermaßen berechnen:

$$d_{x_0} := \sum_r (x - x_0)^2 \, c_t(r, r_0)$$
$$d_{y_0} := \sum_r (y - y_0)^2 \, c_t(r, r_0)$$
$$\sigma := \operatorname{sgn}\Big(\sum_r (x - x_0)\,(y - y_0)\, c_t(r, r_0) \Big)$$

Dabei ergibt sich die Kantenrichtung als

$$D(r_0) = \begin{cases} \sigma \arctan \frac{d_{y_0}}{d_{x_0}} & \text{falls } d_{x_0} \neq 0 \\ \frac{\pi}{2} & \text{sonst} \end{cases}$$

Falls allerdings $d_{x_0} = 0$, so ist $D(r_0) = \frac{\pi}{2}$

Die Berechnungen für die jeweiligen Kantenrichtungen an jedem Pixel werden in der Implementation gleichzeitig mit den Berechnungen für das SUSAN-Prinzip durchgeführt. Auf diese Weise müssen c_t-Werte an jeder Stelle nur einmal berechnet werden.

Die Richtung lohnt sich nur für diejenigen Pixel (i, j) zu bestimmen, für die $A(i, j) > 0$ gilt. Ist die Richtung der Kante bestimmt, so können wir die lokalen Maxima von A entlang der Richtung, die senkrecht zur Kantenrichtung steht, erhalten. Alles, was kein lokales Maximum entlang dieser Richtung ist, wird verworfen. Wir erhalten so ein neues Bild. In Kapitel 3.2 wird darauf eingegangen, wie genau dieser Prozess implementiert wurde.

3.1.3 Ausdünnen

Da viele digitale Bilder eingangs mit Rauschen behaftet sind, ist es manchmal hilfreich, einzelne Antworten zu entfernen und dafür andere hinzuzufügen. Zu diesem Zwecke empfiehlt [ref], dass man einige der Kanten ausdünnt. Dieser Vorgang wird genauer in [ref] beschrieben, wurde aber meinerseits modifiziert. Es wird erneut über das ganze Bild iteriert. Jedes Pixel (i, j) mit einer Antwort $A(i, j) > 0$ wird auf die Anzahl seiner direkten Nachbarn mit $A(x, y) > 0$ überprüft. Als direkte Nachbarschaft werden die acht nächsten Pixel bezeichnet, siehe dazu [grafik]. Angenommen, das Pixel hat...

- **0 Nachbarn:** Entferne die Antwort des Pixels.
- **1 Nachbar:** Überprüfe, ob in einer Reichweite von 3 Pixeln eine Linie mit der gleichen Richtung existiert. Falls ja, verbinde die Pixel miteinander.
- **2 Nachbarn:** Falls das Pixel adjazent zu einer diagonalen Linie ist, entferne es. Falls das Pixel außerhalb einer sonst horizontalen oder vertikalen Linie liegt, verschiebe die Antwort des Pixels in die Lücke der vertikalen oder horizontalen Linie

- **3 Nachbarn:** Falls die drei Nachbarn in einer Linie liegen, entferne die Antwort des Pixels.

3.2 Implementation

Für meine persönliche Implementation des SUSAN-Kantendetektors in Python 3 habe ich eine Reihe von ausgewählten Gütekriterien aus der Softwaretechnik herangezogen:

1. Funktionalität
2. Effizienz
3. Benutzbarkeit
4. Änderbarkeit

Grundsätzlich lässt sich die Struktur der Software folgendermaßen darstellen: ..... Es sei zu jedem Unterpunkt dieses Kapitels gesagt, dass sich die Implementation im Anhang [ref] befindet.

