Computer Vision

Assignment 1 - Image Segmentation

Parmenion Koutsogeorgos - i6328191

Contents

1 Introduction **1**

2 Meanshift algorithm **1**

3 Image segmentation with the meanshift algorithm **3**
3.1 Pre-processing 3
3.2 Applying the meanshift algorithm on an example image 3
3.3 Segmented image reconstruction 4

4 Experiments **5**
4.1 Results & discussion 6
4.1.1 3D feature vectors 6
4.1.2 5D feature vectors 11

1 Introduction

In this report we present our experiments and results of using the meanshift clustering algorithm in order to perform image segmentation. We first test our algorithm on some data sampled from two distinct Gaussian distributions, and then present our findings during the segmentation of 3 images taken from the Berkeley Segmentation Dataset. A demonstration of the experimentation process can be found in the notebook `imageSegmentation_demo.ipynb`.

2 Meanshift algorithm

We first test our implementation of the meashift algorithm on a dataset consisting of 2000 3D points sampled from two Gaussians with distinct means. The expected result consists of 2 clusters, with the peak of each being approximately equal to the mean of each Gaussian.

Non-optimized. We first test the non-optimized version of meanshift. Our parameters are chosen as follows:

- $r = 2$;

- $THRESHOLD = 0.01$.

For the non-optimized version of the meanshift algorithm we get 2 labels $[0, 1]$ with associated peaks

$$[-0.06756875,\ 0.04788814,\ 0.02778451]$$

and

$$[5.04987186,\ 4.98116882,\ 5.0104074]$$

respectively. The process terminates in 5.58s. The clusters obtained can be seen in Figure 1.

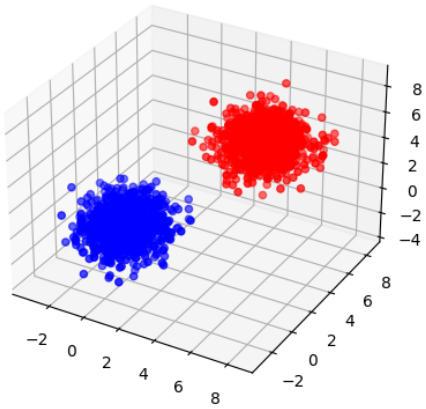


Figure 1: Results of the non-optimized meanshift algorithm applied on the test data.

Optimized. We now test the optimized version of the meanshift algorithm on the same data. We choose the following parameters:

- $r = 2$;
- $c = 4$;
- $THRESHOLD = 0.01$.

Once again, we get 2 labels $[0, 1]$ with associated peaks

$$[-0.06756875,\ 0.04788814,\ 0.02778451]$$

and

$$[5.04987186,\ 4.98116882,\ 5.0104074]$$

respectively. The process terminates in 1.53s, showing that the speedups have a significant effect. Below we see the same two clusters obtained from the algorithm.

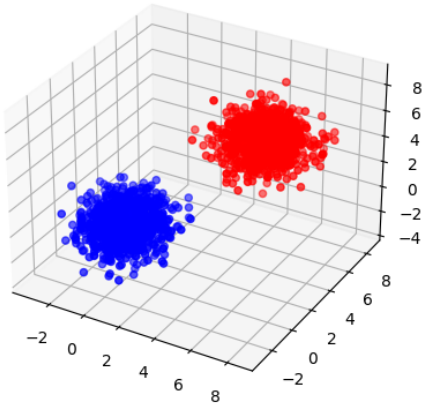


Figure 2: Results of the optimized meanshift algorithm applied on the test data.

3 Image segmentation with the meanshift algorithm

We use the optimized version of the meanshift algorithm to perform image segmentation through clustering.

3.1 Pre-processing

The images are loaded in BGR color format and then converted into CIELAB color format. The resulting arrays are not normalised, and no further pre-processing is applied on the images.

Given an image of shape $(height,\ width,\ 3)$, we construct the following data structures to be given as input to the meanshift algorithm:

- an array of shape $(height \times width,\ 3)$ which is just a flattened version of the original array, with each row containing only the 3 CIELAB channel values;
- an array of shape $(height \times width,\ 5)$, each row of which consists of the 3 CIELAB channel values along with the 2 coordinates (x, y) defining the position of the corresponding pixel.

