EX2_2 Mean Shift

Froschauer Michael - S2410454009

12. April 2025

1 Mean Shift Clustering Algorithmus

Ziel ist es, einen **Mean Shift Clustering Algorithmus** zu entwickeln, der allein basierend auf der Bandwidth Farbcluster in Farbbildern erkennen und gruppieren kann.

Die Bandwidth definiert dabei den Radius der Nachbarschaft (Einflussbereich) eines Punktes im Merkmalsraum – hier also im Farbraum. Sie bestimmt also, wie weit entfernte Punkte noch zur Berechnung eines neuen Schwerpunkts berücksichtigt werden. Eine zu große Bandwidth führt zu wenigen, groben Clustern, während eine zu kleine Bandwidth viele, feinere Cluster erzeugt.

Funktionsweise des Algorithmus

Der Mean Shift Algorithmus basiert auf einem iterativen Verfahren, bei dem Datenpunkte (hier: Pixel im Farbraum) in Richtung dichterer Regionen verschoben werden. Dazu erfolgt in jeder Iteration Folgendes:

- Für jeden Pixel wird die euklidische Distanz zu allen anderen Pixeln berechnet.
- Anhand dieser Distanzen wird mittels eines Gauß-Kernels ein gewichteter Mittelwert gebildet.
- Der Pixel wird in Richtung dieses gewichteten Schwerpunkts verschoben.
- Dieser Schritt wird wiederholt, bis entweder alle Pixel konvergiert sind (d. h. ihre Verschiebung ist kleiner als ein definierter Schwellwert ε) oder eine maximale Anzahl an Iterationen erreicht wurde.

Durch dieses Verfahren bewegen sich die Pixel in Regionen höherer Dichte im Farbraum, was zu einer automatischen Gruppierung (Clustering) führt. Ein großer Vorteil: Die Anzahl der Cluster muss im Vorhinein nicht bekannt sein.

Parallele Umsetzung

Da die Verschiebung jedes Pixels unabhängig von anderen Punkten innerhalb einer Iteration erfolgt, eignet sich der Algorithmus sehr gut zur Parallelisierung. Dadurch kann die Rechenzeit erheblich reduziert werden. Dies wurde in dieser Implementierung auch eingebaut.

Zentrale Hauptfunktion

Die Hauptfunktion `mean_shift_color_pixel` ist das Herzstück des Algorithmus:

```
def mean_shift_color_pixel(in_pixels: np.ndarray,
                           bandwidth: float,
                           epsilon: float = 1e-3,
                           max_iter: int = 1000,
                           iteration_callback=None)
```

Diese Funktion verschiebt iterativ alle Eingabepixel im Farbraum zu Bereichen höherer Dichte mithilfe des Mean Shift Algorithmus und eines Gauß-Kernels. Sobald alle Pixel konvergiert sind oder die maximale Iterationsanzahl erreicht ist, werden die zugehörigen Cluster und deren Schwerpunkte ermittelt und zurückgegeben. Optional kann ein Callback zur Visualisierung zwischengeschaltet werden.

Verarbeitung einzelner Pixel

Die folgende Funktion wird parallel von der Hauptfunktion aufgerufen. Sie prüft zunächst, ob der Pixel bereits konvergiert ist (um unnötige Rechenzeit zu sparen). Falls nicht, wird ein einzelner Mean Shift Schritt berechnet.

```
def process_pixel(i: int, p: np.ndarray, active: bool,
                  original_pixels: np.ndarray, bandwidth: float, epsilon: float)
```

Mean Shift Schritt

Ein einzelner Iterationsschritt im Mean Shift Algorithmus besteht darin, für jedes Pixel den neuen Schwerpunkt basierend auf den benachbarten Punkten innerhalb einer definierten Bandwidth zu berechnen. Der neue Mittelpunkt eines Pixels wird durch das gewichtete Mittel der benachbarten Punkte bestimmt, wobei die Gewichtung durch den Abstand der Pixel zueinander festgelegt wird. Dieser Abstand wird üblicherweise durch eine euklidische Distanz gemessen und mit einer Gaußschen Gewichtsfunktion skaliert. Die Berechnung erfolgt, indem die benachbarten Pixel gewichtet werden, wobei näher gelegene Pixel einen höheren Einfluss auf den neuen Mittelpunkt haben.

```
def mean_shift_step(p: np.ndarray, points: np.ndarray, bandwidth: float)
```

Weitere Hilfsfunktionen

- `color_dist(p1, p2)`
  Berechnet die euklidische Distanz zwischen zwei Farbwerten p_1 und p_2.
- `gaussian_weight(dist, bandwidth)`
  Ermittelt die Gewichtung eines Nachbarpunkts in Abhängigkeit seiner Distanz und der Bandwidth mithilfe einer Gauß-Verteilung.
- `add_point_to_clusters(...)`
  Gruppiert alle konvergierten Pixel zu Clustern und speichert die ursprünglichen Farben sowie die Schwerpunkte.
- `get_centroids(...)`
  Berechnet den finalen Schwerpunkt (Centroid) jedes Clusters aus den enthaltenen Pixeln.

