

Computer Vision

$$\nabla I(u, v) = \left[\frac{\frac{\partial I(u, v)}{\partial u}}{\frac{\partial I(u, v)}{\partial v}} \right] = \left(\frac{\partial I(u, v)}{\partial u}, \frac{\partial I(u, v)}{\partial v} \right)$$

Vorlesung 4:

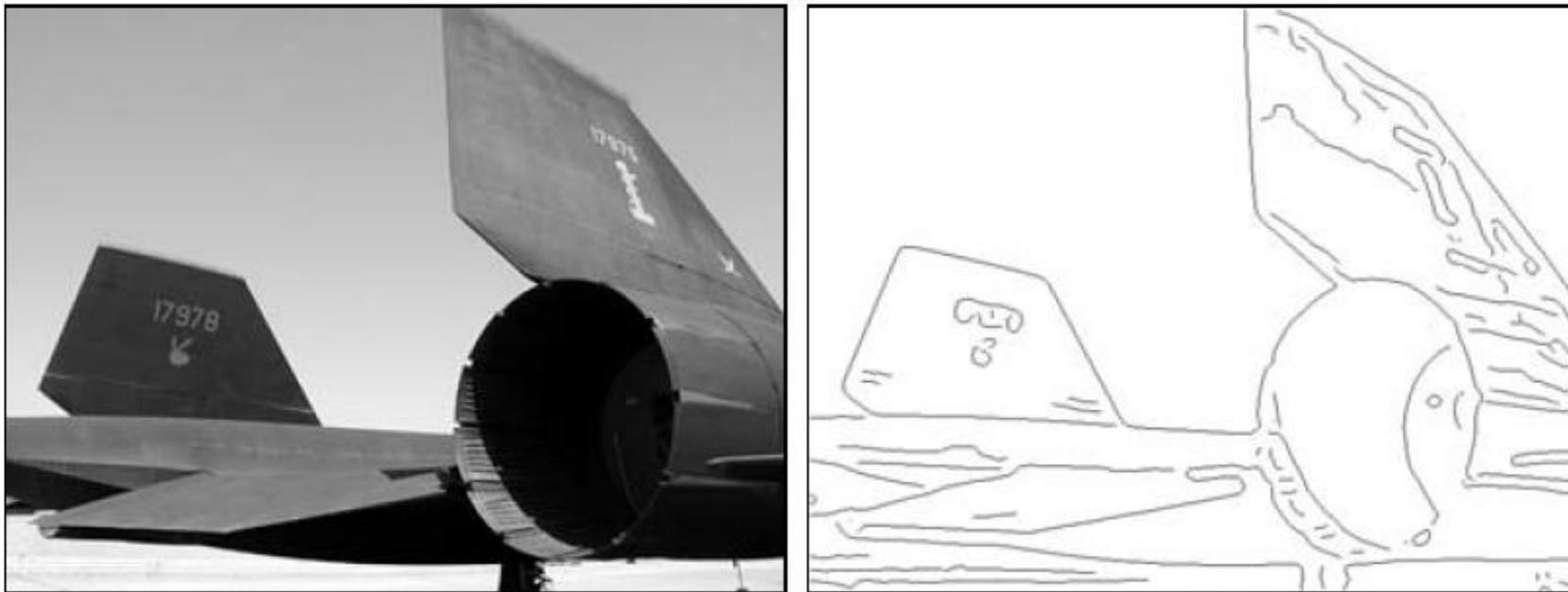
Merkmalserkennung (Kanten, einfache Kurven)

Dr. Xiao Zhao

Bildmerkmale

Kantenerkennung

Kantendetektion – Definition Kante

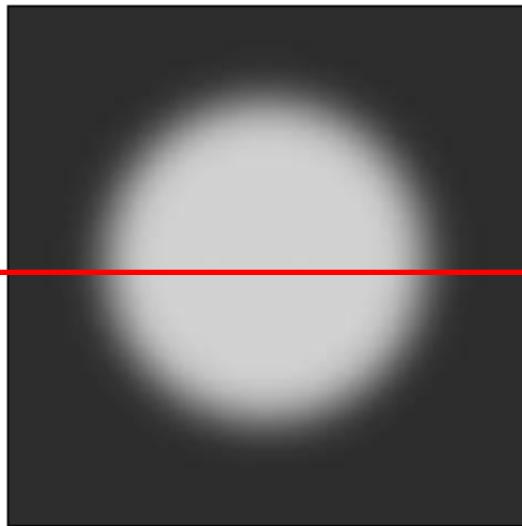


■ Kanten in Grauwertbildern

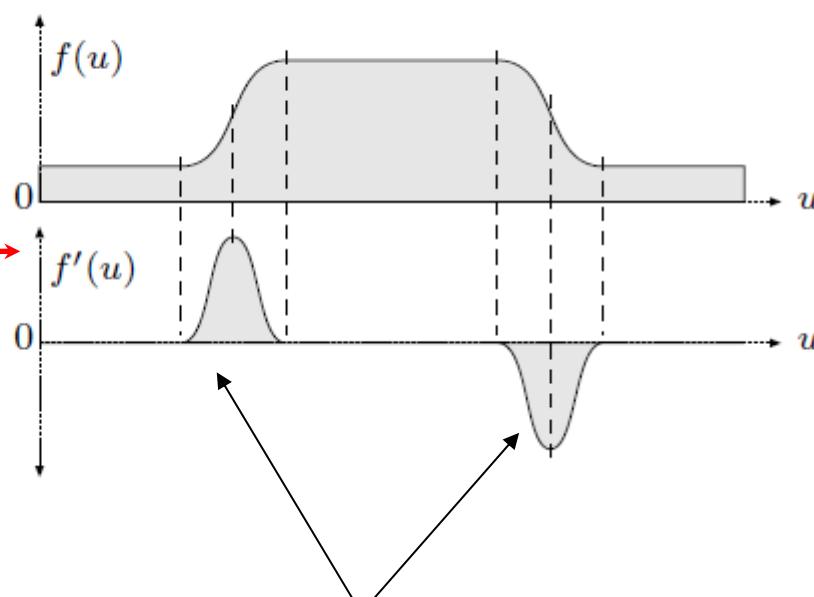
- Sind Bildbereiche mit **abrupten Übergängen** von Hell und Dunkel
- Treten für gewöhnlich an Objektgrenzen auf
- Können durch Schatten oder Textur erzeugt werden
- Sind unabhängig von der Helligkeit
- Spielen eine wichtige Rolle im visuellen Cortex

Kanten in Grauwertbildern

Das Bild



Intensität und Ableitung

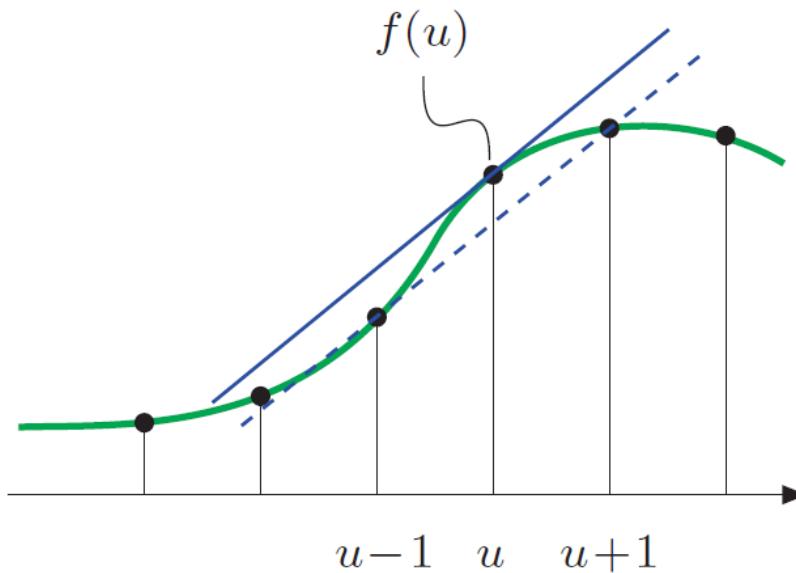


Kanten

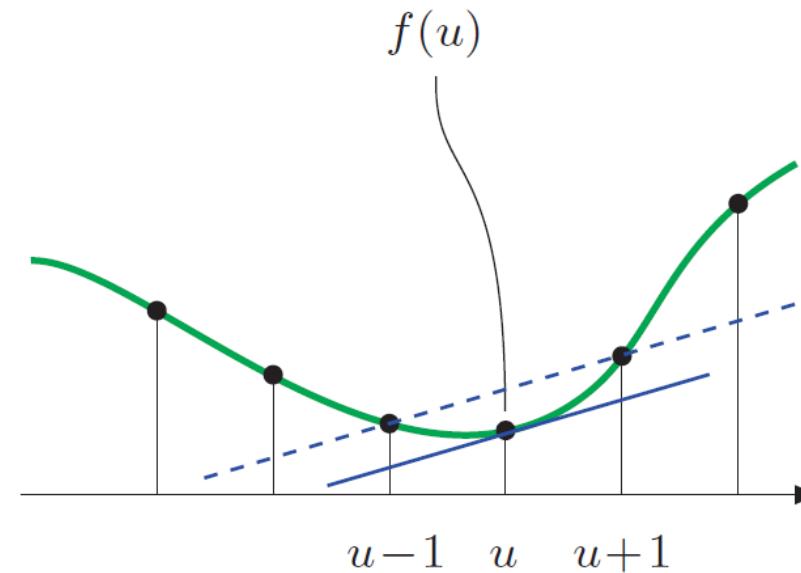
- Die grauen Werte an der roten Linie als eine kontinuierliche Funktion $f(u)$
- Erste Ableitung im 1D Fall.

Kantendetektion

- Eine Kante ist eine **schnelle Änderung** der betrachteten Grauwerte
- Änderung wird durch die Ableitung einer Funktion berechnet
- Für den diskreten eindimensionalen Fall gilt



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



$$\frac{df(u)}{du} \approx f(u+1) - f(u)$$

Kantendetektion

- Je größer die Ableitung, desto stärker die Änderung, desto wahrscheinlicher eine Kante
- Für das Bild \mathbf{I} liegt eine zweidimensionale Funktion vor, weswegen die partiellen Ableitungen benötigt werden:

$$\frac{\partial I(u, v)}{\partial u} \quad \frac{\partial I(u, v)}{\partial v}$$

- Diese bilden den *Gradientenvektor* oder *Gradient* an der Stelle (u, v):

$$\nabla I(u, v) = \begin{bmatrix} \frac{\partial I(u, v)}{\partial u} \\ \frac{\partial I(u, v)}{\partial v} \end{bmatrix} = \left(\frac{\partial I(u, v)}{\partial u}, \frac{\partial I(u, v)}{\partial v} \right)$$

- Mit dem Betrag $|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial u}\right)^2 + \left(\frac{\partial I}{\partial v}\right)^2}$ und Richtung $\theta = \arctan\left(\frac{\frac{\partial I}{\partial v}}{\frac{\partial I}{\partial u}}\right)$

Kanten und Faltungsmasken

- Die diskrete Approximation der Ableitung ist die Differenz

$$\frac{\partial I(u, v)}{\partial u} \approx \frac{\Delta I(u, v)}{\Delta u} = \frac{I(u+1, v) - I(u-1, v)}{2}$$

- Der Differenzenquotient kann als Faltung mit Filtermasken implementiert werden
- In horizontale Richtung gilt:

$$\frac{1}{2} \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$\frac{\Delta I}{\Delta u} = \frac{I(u+1) - I(u-1)}{2}$$

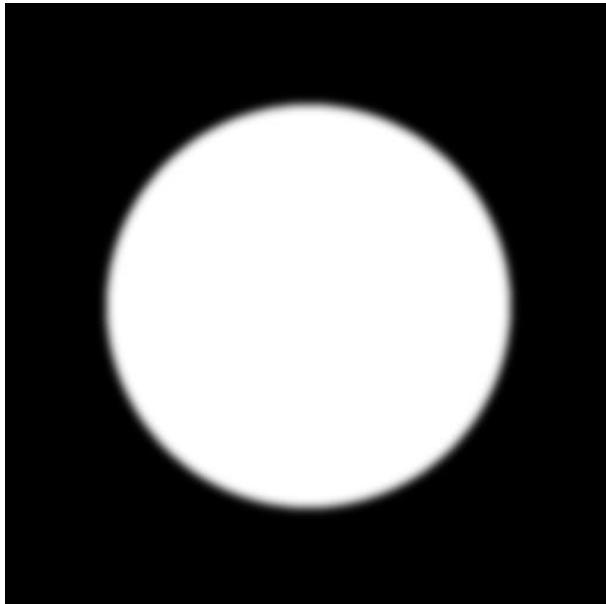
- Für die vertikale Richtung entsprechend:

$$\frac{1}{2} \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

$$\frac{\Delta I}{\Delta v} = \frac{I(v+1) - I(v-1)}{2}$$

Gradienten-Filtermasken

Bild

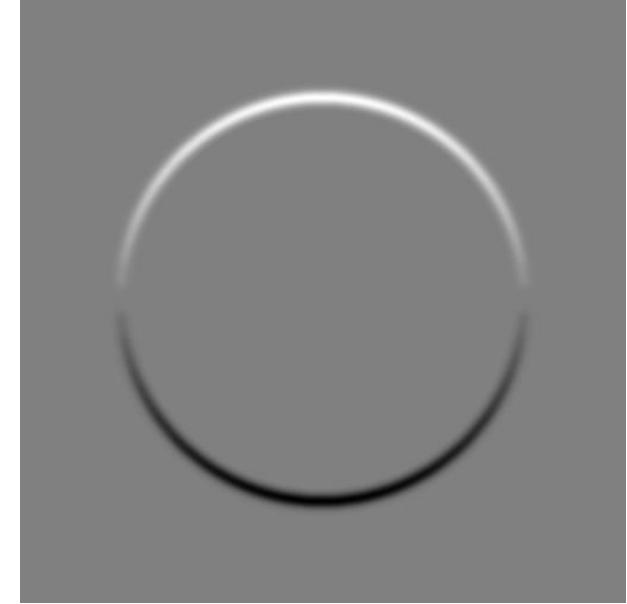


$$\frac{1}{2} \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$



$$\frac{\Delta I}{\Delta u} = \frac{I(u+1) - I(u-1)}{2}$$

$$\frac{1}{2} \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$



$$\frac{\Delta I}{\Delta v} = \frac{I(v+1) - I(v-1)}{2}$$

Filter zur Kantendetektion

- Erste Variante: **Prewitt-Operator**

- Mittelung über drei Zeilen bzw. Spalten mit Rechteckfilter
- die erstreckt sich über drei Zeilen bzw. Spalten, um der Rauschanfälligkeit des einfachen Gradientenoperators entgegenzuwirken
- Eine Annäherung zu $\frac{\partial I(u, v)}{\partial u}$ und $\frac{\partial I(u, v)}{\partial v}$

$$\mathbf{h}_x^P = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$\frac{1}{6} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{h}_y^P = \frac{1}{2} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \cdot \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{6} \cdot \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Filter zur Kantendetektion

- Zweite Variante: **Sobel-Operator**
 - Ähnlich zu Prewitt Operator
 - eine andere Annäherung zu $\frac{\partial I(u, v)}{\partial u}$ und $\frac{\partial I(u, v)}{\partial v}$, mit unterschiedlicher Gewichten
 - Die zentrale Zeile bzw. Spalte haben mehr Gewichte

$$\mathbf{h}_x^S = \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$\frac{1}{8} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{h}_y^S = \frac{1}{2} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \cdot \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$\frac{1}{8} \cdot \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel-Operator: Beispiel

Bild $I(u, v)$



$$\frac{\partial I(u, v)}{\partial u}$$



$$\frac{\partial I(u, v)}{\partial v}$$



$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial u}\right)^2 + \left(\frac{\partial I}{\partial v}\right)^2}$$



Filter zur Kantendetektion

- Dritte Variante: **Roberts-Operator**
 - schätzt die Richtungsableitungen entlang der beiden Diagonalen
 - eine andere Annäherung zu $\nabla_{\vec{w}} I(u, v)$, mit zwei Richtungsvektoren \vec{w}

$$H_1^R = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \text{und} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

0	1
-1	0

-1	0
0	1

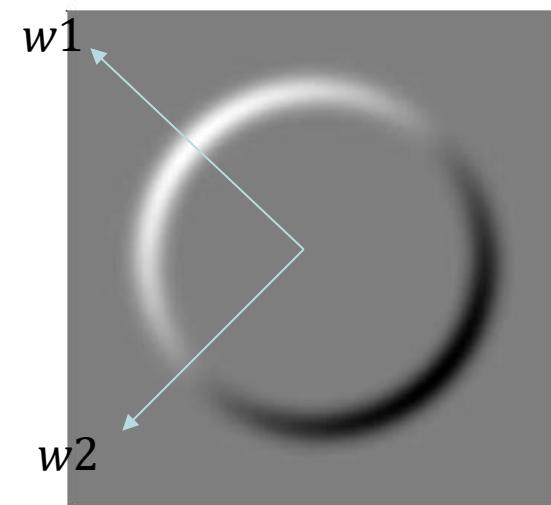
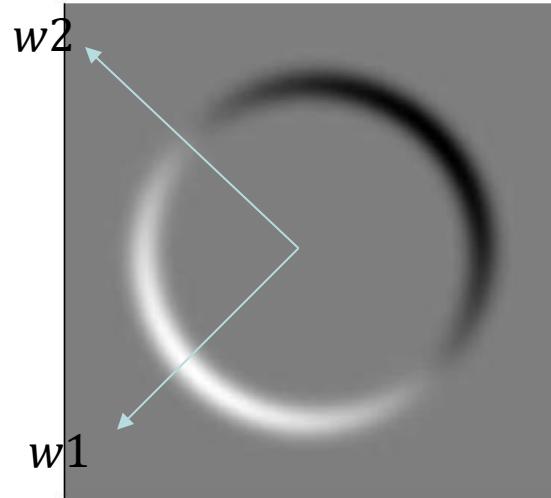
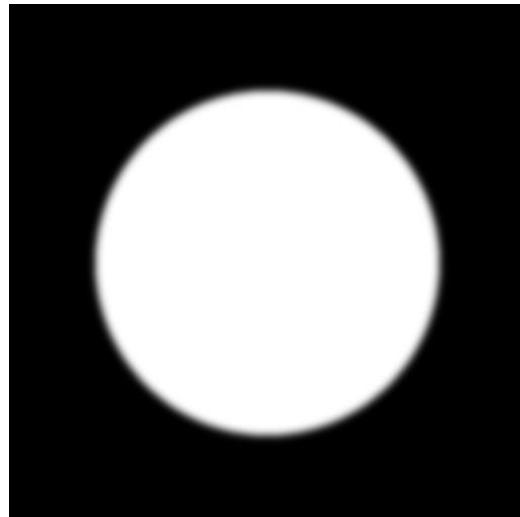
Filter zur Kantendetektion

- **Roberts-Operator** haben unterschiedliche Empfindlichkeit in unterschiedlicher Richtungen, d.h. Richtungsabhängig

$$H_1^R = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \text{und} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$I(u, v) * H_1^R$$

$$I(u, v) * H_2^R$$



w1: Richtung mit max Empfindlichkeit
w2: Richtung mit min Empfindlichkeit

Filter zur Kantendetektion

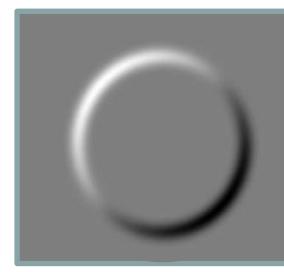
■ Vierte Variante: **Kompass-Operatoren**

- nicht nur ein Paar Filtern für zwei (orthogonale) Richtungen einzusetzen
- sondern einen Satz von Filtern für mehrere Richtungen

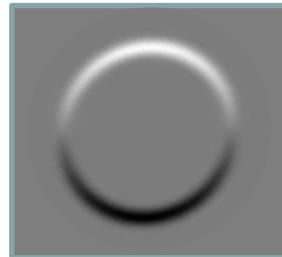
$$I(u, v) * H_0^K$$



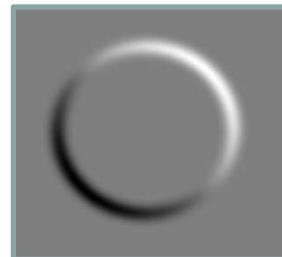
$$I(u, v) * H_1^K$$



$$I(u, v) * H_2^K$$



$$I(u, v) * H_3^K$$



$$H_0^K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$H_2^K = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$H_4^K = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$H_6^K = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$H_1^K = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

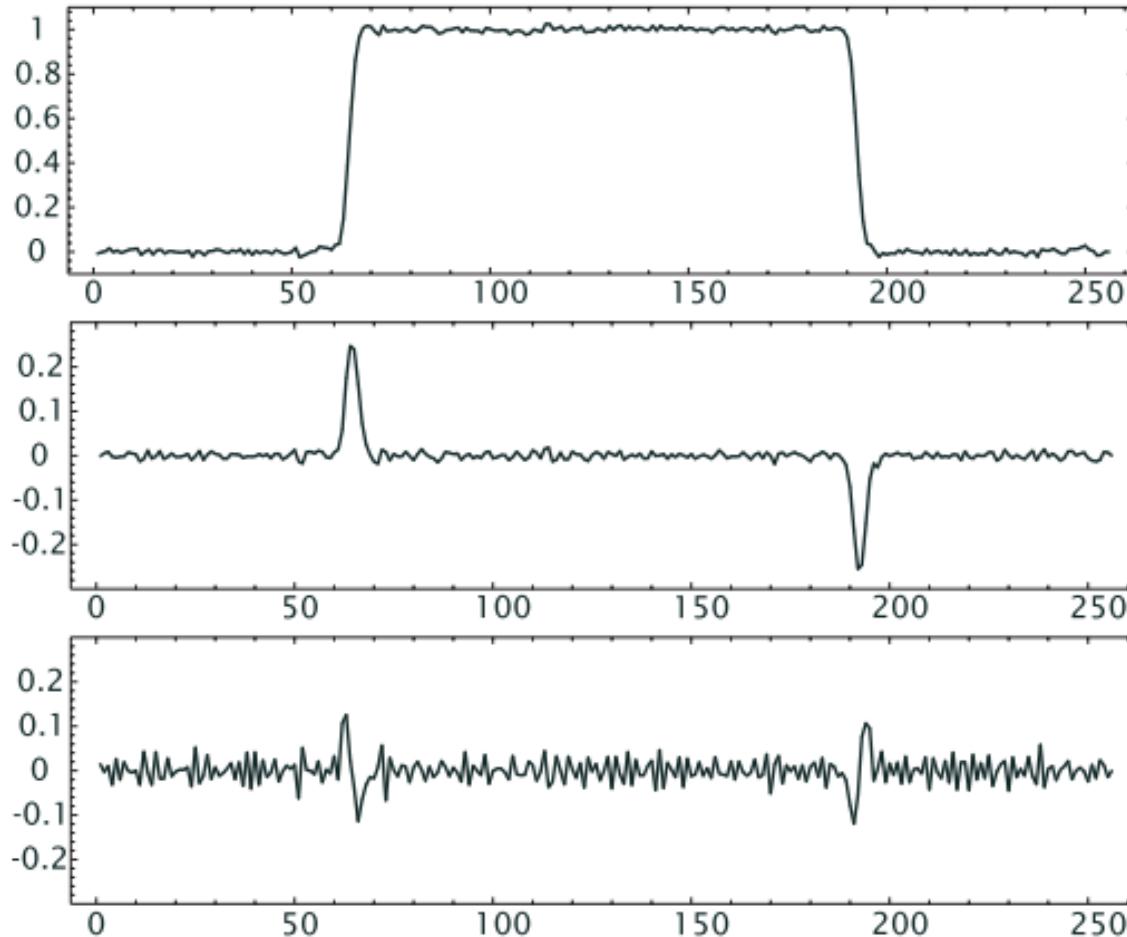
$$H_3^K = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

$$H_5^K = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$H_7^K = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

