Computer Vision

Assignment Report 1: Implementation of Mean Shift Algorithm

Maastricht University

Department of Data Science and Knowledge Engineering

Author

Giorgos Patsiaouras, i6198785

1 Code implementation

The code was implemented using python3 and the libraries skimage, scipy, matplotlib, and progressbar. The code is documented line by line and for every function. In short words the flow followed can be summarized as:

1. Read the image
2. Convert the image to 3d or 5d numerical array
3. Initiate a MeanShiftSegmentation object with custom parameters
4. Call Mean shift function to iterate over points in the image
5. Find the peak for each point and label the point.
6. During the finding peak process, label the points inside the path of the movement of the sphere, in a certain radius (second optimization)
7. When found the peak, label all points inside the current sphere with this label. (first optimization)
8. Create a zeroed numerical array to display the segments of the image.
9. Find the segments from the labels vector and color each segment by the color of the peak.
10. Save and display the picture.

The algorithm was created in a way that the system would be parameterizable. The parameters that can be specified for each execution are:

- Image file path
- Radius
- Parameter c
- 3D or 5D

2 Findings

2.1 With and without optimization in Test data

To debug the algorithm the test dataset provided with the assignment description was used. According to the instructions my algorithm should detect two peaks when using a radius of 2 and c value of 4 (for the optimized algorithm). After executing both the non optimized and optimized versions the results can be displayed on the table below:

Algorithm	Peaks detected	Time findPeaks called	Execution time (sec)
Non Optimized	2	100%	2.14
Both speedups	2	25.30%	0.44

Table 2.1: Execution specifications for mean shift

Time findPeaks called represents for how many points the algorithm tried to find it's peak. The speedup applied is saving almost 75% of the calles to findPeak.

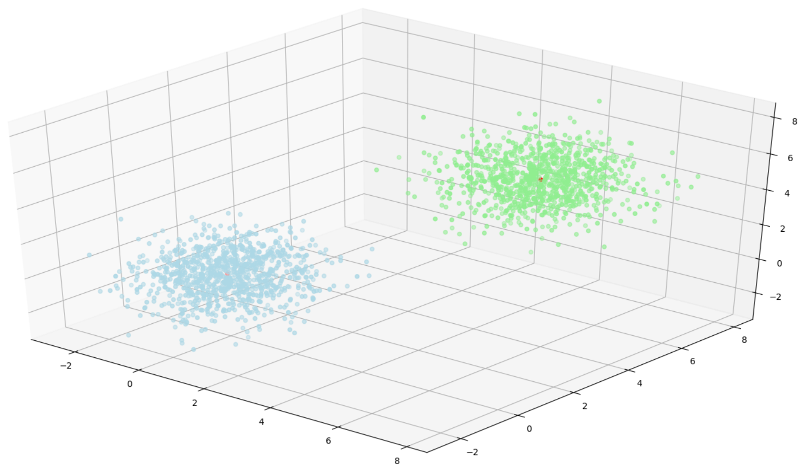

Figure 2.1: Segmentation performed on test dataset

2.2 Effect of radius in the cluster finding

The first experiment that I ran, was to perform the segmentation with different radius values. Radius is used to define the sphere of points around another point. The distance measurement used was, as instructed by the mean shift algorithm,the euclidean distance. Bigger distances imply that the mean of this sphere will be calculated, in more diverged data, thus the segments will be less in number. Lower/smaller sphere would suggest more peaks.