Benutzereingaben

Der Benutzer muss lediglich:

- die Eingabedaten (Bilddaten als Array) und
- die gewünschte Bandwidth

bereitstellen. Der Algorithmus liefert anschließend:

- das verschobene Bild (alle Pixel im Clustermittelpunkt),
- die zugehörigen Clusterlisten und
- die errechneten Schwerpunkte der Cluster.

2 Tests und Visualisierung

Da die Berechnung des Mean Shift Algorithmus sehr rechenintensiv ist, wurden während der Entwicklung zunächst nur Testläufe mit einem kleinen 40x40 Pixel großen Bild sowie mit zweidimensionalen CSV-Daten durchgeführt.

Clustering mit 2D-Daten

Abbildung 1 zeigt das Clustering-Ergebnis der 2D-Punkte aus einer CSV-Datei. Durch die farbliche Segmentierung der einzelnen Cluster ist bereits gut erkennbar, dass das Verfahren korrekt arbeitet.

Eingabedaten:

```
10.91079038931762,8.3894120169044477
9.8750016454811416,9.9092509004598295
...
```

Ergebnis:

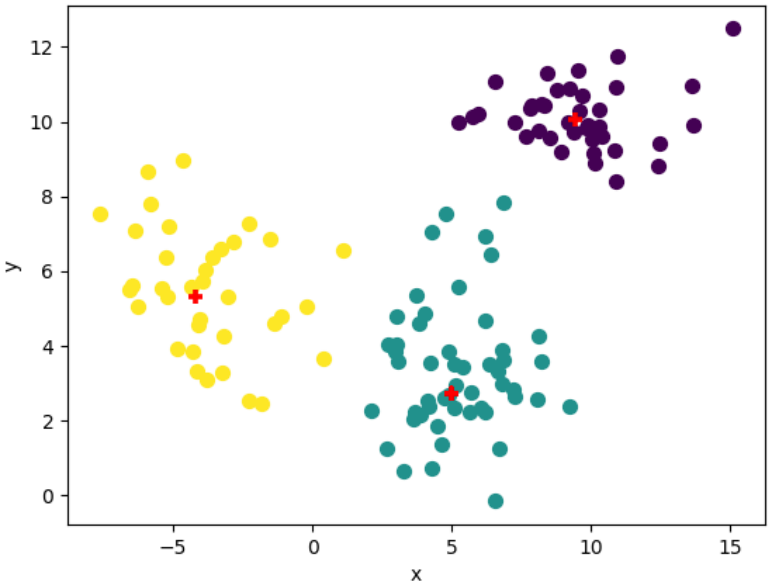

Abbildung 1: Ergebnis mit Farbsegmentierung im 2D-Bereich

Animationsbasierte Visualisierung

Für jede Iteration wird der aktuelle Zustand gespeichert, um eine animierte Darstellung des Clusterprozesses zu ermöglichen. Alle Animationen sind im Anhang als GIF-Dateien verfügbar.

Um das Verhalten des Mean Shift Algorithmus über mehrere Iterationen hinweg besser nachvollziehbar zu machen, wurden die Zustände nach jeder Iteration gespeichert und als animierte GIFs dargestellt. Diese Visualisierungen zeigen sowohl im Color Space als auch im Image Space, wie sich die einzelnen Pixel zu ihren jeweiligen Clusterzentren bewegen und wie sich die Cluster im Laufe der Berechnungen entwickeln. Alle Animationen sind im Anhang als GIF-Dateien verfügbar und bieten eine anschauliche Darstellung der Clusterbildungsprozesse.

Verwendete Testbilder

Für die Evaluierung des Mean Shift Clustering Algorithmus wurden zwei Testbilder gewählt. Beide Bilder wurden in reduzierter Auflösung verwendet, um die Rechenzeit zu verringern.

Abbildung 2: Testbild: Color Monkey

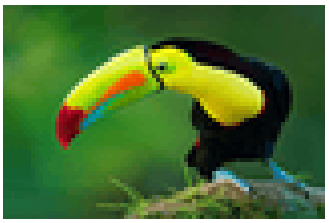

Abbildung 3: Testbild: Bird

Vergleich: Color Monkey – unterschiedliche Bandwidths

Die Wahl der Bandwidth hat einen wesentlichen Einfluss auf die Ergebnisse des Mean Shift Algorithmus. Eine zu kleine Bandwidth führt zu einer Fragmentierung der Cluster, während eine zu große Bandwidth dazu führen kann, dass mehrere Cluster zu einem einzigen zusammengefasst werden.

