IMAGE PROCESSING

Coursework 1:
Simple Image Segmentation

Marios Bikos - CGVI 2015-2016

UCL ENGINEERING
Change the world

Contents

1 Introduction

1.1. Attempts and Achievements

Image segmentation refers to the problem of splitting an image into separate regions to come up with a conclusion about the image and its elements in it and generally simplify its representation into something that can better be understood.

In this report, a series of algorithms is implemented using Matlab to better understand the basic concepts behind basic image segmentation algorithms such as Thresholding, Region Growing and Mean-Shift. Furthermore, ROC curves and methods on how to get the Operating Point of a ROC curve or how to know if a ROC curve is better than another, are presented.

The Matlab code implemented, presents all the results at once to the user. All required tasks were completed.

Brief List:

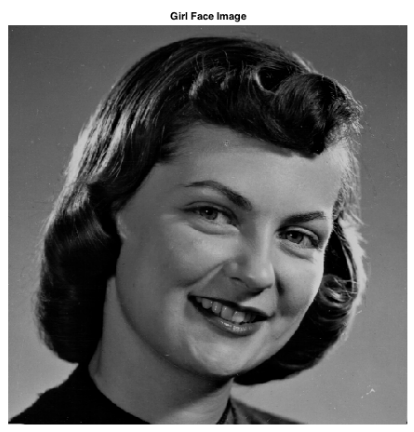
- Core Section
  1. Histogram Plotting and Histogram Comparison
  2. Image Thresholding and Ground Truth Segmentation
  3. ROC Curve and Operating Point Algorithm
- Advanced Section
  1. Region Growing Algorithm
  2. Mean-Shift Algorithm

2
Core Section

2.1. Download the Girl Face image and plot a histogram of the grey levels it contains. (Try plotting a histogram of another grey scale image and comparing the difference) (5 points)

In this first task, we just have to load the required girl face image that is given and plot the image histogram of the grey level intensities it contains.

We decided to both implement code which calculates the histogram of the image and use the built-in function ”histogram” of MatLab. In order to get the histogram, what we did was to count for every gray level intensity level(0-255) the number of pixels that belong to each intensity category. The result is shown below:



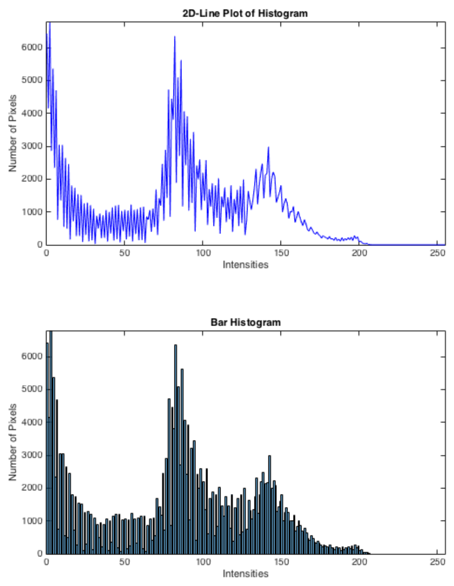


Figure 2.1: The Girl Face image and the 2 representations of its histogram

Next up, we loaded another image in order to compare its histogram with the histogram of the Girl Face image. In this case, we used the well-known Lena image for the comparison. All we had to do, was to plot both histograms on the same figure, utilizing the *hold on* command of MatLab.