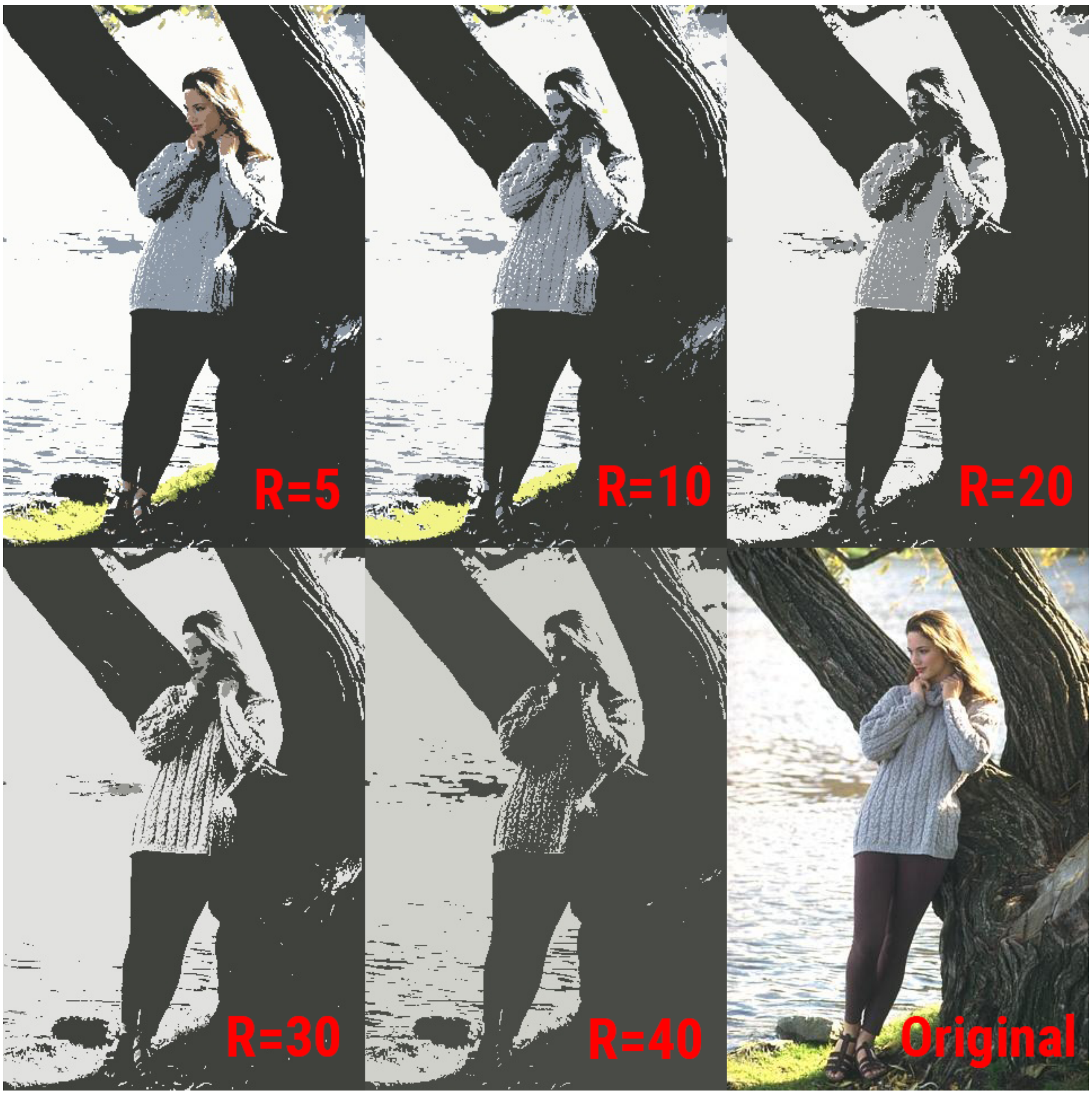

Figure 2.2: Effect of Radius changes in peaks finding. c=4, 3D

In figure 2.2 we can see the effect of the segmentation performed. With radius 5 the algorithm is detecting a high number of peaks. It can still segment big areas of the same texture/color in one, like for example the tree and the girl's trousers. As we move to higher radius, the level of details and number peaks is becoming smaller and smaller. For radius 10, the face of the girl is fading away. In addition for radius 40 we only have two segments. Finally for radius 50 the algorithm finds only one peak and associates everything with that peak.

