Focus peaking - can edge detection improve the performance of current algorithms?

Thomas Schneider
Matrikel-Nr: 60482
Elektro- und Informationstechnik
Hochschule Karlsruhe

CONTENTS

I. Motivation/Einleitung

II. Mathematische Grundlagen

A. Gauß Filter

Der Gauß Filter ist ein linearer Filter, welcher in der Bildverarbeitung zur Glättung des Bildes und Verminderung von Rauschen, vor allem weißem, verwendet. Feinere Strukturen des Bildes gehen hierbei verloren, wobei gröbere erhalten bleiben. Ein Gaußscher Filterkern der Größe (2k + 1)×(2k + 1) kann mit

$$H_{ij} = \frac{1}{2\pi\sigma^2} * e^{-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}}$$

berechnet werden.
In Python wird der Gauß Kern folgendermaßen realisiert:

```
def gaussian_kernel(size, sigma=1):
  size = int(size) // 2
  x, y = np.mgrid[-size:size + 1,
          -size:size + 1]
  normal = 1 / (2.0 * np.pi * sigma ** 2)
  g = np.exp(-((x ** 2 + y ** 2) /
        (2.0 * sigma ** 2))) * normal
  return g
```

B. Sobel Operator

Der Sobel Operator besteht aus zwei 3×3 Faltungskernen, wobei ein Kern dem jeweils anderen um 90◦ gedreht.

-1	0	+1
-2	0	+2
-1	0	+1

G_x

+1	+2	+1
0	0	0
-1	-2	-1

G_y

Fig. 1. *Links der Faltungskern für die X- und rechts für die Y-Richtung.*

Für jedes Pixel werden die Komponenten der Matrix aufsummiert um den Grauwert zu erhalten.
In Python realisieren wir den Sobel Filter folgendermaßen:

```
def sobel_filter(img):
  Gx = np.array([[-1, 0, 1], [-2, 0, 2],
        [-1, 0, 1]], np.float32)
  Gy = np.array([[1, 2, 1], [0, 0, 0],
        [-1, -2, -1]], np.float32)

  Ix = ndimage.filters.convolve(img, Gx)
  Iy = ndimage.filters.convolve(img, Gy)

  G = np.hypot(Ix, Iy)
  G = G / G.max() * 255
  theta = np.arctan(Iy, Ix)

  return G, theta
```

III. How to measure performance of algorithms

BLA BLA BLA TODO

IV. Canny Edge Detection

Der Canny Algorithmus lässt sich in folgende 5 Schritte unterteilen:

1) Noise reduction
2) Gradient calculation
3) Non-maximum suppression
4) Double treshold
5) Edge Tracking by Hysteresis

A. Noise Reduction

Kantenerkennung ist sehr anfällig für Rauschen, da die meisten und ausschlaggebendsten mathematischen Operationen auf Ableitungen basieren. Deshalb muss eventuell vorhandenes Rauschen im ersten Schritt entfernt werden. Hierfür wir beim Canny Algorithmus das Bild mithilfe eines Gauß Filters geglättet. Mit einem Gaußschen Kernel (hier 5x5) wird der Intensitätswert an der Stelle (i,j) durch das gewichtete Mittel der ihn umgebenden Werte ersetzt. Der resultierende "blurring" Effekt hängt unmittelbar mit der Wahl der Kerngröße zusammen - je größer der Kern, desto besser ist auch der blurring Effekt. Mit steigender Kerngröße steigt jedoch auch die benötigte Rechenzeit, weshalb man hier nur einen 5x5 Kern nimmt, um bei einem ausreichend guten Ergebnis noch performant zu sein.

Hier auf das Logo der Hochschule Karlsruhe angewendet erkennt man im rechten Bild eine Unschärfe gegenüber dem linken Bild.

Fig. 2. *Links das original und rechts unter Anwendung des Gauß Filters.*

B. Gradient Calculation

In diesem Schritt wird sowohl die Intensität als auch die Richtung der Kanten durch die Berechnung des Gradienten ermittelt. Eine Kante wird durch eine merkliche Änderung der Intensität benachbarter Pixel deutlich. Um eine Kante zu erkennen ist es also am einfachsten, einen Filter anzuwenden, welcher die Änderung der Intensität in horizontaler wie vertikaler Richtung markiert.

Nach Glättung des Bildes werden nun also die Ableitungen in x (horizontaler) und y (vertikaler) Richtung berechnet. Am effizientesten kann man dies durch eine Faltung des Bildes mit einem Sobel Kern berechnen.

Die Intensität und Richtung berechnen sich also zu

$$|G| = \sqrt{I_x^2 + I_y^2}$$

$$\boxed{\Theta(x,y) = \arctan(\frac{I_y}{I_x})}$$

Bereits nach diesem Schritt hat man schon ein ziemlich gutes Ergebnis in welchem das Ursprungsbild durch Kanten hinreichend dargestellt ist.

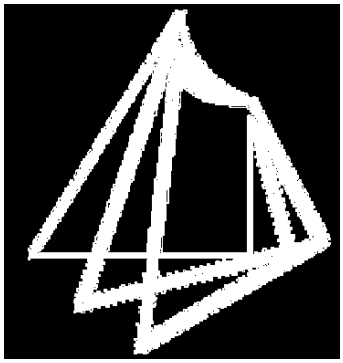

Fig. 3. *Bild nach Anwendung des Sobel Filters*

Man erkennt allerdings, dass die Kanten noch sehr unfein und grob sind und ausgedünnt werden müssen. Außerdem liegen die Intensitätswerte zwischen 0 und 255 - im Idealfall möchten wir entweder 0 oder 255 als Intensitätswert haben um Kanten deutlich hervorzuheben. Hier kommt der dritte Schritt ins Spiel, die Non-Maximum Suppression.

C. Non Maximum Suppression

Die momentan noch mehr als 1 Pixel breiten Kanten werden nun mit der sogenannten Non-Maximum Suppression ausgedünnt. Der Algorithmus durchläuft jeden Punkt der Intensitätsmatrix und findet alle Pixel mit dem maximalen Intensitäts- und Richtungswert.

BILD EINFÜGEN

Der Algorithmus prüft die die jeweils gegenüberliegenden Pixel und vergleicht deren Intensitätswerte. Sollte es keine Nachbarpixel mit höheren Intensitätswerten als das aktuelle Pixel geben, so wird das aktuelle Pixel behalten.