Meanshift Algorithm Assignment

Computer Vision (2018-500-KEN4255)

Assignment Report

Department of Data Science and Knowledge Engineering

Maastricht University

Tomasz Stańczyk (6209867)

1. Overview

This report summarizes the work done over the first assignment from the Computer Vision course. The goal of the assignment was to implement the Mean Shift segmentation algorithm and to use it on provided images.

Firstly, the algorithm implementation is briefly described so as to provide a big picture idea of the applied image segmentation process. Next, initial testing, which was performed to assure the implementation correctness is presented. Subsequently, the first set of experiments, which concerns processing of smaller images, along with their results is discussed. Finally, the second set of experiments, related to bigger, desired images is discussed, including some of the processed images as the experiment outputs.

2. Implementation

The whole implementation has been performed in GNU Octave, a scientific programming language, syntax of which is largely compatible with MATLAB.

Based on the guidelines provided in the assignment description, relevant functions have been implemented: *findpeak*, *meanshift*, *findpeak_opt*, *meanshift_opt* and *imSegment*. The code of each function has been documented and attached to this report. The idea of the whole process is as follows.

The *findpeak* function finds a peak of the probability density of all points from the dataset using the search window with specified radius *r*. It is called for each point separately inside the *meanshift* function. The *meanshift* function associates each point from the dataset to a particular peak. This function returns a matrix representing all the peaks found (a peak per row and a feature per column) and a label vector with the length equal to the number of points, where each label represents the peak id for the corresponding point. For example, having found 5 peaks for 2000 points, representing 3 feature dimensions each, the peak matrix will have size of 5 rows and 3 columns, whereas the 2000-long label vector will only contain values from 1 to 5.

The *meanshift* function is called inside the *imSegment* function. There, the input image is converted into the Lab color space, reshaped into a format matching the *meanshift* and *findpeak* function and then passed to the *meanshift* function so as to receive resulting peaks and labels. Subsequently, each pixel of the image is set to the corresponding peak value according to the assignment included in the labels vector. The modified image is reshaped back to the original (input) size and converted back from the Lab into the RGB color space. The *imSegment* function returns the segmented image as well as the peaks found and labels assigned.

Important note to mention is the fact that originally, the Mean Shift algorithm is slow, especially if it is consider to call *findpeak* function for each pixel separately and independently. Therefore, 2 speedup techniques have been incorporated. First speedup (basin of attraction) associates each data point within the range less or equal than *r* from the found peak exactly to this peak.

The second speedup associates each data point which is within the distance of *r/c* from the search path leading to the found peak precisely to that peak. The most common value of *c* is 4, with several other values considered within the realization of this assignment.

Deployment of these 2 optimization techniques resulted in creation of additional functions: *findpeak_opt* and *meanshift_opt*. When optimized Mean Shift algorithm run is performed, these functions replace their non-optimized counterparts, *findpeak* and *meanshift* respectively. All the relations and dependencies between these functions and *imSegment* are preserved.

3. Initial Testing

In order to assure that the implementation has been realized appropriately, the dataset provided by the tutors has been processed. The *pts.mat* file has been loaded into the Octave environment and passed to both, *meanshit* and *meansift_opt* functions. The output peaks and labels variables along with the dataset points have been further passed to the *plot3dclusters* function, which had also been provided beforehand.

<u>Note</u>: The original data matrix (resulting from loading the pts.mat file) has been transposed before and only then passed to the meanshift functions. Moreover, the resulting peaks matrix has also been transposed before passing it to the plot3dclusters function.

In both cases, identical peaks have been found and the same labels have been assigned to corresponding pixels. The result of ploting is presented below in Figure 1:

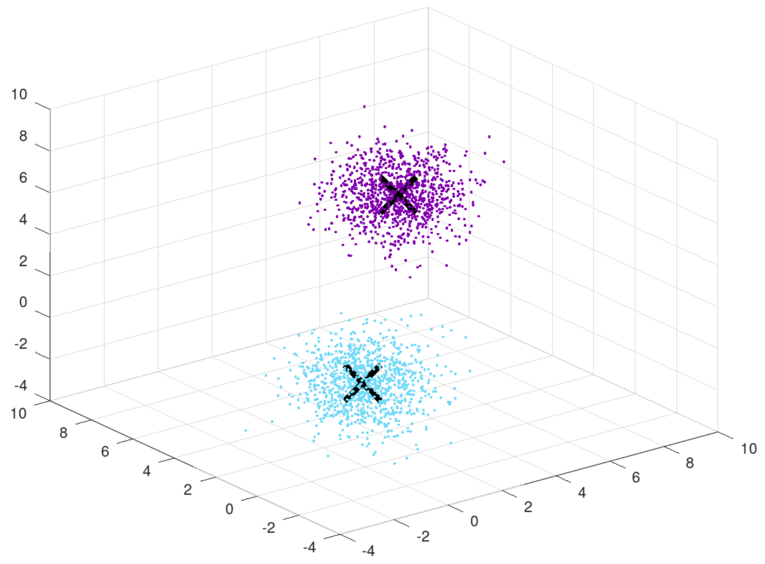

Figure 1 - Result of processing the provided dataset file, visualized with plot3dclusters function

What have differed though, was the time elapsed for the processing. In case of non-optimized version, it took 10.960 s to finish, whereas for optimized version, the time was reduced to 0.89821 s. This example already shows the significance of the aforementioned speedups incorporation. For larger datasets, which represent real images, these optimization techniques will impact the processing time to a higher extent.

4. First Set of Experiments – Smaller Images

During the first set of experiments, 2 frontal face images were being processed. In order to avoid lengthy computations, images with the sizes of 64 by 64 and 128 by 128 pixels were considered.

The original images submitted for segmentation are presented in Figure 2 and Figure 3 respectively:

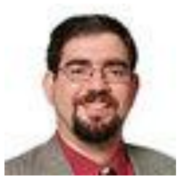

Figure 2 - Image 1 submitted for segmentation (64 by 64 pixels)

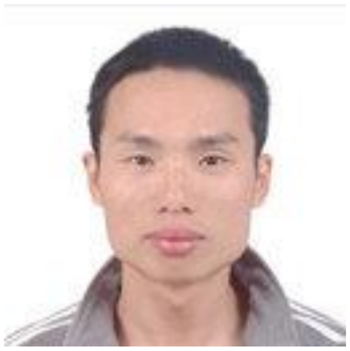

Figure 3 - Image 2 submitted for segmentation (128 by 128 pixels)

Additionally, with the view to improving the segmentation results, certain preprocessing step has been proposed. Before submitting an image to segmentation it had been initially smoothed. To be more precise, the input image have been convolved with an average filter. This operation has been realized simply by using *imsmooth* function from Octave's *image* package:

im = imsmooth(im, "Average");

The reason behind applying smoothing on an input image as a preprocessing step was the fact that the goal of the segmentation is to group together pixels which look similar so as to enhance further processing of the image. When applying smoothing filter (in particular, the average filter), the image becomes less sharp and the (Euclidean) distances between values of particular pixels are already shorter. Such a preprocessed image will most likely contribute to a better result of segmentation in terms of the mentioned pixel grouping.

