CS5487 Programming Assignment 2

NIE Wan 57120844

October 2022

Problem 1 Clustering synthetic data

(a) Implementation of algorithms

Implementation of the three clustering algorithms can be found at the utils.py file in the source code, which was also uploaded on Canvas. According to Prof. Antoni Chan's tips (Lecture 4 Resources/gmm-tips.pdf), there are some implementation details to note:

- *Robustness*: Random initialization is not very robust for K-means and EM-GMM because it could start near a bad local minimum. Here, we prepared two alternative solutions. a) Adopt the *Furthest first* initialization strategy to select initial centers that are furthest apart from each other. b) Run several trials of K-means with different random initializations, and then select the trial that results in the smallest total distance to the centers. For EM-GMM, select the trial with the largest data log-likelihood, i.e., $\log P_{\boldsymbol{\theta}}(X)$. In this project, we adopt the second solution.
- *Regularization*: For EM-GMM, regularizing the covariance matrices Σ can help to prevent overfitting and prevent singularities. Thus, we added a constant to the diagonal, i.e. $\Sigma \leftarrow \Sigma + \alpha I$, where α is a regularization parameter.
- *Convergence*: For EM-GMM, there are several ways to test for convergence of EM, and we adopted the strategy of checking the percent change in data log-likelihood, $\log P_{\boldsymbol{\theta}}(X)$, is below a threshold.

(b) Running result of the three synthetic datasets

In this section, we tested the performance of the three clustering algorithms on three synthetic datasets, and the results are given in Figure 1. Clustering centers/modes are shown by crosses.

(b1) Qualitative performance analysis

Observing the experiment results, we could find that the performance of K-means and EM-GMM is similar, that they both can correctly cluster dataA and dataB into 4 classes while their performance on dataC is poor. In contrast to K-means and EM-GMM, mean-shift performs well not only on dataA/dataB but also on dataC.

(b2) Comment on the algorithms

Compared with dataA and dataB, dataC has a more complicated circle-like geometric shape. When calculating the distance between data points and clustering centers, the Euclidean distance cannot "bend" to suit the data configuration. Similar to K-means, the Gaussian distribution of EM-GMM can not "bend" neither to suit the circle-like data configuration. Compared with K-means and EM-GMM, mean-shift has several *advantages*:

- It can model complex clusters with a nonconvex shape, e.g. the circle-like data configuration in this problem.
- Mean-shift is a non-parametric clustering method and does not need to make any model assumption as like in K-means or EM-GMM.

Meanwhile, mean-shift also suffers from *limitations* that people cannot directly control the number of clusters and should search for a good bandwidth parameter h, which is illustrated in the following 1(c).