3.2 Applying the meanshift algorithm on an example image

In order to speed up execution time for image segmentation we raise the $THRESHOLD$ parameter to 1. This is a reasonable choice, given that we do not normalize the CIELAB channels or the (x, y) values, and therefore a shift of the mean by distance less than 1 induces minimal change in the resulting cluster.

We apply the meanshift algorithm using 3D and 5D feature vectors on the following image of the earth of size 150×154.

Figure 3: Image used for testing of the meanshift algorithm.

We use the following parameters for both 3D and 5D input:

- $r = 20$;
- $c = 4$;
- $THRESHOLD = 1$.

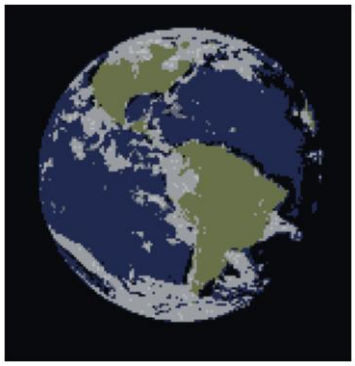
In the 3D case the algorithm terminates after 2.10s and finds 6 clusters, whereas in the 5D case it terminates after 73.25s and finds 68 clusters.

3.3 Segmented image reconstruction

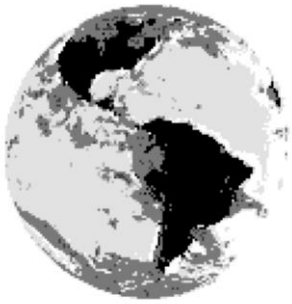
To visualise the results of the segmentation we use two methods.

- `segm_image_peaks` is a method that colors each pixel by the CIELAB (converted to RGB) values of its associated peak.
- `segm_image_clusters` is a method that assigns a different grayscale color value to each cluster label and then colors each pixel by the color associated to its label.

Notice that in the case of a 3D feature vector input, `segm_image_clusters` simply produces a grayscale version of the output of `segm_image_peaks`, since each cluster is identified by a unique color value.



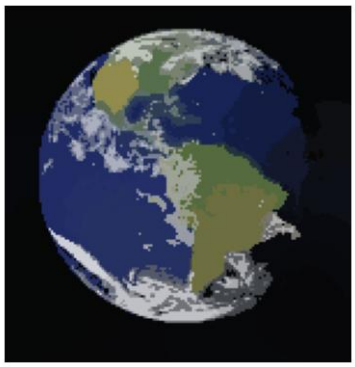
(a) Coloring each pixel by its associated peak color value.



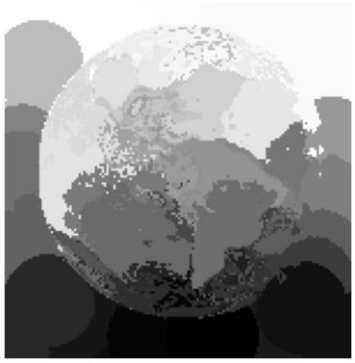
(b) Coloring each pixel by the grayscale value associated to its label.

Figure 4: Performing segmentation on the image of earth using 3D feature vectors.

However in the case of a 5D feature vector input, the two functions produce different results as shown below.



(a) Coloring each pixel by its associated peak color value.



(b) Coloring each pixel by the grayscale value associated to its label.

Figure 5: Performing segmentation on the image of earth using 5D feature vectors.

We observe that taking into account the spatial dimensions leads to clusters being more localised, even though many of them are colored with almost identical color values. The higher number of clusters also explains the significantly longer execution time.

4 Experiments

We test the meanshift segmentation algorithm for the following 3 pictures. In order to reduce computation time, the images are resized from their original size of 481×321 to 361×241 (75% of their original size in each dimension).

(a) Test image 1.

(b) Test image 2.

(c) Test image 3.

Figure 6: Images used for the experiments.

