Einführung in die Parallelverarbeitung
Prof. Claudia Fohry
M.Sc. Rüdiger Nather

Universität Kassel
Wintersemester 2025
Abgabe: 10.2.2026, 10:00 Uhr

Projektaufgaben 2. Teil

OpenMP Offloading

Lösungsideen dürfen nicht ausgetauscht werden. Eventuelle inhaltliche oder organisatorische Fragen stellen Sie bitte im Diskussionsforum in Moodle. Achten Sie dabei darauf, nichts über Ihre Lösung zu verraten. Sollte dies in Ausnahmefällen nicht möglich sein, schicken Sie eine E-Mail an r.nather@uni-kassel.de

Bitte geben Sie die Lösungen bis spätestens **10.2.2026, 10:00 Uhr** per E-Mail ab.

a) `SeamCarving.tar.gz`

In dieser Aufgabe geht es um eine Implementierung von *Seam Carving* mittels OpenMP Offloading. Seam Carving ist ein Bildskalierungsverfahren, welches möglichst wenige wahrnehmbare Verzerrungen im Bild verursacht. Ein Bild, welches eingelesen wird, besteht aus einer 2D-Bildmatrix, deren Bildpunkte (*Pixel*) Tripel der RGB-Farbwerte (Rot, Grün, Blau) sind. Die Farbwerte sind Ganzzahlen im Intervall [0, 255] und können somit bspw. als `char`-Werte gespeichert werden. Das Vergrößern eines Bildes um k Pixel auf der horizontalen Achse hat in unserer Variante folgenden Ablauf:

1. Das eingelesene RGB-Bild wird in ein Grayscale-Bild umgewandelt. Dazu wird für jeden Pixel der Grauwert wie folgt berechnet:

   $$Y = 0.299R + 0.587G + 0.114B$$

   wobei R, G und B die Rot, Grün und Blau-Werte des Pixels sind. Siehe dazu auch: https://en.wikipedia.org/wiki/Grayscale

2. Für jeden Pixel des Grayscale-Bildes wird eine *Pixelenergie* e berechent. Diese setzt sich zusammen aus dem Gradienten (welcher das Vorhandensein von Kanten misst) und der Entropie (welche die "Komplexität" der Umgebung des Pixels misst).

   - Der Gradient e_1 wird mit Hilfe des Sobel Operators berechnet, welcher auf der Anwendung von zwei 3×3-Kerneln (Filtermatrizen) basiert, die auf das Bild angewendet werden, um die Gradienten in horizontaler (G_x) und vertikaler (G_y) Richtung zu berechnen. Die Gradienten zeigen die Änderung der Helligkeit und damit die Position von Kanten.

     Die beiden Sobel-Kernel werden über das gesamte Bild "geschoben" (eine Operation, die als Faltung oder Konvolution bekannt ist). Der Kernel für G_x und G_y sehen folgendermaßen aus:

     $$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

     Für jeden Pixel im Bild wird der entsprechende Bereich unter dem Kernel mit den Werten des Kernels multipliziert und die Ergebnisse summiert, um den Gradienten

in X- bzw. Y-Richtung zu erhalten. Die Stärke des Gradienten an jedem Pixel wird berechnet, in dem die Summe der Beträge der Gradienten in X- und Y- Richtung berechnet wird, was die Stärke der Helligkeitsänderung an diesem Punkt angibt.

- Die Entropie wird mit Hilfe von Histogrammen berechnet. Dafür wird in einer 9×9-Umgebung des Pixels gezählt, wie oft jeder einzelne Grauwert vorkommt. Dabei werden jeweils Bereiche von Grauwerten zusammengefasst.

  Dafür wird der maximale und minimale Grauwert (g_{max} und g_{min}) aus der Umgebung des Pixels bestimmt, und 9 Zähler $\texttt{bin}[i]$ definiert. Der i-te Zähler wird dann jeweils für Werte aus dem Intervall $\left[g_{min} + i\frac{g_{max}-g_{min}}{9}, g_{min} + (i+1)\frac{g_{max}-g_{min}}{9}\right)$ inkrementiert.

  Die Entropie e_{Ent} wird dann mit

  $$e_{Ent} = -\sum_{i=0}^{8} \frac{\texttt{bin}[i]}{81} \cdot \log_2\left(\frac{\texttt{bin}[i]}{81}\right)$$

  berechnet.

Die gesamte Pixelenergie ergibt sich dann als Summe des Gradienten und der Entropie: $e = e_1 + e_{Ent}$.

