Computer Vision

Implementation of the Mean-Shift algorithm for image segmentation and experiments on parameters

Problem Statement

Implementation of the Mean-Shift algorithm to create image segmentation and experiment with parameters and feature space.

1. Implementation of the simple mean-shift algorithm and test it on the given test-data.
2. Optimize the algorithm with two speedup methods and test on the test-data.
3. Read the image and apply the preprocessing steps suggested above.
4. Implement the Mean-shift algorithm, considering the suggested optimizations.
5. Apply the algorithm on the image features, transform the result back to an image and visualize the obtained segmentation.
6. Test different parameters, such as r, c, feature types.
7. Describe the proposed steps for improving the segmentation result and efficiency.

Testing Algorithm on the pts.mat Data

Experiment 0:

Run the following to generate the same result:

```
python main.py --image 'Images\input\img1.jpg' -r 2 -c 4 --feature_type '3D' --down_size_by 1 -experiment 0
```

Comments:

The Optimized algorithm uses two speedup methods:

1. Basin of attraction at the final peak
2. Basin of attraction through search path

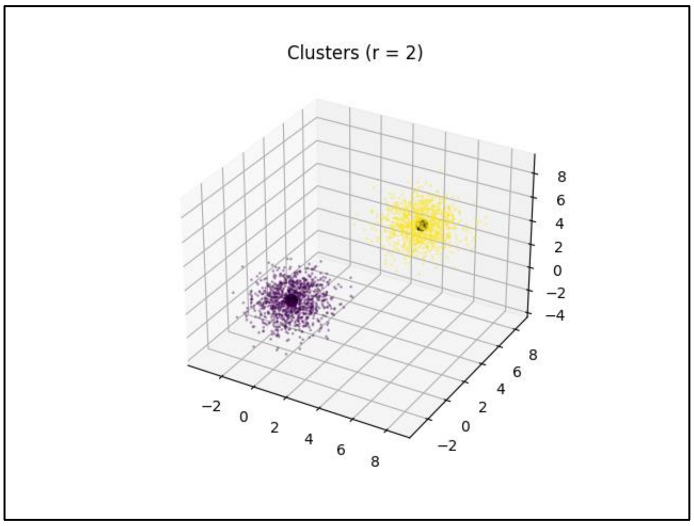


Fig 1: Simple mean-shift algorithm

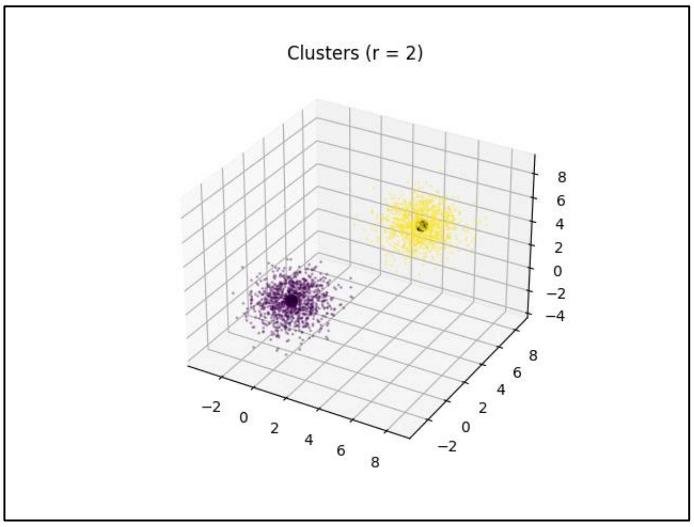


Fig 2: Optimized mean-shift algorithm

The runtime difference between the two versions of the algorithm is:

1. Runtime for simple mean-shift algorithm: 81.45 seconds
2. Runtime for simple mean-shift algorithm: 30.70 seconds

This implies that the speedup optimizations reduce the computational cost by more than half.

* Runtimes are printed after execution

Simple vs Optimized Algorithm on the Images

Experiment 1:

Run the following command to generate similar results:

```
python main.py --image 'Images\input\img1.jpg' -r 10 -c 4 --feature_type '3D' --down_size_by 2 -experiment 1
```

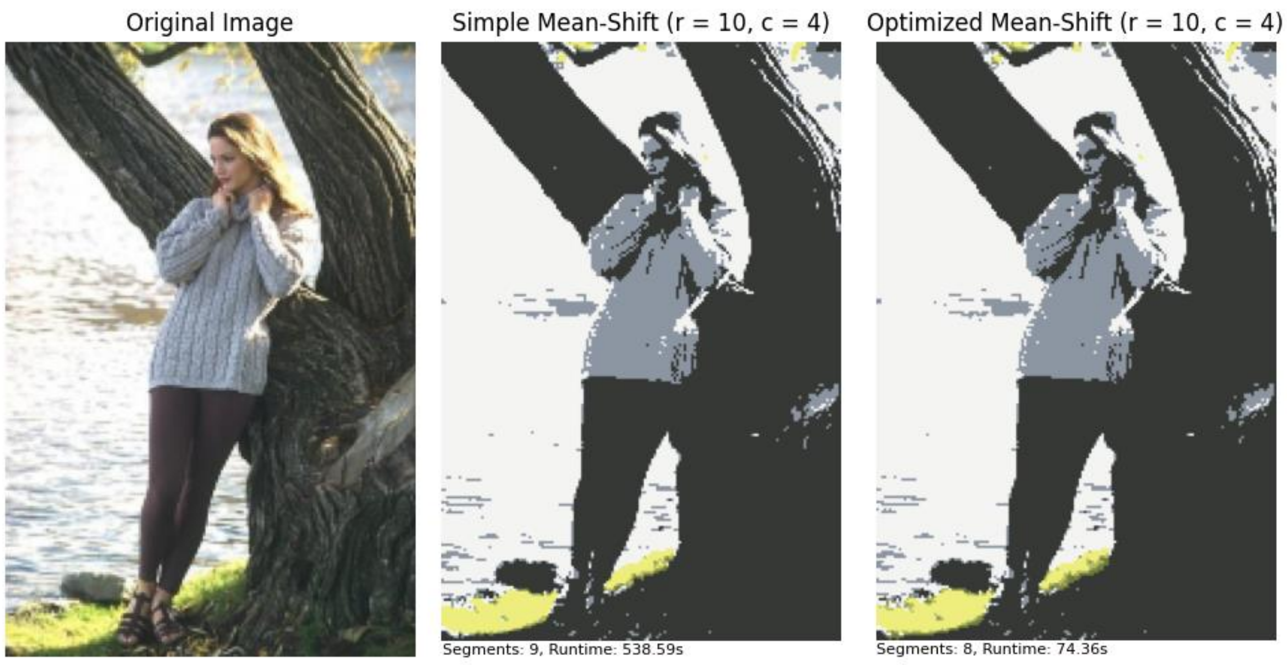
Comments:

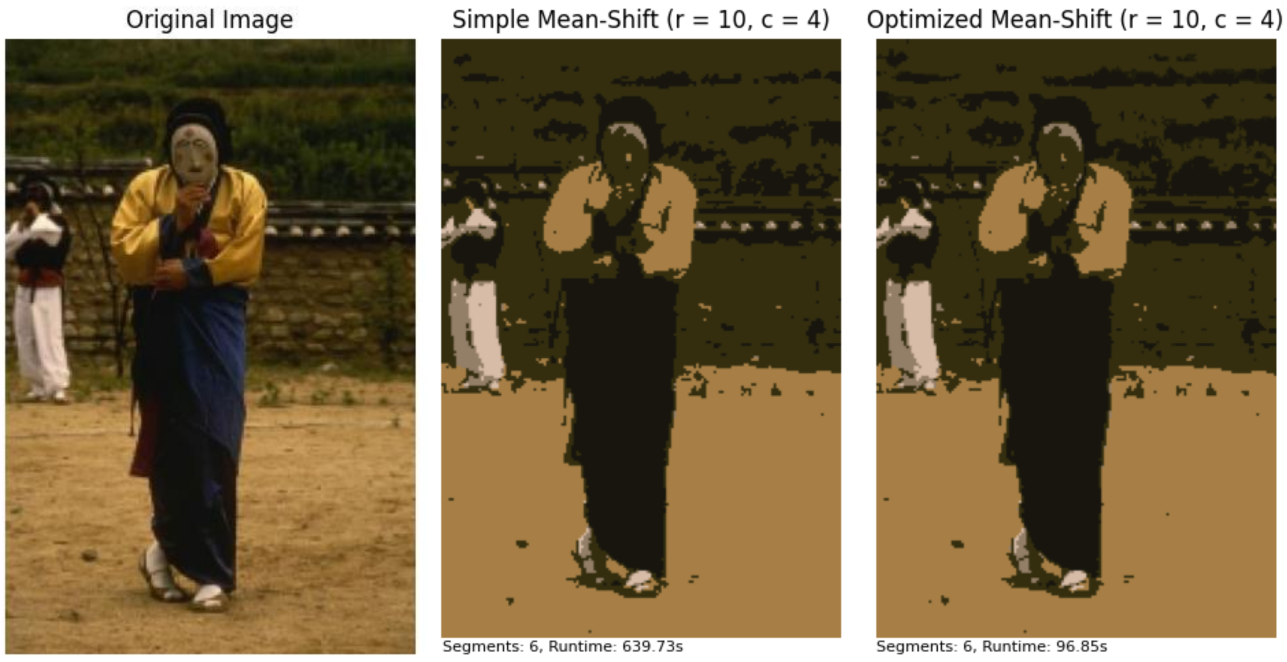
1. The simple algorithm goes through every pixel of the image and assigns a cluster peak label to it. This is computationally very expensive.
2. The optimized algorithm uses two speedup methods to reduce the computational load. The speedups are as follows:
   a. Basin of attraction at the final peak: When the final peak of each pixel is found, all the neighborhood pixels around it within the window with radius r are assigned to that same peak. This reduces the number of pixels to be explored.
   b. Basin of attraction through the search path: When the window shifts from peak to peak while finding the final peak, another window is utilized. This window has a radius of r/c. All the pixels inside this smaller window are assigned to the same peak. This further reduces the number of pixels to be explored.
3. The optimization however comes with a price of wrongly clustering pixels. That is why the value of r is so crucial. I will show the influence of r and c on the segmentation later.
4. The main advantage of the optimization is that the runtime reduces significantly. I will show this in the following page for 3 images. A comparative analysis is shown by implementing the two versions of the algorithm on the images.
5. It is evident that for the same set of parameters, the optimized algorithm takes way less time to execute.

