
# Mean Shift Algorithm

Computer Vision

**I6278383 - Pierre Onghena**

# 1 Abstract

In order to gather insights about segmentation properties and optimization techniques, multiple experiments will be conducted on a smaller image as the Mona Lisa. Herewith, extensive comparisons about the dimensionality of the feature space, radius of the search window and time gain of speedups will be observed to sharpen the intuition. Furthermore, a discussion can be established in the evaluation of the provided images from the Berkley Segmentation Dataset with eventual suggestions of improvement in segmentation.

# 2 Introduction

This report performs a benchmark analysis of the mean shift algorithm in regard to the setting of various segmentation variables. The provided images will be represented in the feature space by their color channels and if opted for also by the corresponding pixel coordinates. Besides the dimensionality of the feature space, a search window can be adjusted to determine the level of segmentation. By intuition, a smaller search area of the traditional algorithm would induce a local convergence with as result an increased segmentation. The downside of seeking a richer segmentation is the duration in which each pixel has to be assigned to a more specific cluster, making the algorithm time consuming.

Therefore, in order to enhance the efficiency in allocating pixels to clusters, speeding approaches are considered. The extent in which they operate is also determined by the same setting about the size of the search window. On the one hand, a marked area along the search path will be assumed to have a convergence to the same cluster as the starting point. On the other hand, once the origin of the cluster with highest density is found, a basin around this point is demarcated to have the property of being attracted to the same cluster. These optimization modifications will be introduced for a quicker convergence, and consequently a time gain over the traditional implementation.

# 3 Mean-shift algorithm

## 3.1 Implementation

This project entailed the development of an image segmentation application based on the mean shift algorithm. Therefore, *algorithm.py* in the code directory contains the different approaches described by the assignment. In addition, *segmentation.py* describes the segmentation in regard to the actual images while also referring to functions listed in *algorithm.py*. For the purpose of pertinence in this report, the use and underhand reasoning behind the functions could be derived from the documented python code. In order to execute this code, different tasks are described in *main.py* to obtain a three-dimensional plot or segmented image dependent on the described parameters.

## 3.2 Verification

The functionality of the functions will be tested on the given data set *pts.mat* to see if the algorithm is implemented correctly. The observation of the data set learns that it can be divided in two subgroups of closely related data points. Thus, the composed algorithm has to distinguish two clusters with the appropriate parameters. As is illustrated in the figure 1, a random color will be assigned to each cluster with as result a correct realization by the algorithm. This conclusion applies for both the traditional approach as the method with additional speedups.

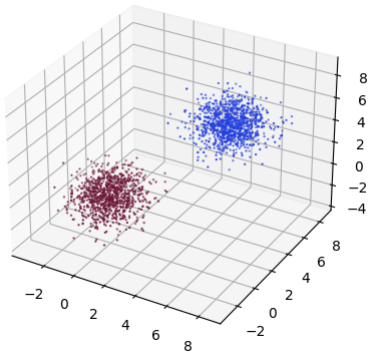



**Figure 1:** plot of pts.mat, r=2 and c=4, 2 clusters

The same reasoning is applicable when an image is simplified into segments. For example, one of the provided images returns the following segmentation according to a low search window of length 10. From the figure 2, it is clear that the hypothesis of a smaller search window will lead to more explicit clusters.

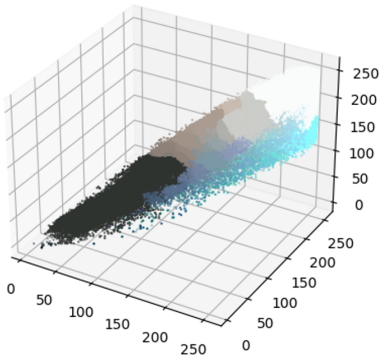



**Figure 2:** image 181091, r=10 and c=4, 235 clusters, 1.343,4s

When taking a look at the processing time, it has a duration of 1343,4s with optimization methods. This while the traditional method without speedups could not finish the clustering within a reasonable time frame. The advantage of implementing the optimization functions is thus approved for handling *imSegment.py* when actual segmentation is being performed. Although the optimization technique reduces the processing time by a lot, it still does not perform as hoped for conducting multiple experiments.