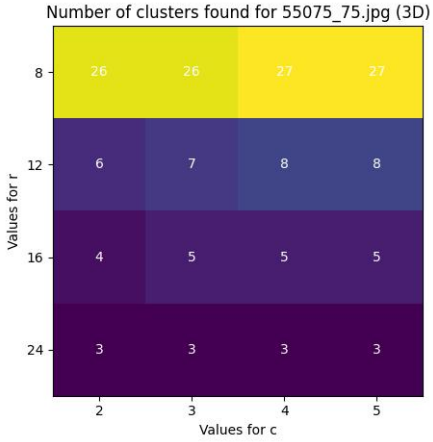
We perform segmentation with both 3D and 5D feature vectors using 16 distinct pair of values (r, c), where:

- r takes the values $(8, 12, 16, 24)$;
- c takes the values $(2, 3, 4, 5)$.

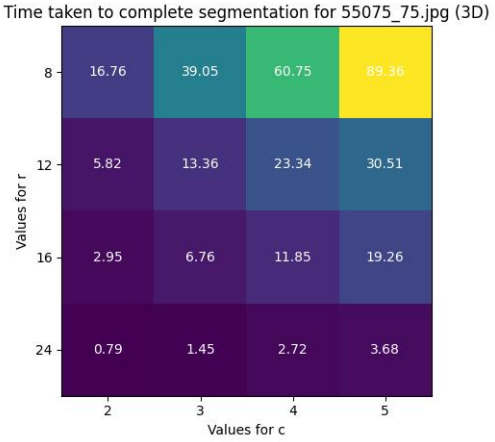
4.1 Results & discussion

4.1.1 3D feature vectors

We first look at the results of segmentation using 3D feature vectors. We plot the number of clusters found and total execution time taken by the segmentation algorithm against the 16 pairs of (r, c) values for each image.

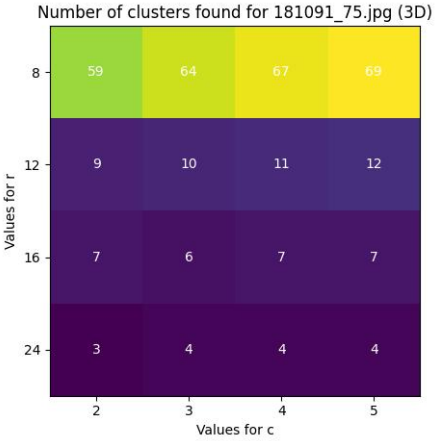


(a) Number of clusters found for each combination of (r, c).

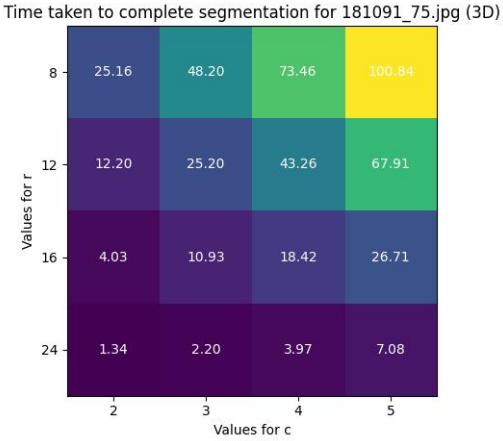


(b) Total execution time for each combination of (r, c).

Figure 7: Comparison of clusters found by and execution time taken by the meanshift segmentation algorithm with 3D feature vectors for test image 1.

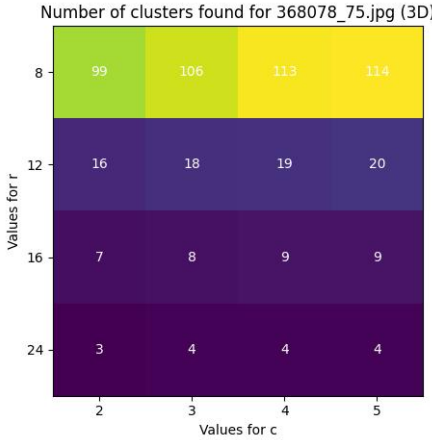


(a) Number of clusters found for each combination of (r, c).

