(a) Originalbild.

(b) Verkleinerung.

Abbildung 1: Vergleich Originalbild mit herkömmlicher Verkleinerung. Es treten Artefakte auf, welche das Objekt im Bild (die Blase) verzerren.

(a) Vertikale Verkleinerung.

(b) Horizontale Verkleinerung.

(c) Vertikale Vergrößerung.

Abbildung 2: Content-Aware Image Resizing: Das Bild wird beim Vergrößern oder Verkleinern so bearbeitet, sodass das Objekt im Bild (die Blase) nicht verändert wird und sich die Veränderungen auf den Hintergrund beschränken. Dies führt für den Betrachter zu optisch besseren Ergebnissen.

CV/task1b — Content-Aware Image Resizing

1 Überblick

Ziel dieser Aufgabe ist es, einen tieferen Einblick in die OpenCV-Library zu bekommen und mit Bild- und Matrizenoperationen vertraut zu werden. Diese Aufgabe besteht darin, einen Image Resizing Algorithmus zu implementieren. Dieser soll mittels Content-Aware Resizing (Seam Carving) [1] Bilder vergrößern bzw. verkleinern. Seam Carving ist ein Prozess, der Pixelpfade in einem Bild berechnet, und iterativ Pfade aus dem Bild löscht oder eingefügt, welche die minimalen Bildinformationen enthalten. Die Bildinformation wird mithilfe einer Energiefunktion bestimmt und im Zuge dieser Aufgabe werden Sie zwei mögliche Methoden zur Energieberechnung kennenlernen – einerseits Berechnung mittels Gradienten und andererseits mittels Entropie in der Bonus Aufgabe.

Die Eingabedaten werden aus JSON-Dateien bereits automatisch eingelesen und können sofort zur Berechnung herangezogen werden. Der Inhalt dieser Aufgabe beschränkt sich einzig und allein auf den Image Resizing Algorithmus, nicht auf etwaigen Aufwand der sich durch die Programmiersprache ergibt. Das Framework ist so aufgebaut, dass Sie die relevanten Funktionen implementieren müssen um die gewünschte Ausgabe zu erreichen.

1.1 Seam Carving mittels Gradientenberechnung

Der Seam Carving Algorithmus basiert auf der Idee, dass man Bilder skaliert, indem man jene Pixel entfernt, welche die wenigsten Bildinformationen enthalten. Dadurch bleiben die wichtigsten Elemente im Bild (jene mit der meisten Information) erhalten. Eine mögliche Methode um Bildinformation zu quantifizieren ist mittels Gradienten. Bildbereiche mit niedrigem Gradienten gehören eher zum Hintergrund und können entfernt werden. In Bildbereichen mit hohen Gradienten sind meistens Objekte, wie z.B. Personen oder Häuser, die nicht entfernt werden sollten.
Beim Seam Carving wird das Bild iterativ um einen Pixel in entweder Breite oder Höhe verkleinert bzw. vergrößert. Dabei wird schrittweise jener Pfad („Seam“) mit geringster Informationsdichte von oberer zur unterer Bildkante (für vertikales Verkleinern) bzw. linker zur rechter Bildkante (horizontales Verkleinern) gesucht und entfernt. Dieser Vorgang wird so lange durchgeführt, bis das Bild die gewünschte Größe erreicht hat. Die entfernten Pixel sind mittels 8-er Nachbarschaft miteinander verbunden und bilden zusammen einen sogenannten Seam. Um den jeweiligen Seam mit der niedrigsten Informationsdichte zu finden, wird eine Gradienten-Maske berechnet. Um die Gradienten zu berechnen wird ein Sobel Filter auf das Bild angewendet. Die Summe der Gradienten-Werte auf einem Seam entsprechen den Kosten des Seam. Nachdem alle Kosten aller Seams berechnet wurden, kann zum Beispiel zum Verkleinern des Bildes der Seam mit den niedrigsten Gesamtkosten aus dem Bild entfernt werden. Da sich das Bild beim Löschen ändert, und die Gradienten (und daher auch die Kosten der Seams) sich ändern können, muss die Maske in jeder Iteration neu berechnet werden um sicher zu gehen, dass immer der aktuelle minimale Seam gelöscht wird.

1.2 [Bonus] Seam Carving mittels lokaler Bildentropie

Die Bonusaufgabe funktioniert größtenteils gleich wie das normale Seam Carving, allerdings wird anstatt einer Gradienten-Maske eine Entropie-Maske zur Energieberechnung verwendet. Hierbei wird für jedes Pixel ein lokales Fenster betrachtet. Für dieses Fenster berechnet man die Entropie und setzt diesen Wert als Informationsenergie für das Pixel im Mittelpunkt des Fensters in der Maske ein. Nachdem alle Entropiefenster berechnet sind, kann der Algorithmus wie oben beschrieben weitergeführt werden.