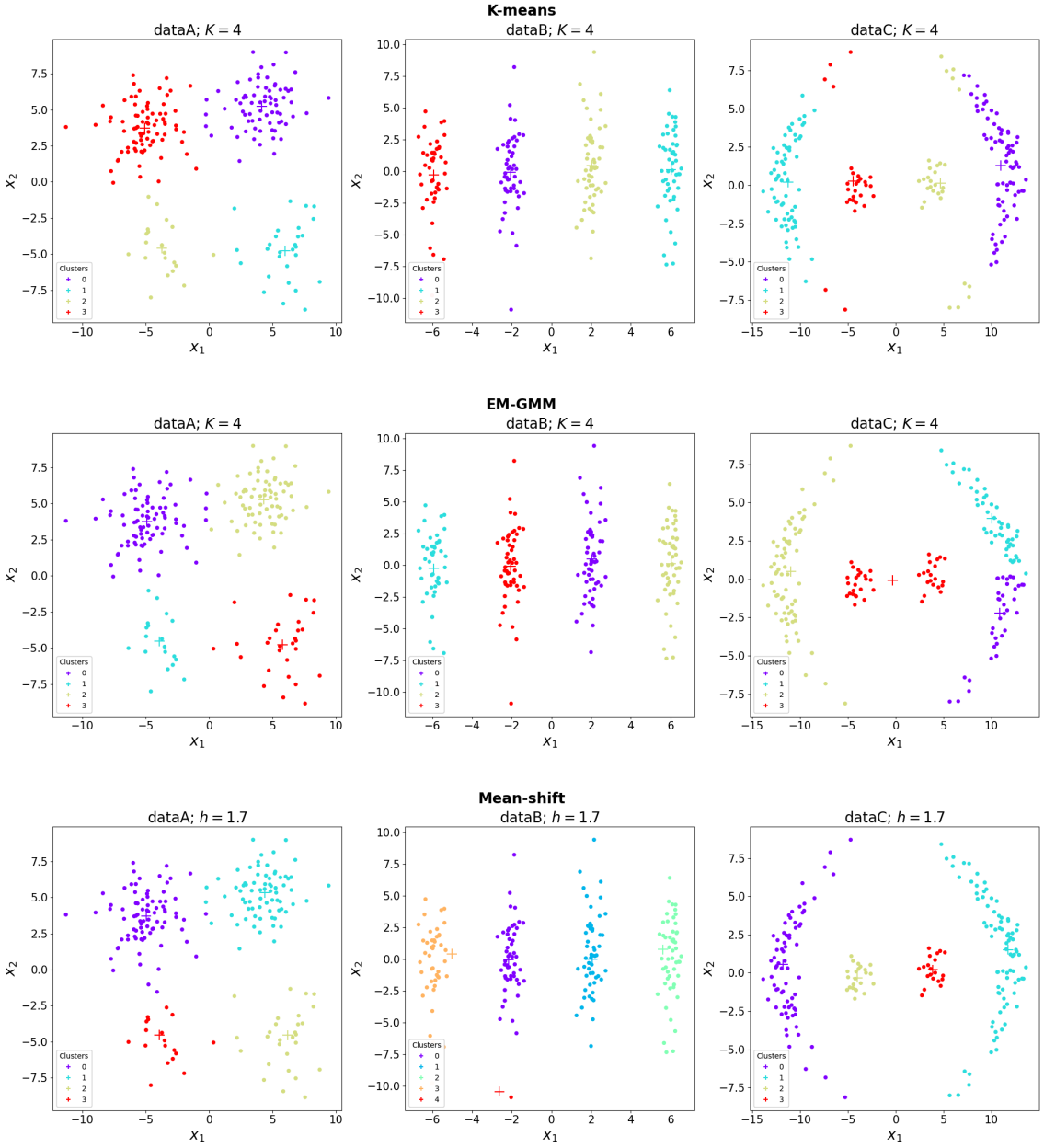


Figure 1: Running result of the three synthetic datasets.

(c) Sensitivity to the bandwidth parameter h

In this section, we tested the sensitivity of mean-shift to the bandwidth parameter h, and the results are given in Figure 2. Clustering modes are shown by crosses. Observing the experiment results, we could find that similar trends exist in dataA, dataB and dataC. When the bandwidth h increases, the number of clustering centers decreases. In conclusion, to get a better clustering result of mean-shift, we must search for a good h, which is a limitation of mean-shift.