The images were submitted to the segmentation also with various number of the features. 2 separate cases were considered: 3D and 5D features. In case of 3D features, R, G and B channel values were taken into account when performing the Meanshift algorithm (especially when calculating the point distances). 5D features also included x and y position coordinates of each pixel from an image.

Various processing parameter combinations have been performed on each of the presented frontal face images. These included:

- using original and pre-smoothed input image,
- applying optimized and non-optimized approach,
- considering 3D and 5D features,
- setting the *r* (searching windows size radius) value to one of the following: {0.5, 1, 2, 5, 10, 25, 50},
- keeping *c*=4.

All cases were tested, i.e. each possible combination of values regarding the options above. This resulted in 2 * 2 * 2 * 7 * 1 = 56 independent image outputs for 1 input image. Selected ones will be presented below.

Furthermore, applying different values of the *c* parameter was also tested. This time, the combinataions included:

- using original and pre-smoothed input image
- applying only optimized approach
- considering 3D and 5D features
- keeping r=10,
- setting c values to one of the following: {1, 2, 5, 10}.

Similarly as in the previous case, all possible value combinations were tested. Therefore, 2 * 1 * 2 *1 * 4 = 16 more output images (for each input image) have been generated.

Selected output images received from segmenting the image from Figure 2 are presented with figures below, including division into subcategories.

Values set as *r*=10, *c*=4, **original** image passed:

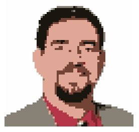
*Figure 4 - output of Image 1, **3D features, non-optimized**, r=10, c=4, original*

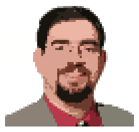
*Figure 5 - output of Image 1, **5D features, non-optimized**, r=10, c=4, original*

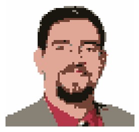
*Figure 6 - output of Image 1, **3D features, optimized**, r=10, c=4, original*

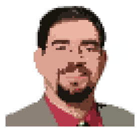
*Figure 7 - output of Image 1, **5D features, optimized**, r=10, c=4, original*

Values set as *r*=10, *c*=4, **pre-smoothed** image passed:

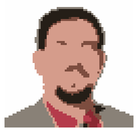
Figure 8 - output of Image 1, ***3D features**, **non-optimized**,* *r=10, c=4, pre-smoothed*

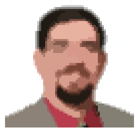
Figure 9 - output of Image 1, ***5D features**, **non-optimized**,* *r=10, c=4, pre-smoothed*

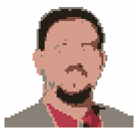
Figure 10 - output of Image 1, ***3D features**, **optimized**,* *r=10, c=4, pre-smoothed*

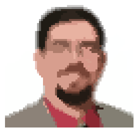
Figure 11 - output of Image 1, ***5D features**, **optimized**,* *r=10, c=4, pre-smoothed*

Value of r=10, various *c* values **3D** features, optimized, pre-smoothed image passed:

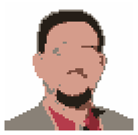
Figure 12 - output of Image 1, 3D features, optimized, r=10, ***c=1**, pre-smoothed*

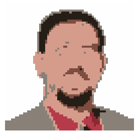
Figure 13 - output of Image 1, 3D features, optimized, r=10, ***c=2**, pre-smoothed*

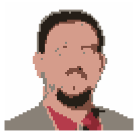
Figure 14 - output of Image 1, 3D features, optimized, r=10, ***c=5**, pre-smoothed*

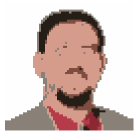
Figure 15 - output of Image 1, 3D features, optimized, r=10, ***c=10**, pre-smoothed*

Value of r=10, various *c* values, **5D** features, optimized, pre-smoothed image passed:

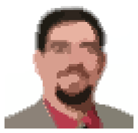
Figure 16 - output of Image 1, 5D features, optimized, r=10, ***c=1**, pre-smoothed*

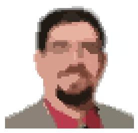
Figure 17 - output of Image 1, 5D features, optimized, r=10, ***c=2**, pre-smoothed*

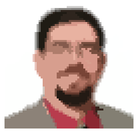
Figure 18 - output of Image 1, 5D features, optimized, r=10, ***c=5**, pre-smoothed*

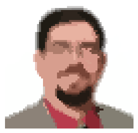
Figure 19 - output of Image 1, 5D features, optimized, r=10, ***c=10**, pre-smoothed*

Similarly as for image from Figure 2, corresponding results for input image presented in Figure 3 are presented below.

Values set as *r*=10, *c*=4, **original** image passed:

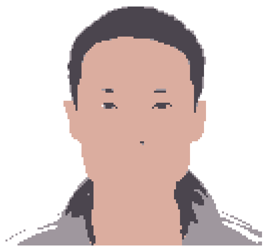
Figure 20 - output of Image 2, ***3D features, non-optimized,*** *r=10, c=4, original*

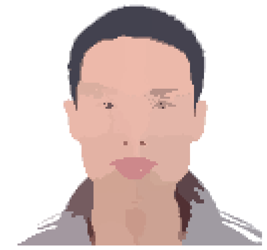
Figure 21 - output of Image 2, ***5D features, non-optimized,*** *r=10, c=4, original*

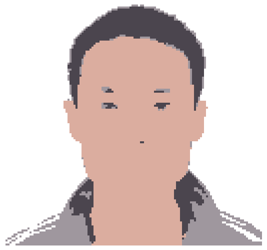
Figure 22 - output of Image 2, ***3D features, optimized,*** *r=10, c=4, original*

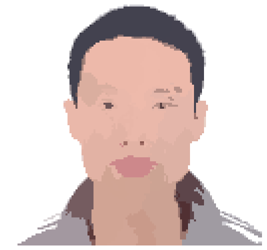
Figure 23 - output of Image 2, ***5D features, optimized,*** *r=10, c=4, original*

Values set as *r*=10, *c*=4, **pre-smoothed** image passed:

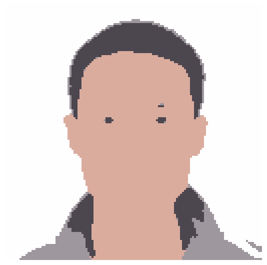
Figure 24 - output of Image 2, ***3D features, non-optimized,*** *r=10, c=4, pre-smoothed*

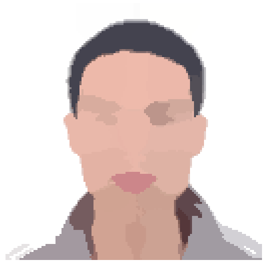
Figure 25 - output of Image 2, ***5D features, non-optimized,*** *r=10, c=4, pre-smoothed*

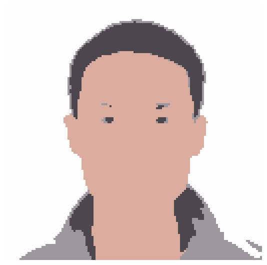
Figure 26 - output of Image 2, ***3D features, optimized,*** *r=10, c=4, pre-smoothed*