2 Aufgaben

Das Beispiel ist in mehrere Unteraufgaben gegliedert. Der prinzipielle Ablauf besteht aus den folgenden Schritten:

- Gradienten Berechnung
- Seam Berechnung mittels Gradienten
- Hervorheben vertikaler Seams
- Bild Verkleinerung: Entfernen von Seams mit geringster Gesamtenergie
  - Entfernen vertikaler Seams
  - Entfernen horizontaler Seams
- Vertikale Bild Vergrößerung

Bonustask:

- Bild Verkleinerung: Entfernen vertikaler Seams mittels Entropie Berechnung
  - Entropie Berechnung
  - Seam Berechnung mittels Entropie
  - Entfernen von Seams mit geringster Gesamtenergie

Diese Aufgabe ist mit Hilfe von OpenCV[1] zu implementieren. Nutzen Sie die Funktionen, die Ihnen OpenCV zur Verfügung stellt und achten Sie auf die unterschiedlichen Parameter und Bildtypen.

[1] http://opencv.org/

2.1 Gradienten Berechnung (1 Punkt)

Dieser Teil der Aufgabenstellung ist in der Funktion `calculateGradientMag(...)` zu implementieren. Hierbei soll ein Gradientenbild mit Hilfe der von OpenCV zur Verfügung gestellten Funktion der Sobel-Filterung erstellt werden. Der Sobel Filter detektiert gerichtete Änderungen im Bild (Gradienten) unter Zuhilfenahme von Ableitungen. Ein großer Gradientenwert eines Bildes deutet auf eine Änderung im Bild hin (z. B. Kanten, Farbübergänge). Für die Ableitung in x-Richtung E_x wird das Bild I mit dem Kern

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{1}$$

gefaltet. Analog wird für die Ableitung in y-Richtung E_y das Bild I mit dem Kern

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{2}$$

gefaltet, sodass sich die Formeln

$$E_x = S_x * I, \quad E_y = S_y * I \tag{3}$$

ergeben. $*$ bezeichnet dabei die Faltungsoperation. Bestimmen Sie zuerst die jeweils *erste* Ableitung in die Richtungen x und y. Wenden Sie dazu den Sobel Filter auf das Grauwertbild an. Wählen Sie für den Parameter der die Bit-Tiefe des Ausgabe-Bildes beschreibt den Typ `CV_32F`. Kombinieren Sie danach die beiden Gradientenbilder gemäß der Formel

$$E = \sqrt{E_x^2 + E_y^2}, \tag{4}$$

wobei E das Gesamtgradientenbild bezeichnet.

Abbildung 3 zeigt das Ergebnis der Gradienten Berechnung.

Hilfreiche OpenCV-Methoden:

- `cv::cvtColor(...)`
- `cv::Sobel(...)`

(a) Originalbild.

(b) Gradienten Matrix E.

Abbildung 3: Eingabe und Ausgabe der Gradienten Berechnung.

2.2 Seam Berechnung mittels Gradienten (3 Punkte)

Dieser Teil der Aufgabenstellung ist in der Funktion `calculateCostImage(...)` zu implementieren. In dieser Funktion sollen alle vertikalen Seams (=kürzeste Pfade) im Bild berechnet werden. Die Anzahl der zu berechnenden Seams wird mit der Parameter `rm_seam_count` definiert. Zur Berechnung der Seams wird eine sogenannte Kosten Matrix M verwendet, wofür die vorher berechnete Gradienten Matrix E benötigt wird. Die kürzesten Pfade werden mittels **Backtracking** aus der Kosten Matrix M berechnet. Die Datenstrukturen für die einzelnen Matrizen sowie zwei Schleifen zur Berechnung der Kosten Matrix M sind im Sourcecode bereits vorgegeben.

Berechnung der Kosten Matrix M und Pfad Matrix P

Beachten Sie, dass im Sourcecode `int-padding` auf die Kosten Matrix M angewendet wurde. Das heißt, dass am linken bzw. am rechten Rand der Kosten Matrix M eine Spalte hinzugefügt wurde, wobei die Einträge dieser Spalte den numerisch höchst möglichen Wert besitzen. Dies wird verwendet, um Spezialfälle bei der Minimum Suche des kürzesten Pfades am linken bzw. rechten Rand des Bildes zu vermeiden.

Um nun die Kosten Matrix zu erstellen, muss jede Zeile (für vertikale Seams) der Kosten Matrix M durchlaufen werden, und der einzelne Wert wird wie folgt berechnet:

- Für die erste Zeile:

$$M(0, j+1) = E(0, j), \tag{5}$$

- Für jede weitere Zeile:

$$M(i, j+1) = E(i, j) + \min\{M(i-1, j+1-1),\ M(i-1, j+1),\ M(i-1, j+1+1)\}, \tag{6}$$

wobei i und j der Zeilen- bzw. Spaltenindex der Gradienten Matrix E ist. Wegen der Verwendung des `int-paddings` muss, wie in der Formel beschrieben, beim Spaltenindex der Kosten Matrix M ein Offset von $+1$ berücksichtigt werden.

