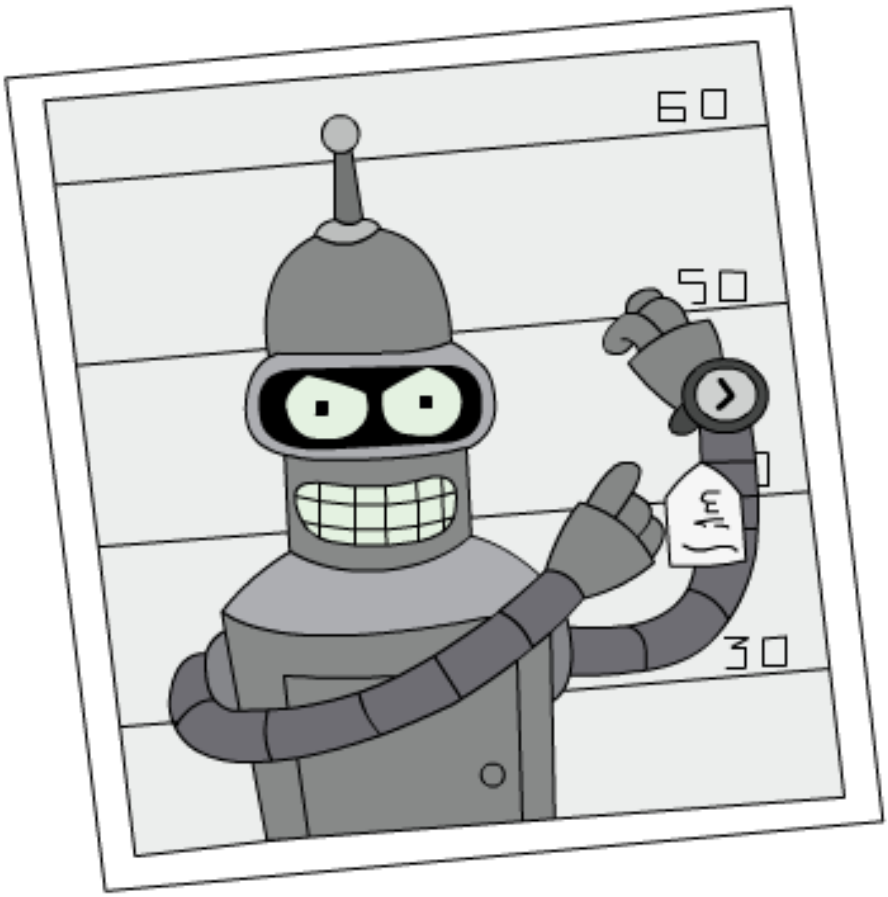
k-Means and Mean-Shift Clustering

ECE4076/5176 Computer Vision
Lab 3 (Weeks 8,9)



Performance Expectations and Guidelines for Lab Completion

As we progress through the course, we expect you to expand your skills and toolset to tackle more complex and involved problems. This lab has been designed to help you achieve that by providing a comprehensive and engaging learning experience. To ensure that you achieve your best possible results, we recommend that you begin working on the lab well in advance of your lab sessions. Additionally, please note that the computation times required for this lab may vary depending on the efficiency of your code. Sometimes, running your code may take longer than 15 minutes if it is not optimized efficiently. Therefore, we encourage you to put in your best efforts and work diligently to achieve the desired results.

Academic integrity

Every lab submission will be screened for any collusion and/or plagiarism. Breaches of academic integrity will be investigated thoroughly and may result in a zero for the assessment along with interviews with the plagiarism officers at Monash University.

Late submissions

Late submission of the lab will incur a penalty of 10% for each day late. That is, with one day delay, the maximum mark you can get from the assignment is 7.2 out of 8, so if you score 7.5, we will (sadly) give you 7.2. Labs submitted with more than a week delay will not be assessed. Please apply for special consideration for late submission as soon as possible (*e.g.*, documented serious illness).