In den folgenden Abbildungen sind die Ergebnisse des Mean Shift Algorithmus mit unterschiedlichen Bandwidth-Werten für das *Color Monkey*-Testbild dargestellt.

Abbildung 4: Bandwidth 5.0

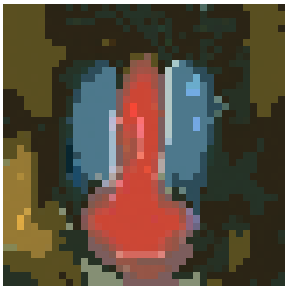

Abbildung 5: Bandwidth 10.0

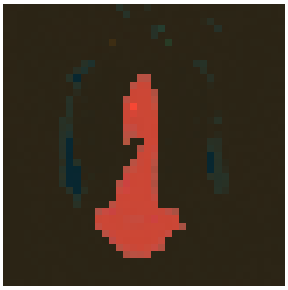

Abbildung 6: Bandwidth 20.0

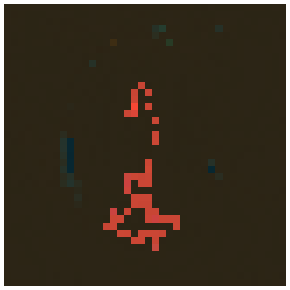

Abbildung 7: Bandwidth 30.0

Vergleich: Bird – unterschiedliche Bandwidths

Ähnlich wie beim *Color Monkey*-Testbild zeigt sich bei den *Bird*-Testbildern, dass die Clusterbildung bei unterschiedlichen Bandwidth-Werten variiert. Die folgenden Abbildungen verdeutlichen, wie der Algorithmus auf größere bzw. kleinere Bandwidth-Werte reagiert.

Abbildung 8: Bandwidth 5.0

Abbildung 9: Bandwidth 10.0

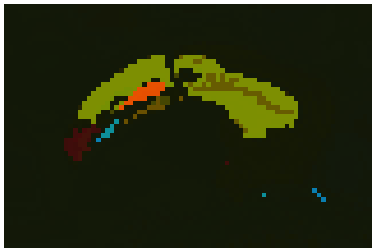

Abbildung 10: Bandwidth 40.0

Vorher-Nachher Vergleich im Farbraum

Die folgenden Abbildungen zeigen den Unterschied zwischen dem ursprünglichen Farbraum der Testbilder und dem Farbraum nach der Clusterbildung.

Color Monkey:

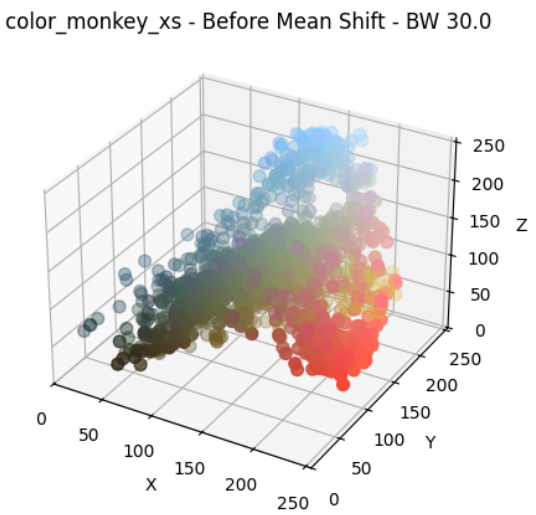

Abbildung 11: Vor Mean Shift – Bandwidth 30.0

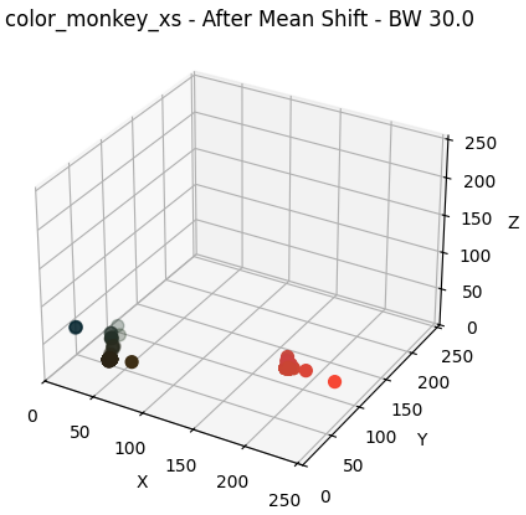

Abbildung 12: Nach Mean Shift – Bandwidth 30.0

Bird:

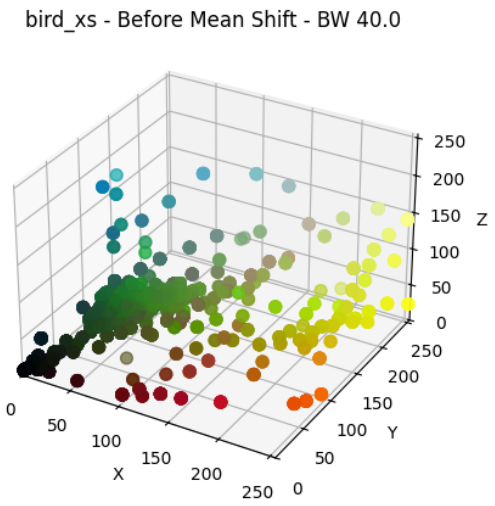

Abbildung 13: Vor Mean Shift – Bandwidth 40.0

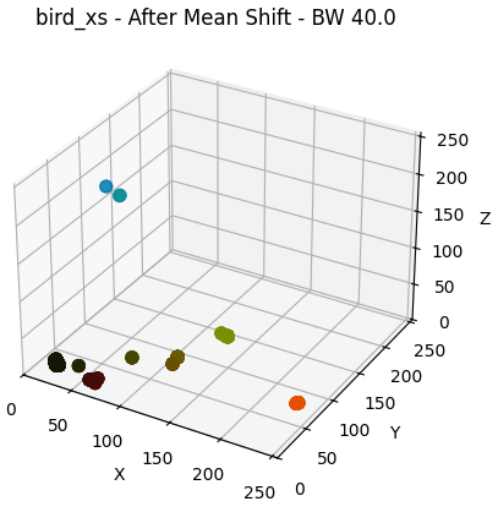

Abbildung 14: Nach Mean Shift – Bandwidth 40.0

Grenzen des Verfahrens bei geringer Bandwidth

Bei sehr kleinen Bandwidth-Werten führt der Mean Shift Algorithmus zu keinem sinnvollen Ergebnis, da die Cluster zu stark fragmentiert werden und keine sinnvolle Gruppierung entsteht. In folgenden Abbildungen sind die schlechten Ergebnisse mit einer Bandwidth von 10.0 zu sehen.

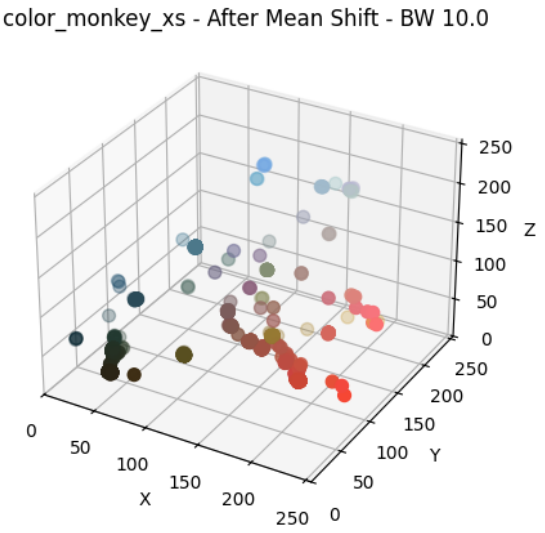

Abbildung 15: Color Monkey – Bandwidth 10.0

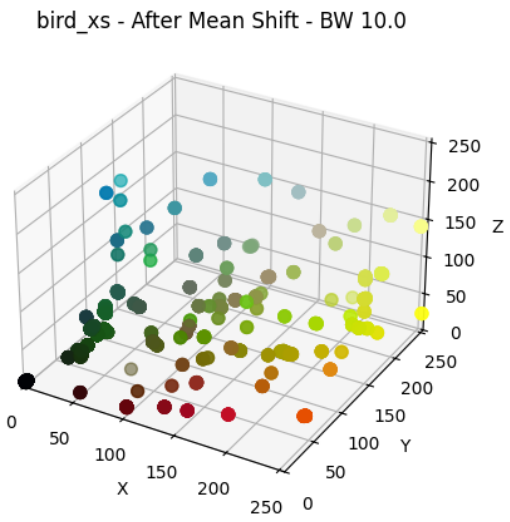

Abbildung 16: Bird – Bandwidth 10.0

Konsolenausgabe zur Laufzeit

Während des Clustering-Prozesses wird die Fortschrittsanzeige regelmäßig auf der Konsole ausgegeben.

- Anzahl der konvergierten Pixel
- Durchschnittlicher Abstand der Pixel zu ihrem Schwerpunkt (Centroid)
- Aktueller Iterationsschritt und Bildgröße

Ein Beispiel für die Konsolenausgabe:

```
Start mean shift clustering with image: color_monkey_xs.jpg
                                        bandwidth: 20.0 for 1600 pixels
Iteration 1 (Max Iteration: 2000, Image size: 40 x 40):
    Number of converged pixels: 11/1600
    Average distance of the pixels from their centroid: 21.5346
...
```