Während der Berechnung der Kosten Matrix M soll ebenfalls eine Pfad Matrix P zum Mitspeichern des Index des vorherigen Pixels im Pfad erstellt werden. Diese ist nötig, um am Ende der Berechnung die minimalen Seams zu finden. Die erste Zeile der Pfad Matrix P wird mit -1 initialisiert. Alle weiteren Zeilen speichern die x-Koordinate des Vorgänger-Pixels, welcher in der vorherigen Zeile steht. Die Pfad Matrix P und die Kosten Matrix M sind in für ein kleines Beispiel in Abbildung 5 dargestellt. Formal erfolgt die Berechnung der einzelnen Pixel $p = (x, y)$ der Pfad Matrix P folgendermaßen:

$$\begin{aligned} P(0, x) &= -1 \\ P(y, x) &= \operatorname*{argmin}_{x-1 \le j \le x+1} M(y-1, j+1), \quad 1 \le y < H, \end{aligned}$$

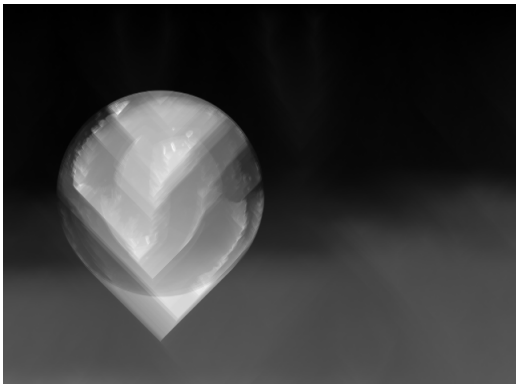
wobei H die Bildhöhe ist. Beachten Sie, dass aufgrund des Paddings hier wiederum ein $+1$ bei der Indizierung von M notwendig ist. Beim Erstellen der Pfad Matrix P muss außerdem die Reihenfolgen der ausgewählten Indizes bei gleichen minimalen Werten in der Kosten Matrix M beachtet werden. Hierbei gilt die Reihenfolge *linkes* Element zuerst, dann *rechtes* Element, sonst *mittleres* Element.

Abbildung 4 zeigt das Ergebnis der Kosten Matrix M. Hierfür wurde die Kosten Matrix M mit dem Befehl `cv::normalize(...)` mit dem Konvertierungscode `CV_8UC1` auf den Wertebereich $[0, 255]$ normalisiert.

Berechung der kürzesten Pfade mittels Pfad Matrix P

Um den kürzesten Pfad aus der Kosten Matrix zu bestimmen wird wie bereits erwähnt sogenanntes Backtracking angewandt. Dabei fängt man von der letzten Zeile in der Kosten Matrix M an und sucht sich jenes Pixel mit den geringsten Kosten. Die Koordinaten (x, y) dieses Pixels beschreiben das letzte Element im Pfad. Der Vorgänger dieses Pixels hat

(a) Originalbild.



(b) Kosten Matrix M (normalisiert).

Abbildung 4: Visualisierung der Kosten Matrix M (normalisiert).

als Koordinaten $(y-1, P(y,x))$. Dies wird so lange wiederholt, bis die erste Zeile erreicht wurde. Abbildung 5 zeigt eine beispielhafte Darstellung von Kosten Matrix M, Gradienten Matrix E und dem eingezeichneten minimalen Seam und die dazugehörige Pfad Matrix P.

2.3 Hervorheben vertikaler Seams (1 Punkt)

Dieser Teil der Aufgabenstellung ist in der Funktion `highlightSeams(...)` zu implementieren. Hierbei werden die minimalen Seams, welche mit der Funktion `calculateCostImage(...)` berechnet wurden, visualisiert. Um die berechneten Seams eines Bildes darzustellen, setzen Sie für jedes Pixel im RGB-Bild, die einem der Seams zugeordnet sind, den blau und grün Wert auf 0, und den rot Wert auf 255.

Folgende Formel wird für das Hervorheben der Pfade im Ausgabebild H verwendet:

$$H(i,j) = \begin{cases} (0,0,255) & \text{wenn } (i,j) \in \text{Pfad } P, \\ I(i,j) & \text{sonst.} \end{cases}$$

Hierbei sind i und j der Zeilen- bzw. Spaltenindex des Eingabebildes I.