3.2.1 Masken

3.2.2 Vergleichsfunktion

Zu Gunsten der Effizienz habe ich für die Vergleichsfunktion eine Lookup-Tabelle implementiert, wie es auch in [quelle] nahgelegt wurde. Beim Initialisieren des `Susan`-Objekts wird ein Feld erzeugt. Das Ziel ist es, alle möglichen Werte, die c_t (siehe 3.1.1) annehmen kann, zu speichern. In unserem Fall ist c_t folgendermaßen definiert:

$$c_t(a,b) := \exp\left(-\left(\frac{I(a)-I(b)}{t}\right)^6\right)$$

Ein Pixel in einem 8-bit Graustufenbild kann $2^8 = 256$ mögliche Intensitäten annehmen, demnach braucht das Feld für die Differenz zweier solcher Graustufenintensitäten 512 Plätze. Regulär werden Felder entweder ab 0 oder 1 indiziert. In Python werden Felder ab 0 indiziert, allerdings bietet Python den Vorteil, dass man mit Index $-i$ das i-te Element von hinten abrufen kann. In der Implementation ist diese Lookup-Tabelle dank dieser speziellen Art der Indizierung einfach und elegant: Der Index $a-b$ des Feldes steht für $c_t(a,b)$.

3.2.3 Non-Maximum-Suppression

Die Non-Maximum-Suppression ist ein Vorgang, bei der jede Kante genauer lokalisiert wird. Entlang einer Kante soll immer nur die maximale Antwort erhalten bleiben. Um

die Non-Maximum-Suppression durchzuführen, wird zunächst für jedes Pixel (i, j) mit $A(i, j) > 0$ die Kantenrichtung $D(i, j)$ bestimmt. Die möglichen Kantenrichtungen werden dann für einen Zwischenschritt kategorisiert. Die Kategorien sind *negativ diagonal*, *vertikal*, *positiv diagonal* und *horizontal*. Die Kategorie bestimmt, welche zwei adjazenten Pixel C für die Non-Maximum-Suppression interessant sind.

Bedingung	**Kategorie**	**Adjazente** C
$D(i,j) \leq -\frac{3}{8}\pi$	negativ diagonal	$\{(i+1, j-1), \quad (i-1, j+1)\}$
$D(i,j) > -\frac{3}{8}\pi, \quad D(i,j) \leq -\frac{1}{8}\pi$	vertikal	$\{(i-1, j), \quad (i+1, j)\}$
$D(i,j) > -\frac{1}{8}\pi, \quad D(i,j) \leq \frac{1}{8}\pi$	positiv diagonal	$\{(i, j-1), \quad (i, j+1)\}$
$D(i,j) > -\frac{3}{8}\pi$	horizontal	$\{(i+1, j-1), \quad (i-1, j+1)\}$

Für jedes Pixel (i, j) wird nun überprüft, ob $(i, j) = \max(C \cup \{(i, j)\})$ gilt. Gilt es nicht, so wird $A(i, j)$ unterdrückt.

3.2.4 Parallelisierung

Das SUSAN-Prinzip sieht vor, die Antwort $A(i, j)$ und die Kantenrichtung $D(i, j)$ aus immer nur mithilfe von Pixeln aus der Maske auf (i, j) zu berechnen. An dieser Stelle kann die Berechnung von A und D parallelisiert werden. In meiner Implementation liefert das Paket `multiprocessing` die nötigen Werkzeuge, um die vorhandenen Ressourcen zusammenzuschließen und die Berechnung von A und D parallel abzuarbeiten.

Der Einfachheit halber habe ich das Bild nur der Höhe nach partitioniert. Eine Partitionierung $\mathcal{S}$ ist eine Menge von nichtnegativen ganzen Zahlen $\{S_1, S_2, ..., S_n, S_{n+1}\}$. Für alle $i \in \{1, 2, ..., n\}$ arbeitet der Job J_i die Partition $(S_i, S_{i+1}]_{\mathbb{Z}}$ ab.

