CS 415 Mini Project 3

Abhishek Basu
UIN: 658369163

Question Answering

Q1. K-means and mean shift are two common clustering algorithms that could be used for segmenting an image. What do you think are their commonalities and differences? (10 pt)

Ans: K-means and mean shift are both clustering algorithms used for tasks such as image segmentation, but they operate differently and have distinct strengths and limitations.

Commonalities

- **Clustering Purpose:** Both algorithms are used to group data points into clusters based on similarity, making them useful for segmenting an image by grouping similar pixels or features.
- **Iterative Nature:** Both are iterative algorithms, continuously refining cluster assignments or means until convergence.
- **Unsupervised:** Both methods are unsupervised, meaning they don't require labeled data or predefined classes, and instead find clusters based on the inherent structure in the data.

Differences

- **Cluster Count:**
  - K-means requires the number of clusters (k) to be specified before running the algorithm. This can be challenging in cases where the optimal number of clusters isn't known.
  - Mean shift does not need the number of clusters specified beforehand. It automatically determines the number of clusters based on data density, making it more adaptive.
- **Clustering Approach:**
  - K-means minimizes the within-cluster sum of squares, essentially finding spherical clusters around a centroid by assigning each point to the nearest cluster mean.
  - Mean shift is a density-based approach that finds clusters by iteratively shifting data points toward regions of higher density. It clusters data points that converge to the same mode, making it better suited for complex and arbitrarily shaped clusters.
- **Sensitivity to Cluster Shape and Size:**

- K-means is effective for clusters with roughly equal variance and spherical shapes but struggles with irregularly shaped clusters.
- Mean shift can handle clusters of arbitrary shapes and varying densities, as it is driven by the data's density gradient rather than assuming a specific structure.

- **Computational Complexity:**
  - K-means is generally faster and has a lower computational complexity, especially when the number of clusters is small. However, it may struggle with large datasets as k increases.
  - Mean shift can be computationally expensive, particularly with high-dimensional data, due to the need to calculate the density for each point in each iteration.

- **Application to Image Segmentation:**
  - K-means often segments an image based on color or intensity differences, suitable for cases where regions are well-separated in feature space.
  - Mean shift is better suited for images with complex textures or variable intensity regions, as it can capture more nuanced structures in the data.

Q2. Suppose you want to cluster five 1D points {0, 1, 2, 3, 4}. You have got 2 initial centroids 3.0 and 4.0 for the K-means algorithm with K=2. Please manually perform two iterations of this Kmeans algorithm. For each iteration, what you need to calculate is (1) the cluster assignment for each point, and (2) the updated centroids. (10 pt)

Ans: if we cluster five 1D points $\{0, 1, 2, 3, 4\}$. we have got 2 initial centroids 3.0 and 4.0 for the K-means algorithm with $K = 2$.

Iteration 1

Cluster Assignment

For each point, calculating the distance to each centroid and assign the point to the nearest one:

- Point 0: Distance to $C_1 = |0 - 3| = 3$, Distance to $C_2 = |0 - 4| = 4$. Assigned to Cluster 1 (closer to C_1).
- Point 1: Distance to $C_1 = |1 - 3| = 2$, Distance to $C_2 = |1 - 4| = 3$. Assigned to Cluster 1.
- Point 2: Distance to $C_1 = |2 - 3| = 1$, Distance to $C_2 = |2 - 4| = 2$. Assigned to Cluster 1.
- Point 3: Distance to $C_1 = |3 - 3| = 0$, Distance to $C_2 = |3 - 4| = 1$. Assigned to Cluster 1.
- Point 4: Distance to $C_1 = |4 - 3| = 1$, Distance to $C_2 = |4 - 4| = 0$. Assigned to Cluster 2.

After this step:

- Cluster 1 contains points $\{0, 1, 2, 3\}$.
- Cluster 2 contains point $\{4\}$.

Updating Centroids
Calculating the new centroid for each cluster by taking the mean of the points in that cluster:

- New $C_1 = \frac{0+1+2+3}{4} = \frac{6}{4} = 1.5$.
- New $C_2 = \frac{4}{1} = 4.0$.

After the first iteration, the updated centroids are:

- $C_1 = 1.5$
- $C_2 = 4.0$.