Abbildung 6 zeigt die hervorgehobenen minimalen Seams. Die Anzahl der eingezeichneten Seams ist gleich dem Parameter `rm_seam_count`. Dies ist am unteren Rand des Bildes gut zu erkennen, da hier noch alle Seams aufgeteilt sind. Je näher die Seams dem oberen Bildrand kommen, desto eher vereinigen sich die Seams zum minimalen Pfad.

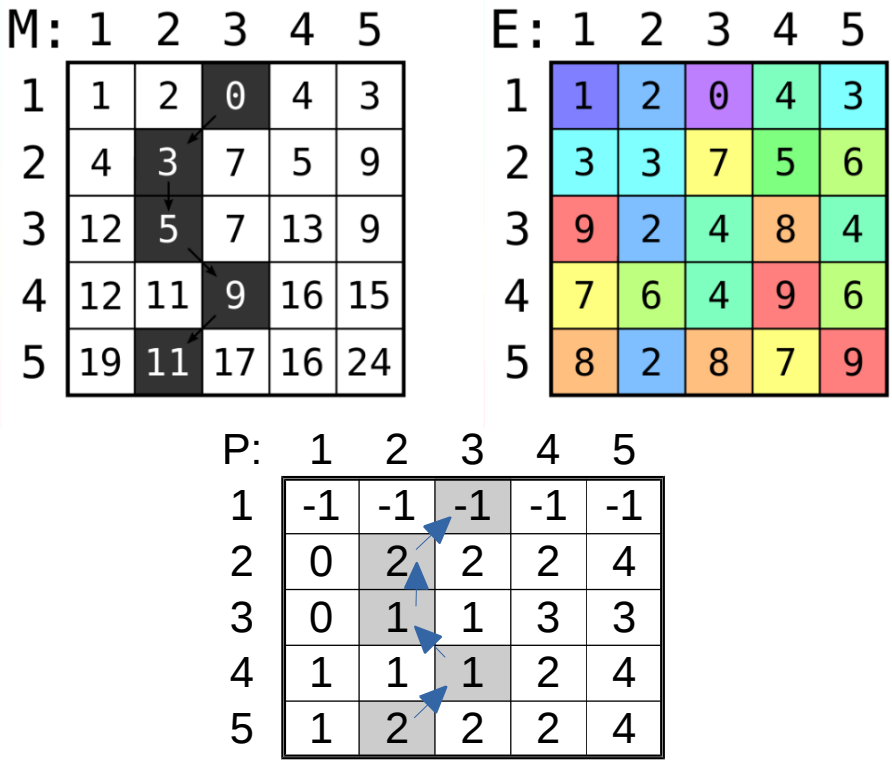


Abbildung 5: Beispielhafte Darstellung einer Gradienten Matrix E, Kosten Matrix M und Pfad Matrix P. Der minimale Pfad ist in der Kosten Matrix M eingezeichnet. Die Pfad Matrix P enthält den Zeilenindex für die darüber liegende Zeile, welcher benötigt wird, um das nächste Element des minimalen Seams mittels Backtracking zu bestimmen. Das Ende des Seams wird in der Pfad Matrix P mit dem Wert -1 gekennzeichnet.

2.4 Entfernen von Seams mit geringster Gesamtenergie

Das Verkleinern eines Bildes ist in der Funktion `reduceImage(...)` zu implementieren. Diese kann sowohl für vertikale, als auch für horizontale Verkleinerung verwendet werden. Lediglich die Übergabeparameter müssen angepasst werden.

2.4.1 Entfernen vertikaler Seams (2.5 Punkte)

Nachdem die minimalen vertikalen Seams mit der Funktion `calculateCostImage(...)` berechnet wurden, kann nun mit dem vertikalen Verkleinern des Bildes begonnen werden. Dazu muss der kürzeste Pfad aus dem Bild entfernt werden. Hierbei müssen folgende Schritte iterativ ausgeführt werden.

- Energieberechung mittels Gradienten (Abschnitt 2.1)

Abbildung 6: Hervorheben der minimalen Seams im Originalbild.

- Berechnung der minimalen Seams (Abschnitt 2.2)
- Löschen des minimalen Seams

Die Energieberechnung mittels Gradienten sowie die Berechnung der minimalen Seams wurde bereits in den vorherigen Aufgaben implementiert. Beim Löschen wird nun das Eingabebild I^t zeilenweise durchlaufen. Für jede Zeile y gibt es genau ein Pixel p, der Teil des kürzesten Pfades P ist: $p = (x, y) \in P$. Dieser Pixel wird beim Kopieren der Zeile in das Ausgabebild I^{t+1} ignoriert. Dieser Vorgang wird mit folgender Formel beschrieben:

$$I^{t+1}(y,j) = \begin{cases} I^t(y,j) & \text{wenn } j < x, \\ I^t(y,j+1) & \text{wenn } j \geq x. \end{cases}$$
$$0 \leq y < H,$$

