Übung

Datenvisualisierung und GPU-Computing

Dozenten: Michael Vetter
michael.vetter@uni-hamburg.de

Sommersemester 2015

Aufgabe 1: *Serielle Codeoptimierung*

Programme können durch Parallelisierung, z.B. mittels Threads, OpenMP oder MPI zum Teil erheblich beschleunigt werden. Bevor allerdings mit der Parallelisierung von Programmen begonnen wird, sollten sich Entwickler aber zunächst einmal Gedanken über die Algorithmen selber machen. Was bringt schließlich die beste Parallelisierung eines Algorithmus mit der Komplexität $O(n^3)$, wenn stattdessen auch ein Algorithmus mit der Komplexität $O(n^2)$ verwendet werden könnte.

Daher soll nun einmal ein möglichst effizienter serieller Algorithmus für das folgende Problem entwickelt werden:
Finden Sie alle Lösungen für x, y und z für die gilt $x + y + z = c$ mit $x, y, z, c \in \mathbb{N}_0$ und $c = const$.

Aufgabe 2: *Paralleles LIC*

Line Integral Convolution (LIC) ist eine Methode zur Strömungsvisualisierung bei der ein Pixelbild mit weißem Rauschen und ein normiertes Vektorfeld zu einem resultierenden Pixelbild kombiniert werden. Diese soll nun in ihrer einfachsten Version implementiert werden. Dazu wird zunächst ein Feld mit der Größe des zu erzeugenden Bildes zufällig mit 0 und 1 belegt. Anschließend wird für jedes Pixel des resultierenden Bildes eine Stromlinie mittels Euler-Integration vorwärts und rückwärts je 50 Schritte mit einem $\Delta d = 0.5$ integriert. Dabei ergibt sich der Wert für das zu berechnende Pixel aus der Summe der Werte "unter der Linie" (siehe Abb. 1).

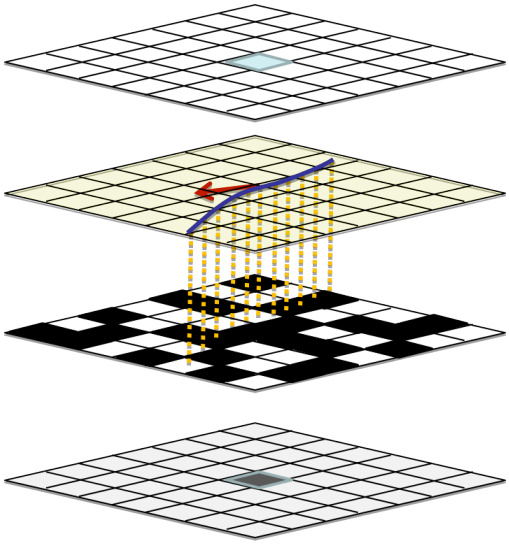

Abbildung 1: Line Integral Convolution (LIC)

Zur Berechnung des Vektorfeldes soll folgende Formel verwendet werde:

$$\vec{v} = \frac{\vec{v}_f}{|\vec{v}_f|}, \text{ mit } \vec{v}_f = \frac{\vec{v}_1}{|\vec{v}_1|^2} + \frac{\vec{v}_2}{|\vec{v}_2|^2}$$

$$\vec{v}_1 = \begin{pmatrix} -(y - \frac{2}{8} * Y) \\ x - \frac{1}{5} * X \end{pmatrix} \qquad \vec{v}_2 = \begin{pmatrix} y - \frac{5}{8} * Y \\ -(x - \frac{7}{10} * X) \end{pmatrix}$$

$\begin{pmatrix} x \\ y \end{pmatrix}$ = aktuelle Position

$\begin{pmatrix} X \\ Y \end{pmatrix}$ = Auflösung

Daraus entsteht eine Darstellung gemäß Abb. 2. Entwickeln Sie zunächst entsprechend dieser Vorgaben einen seriellen Algorithmus zur Berechnung einer solchen LIC-Darstellung. Parallelisieren Sie ihren Algorithmus anschließend mittels OpenMP.

Zum Erzeugen eines BMP-Bildes können Sie ein externes Codefragment, welches unter Stine zum Download bereitsteht verwenden. Dieses bietet eine Klasse *bitmap_image*, von welcher im Wesentlichen der Konstruktor *bitmap_image(int x, int y)* sowie die Funktionen *clear()*, *set_pixel(int x, int y, char R, char G, char B)* und *save_image(string filename)* benötigt werden.

