# Übung 4.1

## Lernziele

* Kanten-Detektion mit 1. Ordnung
* Kanten-Detektions-Filter selbst erstellen
* Kanten-Detektion bei Bildern mit Rauschen
* Gradienten-Richtungen berechnen

## Schritte

* 1. Bild: Achteck
* Kanten-Detektions-Masken erstellen
* Korrelieren
* Kanten anzeigen
* Gradient Magnitude
* Binärisieren
* Kanten sind nicht ein Pixel dick 🡪 Non-maximum suppression benötigt
* Gradienten-Orientierung mit np.arctan2(y,x):
* Ein Bild, das Diagramm, Kreis, Design, Darstellung enthält.

  Automatisch generierte Beschreibung
* 2. Bild: Schallplatte
* Glätten

## Aufgaben

* Lesen Sie das Bild «Zebrastreifen.jpg» ein und detektieren Sie die Kanten der Zebrastreifen mittels Gradient Magnitude. Glätten Sie das Bild vorher, um Rauschen vom Asphalt zu entfernen. Welcher Filter funktioniert besser? Mittelwert- oder Median-Filter?
* Anstatt 2 generische Filtermasken (x- und y-Richtung) zu verwenden, kann man die Kanten der Zebrastreifen gezielt mit einer Filtermaske finden, die diagonale Kanten detektiert. Entwerfen Sie Ihre eigenen Filtermasken der Grösse 3x3 und 5x5, um mit einer Kreuz-Korrelations-Operation die Kanten zu detektieren.
  + Die Idee hinter den Filter-Masken basierend auf der 1. Ableitung ist der Gradient (Pfeil) im Bild:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Dieser Gradient (Pfeil) muss bei der Filter-Maske in die gleiche Richtung zeigen:

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

* Bei dem vorliegenden Bild zeigt der Pfeil diagonal nach unten:

Entsprechend muss die Filtermaske so entworfen werden.

* Vergleich der Filter mit Threshold für Binärisierung
* Mit Mittelwert-Filter (11x11 Filtermaske):
* Ein Bild, das Screenshot, Reihe, Rechteck, Astronomie enthält.

  Automatisch generierte Beschreibung
* Mit Median-Filter (11x11 Filtermaske):
* Ein Bild, das Screenshot, Rechteck, Reihe, Schwarz enthält.

  Automatisch generierte Beschreibung

# Übung 4.2

## Lernziele

* Laplace-Filter
* Laplacian of Gaussian (LoG) Filter
* Bilder skalieren (OpenCV)
* Bilder glätten vor skalieren (Moire Effekt)
* Bild-Pyramide

## Schritte

* «Achteck.png» einlesen
* Laplace-Maske definieren
* Kanten detektieren
* Zero-Crossing zeigen
* Laplace-Filter: Zero-Crossing = Kante bzw. Übergang
  + Aber: Ein Wert zeigt die Kontur *um das Objekt* an, das andere zeigt die *äussersten Pixel des Objekts*!
  + Kanten bisschen verschwommen
* Laplace Filter an «Saturn\_1.jpg» anwenden
* LoG Filter anwenden
* Bilder skalieren
* Filter anwenden 🡪 Kanten besser bei kleinerer Auflösung
* Moire-Effekt demonstrieren

## Aufgaben

* Die Funktion cv2.resize() nimmt als Parameter keine Skalierungs-Faktoren, sondern Ziel-Grössen (Anzahl Spalten, Anzahl Zeilen). Programmieren Sie eine Funktion, die ein Bild anhand eines Skalierungs-Faktors gleichmässig skaliert. Beachten Sie, dass cv2.resize() **nur Integer** als Ziel-Grössen annimmt.
* Laden Sie das Bild «Saturn\_1.jpg» ein und detektieren Sie die Kanten bei verschiedenen Skalierungsfaktoren und mit verschiedenen LoG-Filtergrössen. Bei welchem Faktor werden die Kanten am besten detektiert? Zur Visualisierung der Unterschiede können Sie die Histogramme der Kanten-Bilder plotten.

## DEPRECATED\_Bonus

* Mit Hilfe von Bild-Pyramiden lassen sich Bilder in unterschiedlichen Auflösungen mit dem gleichen Filter untersuchen. Programmieren Sie eine Funktion, die den besten Skalierungs-Faktor ermittelt, um Kanten mit LoG oder Laplace-Filtern zu detektieren.
* Strukturierung:
  + For-Schleife (verschiedene Skalierungs-Faktoren durch-iterieren):
    - Bild skalieren
    - Kanten detektieren
    - Standardabweichung ermitteln (vorher Kanten-Bilder normalisieren / auf Bereich [0;1] skalieren!)
    - Skalierungs-Faktor in einer Liste speichern
    - Standardabweichung in einer Liste speichern
  + Nach for-Schleife den Skalierungsfaktor ausgeben, bei dem die maximale Standardabweichung im Kanten-Bild ermittelt wurde
* Neben dem Kontrast könnten Sie als Beurteilungs-Kriterium ausserdem die Auflösung oder den Skalierungs-Faktor verwenden. So helfen «starke» Kanten z.B. nicht mehr, wenn die Form der zu detektierenden Objekte nicht mehr zu erkennen ist. Das Beurteilungs-Kriterium kann folgendermassen berechnet werden:
  + «Kontrast-Gewinn» berechnen: Standardabweichung des Kantenbildes der Original-Auflösung von Standardabweichung der skalierten Kantenbilder abziehen:

mit *i* den Indizes der skalierten Bilder

* + «Kontrast-Score» berechnen: Kontrast-Gewinn mit Skalierungsfaktor multiplizieren.
  + Skalierungsfaktor mit dem höchsten Kontrast-Score auswählen

🡪 Wenn die Skalierung kleiner wird, muss der Kontrast-Gewinn proportional grösser werden, damit eine Skalierung als beste Skalierung ausgewählt wird.