BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Prof. Dr.-Ing. Kristian Hildebrand

Visual and Scientific Computing
Bildverarbeitung und Seam Carving

WiSe 2019/20

Das Lernziel dieser Übungsaufgabe ist, sich mit Basisalgorithmen der Bildverarbeitung vertraut zu machen und am Beispiel des *Seam Carving* das Optimierungsverfahren und die Programmiermethode *Dynamische Programmierung* selbst zu implementieren.

Aufgabe 1: Bildverarbeitung - Gradientenlängen berechnen (4 Punkte)

Ziel dieser Aufgabe ist es die Gradienten eines Bildes auf Basis des Sobel-Filters selbst zu berechnen. Dieser ist einer der grundlegendsten Bildverarbeitungsfilter und wird in der Vielzahl von Anwendungen als Grundbaustein verwendet.

a) Implementieren Sie zunächst, wie auch schon in der Vorlesung besprochen und gezeigt, eine Funktion (*convolution2D*), die ein Schwarz/Weiß-Bild mit einem sogenannten Kernel (Faltungsmatrix) mit beliebiger Größe faltet. Dabei sollte das Randproblem gelöst werden, indem das Bild am Rand erweitert bzw. vergrößert wird (sogenanntes Padding). Wie in der Vorlesung besprochen, soll das Padding mit der Pixelfarbe des Randes gefüllt werden. **Hinweis:** Schauen Sie sich dazu die Funktion *numpy.pad* an.

b) Probieren Sie einige Kernel aus, die unterschiedliche Effekte erzielen (mindestens 5) und speichern Sie die Ergebnisbilder ab. Zwei dieser Kernel sollten die besprochenen Sobel-Filter in x- und y-Richtungen sein. Wählen Sie weitere sinnvolle Kernel aus, dafür sollten Sie etwas recherchieren.

c) Implementieren Sie die Funktion *magnitude_of_gradients*. Die Funktion bekommt ein RGB-Bild übergeben und soll die Gradientenlängen zurückgeben. Nutzen Sie dafür erneut den Sobel-Filter (in x- und y-Richtung), um die Gradienten eines Bildes zu errechnen. Rufen Sie die Funktion mit einem Beispiel-Bild auf und speichern Sie die Gradientenlängen als Ergebnisbild ab. (siehe Abbildung 1)

Abbildung 1: Links: Ausgangsbild. Rechts: Bild mit den enthaltenen Gradientenlängen des Bildes.

Aufgabe 2: Inhaltsabhängige Bildverzerrung - Seam Carving (6 Punkte)

Seam Carving wurde Ihnen in der Vorlesung vorgestellt. Es ist ein Verfahren, das es ermöglicht, die Größe eines Bildes nicht durch Skalieren oder Beschneiden, sondern durch intelligentes Entfernen von Pixeln mit geringer Wichtigkeit (Energie), zu ändern. Dabei werden die Pixel ihrer Wichtigkeit nach bewertet und beim Verkleinern werden zuerst die unwichtigen Pixel entfernt. Die Wichtigkeit eines Pixels wird dabei aufgrund der Helligkeitsänderungen mit seinen benachbarten Pixeln gemessen.
Das Paper zu diesem Verfahren und ein Video finden Sie unter http://www.faculty.idc.ac.il/ARIK/site/seam-carve.asp. Die folgenden Aufgaben/Funktionalitäten müssen pro Pixelpfad/Seams der entfernt werden soll aufgerufen werden:

a) **Energie berechnen:** Berechnen Sie die Gradientenlängen des aktuellen Bildes. Diese dienen uns als Energie in einem Pixel. Sie können sinnvollerweise die aus Aufgabe 1.c) geschriebene Funktion benutzen/importieren.

b) **Akkumulierte Energie eines Pixels p_i berechnen:**
Implementieren Sie die Funktion *calculate_accum_energy*. Sie bekommt die Energie an jedem Pixel übergeben und soll die akkumulierte Energie zurückgeben. Sei $E(p_i)$ die Energie und $E_a(p_i)$ die akkumulierte Energie von p_i, dann soll gelten:

$$E_a(p_i) = E(p_i) + \min_{p_j \in V}(E_a(p_j))$$

Dabei ist V die Menge aller Vorgänger. Ein Pixel p hat als Vorgänger die angrenzenden Pixel links oberhalb, oberhalb und rechts oberhalb seiner Position in der Matrix, falls sie existieren. D.h. bei Pixeln der obersten Reihe ist $E_a(p_i) = E(p_i)$, da die Menge V leer ist.