Gradienten und Störungen

- Durch die Ableitung einer Funktion wird i.d.R. Rauschen verstärkt

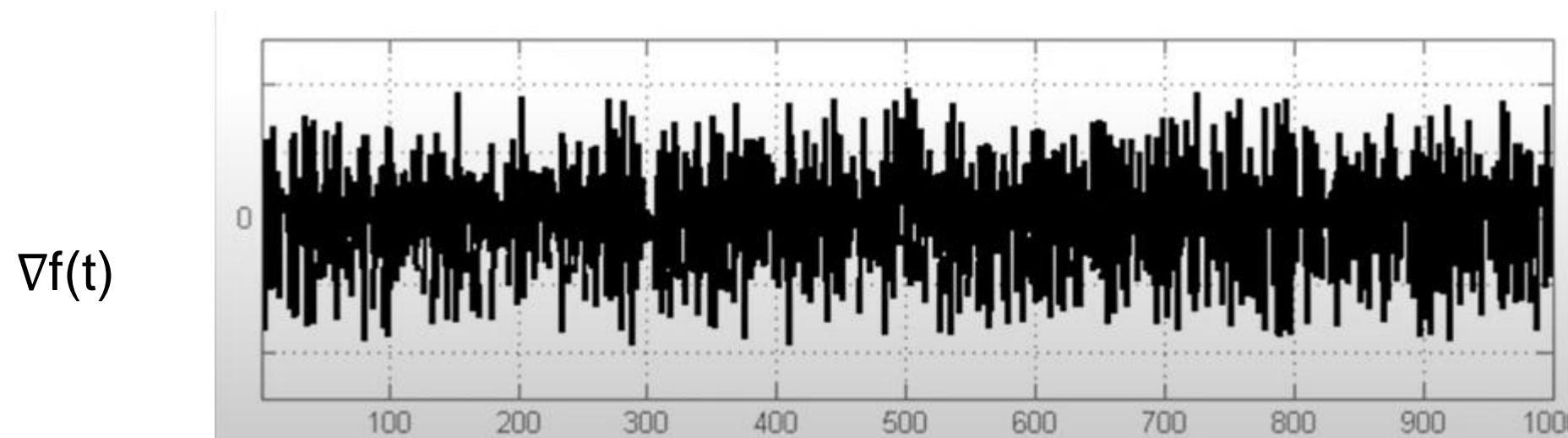
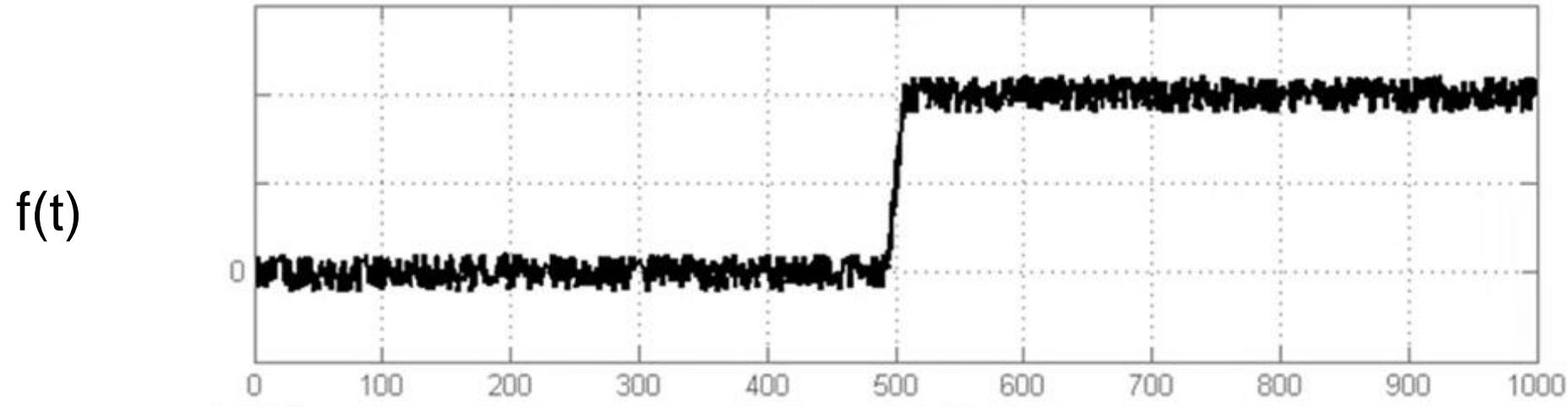


1. Ableitung

2. Ableitung

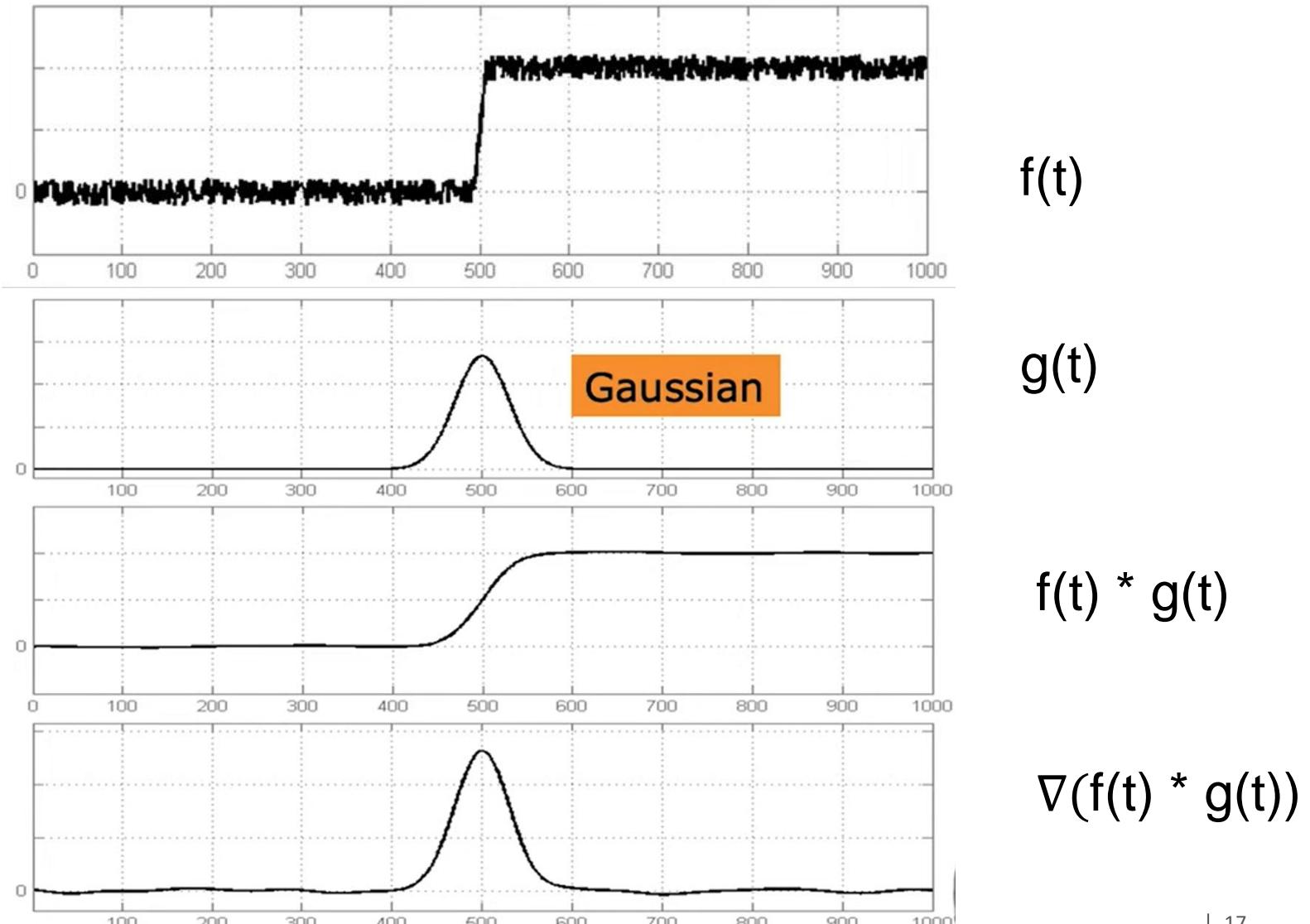
Gradienten und Störungen

- Wenn das Rauschen zu stark ist, können wir nicht erkennen, wo sich die Kante befindet.



Gradienten und Störungen

- Vor der Berechnung von Gradienten sollte man eine Glättung vornehmen, z.B. mittels Gaußfilter



Filter zur Kantendetektion

- Fünfte Variante: **Ableitung-von-Gauß Filter (1. Ordnung)**

- zunächst Gaußfilter zur Glättung anwenden
 - dann Ableitung

- 1D:

$$\nabla(f(t) * g(t)) = \nabla(g(t) * f(t)) = \boxed{\nabla g(t)} * f(t)$$

Ableitung-von-Gauß Filter

Warum?

$$\frac{d}{dx}(f * g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$$

- 2D:

$$\nabla_u(f(u,v) * g(u, v)) = \boxed{\nabla_u g(u, v)} * f(u, v)$$

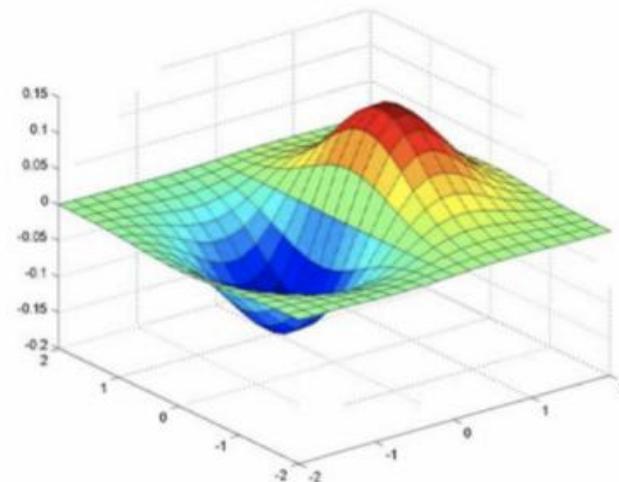
$$\nabla_v(f(u,v) * g(u, v)) = \boxed{\nabla_v g(u, v)} * f(u, v)$$

Filter zur Kantendetektion

- **Ableitung-von-Gauß Filter (1. Ordnung)**

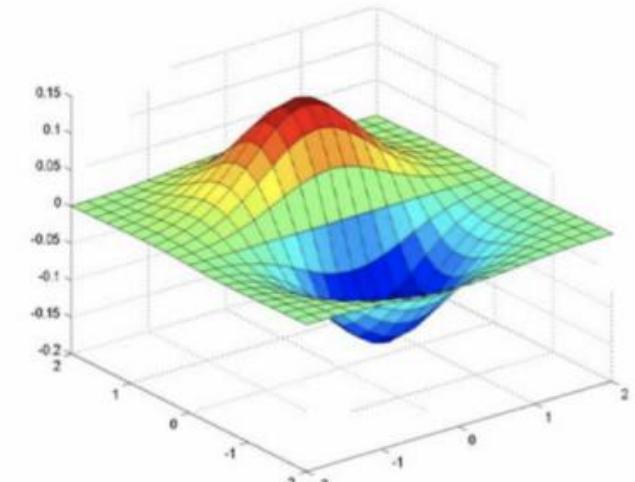
- gegen Rauschen
- richtungsabhängig

$$\nabla_u g(u, v)$$

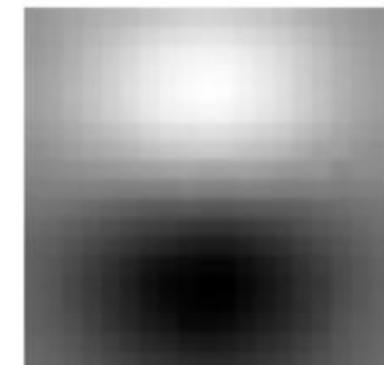
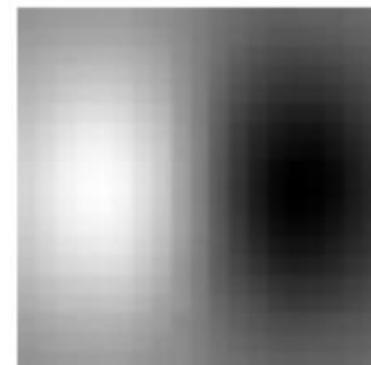


u- Richtung

$$\nabla_v g(u, v)$$



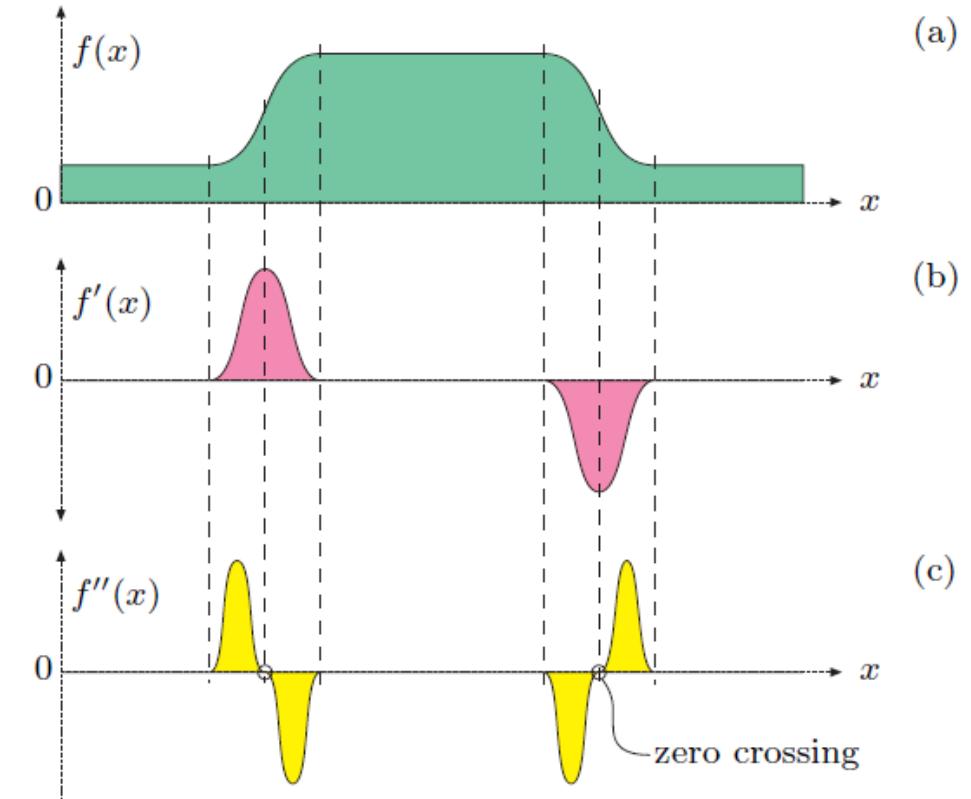
v- Richtung



Filter mit zweiter Ableitung

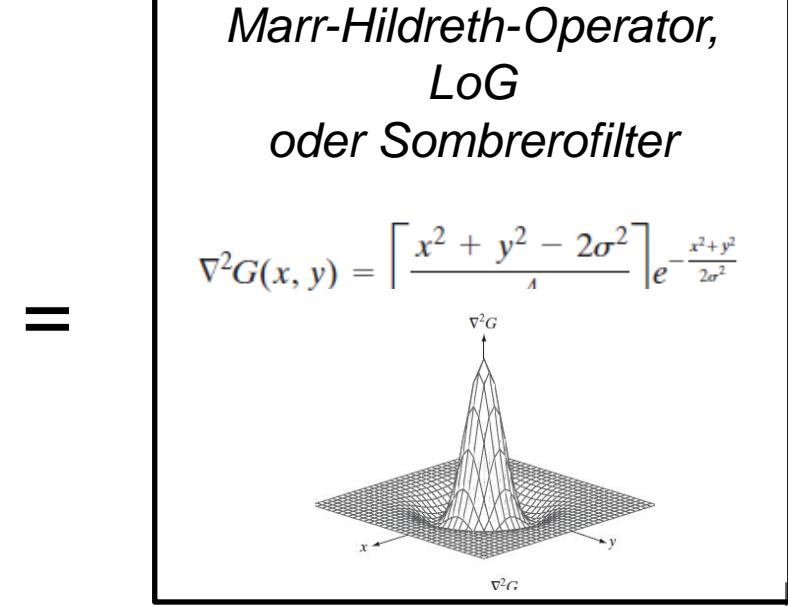
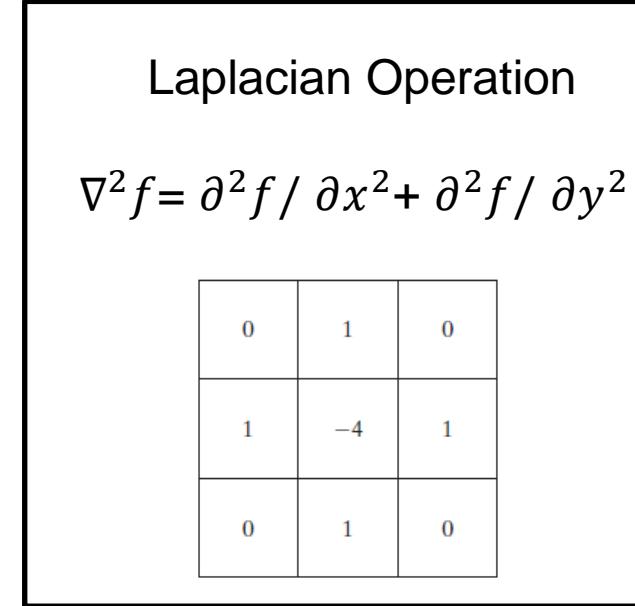
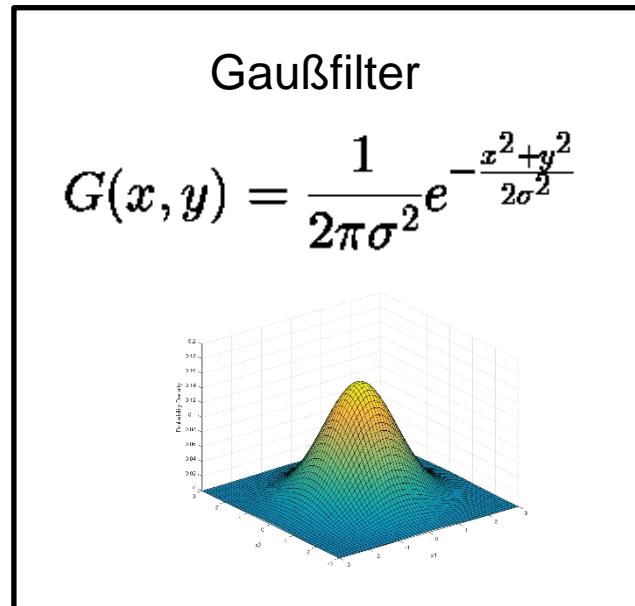
Filter mit zweiter Ableitung

- Bisher haben die bisherigen Filter nur die ersten Ableitungen verwendet
 - der Intensitätsübergang ist breit
 - die Kanten können schwer genau zu lokalisieren
- Filter mit zweiter Ableitung kann das Problem besser lösen
 - Eine Kante ist der Ort, an dem die **zweite Ableitung** der Grauwerte einen **Nulldurchgang** aufweist
 - “Laplacian-of-Gaussian” Operator (LoG)
 - Canny Kantendetektor



“Laplacian-of-Gaussian“ Operator (LoG)

- Die zweite Ableitung verstrtzt Rauschen sehr stark
- LoG :
 - verwendet zuerst eine Gltung durch Gaufilter, dann Laplacian-Filterung
 - Kombiniert die zwei Schritte in ein gemeinsames, lineares Filter
 - Oder als Marr-Hildreth-Operator, oder Sombrerofilter genannt



Laplacian of Gaussian – Herleitung

- Gaußfilter:

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

- LoG:

$$\text{LoG}(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

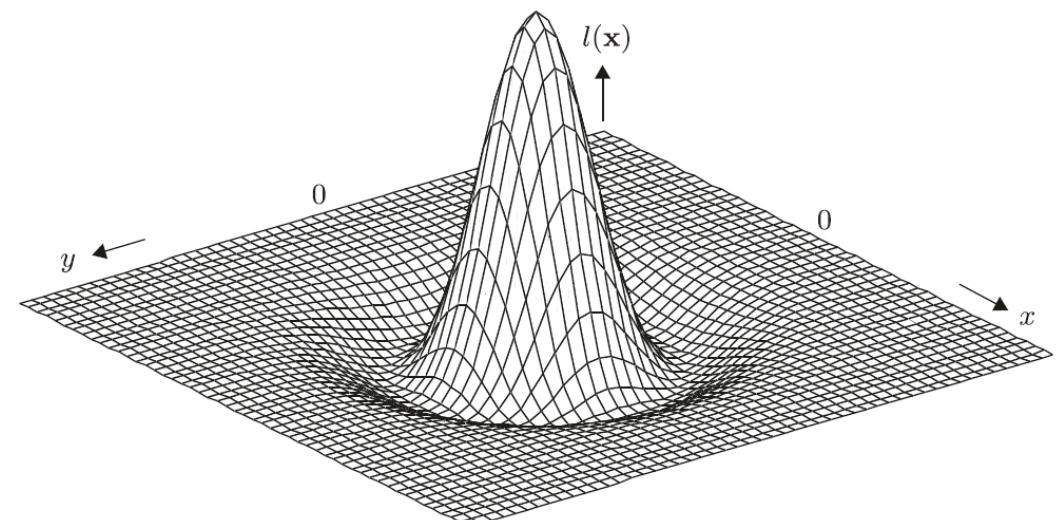
- LoG = verwendet zuerst Gaußfilter, dann Laplacian-Filterung

$$I(u, v) * \nabla^2 G(u, v) = I(u, v) * [G(u, v) * \text{Laplacian}(u, v)]$$

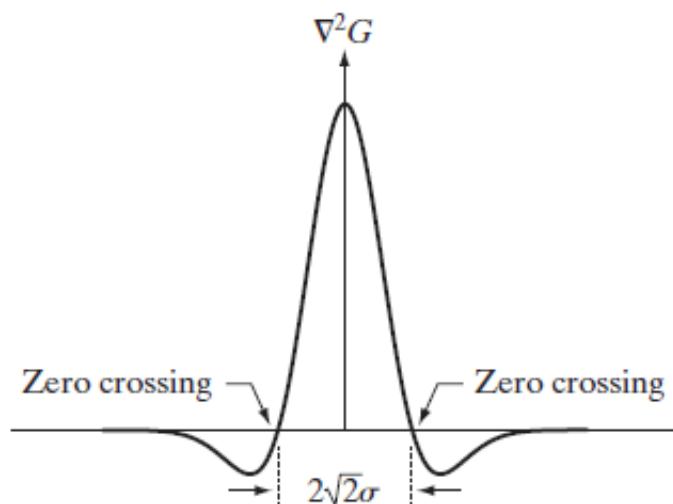
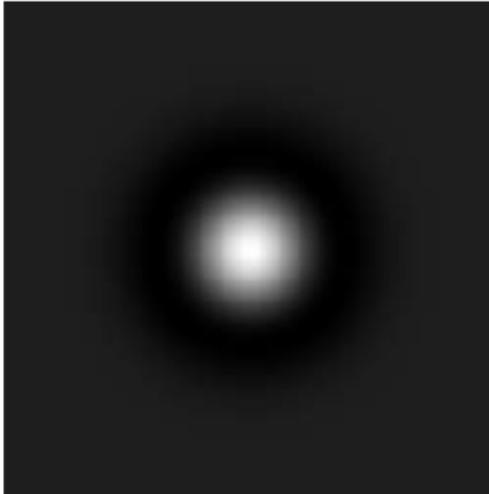
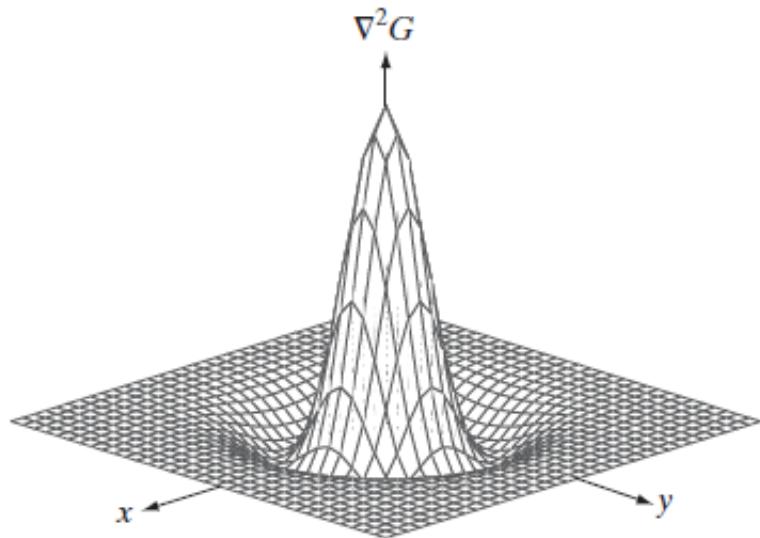
$$= I(u, v) * G(u, v) * \text{Laplacian}(u, v)$$