Running the same experiment using the 5 dimensions model produces the following result:

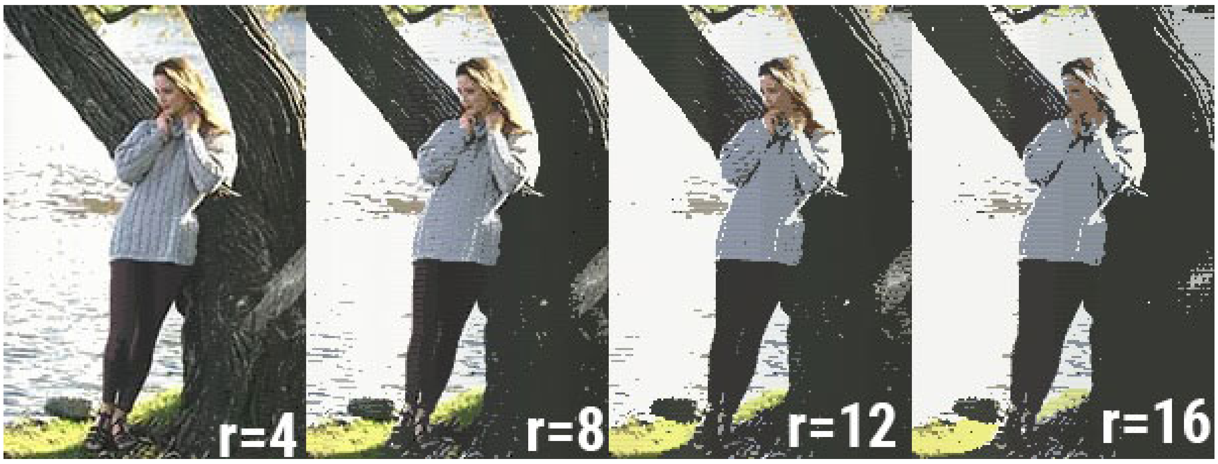

Figure 2.3: Different radius using 5D representation

2.3 Effect of value c of second optimization

In the second optimization implemented, the findPeak function instead of associating with this label only the current point and the points that were in the sphere in the last iteration, it also associates a smaller sphere of the points of all the iterations that it went through until it converged to this one. This optimization saves a small path using the principle that if a point was inside there then it will eventually end at the same peak because it will follow the same path. The radius for this smaller sphere is $radius/c$ where c is number.

In this experiment, I kept the radius stable and changed the c value to investigate the difference that it would have in the segmentation of the image.