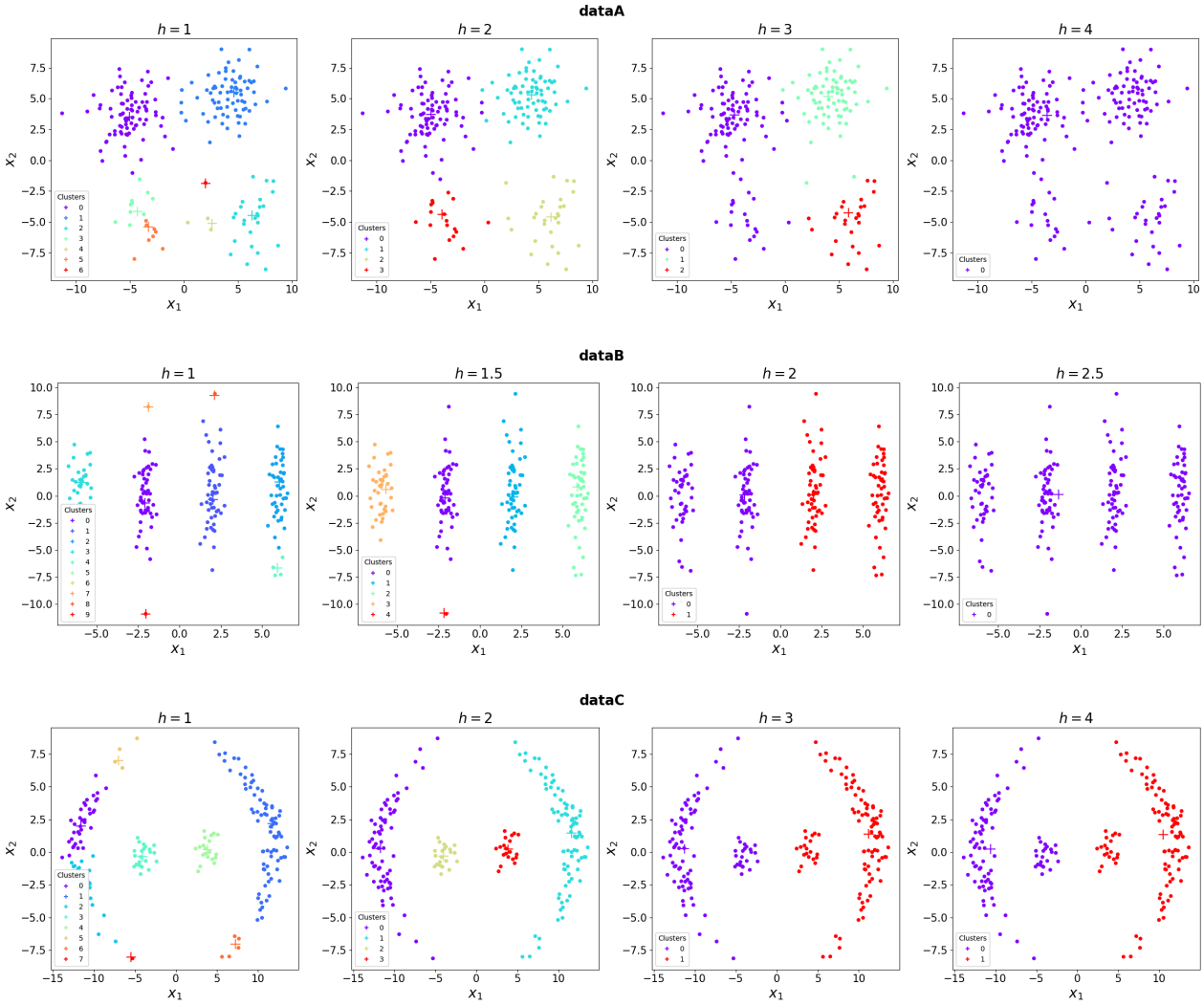


Figure 2: Sensitivity to the bandwidth parameter h.

Problem 2 A real world clustering problem – image segmentation

(a) Segmentation Examples

In this section, we evaluated the three algorithms on 12003.jpg and 299086.jpg. It is worth mentioning that for the input $x = [u, v, c_y, c_x]^T$, the pixel location features (c_y, c_x) has a larger scale than that of the chrominance features (u, v). Therefore, when we run the algorithms without scaling the feature dimensions, the clustering tends to rely more on pixel locations. Take the result of K-means on 12003.jpg as an example (Figure 3). The image is clustered into K segments which are spatially far away from each other. Problem 2(b) gives a solution to this different scales problem, but in 2(a), we scaled the input dimension by $X \leftarrow \frac{X-\mu}{\sigma}$, where μ and σ is the mean and standard deviation of the feature values.

