HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

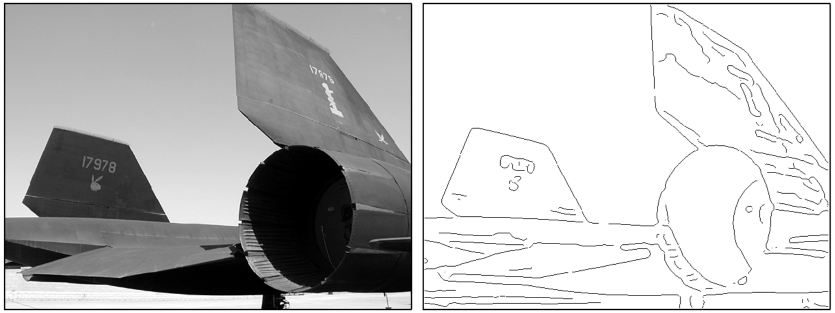
Computergrafik 2
Kantendetektion

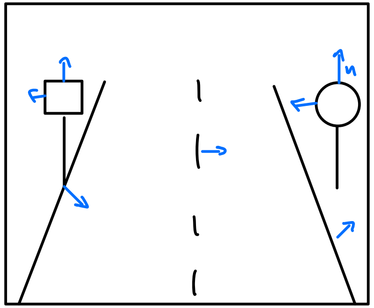
Vorlesung Bachelor Angewandte Informatik / Mediendesigninformatik
Wintersemester 2022/2023

Prof. Dr. Ingo Ginkel

Kantendetektion - Motivation

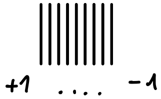
- Kanten spielen eine dominante Rolle im menschlichen Sehen: Bildinhalt ist bereits erkennbar, wenn nur wenige Konturen sichtbar sind (s. Karikaturen).
- Subjektiver Schärfeeindruck eines Bildes steht in direktem Zusammenhang mit seiner Kantenstruktur.
- Ein Bild kann (beinahe) vollständig aus Kanten rekonstruiert werden.
- Vorverarbeitungsschritt zum weiteren „Verstehen“ des Bildes.

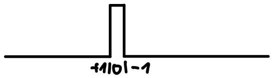

(a) (b)



  - **Konkretes Anwendungsbeispiel:** Finde Linienmarkierungen auf der Straße (Dashcam..), weitere Verarbeitung dann Interpretation der Bedeutung (Unterscheidung Straßenmarkierung vs. andere Kanten etc.), danach z.B. Selektion der relevanten Kanten für Spurhalteassistent, …

Ableitung





Kantendetektion - Grundlagen

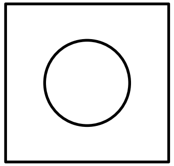
- Kanten sind Bildorte, an denen sich die Intensität auf kleinem Raum stark verändert.
- Die Intensitätsänderung bezogen auf die Bilddistanz wird durch die Ableitung der Bildintensität gemessen. In einer Dimension (z.B. entlang einer Bildzeile):

$$f'(u) = \frac{df(u)}{du}$$

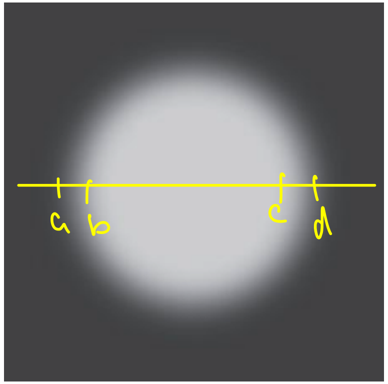
- Nicht notwendig sprunghaft, kann auch „steil“ ansteigen im Sinne eines kontinuierlichen Übergangs

dunkel->hell +

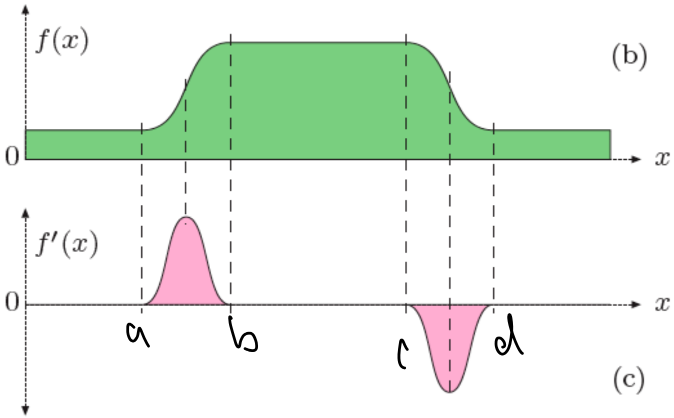
Ergebnis



?

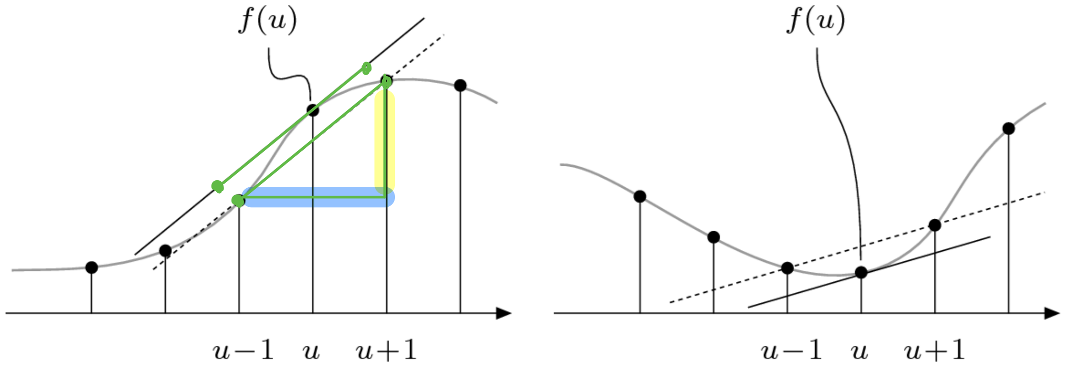


Hell->dunkel –



Kantendetektion - Grundlagen

- Für eine diskrete Funktion ist eine Ableitung nicht definiert
  - **Daher**: Näherung schätzen
  - **Wie**: Lege eine Gerade durch benachbarte Punkte und berechne die Steigung der Geraden



$$\frac{df}{dx}(u) \approx \frac{f(u+1) - f(u-1)}{(u+1) - (u-1)} = \frac{f(u+1) - f(u-1)}{2}.$$

Symmetrische Differenz

Kantendetektion - Grundlagen

$-127 < u < 127$

- **Achtung** beim Wertebereich: $\frac{df}{dx}(u) \approx \frac{f(u+1) - f(u-1)}{(u+1) - (u-1)} = \frac{f(u+1) - f(u-1)}{2}.$

  - $f(u) \in \{0, 1, 255\} \Rightarrow \frac{df}{dx}(u) \in \{-127, \ldots, 127\}$
  - Es wird also zwischen „fallender" und „steigender" Kante unterschieden.
  - Da dieser Wert pro Pixel berechnet wird, bietet es sich an ein Bild-Array für die geschätzten Ableitungen zu speichern
  - Dazu um +127 verschieben (damit es wieder in ein 1-Byte-unsigned-char passt), vor Berechnungen beim Lesen aus dem Bild die Verschiebung aber wieder rückgängig machen!