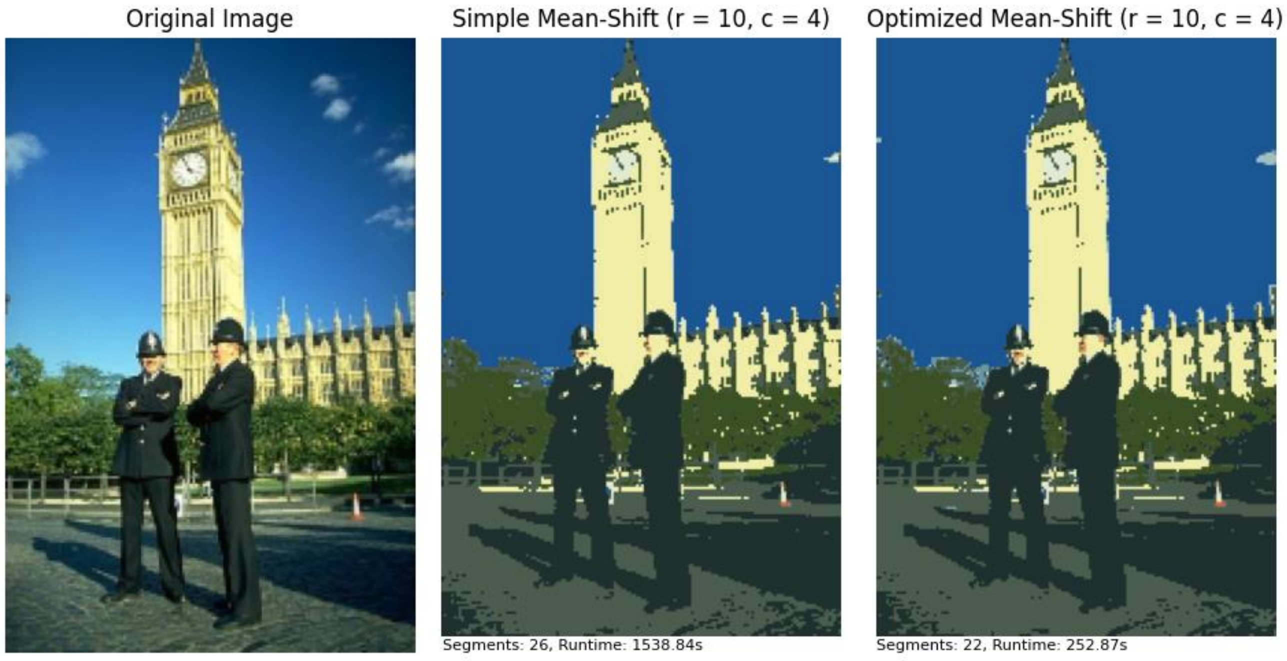
* The best combination of parameters ‘r’ and ‘c’ are found in experiment 2 and experiment 3 below. I am using r = 10 and c = 4 here.

** The images were downscaled to half of the original size to reduce runtime.

*** The number of segments and runtime can be seen under each image.


Original Image
Simple Mean-Shift (r = 10, c = 4)
Optimized Mean-Shift (r = 10, c = 4)
Segments: 9, Runtime: 538.59s
Segments: 8, Runtime: 74.36s


Original Image
Simple Mean-Shift (r = 10, c = 4)
Optimized Mean-Shift (r = 10, c = 4)
Segments: 6, Runtime: 639.73s
Segments: 6, Runtime: 96.85s


Original Image
Simple Mean-Shift (r = 10, c = 4)
Optimized Mean-Shift (r = 10, c = 4)
Segments: 26, Runtime: 1538.84s
Segments: 22, Runtime: 252.87s

Influence of Radius (r)

Experiment 2:

Run the following command to generate the same result (modify the image path if necessary):

```
python main.py --image 'Images\input\img1.jpg' -r 2 -c 4 --feature_type '3D' --down_size_by 2 -experiment 2
```

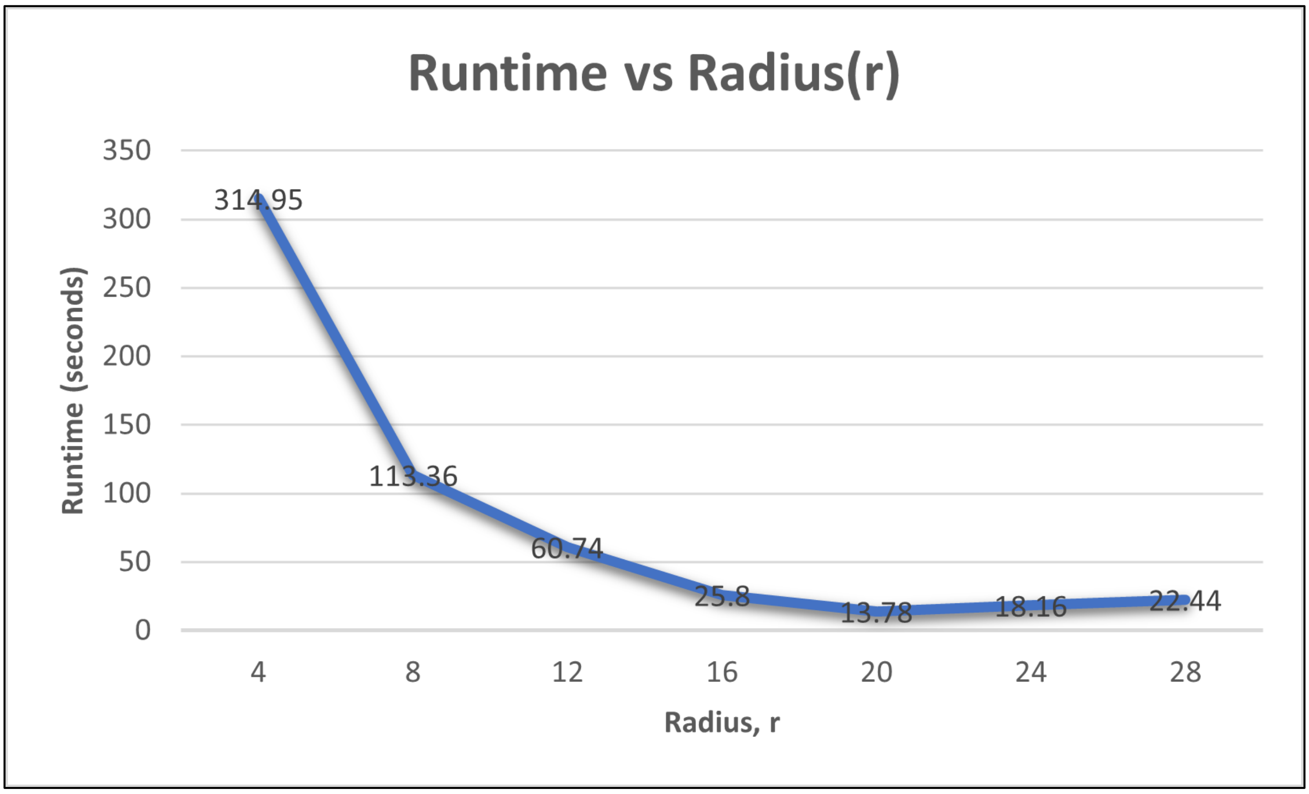
Comments:

1. The radius of the window controls how many neighbors are considered when calculating the mean peak of the pixels.
    a. With smaller radius, the number of peak increases. This implies that more segments are created than necessary.
    b. With higher radius, the number of peak decreases. This implies that many different segments are clustered wrongly as the same segment.
    c. There is a sweet spot between the above two, which gives quite nice segmentation. I will show this influence of r in the following pages for 3 different images.
    d. This inverse relationship between the number of peaks and the radius is also visible in the following graph (Based on the data of image 1).

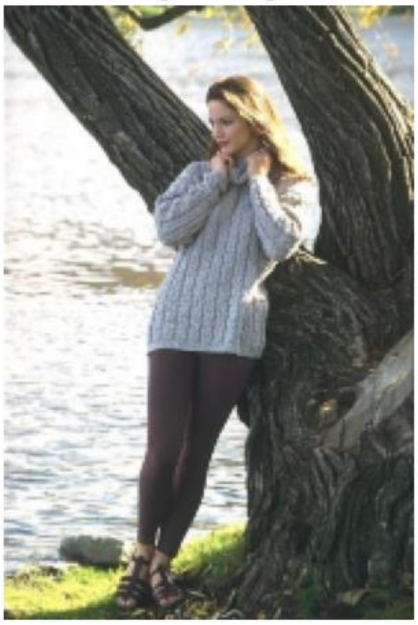
* The images are downsized to half of the original size.

* Parameter c is kept constant at c = 4

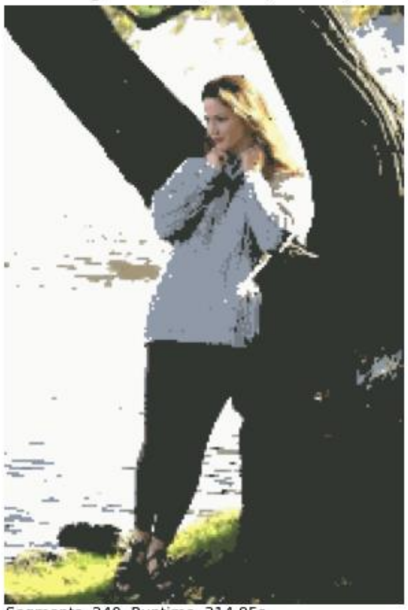
* 3D feature space was used



Original Image

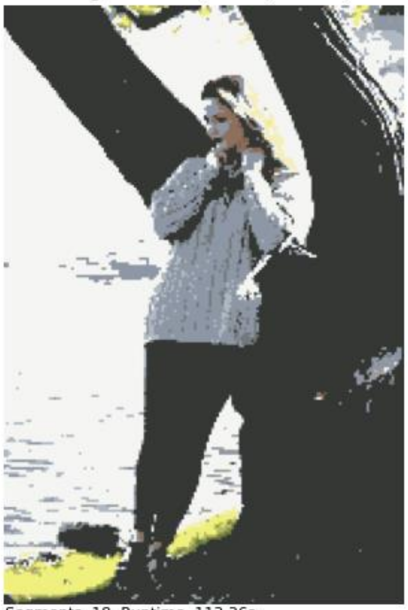


Segmented (r = 4, c = 4)



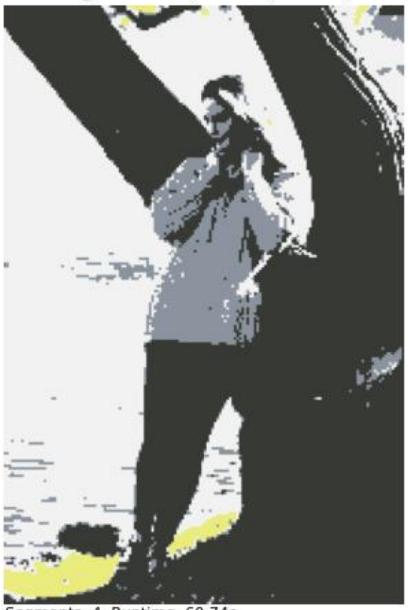
Segments: 240, Runtime: 314.95s

Segmented (r = 8, c = 4)



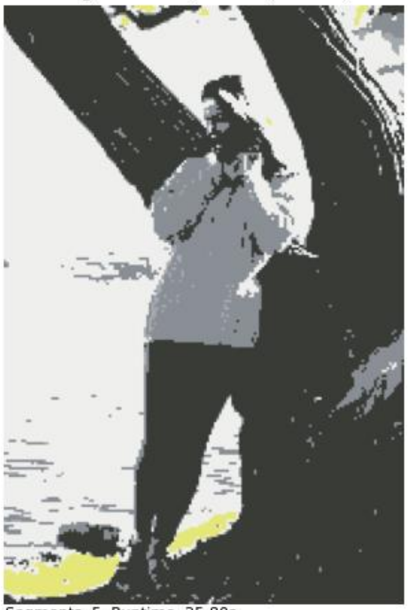
Segments: 18, Runtime: 113.36s

Segmented (r = 12, c = 4)



