1.4 Kantendetektion

a.) Definieren Sie den Begriff Gradient und erläutern Sie welche Informationen er bezüglich einer potentiellen Kante enthält.

- Der Ergebnis Vektor der partiellen Ableitungen entlang einer der Koordinatenrichtungen wird als Gradient bezeichnet. D.h. verfolge die Intensitätsänderungen entlang einer Zeile oder Spalte
- Es ist eine Möglichkeit, um Kanten zu detektieren.
- Dieser ist rotationsinvariant
- Der Betrag des Gradienten entspricht der Stärke des Anstiegs

b.) Die am häufigsten verwendeten Kanten-Operatoren (Prewitt, Sobel) sind separabel. Erläutern Sie was diese Eigenschaft bedeutet und welche Auswirkung dies bei der Implementierung als Filter hat.

- Das bedeutet das man einen 2D-Filter der aufs Bild angewendet werden soll in zwei 1D-Filter separieren kann und die dann hintereinander auf das Bild anwendet.
- Das sorgt dafür das wir eine Art Caching Effekt haben, somit sparen wir uns mathematische Operationen.
- Je größer der Filter ist desto größer wird der Effekt.

c.) Wie kann man mit der 2ten Ableitung Kanten detektieren? Vor-/Nachteile

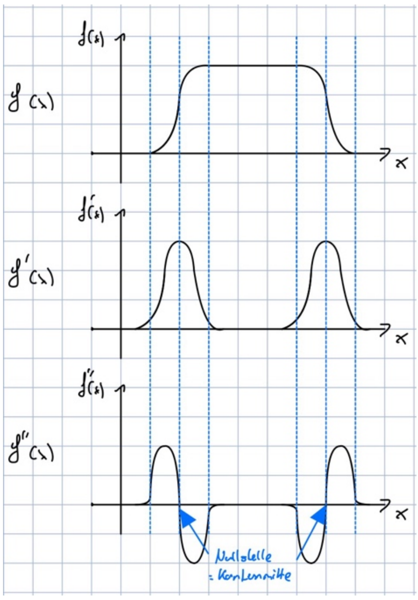
- Die zweite Ableitung ist besser geeignet, wenn es langsame Helligkeitsänderungen gibt. Die zweite Ableitung ist viel empfindlicher
- VT: Kantendicke wird eliminiert!
- NT: etwas aufwändiger

d.) Erläutern Sie kurz das Grundprinzip von Unsharp Masking (USM). Welchen Nachteil hat das Verfahren?

- Für die Kantenschärfung, soll ein künstlicher Mach-Band Effekt hinzugefügt werden
  1. Bild glätten durch z.B. Gaus-Filter --> für die Kantenschärfung
  2. Subtraktion der geglätteten Versionen des Originalbilds
     - M: Maske, I: Bild, H: Filter
     - M = I - I*H
  3. Addition der gewichteten Maske zum Originalbild
     - A: Schärfungsgrad
     - I‘ = I + aM
- NT: Bereiche ohne Kanten verstärktes Rauschen und Kanten können zu scharf werden

e.) Erläutern Sie anhand einer Skizze wie mit Hilfe einer geschätzten zweiten Ableitung eine Kantendetektion realisiert werden kann. In welchen Fällen ist diese Vorgehensweise besser geeignet als ein Sobel- oder Prewitt-Operator.

- Die zweite Ableitung ist besser geeignet, wenn es langsame Helligkeitsänderungen gibt. Da diese viel empfindlicher ist



f.) Erläutern Sie anhand einer ausführlichen Berechnung (also incl. Formeln) wieso die Laplace-Schärfung ein Spezialfall der USM Schärfung darstellt.

- Laplace:

$$H^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \overset{\text{Glättung}}{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}} - 5 \overset{\text{besser gegen Rauschen}}{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}} = 5 \cdot (\tilde{H}^L - \delta),$$

- 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

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

Canny

a.) Geben Sie die strukturelle Funktionsweise des Canny-Algorithmus gemäß der Vorlesung als Pseudocode an

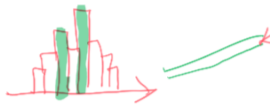
Gesamt-Algorithmus:

```
CannyEdgeDetector(I, σ, t_hi, t_lo)
    Input: I, a grayscale image of size M×N;
    σ, scale (radius of Gaussian filter H^{G,σ});
    t_hi, t_lo, hysteresis thresholds (t_hi > t_lo).
    Returns a binary edge image of size M×N.
    Ī ← I * H^{G,σ}   geglättet                 ▷ blur with Gaussian of width σ
    Ī_x ← Ī * [−0.5 0 0.5]   } Ableitung geschätzt   ▷ x-gradient
    Ī_y ← Ī * [−0.5 0 0.5]^T                     ▷ y-gradient
    (M, N) ← Size(I)
    Create maps:
        E_mag : M×N ↦ ℝ      } Speicher          ▷ gradient magnitude
        E_nms : M×N ↦ ℝ      } allokiert         ▷ maximum magnitude
        E_bin : M×N ↦ {0,1}                     ▷ binary edge pixels
    for all image coordinates (u, v) ∈ M×N do
        E_mag(u,v) ← [Ī_x²(u,v) + Ī_y²(u,v)]^{1/2}
        E_nms(u,v) ← 0
        E_bin(u,v) ← 0
```