Senkrecht
Kanten findet es

- **Bisher**:
  - Kantenschätzung (d.h. hohe Ableitung finden) nur entlang einer Bildzeile möglich (d.h. senkrechte Kanten)

$\Rightarrow$ nur Zeilen oder Spalten analysieren

- **Benötigt:**
  - einen Mechanismus, der Zeilen- und Spaltenweise arbeitet
  - auch „schräge Kanten" und deren Ausrichtung (d.h. Winkel zur Koordinatenachse) sollen gefunden werden.

ein Wert als konstant sehen?

Kantendetektion - Grundlagen

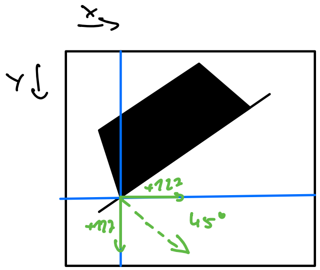
- Bild ist die diskrete Approximation einer 2-dimensionalen Funktion
  - Nutze Ableitung entlang einer der Koordinatenrichtungen (mathematisch: partielle Ableitung), d.h. verfolge die Intensitätsänderung entlang einer Zeile oder Spalte

$$I_x = \frac{\partial I}{\partial x}(u,v) \qquad \text{und} \qquad I_y = \frac{\partial I}{\partial y}(u,v)$$

- **Definition**: Den Vektor der partiellen Ableitungen bezeichnet man als Gradient.

$$\nabla I(u,v) = \underbrace{\begin{pmatrix} I_x(u,v) \\ I_y(u,v) \end{pmatrix}}_{\text{2D-Vektor}} = \begin{pmatrix} \frac{\partial I}{\partial x}(u,v) \\ \frac{\partial I}{\partial y}(u,v) \end{pmatrix}$$

(gesprochen: Nabla von I)



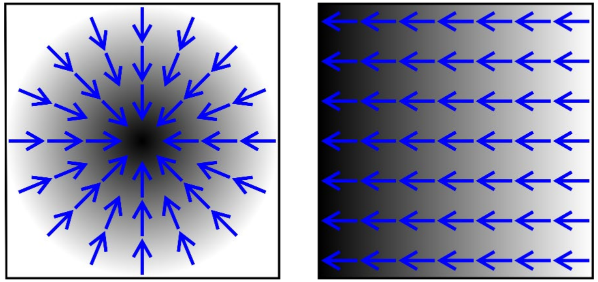
$I_x = +127$
$I_y = +127$

- **Geometrische Interpretation:**
  - Betrachtet man die Bildmatrix als Skalarfeld, so ist der Gradient an einem Punkt ein Vektor, der in Richtung des steilsten Anstieges des Skalarfeldes weist.

Kantendetektion - Grundlagen

Beispiel:

- In den Skalarfeldern rechts gibt der blaue Pfeil (=Gradient) die Richtung des steilsten Anstiegs an
- **Achtung**: anders als in der Bildverarbeitung üblich entspricht schwarz hohen Werten, weiß niedrigen Werten



Quelle: Wikipedia

- Der Betrag des Vektors entspricht der Stärke das Anstiegs.
- Der Betrag des Gradienten $|\nabla I| = \sqrt{I_x^2 + I_y^2},$ ist rotationsinvariant.
  - d.h. Er ist unabhängig von der Orientierung von Bildstrukturen.
  - Diese Eigenschaft ist für die richtungsunabhängige (isotrope) Lokalisierung von Kanten wichtig und daher ist |∇I| auch die Grundlage vieler praktischer Kantendetektoren.

Kantendetektion - Ableitungsfilter

- Realisierung der symmetrischen Differenzen als Filter

$$H_x^{\mathrm{D}} = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{und} \quad H_y^{\mathrm{D}} = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}.$$

Anmerkung: Den Gradienten selbst kann man nicht direkt als linearen Filter realisieren, da es sich um ein vektor-wertiges Ergebnis handelt.

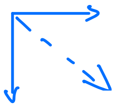
→ nicht 1 pixel

- **Beispiel**:

Intensität I   Ableitung I_x Ableitung I_y $|\nabla I|$

Betrag des Gradiente

anfällig für rauschen



- Skalierung I_x,I_y wie abgespeichert (d.h. $\frac{df}{dx}(u) = 0$ entspricht Grauwert 127)
- Skalierung $|\nabla I|$ so, dass Max- und Min-Wert genau auf Weiß bzw. Schwarz landen.

Einfache Kantenoperatoren – Prewitt-Operator

- **Prewitt**: Verwende Ableitungsfilter, gemittelt über 3 Zeilen bzw. Spalten

$$H_x^{\mathrm{P}} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{und} \quad H_y^{\mathrm{P}} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \qquad \nabla I(u,v) \approx \frac{1}{6} \begin{bmatrix} H_x^P * I \\ H_y^P * I \end{bmatrix}$$

  - Mittelung notwendig wegen Rauschanfälligkeit des einfachen Gradientenoperators in x bzw. y Richtung.

- Der Prewitt-Operator ist separabel: $H_x^{\mathrm{P}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{bzw.} \quad H_y^{\mathrm{P}} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix},$

  - d.h. Es wird eine (Box-)Glättung gerechnet und dann eine Ableitung geschätzt.

- **Bemerkung**: Aufgrund der Kommutativität der Faltung auch umgekehrt möglich, d. h. Glättung nach Berechnung der Ableitung (aber nur in der passenden Richtung!).