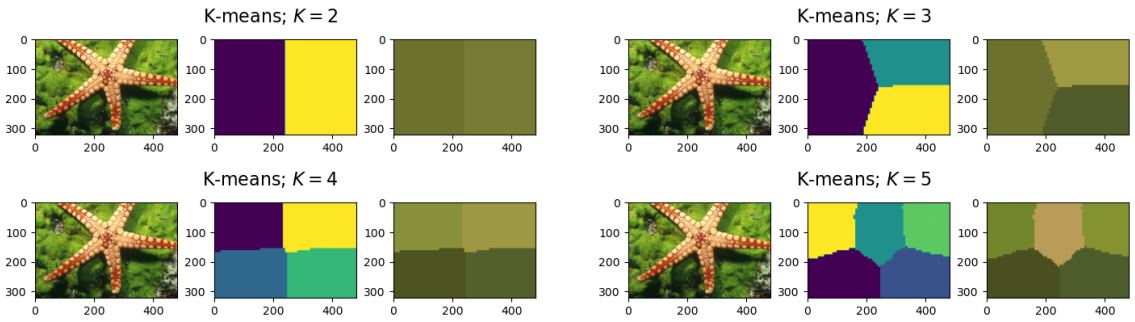
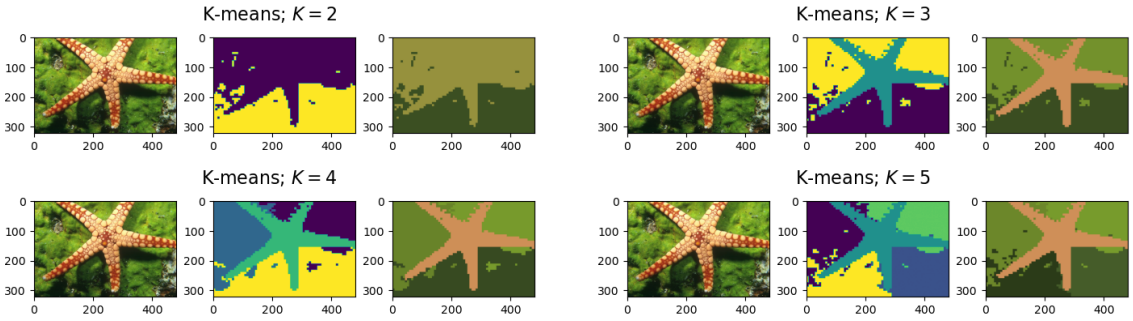


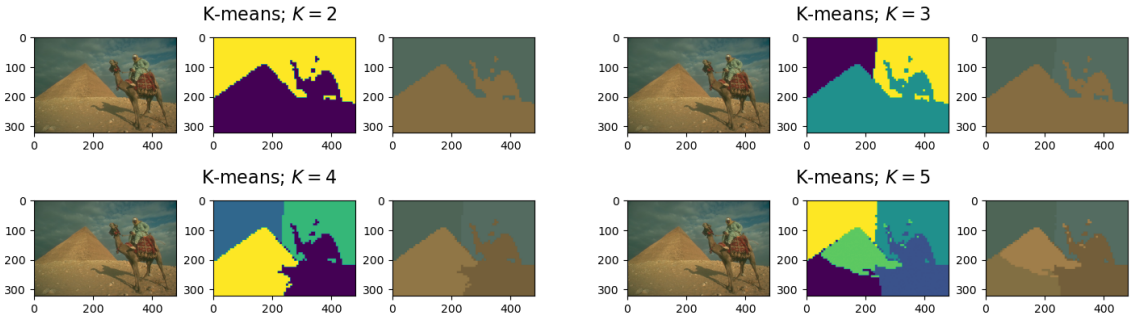
Figure 3: K-means performance on the unstandardized input. Take 12003.jpg as an example.

By analyzing Figures 4 to 6, several observations can be made:

- Qualitatively, by adjusting the bandwidth h, mean-shift could reach a better performance than the other two algorithms.
- For K-means and EM-GMM, the image can be clustered into more segments when K is increasing. While for EM-GMM, the increase of h could result in fewer segments.
- Mean-shift is the most sensible algorithm to parameter change when considering algorithms' sensitivities to parameters. When we change K, the segmentation can still represent the original pattern in the image. However, the segmentation can be totally messy when h is very small or large.
- In contrast to K-means and EM-GMM, mean-shift does not rely on the assumption of the cluster number, but we need to search for a good bandwidth parameter.
- K-means and EM-GMM are faster than mean-shift. Although their performance is related to the initialization, we can run many trials and select the most reasonable one.