Lab Instructions and the Use of Generative AI

- You may **not** use any built-in OpenCV functions for this lab, except for those used for loading/saving or processing (colour, size, ...) an image.
- You may use NumPy for array handling, and vectorizing your code (reducing the number of for-loops) is encouraged.
- You should use Matplotlib to display images and any intermediate results.
- When completing the mean-shift algorithm task in this lab, you may use the MeanShift() function from the *scikit-learn* (aka. *sklearn*) library.
- You may use generative AI unless explicitly stated

Lab Grading

Each lab is worth 8%, and there are a number of sections and tasks with their own weighting. A task is only considered complete if you can demonstrate a working program and show an understanding of the underlying concepts. Note that later tasks should reuse code from earlier tasks.

In this laboratory exercise, you will learn and employ clustering techniques to segment and encode an RGB image with k number of colours. In all the tasks below, you must **not** use the pre-written Python libraries for K-means Clustering; instead, you build your own K-means code.

- Task 1: Distance Function
- Task 2: k-Means Clustering
- Task 3: k-Means++ Initialization
- Task 4: GMM: Analysing Data Distributions
- Task 5: Mean Shift - Data Modelling via its distribution
- Task 6: Comparison between k-Means, k-Means++ Initialization and Mean-Shift

- Discussion questions: These are at the end of each task within the notebook, and require you to answer them when submitting your python notebook.

References

- k-means clustering
- k-means++
- Mean shift
- Week 7 Lecture content

Resources

- Lab3_student_template.ipynb - please complete the lab tasks in this template notebook and use the markdown spaces provided to report your findings/ describe your approach.
- sharon.jpg is located in the lab3 folder
- Example outputs in the *img* folder that links to the notebook

Figure 1: Provided sharon image

Task 1: Distance Function

In this task, you will build a helper function to compute the squared distance from a set of data points to a set of centroids. You are given a small dataset of five data points, X and two centroids, M to test your function.

$$X = \begin{bmatrix} 0.67187976 & 0.44254368 & 0.17900127 \\ 0.55085456 & 0.65891464 & 0.18370379 \\ 0.79861987 & 0.3439561 & 0.68334744 \\ 0.36695437 & 0.15391793 & 0.81100023 \\ 0.22898267 & 0.58062367 & 0.5637733 \end{bmatrix}$$

$$M = \begin{bmatrix} 0.66441854 & 0.08332493 & 0.54049661 \\ 0.05491067 & 0.94606233 & 0.29515262 \end{bmatrix}$$

You will also visualize whether your distances are correct by assigning the data points to the closest centroid and displaying the results on a 3D scatter plot.
You should not use generative AI to complete the distance function.

Task 2: k-Means Clustering and Visualisation

In this task, you will implement the k-Means Clustering and visualize the movement of the centroids. First, you will write a function to generate k number of random centroids as initialization. Then you will write a function that uses these centroids to perform k-Means Clustering using the below steps:

1. Go through all the pixels in the image and calculate which of the k centroids it is closest to.
2. For each pixel, store the index of the nearest centroids at the same pixel location in an index image.
3. Re-compute each centroid by going through all the pixels in the RGB colour image that were assigned to that centroid and taking the mean.
4. Repeat steps 1-3 for a pre-determined number of iterations.

Load the **sharon.jpg** image and use your function to cluster the pixels of the image into four clusters ($k = 4$). Finally, you will display the results.
Run the program several times with different random seeds to see if you always converge on the same answer. Try changing k from 4 to other numbers (from 2 to 10) and see how this affects the output and if the results are reproducible.

After completing the k-means implementation, write code to visualize the clustering process at every iteration. For that, first, you will write code to plot a 3D scatter plot of data points along with the centroids and display the clustered image at once. Then you will use that function inside your k-Means Clustering implementation to observe how the centroids move with each iteration. Draw the RGB colour values from the **sharon.jpg** image as well as the centroid RGB colour values as a 3D scatter plot. The colour of each data point should reflect the centroid colour value that it is closest to.

Plotting all the data points is computationally expensive, so opt for only drawing randomly selected 250 data points. Also, make sure you add a time pause of 1-2 seconds after each iteration so there is enough time for you to observe the convergence of centroids clearly. Have a look at the template notebook for an example output of what it should look like.