Einfache Kantenoperatoren – Sobel-Operator

- **Sobel**: Verwende Ableitungsfilter, gemittelt über 3 Zeilen bzw. Spalten mit stärkerer Gewichtung der mittleren Zeile bzw. Spalte.

$$H_x^{\mathrm{S}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{und} \quad H_y^{\mathrm{S}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

- Der Sobel-Operator ist ebenfalls separabel:

$$H_x^S = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{und} \quad H_y^S = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Gauß filter

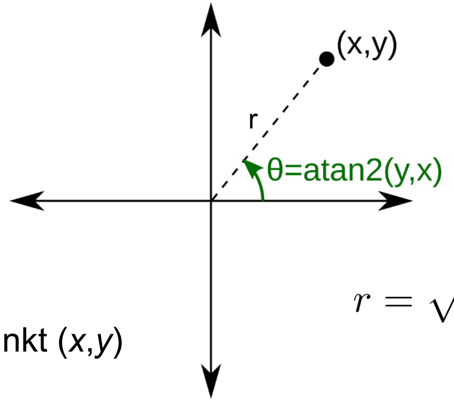
- Gradient: $\nabla I(u,v) \approx \frac{1}{8} \begin{bmatrix} H_x^S * I \\ H_y^S * I \end{bmatrix}$ (Normierung mit Summe der Beträge der Filterkoeffizienten)

Kantendetektion mit Sobel-Operator

- bezeichne, unabhängig ob Prewitt- oder Sobel-Filter, die skalierten Filterergebnisse (Gradientenwerte) mit $I_x = I * H_x \quad \text{und} \quad I_y = I * H_y.$

- Die Kantenstärke ist $E(u,v) = \sqrt{I_x^2(u,v) + I_y^2(u,v)}$
- Kantenrichtung:
  - Falls x>0:

$$\cos\theta = \frac{x}{r},\ \sin\theta = \frac{y}{r} \quad \Rightarrow \frac{y}{x} = \frac{r\cdot\sin\theta}{r\cdot\cos\theta} = \tan\theta \Rightarrow \theta = \tan^{-1}(\frac{y}{x})$$

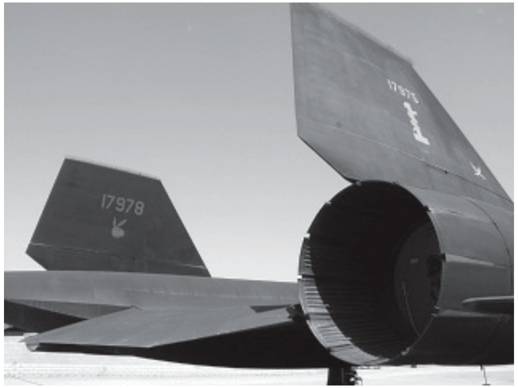


$$r = \sqrt{x^2 + y^2}$$

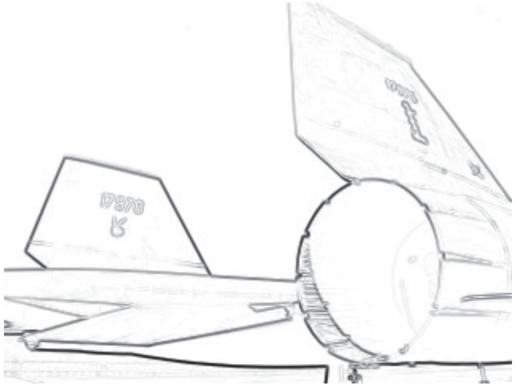
- Funktioniert so wie oben nur für x>0, allgemeiner:
  - Funktion **`atan2(y,x)`** beschreibt den Winkel θ zwischen dem Strahl zum Punkt (x,y) und der positive *x-Achse*, begrenzt auf das Intervall (-π, π].
  - Gedrehte Reihenfolge der Argumente relevant!
  - Ergebnis von **`atan2(y,x)`** dann auf 0,..360 Grad verschieben und skalieren
  - atan2 in jeder gängigen Programmiersprache verfügbar
  - Warum und wieso das den Winkel berechnet: `https://en.wikipedia.org/wiki/Atan2`

Kantendetektion mit Sobel-Operator

- **Beispiel**:



Originalbild



Kantenstärke - invertiert codiert
(schwarz = hoher Betrag des Gradienten)

Kantenrichtung
0° ... 360° in Farbe codiert,
Farbsättigung = Kantenstärke

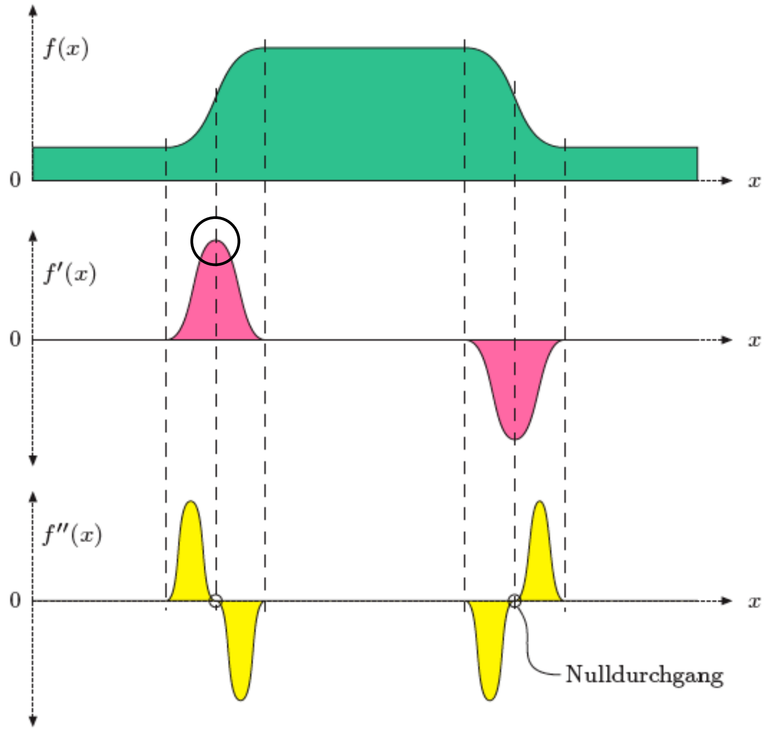
- Das Ergebnis des Kantenwinkels ist vorzeichenbehaftet, d.h. die markierten parallelen Kanten haben unterschiedliche Farben, da einmal “steigend” und einmal “fallend”
- farbliche Darstellung der Kantenorientierung eher unüblich, da visuell schwierig zu interpretieren

Kantendetektion mit der zweiten Ableitung

- Die bisherigen Kantenoperatoren messen nur die erste Ableitung.
- Problematisch sind dabei Kanten mit einem langsamen Helligkeitswechsel, die sich damit nicht genau lokalisieren lassen.