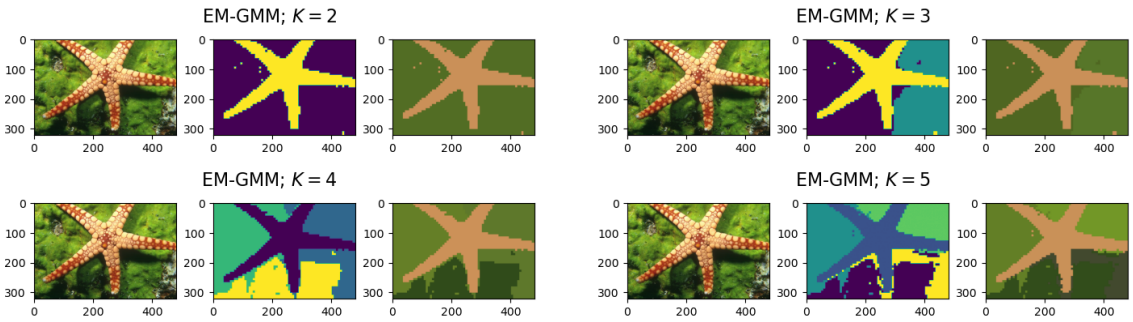


(a) 12003.jpg

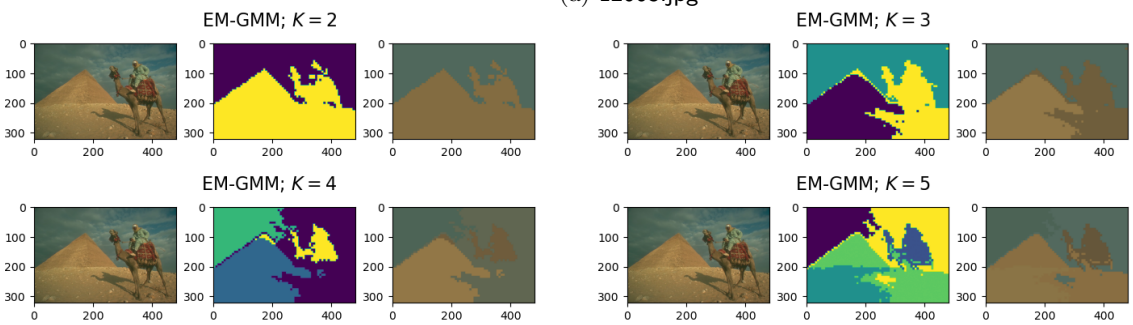


(b) 299086.jpg

Figure 4: K-means performance on 12003.jpg and 299086.jpg

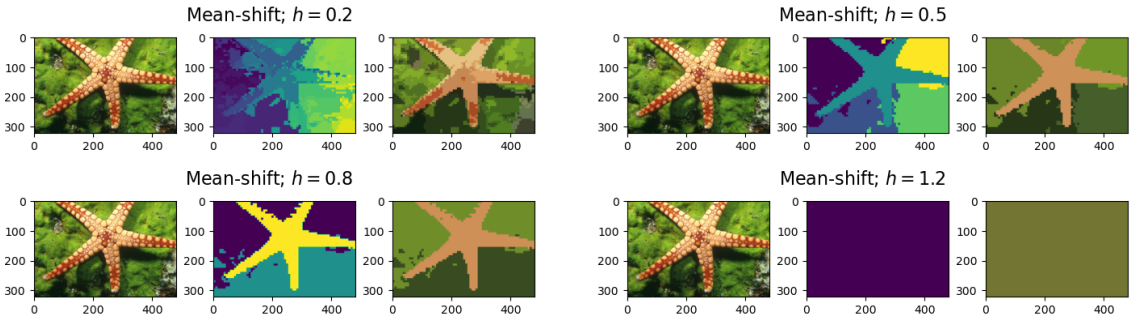


(a) 12003.jpg

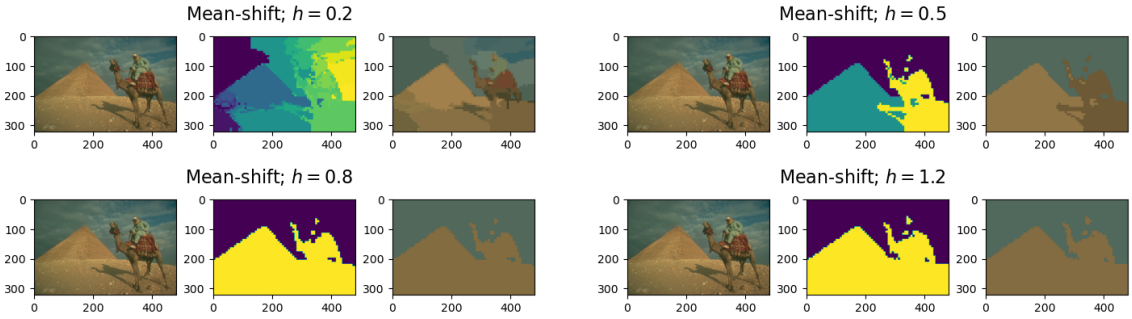


(b) 299086.jpg

Figure 5: EM-GMM performance on 12003.jpg and 299086.jpg



(a) 12003.jpg



(b) 299086.jpg

Figure 6: Mean-shift performance on 12003.jpg and 299086.jpg

(b) Different feature scaling

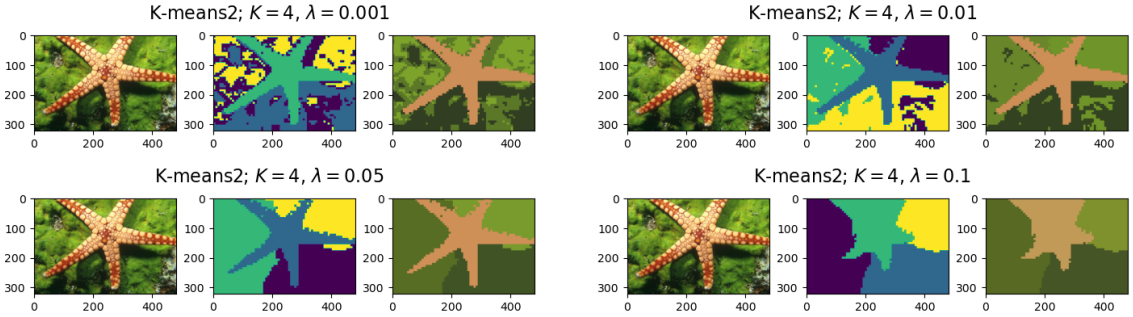
In this section, we scaled the feature values by Equations (1) and (2) rather than using the standardization in 2(a).