E: 40 aE: 40	E: 60 aE: 60	E: 40 aE: 40	E: 10 aE: 10	E: 20 aE: 20
E: 53.3 aE: 93.3	E: 50 aE: 90	E: 25 aE: 35	E: 47,5 aE: 57,5	E: 40 aE: 50
E: 50 aE: 140	E: 40 aE: 75	E: 40 aE: 75	E: 60 aE: 95	E: 90 aE: 140
E: 30 aE: 105	E: 70 aE: 145	E: 75 aE: 150	E: 25 aE: 100	E: 50 aE: 145
E: 65 aE: 170	E: 70 aE: 175	E: 30 aE: 130	E: 30 aE: 130	E: 10 aE: 110

Abbildung 2: Beispiel für Bildenergien und die akkumulierten Energien.

c) **Berechnen der Seam-Mask:** Implementieren Sie die Funktion *create_seam_mask*. Die Funktion bekommt die akkumulierten Energien und gibt einen Pixelpfad durch das Bild in Form einer Bitmaske zurück. Der Pixelpfad soll der zusammenhängende Pfad mit der geringsten Gesamtenergie (summierten $E(p_i)$ Werte) sein. Dieser soll in einer Bitmaske kodiert werden, der Pfad ist durch False Werte makiert, der Rest ist True. Gegeben Die Energiewerte aus Abbildung 2 würde also Folgendes Array zurückgegeben werden:

$$M = \begin{pmatrix} \text{True} & \text{True} & \text{True} & \text{False} & \text{True} \\ \text{True} & \text{True} & \text{False} & \text{True} & \text{True} \\ \text{True} & \text{True} & \text{False} & \text{True} & \text{True} \\ \text{True} & \text{True} & \text{True} & \text{False} & \text{True} \\ \text{True} & \text{True} & \text{True} & \text{True} & \text{False} \end{pmatrix}$$

d) **Entfernen des Seams:** Verkleinern Sie das Bild, indem Sie die fertige Funktion *seam_carve* mit dem ursprünglichen Bild und der Maske aufrufen.

e) **Speichern des Seams:** Zur späteren Visualisierung/Debuggen soll der benutzte Seam in einer globalen Maske gespeichert werden. Nutzen Sie die Funktion *update_global_mask*. Die Maske soll genutzt werden, um alle bisher benutzten Seams im Originalbild rot zu färben. **Hinweis:** Im Vergleich zu der Bitmaske in der Funktion *create_seam_mask* ist die globale Maske invertiert (False → True und True → False).

f) **Speichern der Ergebnisse:** In jedem Zwischenschritt soll das verkleinerte Bild sowie das Orginalbild mit allen bisherigen Seams in rot abgespeichert werden. Am Ende soll das resultierende Bild nochmal extra abgespeichert werden. Die Ergebnisse nach der 20 Iteration sollten so aussehen wie in Abbildung 3 zu sehen.

Abbildung 3: Zwei Ergebnisse des Seam Carving. Links: Ausgabsbild. Mitte: Bild mit den ersten 20 zu entfernenden Pixelpfaden. Rechts: Verkleinertes Bild.

Abgabe Die Bearbeitungszeit der Teilaufgabe ist für ca. zwei Woche ausgelegt. Die Abgabe soll via Moodle bis zu dem dort angegebenen Termin erfolgen. Verspätete Abgaben werden mit einem Abschlag von 3 Punkten je angefangener Woche Verspätung belegt. Geben Sie bitte jeweils nur eine einzige .zip-Datei mit den Quellen Ihrer Lösung ab.

Ich behalte mir vor stichprobenartig Ihre Lösungen in der Übung demonstrieren und erläutern zu lassen. Die Qualität Ihrer Demonstration ist, neben dem abgegebenen Code, ausschlaggebend für die Bewertung!