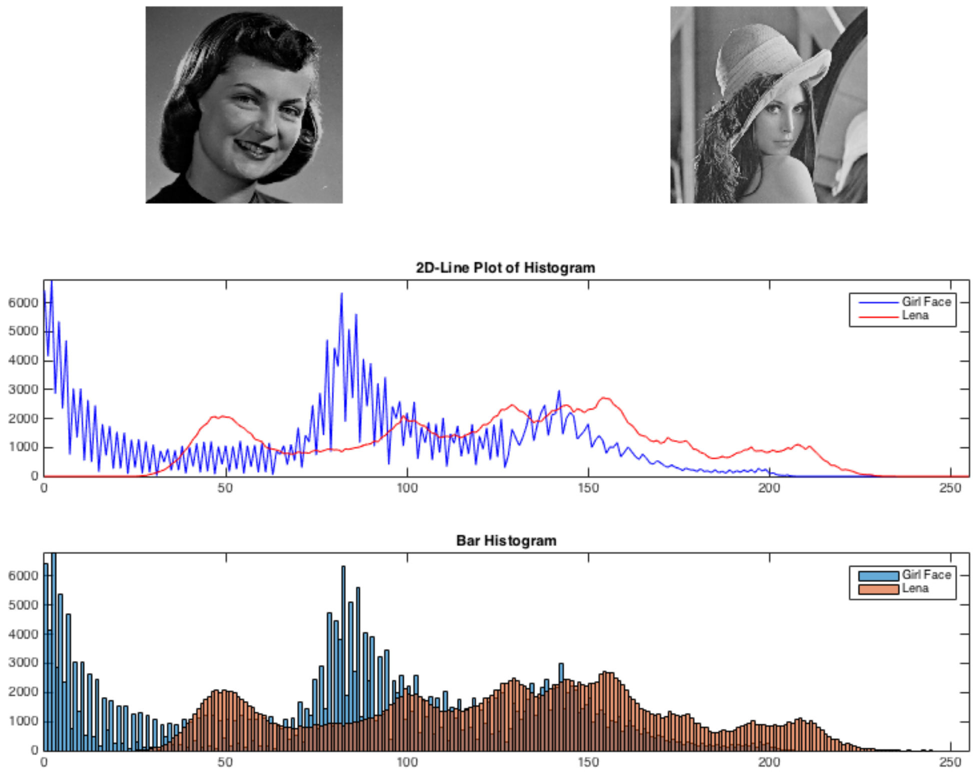


Figure 2.2: The comparison of histograms between Girl Face and Lena images

We can see that the histogram of Lena image has relatively equal values across the intensities, whereas the histogram of girl face image shows that most pixels are in the region 0-100 and mostly around 0 and 80 (approximately). We can also realize that by looking at the images, since the hair of the girl at the girl face image seem darker.

2.2. Write a short program to threshold the image and try to identify a good threshold by trial and error. Create a ground truth segmentation. You can do this using Matlab (look up the function roipoly), or using other painting tools. (15 points)

In this task, we needed to threhold the image which is an easy task if you use a simple thresholding algorithm. In this case, you just have to check each pixel's intensity value and if this value is greater than a threshold value we define, then this pixel's new value is equal to 1, otherwise it is equal to 0 (so the image can have only black or white pixels). Below you can see the results of this algorithm for 5 different thresholds(T=70,90,110,130,150). If we try to identify a good threshold based on the trial and error method, we can see that a good threshold could be the T=110, based on the fact that we want to seperate the skin region from the hair and background. Using this threshold a rather big part of the face skin is segmented correctly.

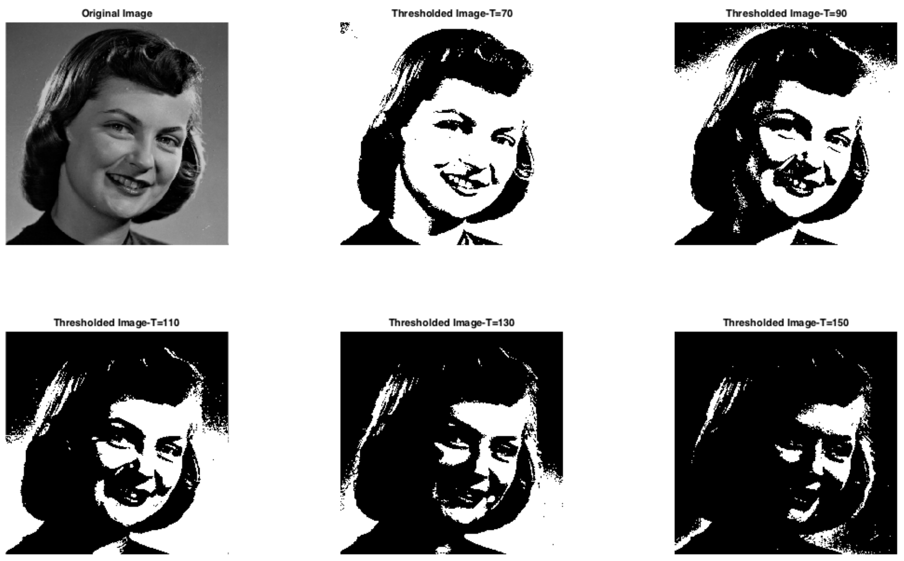


Figure 2.3: The results of thresholding algorithm for different threshold values

Next, a ground truth segmentation of the girl face image had to be created. In order to do this, we used the roipoly function of matlab, which enables users to select a polygon based on points defined by the user. The result of the roipoly function is a new image with white pixels inside the selected area of the polygon. This way, we created a ground truth image, trying to select the region of the face of the girl in the image. The result is shown below:

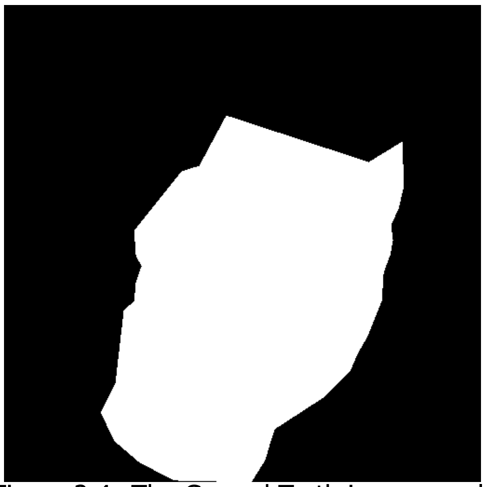


Figure 2.4: The Ground Truth image result

2.3. Use your ground truth segmentation to plot an ROC curve for your thresholding algorithm.What threshold does the ROC curve suggest? You may have to implement an algorithm to find the point closest to a desired Operating Point. How does it compare to your trial and error estimate? (30 points)

Here we had to use the ground truth image segmentation selected by the user to create a ROC curve which gives us several interesting information about out thresholding algorithm. In order to create a ROC curve, one needs to calculate the True Positive,True Negative,False Positive and False Negative pixels of the thresholded image. In order to find the number of each of these elements, we had to compare the thresholded image pixels with the corresponding pixels of the ground truth image. More specifically:

- If the pixel value of the Ground truth image is 1 and the corresponding pixel's value is also 1, then the pixel is considered a **True Positive**
- If the pixel value of the Ground truth image is 1 and the corresponding pixel's value is 0, then the pixel is considered a **False Negative**
- If the pixel value of the Ground truth image is 0 and the corresponding pixel's value is also 0, then the pixel is considered a **True Negative**
- If the pixel value of the Ground truth image is 0 and the corresponding pixel's value is 1, then the pixel is considered a **False Positive**

The code is also shown below:

```
for x=1:size(img,1)
      for y=1:size(img,2)
         if( groundTruth(x,y)==1 && ThresholdedImage(x,y)==1 )
            TP(T+1)=TP(T+1)+1;
         elseif( groundTruth(x,y)==1 && ThresholdedImage(x,y)==0 )
            FN(T+1)=FN(T+1)+1;
         elseif(groundTruth(x,y)==0 && ThresholdedImage(x,y)==0)
            TN(T+1)=TN(T+1)+1;
         elseif(groundTruth(x,y)==0 && ThresholdedImage(x,y)==1)
            FP(T+1)=FP(T+1)+1;
         end
      end
   end
```

After calculating these values we can find the True Positive Rate and the False Positive Rate which we use as the x and y axes for the ROC curve. The TPR and FPR are given as $TPR = \frac{TP}{TP+FN}$ and $FPR = \frac{FP}{FP+TN}$. We do the same thing for all possible thresholds from 0 to 255 and therefore we have 256 points which create a curve if we connect the points.

We were also asked to implement an algorithm to find the point closest to a desired Operating Point. The desired operating point can be the point which is closer to the (0,1) optimal thresholding point in the figure, and also has a gradient $B = \frac{N}{P}$, where $P = TP + FN$ and $N = FP + TN$.

So in order to get this point, we created 2 arrays. The first array includes the distances between each point of the ROC Curve and the point (0,1).In the second array we would like to have the difference between the gradient B and the gradient at each point of the ROC Curve.Since a gradient can be defined using 2 points and not just one(if we don't know the parameters of the curve), therefore we opted to get the gradient for a point k,by calculating the gradient between the point k and the point k+1. This is an approximation that works pretty well.

So we have 2 array with the criteria we need to find the operating point.Since we need to take both criteria equally into account, we created a metric which calculates the best threshold point of the Curve.

```
metricValue=0.5*NormDistFromOptimalPoint+0.5*NormDiffFromCorrectGradient;
```

The operating point is the index of the minimum value of the metricValue variable which corresponds to a specific threshold T. In our case, we found out that his threshold was equal to 91. Below you can see both the ROC Curve with the operating point and the thresholded image using 91 as a threshold.