# 4 Image segmentation

As stated in the previous section, the optimization approach does not process the segmentation within a feasible time frame. Especially when evaluating different parameters, an alternative has to be sought for avoiding slack time. Therefore, the experiments entail an in-depth analysis based on a smaller image picturing the Mona Lisa. Afterwards, the results of the provided bigger images are illustrated with additional suggestions to improve the processing time.

## 4.1 Experiments

The main reason for the introduction of a smaller image is that the execution time of bigger images was not feasible for the 5D feature space setting. For instance, a bigger image needed 6,77s for completing a 3D configuration while it took 1.756s for the 5D space. As a consequence, the Mona Lisa was introduced with dimensions 195 x 261 in order to perform a comparison without limits.

**Figure 3:** Mona Lisa (195 x 261)

### Parameter tuning

During the experiments there will be a combination set of various parameters listed in table 1. In addition, the time frame of execution will be recorded to evaluate the efficiency of the algorithm under certain circumstances. Note that the optimization technique will be permanently applied to *segmentation.py* for an unbiased evaluation and time constraints.

Furthermore, the comparison of segmented images will be focused on the dimensionality of the feature space (3D / 5D) as there will be a clear distinction between both due to the additional information that pixel coordinates have to offer in segmentation. The two extra dimensions cause a higher level of differentiation between pixels so that the optimization technique operates in lower density regions with less ef-

fect. By the previous is meant that two similar color pixels, but divergent in the coordinate system will not converge to the same peak with as result more clustering and thus a higher processing time.

| feature type | 3D / 5D |
|---|---|
| search window r | 30, 20, 10, 2 |
| marked path c | 2, 8 |

**Table 1:** Combination set of parameters

### Search window

From the segmented images illustrated in Appendix A.1, the insight regarding the added value of the pixel coordinates seems to make sense. This can be demonstrated by the differentiation the 5D feature space makes in clustering the hands and frontal face with different colors. Furthermore, the processing time in table 2 increases with a smaller radius as more clustering is required. For one exception, with r=2 and 5D feature space, the elapsed time drops to its lowest point with as result a segmented image similar to the original Mona Lisa.

| | r=30 | r=20 | r=10 | r=2 |
|---|---|---|---|---|
| 3D | 0,27s | 2,63s | 23,02s | 108,51s |
| 5D | 160,29s | 262,27s | 337,86s | 130,28s |

**Table 2:** Elapsed time

### Marked path

Those points within a distance of $r/c$ along the search path are associated with the converged peak. Thus, the influence that the speedup has is inversely related with the constant value $c$. From the experiments in Appendix A.2 it can be seen that there is little alteration in the outcome of the segmented image as opposed to the processing time. The algorithm can be optimized by setting a deliberate constant value in regard to the search window $r$ to obtain an influential marking of points. Table 3 represents a preferential outcome for $c=2$ which suggests a ratio $r/c$ with $c$ being relatively small compared to $r$.

| | c=2 | c=8 |
|---|---|---|
| 3D | 0,70s | 11,87s |
| 5D | 53,00s | 884,42s |

**Table 3:** Elapsed time

## 4.2 Berkeley Dataset

After gathering insights from the experiments, the provided images are segmented according to a functional ratio $r/c$ in order to give a clear illustration of the algorithm. In general, it will be demonstrated again that the smaller the radius length the bigger the clustering will be to result in a more detailed image.

### Performance

Results for the segmentation are presented in Appendix B.1,2,3 with corresponding duration in table 4, 5, 6 respectively.

| | r=30 | r=20 | r=10 | r=2 |
|---|---|---|---|---|
| 3D | 1,74s | 4,32s | 4,40s | 455,26s |
| 5D | 401,19s | 743,43s | 1.145,05s | 1.317,28s |

**Table 4:** 181091

| | r=30 | r=20 | r=10 | r=2 |
|---|---|---|---|---|
| 3D | 2,83s | 2,88s | 20,68s | 392,80s |
| 5D | 284,35s | 415,02s | 947,80s | 1.319,88s |

**Table 5:** 55075

| | r=30 | r=20 | r=10 | r=2 |
|---|---|---|---|---|
| 3D | 449,41s | 93,93s | 19,5s | 2,88s |
| 5D | 419,55s | 692,22s | 1.121,58s | 1.288s |

**Table 6:** 368078

### Suggestion

The performance in previous section indicates a feasible running time but there is room for improvement.

