Maastricht University
Department of Data Science and Knowledge Engineering
Master's Program in Artificial Intelligence

Computer Vision

18th May, 2021

Assignment 1 - Mean Shift

Authors: Dhruv Rathi (i6265197)

https://github.com/dhruv2601/MeanShiftSegmentation

Contents

1 Introduction

Mean-shift algorithm is an image segmentation algorithm that is used in Computer Vision and Image process, the task of segmentation entails better understanding of the features present in an image and the basis for this task is to cluster together the components that are most similar to each other and therefore have a better understanding of an image that is easier to interpret.

In the current assignment, we implement three versions of the mean-shift algorithm, each algorithm is better in performance than the previous and this is attributed to the optimisations that follow the original algorithm. Broadly, two functions are implemented for mean-shift algorithm. Firstly, *findpeak* is responsible for identifying the correct peaks for each data-point of the given data, further, the second function *meanshift* calculates and stores, with the help of the findpeak function, the labels corresponding to each data point and returns the labels as well as peaks list.

2 Overview

2.1 Aim

The aim of the assignment is that given an input data - which can be distributions obtained either directly from data or inferred from images, the output results contains the labels that each data point corresponds to in addition to a list of peaks that the algorithm identifies from the data.

2.2 Input Data

The code should be flexible to incorporate several different kind of formats as well as dimensions of data. For this assignment, we test our algorithms on two kinds of data - points distribution in space, more precisely, *pts.mat* containing 2000 data points in 3 dimensinos distributed over two 3D gaussians. The other type of data that is processed is varied types of image data(Karras, Laine, & Aila, 2019), including small images with faces and the images from Berkley data[1].

2.3 Pre-processing

2.3.1 CIELAB conversion

After the input is presented in the desired format of ((shape), channels), the first step in pre-processing is conversion of the RGB color space to the segmentation oriented CIELAB color space, the CIELAB space is preferred because of more accurate computations of Eucledian distances in this space for the colors interpretable by the human eye.

[1]https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/BSDS300/

2.3.2 Smoothing

The next step is smoothing of the image by applying Blur to it, this step is optional and covered in-depth in the experimentation section, the idea behind using Blur techniques is it makes the color space more linear/smooth and therefore is suitable for segmentation tasks.

2.3.3 Feature - 3D vs 5D

Further, for improved performance of the segmentation task, extending features beyond colors and including spatial information before grouping the similar colors pixels together is an approach that helps in grouping only those parts of the image together which are spatially compatible upto a particular threshold and therefore maintaining localisation data for segmentation. Thus, we provide the option to extend the 3D (L*a*b) space into the 5D (L*a*b + (x,y)) space, where x and y are the positional vectors of each pixel.

2.4 Vanilla Mean-shift Algorithm - *meanshift()*

Post pre-processing, the peak finding algorithm comes into play. The pre-processed output is reshaped into a flat array of data. For each datapoint in this data, a function findpeak is called, which calculates the center of the cluster, i.e. the peak by calculating Eucledian distance between that point and rest of the points in the dataset. The region of search is constrained by parameter - r, i.e. radius. The function MeanShift shifts the window by moving the center to the mean obtained from all the datapoints that are identified with the current window, this shifts this current window to a new position that is towards high density of points. A threshold value, defined as t=0.01 determines the convergence and stops the iterations accordingly. This approach is not time efficient, therefore further two optimisations are proposed.

2.5 Optimisation 1 - *meanShiftOptOne()*

The trick in this optimisation is to avoid iterating through each single datapoint and working on the assumption that points that lie close to the peak are likely to converge to the peak. We will see in the experimentation that this assumption indeed holds true.

2.6 Optimisation 2 - *meanShiftOptTwo()*

The second optimisation in addition to the first one, takes into account that the datapoints that lie within a certain threshold of the search path also converge to the associated peak. We take the threshold as r/c (c denotes a defined constant). For this optimisation, the *meanShiftOptTwo* calls *findpeakopttwo* which calculates the points in the search path and stores a variable *cpts* for the points which indeed satisfy the constraints.