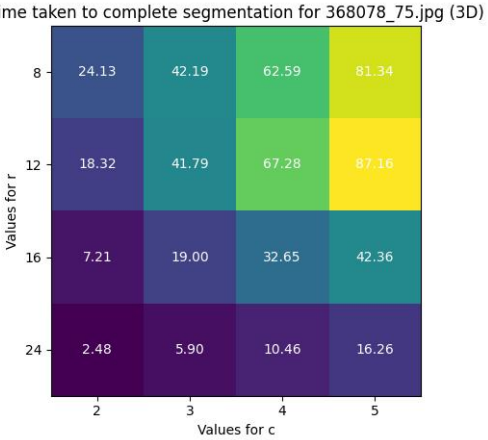


(b) Total execution time for each combination of (r, c).

Figure 8: Comparison of clusters found by and execution time taken by the meanshift segmentation algorithm with 3D feature vectors for test image 2.



(a) Number of clusters found for each combination of (r, c).



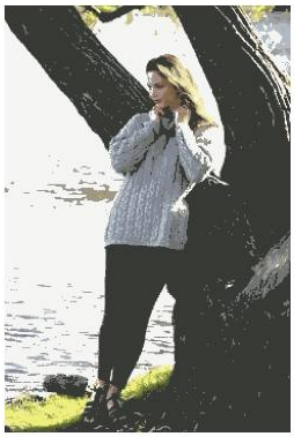
(b) Total execution time for each combination of (r, c).

Figure 9: Comparison of clusters found and execution time taken by the meanshift segmentation algorithm with 3D feature vectors for test image 3.

We observe that as expected, smaller values of r and a small ratio r/c lead to the discovery of more clusters. This also corresponds to increased computational time. In general, the computational time is of similar order of magnitude on the lines where r/c is constant, while generally decreasing as r increases along the same lines.

In general the algorithm applied to 3D feature vectors seems to perform well at separating the distinct colors for small values of r and large values of c. It can generally separate a picture well into its distinct color components, thus providing good segmentation whenever the image contains many shapes of distinct colors. However, many minor details are still captured as different clusters.

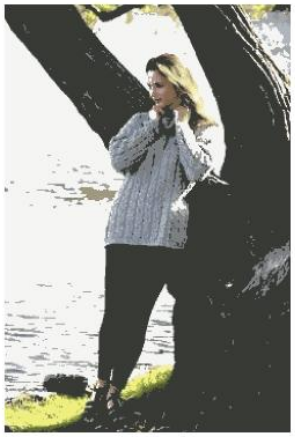
(a) Test image 1 ($r = 8$, $c = 5$, 27 clusters found).



(b) Test image 2 ($r = 8$, $c = 4$, 67 clusters found).

(c) Test image 3 ($r = 8$, $c = 4$, 113 clusters found).

(d) Test image 1 ($r = 8$, $c = 3$, 26 clusters found).



(e) Test image 2 ($r = 8$, $c = 2$, 59 clusters found).

(f) Test image 3 ($r = 8$, $c = 2$, 99 clusters found).

Figure 10: Results of segmentation algorithm for 3D features, small r and large c.

As r and the ratio r/c increase, the total number of clusters decreases. We observe relatively good segmentation results, with more details blurred into each other.

(a) Test image 1 ($r = 12$, $c = 5$, 8 clusters found).

(b) Test image 2 ($r = 12$, $c = 3$, 10 clusters found).

(c) Test image 3 ($r = 12$, $c = 4$, 19 clusters found).

(d) Test image 1 ($r = 12$, $c = 3$, 7 clusters found).

(e) Test image 2 ($r = 12$, $c = 5$, 12 clusters found).

(f) Test image 3 ($r = 12$, $c = 2$, 16 clusters found).

Figure 11: Results of segmentation algorithm for 3D features, medium r/c.

Finally, for large r and r/c we get significantly fewer clusters. Many details are now blurred, however the general shape of the distinct objects is still recognisable.

(a) Test image 1 ($r = 24$, $c = 2$, 3 clusters found).

(b) Test image 2 ($r = 24$, $c = 4$, 4 clusters found).

(c) Test image 3 ($r = 24$, $c = 3$, 4 clusters found).