- **Alternative**: Bestimmung des Nulldurchgangs der zweiten Ableitung.
- Da die zweite Ableitung noch empfindlicher gegen Rauschen ist, muss das Bild gleichzeitig geglättet werden.

- **Gesucht**: 2D Version für Bilder



Kantendetektion mit Laplace Operator

∂^2_{xx} = 2x in "x" Ableit.

- Der Laplace-Operator ist definiert als Summe der zweiten partiellen Ableitungen

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y) = \partial^2_{xx} f(x,y) + \partial^2_{yy} f(x,y)$$

Vorwärts differenz

- Diskrete Näherung:
  - Verwende hier die einfache (nicht symmetrische) Vorwärtsdifferenz für Ableitungen: $\frac{df(u)}{du} \approx f(u+1) - f(u)$
  - Zweite Ableitung ist dann eine Differenz von Differenzen

Rückwärts differenz

$$\frac{d^2 f(u)}{du^2} \approx \frac{df(u)}{du} - \frac{df(u-1)}{du} \approx f(u+1) - f(u) - (f(u) - f(u-1)) = f(u+1) - 2f(u) + f(u-1)$$

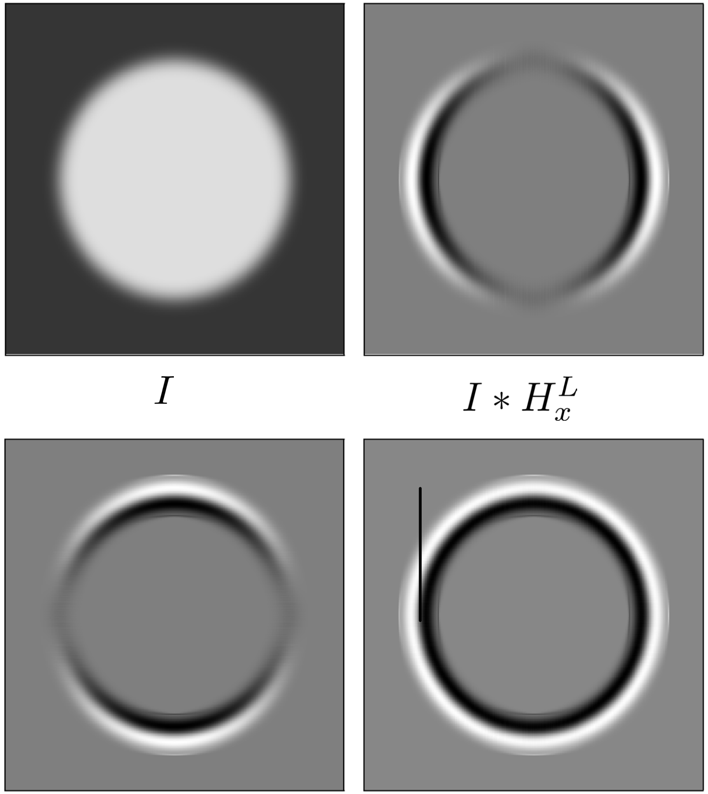
  - Als Filter: $\partial^2_{xx} \approx H^L_x = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$ und $\partial^2_{yy} \approx H^L_y = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$
  - Addiert ergibt sich der zweidimensionale Laplace Filter: $H^L = H^L_x + H^L_y = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
  - Nicht separabel aber effizienter berechenbar als: $I * H^L = I * (H^L_x + H^L_y) = I * H^L_x + I * H^L_y$

Kantendetektion mit Laplace Operator

Beispiel: Anwendung auf Testbild

- Nulldurchgang markiert genaue Kantenposition.
- Trotz der durch die kleinen Filterkerne ziemlich groben Schätzung der Ableitungen ist das Ergebnis fast perfekt isotrop.
- Summe der Koeffizienten ist null, so dass sich in Bildbereichen mit konstanter Intensität die Filterantwort null ergibt (wie bei Gradientenfiltern)
- Weitere gebräuchliche Varianten von 3 × 3 Laplace-Filtern sind:

$$H_8^L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{oder} \quad H_{12}^L = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

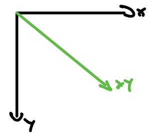


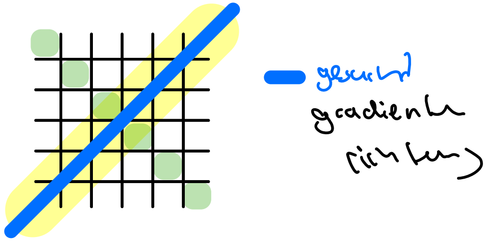
I $\quad$ $I * H_x^L$

$I * H_y^L$ $\quad$ $I * H_x^L + I * H_y^L$

Kanten-Detektion: Canny Algorithmus

- Der von Canny 1986 vorgestellte Kantenoperator ist ein sehr verbreitetes Verfahren und gilt auch heute noch als „State of the Art“.
  - Z.B. in der Bibliothek OpenCV vorhanden.





- **Ziele**
  - Gute Detektion: möglichst alle Kanten detektieren, ohne zu viel Clutter.
  - Gute Lokalisation: minimale Distanz zwischen detektierter und echter Kante
  - Klare Antwort: nur eine Antwort pro Kante
- Der optimale Filter wurde von Canny durch Variationsrechnung abgeleitet.
- In seiner vollständigen Form verwendet der Operator einen Satz von (relativ großen) gerichteten Filtern auf mehreren Auflösungsebenen und fügt die Ergebnisse der verschiedenen Skalenebenen in ein gemeinsames Kantenbild („edge map“) zusammen.
- Meistens wird der Algorithmus allerdings nur im „single-scale“-Modus verwendet, wobei aber bereits damit (bei passender Einstellung des Glättungsradius σ) eine gegenüber einfachen Operatoren deutlich verbesserte Kantendetektion festzustellen ist.
- **Hier**: nur eine Auflösungsstufe.

Kanten-Detektion: Canny Algorithmus

Algorithmus in 3 Arbeitsphasen:

1. Vorverarbeitung: Das Eingangsbild wird mit einem Gaußfilter der Breite σ geglättet. Aus dem geglätteten Bild wird für jede Position der x/y-Gradient berechnet sowie dessen Betrag und Richtung.