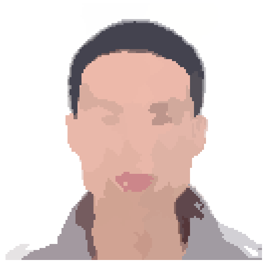
Figure 27 - output of Image 2, ***5D features, optimized,*** *r=10, c=4, pre-smoothed*

Value of r=10, distinct *c* values **3D** features, optimized, pre-smoothed image passed:

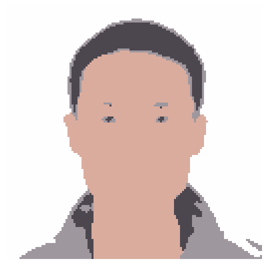
Figure 28 - output of Image 2, 3D features, optimized, r=10, ***c=1****, pre-smoothed*

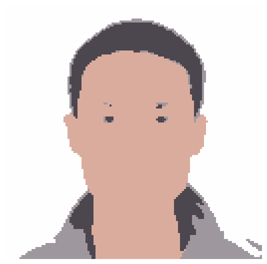
Figure 29 - output of Image 2, 3D features, optimized, r=10, ***c=2****, pre-smoothed*

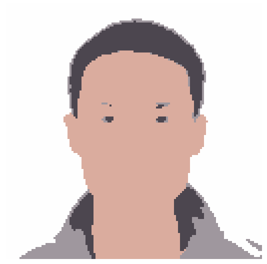
Figure 30 - output of Image 2, 3D features, optimized, r=10, ***c=5****, pre-smoothed*

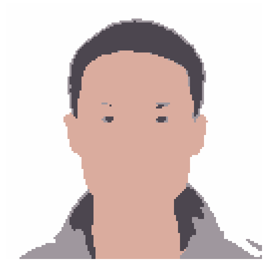
Figure 31 - output of Image 2, 3D features, optimized, r=10, ***c=10****, pre-smoothed*

Value of r=10, distinct *c* values, **5D** features, optimized, pre-smoothed image passed:

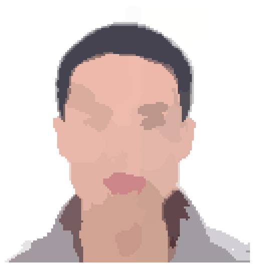
*Figure 32 - output of Image 2, 5D features, optimized, r=10, **c=1**, pre-smoothed*

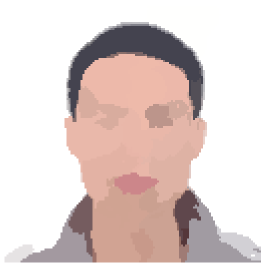
*Figure 33 - output of Image 2, 5D features, optimized, r=10, **c=2**, pre-smoothed*

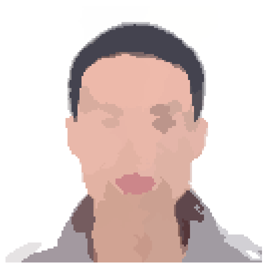
*Figure 34 - output of Image 2, 5D features, optimized, r=10, **c=5**, pre-smoothed*

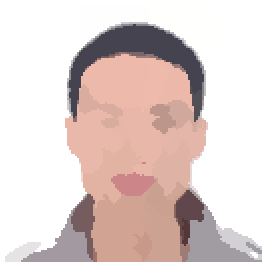
*Figure 35 - output of Image 2, 5D features, optimized, r=10, **c=10**, pre-smoothed*

Figures 4 – 19 and 20 – 35 present output images when applying the pointed parameter values on input images from Figures 2 and 3 respectively.

All the output images received during these experiments have been attached to this report in a .zip file and can be found in the following directories: *ResultsTest1*, *ResultsTest1Smoothed*, *ResultsTest1VariousC* and *ResultsTest1VariousCSmoothed*. Additionally, each of these directories includes .txt files (e.g. *testImage1timeMeasurements.txt*) containing the time elapsed during each segmentation process.

For each of the output images received through all the possible parameter combinations, the time elapsed during the whole segmentation process (i.e. within each run of the *imSegment* function) has been recorded. Table 1 presents corresponding time measurements for each combination of values with c=4, which were registered for input image from Figure 2, with size 64 by 64.

	Face image 1 (64x64) original				Face image 1 (64x64) smoothed			
	Optimized		Non-optimized		Optimized		Non-optimized	
r	3D	5D	3D	5D	3D	5D	3D	5D
0,5	5,025126	9,288658	9,027731	7,01585	5,334882	10,50507	10,16222	7,531541
1	4,887268	8,214371	12,72261	8,078491	4,627701	7,65301	13,47636	8,243177
2	4,442927	7,322492	16,47403	11,24341	4,267136	6,887275	18,73949	11,09028
5	3,843735	8,824238	36,20336	32,60255	2,776345	5,668425	30,77886	28,50489
10	1,894151	5,039384	32,32412	57,99032	1,679621	3,80674	42,42168	54,17359
25	1,155043	1,819647	27,22641	56,54829	1,162704	1,567019	36,60141	62,1645
50	1,163951	1,274677	32,67883	39,97138	1,099416	1,222223	44,71864	62,92317

Table 1 - Time measurements received during the segmentation of the first input image (various r, constant c)

Table 2 displays analogous time measurements (also for the input image from Figure 2), but with constant value of *r*=10 and varying values of *c*.

	Face image 1 (64x64), original, r=10		Face image 1 (64x64), smoothed, r=10	
	Optimized		Optimized	
c	3D	5D	3D	5D
1	1,746708	3,663003	1,510962	2,873627
2	1,869856	4,958949	1,739462	4,026107
5	1,902126	5,158558	1,736863	3,882632
10	1,880793	5,358924	1,704295	3,876425

Table 2 - Time measurements received during the segmentation of the first input image (constant r, various c)

Figures 36 – 39 display time measurement values from Table 1 and 2 plotted relevantly with respect to r or c value.

<u>Note</u>: The testing values of r and c are not uniformly distributed (see Table 1 and 2). The plots presented below are in fact scatter plots including the table values which have been connected appropriately with straight lines for better visualization.