(d) Test image 1 ($r = 16$, $c = 2$, 4 clusters found).

(e) Test image 2 ($r = 16$, $c = 2$, 7 clusters found).

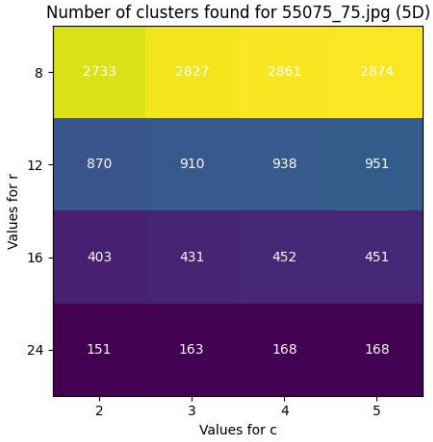
(f) Test image 3 ($r = 16$, $c = 2$, 7 clusters found).

Figure 12: Results of segmentation algorithm for 3D features, large r/c.

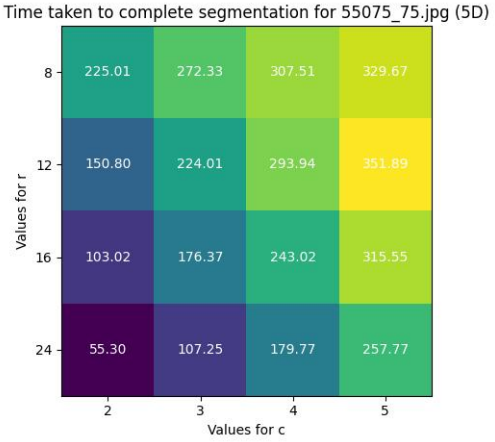
In conclusion, the meanshift algorithm applied to 3D feature vectors is efficient at segmenting images by detecting distinct color clusters, and it does so at a relatively low time cost. The level of detail of the segmented picture can be controlled by changing the values for r and c. By tracing the outline of each cluster one can obtain relatively successful segmentation of the image.

4.1.2 5D feature vectors

We now examine the results of the meanshift segmentation algorithm applied to 5D feature vectors. Once again, we plot the number of clusters found and total execution time taken by the segmentation algorithm against the 16 pairs of (r, c) values for each image.

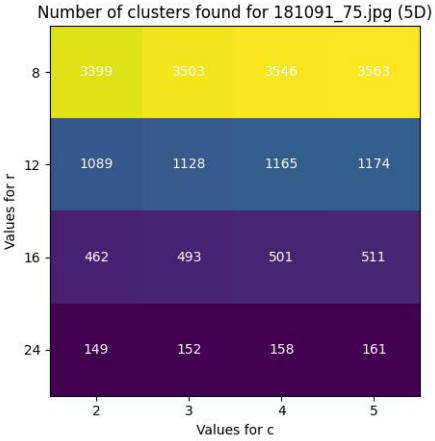


(a) Number of clusters found for each combination of (r, c).

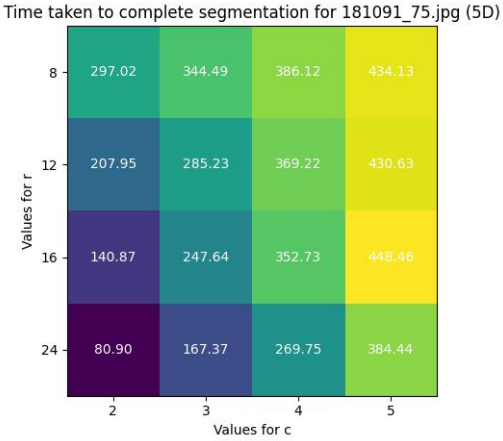


(b) Total execution time for each combination of (r, c).

Figure 13: Comparison of clusters found and execution time taken by the meanshift segmentation algorithm with 5D feature vectors for test image 1.