“Eigenschaften der Faltung”

*Marr-Hildreth-Operator,
LoG
oder Sombrero filter*



LoG Maske (n=5)



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Source: Gonzalez, Woods, Digital Image Processing, 3rd Edition, 2008

LoG: Eigenschaft und Anwendungsschritte

- Gaußfilter: σ steuert die Unschärfeeffekte

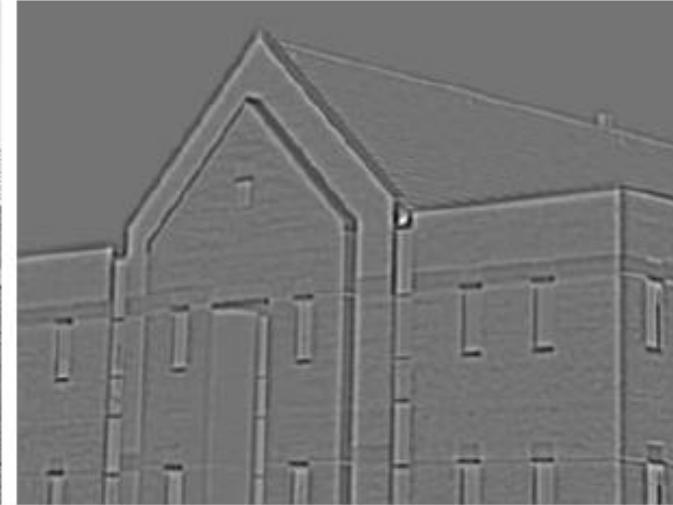
$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Laplacian Operator:
 - Kantendetektor mit erster Ableitung ist richtungsabhängig
 - Vorteil: **isotrop** = rotationsinvariant, nicht richtungsabhängig

- Anwendung der LoG Operator für Kantendetektion
 - Step 1: Wählen σ . Gaußfilter anwenden
 - Step 2: Wählen n in der Laplacian Maske. Anwenden Laplacian-Operator.
 - Step 3: Nulldurchgang finden

LoG Operator: Beispiel

Bild $I(u, v)$



$I(u, v) * \nabla^2 G(u, v)$
 $\sigma = 4, n = 25.$

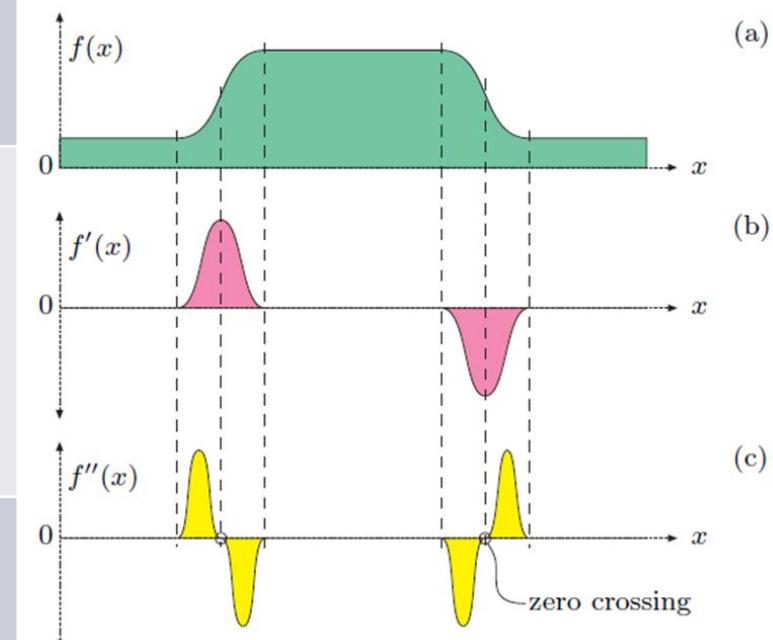
Nulldurchgänge
(mit Schwelle=0)



Nulldurchgänge
(mit Schwelle>0)

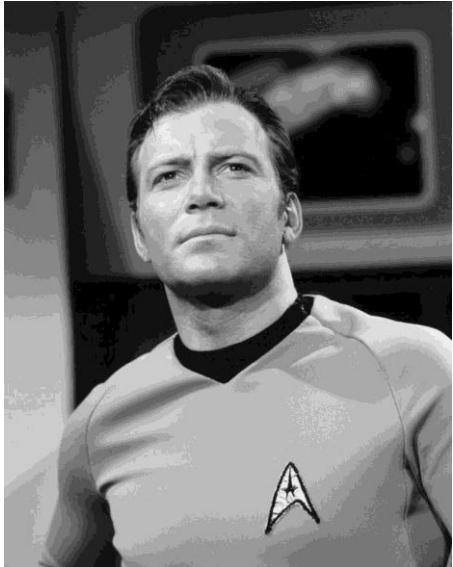
Gradient vs. Laplacian für Kantendetektion

Gadient	Lapacian
Ergeben den Ort und die Richtung	Ergeben nur den Ort angeben
Die Erkennung basiert auf dem maximalen Schwellenwert	Die Erkennung basiert auf dem Nulldurchgang
die erkannten Kanten können breit sein	die erkannte Kante ist dünn



LoG – Beispiel

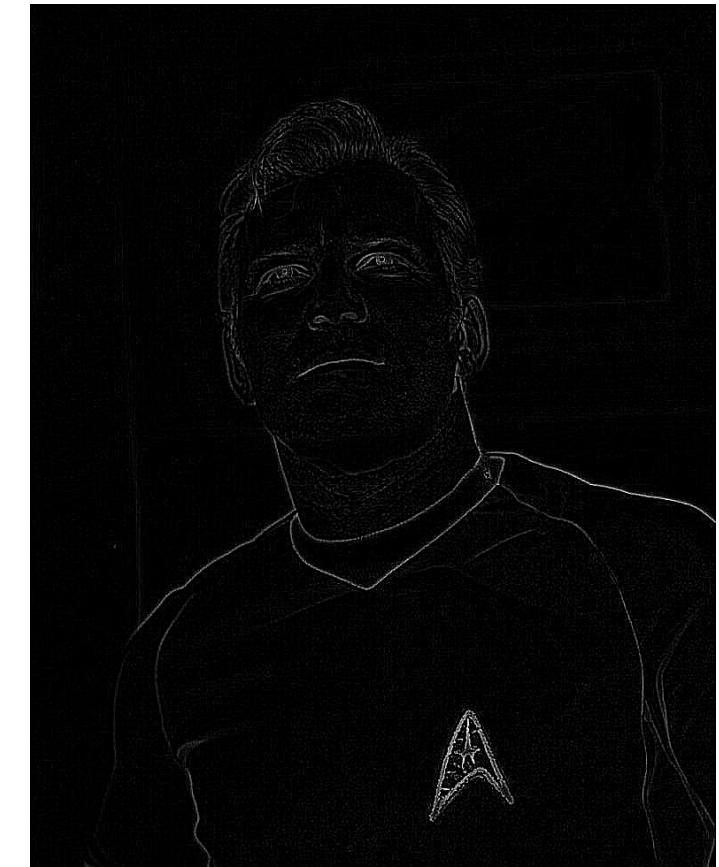
- Der LoG zeigt in der Regel dünnere sowie eine bessere Lokalisierung der Kanten im Vergleich zum Sobel-Operator
- Man kann daher auch den Sobel-Operator im Canny-Algorithmus durch einen LoG ersetzen



Sobel-Filter, Betrag



LoG, Betrag



Kanten und geglättete Gradienten

Canny Kantendetektor

Grundlagen: Gradient und Orientierung

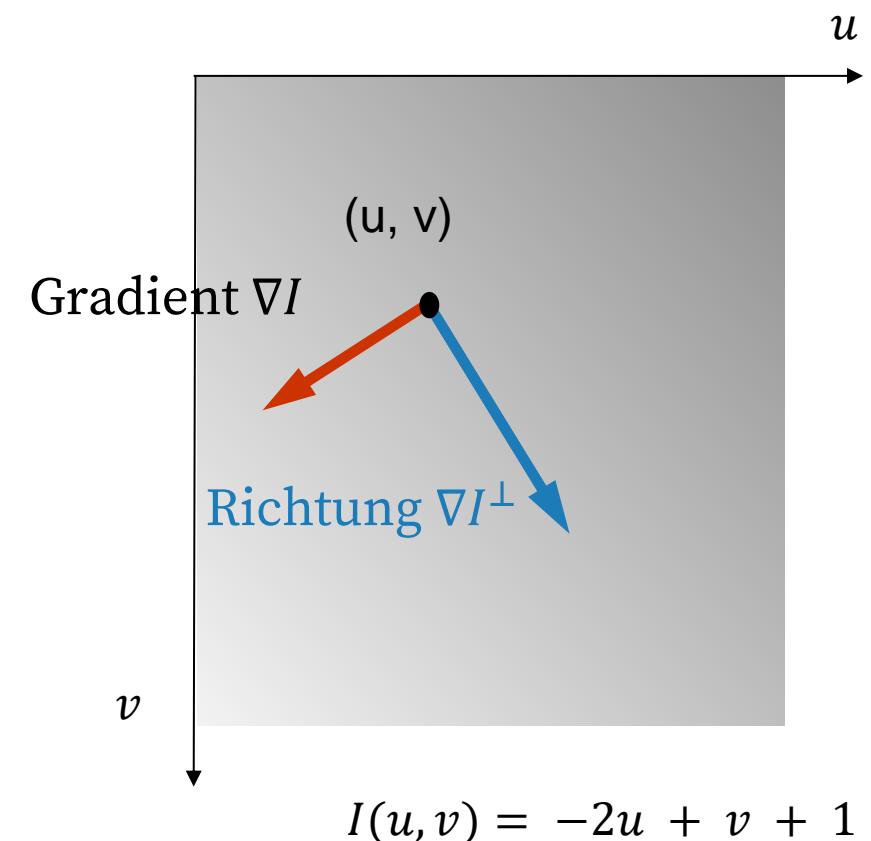
- Betrachten eine beliebige Funktion $I(u, v)$, $u \in \mathbb{R}$, $v \in \mathbb{R}$

- Der Gradient zeigt in die Richtung der größten Grauwertänderung

$$\nabla I(u, v) = \left(\frac{\partial I}{\partial u}, \frac{\partial I}{\partial v} \right)$$

- In der orthogonalen Richtung gibt es örtlich keine Grauwertänderung:

$$\nabla I(u, v)^\perp = \left(\frac{\partial I}{\partial v}, -\frac{\partial I}{\partial u} \right)$$



Beispiel: $\nabla I = (-2, 1)$

$$\nabla I(u, v)^\perp = (1, 2)$$

Kantendetektion – Canny-Methode

- Der Canny-Kantendetektor besteht grob aus folgenden Schritten:
 1. Gradientenberechnung (auf geglättetem Bild)
 2. Kantenverdünnung mit z.B. Non-Maxima-Suppression
 3. Kantenverfolgung (Hystereseschwellwert)

Gradientenberechnung (auf geglättetem Bild)

- Unterschritte:
 - a) Glätterung mit Gausßfilter : $I(u, v) * G(u, v)$
 - b) Berechnen die Ableitungen (1. Ordnung): $\nabla_u(I * G)$ und $\nabla_v(I * G)$
 - z.B. mit Sobel Maske
 - c) Berechnen die Gradientenmagnitude:

$$M(u, v) = \sqrt{\nabla_u(I * G)^2 + \nabla_v(I * G)^2}$$

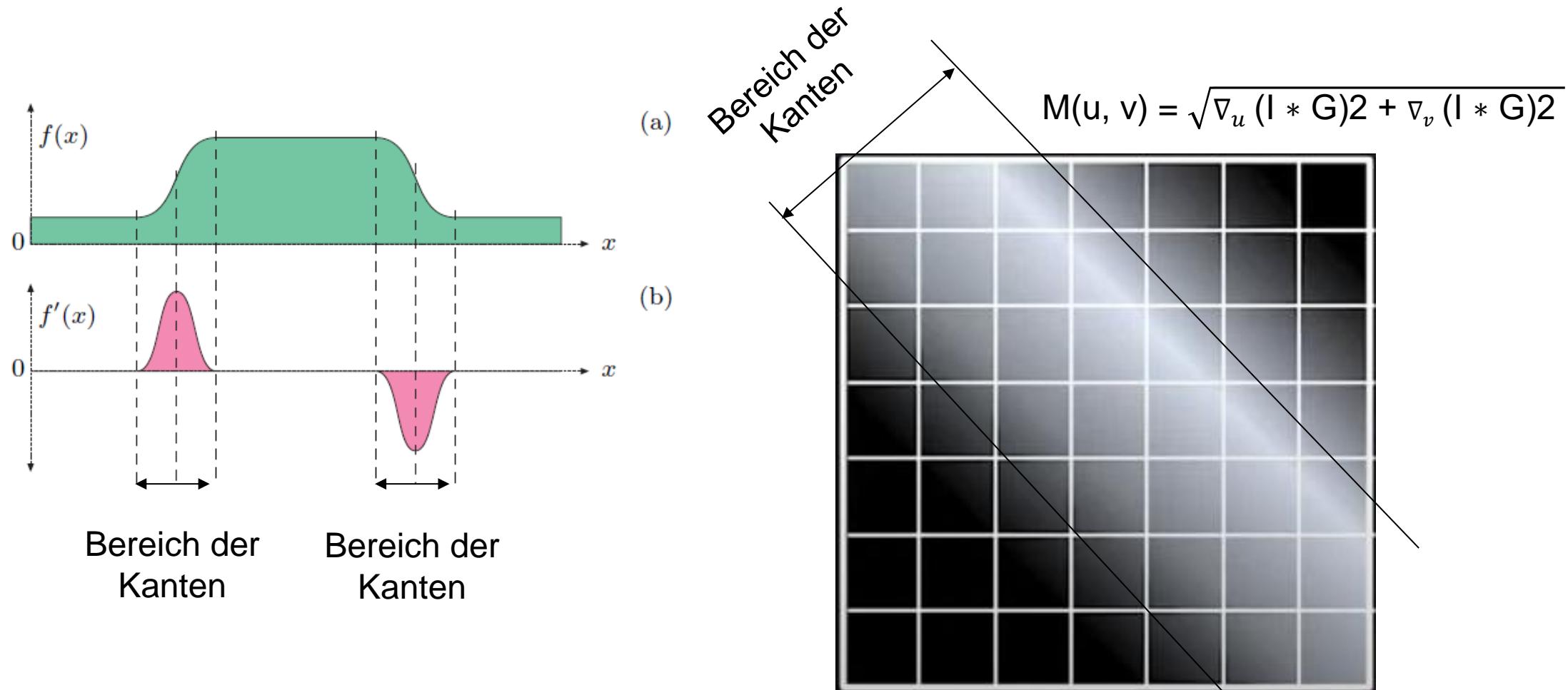
- d) Berechnen die Winkel:

$$\alpha(u, v) = \tan^{-1} \left[\frac{\nabla_u(I * G)}{\nabla_v(I * G)} \right]$$

Alle diese Vorgänge sind bereits besprochen worden.

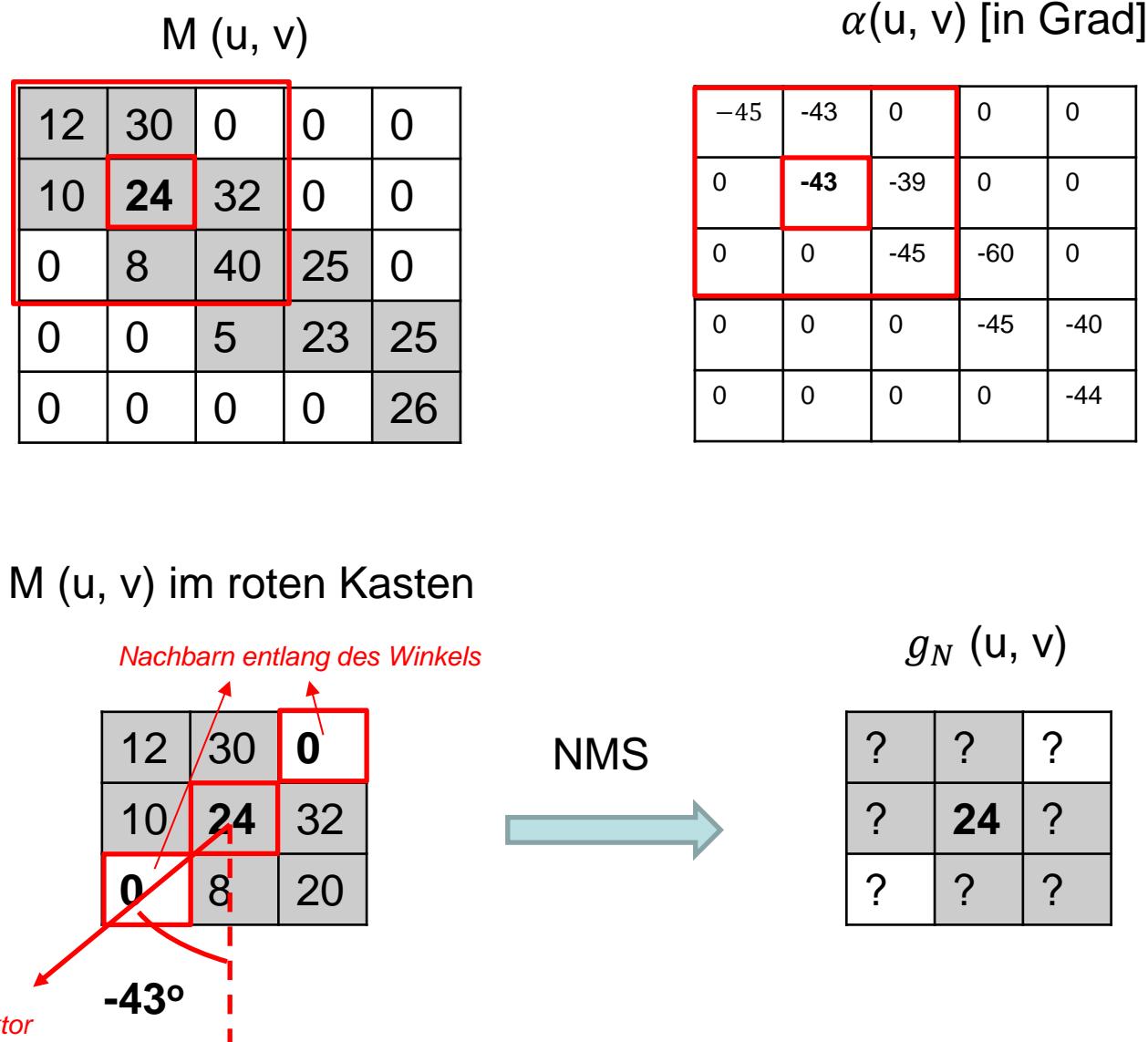
Kantenverdünnung mit Non-Maxima-Suppression

- Problem: im Bild $M(m, n)$, der Bereich der Kanten enthält in der Regel mehrere Pixel



Non-Maxima-Suppression (NMS)

- An jeder Position (u,v) erhalten wir $M(m, v)$ und $\alpha(u, v)$
- An jeder Position (u,v) , nur die Pixel entlang des Winkels $\alpha(u, v)$ werden in MNS berücksichtigt
- Für jede Position (u,v) , wenn $M(x, y)$ kleiner ist als seiner Nachbarn (entlang des Winkels), dann ist $g_N(x,y)=0$
 - dieses Pixel für eine Kante wird entfernt
 - die Kante ist daher dünner geworden!
- andernfalls $g_N(x,y)=M(x,y)$



Non-Maxima-Suppression (NMS)

$M(u, v)$

12	30	0	0	0
10	24	32	0	0
0	8	40	25	0
0	0	5	23	25
0	0	0	0	26

$\alpha(u, v)$

-45	-43	0	0	0
0	-43	-39	0	0
0	0	-45	-60	0
0	0	0	-45	-40
0	0	0	0	-44

NMS zu jeder Position

$g_N(u, v)$

24	32	0
0	40	25
0	0	23

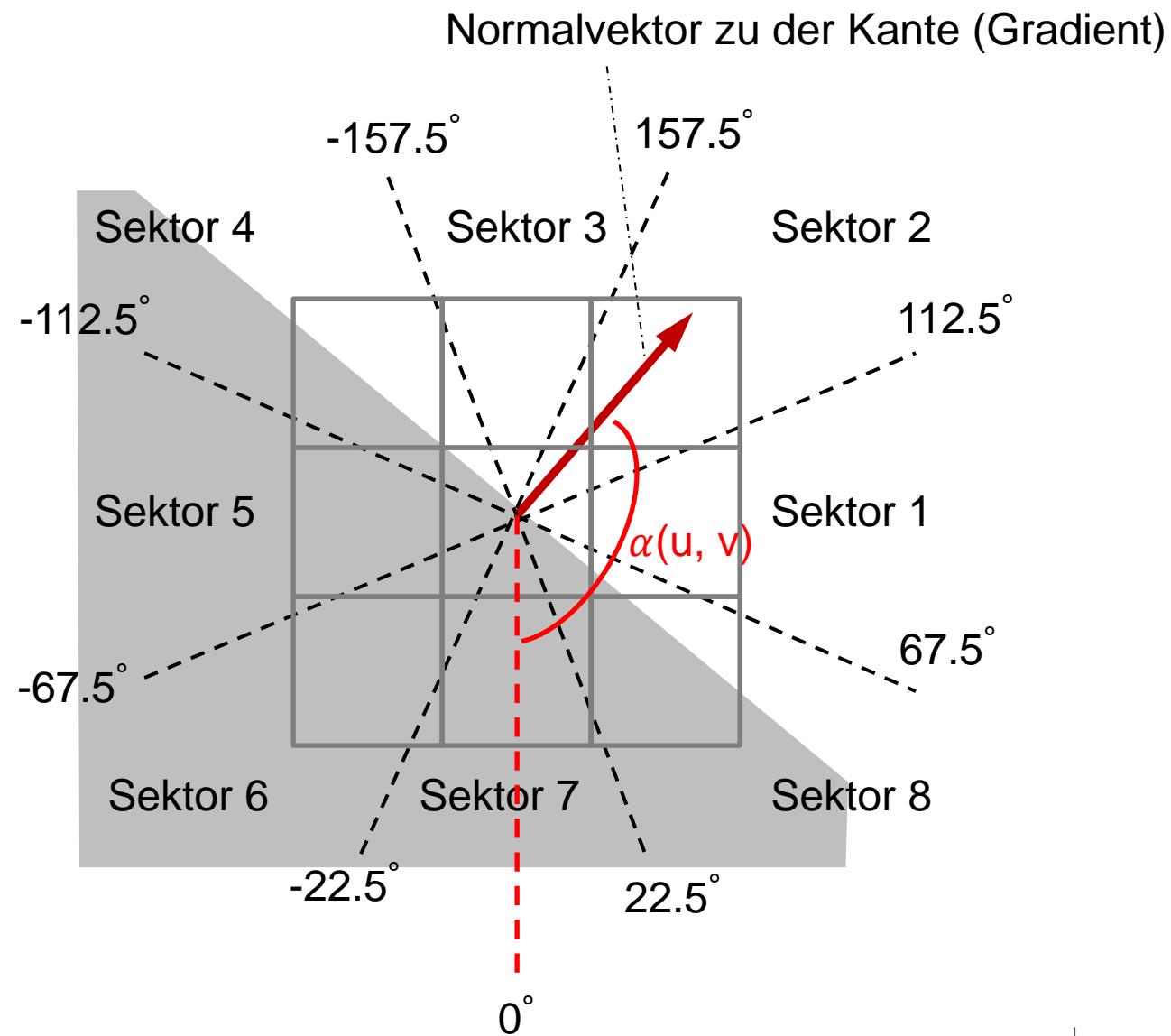
die Kante ist daher dünner geworden!