wobei H die Bildhöhe beschreibt, $I^t(i,j)$ das Bild bei Iteration t, $I^{t+1}(i,j)$ das um einen Seam reduzierte Bild, und $1 \leq t \leq rm\_seam\_count$. Der Parameter `rm_seam_count` gibt an, wie oft diese Schritte wiederholt werden müssen. I^1 wird dem Eingabebild I initialisiert. Hierbei ist zu beachten, dass sich das Bild nach dem Löschen des minimalen Seams geändert hat, und deshalb die Ergebnisse der Energieberechnung sowie der minimalen Seams neu berechnet werden müssen. Abbildung 7 zeigt das Ergebnis des vertikalen Verkleinerns.

2.4.2 Entfernen horizontaler Seams (0.5 Punkte)

Aufgabe ist es nun, das Bild horizontal zu Verkleinern. Dazu müssen die minimalen Seams von der linken zur rechten Bildkante berechnet und entfernt werden. Hierfür ist es

(a) Originalbild.

(b) Vertikale Verkleinerung.

Abbildung 7: Resultat vertikale Verkleinerung.

sinnvoll die OpenCV Methode `cv::transpose(...)` auf das Eingabebild anzuwenden. Nun kann der im Abschnitt 2.4.1 implementierte Algorithmus verwendete werden, um das Bild horizontal zu verkleinern. Abbildung 8 zeigt das Ergebnis des horizontalen Verkleinerns.

(a) Originalbild.

(b) Horizontale Verkleinerung.

Abbildung 8: Resultat horizontale Verkleinerung.

Hilfreiche OpenCV-Methoden:

- `cv::transpose(...)`

2.5 Vertikale Bild Vergrößerung (2 Punkte)

Dieser Teil der Aufgabenstellung ist in der Funktion `enlargeImage(...)` zu implementieren. Um ein Bild zu vergrößern werden jene Seams verwendet, die beim Verkleinern berechnet werden. In jedem Schritt wird dabei beim Vergrößern parallel der Algorithmus zum Verkleinern auf das Bild I angewandt. Jene Seams, die beim verkleinerten Bild entfernt wurden, müssen beim vergrößerten Bild eingefügt werden. Beim Einfügen ist ebenfalls zu beachten, dass die Koordinaten der Seams vom kleinen Bild auf das große Bild zurückgerechnet werden müssen. Dies geschieht mit Hilfe der Offset Matrix O.

Berechnung der Offset Matrix O

Der Wert der Offset Matrix $O(i,j)$ beschreibt den vertikalen Offset für den Seam Pixel $p = (x,y) \in P$. Dieser vertikale Offset gibt an, um wie viele Pixel ein Seam, der im verkleinerten Bild gefunden wurde, im Ausgabebild verschoben werden muss. Die Offset Matrix O wird zeilenweise aktualisiert. Für eine Zeile i wird die Matrix wie folgt verändert:

$$O(i,j) = \begin{cases} O(i,j+1)+2 & \text{wenn } x \leq j < W-1, \\ O(i,j) & \text{sonst.} \end{cases}$$

W bezeichnet hier die Breite der Offset Matrix O. Der Grund warum der Offset um $+2$ erhöht wird ist, da sich jeweils beim Verkleinern und Vergrößern der Spaltenindex der nachfolgenden Pixel um -1 verschiebt. Darüber hinaus ändern sich bisherigen Koordinaten in der Offset-Matrix nach jedem Löschen eines Seams im verkleinerten Bild um 1. Daher müssen alle Pixel rechts nach dem Seam um eine Position verschoben werden. Die Offset Matrix O wird daher benötigt um etwaige Koordinatenfehler auszugleichen. Ein Beispiel, das die Funktionsweise der Offset Matrix O illustriert, ist in Abbildung 9 zu finden.

Einfügen eines Seams

Beim Einfügen eines neuen Seams ergibt sich der Farbwert jedes neuen Pixels des Seams als Mittelwert zwischen linkem und rechtem Nachbar-Pixel. Das Bild wird beim Einfügen zeilenweise durchgegangen. Für jede Zeile gibt es einen Pixel p der Teil des Seams P ist: $p = (x,y) \in P$. Für jede Zeile y wird das Bild nun folgendermaßen aktualisiert:

$$I^{t+1}(y,j) = \begin{cases} I^t(y,j) & \text{wenn } j < x + O(y,x) + 1, \\ \left\lfloor \frac{I^t(y,j)}{2.0} \right\rfloor + \left\lfloor \frac{I^t(y,j-1)}{2.0} \right\rfloor & \text{wenn } j = x + O(y,x) + 1, \\ I^t(y,j-1) & \text{wenn } j > x + O(y,x) + 1. \end{cases}$$