- **Zwischenergebnis nach 1**: über mehrere Pixelbreiten steigende Intensitäten erzeugen mehrere Pixel nebeneinander als Kandidaten für die Kante

2. Kantenlokalisierung: Als Kantenpunkte werden jene Positionen markiert, an denen der Betrag des Gradienten ein lokales Maximum entlang der zugehörigen Gradientenrichtung aufweist.

- **Zwischenergebnis nach 2**: „Dicke" der Kanten damit beseitigt, bleiben zu viele Pixel, die zwar Intensitätsänderung haben, aber nicht Teil eines längeren Kantenzuges sind.

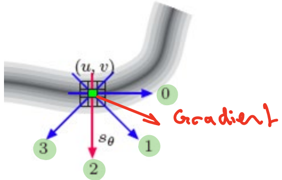
3. Kantenselektion und -verfolgung: Im abschließenden Schritt werden unter Verwendung eines Hysterese-Schwellwerts zusammenhängende Ketten von Kantenelementen gebildet.

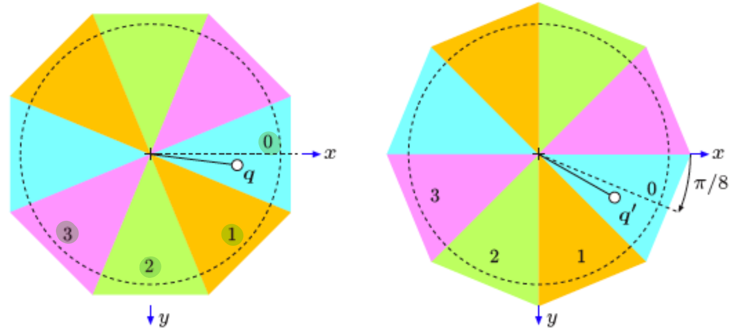
- **Aufgabe**: Finde also rekursiv diejenigen Pixel, deren Nachbarpixel auch Kantenkandidaten sind

Kanten-Detektion: Canny Algorithmus

Ziel: „Kantendicke schrumpfen“

- Gradientenrichtung gibt Richtung quer zur Kante an
- Die Gradientenrichtung an der Position (u, v) wird grob in vier diskrete Winkel $s_\theta \in \{0, 1, 2, 3\}$ quantisiert (a).
  - **Grund**: einfache Navigation zu den Nachbarn mit Indices!





(a) (b)

Konstante

GetOrientationSector(d_x, d_y)
Returns the orientation sector s_θ for the 2D vector $(d_x, d_y)^\intercal$. See Fig. 6.12 for an illustration.

$\begin{pmatrix} d'_x \\ d'_y \end{pmatrix} \leftarrow \begin{pmatrix} \cos(\pi/8) & -\sin(\pi/8) \\ \sin(\pi/8) & \cos(\pi/8) \end{pmatrix} \cdot \begin{pmatrix} d_x \\ d_y \end{pmatrix}$ ▷ rotate $\begin{pmatrix} d_x \\ d_y \end{pmatrix}$ by $\pi/8$

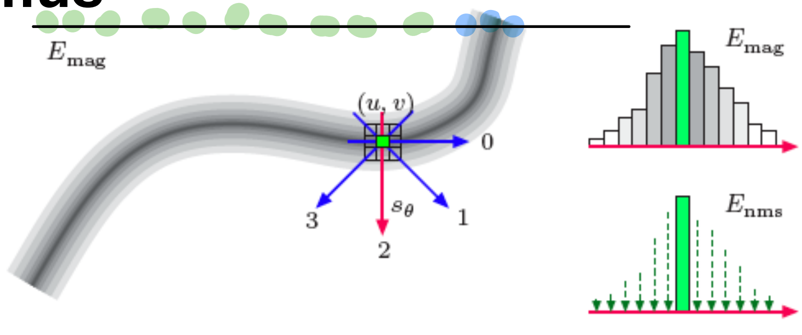
if $d'_y < 0$ **then**
  $d'_x \leftarrow -d'_x, \quad d'_y \leftarrow -d'_y$ ▷ mirror to octants $0, \ldots, 3$

$$s_\theta \leftarrow \begin{cases} 0 & \text{if } (d'_x \geq 0) \wedge (d'_x \geq d'_y) \\ 1 & \text{if } (d'_x \geq 0) \wedge (d'_x < d'_y) \\ 2 & \text{if } (d'_x < 0) \wedge (-d'_x < d'_y) \\ 3 & \text{if } (d'_x < 0) \wedge (-d'_x \geq d'_y) \end{cases}$$

return s_θ. ▷ sector index $s_\theta \in \{0, 1, 2, 3\}$

- Winkelberechnung des Gradienten mit x-Achse aufwändig
- Daher: q um π/8 nach q' rotieren,
- Dann performante Bestimmung des Oktanten mit Vergleichen und logischer Verknüpfung (keine Winkelberechnung!)
- Richtungsvektoren in den anderen Oktanten werden in die Oktanten s_θ = 0, 1, 2, 3 gespiegelt.

Kanten-Detektion: Canny Algorithmus



(a) (b)

Nachdem diese Richtung gefunden ist:

- Als mögliche Kantenpunkte werden nur jene Elemente betrachtet, an denen das (eindimensionale) Kantenprofil in der Richtung s_θ ein lokales Maximum ist (b).
- Betrachte dazu die Indexnachbarn in der diskreten Gradienten-Richtung
- Die Kantenstärke aller anderen Elemente wird auf Null gesetzt („suppressed").

Speichern:

init I - Bild
" I_x - Bild copy mit x Ableitungen
" I_y - Bild copy mit y Ableitungen
E_{mag} - " größe des Gradienten

Bild bool(E_{nms}
^ is local max

- Damit Schritt 2 des Algorithmus abgearbeitet

IsLocalMax$(E_{mag}, u, v, s_\theta, t_{lo})$
Determines if the gradient magnitude E_{mag} is a local maximum at position (u, v) in direction $s_\theta \in \{0, 1, 2, 3\}$.

$m_C \leftarrow E_{mag}(u, v)$
if $m_C < t_{lo}$ **then**
  return false
else

$$(m_L, m_R) \leftarrow \begin{cases} (E_{mag}(u-1, v), E_{mag}(u+1, v)) & \text{if } s_\theta = 0 \\ (E_{mag}(u-1, v-1), E_{mag}(u+1, v+1)) & \text{if } s_\theta = 1 \\ (E_{mag}(u, v-1), E_{mag}(u, v+1)) & \text{if } s_\theta = 2 \\ (E_{mag}(u-1, v+1), E_{mag}(u+1, v-1)) & \text{if } s_\theta = 3 \end{cases}$$

  return $(m_L \le m_C) \wedge (m_C \ge m_R)$.