A way to obtain a quicker segmentation could be enhanced by preprocessing a noise reduction to the image. More precisely, the image will be smoothed through a low-pass filter in order to reach the visual effect of blur. The technique entails a decline of image detail but on the other hand produces a less pixelated image which makes bigger images less intensive to process. Therefore, a Gaussian kernel could be applied as low-pass filter but also an average or median value of a group of pixels can blur the image. Take for instance image 368078 which initially needed 692,23s and with smoothing the performance was reduced to 356,29s. The method proposes less computational complexity which makes it more feasible for real-time image processing. Nevertheless, the technique is only considered useful for low level clustering as the segmented figure 4 lost its detail, contrast and edges.

**Figure 4:** 368078 smoothed - r=20, c=2, 5D

# 5 Conclusion

In this experimental report, an algorithm was developed for the segmentation of color images. What strikes the most is the level of detail that is obtained from image 368078 with high level clustering in Appendix B.3. This could be due to the contiguous color values being divergent in the CIELAB color space as opposed to the colors in image 55075.

Furthermore, the accomplished segmentation is successful but there seem to be some restrictions regarding the duration in spite of the applied speedups. The richness added by the pixel coordinates is a prime example of an intensified computational complexity. However, this can be compensated by a thoughtful setting of the parameters ($c$=2) as the experiments illustrated. In general, the goal of segmentation and thus simplifying an image based on their color densities is attained by the mean shift algorithm.

# Appendices

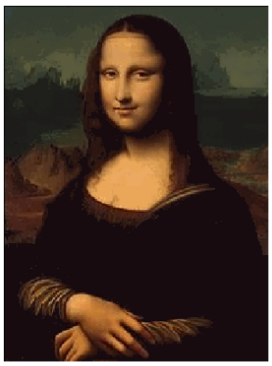
## A Experiments

### A.1 Search window $r$

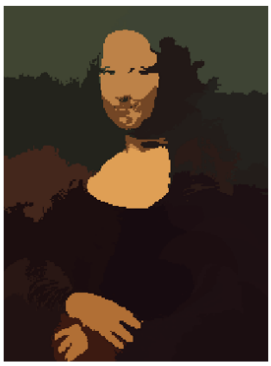
*3D, r=30, c=4*

*3D, r=20, c=4*

*3D, r=10, c=4*

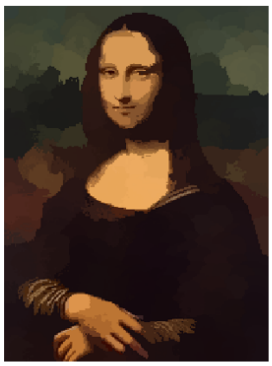


*3D, r=2, c=4*



*5D, r=30, c=4*

*5D, r= 20, c=4*



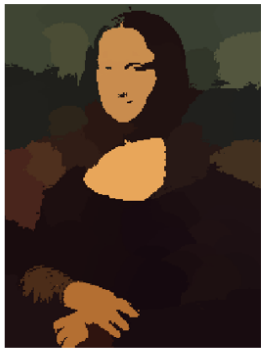
*5D, r=10, c=4*

*5D, r= 2, c=4*

## A.2 Marked path $c$

*3D, r=20, c=2*

*3D, r=20, c=8*



*5D, r=20, c=2*

*5D, r=20, c=8*

# B Berkeley Dataset

## B.1 Image 181091

3D, r=30 , c=2

3D, r=20 , c=2

3D, r=10 , c=2

3D, r=2 , c=2

5D, r=30 , c=2

5D, r=20 , c=2

5D, r=10 , c=2

5D, r=2 , c=2

## B.2 Image 55075

*3D, r=30, r=2*

*3D, r=20, r=2*

*3D, r=10, r=2*

*3D, r=2, r=2*

*5D, r=30, r=2*

*5D, r=20, r=2*

*5D, r=10, r=2*

*5D, r=2, r=2*

## B.3 Image 368078

3D, r=30, c=2

3D, r=20, c=2

3D, r=10, c=2

3D, r=2, c=2

5D, r=30, c=2

5D, r=20, c=2

5D, r=10, c=2

5D, r=2, c=2