Non-Maxima-Suppression (NMS)

- **Non-Maxima-Suppression** (Nicht-Maxima-Unterdrückung)
 - die Kantenrichtung $\alpha(u, v)$ wird bei der Suppression berücksichtigt
 - Nachbarsektoren und benachbarte Pixel entlang der normalen Richtung der Kante für NMS werden basierend auf $\alpha(u, v)$ entschieden
 - Unter den benachbarten Punkten beibehalten nur der maximale Gradient Betrag $M(m, v)$

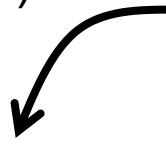
Non-Maxima-Suppression (NMS)

- die Richtungen in 8 Sektoren aufzuteilen
- Z.B. wenn $\alpha(u, v) = 120^\circ$, dann Sektor 2 und Sektor 6 als Nachbarschaften
- Z.B. wenn $\alpha(u, v) = -70^\circ$, dann Sektor 5 und Sektor 1 als Nachbarschaften



Geglättete Gradienten

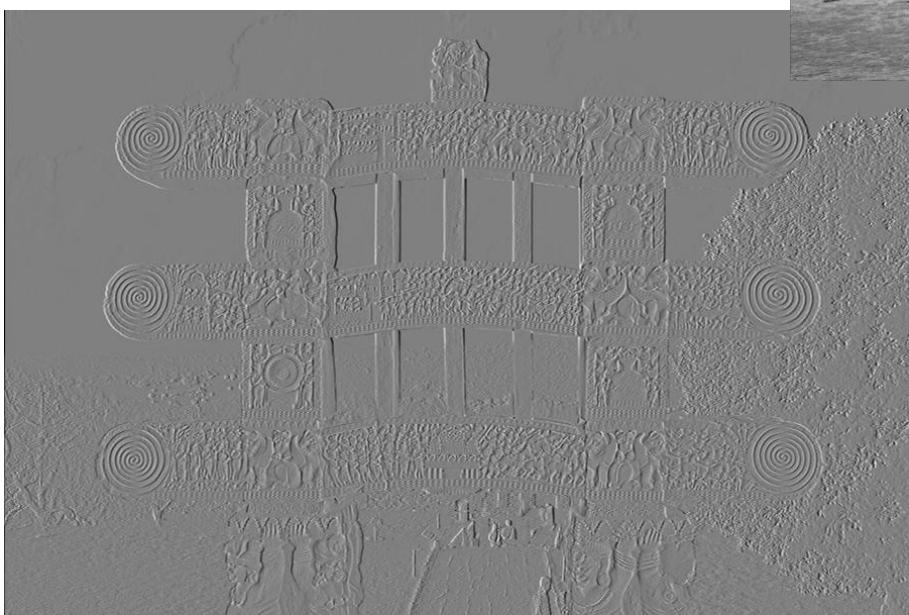
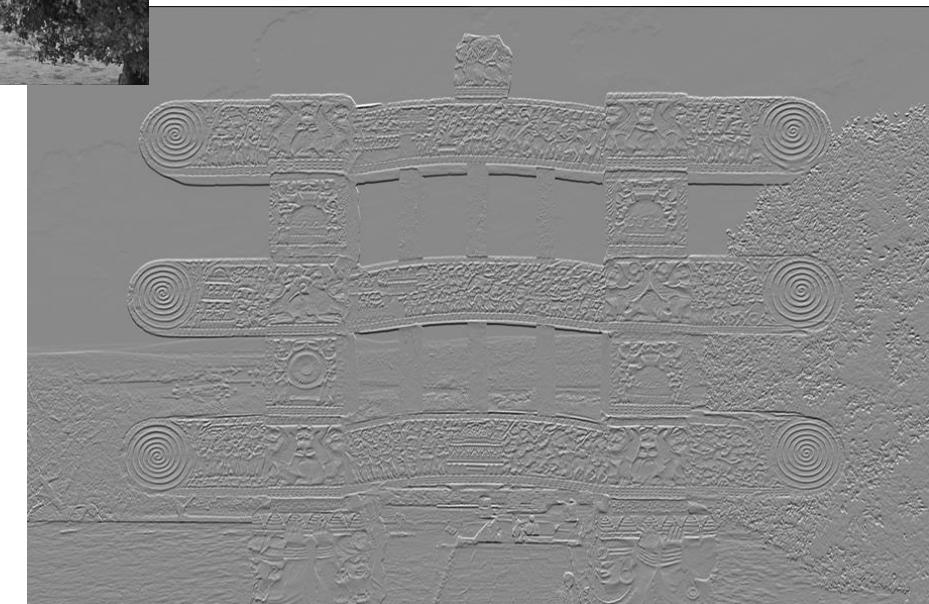
$$\nabla_u (I^* G)$$



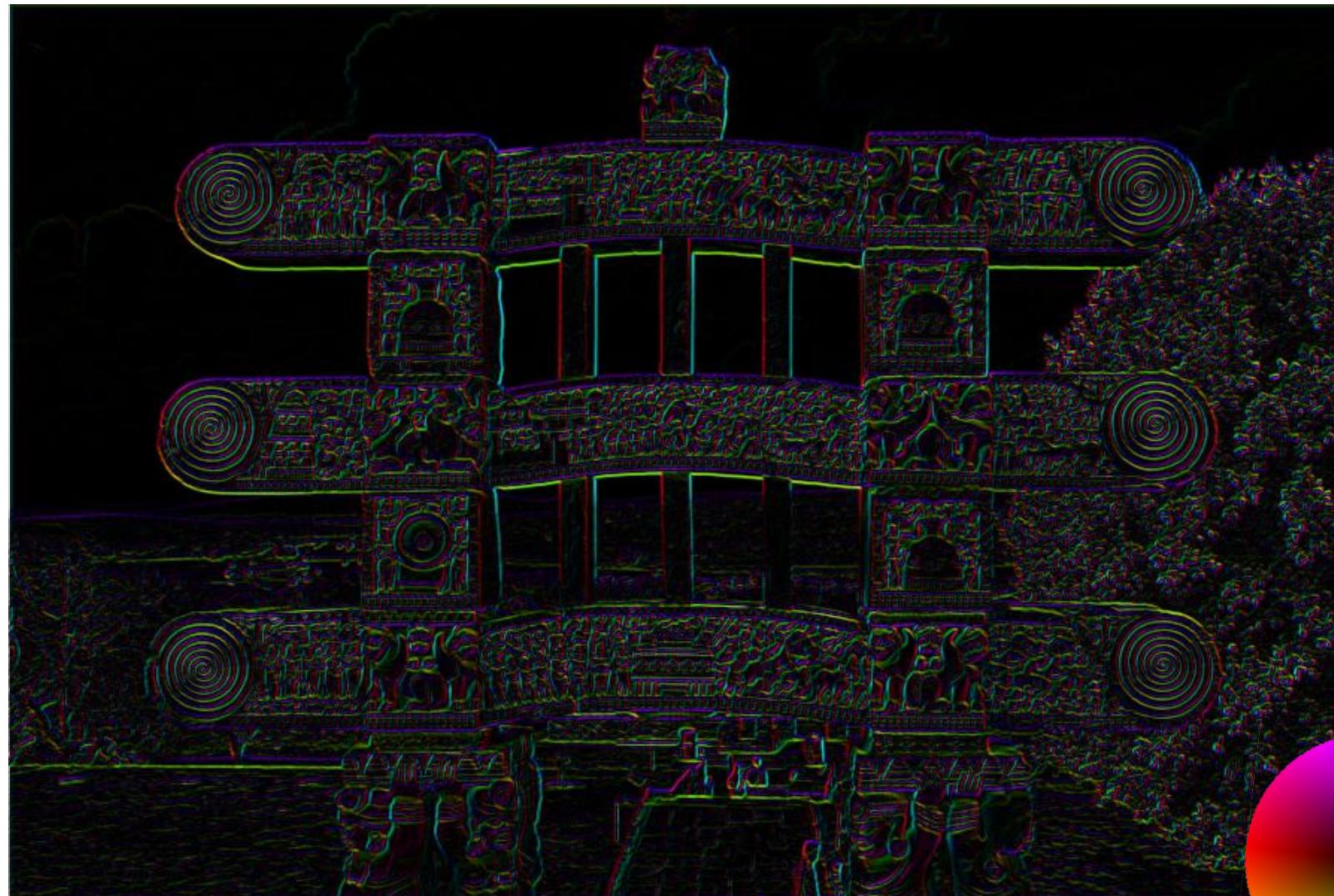
$$\begin{matrix} u \\ v \end{matrix}$$



$$\nabla_v (I^* G)$$



Gradientenbilder – Darstellung Gradient



Helligkeit: Betrag $M(u, v)$

Farben: Richtung $\alpha(u, v)$

Non-Maxima-Suppression – Ergebnis

- Kanten werden verdünnt

Gradient vor der Non-Maxima-Suppression (NMS)



Gradient nach NMS



Kantendetektion – Canny-Methode

- Der Canny-Kantendetektor besteht grob aus folgenden Schritten:
 1. Gradientenberechnung (auf geglättetem Bild)
 2. Kantenverdünnung mit Non-Maxima-Suppression
 3. Kantenverfolgung mit Hystereseschwellwert

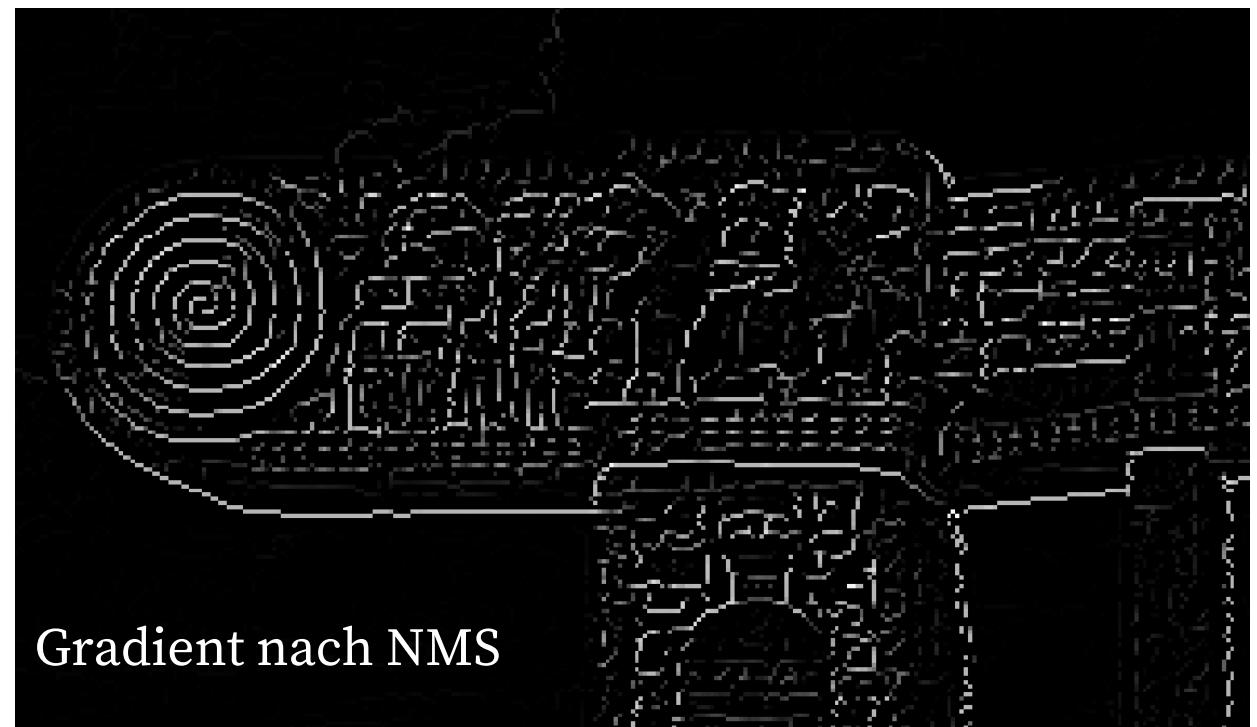
Idee: Kantenverfolgung

- Nach NMS erhalten wir ein Kantenbild
- Idee: Pixel mit relativ großen Werten **erscheinen** als Kante
- Kantenverfolgung: Die falschen Kantenpixel mit Hilfe von Schwellenwertverfahren zu reduzieren

Einfacher Schwellenwert:

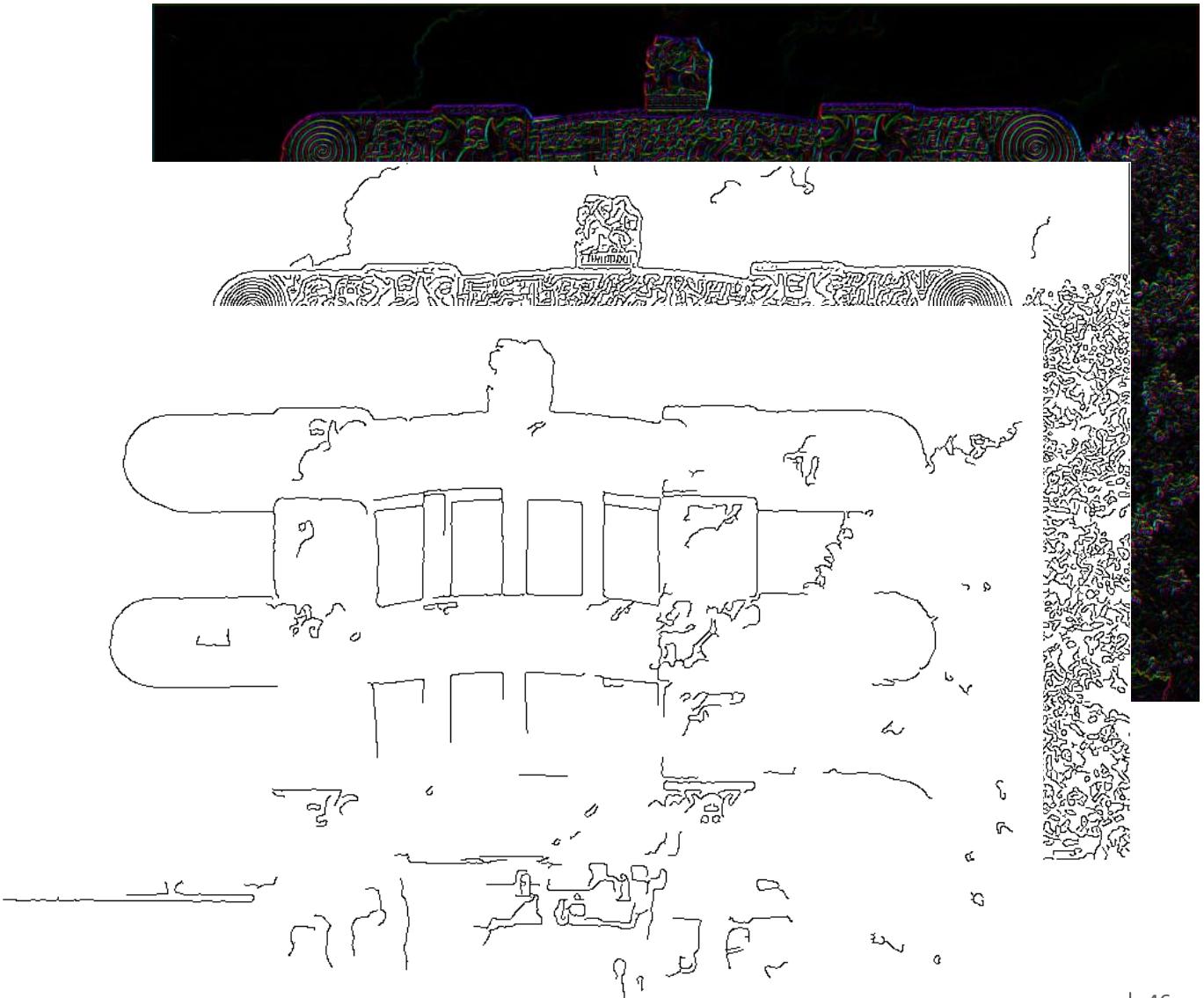
$$g_{Verfolgung}(u, v) = \begin{cases} g_{NMS}(u, v), & \text{if } g_{NMS}(u, v) \geq T_0 \\ 0, & \text{otherwise} \end{cases}$$

$$g_{NMS}(u, v)$$



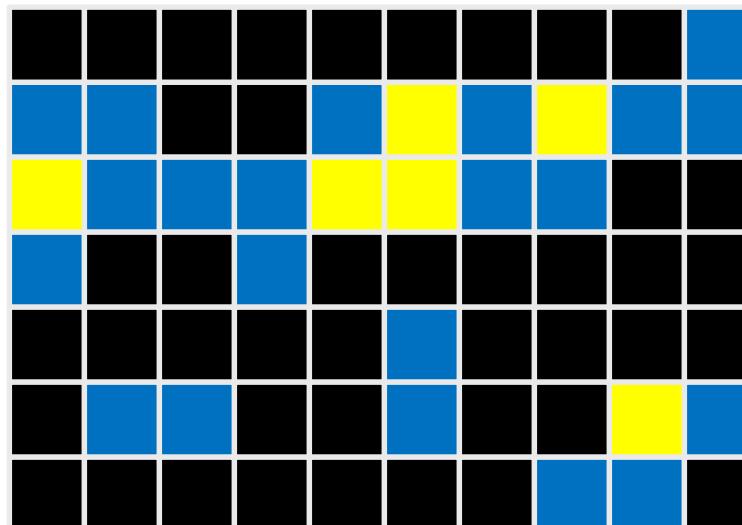
Schwellwert

- Was ist der sinnvolle Schwellwert für den Betrag?
 - Kleiner Schwellwert: Zuviel Störungen und Rauschen im Bild
 - Hoher Schwellwert: Konturen sind nicht zu Kanten verbunden
 - Lösungsansatz: Doppelter Schwellwert (**Hysterese Schwellwert**)



Hysterese Schwellwert

- Es wird ein hoher Schwellwert H und niedriger Schwellwert L definiert
- Alle Kanten mit Betrag kleiner L werden verworfen
- Alle Kanten mit Betrag größer H werden sicher behalten
- Eine Kante P mit Betrag $L < P < H$ wird nur behalten, **wenn es eine Verbindung zwischen P mit $\text{Beträgen} > L$ gibt**, welche P mit einer Kante $> H$ verbinden
- In der Praxis wird dies oftmals mit einer zusätzliche Kantenverfolgung kombiniert