Kanten-Detektion: Canny Algorithmus

Wird nachher die zu Kante gehören können?

Kantenverfolgung mit Hysterese-Schwellwert

- benachbarte Kantenpunkte, die in der vorherigen Operation als lokale Maxima verblieben sind, zu zusammenhängenden Folgen verketten.
- Dazu wird eine Schwellwertoperation mit Hysterese verwendet, mit zwei unterschiedlichen Schwellwerten t_{hi} und t_{lo}, (wobei $t_{hi} > t_{lo}$).
- Das Bild wird nach Elementen mit Kantenstärke $\mathbf{E}_{nms}(u, v) \geq t_{hi}$ durchsucht,
- Sobald ein solches (bisher nicht besuchtes) Pixel gefunden ist, wird eine neuer Kantenfolge angelegt und alle zusammenhängenden Positionen (u', v') angefügt, solange $\mathbf{E}_{nms}(u', v') \geq t_{lo}$.
- Dadurch entstehen nur Kantenfolgen, die zumindest ein Element mit einer Kantenstärke größer als t_{hi} aufweisen und keinen Kantenpunkt mit Kantenstärke unter t_{lo}.

TraceAndThreshold$(E_{\mathrm{nms}}, E_{\mathrm{bin}}, u_0, v_0, \mathsf{t}_{\mathrm{lo}})$
Recursively collects and marks all pixels of an edge that are 8-connected to (u_0, v_0) and have a gradient magnitude above t_{lo}.
$E_{\mathrm{bin}}(u_0, v_0) \leftarrow 1$ ▷ mark (u_0, v_0) as an edge pixel
$u_{\mathrm{L}} \leftarrow \max(u_0 - 1, 0)$ ▷ limit to image bounds
$u_{\mathrm{R}} \leftarrow \min(u_0 + 1, M - 1)$
$v_{\mathrm{T}} \leftarrow \max(v_0 - 1, 0)$
$v_{\mathrm{B}} \leftarrow \min(v_0 + 1, N - 1)$
for $u \leftarrow u_{\mathrm{L}}, \ldots, u_{\mathrm{R}}$ **do**
  for $v \leftarrow v_{\mathrm{T}}, \ldots, v_{\mathrm{B}}$ **do**
    if $(E_{\mathrm{nms}}(u, v) \geq \mathsf{t}_{\mathrm{lo}}) \wedge (E_{\mathrm{bin}}(u, v) = 0)$ **then**
      TraceAndThreshold$(E_{\mathrm{nms}}, E_{\mathrm{bin}}, u, v, \mathsf{t}_{\mathrm{lo}})$
return

rekursion? Rekursiv

Kanten-Detektion: Canny-Algorithmus

Gesamt-Algorithmus:

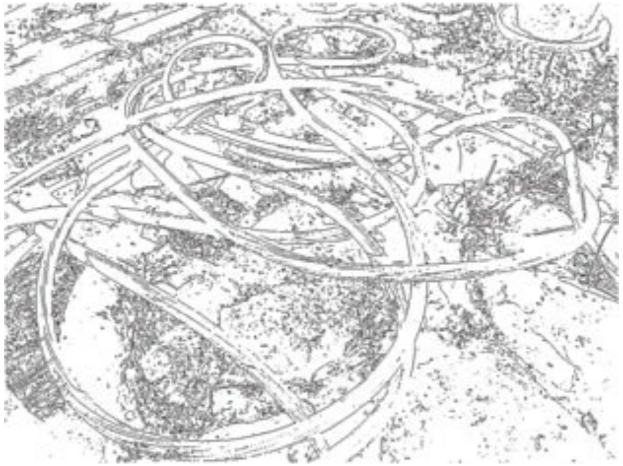
CannyEdgeDetector$(I, \sigma, \mathrm{t_{hi}}, \mathrm{t_{lo}})$
Input: I, a grayscale image of size $M \times N$;
σ, scale (radius of Gaussian filter $H^{G,\sigma}$);
$\mathrm{t_{hi}}$, $\mathrm{t_{lo}}$, hysteresis thresholds ($\mathrm{t_{hi}} > \mathrm{t_{lo}}$).
Returns a binary edge image of size $M \times N$.

$\bar{I} \leftarrow I * H^{G,\sigma}$ ▷ blur with Gaussian of width σ
$\bar{I}_x \leftarrow \bar{I} * [-0.5 \;\; \mathbf{0} \;\; 0.5]$ ▷ x-gradient
$\bar{I}_y \leftarrow \bar{I} * [-0.5 \;\; \mathbf{0} \;\; 0.5]^\intercal$ ▷ y-gradient
$(M, N) \leftarrow \mathsf{Size}(I)$
Create maps:
  $E_{\mathrm{mag}} : M \times N \mapsto \mathbb{R}$ ▷ gradient magnitude
  $E_{\mathrm{nms}} : M \times N \mapsto \mathbb{R}$ ▷ maximum magnitude
  $E_{\mathrm{bin}} : M \times N \mapsto \{0, 1\}$ ▷ binary edge pixels
for all image coordinates $(u, v) \in M \times N$ **do**
  $E_{\mathrm{mag}}(u, v) \leftarrow \left[\bar{I}_x^2(u, v) + \bar{I}_y^2(u, v)\right]^{1/2}$
  $E_{\mathrm{nms}}(u, v) \leftarrow 0$
  $E_{\mathrm{bin}}(u, v) \leftarrow 0$
for $u \leftarrow 1, \ldots, M-2$ **do**
  for $v \leftarrow 1, \ldots, N-2$ **do**
    $d_x \leftarrow \bar{I}_x(u, v), \quad d_y \leftarrow \bar{I}_y(u, v)$
    $s_\theta \leftarrow \mathsf{GetOrientationSector}(d_x, d_y)$ ▷ Alg. 6.2
    if $\mathsf{IsLocalMax}(E_{\mathrm{mag}}, u, v, s_\theta, \mathrm{t_{lo}})$ **then** ▷ Alg. 6.2
      $E_{\mathrm{nms}}(u, v) \leftarrow E_{\mathrm{mag}}(u, v)$ ▷ only keep local maxima
for $u \leftarrow 1, \ldots, M-2$ **do**
  for $v \leftarrow 1, \ldots, N-2$ **do**
    if $(E_{\mathrm{nms}}(u, v) \geq \mathrm{t_{hi}}) \wedge (E_{\mathrm{bin}}(u, v) = 0)$ **then**
      $\mathsf{TraceAndThreshold}(E_{\mathrm{nms}}, E_{\mathrm{bin}}, u, v, \mathrm{t_{lo}})$ ▷ Alg. 6.2
return E_{bin}.