Iteration 2
Cluster Assignment
With the updated centroids, re-calculating the distances and assign each point to the nearest centroid:

- Point 0: Distance to $C_1 = |0 - 1.5| = 1.5$, Distance to $C_2 = |0 - 4| = 4$. Assigned to Cluster 1.
- Point 1: Distance to $C_1 = |1 - 1.5| = 0.5$, Distance to $C_2 = |1 - 4| = 3$. Assigned to Cluster 1.
- Point 2: Distance to $C_1 = |2 - 1.5| = 0.5$, Distance to $C_2 = |2 - 4| = 2$. Assigned to Cluster 1.
- Point 3: Distance to $C_1 = |3 - 1.5| = 1.5$, Distance to $C_2 = |3 - 4| = 1$. Assigned to Cluster 2.
- Point 4: Distance to $C_1 = |4 - 1.5| = 2.5$, Distance to $C_2 = |4 - 4| = 0$. Assigned to Cluster 2.

After this step:

- Cluster 1 contains points $\{0, 1, 2\}$.
- Cluster 2 contains points $\{3, 4\}$.

Updating Centroids
Calculating the new centroids for each cluster:

- New $C_1 = \frac{0+1+2}{3} = \frac{3}{3} = 1.0$.
- New $C_2 = \frac{3+4}{2} = \frac{7}{2} = 3.5$.

After the second iteration, the updated centroids are:

- $C_1 = 1.0$
- $C_2 = 3.5$.

Summary of Two Iterations

- Iteration 1: Cluster assignments were $\{0, 1, 2, 3\}$ in Cluster 1 and $\{4\}$ in Cluster 2. Centroids updated to $C_1 = 1.5$ and $C_2 = 4.0$.
- Iteration 2: Cluster assignments were $\{0, 1, 2\}$ in Cluster 1 and $\{3, 4\}$ in Cluster 2. Centroids updated to $C_1 = 1.0$ and $C_2 = 3.5$.

After these two iterations, the algorithm is stabilizing, with points clustering around centroids 1.0 and 3.5.

Q3. Please describe in your own words (1) the process of using histogram for color segmentation, and (2) the process of using a Gaussian model for color segmentation. Hint: both processes comprise a training stage and a testing stage. (15 pt)

Ans: (1) Histogram-Based Color Segmentation

- **Training Stage:** To use a histogram for color segmentation, we first build a histogram model of the target color based on sample images or regions containing that color. In a color space like HSV or RGB, each bin in the histogram represents a specific color range, and we populate these bins with counts of how frequently each color appears in the training data. This histogram acts as a statistical representation of the target color distribution.
- **Testing Stage:** In the testing phase, we compare the colors of each pixel in a new image to the histogram model. If a pixel's color has a high count in the histogram (indicating it's a color we want to detect), it is classified as part of the target region. By applying a threshold to the histogram values, we create a mask that highlights only the pixels that match the target color distribution, allowing us to isolate and segment the color.

(2) Gaussian Model for Color Segmentation

- **Training Stage:** Using a Gaussian model for color segmentation involves fitting a Gaussian (normal) distribution to the color values of the target region. In the training phase, we collect color samples of the target from images and calculate the mean and variance for each color channel. The Gaussian model describes how the target color values are distributed around this mean, with the variance indicating the spread or range of colors considered part of the target.
- **Testing Stage:** In the testing phase, we evaluate each pixel in the new image to see if its color is likely to belong to the target color based on the Gaussian model. For each pixel, we calculate how close its color is to the mean of the Gaussian distribution, taking into account the variance. If a pixel's color falls within an acceptable range (usually within a set number of standard deviations from the mean), it is considered part of the segmented color region. This probabilistic approach is adaptable and helps capture colors that vary slightly due to changes in lighting or shading.

Q4. A 5x5 grayscale image is given below. Compute the Hessian matrix for the 3x3 blue window (W). The image gradient can be calculated by finite difference. (15 pt)

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 1 \\ 1 & 2 & 2 & 1 & 1 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Ans : Given a 5x5 grayscale image, compute the Hessian matrix for the 3x3 window W. The image gradient can be calculated by finite difference.

Provided 5x5 Image:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 1 \\ 1 & 2 & 2 & 1 & 1 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The specified 3x3 window W is:

$$W = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

Compute the Gradients I_x and I_y To calculate the gradients I_x and I_y within this 3x3 window, we use finite differences:

- **x-Derivative (I_x):**

$$I_x = \begin{bmatrix} 0-1 & 2-0 & 0-2 \\ 2-2 & 1-2 & 2-1 \\ 1-2 & 0-1 & 2-0 \end{bmatrix} = \begin{bmatrix} -1 & 2 & -2 \\ 0 & -1 & 1 \\ -1 & -1 & 2 \end{bmatrix}$$

- **y-Derivative (I_y):**

$$I_y = \begin{bmatrix} 2-1 & 2-0 & 1-2 \\ 1-0 & 1-2 & 0-1 \\ 2-2 & 1-2 & 0-1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 \\ 1 & -1 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

Compute the Components of the Hessian Matrix H The Hessian matrix H is given by:

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix}$$

calculating each component by summing over all elements in W:

1. Calculating $\sum I_x(x,y)^2$:

$$\sum I_x(x,y)^2 = (-1)^2 + 2^2 + (-2)^2 + 0^2 + (-1)^2 + 1^2 + (-1)^2 + (-1)^2 + 2^2 = 10$$

2. Calculating $\sum I_y(x,y)^2$:

$$\sum I_y(x,y)^2 = 1^2 + 2^2 + (-1)^2 + 1^2 + (-1)^2 + (-1)^2 + 0^2 + (-1)^2 + (-1)^2 = 10$$

3. Calculating $\sum I_x(x,y)I_y(x,y)$:

$$\sum I_x(x,y)I_y(x,y) = (-1)(1)+(2)(2)+(-2)(-1)+(0)(1)+(-1)(-1)+(1)(-1)+(-1)(0)+(-1)(-1)+(2)(-1) = 7$$

Construct the Hessian Matrix H Using the calculated values, we can construct the Hessian matrix H:

$$H = \begin{bmatrix} \sum I_x(x,y)^2 & \sum I_x(x,y)I_y(x,y) \\ \sum I_x(x,y)I_y(x,y) & \sum I_y(x,y)^2 \end{bmatrix} = \begin{bmatrix} 10 & 7 \\ 7 & 10 \end{bmatrix}$$

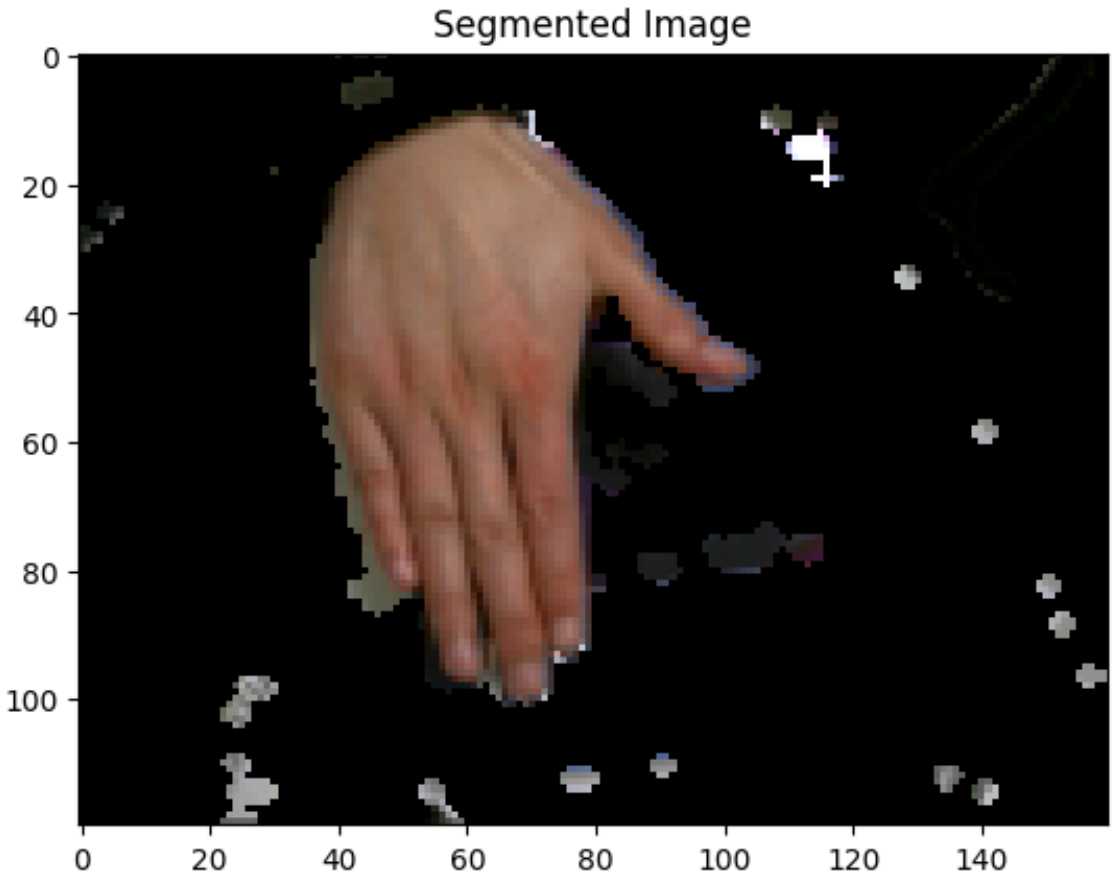
Final Answer: The Hessian matrix H for the 3x3 window W is:

$$H = \begin{bmatrix} 10 & 7 \\ 7 & 10 \end{bmatrix}$$

Programming

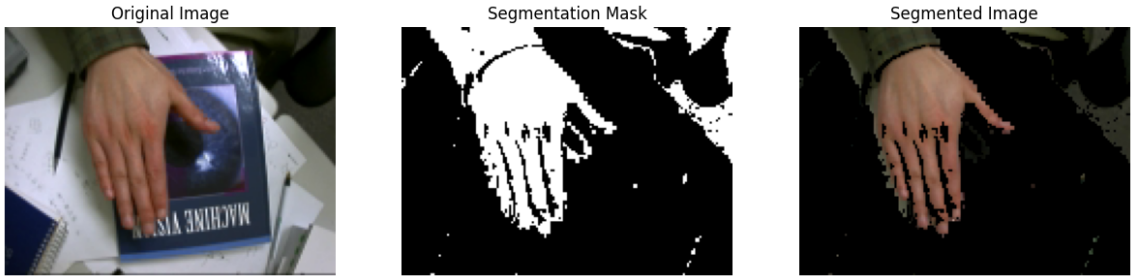
P1. In our code tutorial for histogram-based color segmentation, the result is not very satisfactory because there is only one training image and it does not capture all the color variations of skins well. You can follow the steps below to improve the performance.

- Collect at least 10 different skin patches from the Internet or your own photo gallery. Collect training data from the testing image is prohibited. (10 pt)
- Modify the code or build your own code from scratch to train a new model (that is, to calculate the HS histogram) on all these patches. Use this new model to segment testing_image.bmp to obtain improved results. (10 pt)



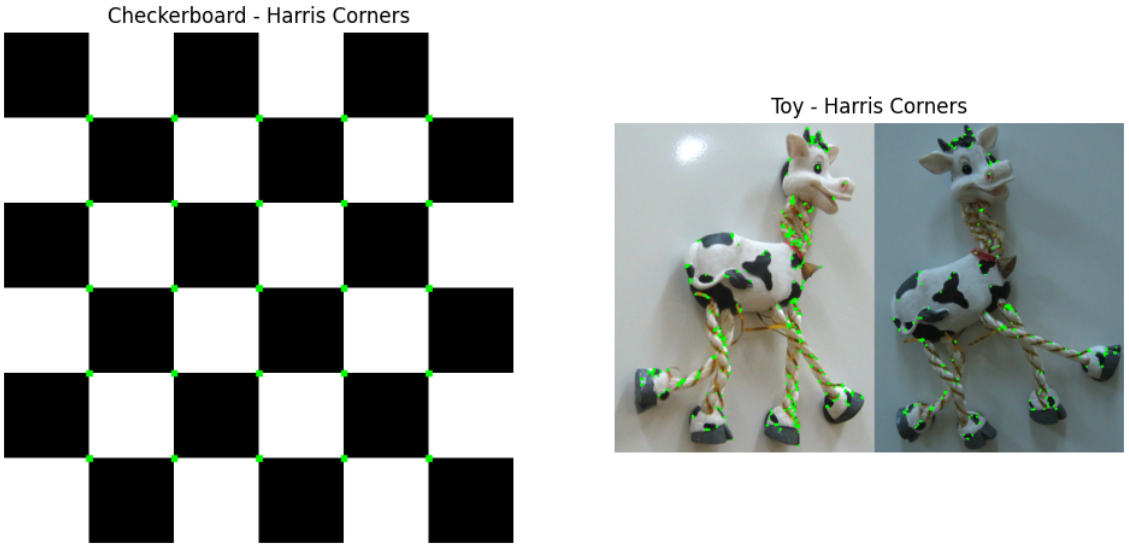
P2. In addition to building a skin color model based on the histogram, an alternative approach is to model the skin color via a 2D Gaussian distribution, with mean μ and covariance matrix $\sum$ $d = 2$ is the dimension of the random vector.

- Use the training data collected in P1 to build a Gaussian-based skin color model. What you need to do is to estimate the mean vector and the covariance matrix of the 2D Gaussian distribution. Hint: the Gaussian distribution models continuous random variables; thus color space quantization is not required. (10 pt)
- Calculate the skin probability for each pixel in testing_image.bmp based on the estimated Gaussian distribution and set a threshold to perform segmentation. (15 pt)



P3. Self-study the Harris corner detector implemented in OpenCV and test it on checkerboard.png and toy.png.

(5 pt)



Submission

Please follow the instructions below for submission.

- You need to upload two files to Blackboard: a PDF file and a .py file. Do not compress them into a single ZIP file.
- The PDF file contains all your solutions to this homework. For Question Answering, you can either type answers or handwrite them and take a photo. For Programming, you need to include the output of the program such as a processed image.
- The .py file contains all your code for the programming problems.