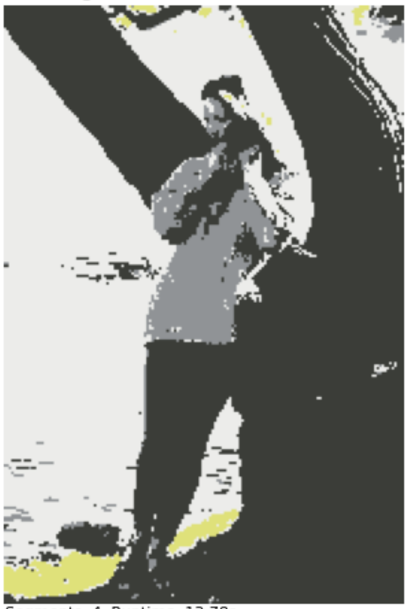
Segments: 4, Runtime: 60.74s

Segmented (r = 16, c = 4)



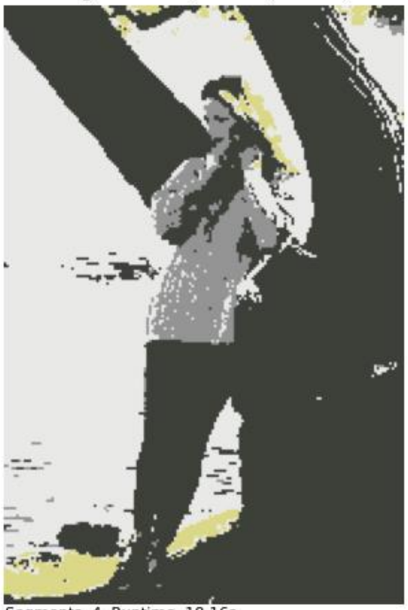
Segments: 5, Runtime: 25.80s

Segmented (r = 20, c = 4)



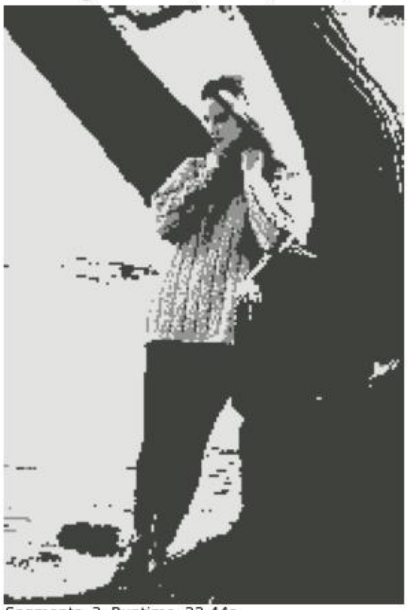
Segments: 4, Runtime: 13.78s

Segmented (r = 24, c = 4)



Segments: 4, Runtime: 18.16s

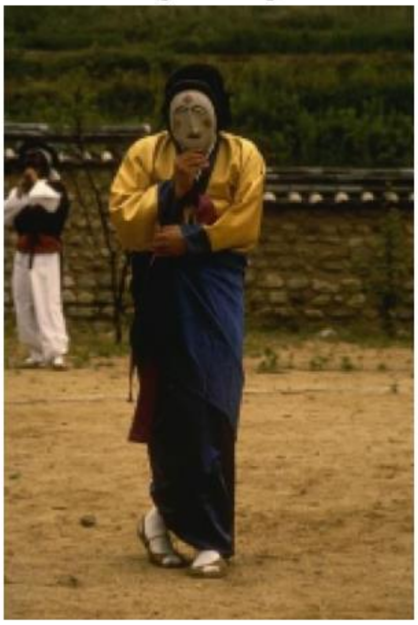
Segmented (r = 28, c = 4)



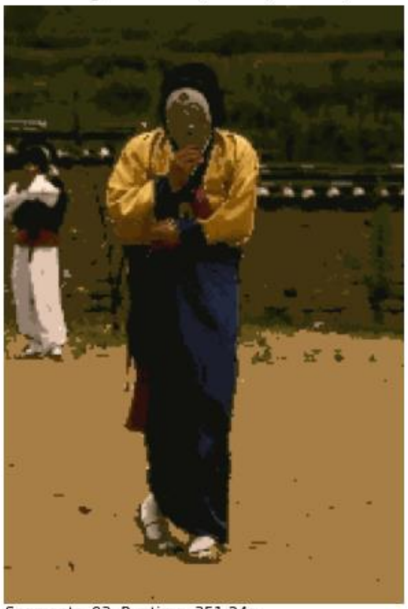
Segments: 3, Runtime: 22.44s

- From visual inspection, it is seen that better segmentation can be found between r = 8 and r = 12. More refined search can be performed in this range.
- I suggest r = 10 for this image.
- As r increases, the runtime decreases.
- Number of segments also decreases with increasing radius.

Original Image

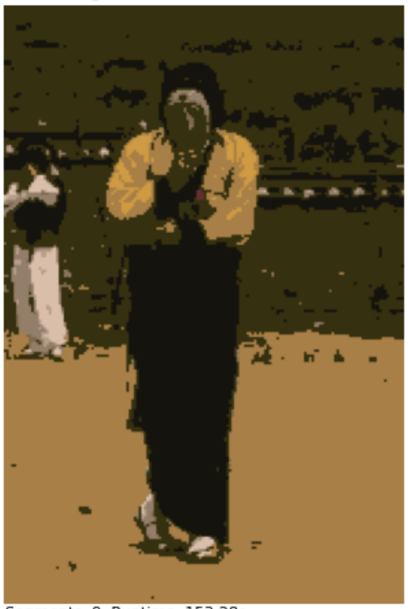


Segmented (r = 4, c = 4)



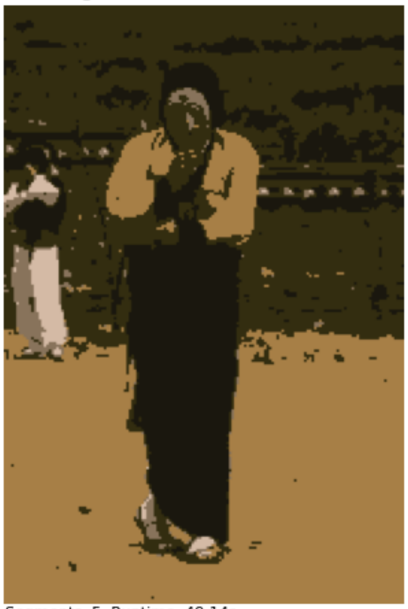
Segments: 93, Runtime: 351.24s

Segmented (r = 8, c = 4)



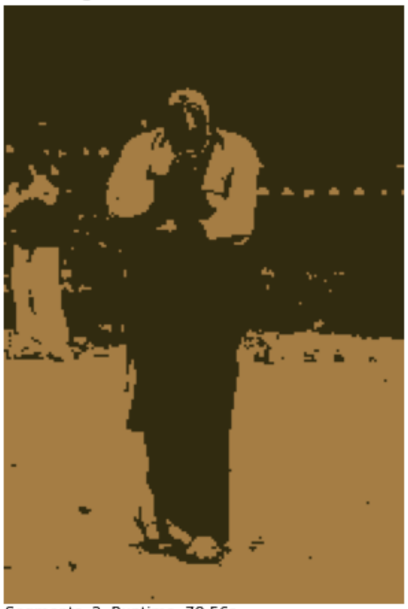
Segments: 9, Runtime: 153.28s

Segmented (r = 12, c = 4)



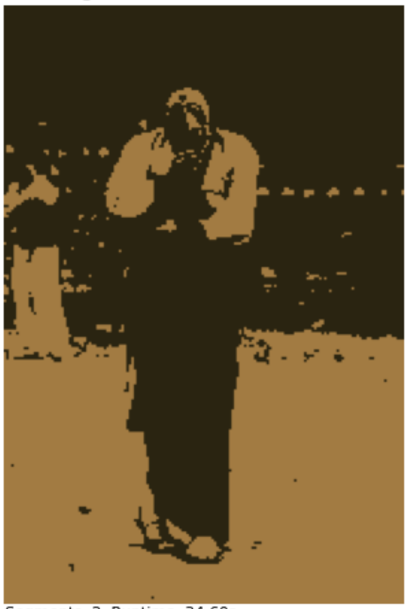
Segments: 5, Runtime: 49.14s

Segmented (r = 16, c = 4)



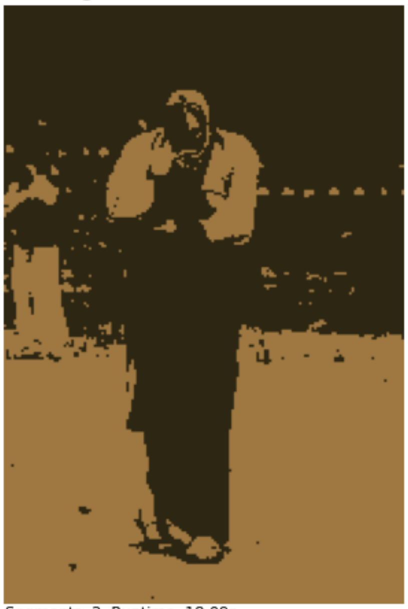
Segments: 2, Runtime: 78.56s

Segmented (r = 20, c = 4)



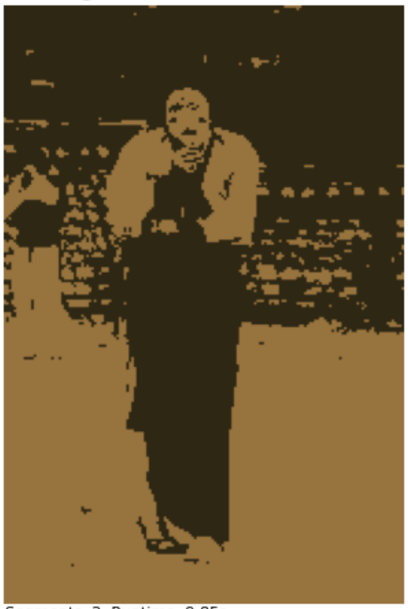
Segments: 2, Runtime: 34.60s

Segmented (r = 24, c = 4)



Segments: 2, Runtime: 18.09s

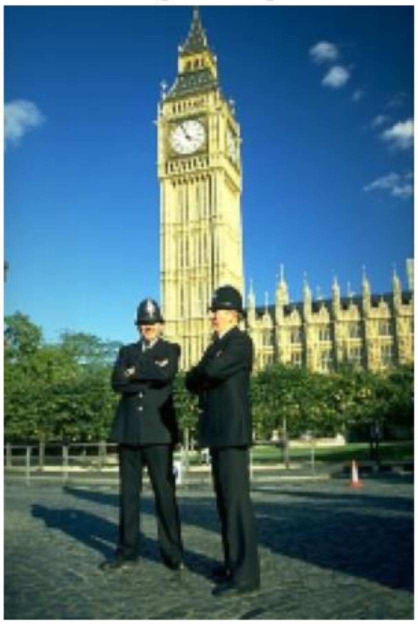
Segmented (r = 28, c = 4)



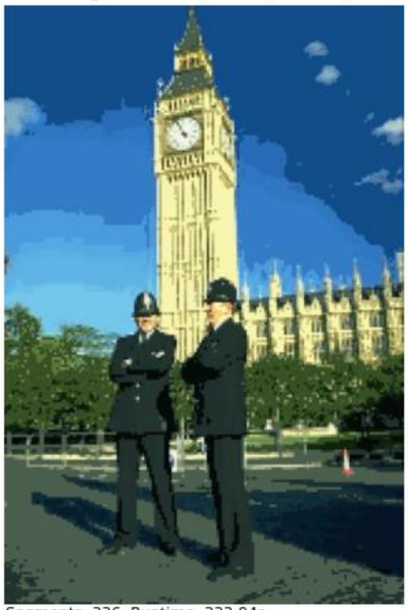
Segments: 2, Runtime: 9.85s

- From visual inspection, it is seen that better segmentation can be found between r = 8 and r = 12. More refined search can be performed in this range.
- I suggest r = 10 for this image.
- As r increases, the runtime decreases.
- Number of segments also decreases with increasing radius.