$$d(x, x') = \left\| \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u' \\ v' \end{bmatrix} \right\|^2 + \lambda \left\| \begin{bmatrix} c_x \\ c_y \end{bmatrix} - \begin{bmatrix} c'_x \\ c'_y \end{bmatrix} \right\|^2 , \tag{1}$$

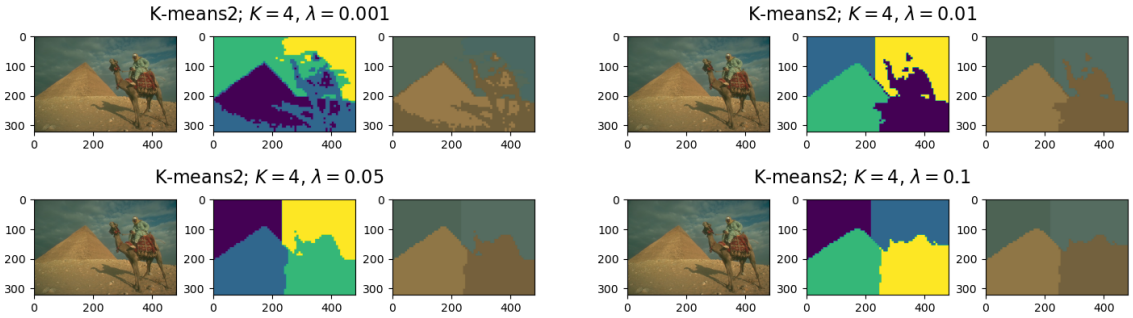
$$k(x, x') = \frac{1}{(2\pi)^2 h_p^2 h_c^2} \exp\{-\frac{1}{2h_c^2} \left\| \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u' \\ v' \end{bmatrix} \right\|^2 - \frac{1}{2h_p^2} \left\| \begin{bmatrix} c_x \\ c_y \end{bmatrix} - \begin{bmatrix} c'_x \\ c'_y \end{bmatrix} \right\|^2 \} . \tag{2}$$