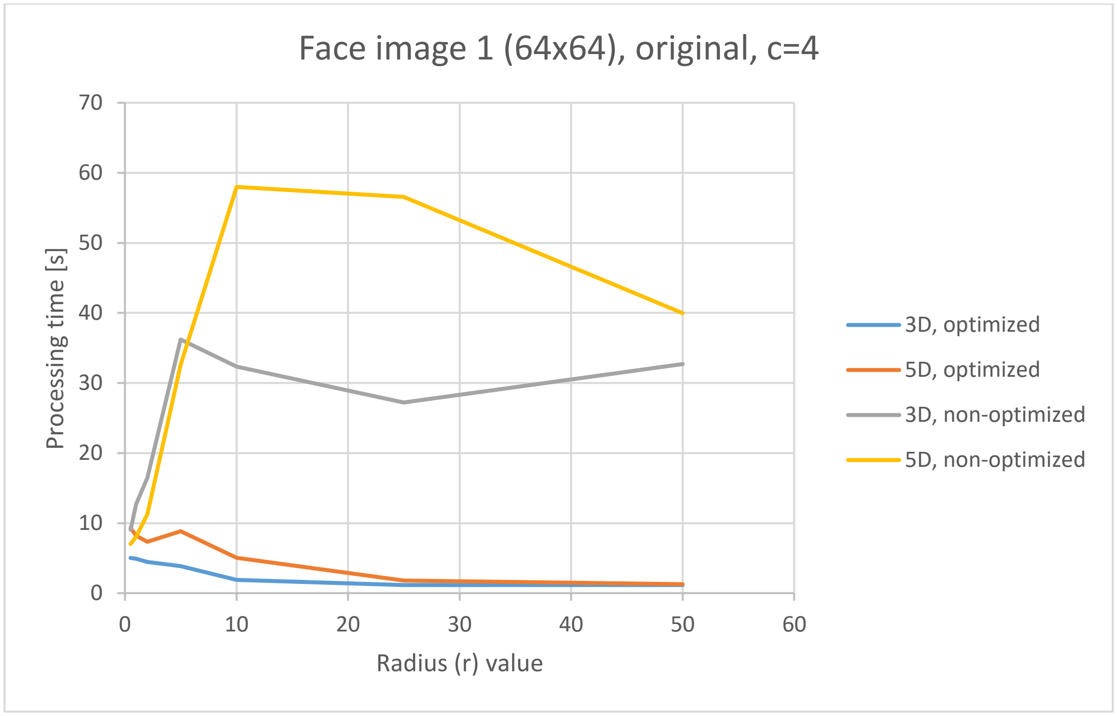

Figure 36 – Plotting time measurement values of first image processing with respect to r value (original image as an input)

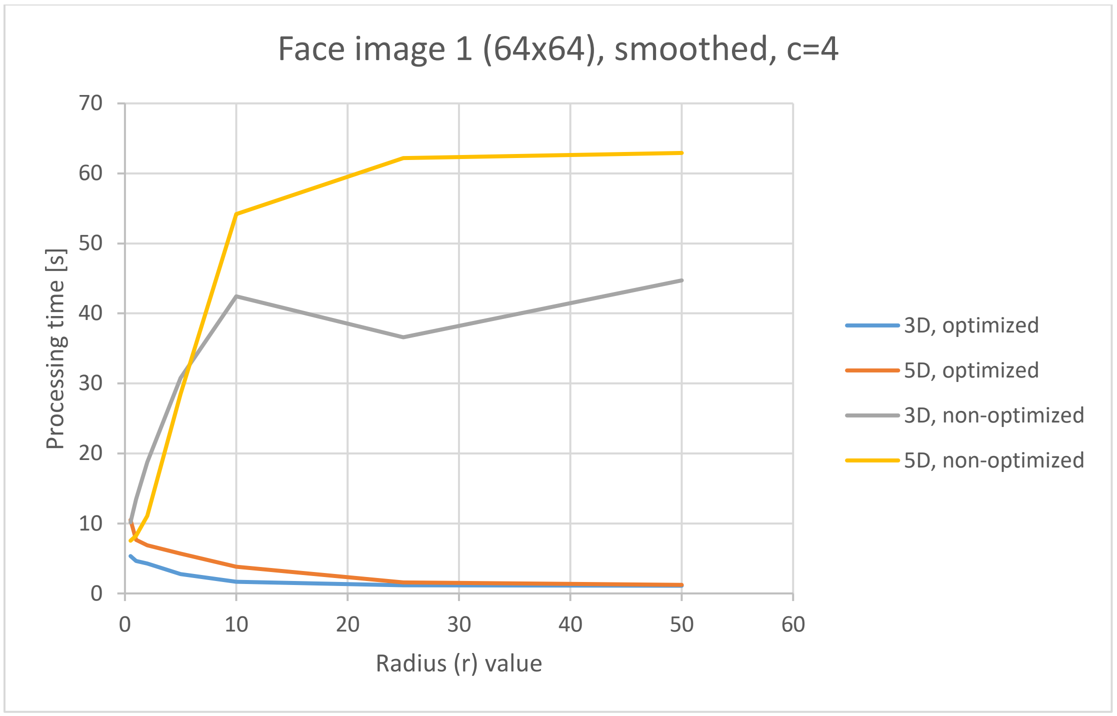

Figure 37 - Plotting time measurement values of first image processing with respect to r value (pre-smoothed image as an input)

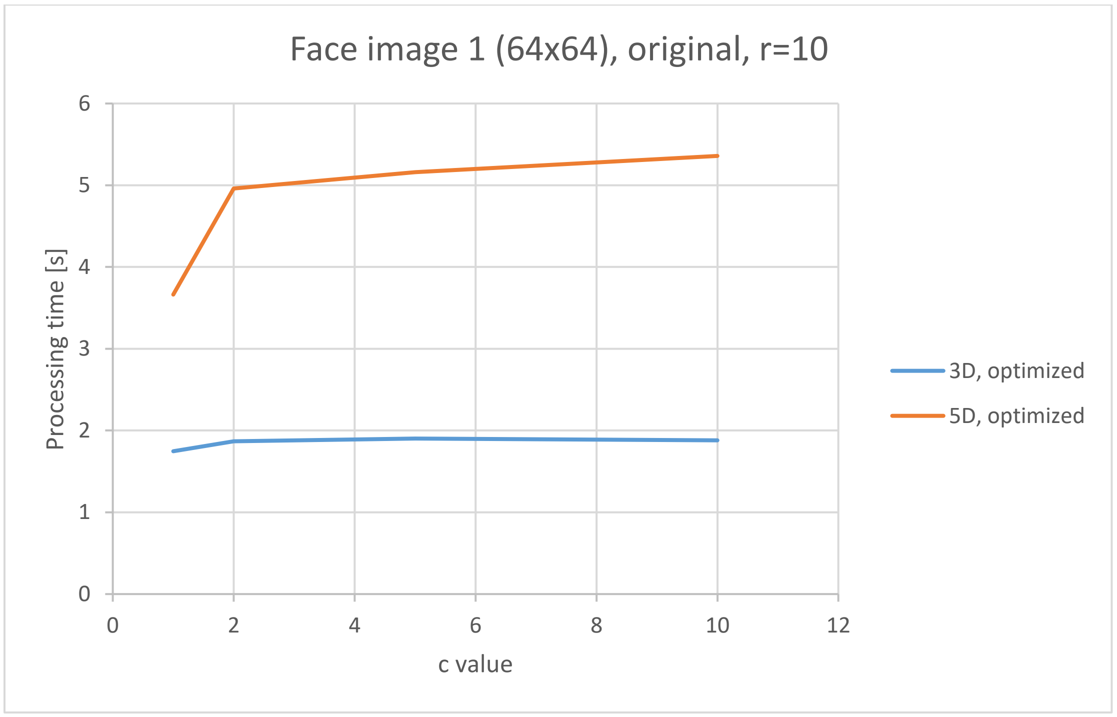

Figure 38 - Plotting time measurement values of first image processing with respect to c value (original image as an input)