Schritt 1: Vorverarbeitung

erste letzte Pixel entfernt keine Randbehandlung

```
    for u ← 1,...,M−2 do
        for v ← 1,...,N−2 do
            d_x ← Ī_x(u,v),  d_y ← Ī_y(u,v)
            s_θ ← GetOrientationSector(d_x, d_y)   Suchrichtung   ▷ Alg. 6.2
            if IsLocalMax(E_mag, u, v, s_θ, t_lo) then            ▷ Alg. 6.2
                E_nms(u,v) ← E_mag(u,v)                 ▷ only keep local maxima
    for u ← 1,...,M−2 do
        for v ← 1,...,N−2 do   weiterer Schwellwert
            if (E_nms(u,v) ≥ t_hi) ∧ (E_bin(u,v) = 0) then
                TraceAndThreshold(E_nms, E_bin, u, v, t_lo)       ▷ Alg. 6.2
    return E_bin.
```



1 Pixel dazwischen erlaubt

b.) Welche Parameter als Eingabe notwendig

- I: Originalbild, sigma, t_{hi}: Schwellwert für lokale Maxima ,t_{lo}: Schwellwert für lokale Minima

c.) Canny-Edge-Kantendetektion und -Verfolgung erklären

1) Vorverarbeitung:
   - Im ersten Schritt wird das Eingangsbild mit einem Gauss-Filter geglättet, um Rauschen zu entfernen. (σ Breite)
   - Dann werden die Gradienten für die x- und y-Achse bestimmt und der Gradientenbetrag sowie die Richtung berechnet.
2) Kantenlokalisirung:
   - Im zweiten Schritt werden lokale Maxima entlang der Gradientenrichtung als Kantenpunkte markiert und die Kantendicke entfernt.
     - ➔ Kantendicke entfernt
3) Kantenselektion und -verfolgung
   - Im letzten Schritt werden zusammenhängende Kantenketten gebildet, unter Verwendung eines Hysterese Schwellwerts.

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

d.) Erläutern Sie die Funktionsweise von getOrientationSector() und die Parameter die die Funktion entgegennimmt.

- Es gibt die Richtung/Orientierung von Kanten an.
- (Es geht darum, die Gradientenrichtungen der Pixel in verschiedene Sektoren einzuteilen.)
- dx: Gradient x-Richtung, dy: Gradient y-Richtung

e.) Schreiben Sie den Pseudocode für getOrientationSector()

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. Konstant

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

if $d'_y < 0$ **then**

$\quad d'_x \leftarrow -d'_x, \quad d'_y \leftarrow -d'_y$ $\qquad \triangleright$ mirror to octants $0, \ldots, 3$

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

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

-

f.) Erläutern Sie die Funktionsweise von isLocalMax() und die Parameter, die die Funktion entgegennimmt.

- Es bestimmt, ob ein gegebener Pixel ein lokales Maximum auf einer Kante darstellt.
- Es geht darum, den Gradientenwert des Pixels mit dem Gradientenwert der benachbarten Pixel zu vergleichen. (Alle weiteren Elemente werden auf 0 gesetzt)

- E_mag: Einträge der Gradienten-Beträge
- t_lo: Schwellwert für lokale Minima
- sg, orientation: Richtung des Gradienten. Kriegt man von GetOrientationSector()
- u, v: Laufparameter

g.) Schreiben Sie den Pseudocode für isLocalMax()

(a) (b)

bool

IsLocalMax$(E_{\mathrm{mag}}, u, v, s_\theta, \mathrm{t_{lo}})$

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

$m_C \leftarrow E_{\mathrm{mag}}(u, v)$

if $m_C < \mathrm{t_{lo}}$ **then** — wenn kleiner als Schwellwert

  return false

else

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

  return $(m_L \leq m_C) \wedge (m_C \geq m_R)$.

-

h.) Warum dreht man um pi/8? getOrientationSector()

- Damir wir sehr einfach die Richtung des Gradienten bestimmen können statt Winkel zu bestimmen
- Wir rotieren die Gradienten, was bedeutet, dass wir sin() und cos() benutzen müssen. Das klingt zunächst schlecht, aber das tun wir für einen Winkel pi/8 und nicht für einen beliebigen Winkel. Deshalb muss es nur einmal ausgerechnet werden und ist effizient