# Projekt A: Heatmap

6. Januar 2023

Ziel dieser Projektaufgabe ist es, ein Programm zu schreiben, welches aus gegebenen Messdatensätzen eine Bitmap Bilddatei (BMP-Format) im Stil einer *Heatmap* erzeugen kann. Bitte lesen Sie sich dieses Aufgabenblatt sorgfältig und vollständig durch, um Informationen und Hinweise für die Lösung zu erhalten. Beachten Sie außerdem die Anweisungen auf dem allgemeinen Informationsblatt zu den Projektaufgaben. Bitte reichen Sie die Ergebnisse Ihrer Meilensteine 2 Tage vor dem jeweiligen Termin mit Ihrem Tutor über Git ein.

## Spezifikation

Im Zusammenhang mit dem Sedimentmanagement der Elbe nimmt ein Expertenteam regelmäßig Messdaten vom Höhenprofil des Flussgrundes mit Hilfe eines Unterwasserroboters auf. Diese Daten werden in einer CSV-Datei abgespeichert und umfassen die $x$-, $y$- und $z$-Koordinaten des Profils. Das Team hat bislang zur Visualisierung der Messdaten gängige Tabellenkalkulationssoftware verwendet, welche Höhenliniendiagramme erstellt hat. Bei größeren Datenmenge ist diese allerdings nur schwer handhabbar. Außerdem kommt es beim Austausch mit anderen Teams immer wieder zu Komplikationen, da diese andere Softwareversionen verwenden, die das Öffnen der Dateien ohne größeren Aufwand unmöglich machen. Von daher beschließt das Expertenteam eine Software-Entwicklungs-Firma zu engagieren, die ihnen ein Programm zum Aufbereiten der Messdaten schreiben soll.

In einem gemeinsamen Kick-Off-Meeting wurde eine Anforderungsspezifikation ausgearbeitet, die folgende Punkte enthält:

1. Datenformat / Eingabe
   - Eigenschaften der Daten
     - Die Messdatensätze bestehen aus drei Dezimalwerten ($x$, $y$, $z$-Koordinaten) in der Einheit Meter.
     - Die Daten sind als regelmäßiges Grid mit konstanten Schrittweiten in $x$- und $y$-Richtung aufgebaut
   - Die Messdaten liegen als CSV-Dateien vor (Dateiendung *.csv*)[1]. und sind wie folgt aufgebaut:
     a) Die 1. Zeile enthält die minimale und maximale $x$-Koordinate sowie die Auflösung des Grids in $x$-Richtung, d. h. $x_{\min}, x_{\max}, x_{\text{step}}$.
     b) Die 2. Zeile enthält die minimale und maximale $y$-Koordinate sowie die Auflösung des Grids in $y$-Richtung, d. h. $y_{\min}, y_{\max}, y_{\text{step}}$.
     c) Es folgen die Messdaten $(x, y, z)$ in aufsteigender Sortierung nach $x$- und $y$-Koordinaten.
     d) Die Datensätze sind vollständig, d.h. alle Koordinaten sind enthalten.

[1] Bei CSV-Dateien handelt es sich um Dateien mit Datensätzen im Textformat. Die Datensätze sind typischerweise mit einem Zeilenumbruch getrennt, die einzelnen Elemente eines Datensatzes durch Kommata (Comma-Separated Values (CSV)).

e) Die Werte innerhalb aller Zeilen sind komma-getrennt.