2.7 Segmentation Output

The function *generateSegmentedIm()* takes as input the flattened output from meanshift as the data, peaks and labels. The segmented image is calculated by picking the labels and their corresponding peak values. Additionally, the output segmented image is produced by selecting only the color features in case of 5D feature space(L*a*b + (x,y)).

3 Experiments

3.1 Improvements over the two 3D Gaussians dataset

The first set of experiments are performed on the provided dataset, *pts.mat*. Figure 1 represents a plot where the expected output can be observed as the algorithm was able to identify two different clusters based on the peaks and labels. This proves that the clustering approach works successfully.

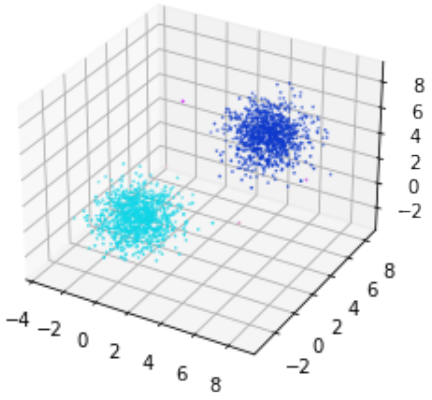


Figure 1: Cluster Plot of pts.mat

Further, the dataset is used to find out how the optimisations perform timewise. As observed from Table 1, the first optimisation is significantly better than the vanilla implementation and on a similar scale, the second optimisation is a great improvement over the first one and exponentially better than the vanilla implementation.

Performance Evaluation	
Function	Time Taken(seconds)
meanshift()	1.356
meanshiftOptOne()	0.447
meanshiftOptTwo()	0.0336

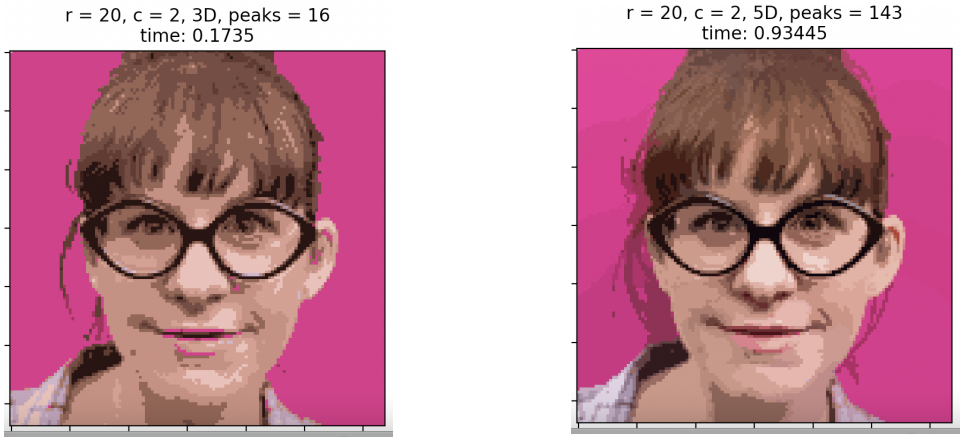
Table 1: Evaluation of meanshift's implementations

3.2 Function evaluation on Images

The implementation of the algorithm are tested on different types of images, with varying characteristics, features, size and color-maps. The experiments are performed over both the features - 3D and 5D, with varying radius and constant c and further make performance observations and observe the effect with changes in r and c, from our results obtained.

3.2.1 Facial images

The dataset(Karras et al., 2019) provides various images with contrasting background features while the majority of the photo is acquired by facial features.