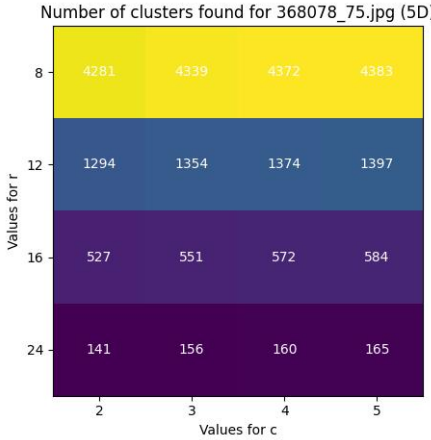


(a) Number of clusters found for each combination of (r, c).

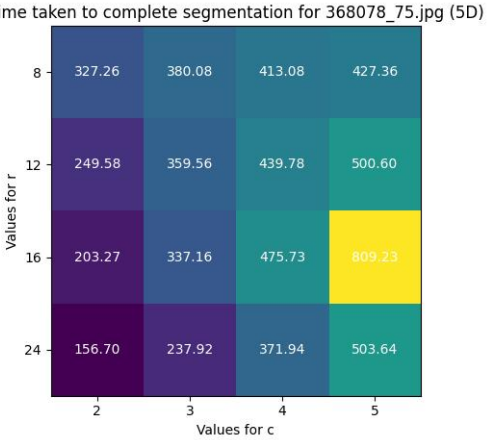


(b) Total execution time for each combination of (r, c).

Figure 14: Comparison of clusters found and execution time taken by the meanshift segmentation algorithm with 5D feature vectors for test image 2.



(a) Number of clusters found for each combination of (r, c).



(b) Total execution time for each combination of (r, c).

Figure 15: Comparison of clusters found and execution time taken by the meanshift segmentation algorithm with 5D feature vectors for test image 3.

We observe that the algorithm tested with the same parameters as before for 5D feature vectors produces significantly more clusters, and has much longer execution time. The trends we observed about the number of clusters and the total execution time with respect to r and r/c are still present.

This time however, for small and medium values of r and r/c we get essentially no useful segmentation, since the clusters found are very localized. This can be seen by the grayscale coloring of the clusters according to their label, which looks like random noise.

(a) Test image 1 ($r = 8$, $c = 4$, 2861 clusters found).

(b) Test image 2 ($r = 12$, $c = 3$, 1128 clusters found).

(c) Test image 3 ($r = 16$, $c = 5$, 584 clusters found).

(d) Grayscale coloring of the clusters for image 1.

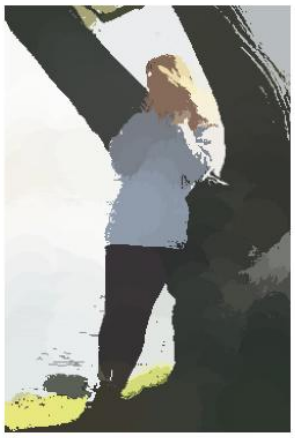
(e) Grayscale coloring of the clusters for image 2.

(f) Grayscale coloring of the clusters for image 3.

Figure 16: Results of segmentation algorithm for 5D features, small/medium r and r/c.

For larger r and r/c we observe some more useful segmentation happening. Moreover, it is clear that considering the spatial coordinates of each pixel has a blurring effect on the reconstructed images, as small lines and details get blended with their neighbours, thus only leaving the general shape of each object visible. This is a useful feature when we are interested to trace only the outline of each large object in the picture.

(a) Test image 1 ($r = 24$, $c = 2$, 151 clusters found).



(b) Test image 2 ($r = 24$, $c = 2$, 149 clusters found).

(c) Test image 3 ($r = 24$, $c = 2$, 141 clusters found).

(d) Grayscale coloring of the clusters for image 1.

(e) Grayscale coloring of the clusters for image 2.

(f) Grayscale coloring of the clusters for image 3.

Figure 17: Results of segmentation algorithm for 5D features, small/medium r and r/c.

One way to achieve more useful segmentation would be to join neighbouring clusters together in case the colour values of their peaks agree. We could also attempt to use the 5D feature vectors with larger values for r and c. We notice however that this also has the effect of making the boundaries of each object less clear, as neighbouring pixels right on the boundary are sometimes put in the same cluster. Thus, segmentation performed through this algorithm will be more successful at only keeping the outside border of each object, at the cost of this border being less precise.