2. Datenverarbeitung und Ausgabe
   - Aus den Eingabedaten soll eine **Heatmap** erzeugt und als Bitmap Bilddatei im BMP-Format (Dateiendung *.bmp*) gespeichert werden.
   - Die Auflösung der Bilddatei soll mindestens der Anzahl von Messpunkten in $x$- und $y$-Richtung entsprechen.
   - Größere Auflösungen sollen unterstützt werden. Hierzu muss ein Verfahren zur Bestimmung von Zwischenpunkten (im Grid) implementiert werden.
   - Die Höhen-/Tiefen ($z$-Koordinate) sollen in 10 diskreten Stufen farblich dargestellt werden.
   - Die Farbskalierung soll von Dunkelblau für den tiefsten Punkt im Messdatensatz über Grün zu Rot als höchstem Punkt reichen.
3. Eingabe
   - Auswahl des jeweiligen Messdatensatzes als Kommandozeilenparameter
   - Angabe der Bildgröße (Breite und Höhe in Anzahl Pixel) über zwei Kommandozeilenparameter (optional: automatische Berechnung der minimalen Auflösung, wenn Breite/Höhe nicht angegeben)
   - Beispiel: `./bitmap harburger-binnenhafen.csv 1000 1000`
     Erstellt eine Heatmap aus den Daten in *harburger-binnenhafen.csv* mit einer Auflösung von 1000×1000 Pixeln

Hinweise

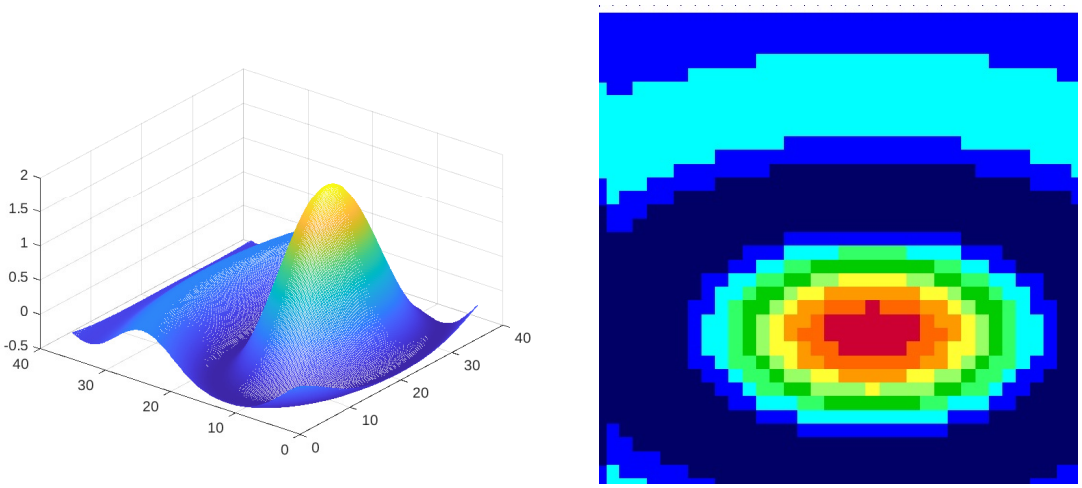
- Die Anzahl der Datenpunkte (Rasterpunkte) ist nicht explizit angegeben, kann aber aus den anderen Angaben berechnet werden.
- Zur Interpolation der Daten gibt es keine Vorgaben. Sie können hier gleichfarbig auffüllen, linear interpolieren oder ein anderes sinnvolles Verfahren implementieren.
- Die Rasterdaten und das Bitmap lassen sich sehr effizient als Array darstellen.
- Zum Erzeugen der BMP-Grafik wird eine Bibliothek zur Verfügung gestellt (s. u.). Diese verwendet das 32-bit Pixelformat[2] (32 bpp) `0x00RRGGBB`.

## Heatmaps

Um ein besseres Verständnis der Aufgabenstellung zu erhalten, stellen wir Ihnen kurz die wesentlichen Aspekte von Heatmaps zusammen. Bei Heatmaps handelt es sich um eine alternative Darstellung von dreidimensionalen Daten wie sie in Fig. 1a dargestellt sind. Die $z$-Komponente wird farblich koodiert und in einem zweidimensionalen Raster dargestellt. Diese Darstellungsart besitzt z.B. den Vorteil, dass keine Bildteile verdeckt werden. Allerdings gibt es auch Nachteile. Eine Gegenüberstellung ist hier aber nicht zielführend.

Häufig bietet es sich an, diskrete Farbwerte zur Darstellung zu verwenden, da hiermit die visuelle Interpretation der Daten einfacher ist (feingranulare Schattierungen bzw. Farbübergänge lassen sich nur schwer mit dem menschlichen Auge erfassen und vergleichen). Eine äquivalente Darstellung der 3D-Daten als Heatmap mit 10 Farben ist in Fig. 1b dargestellt.