(a) Output 3D

(b) Output 5D

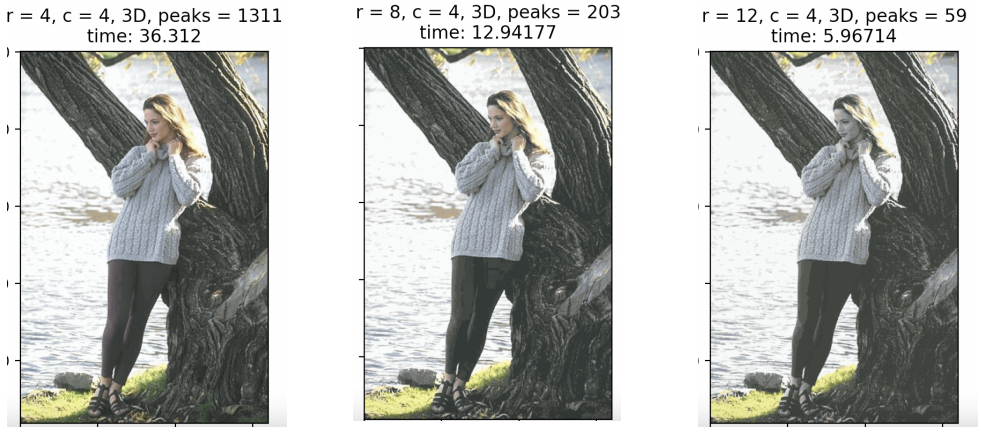
Figure 2: Plots of face images, same r and c, different types - 3D and 5D

Observations -

The most interesting observation from this experiment is the difference that selecting the feature type brings, the lips of the person in the 3D output matches the background color, thus making evidently clear that the peak selection is incorrect for the lips region. On the other hand, for the 5D feature, there is no mixing of colors because the 5D feature contains spatial features as well and therefore the background and lips being very far each other in the image, their peaks are not combined. Additionally, it can be observed that as the number of features increases the computation time increases.

3.2.2 Berkley Dataset - Image 1

First experiment on this image was performed for three different values of r while keeping c as constant, c=4, this set was performed for 3D features. The results are measured on the following radius, r=[4,8,12].



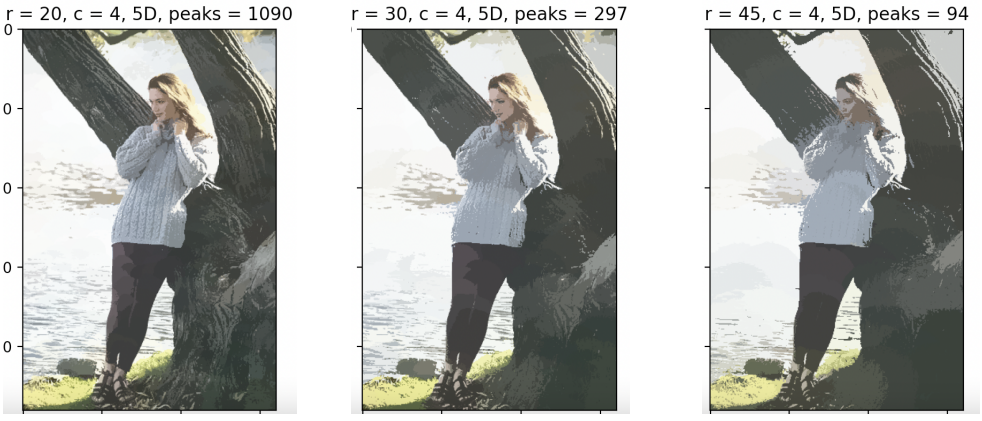
(a) r=4 (b) r=8 (c) r=12

Figure 3: Plots of image1, with different r and same c. For 3D.

Observations-

The first clear observation is that with the increase in radius, the algorithm becomes faster in computation time. Hence, our assumption stated in 2.6 holds correctly. Secondly, as the radius increases the number of peaks decreases. This can be attributed to the increase in window sizes and therefore the search path covered, which helps in achieving the convergence faster.

