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

Sommersemester 2012
Übungsblatt 1
25.04.2012

Übung Computergrafik 2

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum **16.05.2012** abzugeben.

Form der Abgabe

- Die Übungen können in Gruppen von 2–3 Studenten 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 die 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:
Dieses Übungsblatt dient zur Einarbeitung in die Programmierumgebung für unsere Vorlesung, *Eclipse* mit *PyDev*, ein Plugin für die Entwicklung in der Programmiersprache Python[2], sowie der Numerik-Library *NumPy* [3] und der Visualisierungslibray *Matplotlib* [4]. Desweiteren werden Sie lernen, Farbbilder in Graustufenbilder umzuwandeln, und wie man den Kontrast von kontrastarmen Graustufenbildern durch Grauwertspreizung erhöhen kann.

Aufgabe 1 Entwicklungsumgebung installieren (⋆)

In dieser Aufgabe installieren Sie, wie in den Folien zur Übung 1 beschrieben Python, Eclipse, PyDev, NumPy und Matplotlib:

a) Installieren Sie Python v. 2.7.x und machen Sie sich mit den Grundlagen von Python vertraut. Dies geht am schnellsten durch Lesen eines Tutorials, z.B. http://docs.python.org/tutorial/.

b) Installieren Sie Eclipse und das PyDev-Plugin

c) Installieren Sie Numpy und Matplotlib und lernen Sie die Grundkonzepte von Numpy (http://www.scipy.org/Tentative_NumPy_Tutorial) und Matplotlib (http://matplotlib.sourceforge.net/users/pyplot_tutorial.html) kennen.

[1] https://uniworx.ifi.lmu.de/
[2] http://www.python.org
[3] http://numpy.scipy.org/
[4] http://matplotlib.sourceforge.net/

Aufgabe 2 Konvertierung von Farbbildern zu Grauwerten ($\star$)

Digitale Farbbilder zur Darstellung auf dem PC werden meistens im RGB-Format gespeichert. Jedes Pixel eines Bildes besitzt also drei Intensitätswerte für jeweils rot, grün und blau. Möchte man aus das Farbbixel ein Graustufenpixel umwandeln, so wäre ein naiver Ansatz, einfach den Mittelwert der drei Werte zu nehmen. Dies ist jedoch in Anbetracht der Farbwahrnehmung des menschlichen Auges suboptimal, denn die einzelnen Farbkanäle werden unterschiedlich stark vom Auge wahrgenommen. Eine standardisierte Annäherung an die korrekte Graustufenintensität Y, basierend auf den Primärfarben des Fernsehstandards NTSC, ergibt die folgende Formel:

$$Y = 0.2126R + 0.7152G + 0.0722B$$

a) Implementieren Sie eine Python-Methode, die ein Farbbild einliest und über den Mittelwert ein korrespondierendes Graustufenbild berechnet.

b) Implementieren Sie nun eine Methode, die die oben genannte Formel zur Berechnung der korrekten Intensität verwendet.

c) Vergleichen Sie beide erzeugten Graustufenbilder und beschreiben Sie kurz die Unterschiede im optischen Eindruck (falls vorhanden).

Aufgabe 3 Thresholding und Grauwertspreizung ($\star\star$)

a) **Thresholding** Es kommt während der Bildverarbeitung vor, dass man gewisse Pixel, deren Intensität über oder unter einem gewissen Grenzwert T liegt löschen oder auf einen fixen Wert M (meistens 255 oder 0) setzen möchte. Das Ausschließen oder Ändern bestimmter Pixel mittels eines Grenzwertes heißt auf Englisch *Thresholding*. Es gibt eine Reihe verschiedener Thresholding Funktionen, die hier Beispielhaft als Pseudocode aufgeschrieben sind:

- THR_BINARY: `IF src[i] > T THEN dst[i] = M ELSE dst[i] = 0`
- THR_BINARY_INV: `IF src[i] > T THEN dst[i] = 0 ELSE dst[i] = M`
- THR_TRUNC: `IF src[i] > T THEN dst[i] = M ELSE dst[i] = src[i]`
- THR_TOZERO_INV: `IF src[i] > T THEN dst[i] = 0 ELSE dst[i] = src[i]`
- THR_TOZERO: `IF src[i] > T THEN dst[i] = src[i] ELSE dst[i] = 0`

`src[i]` sind jeweils die Pixel des Eingabebildes, `dst[i]` sind die Pixel des Ausgeabebildes. `i` ist der Index über die Pixel im Ein-/Ausgabebild.