$I^t(i,j)$ ist das Bild bei Iteration t, $I^{t+1}(i,j)$ ist das um ein Pixel vergrößerte Bild, wobei $1 \leq t \leq add\_seam\_count$. Wichtig ist, dass jedes neue Pixel jeweils rechts vom ursprünglichen

O:

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

I:

	1	2	3	4	5
1	128	126	129	130	210
2	127	128	130	127	211
3	125	127	124	129	210
4	126	128	126	131	213
5	127	129	123	134	214

I:

	1	2	3	4	5
1	128	126	129	130	210
2	127	128	130	127	211
3	125	127	124	129	210
4	126	128	126	131	213
5	127	129	123	134	214

O:

	1	2	4	5
1	0	0	2	2
2	0	0	2	2
3	0	0	2	2
4	0	0	2	2
5	0	0	2	2

I:

	1	2	4	5
1	128	126	130	210
2	127	128	127	211
3	125	127	129	210
4	126	128	131	213
5	127	129	134	214

I:

	1	2	3	3,5	4	5
1	128	126	129	129	130	210
2	127	128	130	128	127	211
3	125	127	124	126	129	210
4	126	128	126	128	131	213
5	127	129	123	128	134	214

O:

	2	4	5
1	2	4	4
2	2	4	4
3	2	4	4
4	2	4	4
5	2	4	4

I:

	2	4	5
1	126	130	210
2	128	127	211
3	127	129	210
4	128	131	213
5	129	134	214

I:

	1	1,5	2	3	3,5	4	5
1	128	127	126	129	129	130	210
2	127	127	128	130	128	127	211
3	125	125	127	124	126	129	210
4	126	127	128	126	128	131	213
5	127	127	129	123	128	134	214

O:

	2	5
1	2	6
2	2	6
3	2	6
4	2	6
5	2	6

I:

	2	5
1	126	210
2	128	211
3	127	210
4	128	213
5	129	214

I:

	1	1,5	2	3	3,5	4	4,5	5
1	128	127	126	129	129	130	170	210
2	127	127	128	130	128	127	168	211
3	125	125	127	124	126	129	169	210
4	126	127	128	126	128	131	171	213
5	127	127	129	123	128	134	174	214

Abbildung 9: Beispiel für die Funktionsweise der Offset Matrix. In der linken Spalte ist die Offset Matrix O zu sehen. In der mittleren Spalte ist das verkleinernte Bild angezeigt. In der rechten Spalte ist das vergrößerte Bild angezeigt. Seams, die zu entfernen sind werden in der mittleren Spalte grau angezeigt. Eingefügte Seams sind grün markiert. Die Offset Matrix gibt stets die Offsets zum vergrößerten Bild an, bzw. $\frac{O}{2}$ die Offsets zum Eingabebild. Wenn man zum Beispiel die Koordinaten für den Pixel an der Stelle $p = (x, y)$ finden möchte, so errechnet sich seine Koordinaten im vergrößerten Bild mittels: $O(y, x) + x$ bzw. im Eingangsbild mittels $\frac{O(y,x)}{2} + x$. Für die Koordinate $p = (1, 0)$ würde das $O(0, 1) + 1 = 6 + 1 = 7$ bzw. $\frac{O(0,1)}{2} + 1 = 3 + 1 = 4$ ergeben, die beide zur Spalte mit Nummer 5 im vergrößerten bzw. verkleinerten Bild zeigen.

Pixel eingefügt wird. Falls der Seam am Bildrand verläuft, so verwenden Sie einfach den Pixel am Rand anstatt zu interpolieren. Abbildung 10 veranschaulicht diese Interpolation. Abbildung 11 zeigt das Ergebnis des vertikalen Vergrößerns.

	1	2	3	4
1	230	235	237	128
2	231	234	236	124
3	233	235	238	127

(a) Originalbild mit Seam. Der gefundene Seam ist grau markiert.

	1	2	3	4	5
1	230	235	235	237	128
2	231	234	235	236	124
3	233	235	238	182	127

(b) Vergrößertes Bild. Die neu eingefügten und interpolierten Werte sind grün markiert.

Abbildung 10: Einfügen eines neuen Seams. Im linken Bild ist der Seam, der gefunden wurde grau hinterlegt. Im rechten Bild ist das neue, um ein Pixel breitere Bild, dargestellt. Die neuen Werte (grün markiert) werden rechts vom Seam eingefügt.

(a) Originalbild.

(b) Vertikale Bild Vergrößerung.

Abbildung 11: Resultat vertikale Bild Vergrößerung.