	$ \nabla I \leq L$
	$L < \nabla I < H$
	$ \nabla I \geq H$

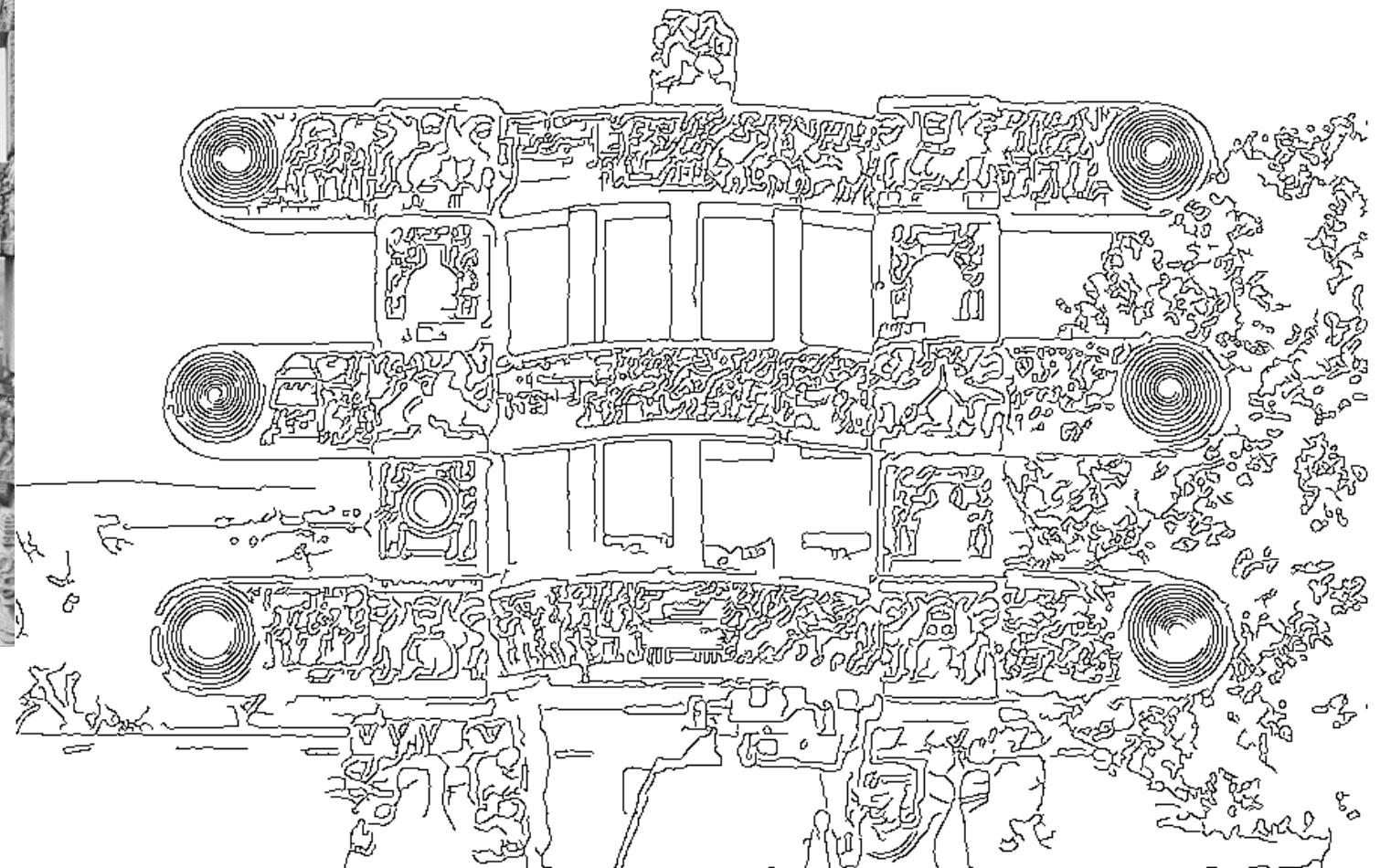
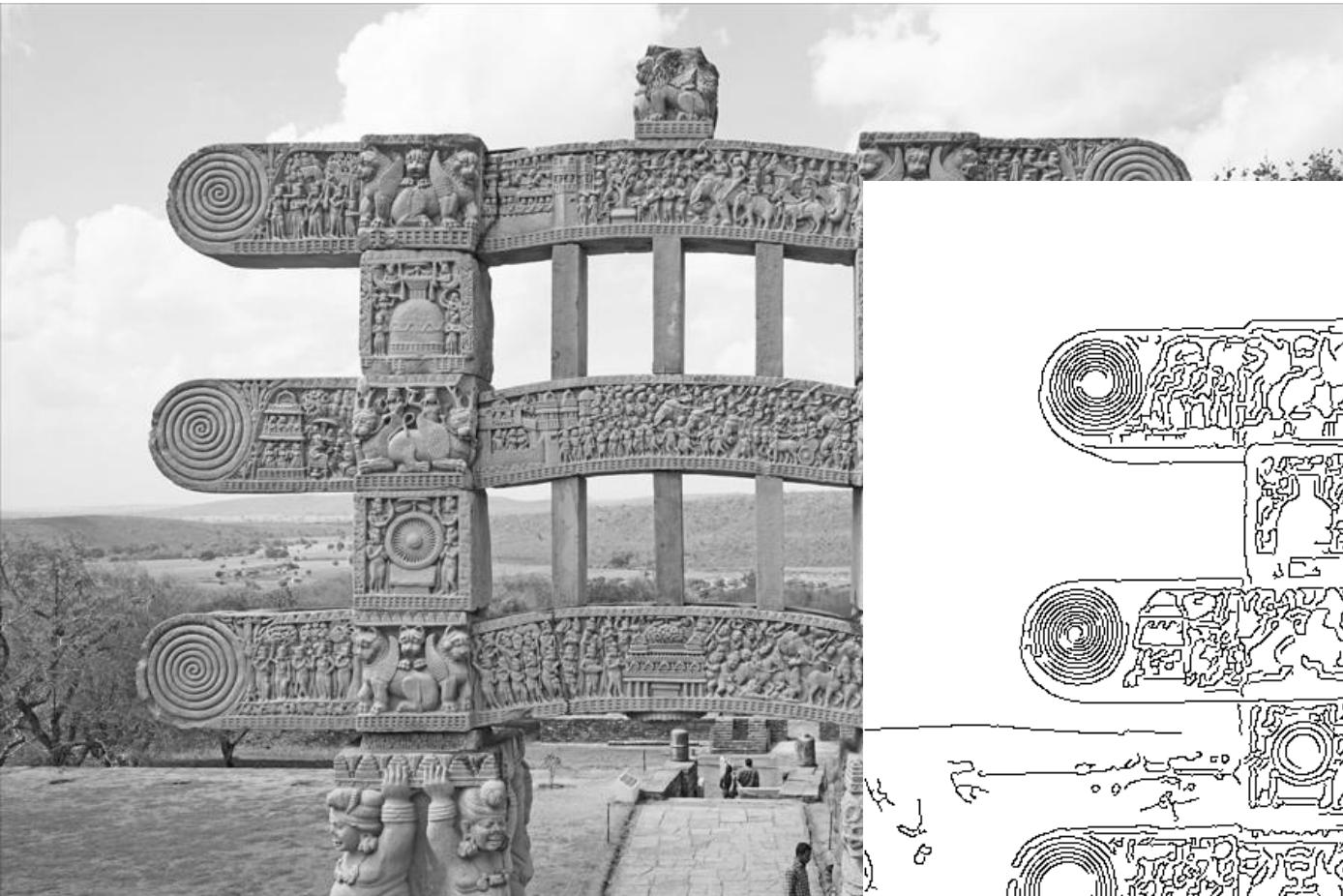
Canny Kantendetektor

Der Canny-Kantendetektionsalgorithmus kombiniert folgende Schritte:

1. Bildglättung mit Gaußfiltern
2. Berechnung des Gradienten (Betrag & Winkel) mit Filtern
3. Anwendung von Non-Maxima-Suppression zur Bestimmung lokaler Maxima und Kantenverdünnung
4. Anwendung von Hysterese Schwellwertoperation zur Erkennung und Verbindung von Kanten.

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

Canny Kantendetektor



Warum Kantenerkennung?

- Welche Information kann mit Kanten gewonnen werden?
- Kanten treten oftmals an Objektgrenzen auf und werden für die **Bildsegmentierung** eingesetzt
- Nachdem wir die Kantenerkennung angewendet haben, erhalten wir verschiedene Kanten. Dann können wir z.B. *Hough-Transformation* anwenden, um eine bestimmte Form zu erkennen, z. B. Linien oder Kreise.

Beispiel: Segmentierung mit Kanten

- Bild mit Erdbeeren: Ziel ist Erkennung und Lokalisierung der Erdbeeren



Beispiel: Segmentierung mit Kanten

- Bild mit Erdbeeren: Canny Kantendetektor



Beispiel: Segmentierung mit Kanten

- Bild mit Erdbeeren: Trainierter Kantendetektor (klassisch)



Beispiel: Segmentierung mit Kanten

- Bild mit Erdbeeren: Tiefe neuronale Netze, DexiNed

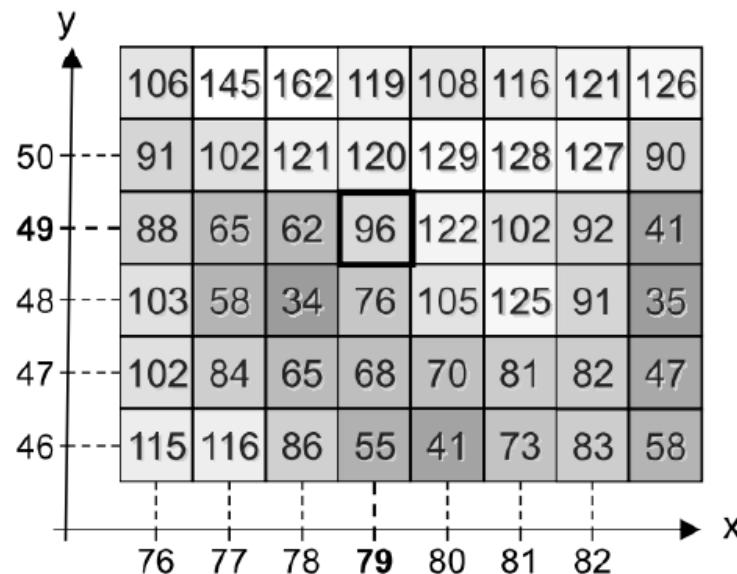


Zusammenfassung Kanten

- Kanten sind ein wichtiger Bestandteil der Wahrnehmung (biologisch und technisch)
- Kanten sind Informationsträger
 - Systemtheoretisch entsprechen sie hohen Frequenzen
- Kantendetektion erfolgt mit Ableitungen erster oder zweiter Ordnung
 - Erste Ableitung (Gradient): Gradientenfilter (einfach, Prewitt, Sobel), Canny-Kantendetektor
 - Zweite Ableitung (Laplacian-Operator): Laplacian of Gaussians (LoG), Difference of Gaussians (DoG), Hochpassfilter für die Bildschärfung (USM)

Beispielaufgabe

Es sei der folgende Ausschnitt eines Grauwertbildes gegeben:



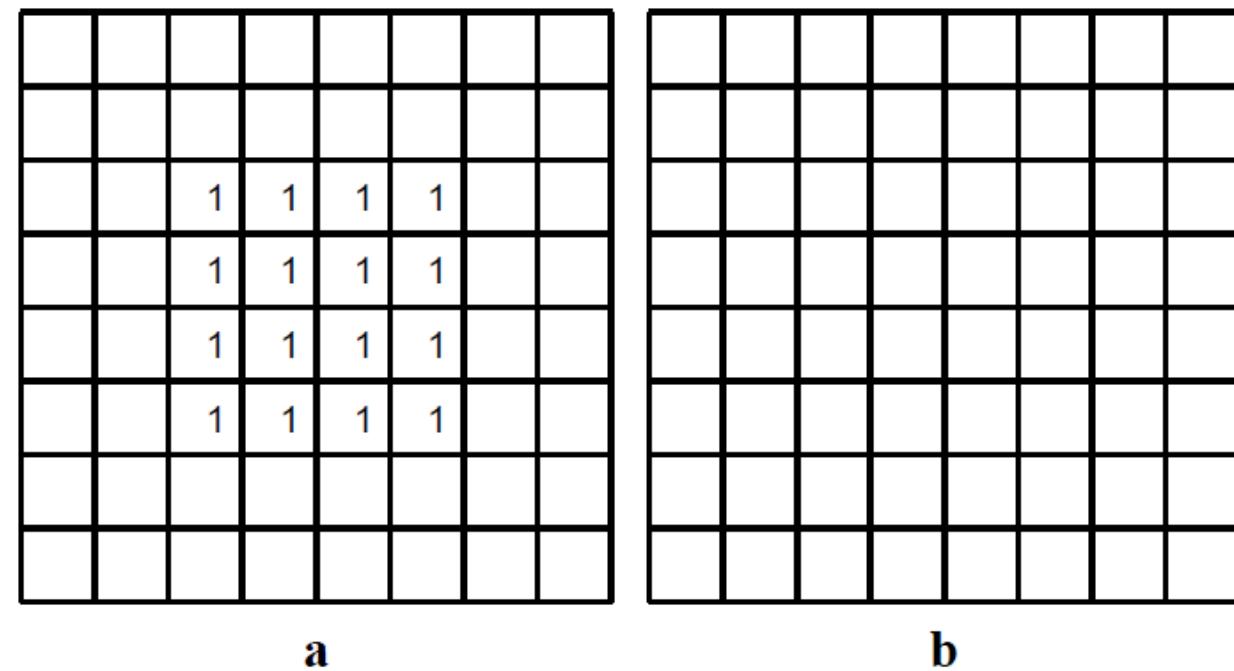
Berechnen Sie den Wert des eingerahmten Bildpunktes an der Position (79,49), für eine Filterung mit

- a) einem Mittelungsfilter (Typus *box*) der Größe 3×3 .
- b) einem Gradientenfilter der Größe 1×3 .
- c) Sobel- oder Prewitt-Filter (Sie können wählen) in x -Richtung.
- d) einem Medianfilter (*rank order filter*) der Größe 3×3 .

Beispielaufgabe

- Gegeben sei der Filterkern
 - a) Um was für ein Filter handelt es sich (Tiefpass, Hochpass, Bandpass etc.)?
 - b) Filtern Sie das Bild in Abb. 8.1a) mit dem Filterkern. Für welche Kanten (horizontale, vertikale, diagonale) ist er empfindlich?
 - c) Zeigen Sie, dass dieses Filter aus der zweiten Ableitung des Bildes nach x entsteht

$$h(u, v) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

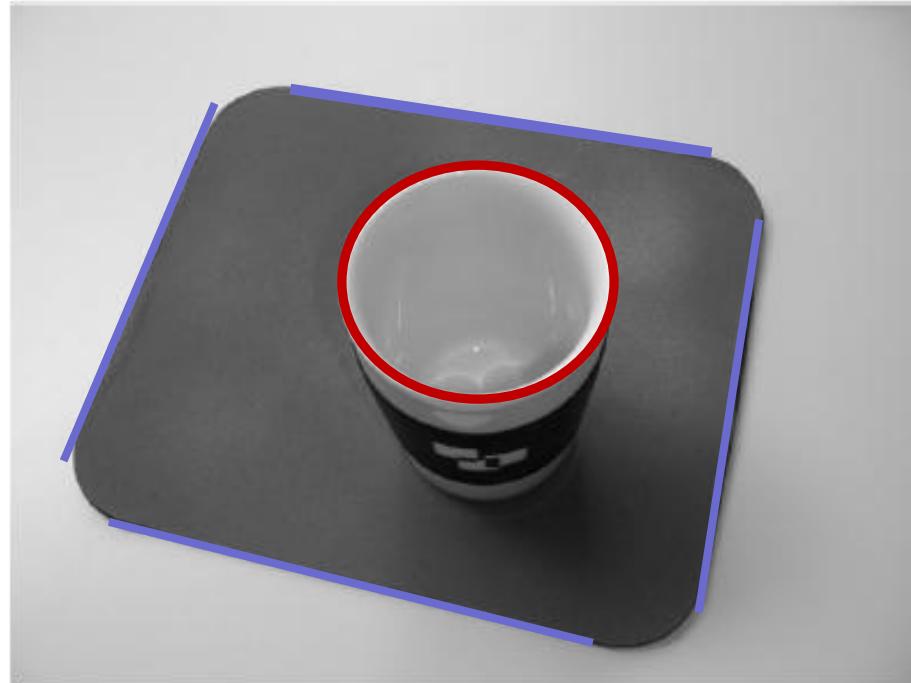


Hinweis: Bei dieser Filterung können negative Grauwerte auftreten. Sie können diese stehen lassen. Pixel ohne Ziffern seien 0.

Hough-Transformation: Erkennung einfacher Kurven (Linie, Kreis, Ellipse)

Problem

- Einfache Formen finden sich häufig in vom Menschen geschaffenen Objekten, wie Linien, Kreise, Ellipsen



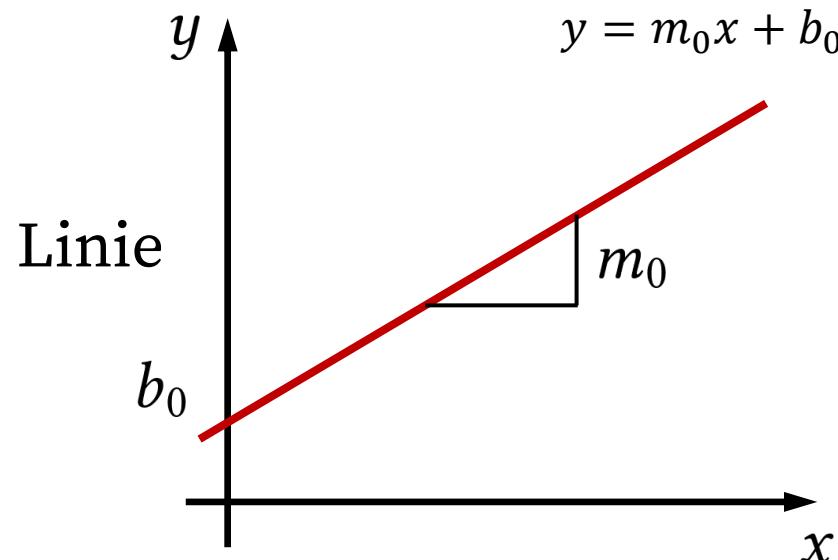
Wie kann ein Computer den Kreis (rot) und die 4 Linien (blau) in diesem Bild finden?

- Wie kann ein Computer die Räder eines Fahrrads in diesem Bild finden?

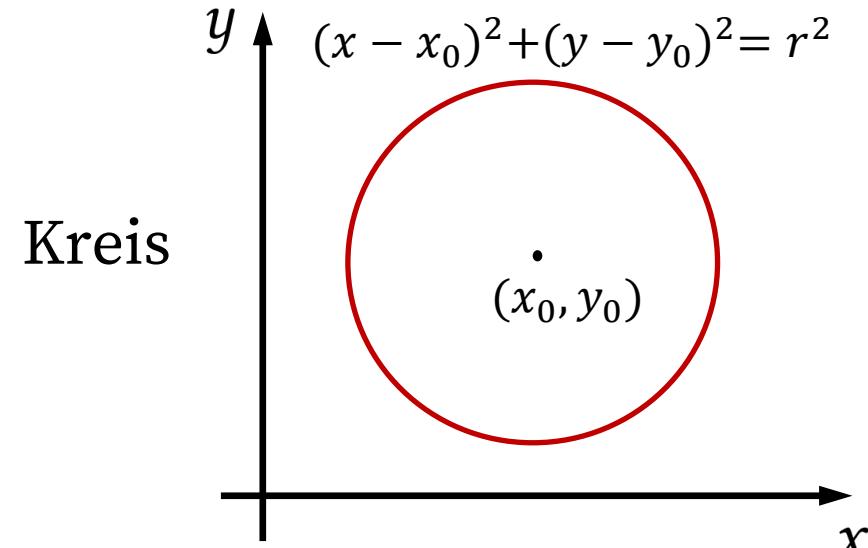


Hough-Transformation

- wird verwendet, um einfache Grenzen zu finden, die durch mathematische Gleichungen mit wenigen (≤ 5) Parametern definiert werden können



2 Parameter: m_0, b_0



3 Parameter: x_0, y_0, r

Hough-Transformation: Erster Schritt

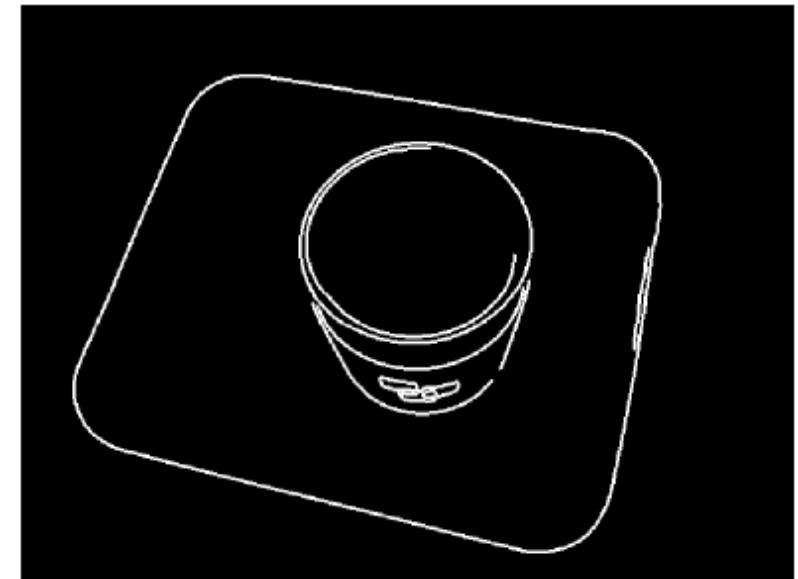
- Hough-Transformation startet mit Kantendetektor und Binarisierung



Kantendetektor
+
Binarisierung



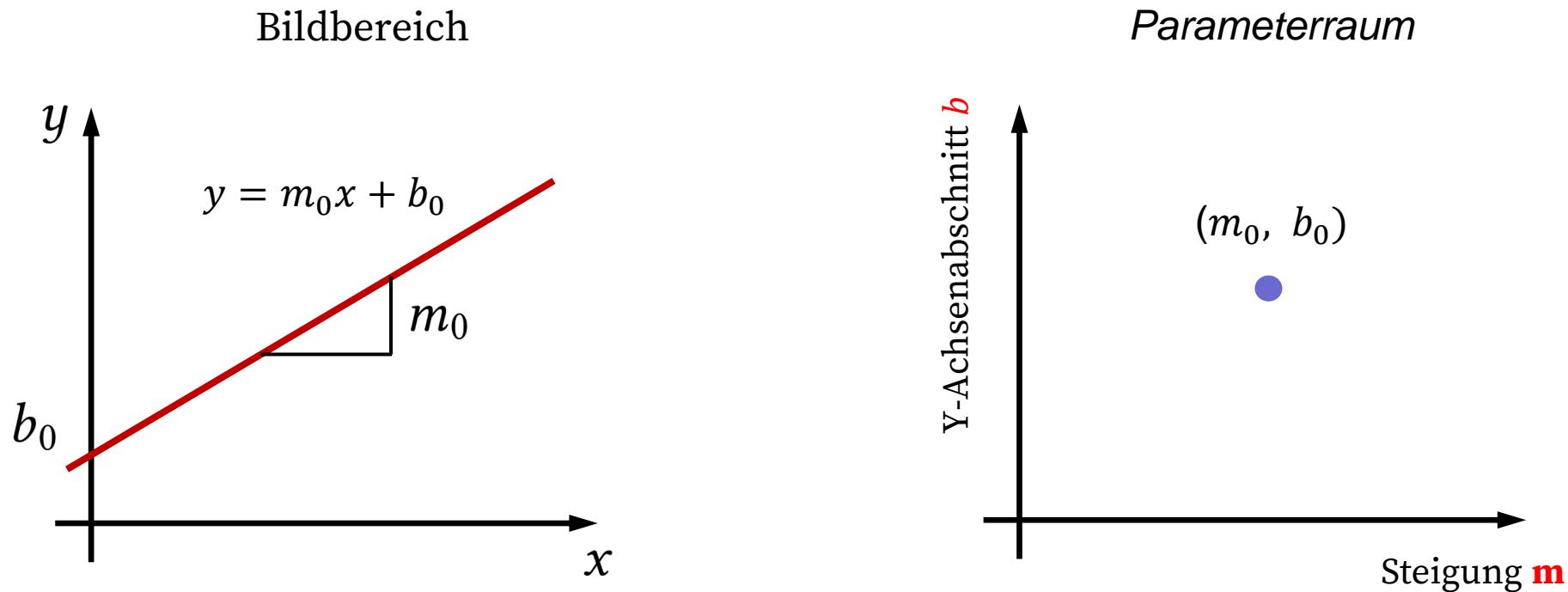
Binärbild mit Kanten



- Für jeden Kantenpunkt (Wert = 1) im Binärbild, wird Hough-Transformation angewendet.