Original Image

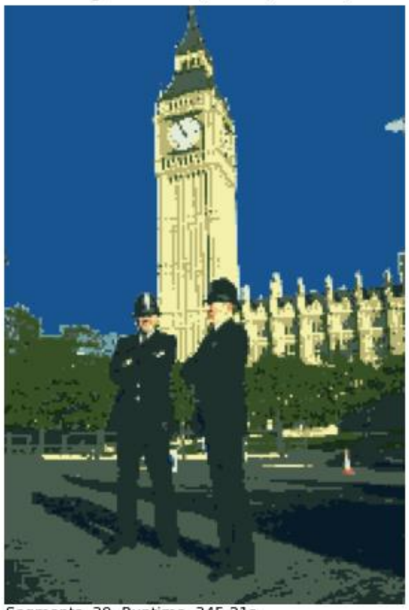


Segmented (r = 4, c = 4)



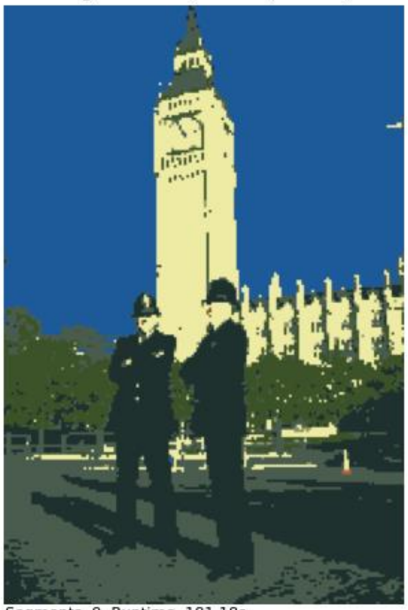
Segments: 336, Runtime: 333.94s

Segmented (r = 8, c = 4)



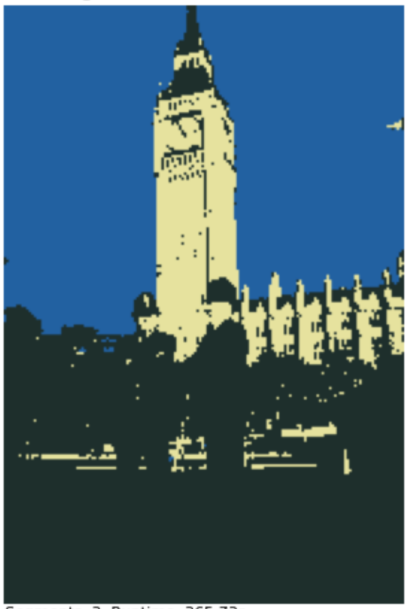
Segments: 39, Runtime: 345.21s

Segmented (r = 12, c = 4)



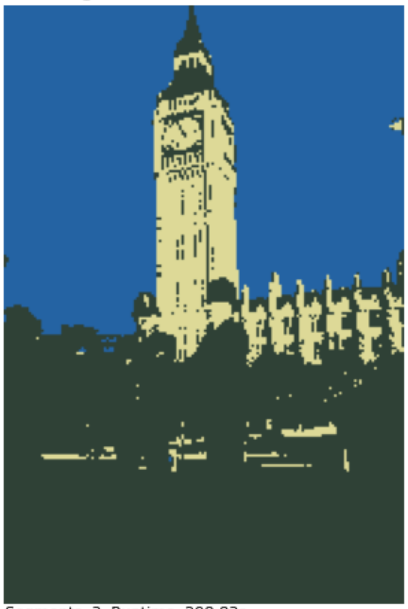
Segments: 9, Runtime: 181.18s

Segmented (r = 16, c = 4)



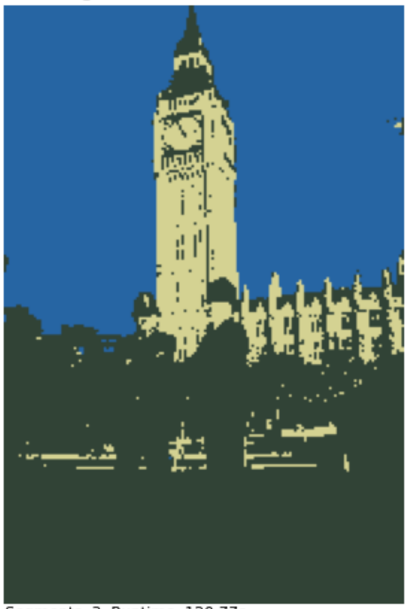
Segments: 3, Runtime: 365.73s

Segmented (r = 20, c = 4)



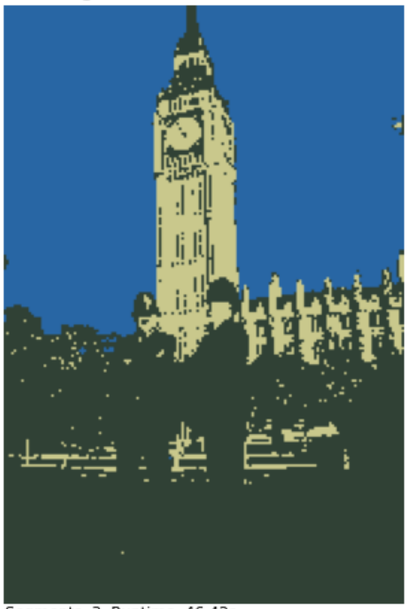
Segments: 3, Runtime: 298.83s

Segmented (r = 24, c = 4)



Segments: 3, Runtime: 120.77s

Segmented (r = 28, c = 4)



Segments: 3, Runtime: 46.42s

- From visual inspection, it is seen that better segmentation can be found between r = 8 and r = 12. More refined search can be performed in this range.
- I suggest r = 10 for this image.
- As r increases, the runtime decreases.
- Number of segments also decreases with increasing radius.

Influence of c

Experiment 3:

Run the following command to generate similar results:

```
python main.py --image 'Images\input\img3.jpg' -r 10 -c 4 --feature_type '3D' --down_size_by 2 -experiment 3
```

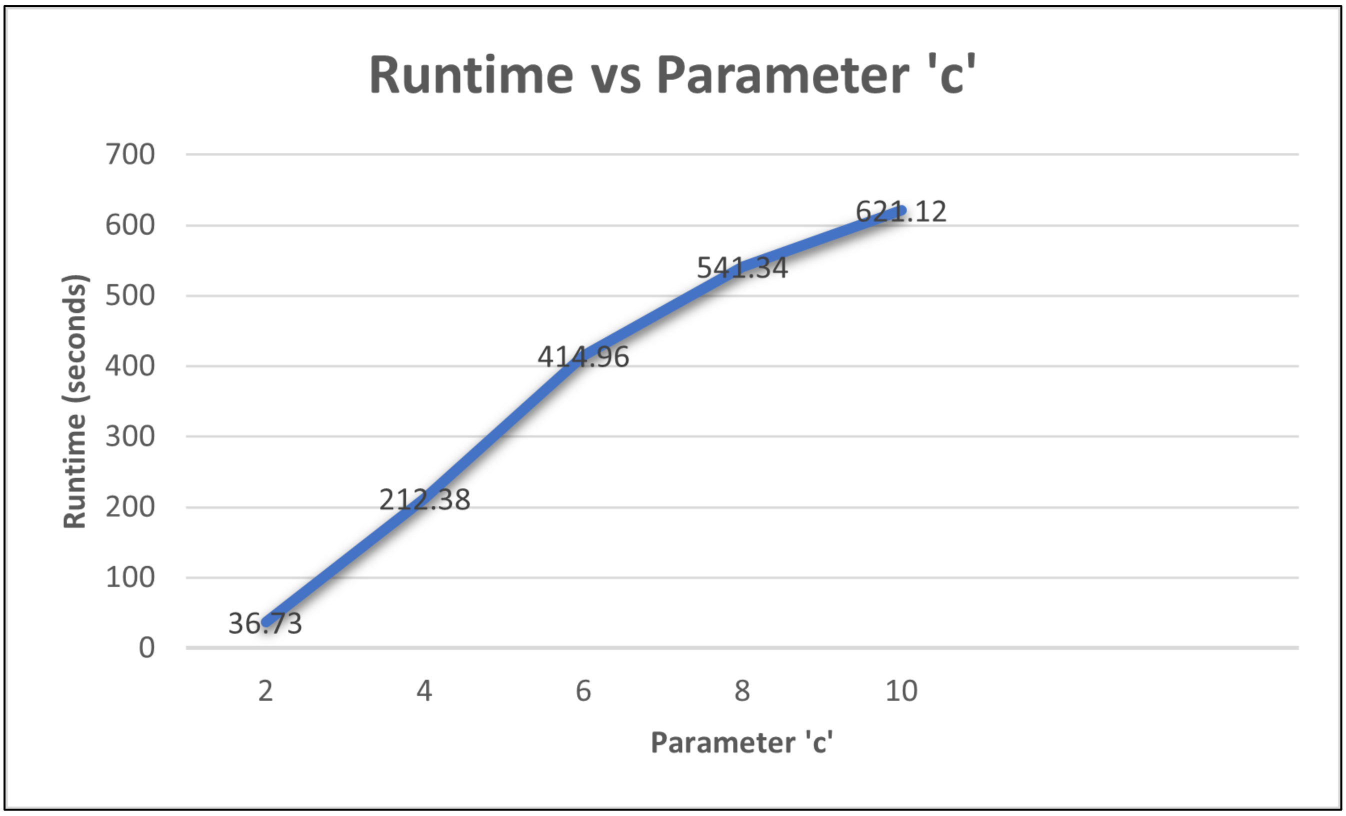
Comments:

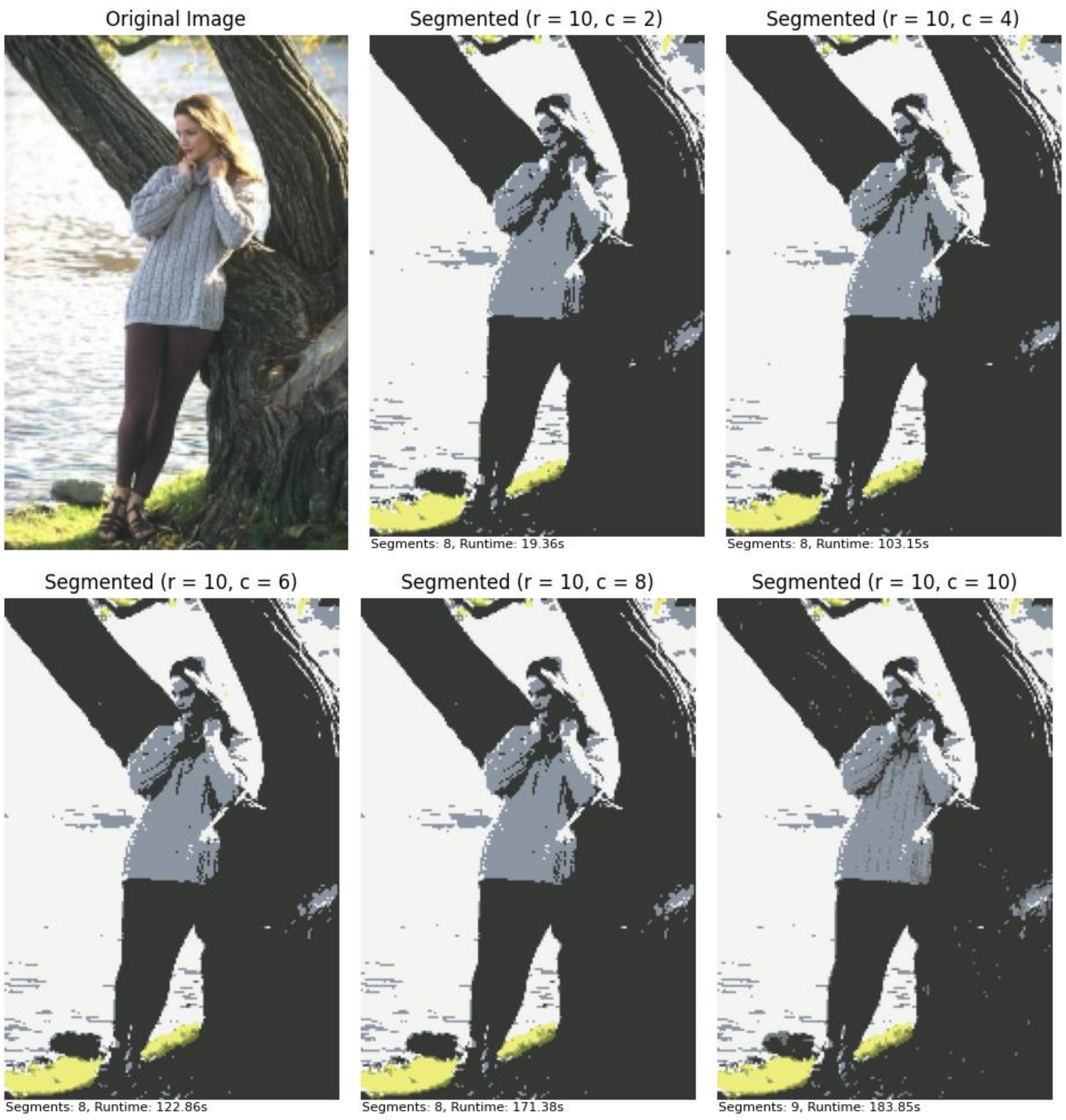
1. The parameter ‘c’ controls the second speedup method. Each time the radius is shifted to a new peak while searching for the final peak of a data point, a window of r / c is utilized. All the points inside this window are assigned to the same peak. This process speeds up the algorithm greatly.
   a. With smaller c, the number of peak decreases. This implies that r/c increases, and more points are assigned to the same peak during search path speedup.
   b. With higher c, the number of peak increases. This implies that r/c decreases, and less points are assigned to the same peak during search path speedup.
   c. The influence of c is not so prominent as the influence of r. However, c limits the window size and thus reduces the risk of wrong clustering. I will show this influence of c in the following pages for 3 different images.
   d. This proportional relationship between the number of peaks and c is also visible in the following graph (Based on the data of image 3).

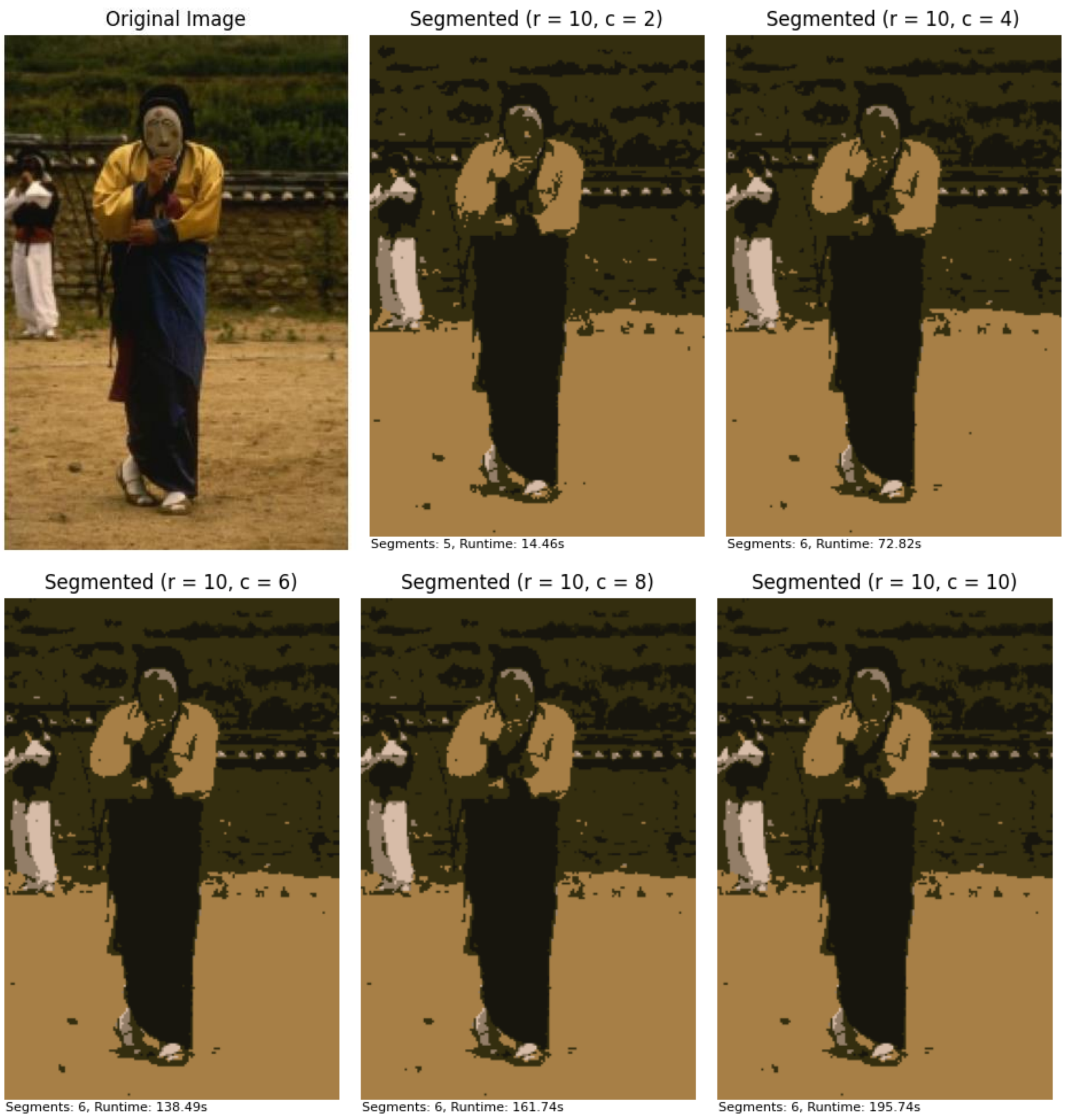
* The images are downsized to half of the original size.

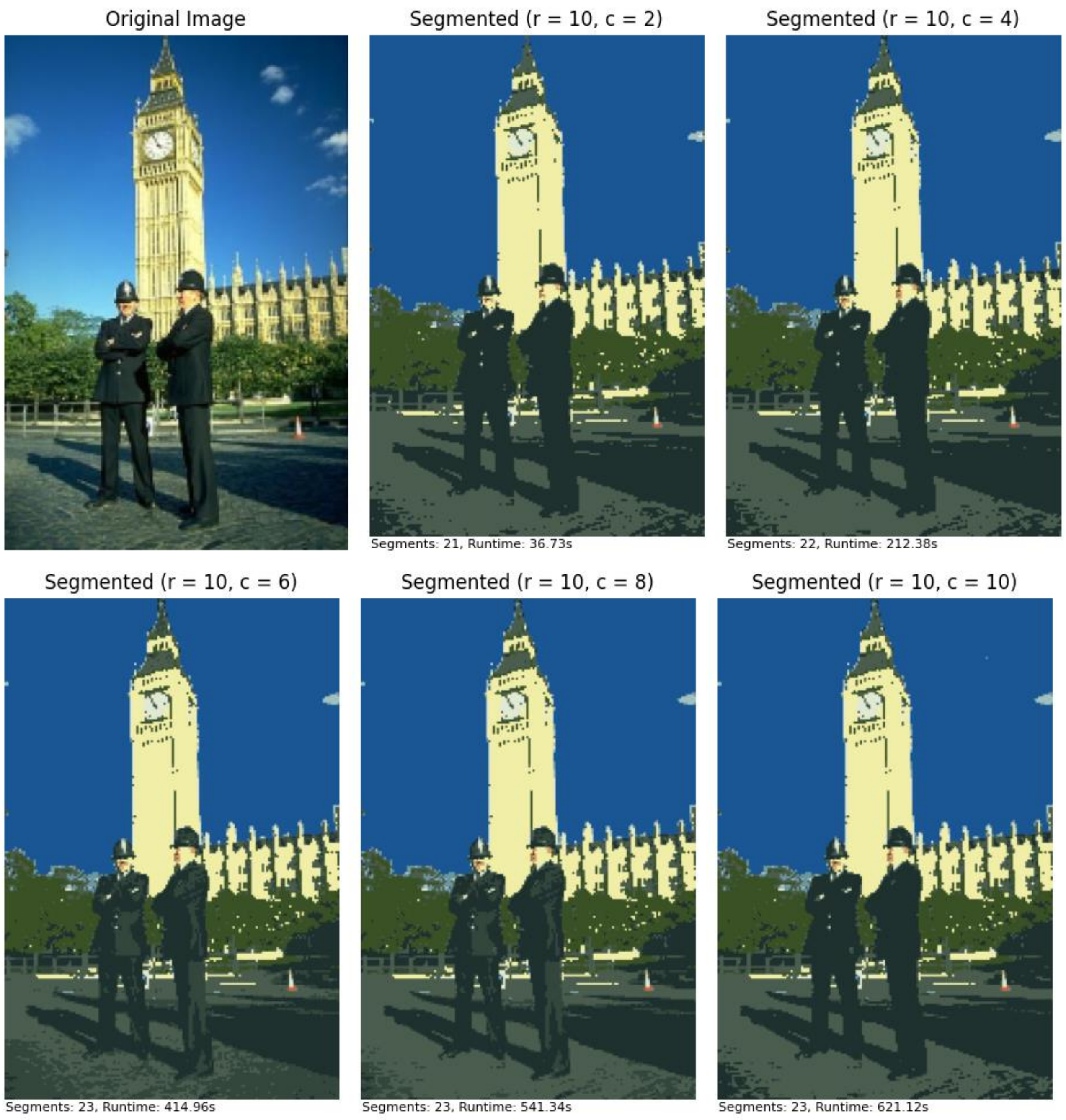
* Parameter r is kept constant at r = 10