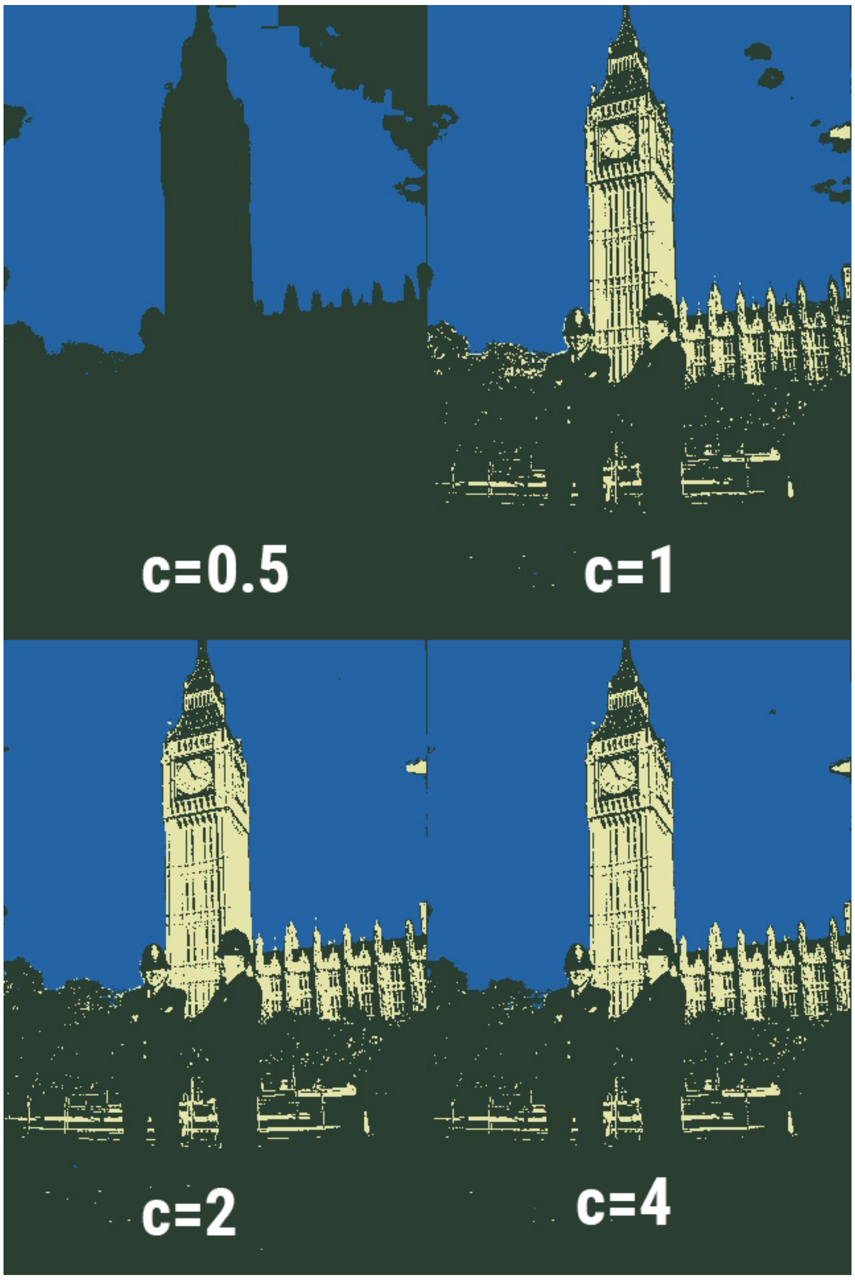

Figure 2.4: Effect of c parameter in the clustering with radius 20 and 3d representation.

2.4 3D vs 5D

The mean shift algorithm is independent of the dimensions of the point fed into it. In the begining we gave an image as input to the algorithm containing 3 dimensions. The 3 dimensions corresponded to the RGB color values of the point and later to the LAB values. The LAB values are a different color space format where L stands for lightness, A is a value going from white to black, and B is a value going from green to red. Although this approach performs relatively well, it misses an important piece of information, the position of the point in the original image.

In order to tackle this problem we use a 5 dimensions representation. This representation holds 2 more dimensions containing the x, y positional values of the point in the original image. This information is giving us the option to calculate distances based on the color but at the same time on the distance that the actual points have between them. That is extremely useful because it helps the algorithm to not associate points that are in opposite directions in the image under the same peak, but constrain them in a smaller region around them.

Below we can see a few examples of image segmentation using both 3D and 5D representation with exactly the same parameters.