Hough-Transformation

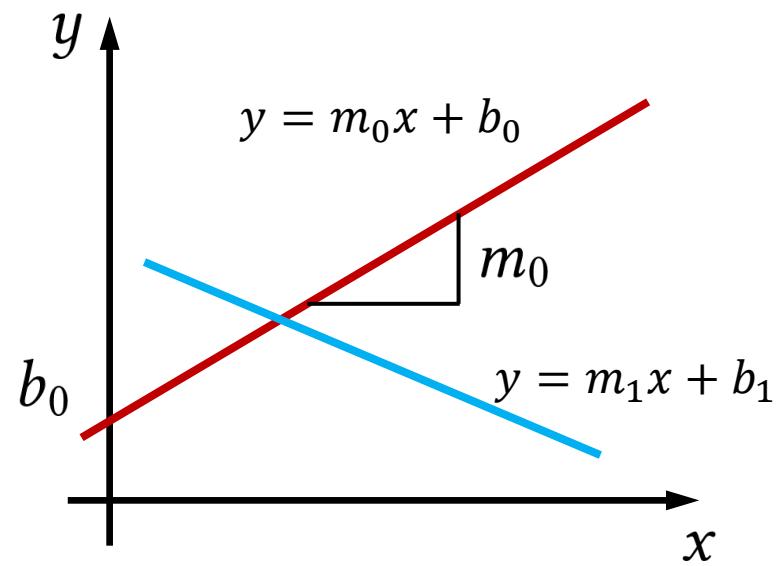
- Grundlagen: die *Grenzen* (z.B. *Linien*) im Bildraum können in einem *Parameterraum* als *Punkte* dargestellt werden



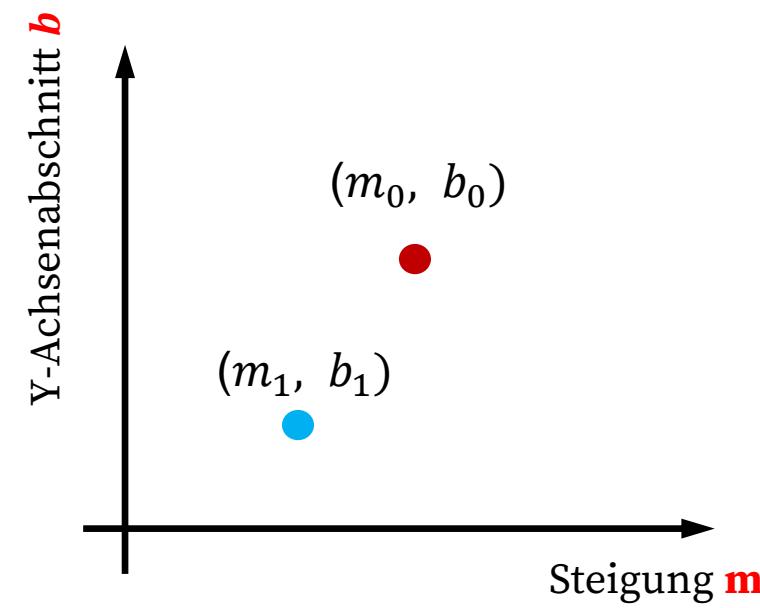
Hough-Transformation – Parameterraum

- *Zwei Linien, dann zwei Punkte*

Bildbereich

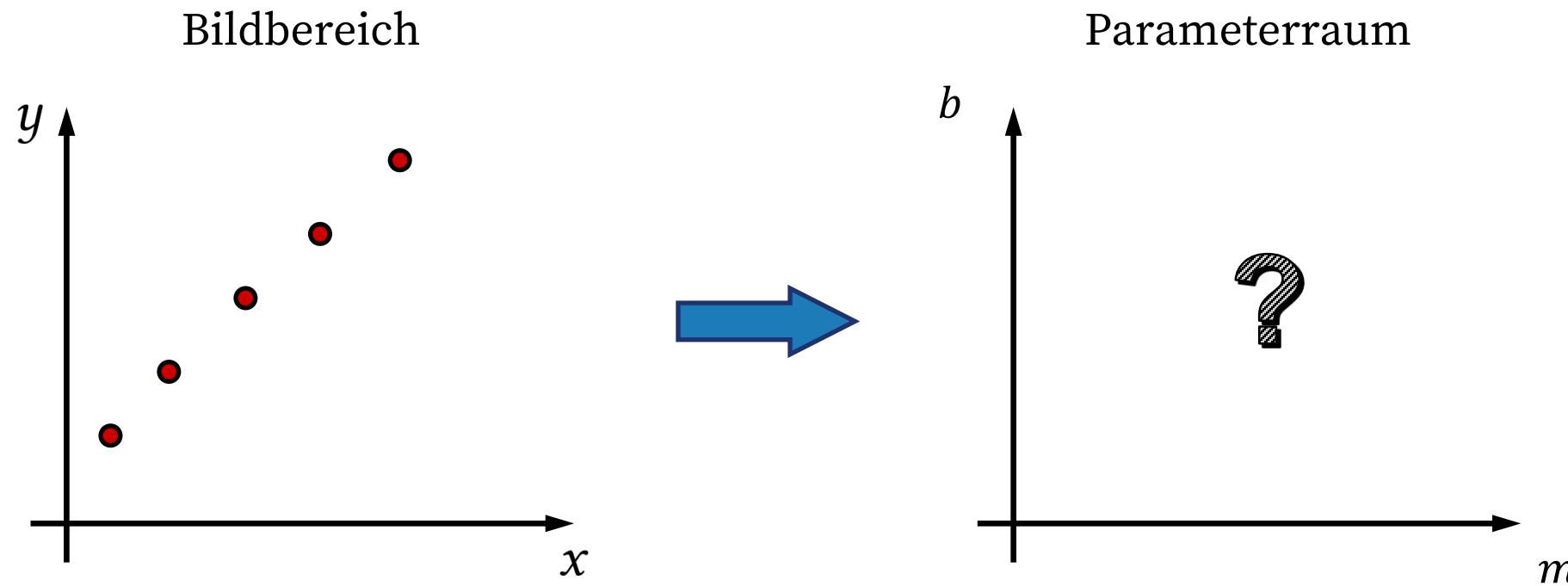


Parameterraum



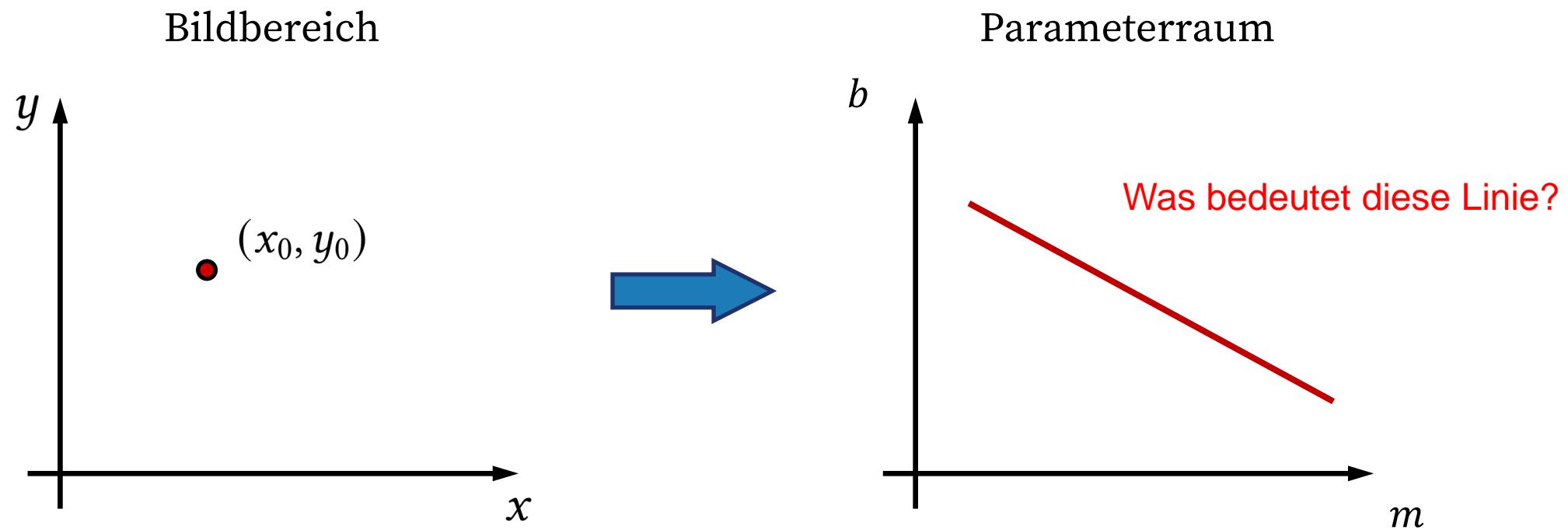
Hough-Transformation – Parameterraum

- Aber in Wirklichkeit haben wir keine Linie im Bild. Wir haben nur viele Punkte.
- Wie finden wir die Parameter der Linie?



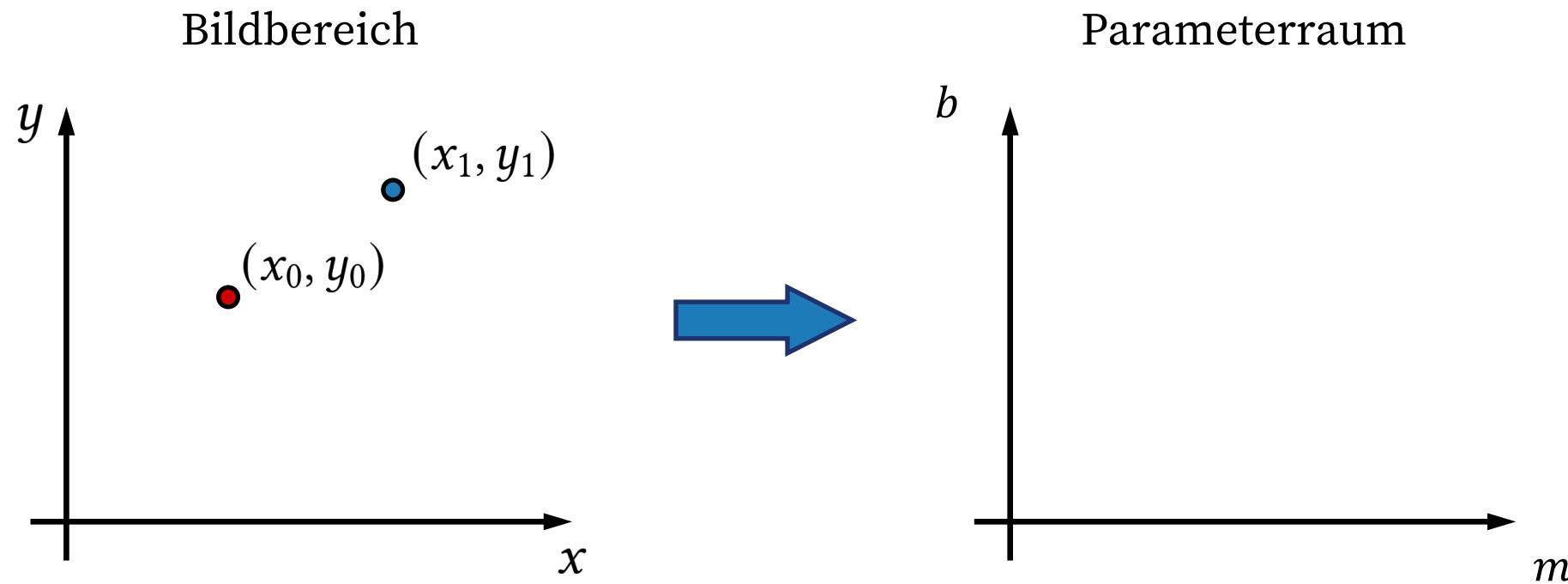
Hough-Transformation

- Ein **Punkt** (x_0, y_0) im Bild wird auf was im Parameterraum abgebildet?
 - Aus $y_0 = m \cdot x_0 + b$ folgt $b = -x_0m + y_0$
 - Und damit eine Linie im Parameterraum



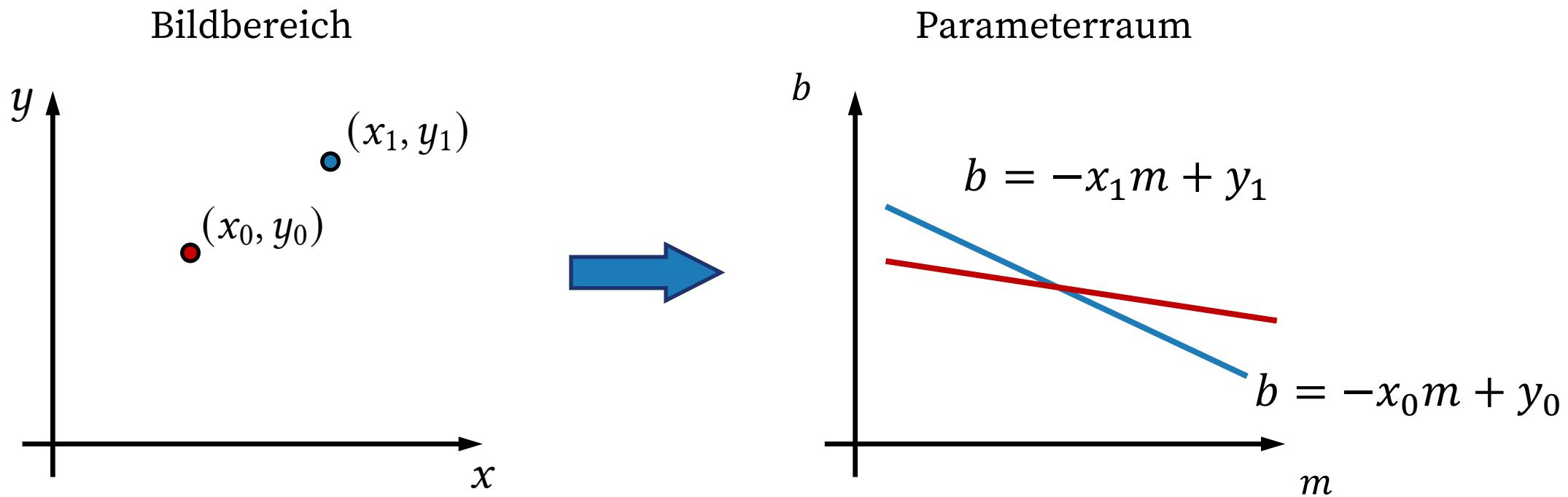
Hough-Transformation

- Mehrere Punkte definieren mögliche Linien



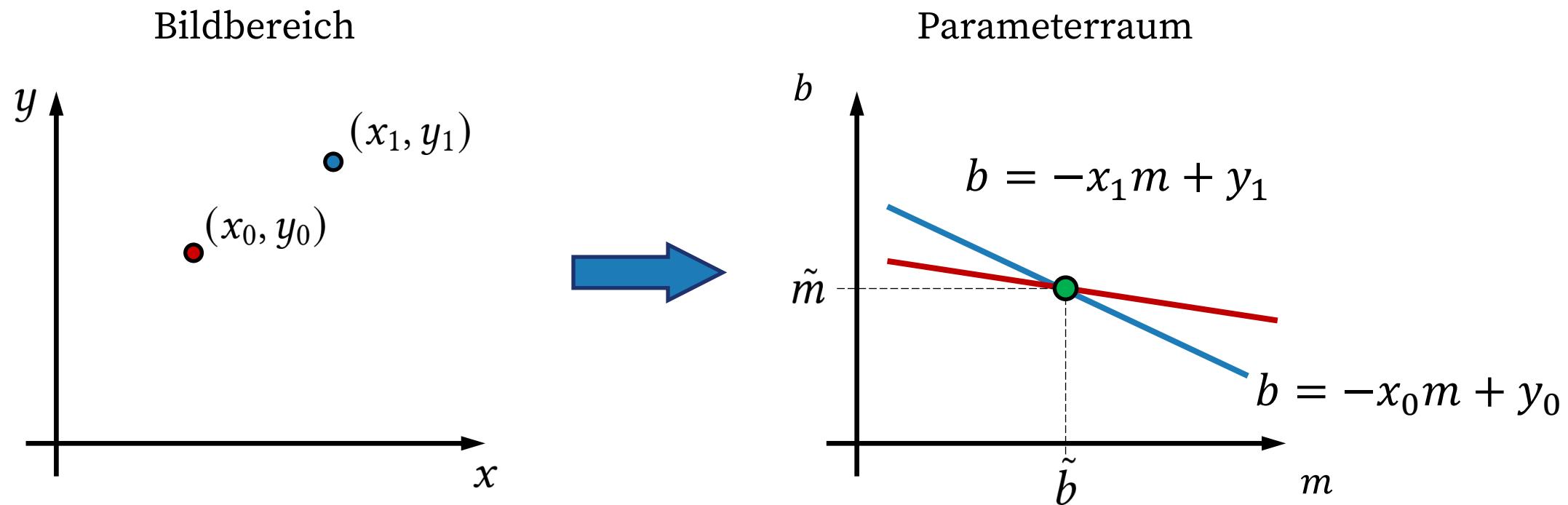
Hough-Transformation

- Mehrere Punkte definieren mögliche Linien
 - Der Schnittpunkt im Parameterraum bildet die gerade im Bildbereich ab



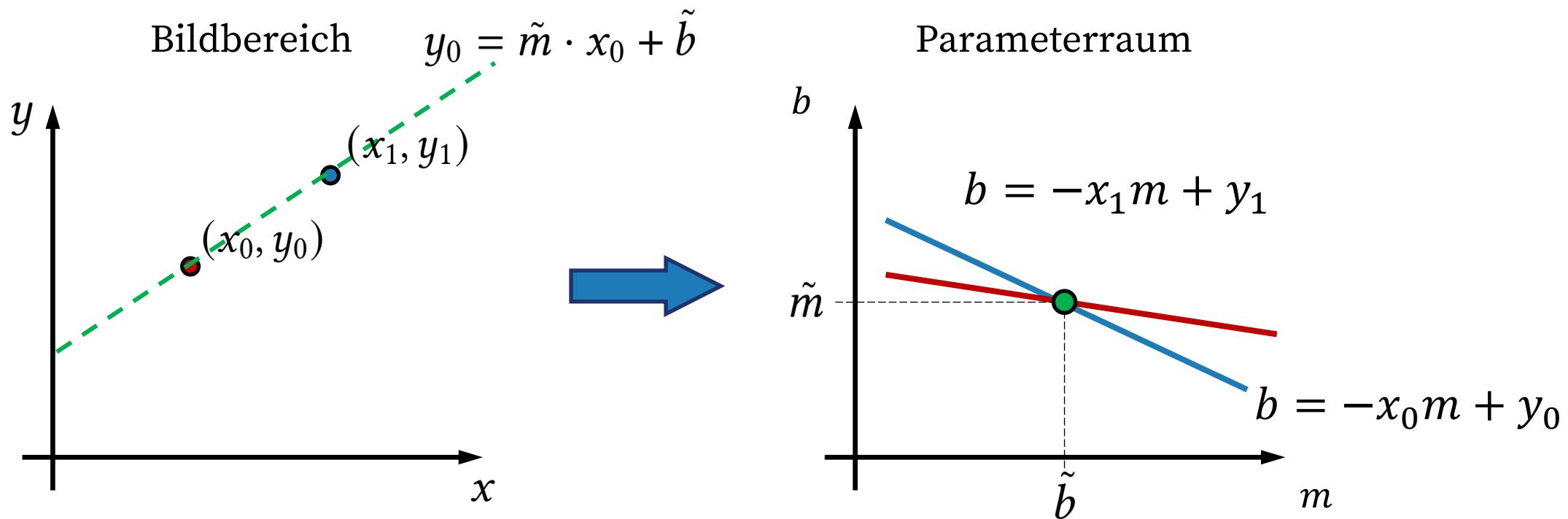
Hough-Transformation

- Mehrere Punkte definieren mögliche Linien
 - Der Schnittpunkt im Parameterraum bildet die gerade im Bildbereich ab



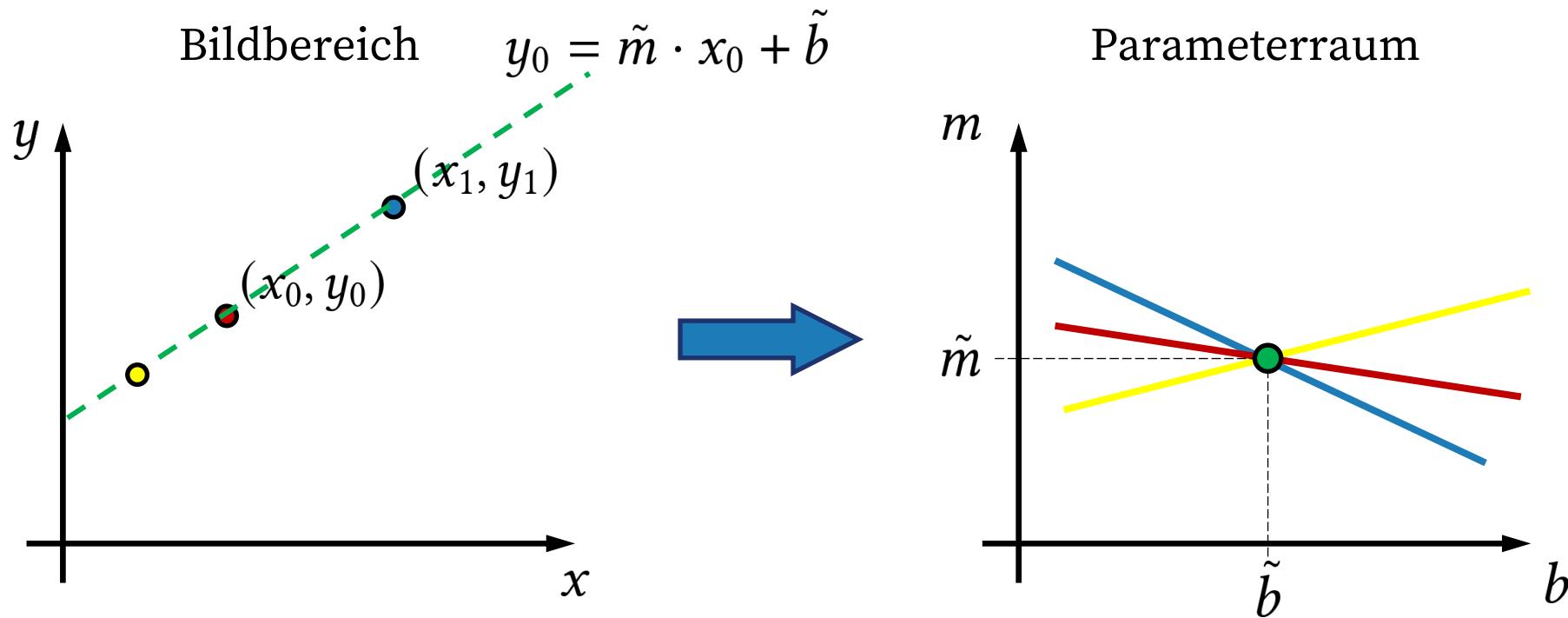
Hough-Transformation

- Mehrere Punkte definieren mögliche Linien
 - Der Schnittpunkt im Parameterraum bildet die gerade im Bildbereich ab



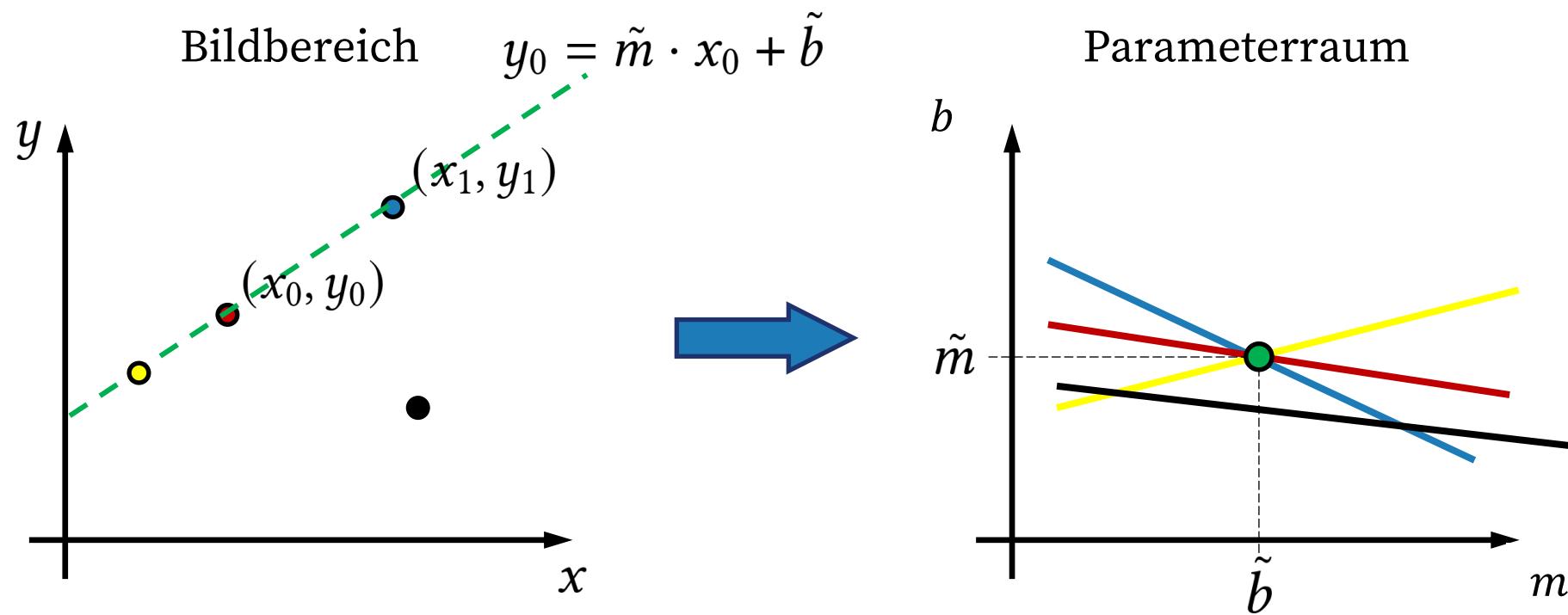
Hough-Transformation

- Mehrere Punkte definieren mögliche Linien
 - Der Schnittpunkt im Parameterraum bildet die gerade im Bildbereich ab