2.6 BONUS: Vertikale Bild Verkleinerung mittels Entropie Berechnung (3 Punkte)

Dieser Teil der Aufgabenstellung ist in den Funktionen `calculateLocalEntropy(...)` und `reduceEntropy(...)` zu implementieren. Eine Gradientenmatrix ist nur eine mögliche Metrik die Bildinformation der Pixel quantitativ darzustellen. Eine weitere solche Energiefunktion des Bildes stellt die lokale Entropie dar. Wir definieren die globale Entropie H eines Graustufen Bildes durch die Formel:

$$H = -\sum_{k=0}^{M-1} p_k \log_2(p_k), \tag{7}$$

wobei M die Anzahl der Pixel im Bild ist und p_k die Wahrscheinlichkeit die mit der Graustufe k assoziiert wird ist. Da viele Grauwertstufen 0 sind und $\log_2(0) = -\infty$, definieren wir $0 \cdot \log_2(0) = 0$ um Multiplikation und Addition mit ∞ zu vermeiden.

Um die Wahrscheinlichkeit p_k zu berechnen wird ein Histogramm H_I für die Grauwerte im Bild berechnet. Dabei wird das Histogramm der Größe 256 zunächst mit 0 initialisiert. Für alle Pixel (i,j) im Bild wird nun der Eintrag $H_I[I(i,j)]$ um 1 erhöht, wobei $I(i,j)$ der Grauwert des Bildes I an der Stelle (i,j) ist. Die Wahrscheinlichkeit p_k berechnet sich nun durch:

$$p_k = \frac{H_I[k]}{M}, \tag{8}$$

wobei M die Anzahl der Pixel im Bild ist.

Die obige Formel ergibt die Entropie für das gesamte Bild. Dies ist allerdings als Energiefunktion für Seam Carving nicht sehr nützlich, da dies ein skalarer Wert ist der die Energie des gesamten Bildes beschreibt. Eine Möglichkeit dieses Problem zu lösen ist, mit einem Fenster das gesamte Bild zu scannen, und für jedes Pixel im Bild die Entropie lokal in einem Fensters zu berechnen. Daraus ergibt sich die lokalisierte Entropie Matrix.

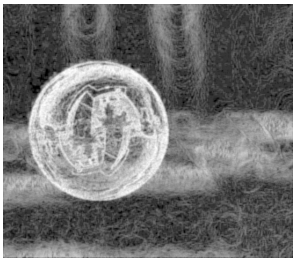
Die Funktion `calculateLocalEntropy(...)` soll das ursprüngliche RGB Bild in ein Graustufenbild verwandeln. Danach wird über jedes Pixel des Graustufenbildes iteriert und ein Fenster mit Seitenlänge von 5 Pixel betrachtet. An den Rändern soll das Bild mittels Duplikation erweitert werden (`CV_BORDER_REPLICATE`) um dem Fenster Platz zu geben. Analog zum Task 1a definieren wir das Fenster auf Pixel (i,j) durch einen Anker-Pixel im Zentrum des Fensters ($origin = (i,j)$) und der Nachbarschaft $origin \pm 2$. Berechnen Sie die Entropie H des Fensters gemäß der obigen Formel und speichern Sie den Wert H in eine Matrix mit `CV_32FC1`. Benutzen Sie diese Matrix als Energiematrix für die Seamberechnung in `reduceEntropy(...)`.

Normalisieren Sie die Entropie des Ausgabebildes auf den Wertebereich $[0,255]$. Das Resultat solle den Typ `CV_8UC1` haben. Als nächstes implementieren Sie die Funktion

`reduceEntropy(...)` wie in `reduceImage(...)` beschrieben, verwenden Sie dabei `calculateLocalEntropy(...)` anstatt `calculateGradientMag(...)`.

Abbildung 12 zeigt das Ergebnis dieser beiden Funktionen.

(a) Originalbild.



(b) Entropie (normalisiert).

(c) Vertikale Verkleinerung mittels Entropie.

Abbildung 12: Resultat der Entropie Berechnung und vertikale Verkleinerung.

Hilfreiche OpenCV-Methoden:

- `cv::copyMakeBorder(...)`

3 Ein- und Ausgabeparameter

Folgende Parameter sind in den Konfigurationsdateien angegeben:

- Eingabe - Pfad zum Originalbild: `image_path`
- Eingabe - Name des Bildes: `name_image`
- Eingabe - Pfad zum Ausgabe Ordner: `output_dir`
- Eingabe - Anzahl der zu löschenden Seams: `delete_seams_count`
- Eingabe - Anzahl der zu duplizierenden Seams: `enlarge_seams_count`
- Eingabe - Anzahl der zu löschenden Seams mittels Entropie: `delete_entropy_count`
- Ausgabe - Gradientenbild: `00_out_grad_mag`
- Ausgabe - Kosten Matrix: `01_out_costImage`
- Ausgabe - Hervorgehobene Seams: `02_out_seams`
- Ausgabe - Vertikal verkleinert: `03_out_vertical`
- Ausgabe - Horizontal verkleinert: `04_out_horizontal`
- Ausgabe - Seams hinzugefügt: `05_out_enlarge`
- Ausgabe - Bonus Entropie: `06_out_entropy`
- Ausgabe - Bonus vertikal verkleinert mittels Entropie: `07_out_bonus`