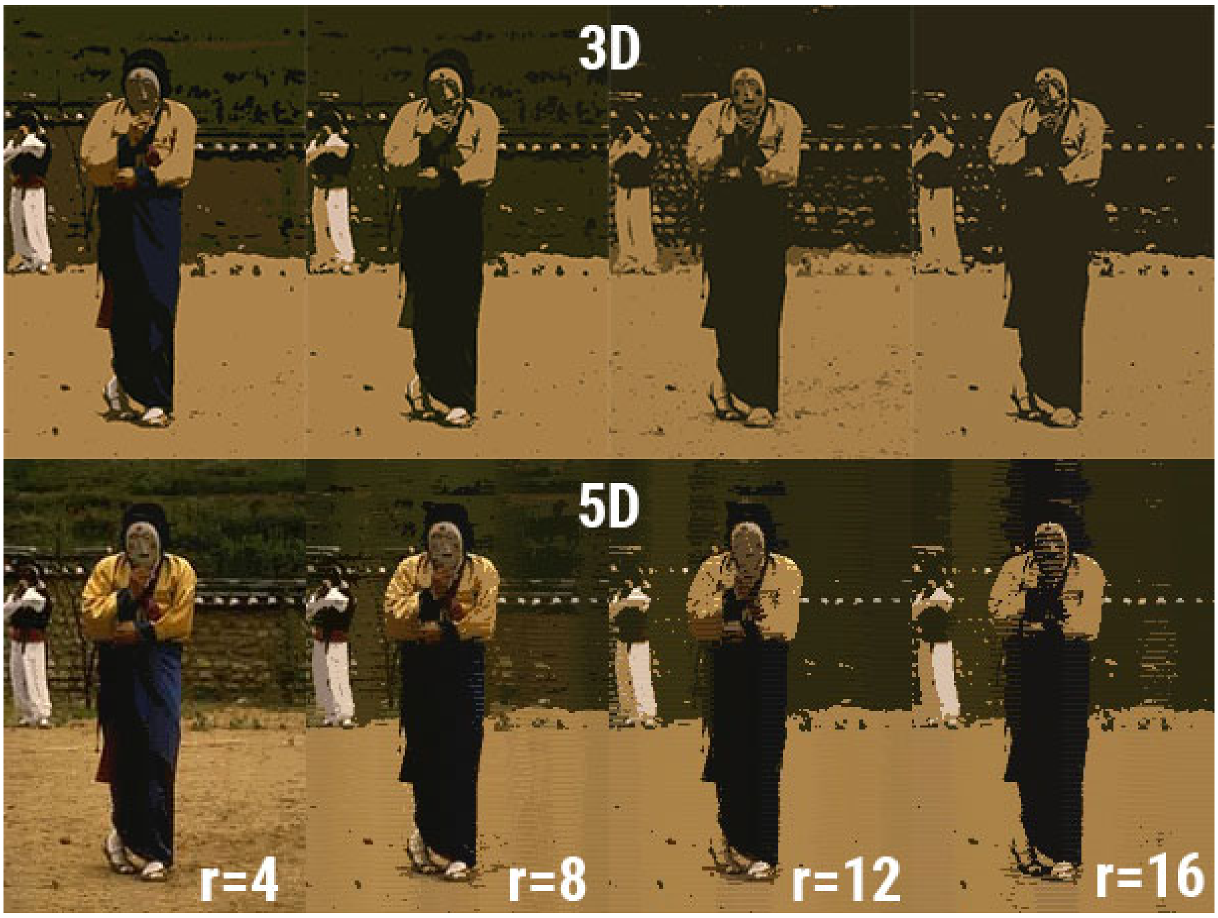

Figure 2.5: 3D vs 5D Side to Side comparison

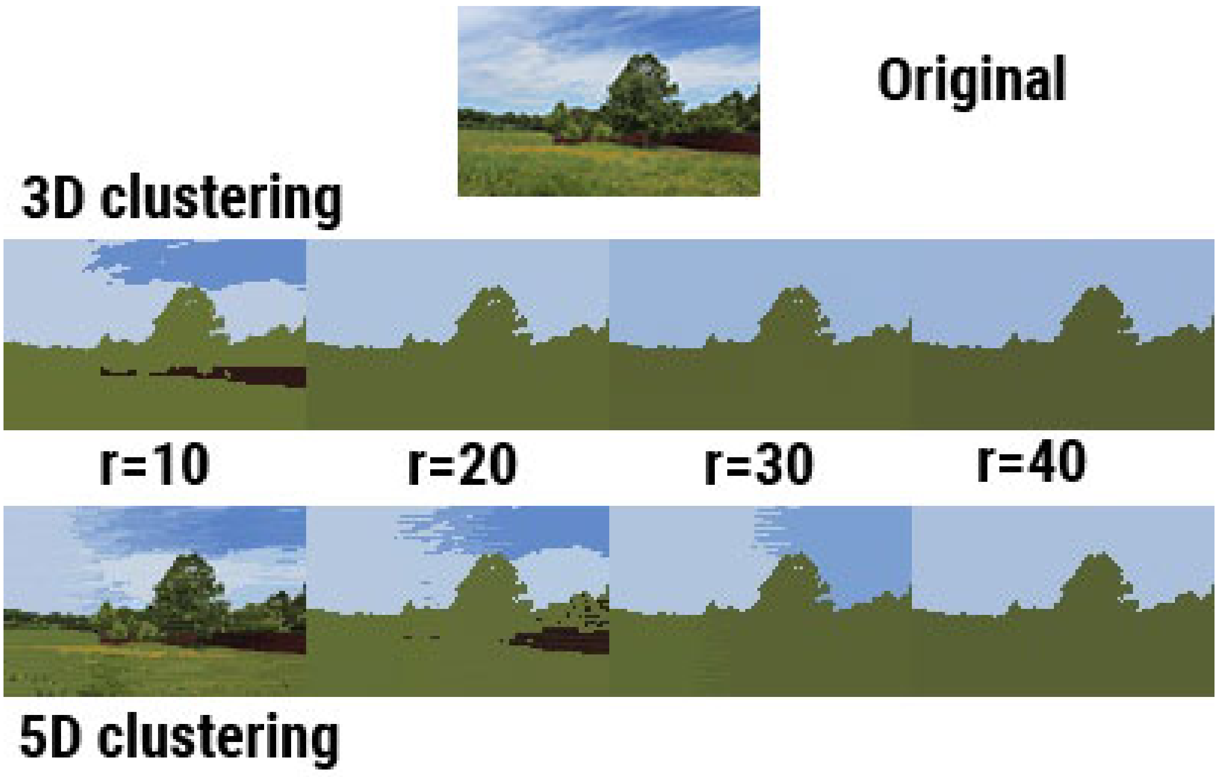

Figure 2.6: A more simplified image containing a landscape with a tree, a field, and sky. Comparison of 3D and 5D

In figure 2.6 we can clearly see that for radius 10, the 5D representation is creating more peaks and more segments because even though the segments are containing similar color, they are far away from each other and thus belonging in a different segment.

3 Can you suggest preprocessing steps in order to improve the segmentation results?.

To improve the outcome of a segmentation algorithm there are a few steps that we can follow.

3.1 Histogram Equalization

A lot of times images are either too bright, or too dark resulting in a histogram representation that has a lot of peaks and low points in the tonality values. When you equalize an image, you move this high peaks from the edges of the histogram to the center but also reduce the height difference, thus resulting in a higher global contrast image. Having a higher contrast means that the color are more vivid thus their numerical representation have bigger differences, making segmentation easier. Also parts that were not distinguishable are now visible and able to be segmented.