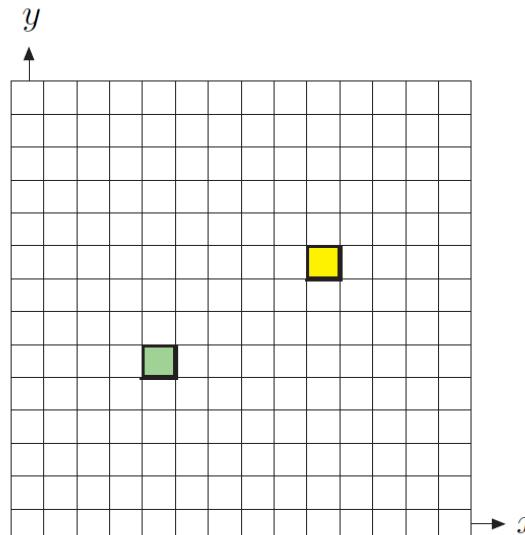
Hough-Transformation

- Wenn ein Punkt (schwarz) nicht auf der Linie liegt, kein Schnittpunkt

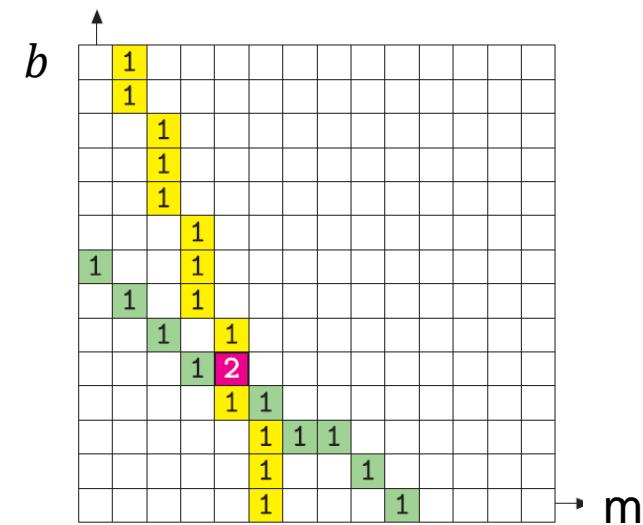


Hough-Transformation Algorithm

1. Diskretisieren den Parameterraum
2. Erstellen ein akkumulierendes Array $A(m, b)$ für den Parameterraum
3. Für jeden Punkt (x, y) , $\mathbf{A}(m, b)++$, wenn (m, b) auf der Linie $b = -xm + y$ liegt
4. das Maximum von $A(m, b)$ finden



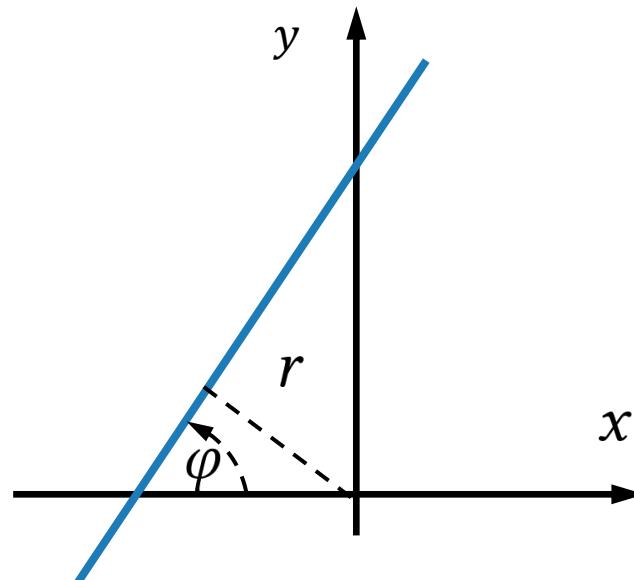
Bildbereich



$A(m, b)$ für Parameterraum

Hough-Transformation – Bessere Parametrisierung

- Probleme mit dem Geraden-Parameterraum
 - Keine begrenzten Parameterwerte für \mathbf{m} und \mathbf{b}
- Die Lösung wird über eine alternative Parametrierung in der Hesse Normalform erreicht $\varphi: [0, \pi]$



Bildbereich (x, y)

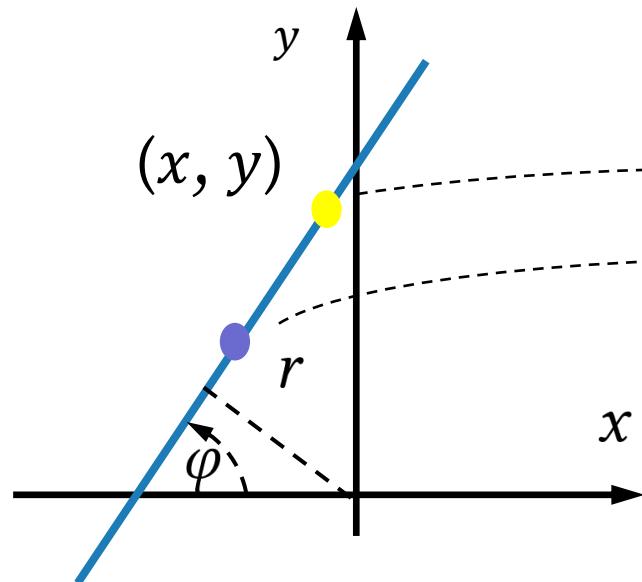
$$x \sin(\varphi) - y \cos(\varphi) + r = 0$$

- $\varphi: [0, \pi]$

- $r: [-r_{max}, r_{max}]$

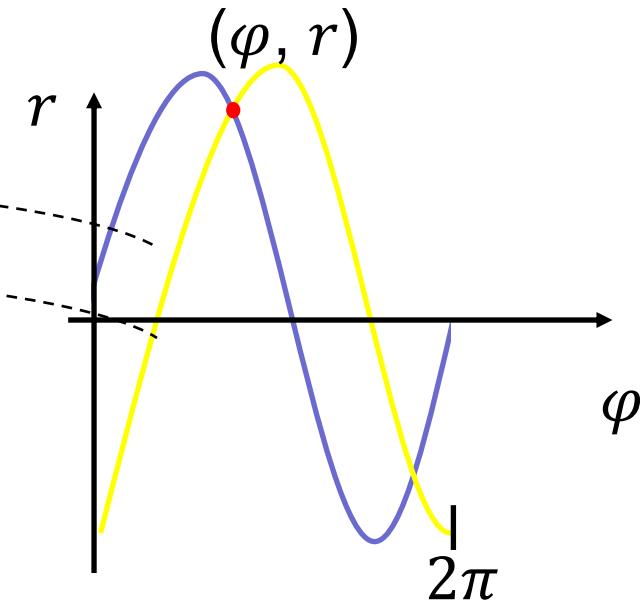
Hough-Transformation – Bessere Parametrisierung

$$x \sin(\varphi) - y \cos(\varphi) + r = 0$$



Bildbereich (x, y)

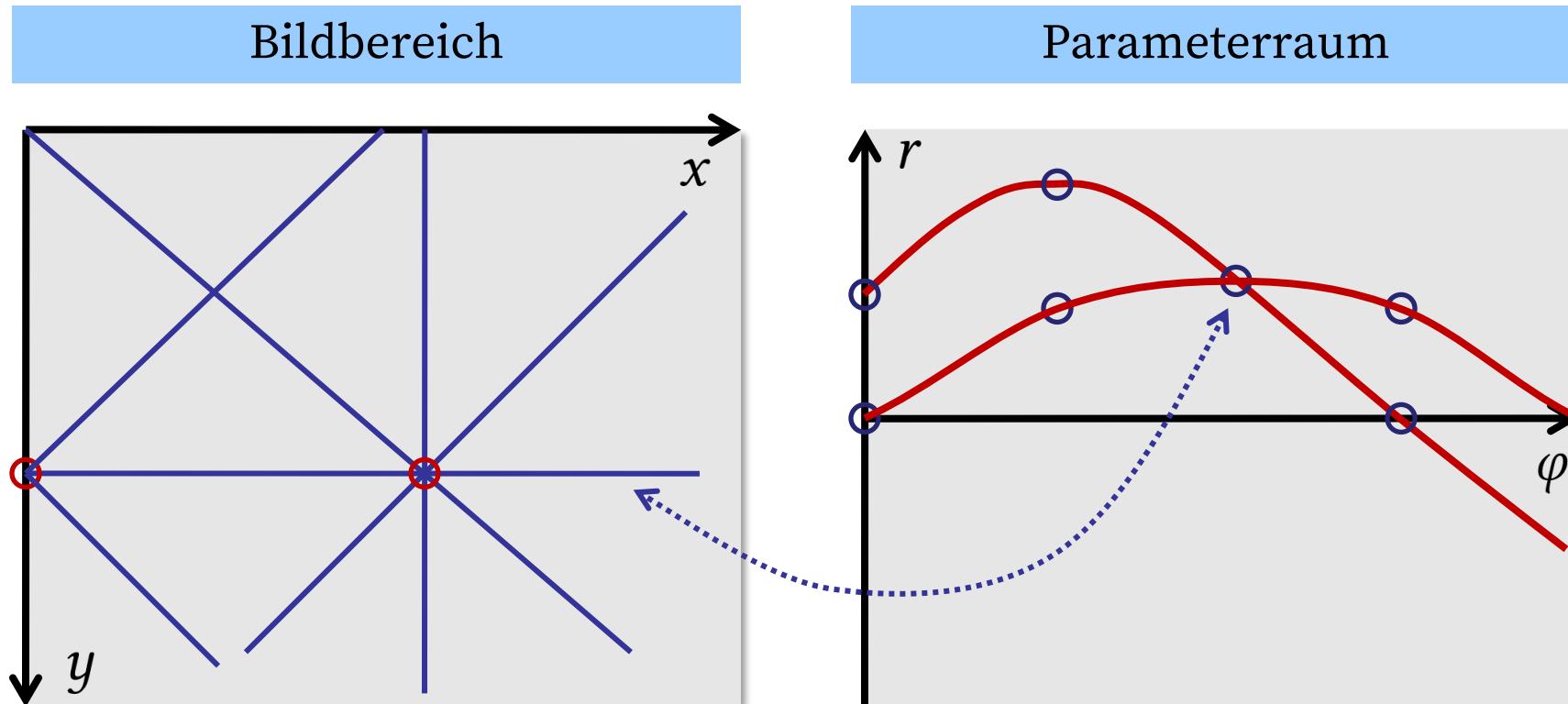
$$x \sin(\varphi) - y \cos(\varphi) + r = 0$$



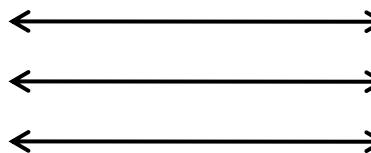
Parameterraum (φ, r)

Hough-Transformation - Bessere Parametrisierung

- Die verbesserte Parametrisierung führt zu leicht abgewandeltem Verhalten



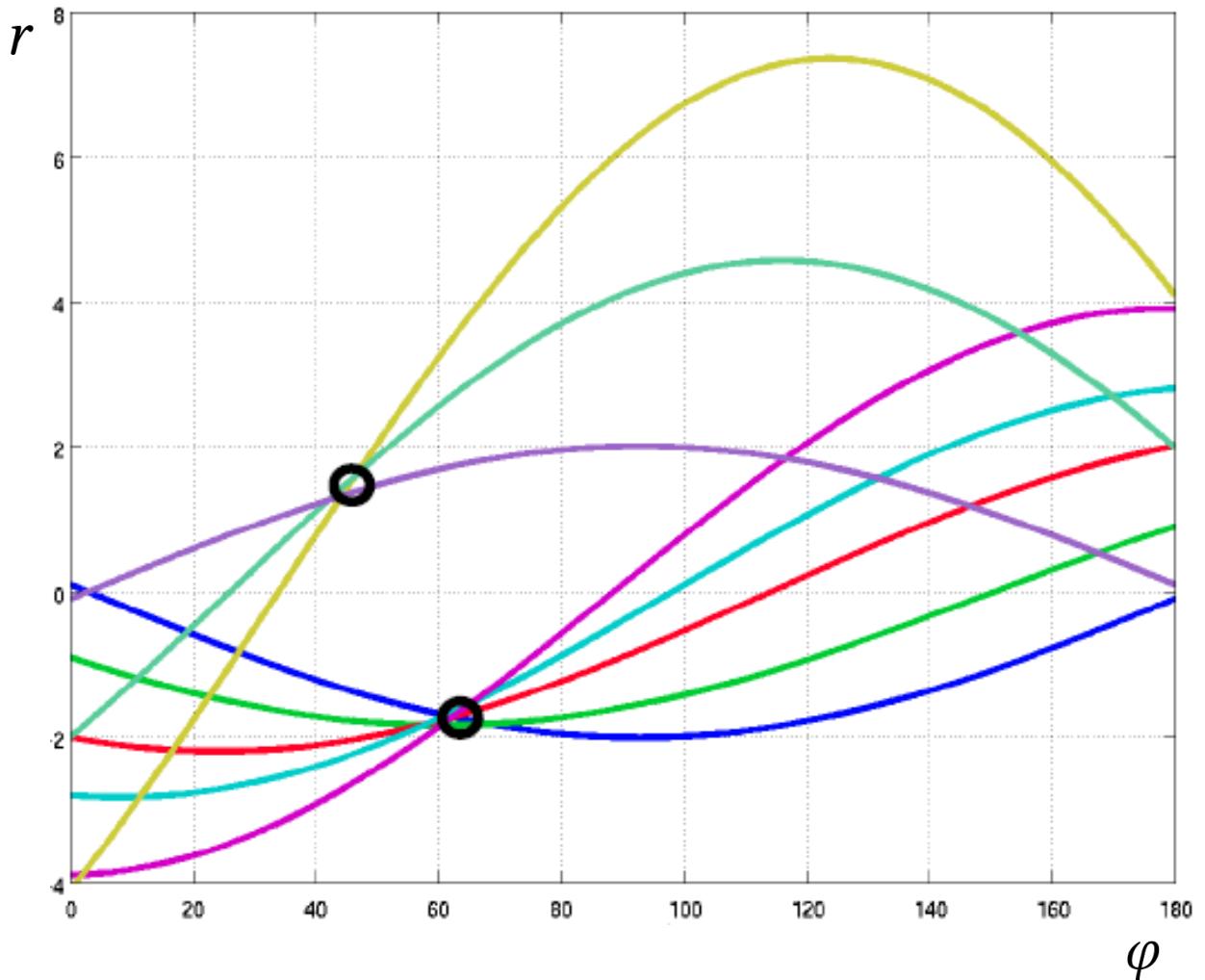
Linien
Punkte
Punkte auf einer Linie



Punkte
Sinuskurven
Sich schneidende Sinuskurven

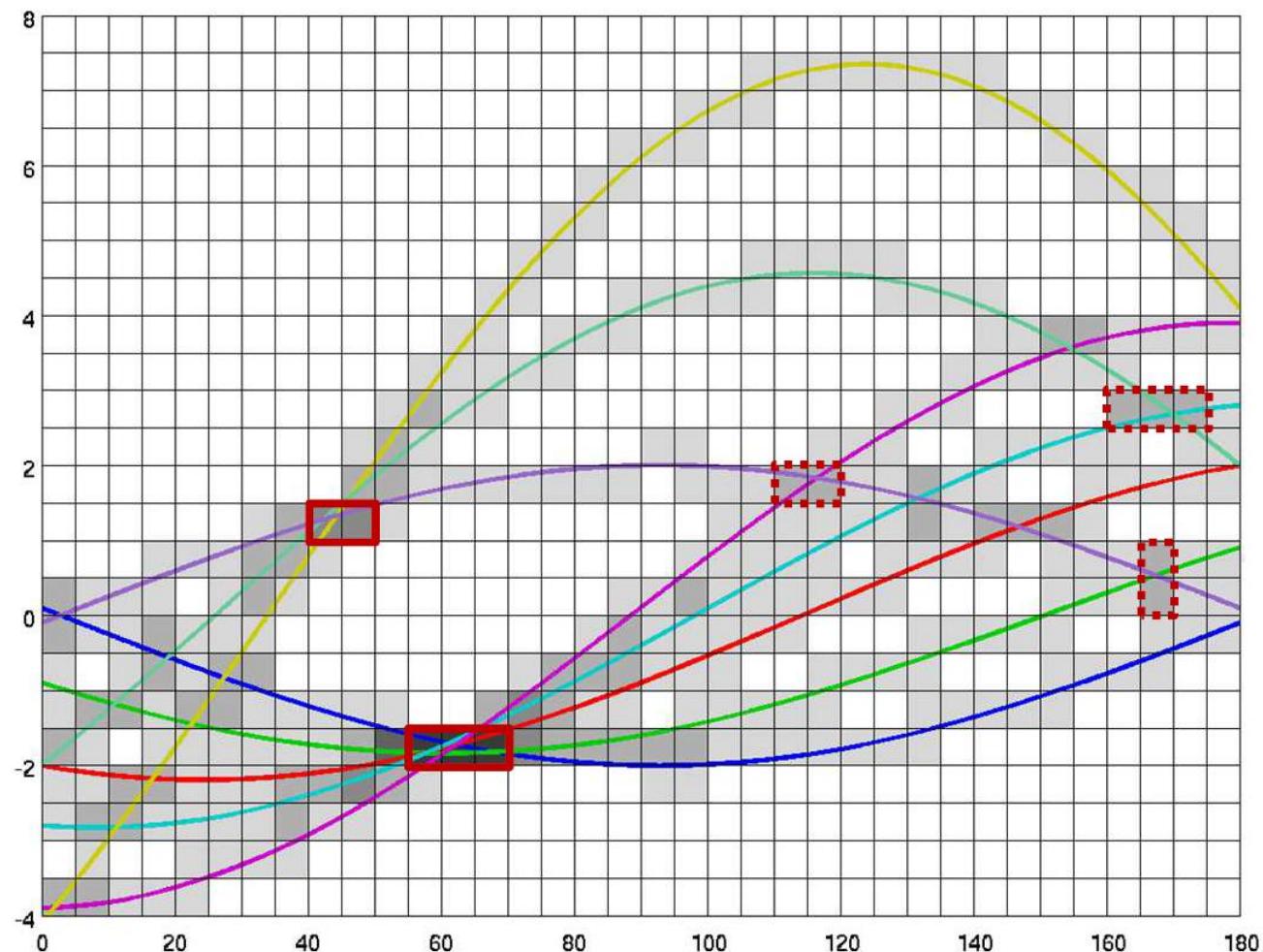
Hough-Transformation – Realisierung

- In der Anwendung:
 - Typischerweise, es gibt nicht nur einen exakten Schnittpunkt im Parameterraum
 - Es können mehrere Lienen vorhanden sein, die sich überlagern



Hough-Transformation Algorithm

- Es müssen Bereiche mit hoher Dichte im Parameterraum gefunden werden
 - Es wird ein diskretes Gitter mit Akkumulatorzellen genutzt
 - Für jede Gitterzelle werden die Sinuskurven gezählt, welche sie schneiden
 - Die lokalen Maxima im Akkumulator sind die Geradenparameter



Hough-Transformation – Ablauf

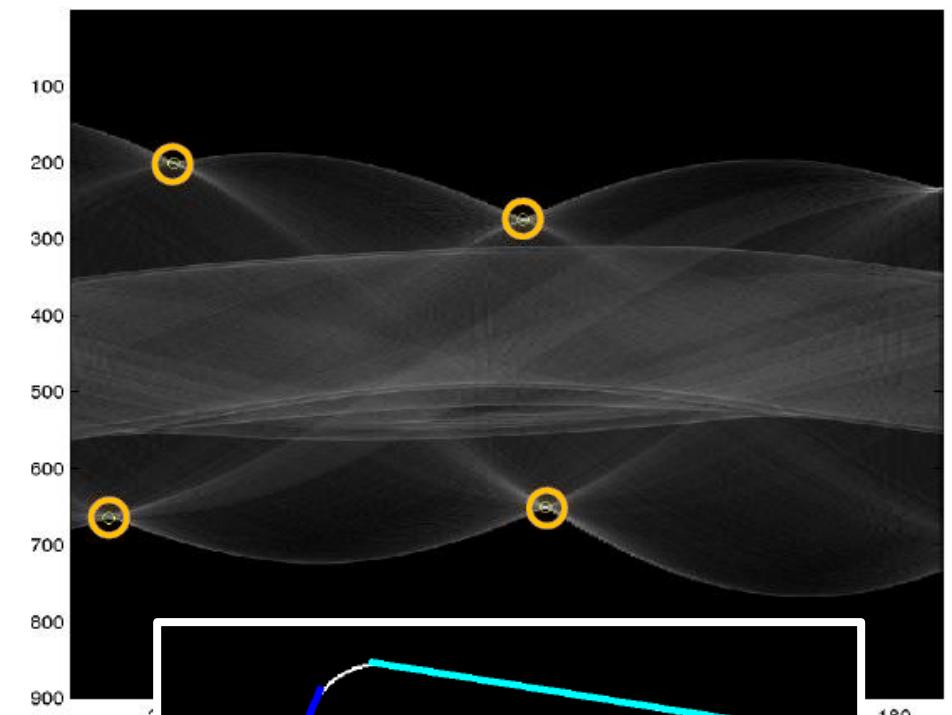
0. Originalbild



1. Binärbild mit Kanten
-> Canny-Kantendetektor

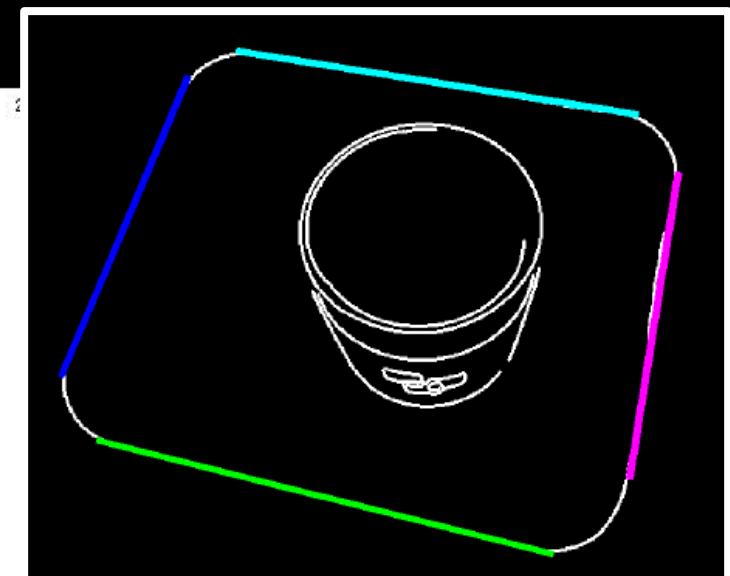


2. Transformation in
den Parameterraum



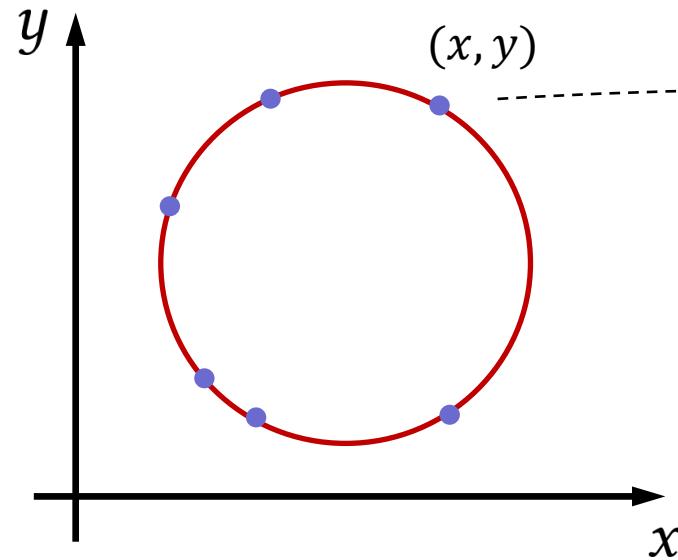
3. Bestimmung der
lokalen Maxima

4. Linien der Maxima
extrahieren



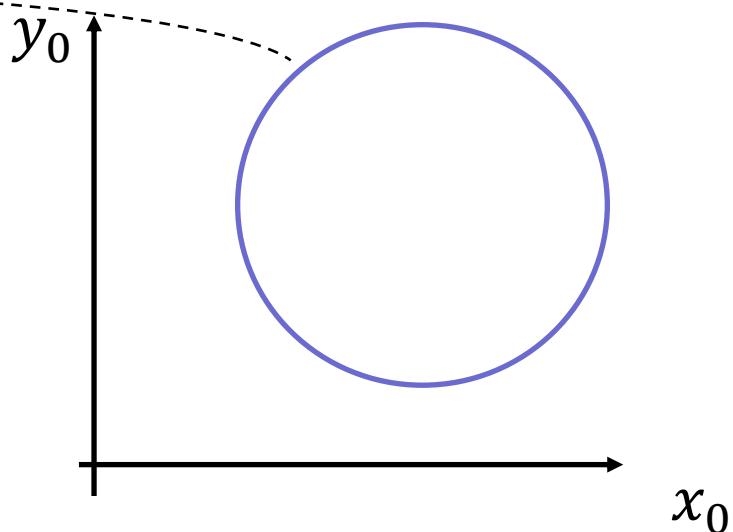
Hough-Transformation – Für Kreis

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$



Bildraum (x, y)