Kanten-Detektion: Canny-Algorithmus

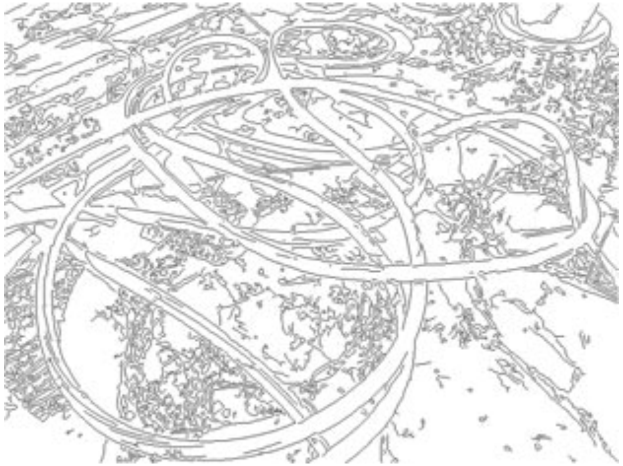
Vergleich Canny-Algorithmus vs. Sobel-Operator

Original



Sobel-Operator

- zu viel information
- nicht erkennen der relevanten Kanten



Canny-Algorithmus

besser

- Anstiegs bereich in der Mitte der Kante
- 1 pixel breit

* Sprung Kaus 5

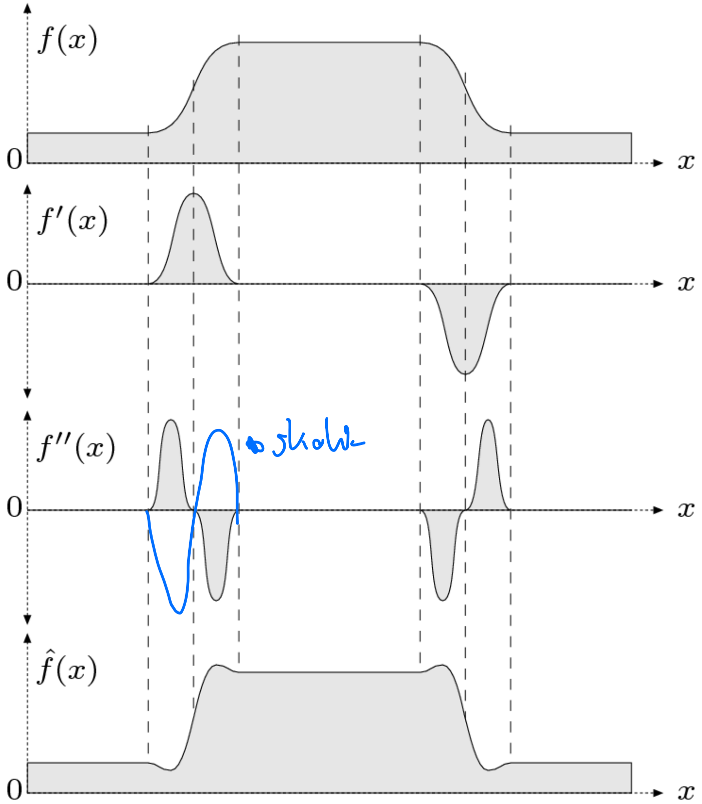
Kantenschärfung mit Laplace Filter

Grundidee: Überhöhung der Kanten durch Subtraktion der zweiten Ableitung lässt das Bild schärfer erscheinen.

- Vorgehen:

$$I' = I - wH^L * I$$

  w bestimmt die Stärke der Schärfung.



Bemerkungen:

- Es handelt sich im Wesentlichen um einen künstlich erzeugten Mach-Band-Effekt!
- Schärfung verstärkt auch das Bildrauschen.
- Sinnvoll nur mit eher kleinen Werten für w.

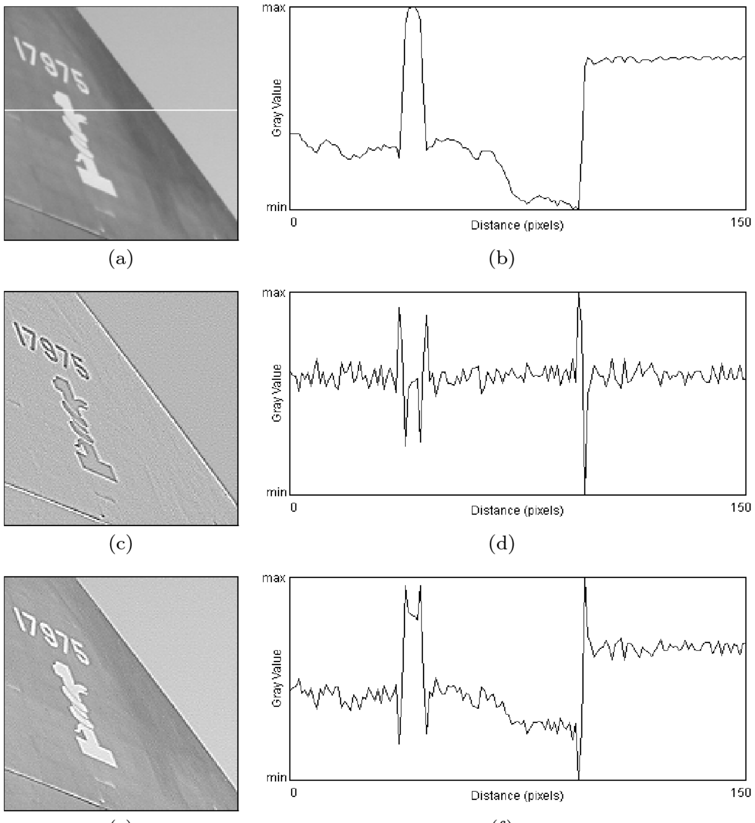
Kantenschärfung mit Laplace Filter

Beispiel: einfache Kantenschärfung mit dem Laplace-Filter.