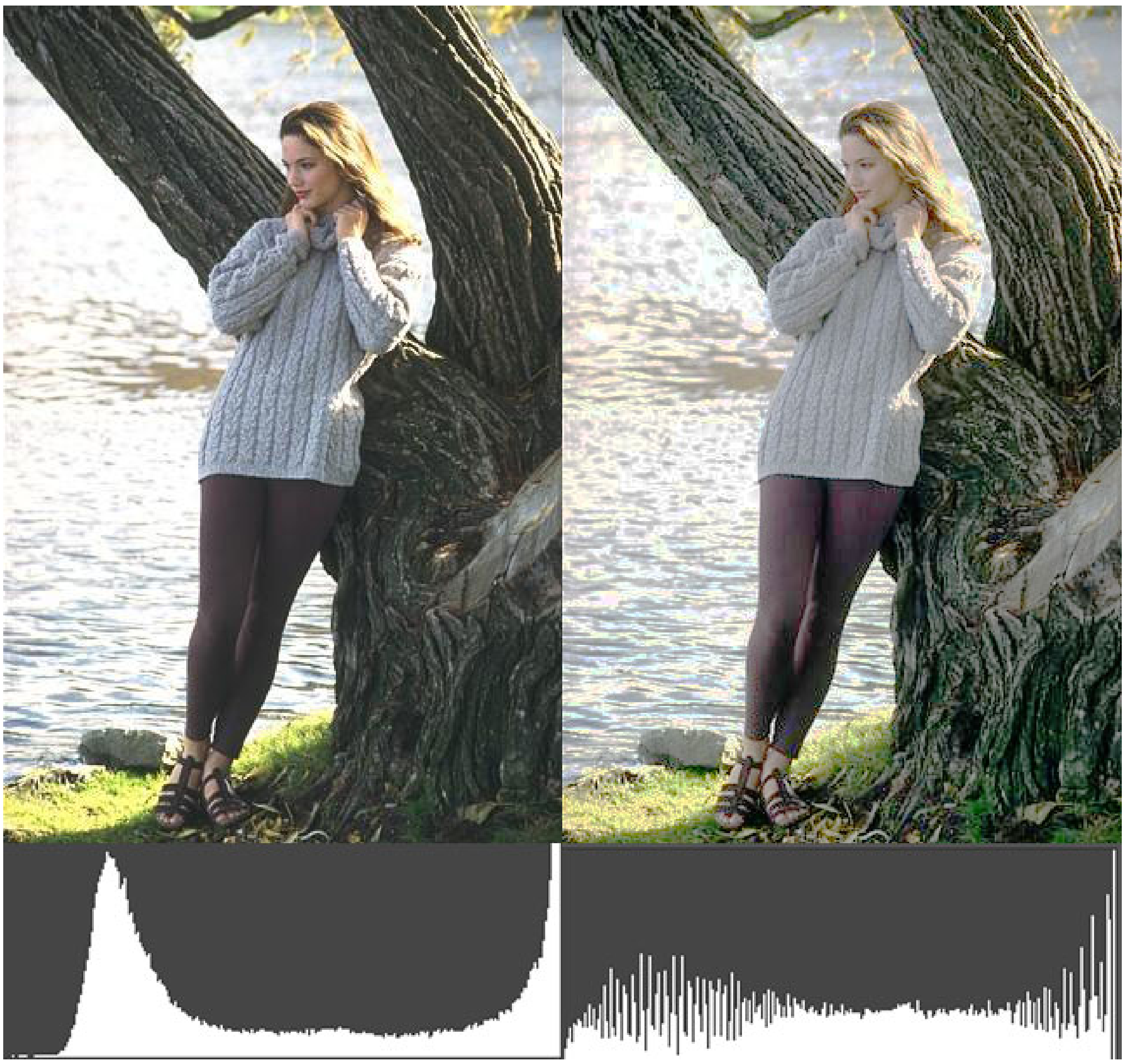
Figure 3.1: Histogram representation before and after equalization.

3.2 Noise reduction

When segmenting we noticed that the mean shift algorithm was finding peaks, not because the region was different but because there was salt and pepper noise on the image. Usually formed as horizontal or vertical lines can distort the outcome of the segmentation. In that case we can use a weighted median filter to get rid of this noise and make the image smoother and homogeneous.