Implementieren Sie für alle Threshold-Funktionen jeweils eine Python-Methode, die ein Eingabebild in Graustufen, sowie die Parameter T und M einliest, das jeweilige Thresholding anwendet, und ein Ausgabebild zurückgibt. Beschreiben Sie informell die Auswirkungen und Charakteristika der verschiedenen Threshold Funktionen anhand von Beispielausgaben, die Sie durch ihre Python-Methoden erhalten haben.

b) **Grauwertspreizung** Betrachtet man das Histogramm kontrastarmer Bilder, so fällt auf, dass die maxima und minima Grauwerte sehr dicht aneinander liegen. Es wird also nicht der komplett verfügbare Kontrastumfang des Bildes genutzt. Diesem Problem kann man mit der *Grauwertspreizung* begegnen. Die Grauwertspreizung führt dazu dass die Minima und Maxima der Grauwerte maximal auseinander leigen, und somit die Verteilung der Grauwerte „gespreizt“ wird. Dies ist in Abbildung 1 beispielhaft dargestellt:

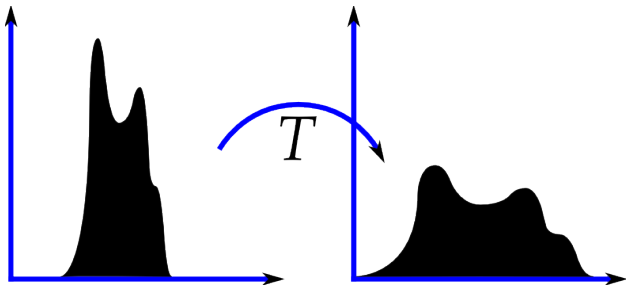

Abbildung 1: Histogrammspreizung zur Kontrastvergrößerung des Kontrastarmen Bildes.

Die berechnung der Grauwertspreizung erfolg durch lineares Transformieren des genutzten Grauwertebereichs $R_{src} = \{g_{min}, \ldots, g_{max}\}$ auf den gesamt verfügbaren Bereich $R_{max} = \{0, \ldots, G\}$ (in unserem Falle $G = 255$).

Finden Sie eine geeignete Funktion $T_{stretch}(g)$, die für ein gegebenes Graustufenbild die Grauwerte von R_{src} streckt und nach R_{max} abbildet. *Hinweis:* Bestimmen Sie dazu das Intervall $g_{max} - g_{min}$. des jeweiligen Eingabebildes.

Implementieren Sie eine Python-Methode, die ein Graustufenbild als Eingabe hat und auf die Eingabe die Grauwertspreizung anwendet. Diskutieren Sie die Auswirkungen der Grauwertspreizung anhand von Ein- und Ausgabebildern sowie deren Histogramme. Verwenden Sie dazu Bilder mit unterschiedlichen hell-dunkel-Verteilungen, z.B. ein Bild mit großen hellen und wenigen dunklen Bereichen und umgekehrt.

Sehr Wichtig: Da Matplotlib schon automatisch zur besseren Kontrastdarstellung die Grauwertspreizung automatisch vornimmt, muss diese bei der Bilddarstellung in dieser Aufgabe wie folgt deaktiviert werden, sonst sehen Sie die Auswirkungen der Grauwertspreizung nicht:

```
matplotlib.pyplot.imshow(<image>, norm=matplotlib.colors.NoNorm())
```