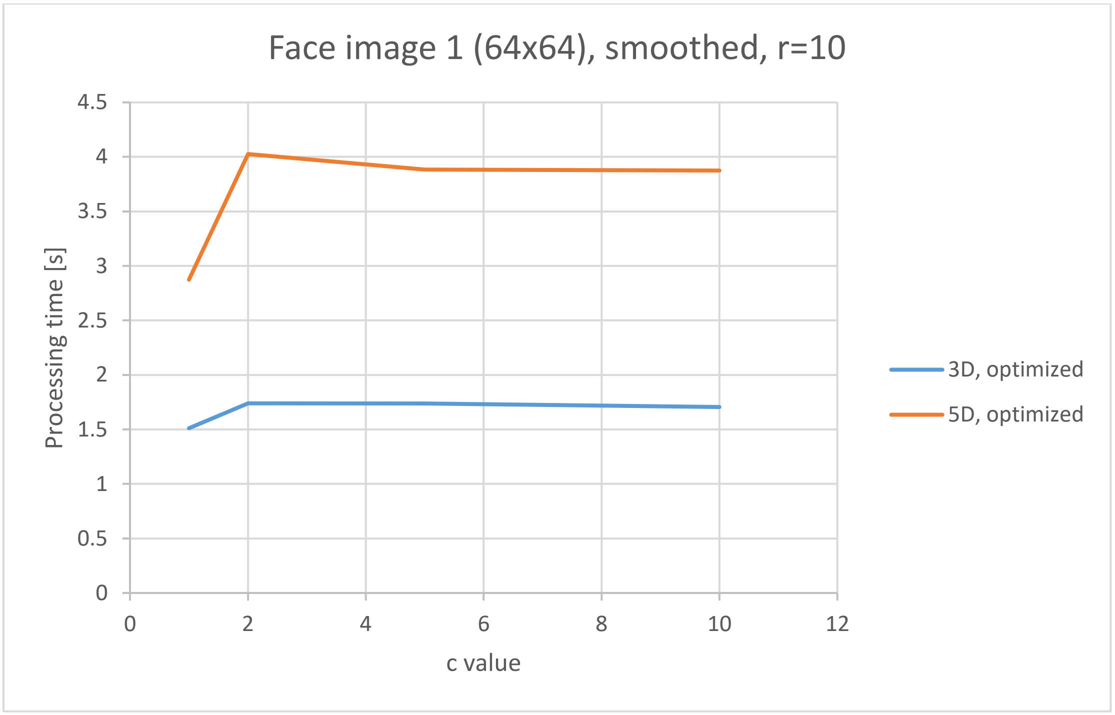

Figure 39 - Plotting time measurement values of first image processing with respect to c value (pre-smoothed image as an input)

Table 3 presents corresponding time measurements for each combination of values with c=4, which were registered for input image from Figure 3, with size 128 by 128.

	Face image 2 (128x128) original				Face image 2 (128x128) smoothed			
	Optimized		Non-optimized		Optimized		Non-optimized	
r	3D	5D	3D	5D	3D	5D	3D	5D
0,5	16,10165	61,28149	57,3425	43,14716	20,59838	65,44631	63,49214	43,2213
1	15,16685	48,48824	79,87207	47,68368	17,546	48,01895	110,9436	47,49932
2	13,58948	35,33578	152,3311	69,07482	12,99352	31,6677	179,1764	67,0574
5	7,567644	25,07761	195,2426	176,073	7,03515	22,07494	229,4588	174,822
10	5,278166	20,55925	107,2996	448,5938	5,136068	16,57424	119,438	404,2788
25	4,9258	7,342417	124,4411	543,1711	4,799114	7,172777	163,4405	573,3482
50	4,885721	5,452441	142,5358	575,0755	4,82633	5,463766	149,472	586,3065

Table 3 - Time measurements received during the segmentation of the second input image (various r, constant c)

Table 4 displays analogous time measurements (still for the input image from Figure 3), with constant value of *r*=10 and varying values of *c* though.

	Face Image 2 (128x128), original, r=10		Face Image 2 (128x128), smoothed, r=10	
	Optimized		Optimized	
c	3D	5D	3D	5D
1	5,398694	13,55542	5,22516	13,94807
2	5,486216	20,05317	5,398278	16,28751
5	5,499611	20,54784	5,376924	18,52211
10	5,539128	20,58061	5,383192	17,04209

Table 4 - Time measurements received during the segmentation of the first input image (constant r, various c)

Figures 40 – 43 present time measurement values from Table 3 and 4 plotted relevantly with respect to the value of r or c.

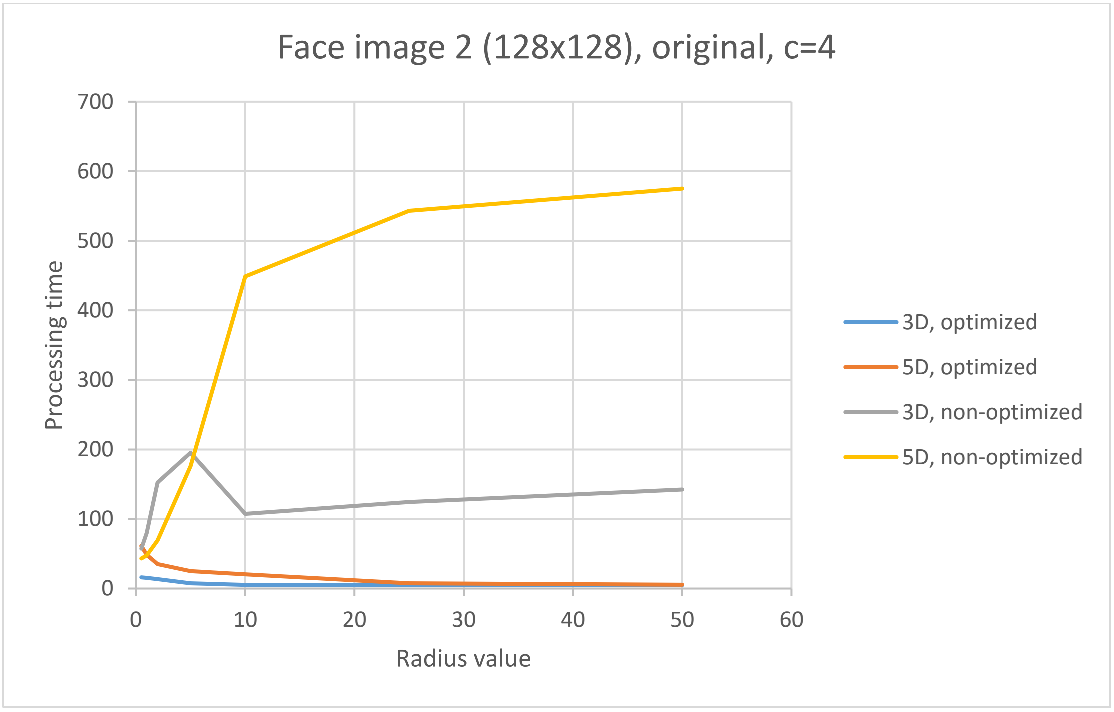

Figure 40 - Plotting time measurement values of second image processing with respect to r value (original image as an input)