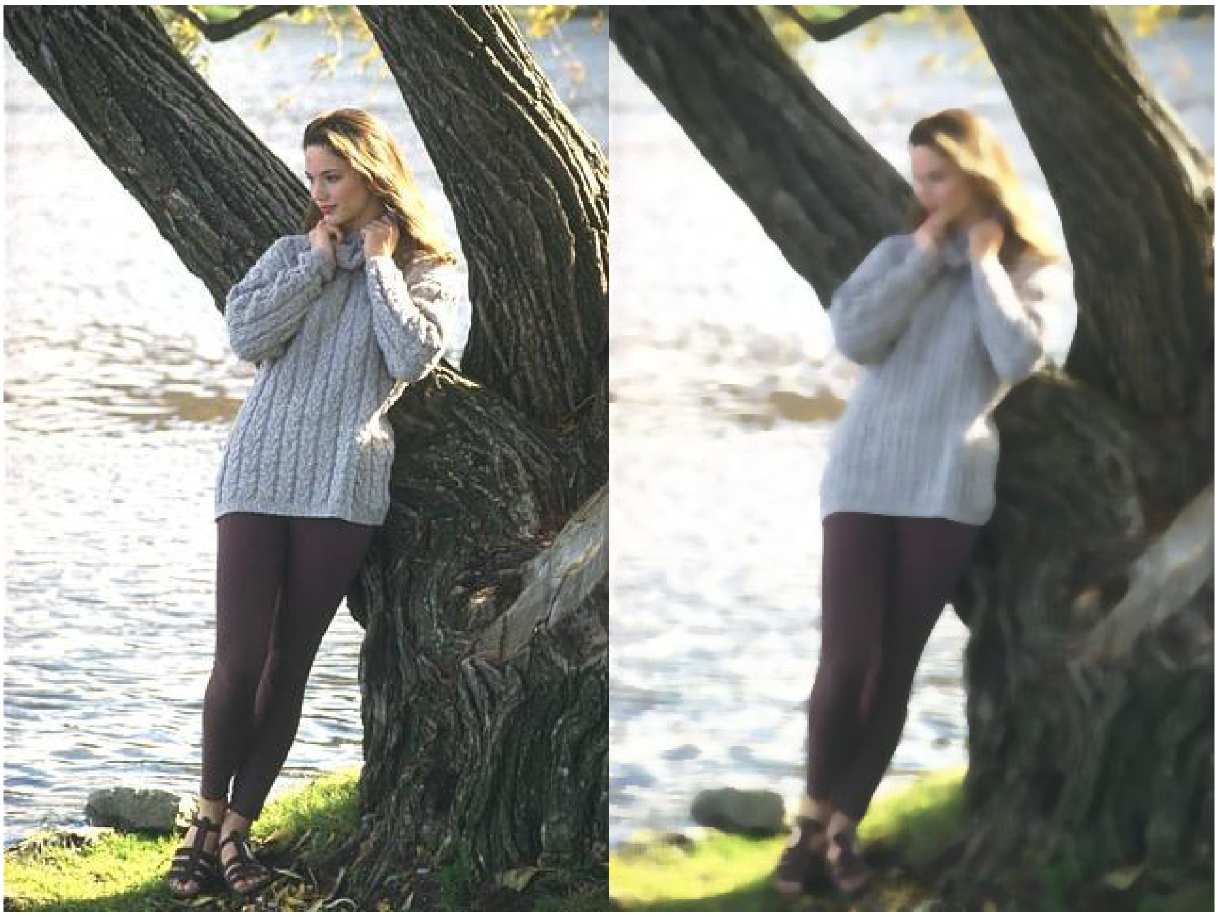
Figure 3.2: Original on the left / Median filter applied on the right

3.3 Image normalization

A simple also preprocessing process would be normalization of the pixel intensity values. By using normalization we can achieve consistency between our image data.

4 Example of segmentation with preprocessed image

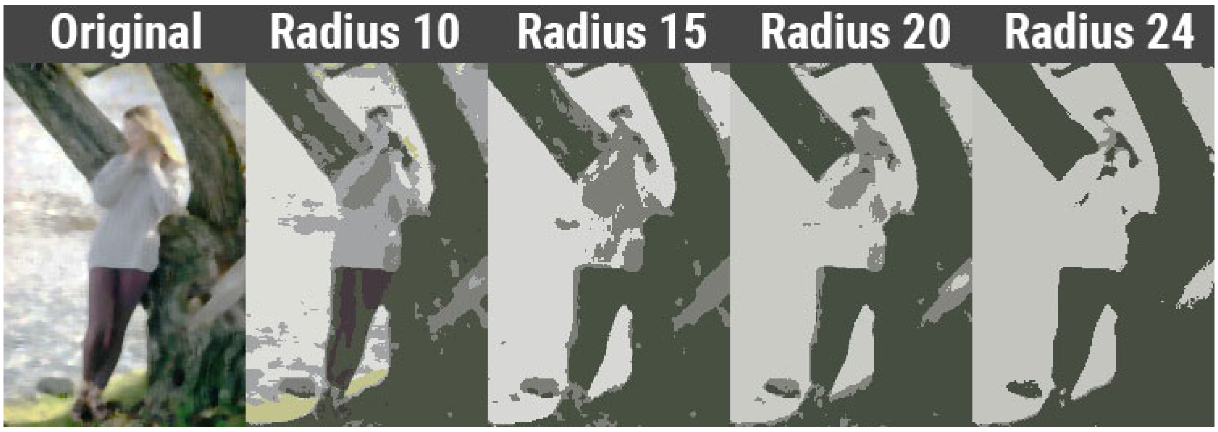

Figure 4.1: Preprocessed image segmentation after applying histogram normalization and mean filter