Part two of this experiment - The second experiment on this image was performed for three different values of r while keeping c as constant, c=4, this set was performed for 5D features. The results are measured on the following radius, r=[20,30,45].



(a) r=20 (b) r=30 (c) r=45

Figure 4: Plots of image1, with different r and same c. For 5D.

Observations-

Similar observations can be seen as above for change in r and c. The interesting result of this part is the how the similar colored pixels are grouped better together and

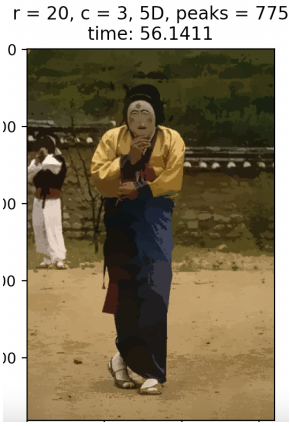
one part of the image does not influence other distinct part of the image. Therefore, proving that 5D is a better choice for this image.

3.2.3 Berkley Dataset - Image 2

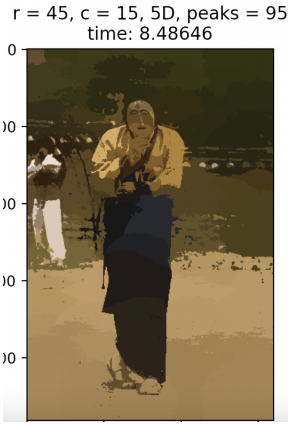
For this image, we compare different values of r and c over both feature types - 3D and 5D.

(a) Subfig 1

(b) Subfig 2



(c) Subfig 3



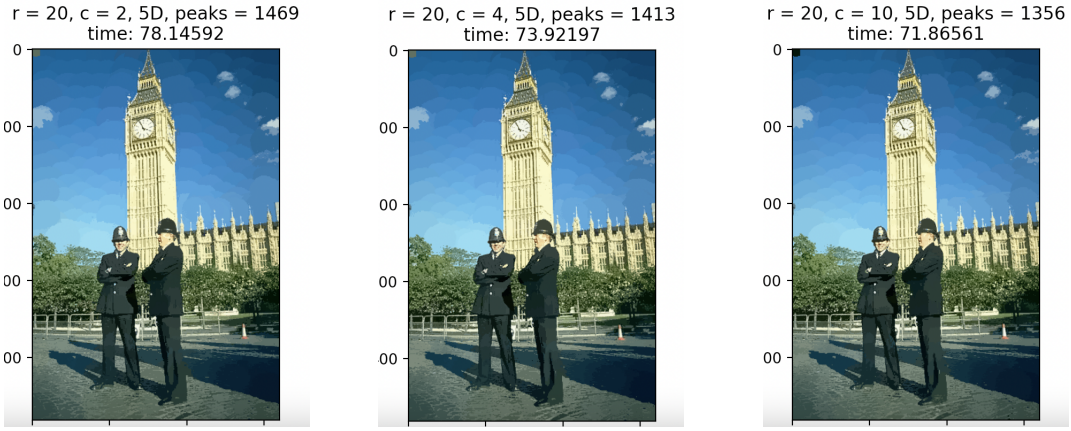
(d) Subfig 4

Figure 5: Plots of image2

The results shows that with increase in c from 4 to 10 while keeping the radius same, r=20, the number of peaks do not change, while the execution time indeed decrease. Consequently, the segmentation results do not differ by a lot as well. On the other hand, with with same radius and c=3, but with 5D features, the segmentation has improved, while being able to correctly annotate local peaks and labels, compared to 3D data, the man in the image is segmented from the background in the 5D(as can also be observed with increase in number of peaks).

3.2.4 Berkley Dataset - Image 3

In this section, we will perform two experiments - first would be with different values of c when the radius remains the same, this will be performed on 5D features. The second experiment will be dealing with changes in quality of segmentation when different kinds of blurs are introduced in the pre-processing before performing the segmentation.
First experiment: The radius, r=20, while c=[2,4,10] for 5D features.