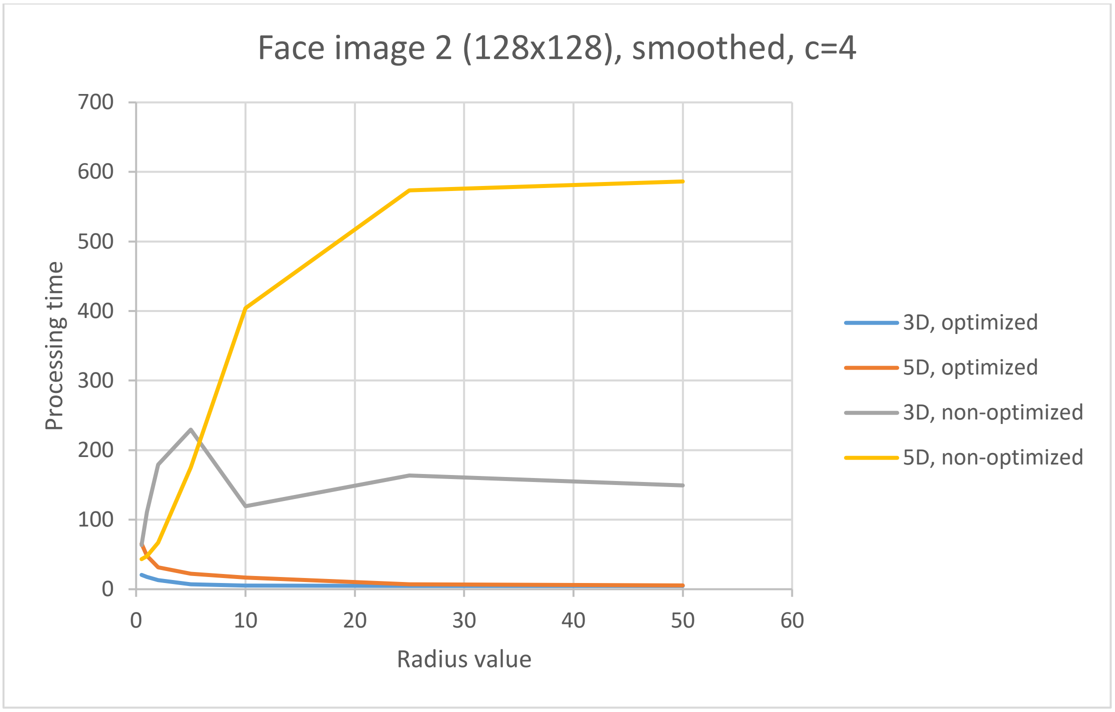

Figure 41 - Plotting time measurement values of first image processing with respect to r value (pre-smoothed image as an input)

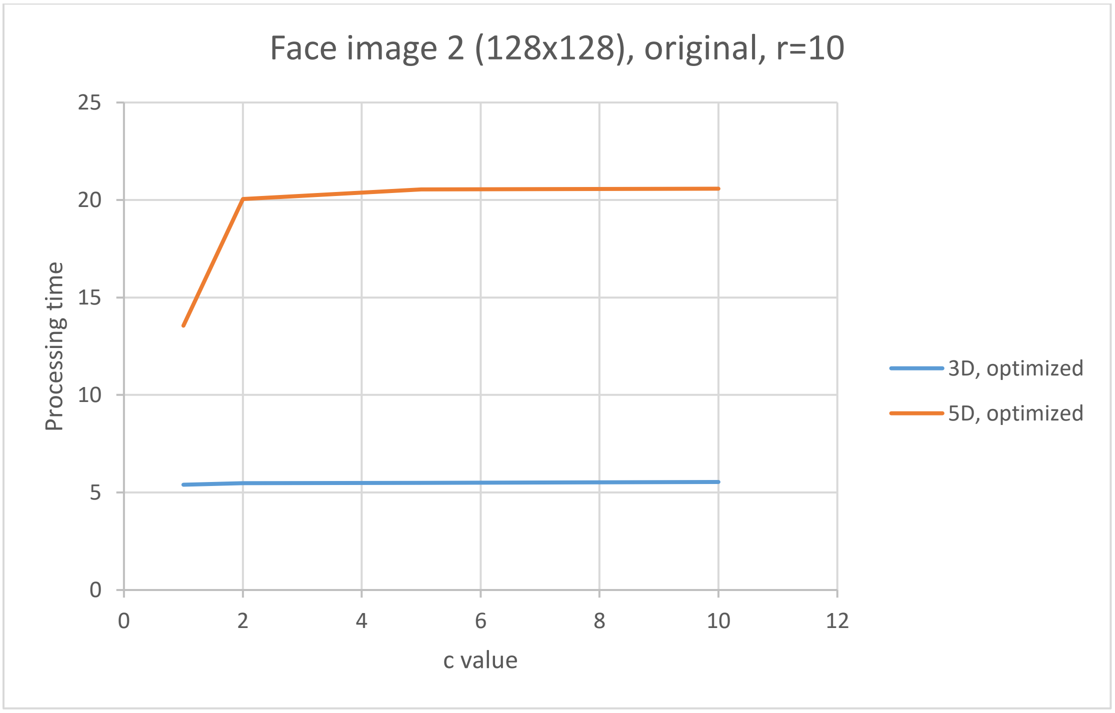

Figure 42 - Plotting time measurement values of first image processing with respect to c value (original image as an input)

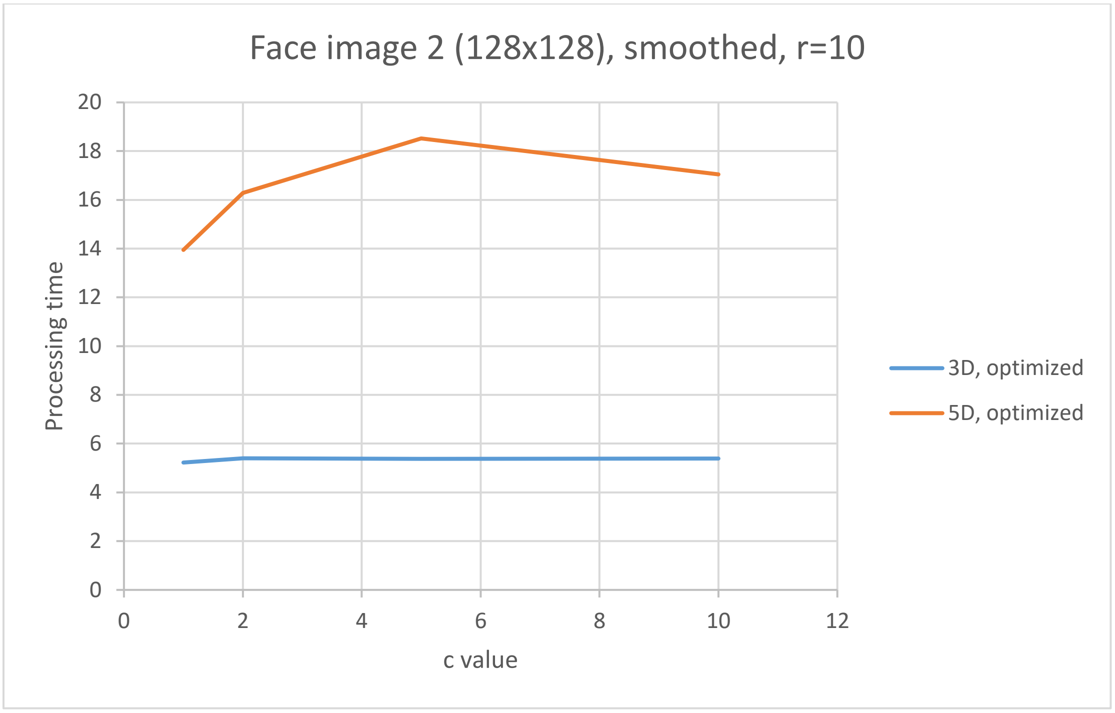

Figure 43 - Plotting time measurement values of first image processing with respect to c value (pre-smoothed image as an input)

