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

Sommersemester 2012
Übungsblatt 2
16.05.2012

Übung Computergrafik 2

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum **24.05.2012** 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:

In dieser Übung werden Moiré-Effekte und Subsampling, Kontrastverstärkung von Grauwertbildern mit Color Maps und die globale und lokale Histogrammlinearisierung behandelt.

Sämtliche benötigte Bilddateien können auf der Vorlesungsseite unter

http://www.medien.ifi.lmu.de/lehre/ss12/cg2/uebungen/ue2Images.zip

heruntergeladen werden.

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

Aufgabe 1 Subsampling und Moiré-Effekte (⋆⋆)

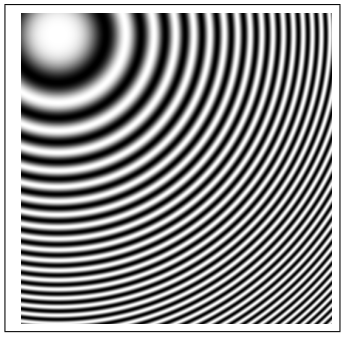

Abbildung 1: Zone Plate

Beim Subsampling (Abtastung mit geringerer Auflösung) von Bilddaten können Moiré-Effekte auftreten. Am Beispiel eines *Zone Plates* (Abbildung 1) lassen sich Moiré-Effekte sehr deutlich untersuchen. Folgender Python-Code erzeugt ein Zone Plate:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

def zone_f(coords, center, k=0.001):
    deltaX = center[0] - coords[0]
    deltaY = center[1]- coords[1]
    return  np.cos(k * (deltaX*deltaX + deltaY*deltaY))

N=512
center = np.array([35,60])
plate_img = np.zeros((N,N))

for i in xrange(img.shape[0]):
    for j in xrange(img.shape[1]):
        plate_img[i][j] = zone_f(np.array([i,j]), center)
```

Implementieren Sie nun eine Python-Funktion, die eine 512×512 Zone Plate mit immer geringerer Auflösung abtastet. D.h. jeweils nur jeden 2.,4.,8.,16. Pixel aus der ursprünglichen Datei übernehmen und daraus ein neues resultierendes Bild erstellen. Zeigen Sie die vier Ergebnisse des Subsamplings in ihrer Abgabe.

Aufgabe 2 Kontrastverstärkung von Grauwertbildern mit Color-Maps (⋆⋆)

Hinweis: Für das Öffnen der Höhenstufendateien in Teilen (b) und (c) nutzen sie bitte folgende Funktion, die sicherstellt dass die Graustufenwerte als Integer im Bereich 0-255 vorliegen:

```
def imread_int(filename):
    img = plt.imread(filename)
    return np.array(img*255, dtype='int')
```

a) Die Bilddatei **colormap.png** stellt eine Color-Map dar. Öffnen Sie diese Datei als Numpy-Array und bestimmen sie die nichtlinearen monotonen Abbildungungsfunktionen $redi(g), greeni(g), bluei(g$ der Grauwerte für die Kanäle Rot-Grün-Blau. Genererieren Sie für jede dieser Funktionen eine Plot-Darstellung mit Matplotlib.

b) Benutzen sie die in (a) gewonnenen Funktionen um folgendes Höhenstufenbild (Height Map), **usa_small.png**, (Abbildung 3) kontrastreich einzufärben.

Abbildung 2: colormap.png

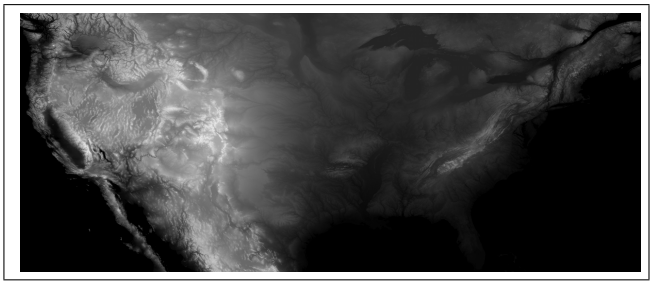

Abbildung 3: usa_small.png

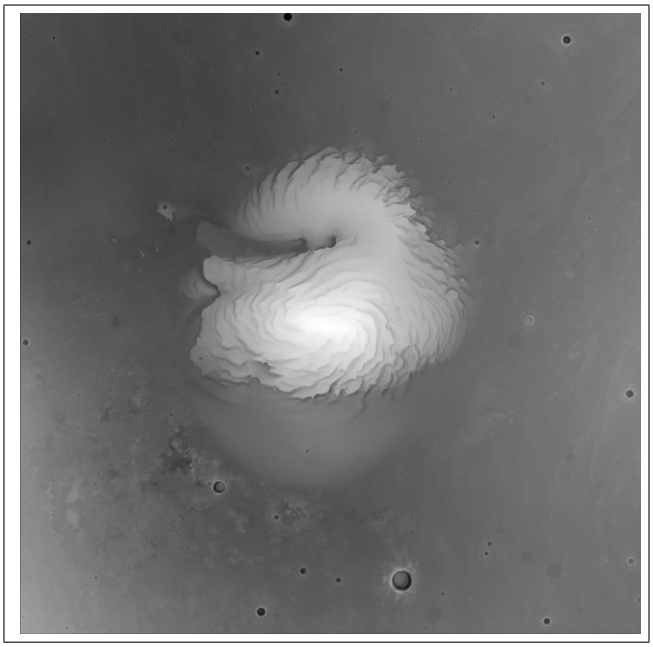

Abbildung 4: mars-nordpol.png

c) Die NASA hat mit einer neuen Sonde eine Radar-Geländevermessung des Nordpols des Mars durchgeführt und die Height Map in Abbildung 4 (**mars-nordpol.png**) erhalten.

   Entwicklen Sie eine geeignete Color-Map zur Einfärbung dieses Bildes, und Färben Sie den Norpol des Mars ansprechend ein.

Aufgabe 3 Histogrammlinearisierung (⋆⋆⋆)

Wenden Sie alle Histogrammlinearisierungsmethoden in dieser Aufgabe auf die Bilddateien **linearize{1-3}.png** sowie auf ein Bild ihrer Wahl an. Verwenden Sie auch in dieser Aufgabe zum Laden der Bilddateien die Funktion `imread_int()` aus Aufgabe 2.

a) Implementieren Sie anhand der Beschreibung in den Vorlesungsfolien eine Python-Methode zur *globalen* Histogrammlinearisierung, und wenden Sie diese auf die o.g. Bilder an. Beschreiben Sie, falls vorhanden, auftretende Probleme in den Ausgabebildern.

b) Implementieren Sie eine Python-Methode, die eine *adaptive* Histgrammlinearisierung (AHE) durchführt. Überlegen Sie sich wie der Code ggf. optimiert werden könnte. Wenden Sie AHE

auf die o.g. Bilder an. Diskutieren Sie, ob die Bildqualität sich im Vergleich zum vorherigen Verfahren verbessert hat.

Skalieren Sie für diese und die nächste Aufgabe aus Performanzgründen linearize{1-3}.png[2] auf eine Breite von 256 Pixeln herunter. Verwenden Sie einen lokalen Histogrammradius von 8 Pixeln. Pixel(-adressen), deren Koordinaten außerhalb der Bildränder liegen, werden beim Berechnen der Wahrscheinlichkeitsfunktion der Bildintensitäten im Histogramm *nicht* mitgezählt.

c) Um die Bildeigenschaften bei manchen Spezialanwendungen (z.B. in der Medizinischen Bildverarbeitnug) noch weiter zu verbessern, gibt es die *kontrastlimitierende adaptive* Histogrammlinearisierung (CLAHE). In dieser Variante wird die Steigung der kumulativen Wahrscheinlichkeitsfunktion durch einen Parameter g_{max} begrenzt.

   Implementieren Sie CLAHE und wenden Sie es auf die o.g. Bilder an. Beschreiben Sie die Veränderungen in der Darstellungsqualität (falls vorhanden) für g_{max} Werte von 5/255,10/255 und 20/255.

d) Erklären Sie, warum die diskrete Histogramm-Linearisierung i.a. kein flaches Histogramm ergibt.

[2] Fotos z.T. von den Flickr Usern ross.gady und rumjaku (CC-BY-SA-NC)