Aufgabe 3: *Hello MPI*

Schreiben Sie ein mittels MPI parallelisiertes Programm, bei welchem jeder Prozess seine ID sowie die Anzahl der gestarteten Prozesse ausgibt und terminiert.

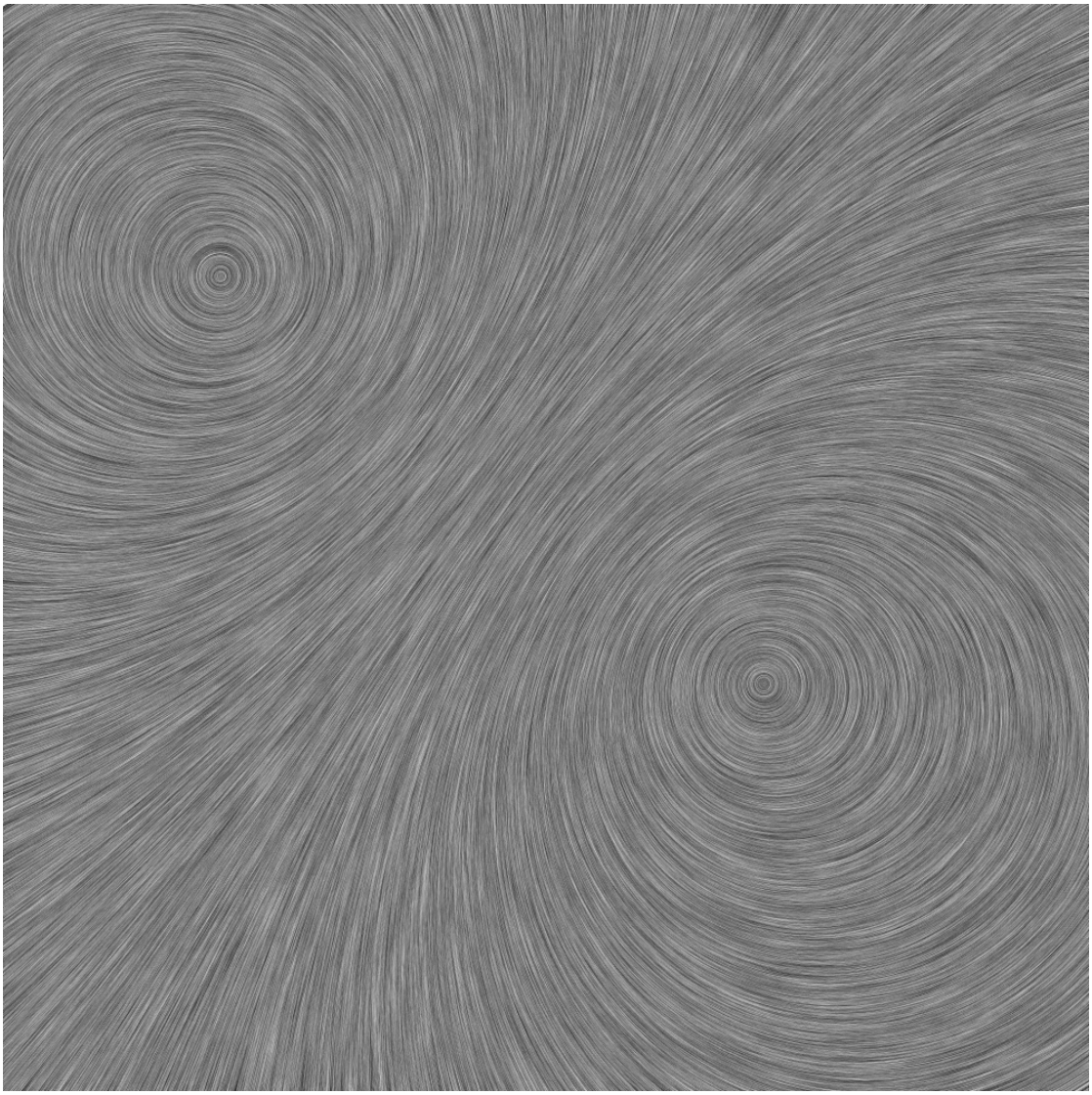

Abbildung 2: Line Integral Convolution (LIC) des zu implementierenden Vektorfeldes

Aufgabe 4: *Paralleles LIC - MPI*

Parallelisieren Sie den Algorithmus zur Linienintegralfaltung aus Aufgabe 2 nun für Distributed-Memory-Systeme mittels MPI.