Task 3: k-Means++ Initialization

In this task, you will implement a variant of the k-Means Clustering algorithm called the k-Means++ algorithm. In the k-Means++ algorithm, only the initialization of the centroids will change. The rest of the implementation is the same as the conventional K-Means Clustering with random initialization. Therefore, first, you will write a function to generate centroids using the k-Means++ initialization by following the below steps:

1. Choose the first centroid randomly from the data points.
2. Compute the squared distance d^2 between every data point x and the selected centroids so far. Record the minimum value of d^2 as $\mathcal{D}(x)$ for data point x.
3. Compute the probability of selecting each datapoint as the next centroid as:

$$\mathcal{P}(x) = \frac{\mathcal{D}(x)}{\sum_x \mathcal{D}(x)} \tag{1}$$

4. Randomly draw a new centroid based on the computed probability distribution. HINT: Compute an array that represents the cumulative distribution from the probability values using the **cumsum** function. Generate a random number r between 0 and 1. The new centroid corresponds to the first element value in the cumulative distribution array that is greater than the value r.
5. Repeat steps 2-4 until k number of centroids have been chosen.

After having chosen k number of centroids, proceed with the standard k-Means algorithm using the selected centroids. Load the **sharon.jpg** image and use your k-Means++ implementation to cluster the pixels of the image into four clusters ($k = 4$). Finally, you will display the results.

Plotting all the data points is computationally expensive, so you may only draw randomly selected 250 data points. Also, make sure you add a time pause of 1-2 seconds after each iteration so there is enough time for you to observe the convergence of centroids clearly.

Task 4: GMM: Analysing Data Distributions

In this task, you will analyse a toy data set and understand its distribution via Gaussian Mixture Models (GMMs). GMMs are probabilistic models that represent the probability distribution of data as a weighted sum of multiple Gaussian distributions. Each Gaussian distribution in the mixture is referred to as a component and is characterized by its mean, covariance, and weight. GMMs are commonly used for tasks such as clustering and density estimation, and can be applied to generate new samples, assign data points to components, and estimate probability densities in the feature space.
A multivariate Gaussian distribution is defined as:

$$\mathcal{N}\big(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\big) = \frac{1}{\sqrt{(2\pi)^n \det\big(\Sigma\big)}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big)\,. \tag{2}$$

And the probability distribution function (PDF) of a GMM with k components is

$$f(x) = \sum_{i=1}^{k} \omega_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{3}$$

For this task, you will:

1. Generate a toy dataset with different means, standard deviations, and weights
2. Plot the PDF of the generated dataset
3. Estimate a PDF from the data

Task 5: Mean Shift - Data Modelling via its distribution

In this task, you will apply the mean shift algorithm to iteratively identify clusters. The mean shift clustering algorithm is a density-based technique that identifies clusters by iteratively shifting data points towards regions of higher density, without specifying the number of clusters in advance. It is particularly effective for handling complex and irregularly shaped clusters, and is commonly used in image segmentation, object tracking, and other applications.
Furthermore, you will also implement two kernels[1] (flat and Epanechnikov) which will be used for the mean-shift algorithm and do a quick comparison between them. The two kernels have the following forms.

Flat kernel:

$$k_{\mathrm{f}}(x,z) = \begin{cases} 1 & \text{if } \|x-z\|^2 \le h^2 \\ 0 & \text{otherwise} \end{cases}\,. \tag{4}$$

Epanechnikov kernel:

$$k_{\mathrm{e}}(x,z) = \begin{cases} \frac{3}{4}\Big(1 - \frac{\|x-z\|^2}{h^2}\Big) & \text{if } \|x-z\|^2 \le h^2\,, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In both cases, h is the bandwidth of the kernel and x and z are two data samples (*e.g.*, pixels of an image).

To apply the mean-shift (MS) algorithm, you will

1. Get an appreciation of the flat kernel density function by applying it to the toy dataset from the previous task,

[1] Kernels are used to smooth out probability density functions (PDFs)

2. Move onto the Epanechnikov-kernel function (a popular kernel choice for the MS algorithm),

3. Apply MS to the **sharon.jpg** and analyse how many colours arise from the image.

4. Compare the MS results to the ones obtained via k-means clustering.

Task 6: Comparison between k-Means, k-Means++ Initialization and Mean-Shift

In this task, you will visualize k-Means, k-Means++ initialization, and the mean-shift algorithm by plotting the final clustering results of each method together with their final cluster centers. For the two k-Means methods, you will additionally visualise their initial centroids. Discuss the results and any noticeable differences that you may have encountered between the different algorithms.

Discussion Questions

These are discussion questions that need to be answered when submitting your python notebook.