- Originalbild und Profil der markierten Bildzeile
  - Abbildungen (a) und (b)
- Ergebnis des Laplace-Filters H^L,
  - Abbildungen (c) und (d)

f''

- geschärftes Bild mit Schärfungsfaktor w = 1.0
  - Abbildungen (e) und (f)
- **sichtbar**: prinzipieller Effekt der Schärfung von Kanten
- **aber auch**: verstärktes Rauschen in Bereichen ohne Kanten.



(a) (b) (c) (d) (e) (f)

mehr rauschen durch skalieren

Kantenschärfung mit Unsharp masking (USM)

Verallgemeinerung der Schärfung:

1. Erzeugung einer geglätteten Version des Bildes (z.B. mit einem Gaußfilter oder einer anderen Glättung *H*)

2. Subtraktion der geglätteten Version vom Originalbild:

$M = I - I * H$ (Ergebnis heißt Maske)

3. Addition der gewichteten Maske zum Originalbild

$$\begin{aligned} I' &= I + aM \\ &= (1+a)I - aI * H \end{aligned}$$

- a kontrolliert den Schärfungsgrad (a in [0.2; 4]),
- die Breite σ des Gaußfilters die Rauschempfindlichkeit.

Zusammenhang Laplace-Filter und USM

- Laplace:

$$H^{\mathrm{L}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} - 5 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = 5 \cdot (\tilde{H}^{\mathrm{L}} - \delta),$$

- Damit entspricht die Laplace Schärfung

$$\begin{aligned} I' &= I - wH^L * I \\ &= I - 5w(\hat{H}^L * I - I) \\ &= I + 5w(I - \hat{H}^L * I) \\ &= I + 5wM \end{aligned}$$

einer USM Schärfung mit

$$\hat{H}^L = \frac{1}{5} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{und} \quad a = 5w$$

Kantenschärfung mit Unsharp masking (USM)

Schärfung abhängig von lokalem Bildkontrast

- Oft zusätzlich Mindestwert für den lokalen Bildkontrast, ab dem eine Schärfung vorgenommen wird.
- typischerweise gemessen durch den Betrag des Gradienten |∇I|, ab dem eine Schärfung an der Stelle (u, v) stattfindet.

$$\check{I}(u,v) \leftarrow \begin{cases} I(u,v) + a \cdot M(u,v) & \text{für } |\nabla I|(u,v) \geq t_c, \\ I(u,v) & \text{sonst.} \end{cases}$$

- **Effekt**: ohnehin eher scharfe Kanten werden noch schärfer, weichere Kanten bzw. Farbverläufe bleiben ungeschärft.

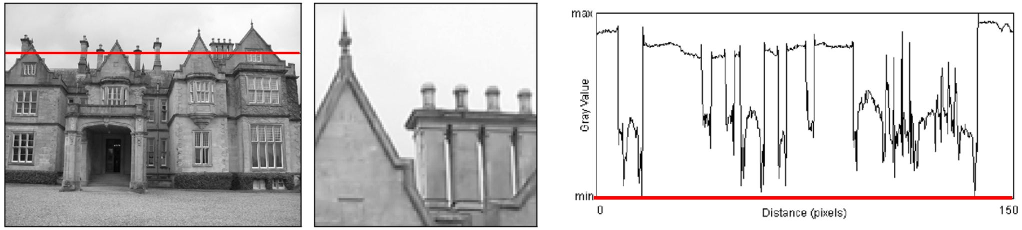
Kantenschärfung mit Unsharp masking (USM)

Beispiel

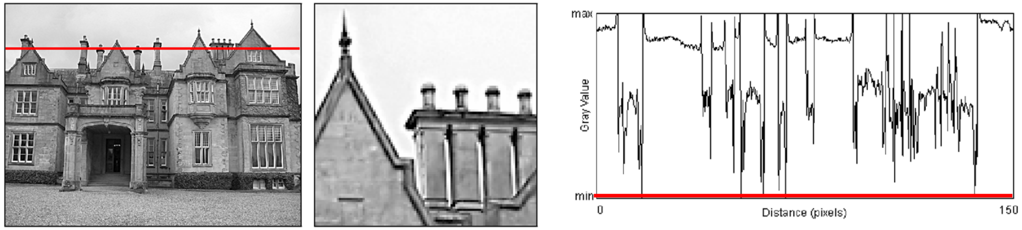
- Originalbild (obere Reihe)
- USM-Filter für unterschiedliche Radien σ = 2.5 und 10.0. (mittlere und unterer Reihe)
- Der Wert des Parameters a (Stärke der Schärfung) ist 100%.

Vorteil USM gegenüber Laplace:

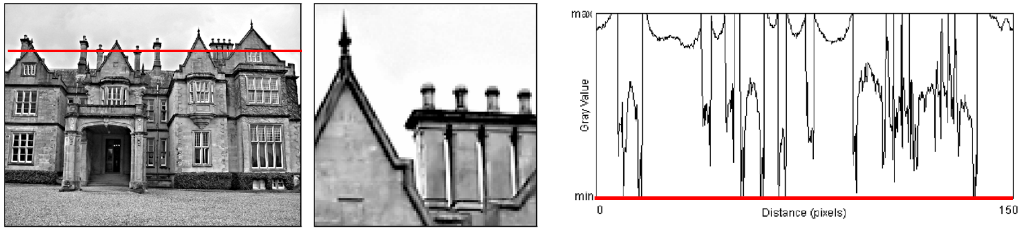
- Geringere Rauschanfälligkeit wegen Glättung
- Einstellbarkeit durch Parameter



Original



$\sigma = 2.5$



$\sigma = 10.0$

Kantendetektion - Fazit

Kantendetektion über geschätzte erste bzw. zweite Ableitungen

- Nutzung der Filter-Algorithmik
- Problem ist generell die Rauschanfälligkeit der Verfahren
  - Größerer Intensitätsunterschied könnte Rauschen aber auch Kante sein

Generell:

- bisherige Verfahren detektieren Kanten auf „Pro-Pixel-Basis" wie z.B. bei Sobel
- Oder auf Pixel-Ketten wie bei Canny

Problem:

- Keine Informationen vorhanden über ganze Objekte die dann aus Pixeln approximiert werden
- Pixel-Fehler bedeuten immer Unterbrechung der Kante, daher Objekt Rekonstruktion aus Pixeln (wie bei Canny rudimentär versucht) immer problematisch ohne Kenntnisse über die zu suchende Form.

Reale Anforderungen:

- Praxis-Aufgaben meist spezieller als „Finde Helligkeitsunterschiede", eher „Finde Objekt xy"