As shown in Figure 7, the weights of pixel locations increase when we increase λ, and as a result, points close in 2D locations are grouped together despite their totally different colors (e.g., $\lambda = 0.1$). When the weights of chrominance values increase (decrease λ), points with similar colors tend to be clustered into the same group (e.g., $\lambda = 0.001$).

For the mean-shift algorithm (Figure 8), a similar trend can be found. When we increase h_c/h_p, e.g., $h_c/h_p = 1.0$, the weights of pixel locations increase, and the algorithm tends to group the nearby data points into the same group. But for small h_c/h_p, e.g., $h_c/h_p = 0.05$, it tends to group the data points with similar colors (ignoring the location effects).

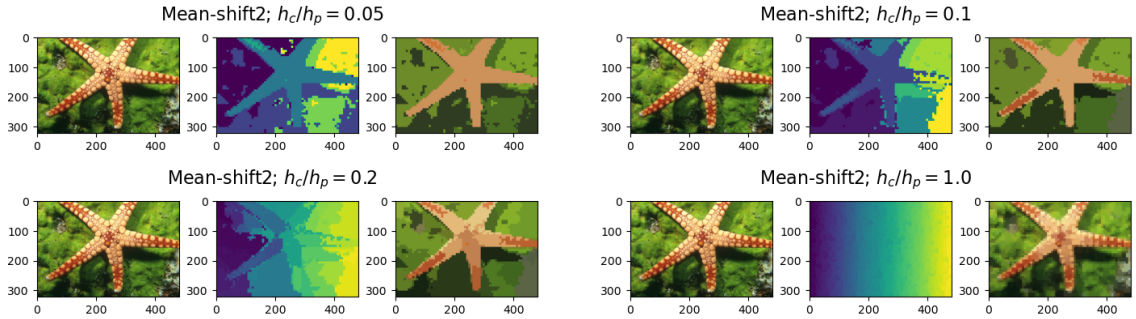


(a) 12003.jpg

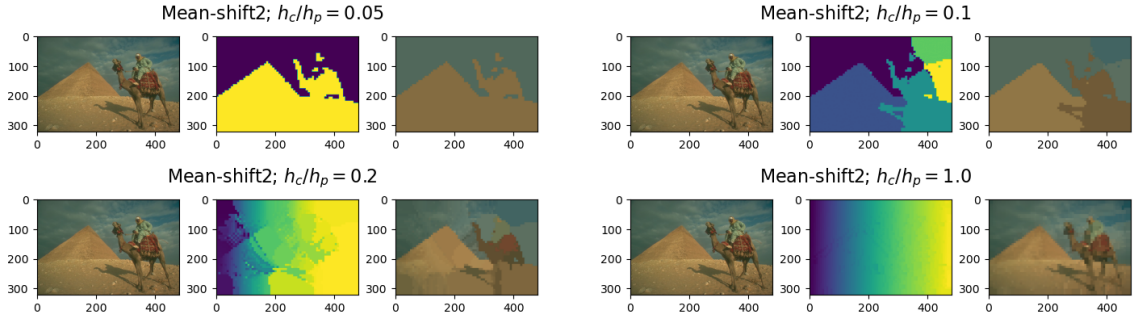


(b) 299086.jpg

Figure 7: K-means performance on 12003.jpg and 299086.jpg with different λ.



(a) 12003.jpg



(b) 299086.jpg

Figure 8: Mean-shift performance on 12003.jpg and 299086.jpg with different h_c/h_p. h_c is set to 4.