* 3D feature space was used




Original Image
Segmented (r = 10, c = 2)
Segments: 8, Runtime: 19.36s
Segmented (r = 10, c = 4)
Segments: 8, Runtime: 103.15s
Segmented (r = 10, c = 6)
Segments: 8, Runtime: 122.86s
Segmented (r = 10, c = 8)
Segments: 8, Runtime: 171.38s
Segmented (r = 10, c = 10)
Segments: 9, Runtime: 183.85s


Original Image
Segmented (r = 10, c = 2)
Segmented (r = 10, c = 4)
Segments: 5, Runtime: 14.46s
Segments: 6, Runtime: 72.82s
Segmented (r = 10, c = 6)
Segmented (r = 10, c = 8)
Segmented (r = 10, c = 10)
Segments: 6, Runtime: 138.49s
Segments: 6, Runtime: 161.74s
Segments: 6, Runtime: 195.74s


Original Image
Segmented (r = 10, c = 2)
Segments: 21, Runtime: 36.73s
Segmented (r = 10, c = 4)
Segments: 22, Runtime: 212.38s
Segmented (r = 10, c = 6)
Segments: 23, Runtime: 414.96s
Segmented (r = 10, c = 8)
Segments: 23, Runtime: 541.34s
Segmented (r = 10, c = 10)
Segments: 23, Runtime: 621.12s

Influence of Feature Space (3D vs 5D)

Experiment 4:

Run the following command to use 3D or 5D feature_type:

```
python main.py --image 'Images\input\img1.jpg' -r 10 -c 2 --feature_type '5D' --down_size_by 2 -experiment 4
```

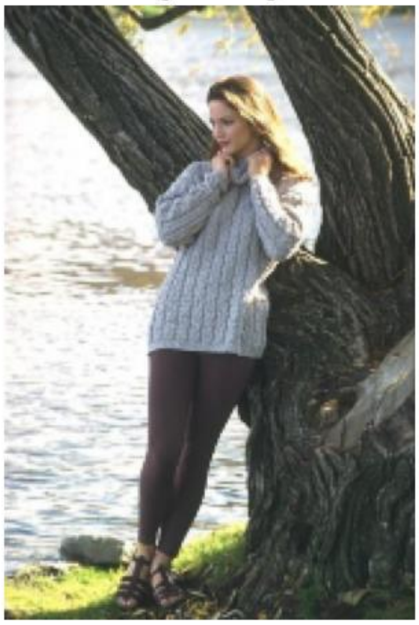
Comments:

3D: Simple RGB images. Each color channel is a feature.

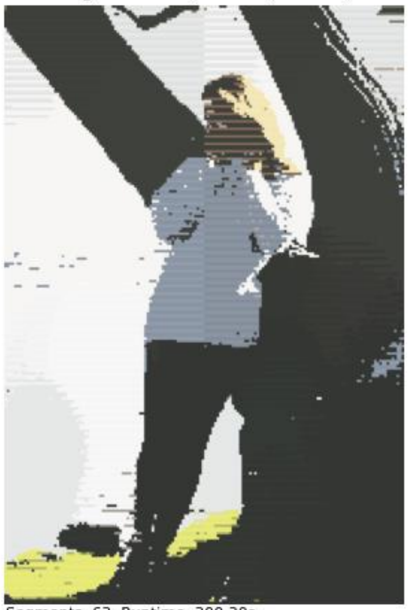
5D: Spatial positions (x, y) of each pixel are added as additional features.

1. First, the best value of r is explored for the 5D feature type.
2. Then, a comparison between the best cases of the 3D and 5D are shown.
3. It is seen that for 5D, higher r value is required to generate better segmentation. Similarly, it can be derived that lower c value is required. c = 2 was used in this case.
4. In the 5D images horizontal lines can be seen. This is because the positional features are created by repeating the 1D array from the pixels. This creates the pattern.

Original Image

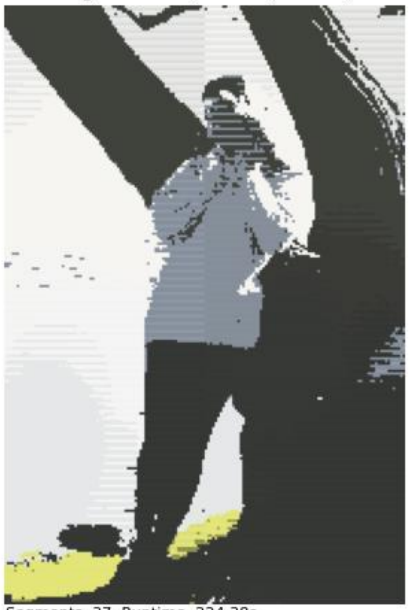


Segmented (r = 20, c = 2)



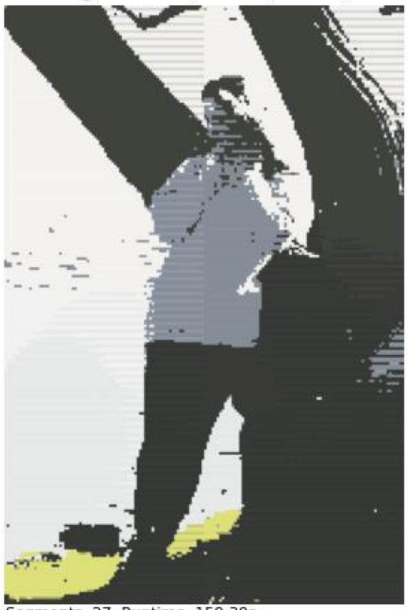
Segments: 63, Runtime: 300.30s

Segmented (r = 24, c = 2)



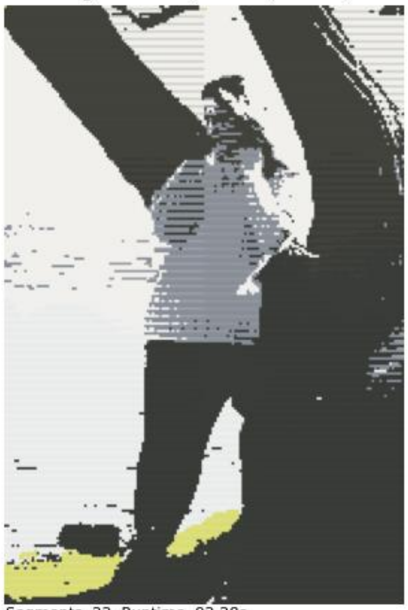
Segments: 37, Runtime: 224.38s

Segmented (r = 28, c = 2)



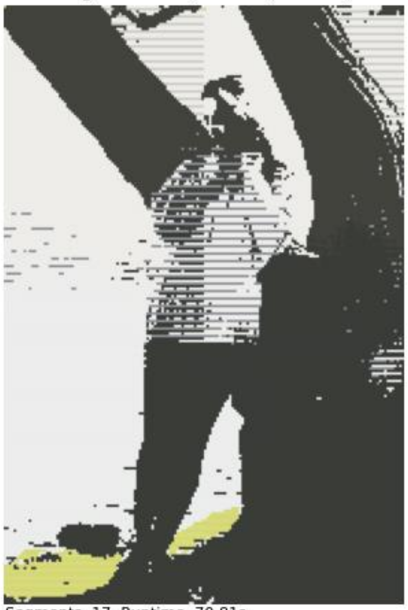
Segments: 27, Runtime: 159.39s

Segmented (r = 32, c = 2)



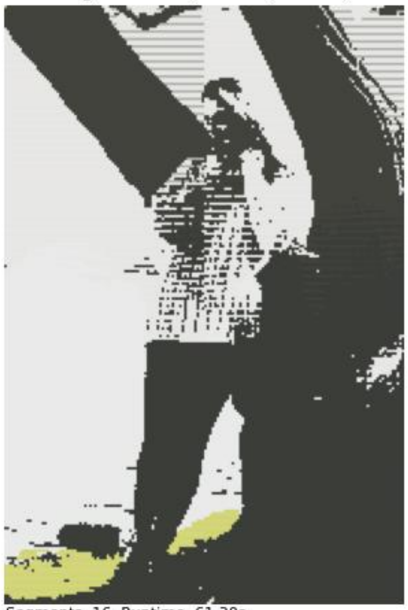
Segments: 22, Runtime: 92.28s

Segmented (r = 36, c = 2)



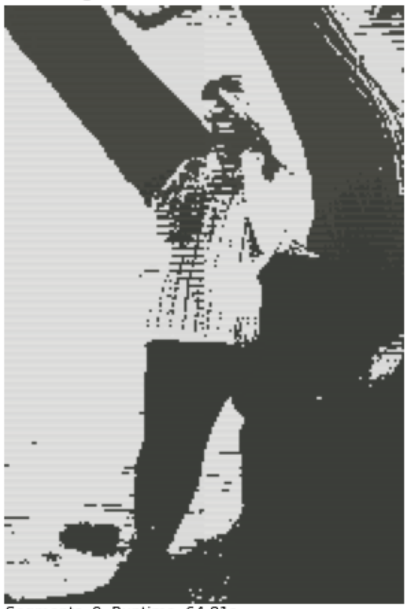
Segments: 17, Runtime: 70.81s

Segmented (r = 40, c = 2)



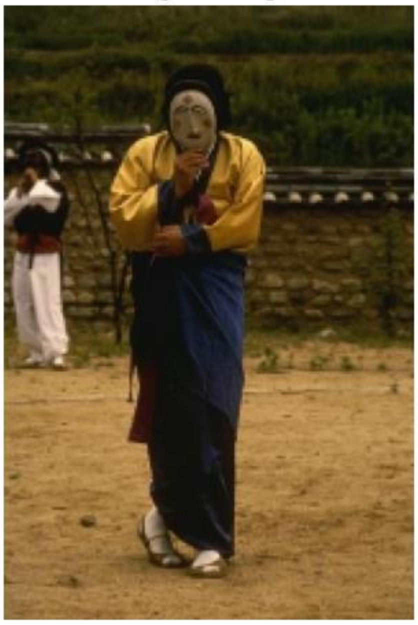
Segments: 16, Runtime: 61.30s

Segmented (r = 44, c = 2)