(a) c=2 (b) c=4 (c) c=10

Figure 6: Plots of image3, with different c and same r. For 5D.

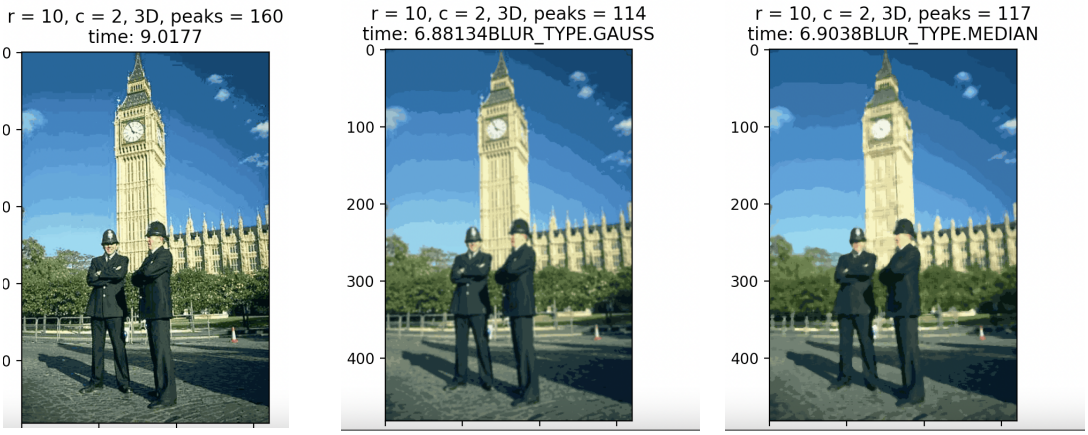
Observations-

The first clear observation is that with the increase in value of c, the algorithm becomes faster in computation time. Secondly, as the c increases the number of peaks decreases. This can be attributed to the increase in window sizes as threshold=r/c, therefore as c increases the threshold of similar peak collective factor decreases, and therefore the search path covered, which helps in achieving the convergence faster.

Second part of the experiment: Observing changes in quality of segmentation when different kinds of blurs are introduced in the pre-processing before performing the segmentation. The radius r=10, c=2 and for one iteration - Gaussian Blur with kernel size of (5,5), while the Blur used for the second iteration has kernel size of 5. The smoothing is done using OpenCV.

Observations-

The most important observation from this experiment is that after smoothing an image, the numbers of peaks decreases if other configurations remains the same. This can be attributed to the fact that smoothing decreases the number of features, as it is essentially an averaging technique. Parallel to this, the time taken for segmentation decreases as well. It should be noted that smoothing is not universally a good technique, and depends on the characteristics of the image. An instance of



(a) Normal MeanShiftOpt2() (b) Gaussian Blur (c) Median Blur

Figure 7: Plots of image3, Blur experiment

this argument is that smoothing looses the segment boundaries and other useful information.

4 Conclusion

Overall, the Mean-shift algorithm is an efficient algorithm for small images and employees a novel method of representing features in an image based on color profile and spatial profile. We observed from our experiments, that as the constant c increases in value for the optimised mean-shift algorithms, it results into increase in number of peaks, consequently a more detailed segmentation. However, one major drawback of this algorithm is that it is computationally intensive and therefore only suitable for small images as used in this assignment. For high-resolution images it is impractical to use this version of mean-shift algorithm. We explored the effect of smoothing the images, and concluded that it is helpful in case-by-case basis, as it tends to loose important information like edges that could be used for segmentation. We explored various combination of radius and c values over different kinds of images and observed that varied combinations of configurations works for different images and it essentially depends on the features of the object we want to segment.

References

Karras, T., Laine, S., & Aila, T. (2019). *A style-based generator architecture for generative adversarial networks.*