Unter der stark vereinfachten Annahme, dass alle Kerne gleich viele Fließkommazahloperationen pro Zeiteinheit abarbeiten können und gleichgroße Partitionen etwa gleichviel Rechenzeit benötigen, gelte folgende Aussage: Eine Partitionierung $\mathcal{S}$ für die gilt $\#\mathcal{S} = n + 1$ ist genau dann optimal, wenn für alle Partitionen gilt

$$|\#(S_i, S_{i+1}]_{\mathbb{Z}} - \#(S_j, S_{j+1}]_{\mathbb{Z}}| \leq 1, \quad \forall i \neq j, \quad i, j \in \{1, 2, ..., n\}$$

Angenommen der Computer, auf dem die Software läuft, besitzt n Kerne, kann also n Jobs $J_1, J_2, ..., J_n$ gleichzeitig abarbeiten, und ein Bild der Höhe H wird eingegeben. Sei $k := H \bmod n$ und $z := \left\lfloor \frac{H}{n} \right\rfloor$. Dann ist die optimale Partitionierung $\mathcal{S}$ nach der Höhe gegeben durch:

$$
\begin{aligned}
S_1 &= 0 \\
S_2 &= S_1 + z + 1 \\
S_3 &= S_2 + z + 1 \\
&\vdots \\
S_{k-1} &= S_{k-2} + z + 1 \\
S_k &= S_{k-1} + z + 1 \\
S_{k+1} &= S_k + z \\
&\vdots \\
S_n &= S_{n-1} + z \\
S_{n+1} &= S_n + z = H
\end{aligned}
$$

Die Jobs $J_1, J_2, ..., J_n$ arbeiten das komplette Bild ab, denn es gilt

$$\bigcup_{i=1}^{n} (S_i, S_{i+1}]_{\mathbb{Z}} = (0, H]_{\mathbb{Z}} = [1, H]_{\mathbb{Z}}$$

3.2.5 Ausdünnen

Das Ausdünnen, welches in [ref] näher beschrieben wird,

3.3 Rauschunterdrückung

3.4 Numerische Experimente

3.5 Heuristik

Für diese Heuristik nehmen wir an, dass das Bild eingangs eine Bildfunktion $I : \mathbb{R} \to \mathbb{R}$ ist. Der SUSAN-Kantendetektor beruht auf folgendem Prinzip: Für einen Schwellwert t gibt es, abhängig von der spezifischen Stelle im Bild x_0 eine untere Grenze $x_0 + a(x_0)$ und eine obere Grenze $x - b(x_0)$ der USAN. Dann gilt an den Grenzen:

$$
\begin{aligned}
I(x_0 + a(x_0)) &= I(x_0) + t \\
I(x_0 - b(x_0)) &= I(x_0) - t.
\end{aligned}
$$

Das Prinzip des SUSAN-Kantendetektors lässt sich folgendermaßen formulieren: An genau den Stellen, an denen $n(x_0) = a(x_0) + b(x_0)$ ein Minimum hat, ist die Antwort $A(x_0) > 0$. Wir erhalten unter der Annahme, dass n an x_0 ein lokales Minimum erreicht:

$$n'(x_0) = a'(x_0) + b'(x_0) = 0$$

Die obigen Gleichungen können wir nach a und b umstellen. So erhalten wir

$$a(x_0) = x_0 - x(I(x_0) + t)$$
$$b(x_0) = x(I(x_0) - t) - x_0,$$

wobei x die Position ist. Damit ist $n(x_0) = x(I(x_0) + t) - x(I(x_0) - t)$ und daraus folgt

$$n'(x_0) = x'(I(x_0) + t) \cdot I'(x_0) - x'(I(x_0) - t) \cdot I'(x_0) = 0,$$

anders formuliert,

$$x'(I(x_0) + t) \cdot I'(x_0) = x'(I(x_0) - t) \cdot I'(x_0)$$

und damit

$$x'(I(x_0) + t) = x'(I(x_0) - t),$$

falls die Bildfunktion nicht konstant ist. Es folgt

$$(I(x_0) + t)' = (I(x_0) - t)'.$$

Lassen wir t gegen Null laufen, so finden wir

$$\lim_{t \to 0} (I(x_0) + t)' - (I(x_0) - t)' = I''(x_0) = 0.$$

Demnach treffen sich die maximale Antwort des SUSAN-Kantendetektors und die zweite Ableitung der Bildfunktion. Es sei dazu gesagt, dass jegliche Ableitungen nur zum Zwecke der Demonstration dienen. Im tatsächlichen SUSAN-Kantendetektor wird keine Ableitung auf dem diskreten Bild berechnet.