Ludwig-Maximilians-Universität München
LFE Medieninformatik, Mobile Interaction Group
Prof. Dr. Michael Rohs
Dipl.-Inform. Sven Kratz

Sommersemester 2012
Übungsblatt 8
27.06.2012

Übung Computergrafik 2

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum **04.07.2012** abzugeben.

Form der Abgabe:

- Die Übungen können in Gruppen von 2–3 Studierenden bearbeitet werden.
- Die Abgaben bestehen aus den Python-Quelltexten samt aller Bilder, die für die Bearbeitung der Übung verwendet wurden. Idealerweise wird das gesamte PyDev-Projektverzeichnis für die jeweilige Übung über Eclipse als .zip Archiv exportiert. Wichtig: Zur Vermeidung von Namenskonflikten in Eclipse, euren Nachnamen bitte als Präfix vor die Namen der PyDev-Projekte setzen, z.B.: „müller-übung-x“.
- Teilaufgaben, deren Python-Dateien wegen Syntaxfehlern nicht ausführbar sind, werden nicht weiter korrigiert.
- Textaufgaben sollten in Form einer .doc, .odt oder (bevorzugt) als PDF-Datei abgegeben werden, und sollten (falls erforderlich) die benötigten Ausgabebilder der Aufgaben enthalten.
- Alle Übungsabgaben erfolgen über UniWorx[1].

Inhalt: In diesem Übungsblatt behandeln wir das Finden von Eckpunkten mit dem Harris Corner-Detektor sowie Segmentierung mit dem Algorithmus von Otsu.

[1] https://uniworx.ifi.lmu.de

Aufgabe 1 Harris Corner-Detektor (⋆⋆⋆)

Unser erster Schritt bei der automatischen Erkennung von Gebäudefassaden ist die Lokalisierung robuster Merkmale im Bild. Dazu sollen Ecken mit Hilfe des Harris Corner-Detektors gefunden werden. Verwenden Sie die Bilddateien **fassade1.png** und **fassade2.png** aus den Bildmaterialien zu diesem Übungsblatt.

Abbildung 1: Harris-Corners im Bild markiert.

Gehen Sie wie folgt vor:

- Verwenden Sie den Sobel-Operator, um die Ableitungen f_x und f_y in x- bzw. y-Richtung zu bestimmen.
- Berechnen Sie $f_{xx} = f_x^2$, $f_{yy} = f_y^2$ und $f_{xy} = f_x f_y$.
- Berechnen Sie $f_{xxSum} = f_{xx} * w$, wobei "*" der Konvolutionsoperator ist und $w = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.
  Entsprechend für f_{yySum} und f_{xySum}.
- Berechnen Sie für jeden Punkt (x,y) des Bildes die Matrix
  $M(x,y) = \begin{pmatrix} f_{xxSum}(x,y) & f_{xySum}(x,y) \\ f_{xySum}(x,y) & f_{yySum}(x,y) \end{pmatrix}$
  sowie daraus für jedem Bildpunkt das Maß für die "Stärke" der Ecke
  $C(x,y) = \det(M(x,y)) - k \cdot trace(M(x,y))^2$.
  Wählen Sie $k = 0.05$.
- Wenden Sie non-maxima-Unterdrückung in einem Bereich ± 2 in x- und y-Richtung um das jeweils betrachtete Pixel $c \in C$ herum an. Falls eine andere Eckstärke in diesem Bereich größer ist, wird das Pixel im Resultat auf 0 gesetzt, sonst auf c.
- Sortieren Sie das Ergebnis absteigend nach Eckstärke und stellen Sie die 200 stärksten Ecken als Overlay auf dem Eingabebild dar ("plt.scatter"), wie im obigen Beispielbild.

a) Implementieren Sie den Harris Corner-Detektor wie oben beschrieben. Zur Abgabe gehören die beiden Bilder mit den überlagerten, detektierten Ecken sowie der Quellcode.

Hinweis: Die Sortierung lässt sich wie folgt erledigen. Angenommen A ist ein $(n \times 3)$-Array, mit x-Koordinaten in A(:,0), y-Koordinaten in A(:,1) und Eckstärken in A(:,2). Dann lässt sich die aufsteigende Sortierung nach Eckstärke erreichen durch: A = A[A[:,2].argsort(),:].

Hinweis: Sie können für den Sobel-Operator `scipy.Ndimage.sobel` verwenden.

Aufgabe 2 Segmentierung (⋆⋆⋆)

Für eine mobile Applikation, die den Benutzer beim Kochen unterstützen soll, sollen auf dem Tisch liegende Reiskörner gezählt werden. Dabei sollen andere Objekte, wie z.B. ebenfalls auf dem Tisch liegende Kichererbsen nicht mitgezählt werden. Die Objekte sollen anhand der Länge ihrer Hauptachse und Nebenachse unterschieden werden. Die notwendigen Bilddateien **reis{1-2}.png** finden Sie in den Bildmaterialien zu diesem Übungsblatt. Gehen Sie zur Erkennung der Reiskörner wie folgt vor:

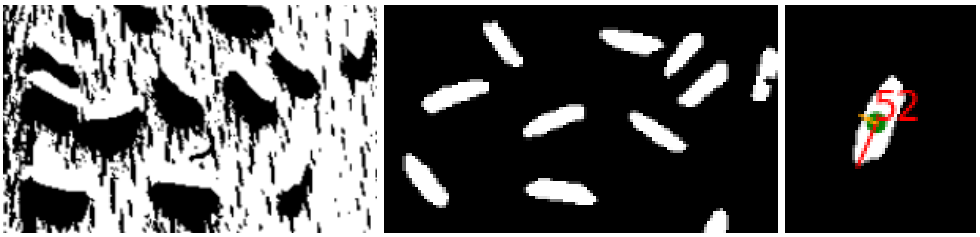

Abbildung 2: *Links:* Reis nach Algorithmus von Otsu. *Mitte:* Median-Filterung vor Algorithmus von Otsu. *Rechts:* Hauptachse (rot) und Nebenachse (orange).

a) Zunächst soll die ungleichmäßige Beleuchtung aus den Bildern entfernt werden. Wenden Sie dazu, gemäß den Folien 9,33-36, ein 11x11-Maximumfilter an. Wenden Sie anschließend ein Gaußfilter mit $\sigma = 35$ an, um das Shading-Bild zu glätten. Dividieren Sie dann das Ursprungsbild durch das Shading-Bild (evtl. $+\varepsilon$, um Division durch 0 zu vermeiden). Verwenden Sie für das Filtern die Funktionen maximum_filter und gaussian_filter aus dem Package scipy.ndimage.

b) Implementieren Sie den Algorithmus von Otsu (Folien 9, 30-32) und wenden Sie ihn auf das von Shading befreite Bild an. Das Ergebnis sollte aussehen, wie in Abbildung 2 links, da Strukturen des Holzes der Tischoberfläche in das Thresholding mit einfließen.

c) Um die Strukturen des Holzes zu entfernen, wenden Sie **vor** dem Algorithmus von Otsu ein möglichst kleines Medianfilter an. Das Ergebnis sollte nun aussehen, wie in Abbildung 2 mitte. Wie groß muss das Filter sein, um die unerwünschten Strukturen zu entfernen? Was passiert, wenn das Medianfilter zu groß gewählt wird?

d) Der nächste Schritt besteht darin, die Segmente zu benennen. Wenden Sie dazu region labeling und flood fill an (Folien 9,38-39). Der Hintergrund soll das Label 0 bekommen. Berechnen Sie für jedes Segment Mittelpunkt und Größe und speichern Sie die Werte in Arrays. Geben Sie das jeweilige Label am Mittelpunkt des Segments (x,y) als Text aus (siehe Abbildung 2 rechts). Benutzen Sie dazu folgende Funktion: plt.text(x, y, str(label), color='red').

e) Um die Orientierung und Länge der Haupt- und Nebenachse der Segmente zu berechnen, wird ähnlich vorgegangen, wie bei der Harris-Matrix (Folien 9, 14-15): Der Eigenvektor der

Kovarianzmatrix M mit dem größten Eigenwert stellt die Hauptachse dar (die Richtung mit der größten Varianz, d.h. der größten Abweichung zum Mittelpunkt). Berechnen Sie zunächst M wie folgt:

$$M(segment) = \begin{pmatrix} f_{xxSum}(segment) & f_{xySum}(segment) \\ f_{xySum}(segment) & f_{yySum}(segment) \end{pmatrix}$$

mit

$$f_{xxSum}(segment) = \frac{1}{segment.size} \sum_{(x,y) \in segment} (x - segment.center.x)^2$$

$$f_{yySum}(segment) = \frac{1}{segment.size} \sum_{(x,y) \in segment} (y - segment.center.y)^2$$

$$f_{xySum}(segment) = \frac{1}{segment.size} \sum_{(x,y) \in segment} (x - segment.center.x)(y - segment.center.y)$$

Die beiden Eigenwerte e und Eigenvektoren v lassen sich nun berechnen mit

$$e, v = np.linalg.eig(M).$$

Die Eigenwerte stellen dabei die Varianz in Richtung der Hauptachse bzw. orthogonal dazu (Nebenachse) dar. Wie groß ist durchschnittlich das Verhältnis der Standardabweichungen (Wurzel der Varianz) von Hauptachse zu Nebenachse bei den Reiskörnern in **reis1.png**? Merken Sie sich diesen Wert für den Aufgabenteil (g).

f) Zeichnen Sie für jedes Segment die Eigenvektoren ausgehend jeweils vom Zentrum eines Segments in das Bild ein (wie in Abbildung 2 rechts). Verwenden Sie dazu die Funktion pp.plot(X, Y, color='red') für die Hauptachse und pp.plot(X, Y, color='orange') für die Nebenachse. Die Länge soll dem doppelten der Standardabweichungen entsprechen.

g) Verwenden Sie nun **reis2.png** als Eingabe. Markieren Sie alle Objekte, bei denen das Verhältnis der Standardabweichungen von Hauptachse zu Nebenachse bei den Reiskörnern dem vorher ermittelten Wert entspricht (bzw. einer von Ihnen zu findenden prozentualen Toleranz um diesen Wert). Werden alle Reiskörner und nur die Reiskörner korrekt markiert? Was passiert, wenn die Toleranz zu klein bzw. zu groß gewählt wird? Welche weiteren Parameter könnten zum Finden von Reiskörnern verwendet werden?

Hinweis: Sie können auch hier, z.B. für das Maximumfilter, Funktionen aus `scipy.Ndimage` verwenden.

Hinweis: Aufgabenteile e,f,g sind optional.