[2] https://de.wikipedia.org/wiki/Windows_Bitmap




(a) Höhenprofil (b) Heatmap

Abbildung 1: Beispiel einer Heatmap aus gegebenen Messdatensatz

## Hinweise zur Implementierung

Beachten Sie die Aufgabenstellungen auf dem Blatt *Allgemeine Informationen*. Darüber hinaus finden Sie im Folgenden einige Hinweise, die Ihnen beim Lösen dieser Projektaufgabe helfen sollen.

### Datenverarbeitung

- Auslesen der Messdaten aus einer Datei
  - Die Messdatensätze wurden im CSV-Dateiformat gespeichert. Die Daten können zeilenweise mit der Funktion `fscanf()` und einem geeigneten Formatstring eingelesen werden.
  - Die Größe des Grids der Messdaten und damit auch die Anzahl der Gridpunkte lassen sich aus den übrigen Informationen berechnen.
  - Die Messdaten sollten zur weiteren Verarbeitung in einem Array gespeichert werden. Eine statische Speicherallokation (feste Größe) ist nicht zielführend; verwenden Sie stattdessen dynamische Speicherverwaltung, um auch mit sehr großen Bildern bzw. Messdaten umgehen zu können.
- Messdaten-Cluster
  - Definieren Sie Cluster für die Höhenkoordinaten ($z$-Koordinate) und ordnen Sie jedem Cluster eine Farbe zu, sodass Sie den Forderungen aus der Spezifikation entspricht.
  - Bestimmen Sie Zuordnung der Cluster zu den Höhenkoordinaten ($z$-Koordinate) auf Basis der tatsächlichen Messdaten. Hierdurch erreichen Sie eine sinnvolle Zuordnung von Messdaten zu Clustern.
  - Verwenden Sie das 32 bpp Farbformat des BMP-Dateiformats. Wählen Sie eine geeignete und elegante Möglichkeit, um den Clustern die Farbwerte zuzuordnen.
- Benutzerfreundlichkeit

- Denken Sie daran, dass Sie später nur die ausführbare Datei an die Anwender:innen übergeben werden. Ihr Programm sollte also in der Lage sein beliebige Messdatensätze zu verarbeiten. Gleiches gilt auch für die frei wählbare Breite bzw. Höhe der Heatmap. Mit Hilfe von Eingaben über Kommandozeilenparameter können Sie das Problem lösen.
- Ihnen ist nicht bekannt, wie groß der zu verarbeitende Messdatensatz ist.

## Bitmap-Dateiformat

Das Bitmap-Dateiformat BMP ist ein Dateiformat für Rastergrafiken zum Darstellen von zweidimensionalen digitalen Bildern. Die Grundidee im Zusammenhang mit der geforderten Aufgabe ist, dass Sie ein Raster an Bildpunkten mit einer definierten Breite und Höhe erstellen und jedem Rasterpunkt dann den Farbwert entsprechend der Höhenkoordinate $z$ zuordnen.

- Um Ihnen die Arbeit zu erleichtern, haben wir Ihnen eine Vorlage für eine BMP-Bibliothek bestehend aus C-Datei und Header zur Verfügung gestellt (verfügbar im öffentlichen Repository für die Veranstaltung). Diese Bibliothek enthält die Funktionen `bmp_create()` und `bmp_write_N_byte()`. Die Funktion `bmp_write_N_byte()` wird dabei lediglich in `bmp_create()` aufgerufen und ist daher kein Bestandteil des öffentlichen Interfaces im Header. Die Funktion `bmp_create()` erzeugt eine BMP-Datei auf Basis des 32 bpp Farbformats. Hierzu müssen Sie den Dateinamen, die Pixelkarte und deren Breite und Größe übergeben. Beachten Sie, dass es sich bei dem Array um ein eindimensionales Array handelt.
  Schauen Sie sich die Implementierung der Funktionen an. Sie erhalten dort Informationen über deren Funktion und geforderte Datentypen.
- Überlegen Sie sich eine Möglichkeit, wie Sie die Datenpunkte aus dem kartesischen Koordinatensystem in die Position der Rasterpunkte in der Heatmap umrechnen können.
- Ihre BMP-Datei sollte variabel skalierbar sein – die Nutzer:innen können die Höhe und Breite für die Heatmap wählen. Es wird dabei vorkommen, dass die Anzahl der gewählten Pixel nicht mit der Anzahl der Messdaten übereinstimmt. Dennoch sollten alle Bildpunkte der finalen Heatmap mit einer Farbe belegt sein. Weiterführende Informationen hierzu sind im nachfolgenden Abschnitt dargestellt.