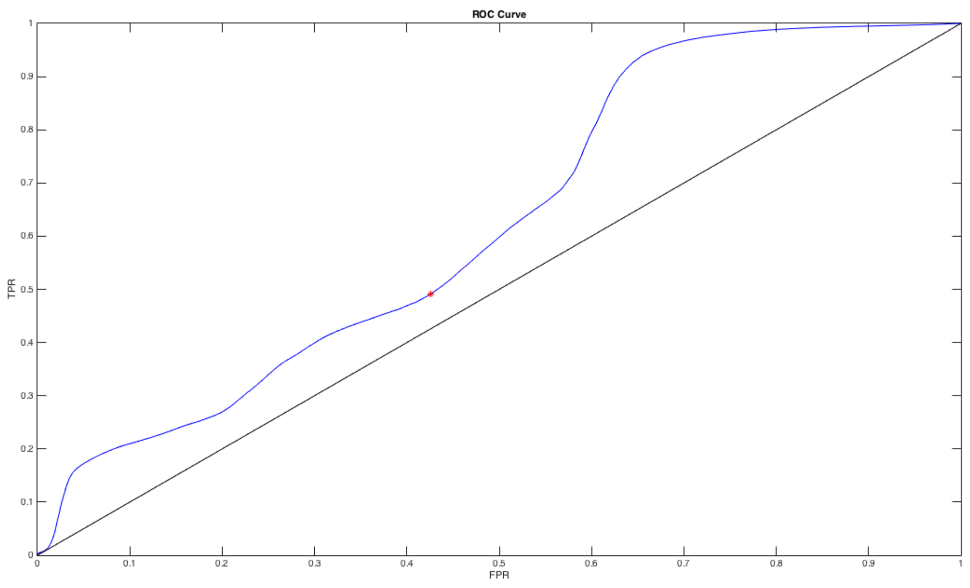


Figure 2.5: The ROC Curve(blue) and the Operating Point(red)

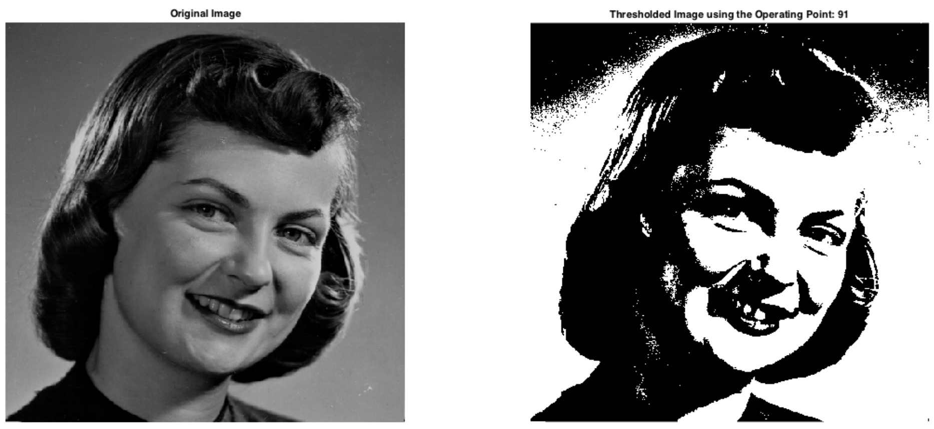


Figure 2.6: The result of thresholding using T=91

3

Advanced Section

3.1. Implement a region growing algorithm and compare its performance at segmenting the face in the image above with the performance of the thresholding algorithm. Show the seeds you chose. (25 points)

In this first advanced section task, it was required to code a region growing algorithm,check the results and compare the performance with the one of the thresholding algorithm.

In the region growing algorithm, we start from a seed pixel(or more) and we examine each of its neighboring pixels(in this case 4-neighbors) to determine whether the pixel neighbors should be added to the region based on the validity of a condition. In this case, following the theory of the course and slides of Prof. Agapito, we considered that this condition is defined as whether or not the neighbor pixel is above a certain threshold T. (Although one could also check if the difference between the intensities of each pixel and its neighbors is lower than a threshold).

The methodology we followed to implement region growing algorithm was to let the user select some initial seeds and then for each seed pixel, we select the valid 4-neighboring pixels(if they are valid-which means they are inside the border of the image dimensions) and check if the condition we mentioned is true. If so, we change the value of the final image to 1 and add this pixel to a queue of border points that we have to check in the next iteration.After we check the neighbors of each pixel, we delete it from the top of the list. This goes on and on until the border list has 0 elements inside.

In our case the seed pixel was at (422,257) which is around the chin of the girl face. However our program also works if users decides he want to use more seeds.If this happens, we calculate the new image for each seed and then we combine them together to merge all white pixels together.

Below you can see the result of this approach for a threshold T=95.