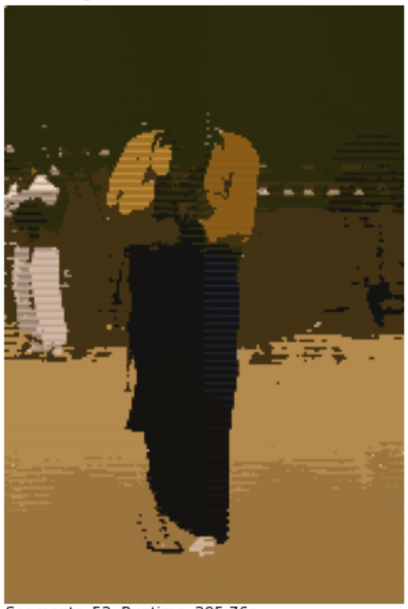


Segments: 9, Runtime: 64.81s

Original Image

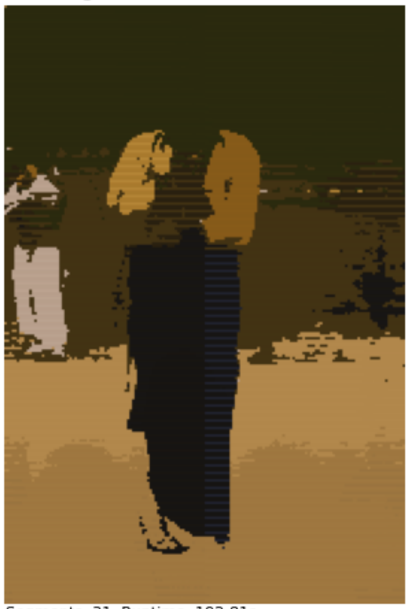


Segmented (r = 20, c = 2)



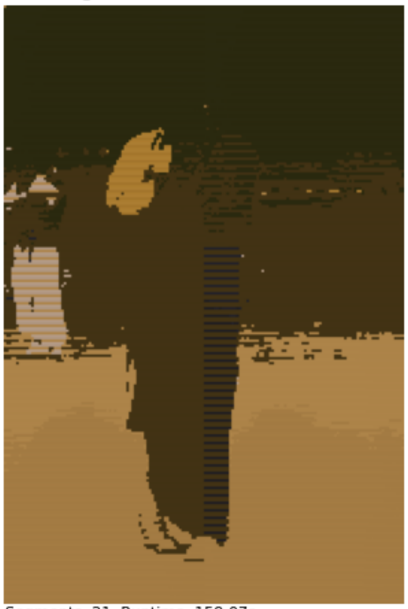
Segments: 52, Runtime: 295.76s

Segmented (r = 24, c = 2)



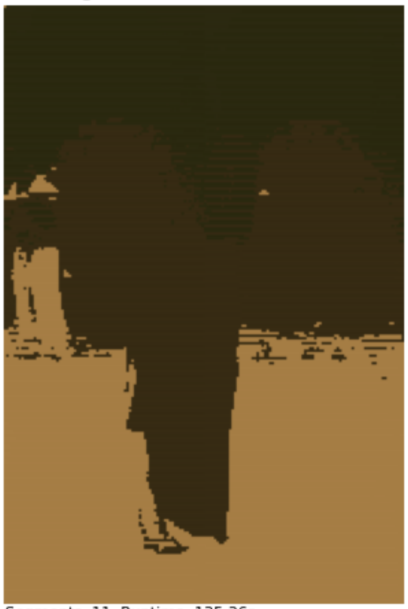
Segments: 31, Runtime: 192.81s

Segmented (r = 28, c = 2)



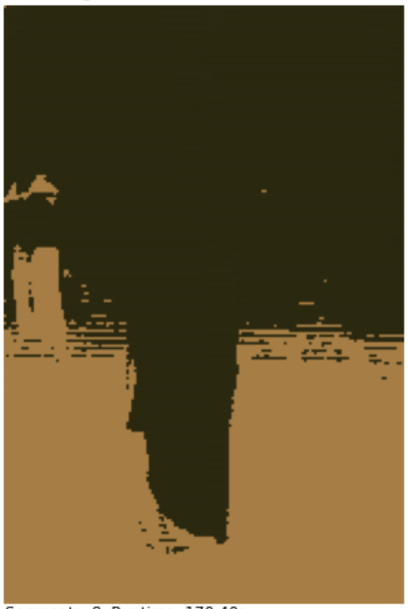
Segments: 21, Runtime: 159.07s

Segmented (r = 32, c = 2)



Segments: 11, Runtime: 135.36s

Segmented (r = 36, c = 2)

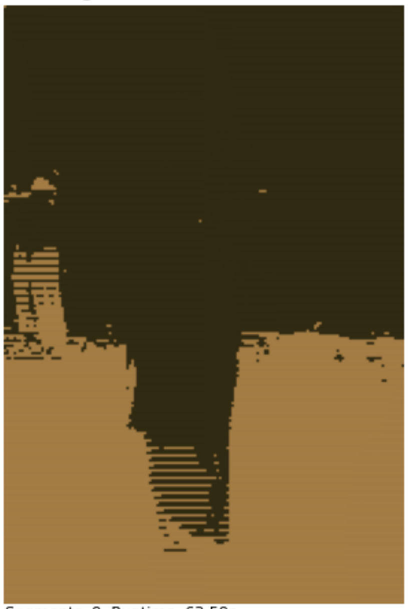


Segments: 8, Runtime: 170.40s

Segmented (r = 40, c = 2)

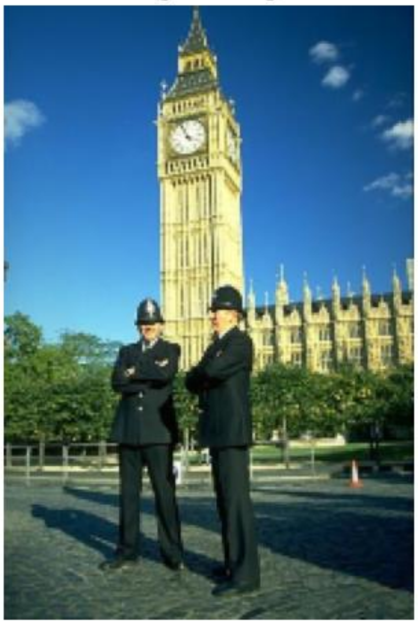
Segments: 7, Runtime: 101.44s

Segmented (r = 44, c = 2)

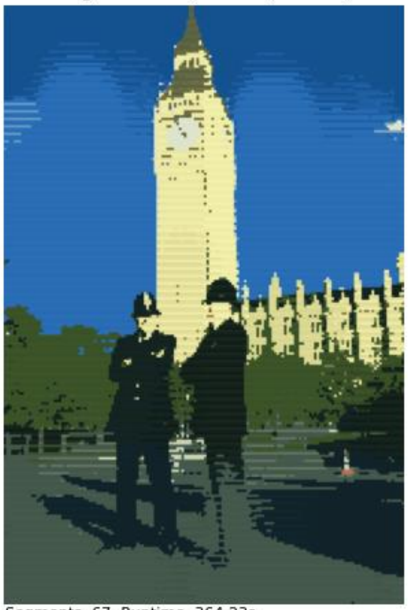


Segments: 9, Runtime: 63.59s

Original Image

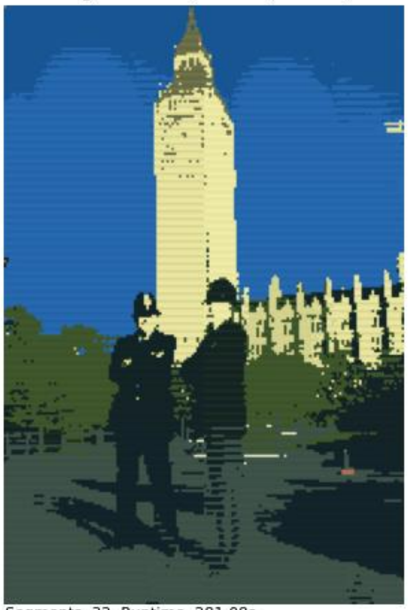


Segmented (r = 20, c = 2)



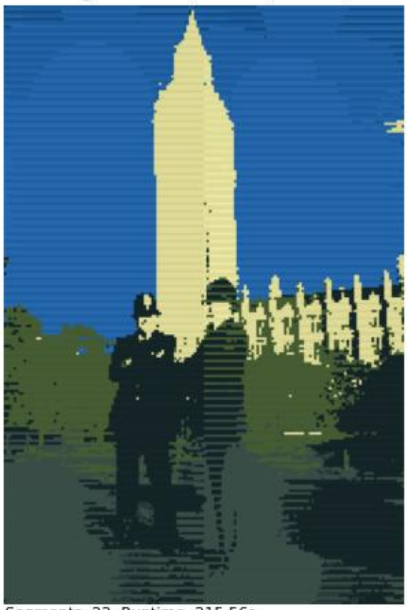
Segments: 67, Runtime: 364.23s

Segmented (r = 24, c = 2)



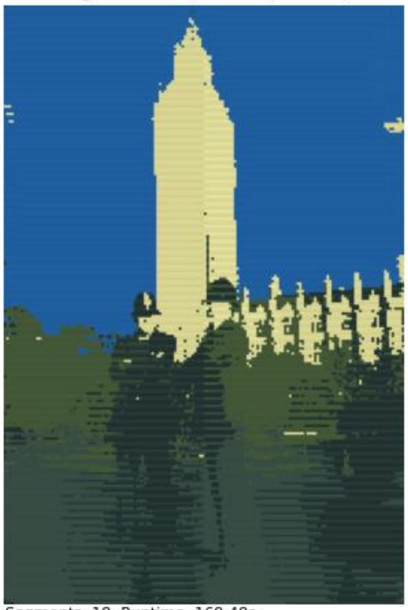
Segments: 32, Runtime: 281.08s

Segmented (r = 28, c = 2)



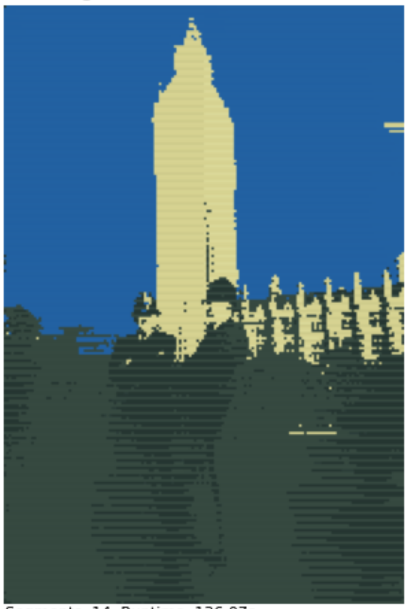
Segments: 22, Runtime: 215.56s

Segmented (r = 32, c = 2)



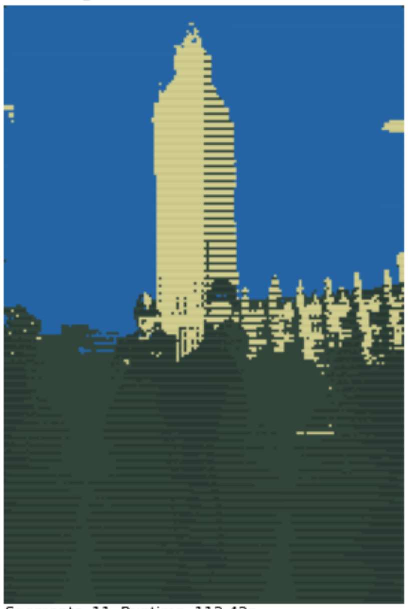
Segments: 18, Runtime: 169.48s

Segmented (r = 36, c = 2)