3. Finde k Pfade von Pixeln der obersten Zeile bis zur untersten Zeile, deren Summen der Pixelenergien am kleinsten sind. Ein Pfad enthält nur benachbarte Pixel. Zwei Pixel sind benachbart, wenn sich deren horizontale und vertikale Koordinate maximal um 1 unterscheiden. Bspw. sind die Pixel $(5,5)$ und $(6,6)$ benachbart. D.h., es gibt horizontale, vertikale und diagonale Nachbarn. Diese Pfade werden *Seams* genannt.

   Für diesen Schritt bietet sich die dynamische Programmierung an: Die minimalen Pixenergiesummen der Pixel der obersten Zeile sind gleich ihrer eigenen Pixelenergien. Zeilenweise, beginnend bei der zweiten Zeile, wird die minimale Pixelenergiesumme jedes Pixels aus der eigenen Pixelenergie und der minimalen Pixelenergiesumme der Nachbarn in der darüberliegenden Zeile berechnet. Die folgenden drei Abbildungen (aus https://en.wikipedia.org/wiki/Seam_carving) illustrieren den Ablauf. Im Gegensatz zu Schritt 1 sind in diesem Schritt die äußeren Pixel nicht mit den Pixeln auf der anderen Seite der Bildmatrix benachbart (also kein Torus). Die roten Zahlen sind die Pixelenergien aus Schritt 1. Die schwarzen Zahlen stellen die in diesem Schritt berechneten minimalen Pixelenergiesummen dar.

   Die minimalen Pixelenergiesummen der zweiten Zeile werden für jeden Pixel aus der eigenen Pixelenergie, sowie dem kleinsten Wert der drei bzw. zwei Nachbarn in der darüberliegenden Zeile gebildet:

   Dieser Prozess setzt sich bis in die unterste Zeile fort:

   Der Seam mit der kleinsten Pixelenergiesumme kann nun durch Suchen eines Minimums in der untersten Zeile gefunden werden. Wir können den Seam durch Ablaufen von unten nach oben konstruieren:

   Wir bestimmen k Seams durch Auswählen der k kleinsten Pixelenergiesummen der untersten Zeile.

4. Für jeden Pixel $p = (x, y)$ eines Seams wird nun ein neuer Pixel mit der gleichen Farbe, Pixelenergie und Pixelenergiesumme von p erzeugt. Jeder neue Pixel wird jeweils rechts neben p an Position $(x+1, y)$ eingefügt. Dabei verschieben sich alle alten Pixel auf der rechten Seite des Seams um 1 Pixel nach rechts. Das Bild ist nach diesem Schritt um eine Spalte verbreitert. Dieser Schritt wird für alle k Seams wiederholt. Die Pixelenergien und minimalen Pixelenergiesummen werden nach dem Einfügen einer Spalte nicht neu bestimmt.

Folgendes Beispiel zeigt ein Ausgangsbild (`moon.jpg`, RGB) und das entsprechende Ergebnisbild (Grayscale) nach einer Verbreiterung von 400 Pixeln:

Aufgabenstellung: Vorgegeben ist ein sequentielles Programm (verfügbar in Moodle), welches die oben beschriebene Seam Carving-Variante implementiert. Weitere Details des Algorithmus können dem sequentiellen Beispielprogramm entnommen werden. Parallelisieren Sie das vorgegebene Programm möglichst effizient durch Einfügen von OpenMP Offloading-Direktiven und optimieren Sie es hinsichtlich aller in der Vorlesung behandelten Performancekriterien. Sie dürfen das Verfahren und den Programmcode beliebig anpassen, insofern dies Ihnen sinnvoll erscheint und die Programmausgabe weiterhin mit dem sequentiellen Programm übereinstimmt. Sie dürfen Teile der Berechnung auf dem Host ausführen.

Das Programm wird wie folgt aufgerufen:

```
> ./enlarge inputFile outputFile k
```

Dabei gibt:

- `inputFile` den Dateipfad, der zu verändernden JPEG-Bilddatei,
- `outputFile` den Dateipfad, an dem das veränderte JPEG-Bild gespeichert werden soll, und
- `k` die Anzahl der hinzuzufügenden Bildmatrixspalten an.

Messen Sie innerhalb des Programms die Ausführungszeit. Die Zeitmessung soll direkt nach dem Einlesen der JPEG-Bilddatei beginnen, und direkt vor dem Schreiben der veränderten JPEG-Bilddatei stoppen. Beispiel:

```
> ./enlarge moon.jpg moon_enlarged.jpg 100
Execution time: 2.02 sec
```

Sie können das Beispielprogramm nutzen, um Ihr paralleles Programm auf Korrektheit zu überprüfen. Als Eingabe für Ihr Programm können Sie bspw. das obige Beispielbild `moon.jpg`, das in Moodle verfügbar ist, verwenden.