## Interpolation

Eine wesentliche Herausforderung bei der Implementierung ist das Färben der Pixel, wenn die Auflösung des Bildes größer ist als die Auflösung der Messdaten; sprich: es gibt mehr Bildpunkte als Datensätze. Betrachten wir als Beispiel den Datensatz *test9.csv*:

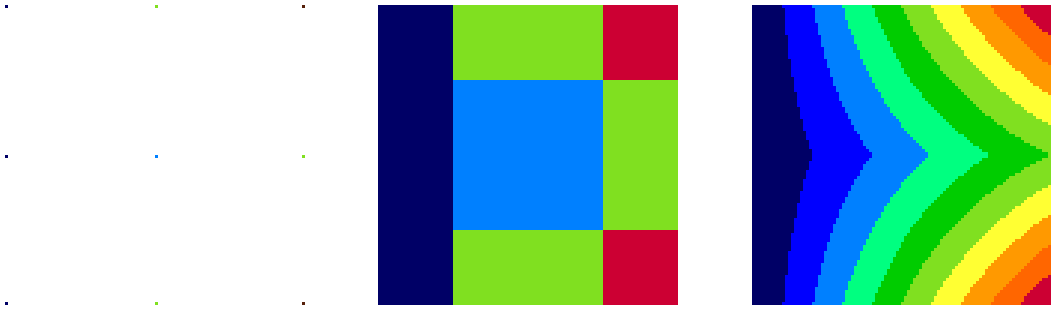
```
0,1,0.5
0,1,0.5
0,0,0
0,0.5,0
0,1,0
0.5,0,0.5
0.5,0.5,0.25
0.5,1,0.5
1,0,1
1,0.5,0.5
1,1,1
```

Soll dieser mit einer Auflösung von 50×50 Pixeln als Heatmap mit 10 Farben dargestellt werden, sind nur 9 Bildpunkte unmittelbar bestimmt. In diesem Fall müssen Sie Zwischenwerte für alle anderen Pixel bestimmen. Hierfür gibt es verschiedene Ansätze, die in Fig. 2 dargestellt sind. Die Rasteroption in Fig. 2a ist dabei nicht erstrebenswert.

Wenn Sie wie folgt vorgehen, lässt sich das Problem gut handhabbar lösen:

1. Belegen Sie im ersten Schritt nur die Pixel, die genau den Messpunkten entsprechen. Sie sollten als Ergebnis ein gleichmäßig verteiltes Raster wie in Fig. 2a erhalten.
2. Im nächsten Schritt sollten Sie sich eine Methodik überlegen, die bislang noch nicht eingefärbten Bildpunkte mit einer Farbe zu besetzen.
   - Eine recht einfache Möglichkeit ist das Übernehmen der Farbe des nächstgelegenen, bereits gefärbten Pixels. Ein mögliches Ergebnis ist in Fig. 2b dargestellt.
   - Alternativ können Sie auch eine lineare Interpolation zwischen benachbarten Messpunkten nutzen. Beim erfolgreichen Implementieren, erhalten Sie dann das Endergebnis in Fig. 2c.

Selbstverständlich können Sie auch andere Implementierungen ausprobieren.



(a) Rasterisierung mit weißem Hintergrund

(b) Interpolation mit nächstem definierten Punkt (*nearest neighbor*)

(c) Lineare Interpolation

Abbildung 2: Vergleich einer einfachen Rasterisierung und verschiedenen Iterpolationsarten