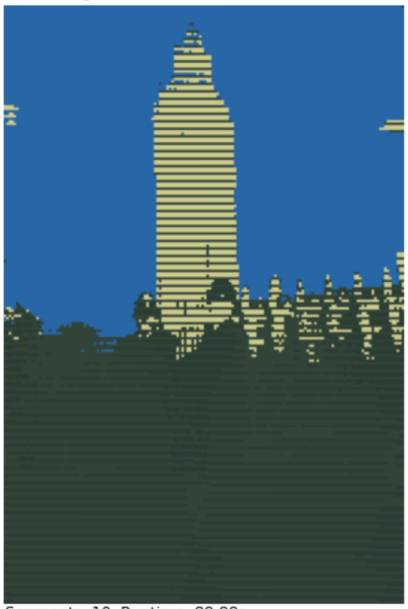
Segments: 14, Runtime: 136.07s

Segmented (r = 40, c = 2)



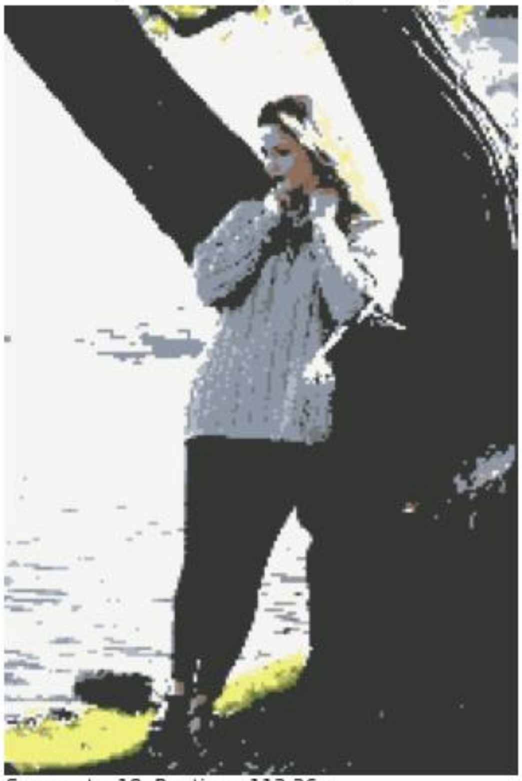
Segments: 11, Runtime: 112.42s

Segmented (r = 44, c = 2)



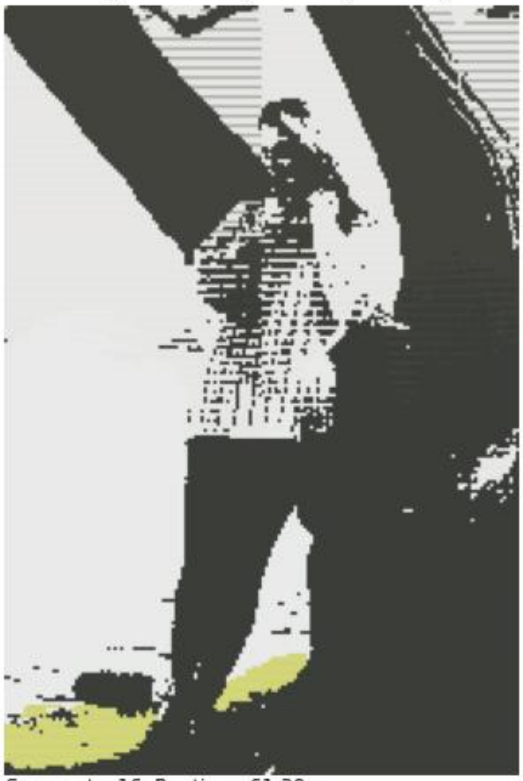
Segments: 10, Runtime: 89.99s

Segmented (r = 8, c = 4)



Segments: 18, Runtime: 113.36s

Segmented (r = 40, c = 2)



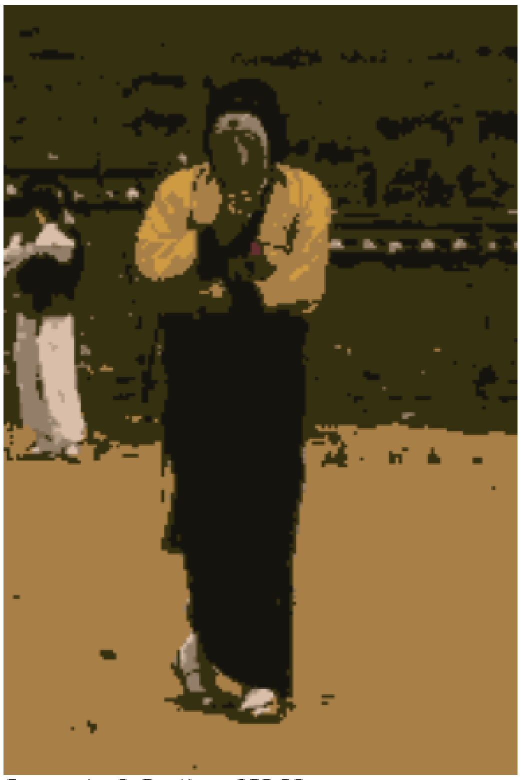
Segments: 16, Runtime: 61.30s

<u>3D vs 5D</u>

Left: 3D

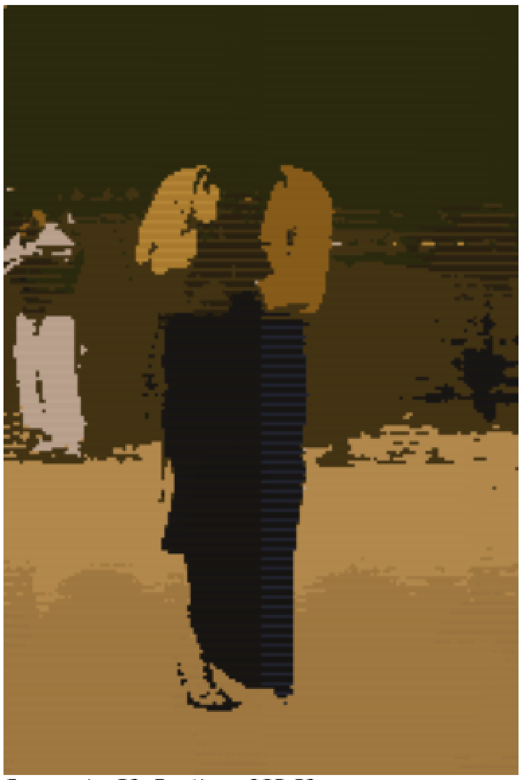
Right: 5D

Segmented (r = 8, c = 4)



Segments: 9, Runtime: 153.28s

Segmented (r = 24, c = 2)

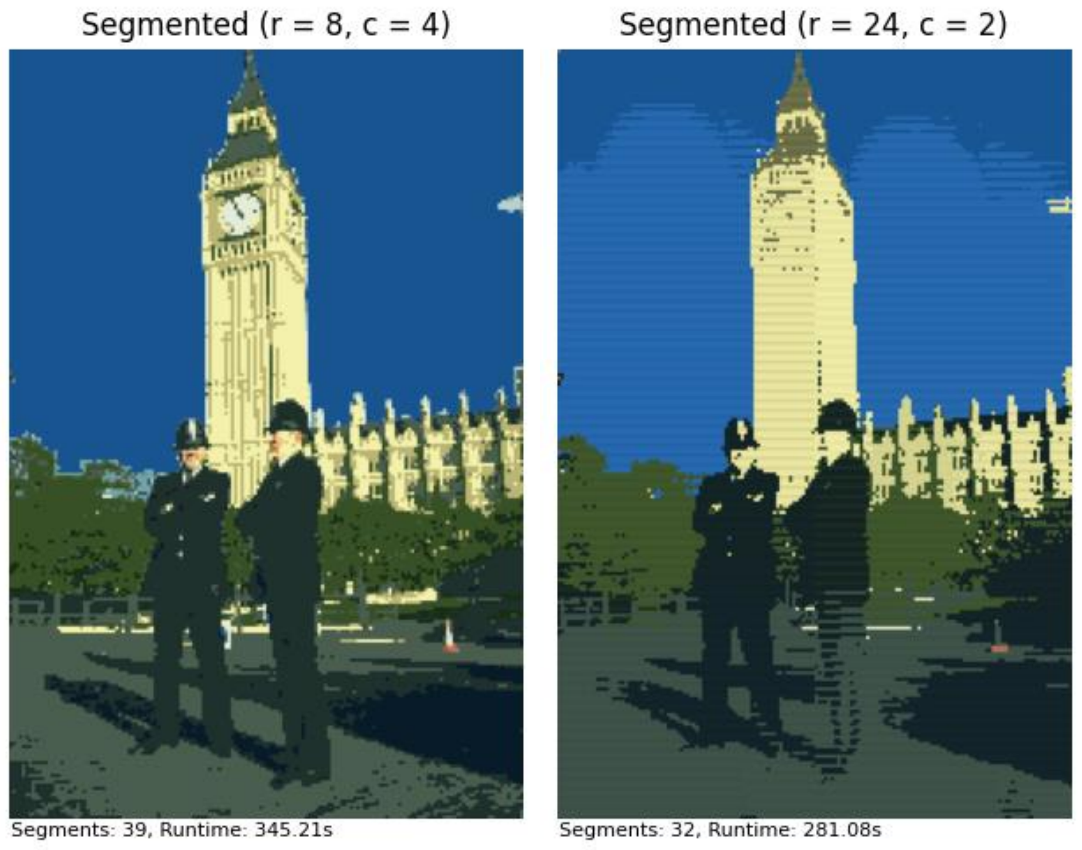


Segments: 31, Runtime: 192.81s

<u>3D vs 5D</u>

Left: 3D

Right: 5D


Segmented (r = 8, c = 4)
Segments: 39, Runtime: 345.21s
Segmented (r = 24, c = 2)
Segments: 32, Runtime: 281.08s

<u>3D vs 5D</u>

Left: 3D

Right: 5D

Proposal on Processing

Some additional processing steps can be applied to improve the segmentation results in image segmentation using the mean-shift algorithm:

1. Pre-processing (Image Enhancement):

We are using colors to create the segments. So, contrast enhancement, histogram equalization, or adaptive histogram equalization can increase the contrast between different regions of the image. This can enhance the features by making the image more detailed.

2. Filtering:

Noise can be clustered as separate segments. This can create discontinuation in a segment. So, image filtering techniques, such as Gaussian smoothing or median filtering, can be used to reduce noise. Filtering can also help remove outliers.

3. Adaptive Parameter Selection:

Different parameter combinations can work better with different images. So, an adaptive parameter search based on image statistics can help choose the best case for that image. The experiments done here are an example of adaptive parameter selection. I experimented with three parameters mostly: radius(r) of the window, parameter ‘c’ to control second speedup, and feature space (3D and 5D).

4. Post-processing (Refinement and Smoothing):

Post-processing techniques such as morphological operations (e.g., erosion, dilation, opening, closing) can be used to refine the segmented images. This can help refine the segmentation boundaries and eliminate small, isolated regions or noise.

5. Incorporate Prior Knowledge:

Prior knowledge or additional information about the objects or regions of interest in the image can be helpful. This can guide the parameter selection and improve the accuracy than random selection of parameters.

The effectiveness of these processing steps can vary depending on the specific characteristics of the images and the application requirements.