5. Second Set of Experiments – Bigger Images

During the second set of experiments, 3 bigger images of the same size (321 by 481 pixels), as referenced in the assignment instructions, were being processed.

Similarly to the previous set of experiments, various processing parameter combinations have been performed on each of the desired images. These included:

- using original and pre-smoothed input image,
- applying only optimized approach,
- considering 3D and 5D features,
- setting the *r* (searching windows size radius) value to one of the following: {10, 20, 30},
- keeping *c*=4.

Once more, each possible combination of values regarding the options above have been considered. This resulted in 2 * 1 * 2 * 3 * 1 = 12 independent image outputs for 1 input image.

Due to the time constraints of this assignment and unreasonably long time elapsed during processing bigger images (comparing to the frontal face images from the previous part), only optimized approach has been applied.

Selected output images are presented below along with the original input images. The parameters applied and number of peaks found is mentioned for each output image displayed. Each comparison is juxtaposed with respect to different kind of parameters (3D vs. 5D features, different value of *r*, original image vs. smoothed).

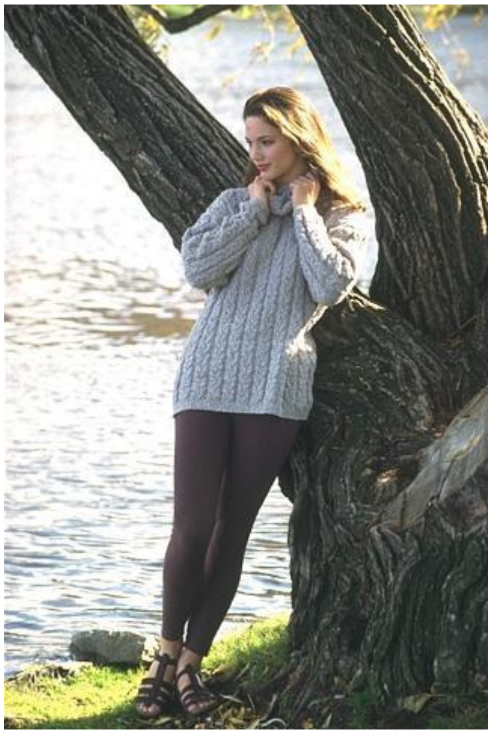

Figure 44 – Original first bigger image submitted for segmentation

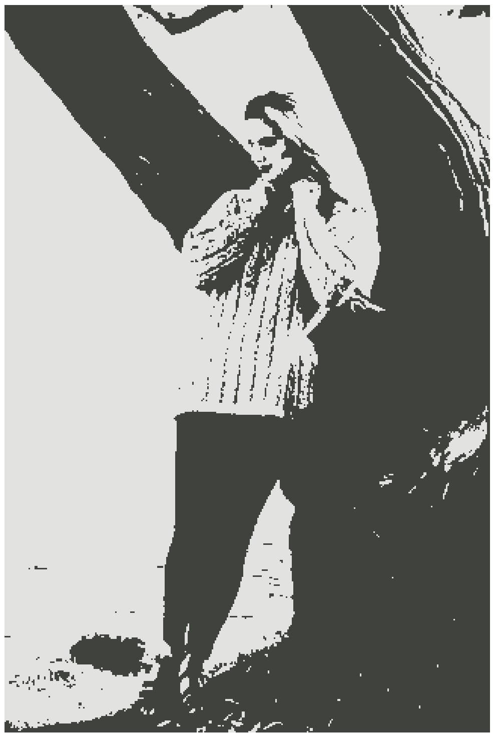

Figure 45 - output of first bigger image,
*r=30, **3D features**, pre-smoothed,*
<u>*2 peaks found*</u>

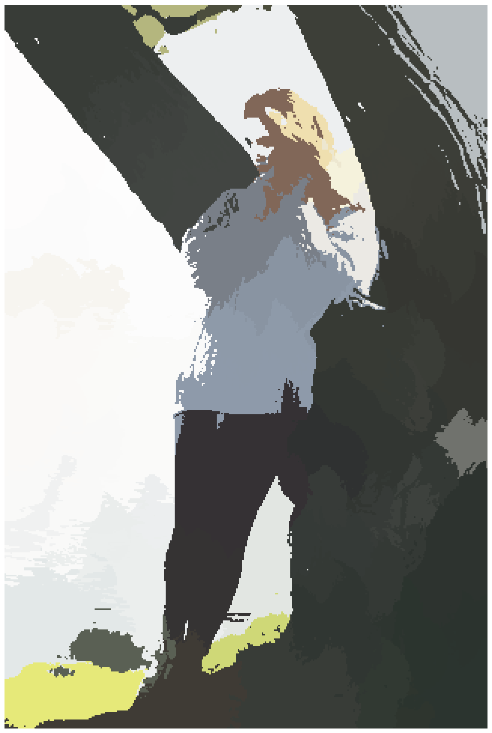

Figure 46 - output of first bigger image,
*r=30, **5D features**, pre-smoothed,*
<u>*67 peaks found*</u>