$$(x_0 - x)^2 + (y_0 - y)^2 = r^2$$

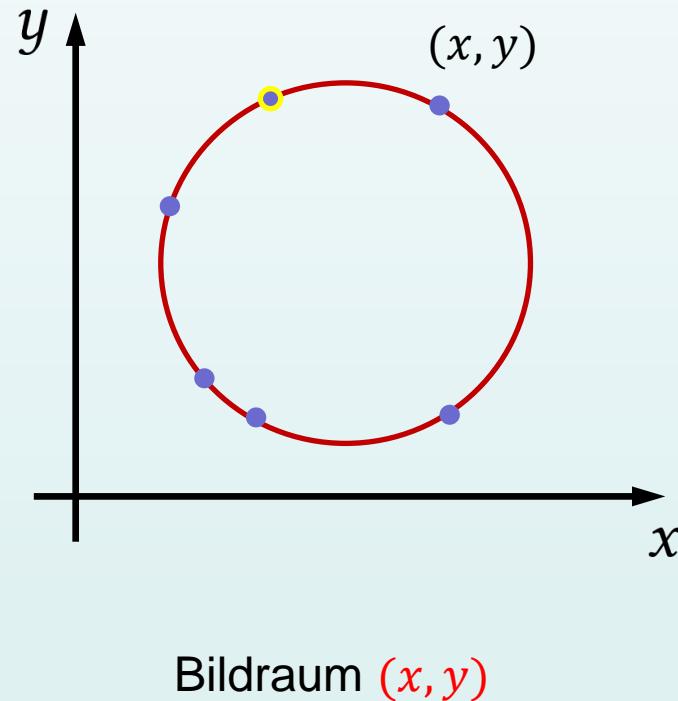


Parameterraum (x_0, y_0)

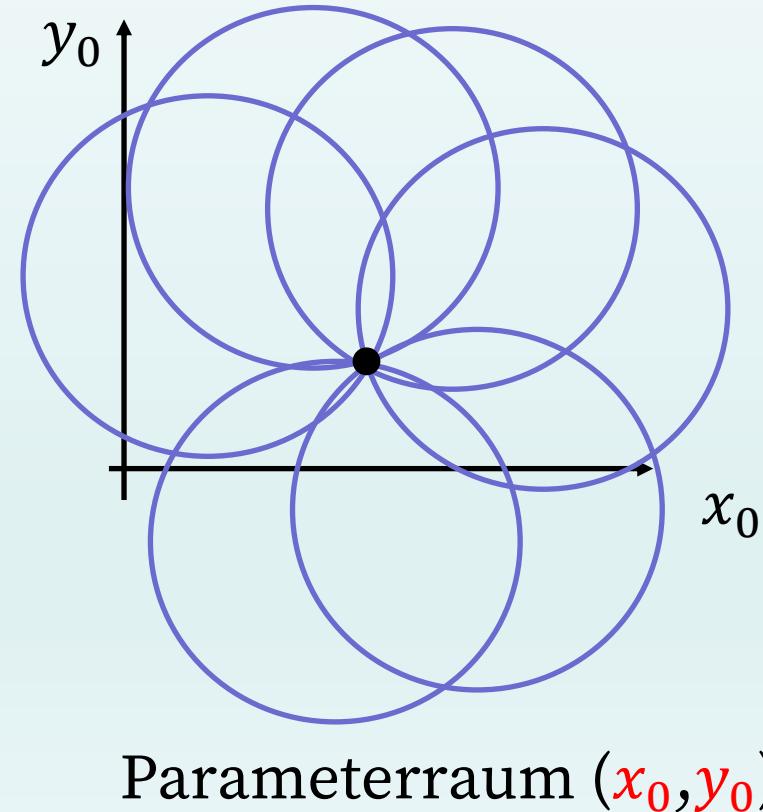
- angenommen, dass r bekannt ist

Hough-Transformation – Für Kreis

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$



$$(x_0 - x)^2 + (y_0 - y)^2 = r^2$$



Hough-Transformation: Fazit

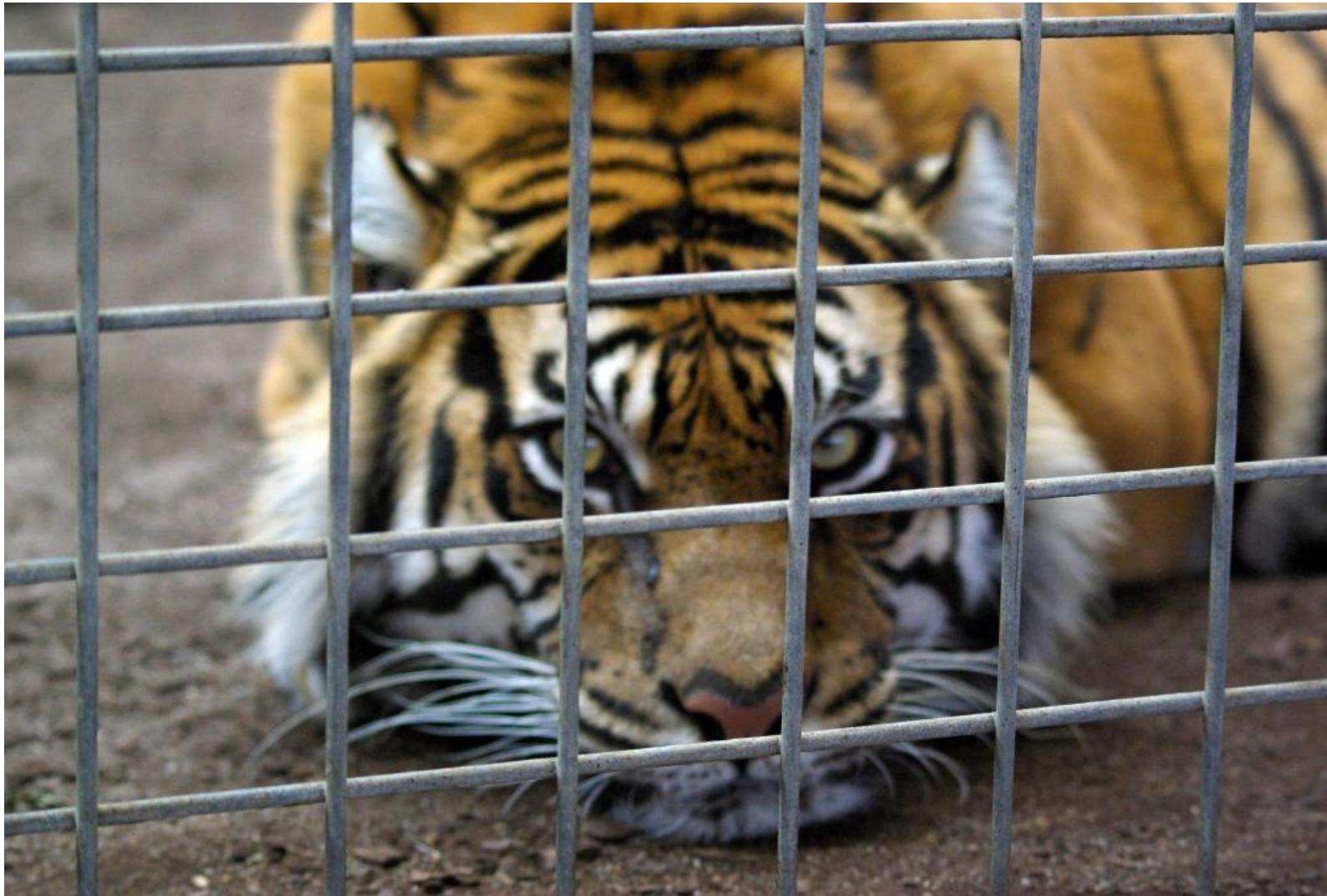
- Grundlegend besteht der Algorithmus aus folgenden Schritten
 1. Kantenerkennung
 2. Binarisierung
 3. Vorverarbeitung (ggf. Kantenverdünnung)
 4. Transformation in den Parameterraum
 5. Maximasuche im Akkumulator und Liniengewinnung
- 
- z.B. Canny-Kantendetektor

Hough-Transformation – Eigenschaften

- Ergebnisse hängen von Größe und Auflösung des Akkumulators ab
- Die lokalen Maxima zu bestimmen ist in der Praxis oftmals kompliziert
- Die Richtung der Gradienten wird ignoriert
- Der Akkumulator wird bei natürlichen Bildern „geflutet“

- Erweiterungen der Hough-Transformation
 - Kann für alle **parametrisierbar** Kurven eingesetzt werden
 - Leider sinkt die Effizienz deutlich mit Anzahl der Parameter
 - **Randomisierte Hough-Transformation** (reduziert Suchzeit, Ergebnis sind Linien-Segmente)
 - **Generalisierte Hough-Transformation**

Hough-Transformation – Beispiel

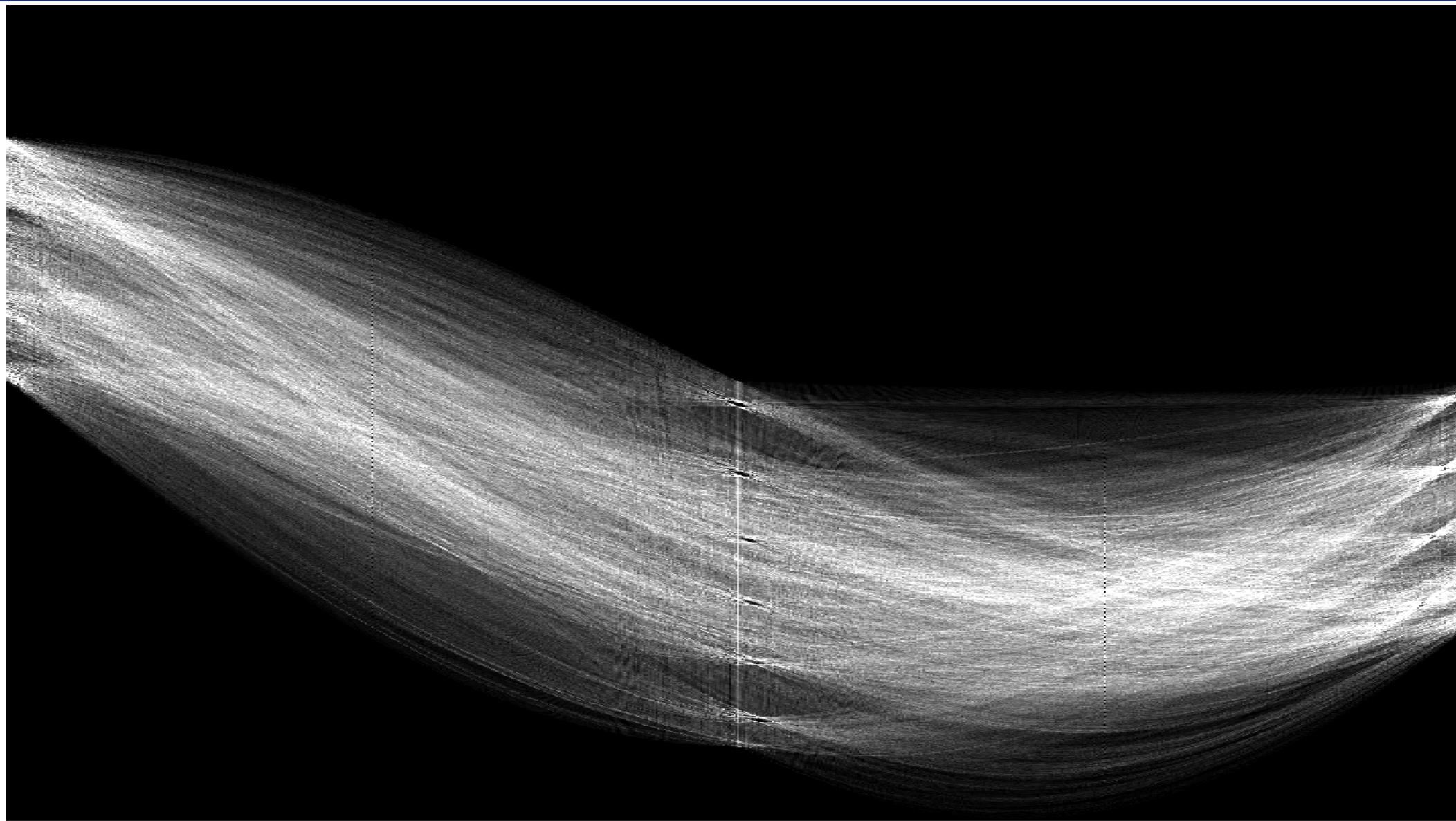


Hough-Transformation – Beispiel

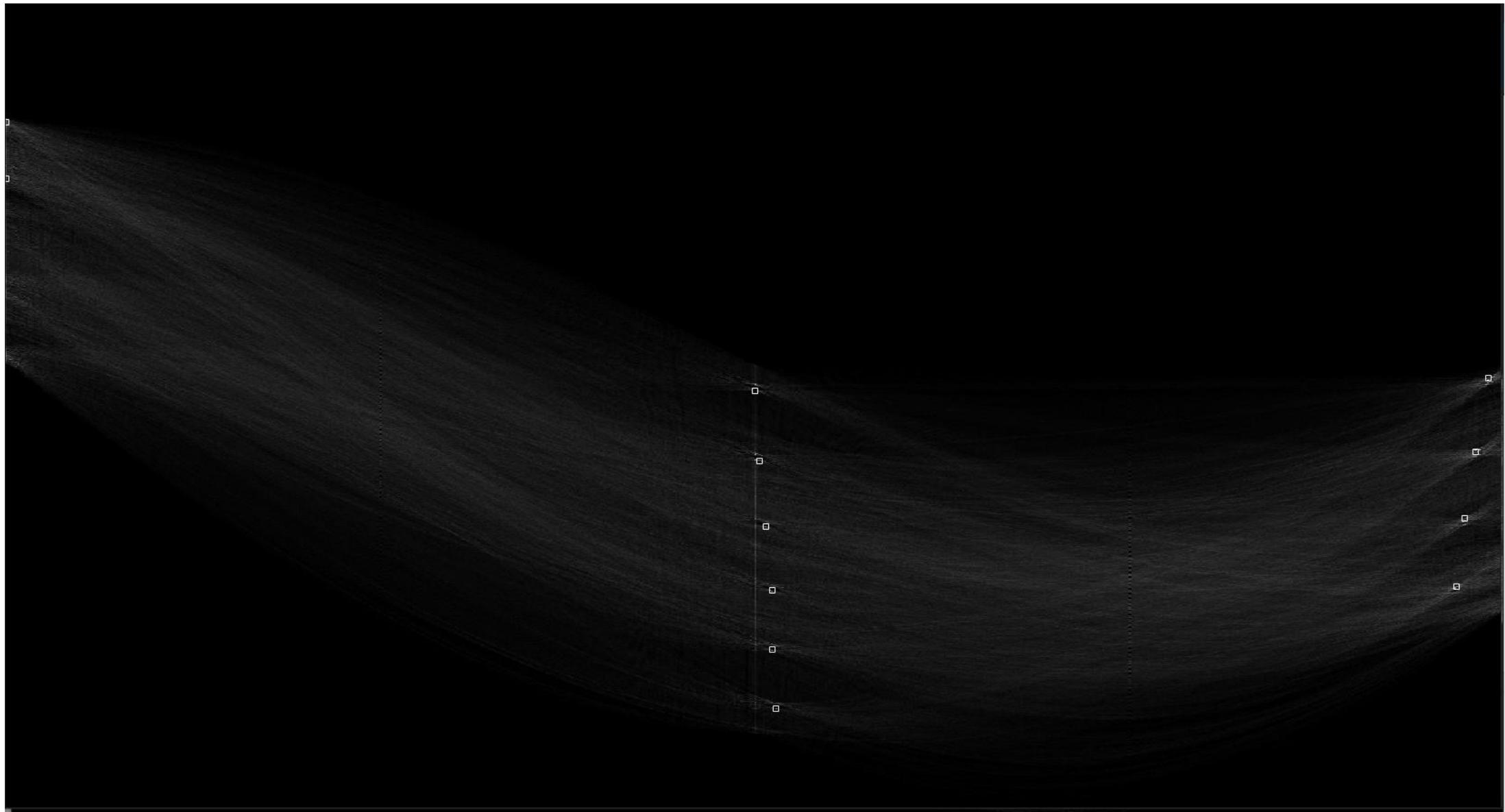


Binärbild

Hough-Transformation – Beispiel

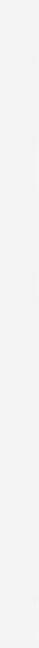
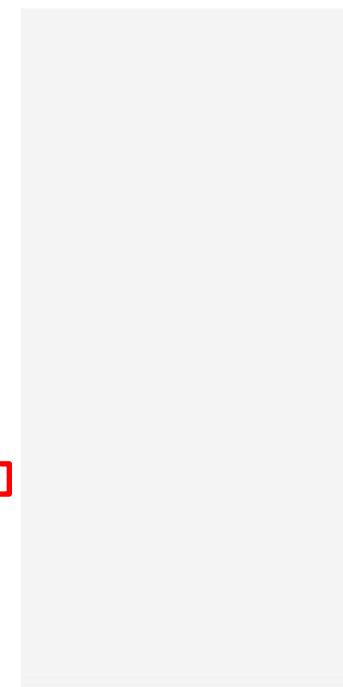
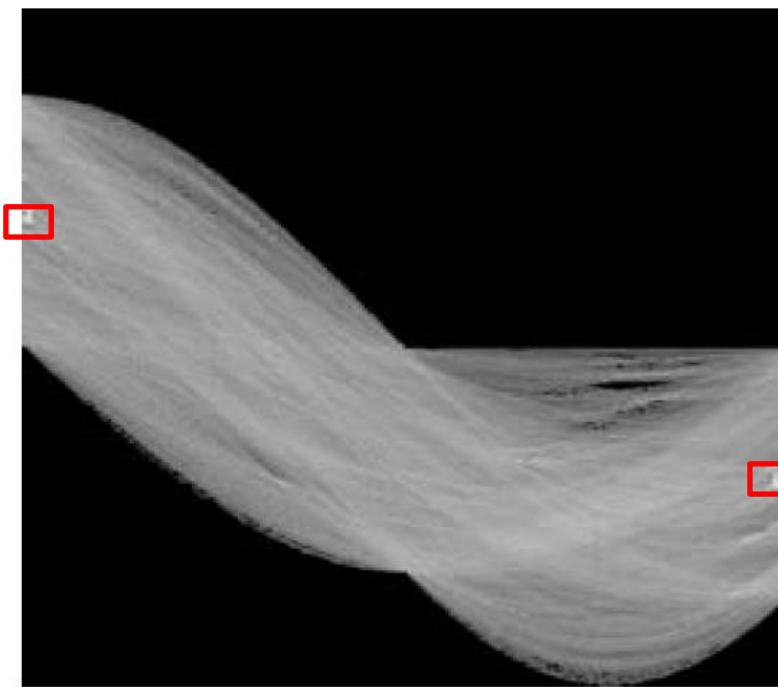
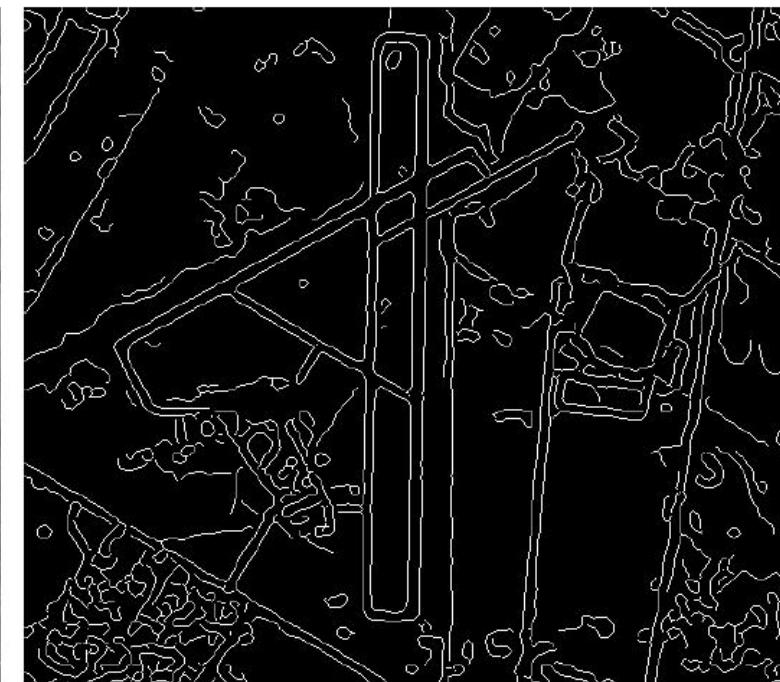


Hough-Transformation – Beispiel



Hough





Zusammenfassung

- Wo kann man mehr erfahren?
 - Burger, Kapitel 5, 6
 - Gonzalez, Kapitel 10
 - Szeliski, Kapitel 7



Referenz

- [1] Burger, Burge, Digitale Bildverarbeitung: Eine algorithmische Einführung, 3rd ed., 2015
- [2] Gonzalez, Woods, Digital Image Processing, 4th ed., 2017
- [3] Szeliski, Computer Vision: Algorithms and Applications, 2011