4 Programmgerüst

Die folgende Funktionalität ist in dem vom ICG zur Verfügung gestellten Programmgerüst bereits implementiert und muss von Ihnen nicht selbst programmiert werden:

- Die Konfigurationsdatei (JSON) wird vom Programmgerüst gelesen.
- Lesen des Eingabebildes und der Eingabeparameter

- Schreiben der Ausgabebilder in die dafür vorgesehenen Ordner

5 Abgabe

Die Aufgaben bestehen jeweils aus mehreren Schritten, die zum Teil aufeinander aufbauen, jedoch unabhängig voneinander beurteilt werden. Dadurch ist einerseits eine objektive Beurteilung sichergestellt und andererseits gewährleistet, dass auch bei unvollständiger Lösung der Aufgaben Punkte erzielt werden können.

Wir weisen ausdrücklich darauf hin, dass die Übungsaufgaben von jedem Teilnehmer eigenständig gelöst werden müssen. Wenn Quellcode anderen Teilnehmern zugänglich gemacht wird (bewusst oder durch Vernachlässigung eines gewissen Mindestmaßes an Datensicherheit), wird das betreffende Beispiel bei allen Beteiligten mit 0 Punkten bewertet, unabhängig davon, wer den Code ursprünglich erstellt hat. Ebenso ist es nicht zulässig, Code aus dem Internet, aus Büchern oder aus anderen Quellen zu verwenden. Es erfolgt sowohl eine automatische als auch eine manuelle Überprüfung auf Plagiate.

Die Abgabe der Übungsbeispiele und die Termineinteilung für die Abgabegespräche erfolgt über ein Webportal. Die Abgabe erfolgt ausschließlich über das Abgabesystem. Eine Abgabe auf andere Art und Weise (z.B. per Email) wird nicht akzeptiert. Der genaue Abgabeprozess ist im TeachCenter beschrieben.

Die Tests werden automatisch ausgeführt. Das Testsystem ist zusätzlich mit einem Timeout von 7 Minuten versehen. Sollte Ihr Programm innerhalb dieser Zeit nicht beendet werden, wird es vom Testsystem abgebrochen. Überprüfen Sie deshalb bei Ihrer Abgabe unbedingt die Laufzeit Ihres Programms.

Da die abgegebenen Programme halbautomatisch getestet werden, muss die Übergabe der Parameter mit Hilfe von entsprechenden Konfigurationsdateien genauso erfolgen wie bei den einzelnen Beispielen spezifiziert. Insbesondere ist eine interaktive Eingabe von Parametern nicht zulässig. Sollte aufgrund von Änderungen am Konfigurationssystem die Ausführung der abgegebenen Dateien mit den Testdaten fehlschlagen, wird das Beispiel mit 0 Punkten bewertet. Die Konfigurationsdateien liegen im JSON-Format vor, zu deren Auswertung steht Ihnen `rapidjson` zur Verfügung. Die Verwendung ist aus dem Programmgerüst ersichtlich.

Jede Konfigurationsdatei enthält zumindest einen Testfall und dessen Konfiguration. Es ist auch möglich, dass eine Konfigurationsdatei mehrere Testfälle enthält, um gemeinsame Parameter nicht mehrfach in verschiedenen Dateien spezifizieren zu müssen. In manchen Konfigurationsdateien finden sich auch einstellbare Parameter, die in Form eines `select` Feldes vorliegen. Diese sollen die Handhabung der Konfigurationsdateien erleichtern und

ein einfaches Umschalten der Modi gewährleisten.

Es steht Ihnen frei, z.B. zu Testzwecken eigene Erweiterungen zu implementieren. Stellen Sie jedoch sicher, dass solche Erweiterungen in Ihrem abgegebenen Code deaktiviert sind, damit ein Vergleich der abgegebenen Arbeiten mit unserer Referenzimplementierung möglich ist.

Die Programmgerüste, die zur Verfügung gestellt werden, sind unmittelbar aus unserer Referenzimplementierung abgeleitet, indem nur jene Teile entfernt wurden, die dem Inhalt der Übung entsprechen. Die Verwendung dieser Gerüste ist nicht zwingend, aber Sie ersparen sich sehr viel Arbeit, wenn Sie davon Gebrauch machen.

Literatur

[1] Shai Avidan, Ariel Shamir. Seam Carving for Content-Aware Image Resizing. In Proceedings of ACM SIGGRAPH, 2007.