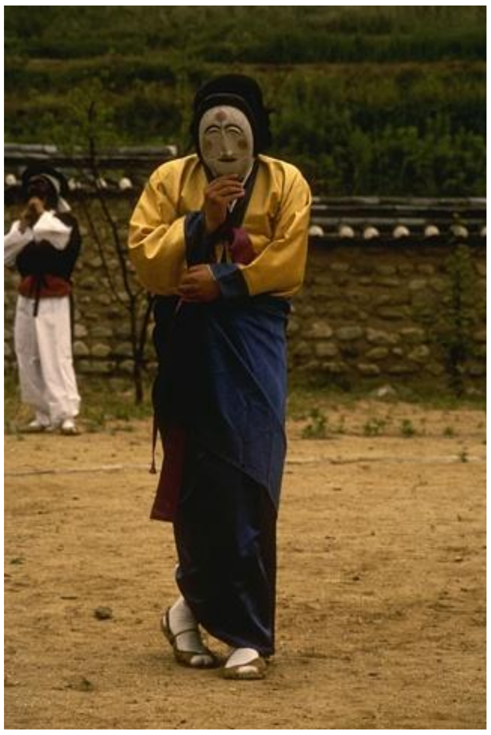

Figure 47 - Original second bigger image submitted for segmentation

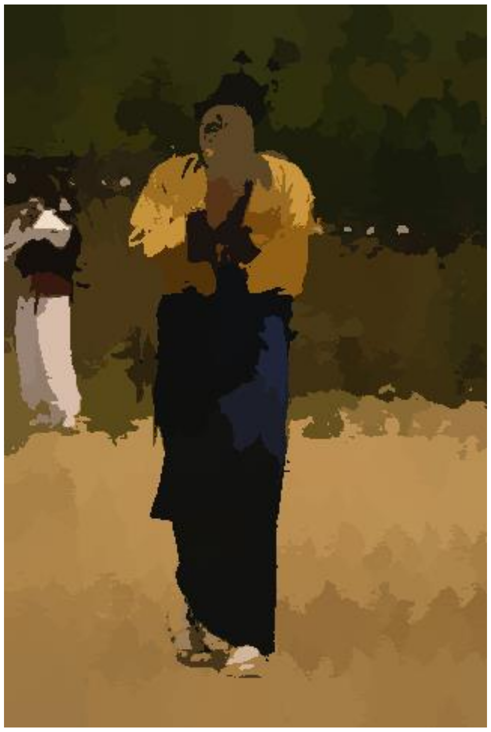

Figure 48 - output of second bigger image,
***r=20**, 5D features, pre-smoothed,*
<u>209 peaks found</u>

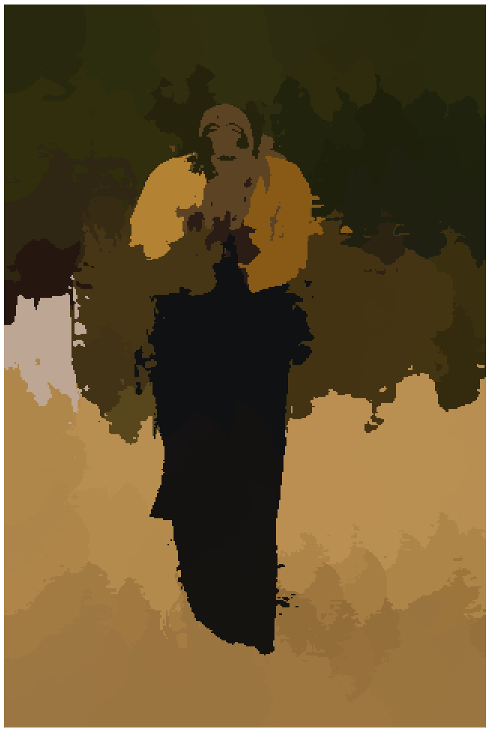

Figure 49 - output of second bigger image,
***r=30**, 5D features, pre-smoothed,*
<u>81 peaks found</u>

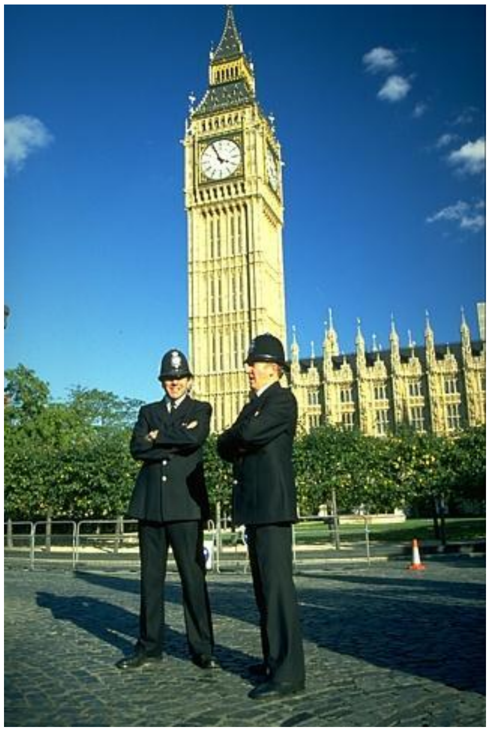

Figure 50 - Original third bigger image submitted for segmentation

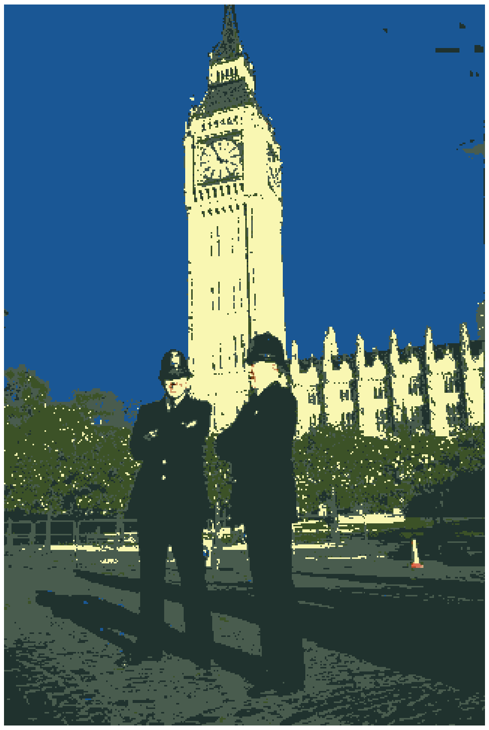
Figure 51 - output of third bigger image,
r=10, 3D features, ***original****,*
<u>15 peaks found</u>

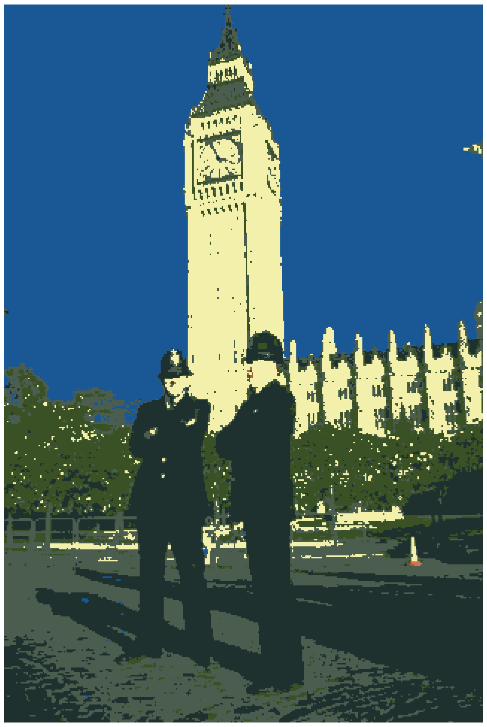
Figure 52 - output of third bigger image,
r=10, 3D features, ***pre-smoothed****,*
<u>12 peaks found</u>

Analogously to the previous set of the experiments, all the output images received for these 3 desired images have been attached to this report in a .zip file and can be found in the directory called *ResultTestBigPictures*. Additionally, this directory includes relevant .txt files (e.g. *Image_1.txt*) containing not only the time elapsed during each segmentation process, but also the number of peaks found for each output image.