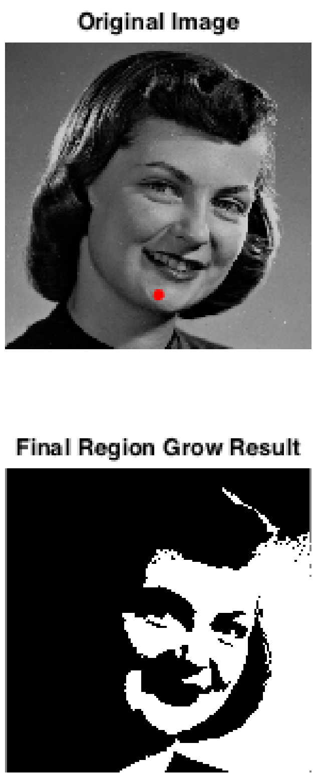


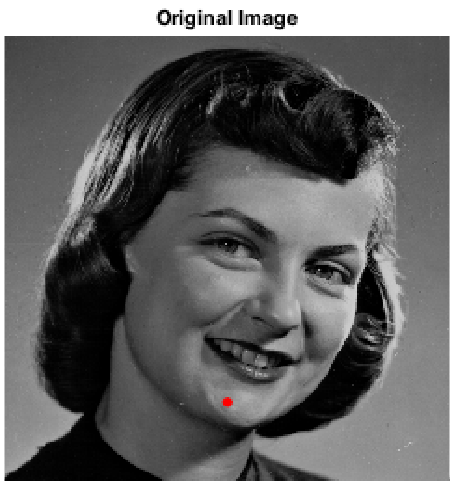
Figure 3.1: The region growing example for T=95

In order to get the ROC curve for the region growing algorithm, we just need to do the same thing for every threshold value from 0-255.!!It is worth mentioning that this part of the code takes a while to be completed since we have to do region growing 256 times!!

This way we can get the ROC curve and if we plot it on the same figure as the previous ROC curve from thresholding algorithm, we can compare them. To compare the performance we need to compare the area under each ROC curve. To do this we used trapz function of matlab which integrates the area below the curve and so we have an approximation of the area. The values we got were:

- Area of Region Growing: 0.65 or 65%
- Area of Thresholding Algorithm: 0.62 or 62%

So we can see that the performance of region growing algorithm is better.



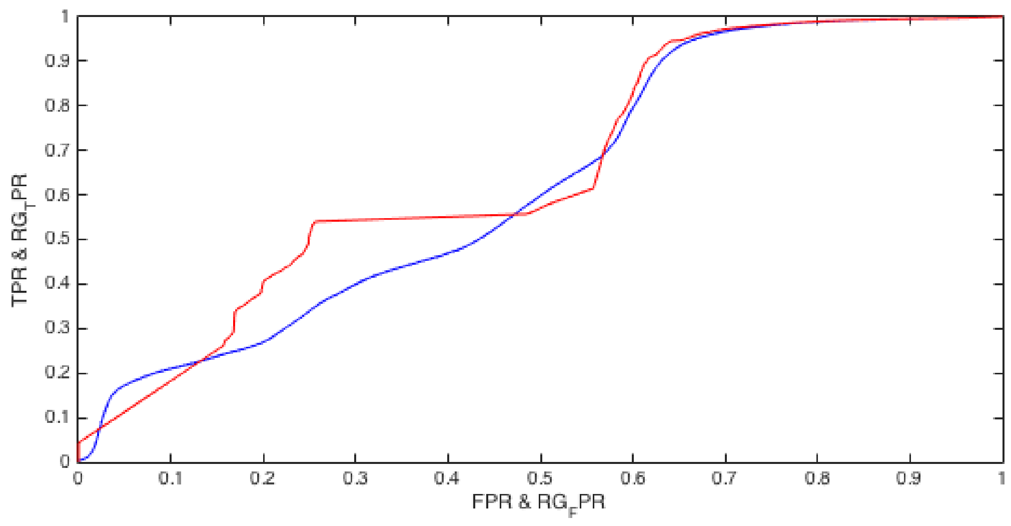


Figure 3.2: The selected seed(red point) and the comparison of ROC Curves between Region Growing(red line) and Thresholding(blue line) algorithms

3.2. Implement the mean-shift algorithm and show the results that you achieve for different values of the "radius" parameter. (25 points)

In this task, it is required to implement the mean-shift algorithm and test the result images for several different radius parameters.

In our case, the only things we will use are intensity values of the histogram, so feature space will only be one-dimensional. The methodology we followed was:

1. For every intensity value(0-255) create a window with a width of 2*radius+1 around the

intensity value.

2. We calculate the weighted average of each window based on this formula:

$$w_m = \frac{\sum_{w_i} b_i f(b_i)}{\sum_{w_i} f(b_i)} \tag{3.1}$$

where b_i is the intensity and $f(b_i$ the number of pixels of this intensity.

3. Shift the window toward the new w_m.

4. Repeat until convergence

So, at the end of the day, we have created a new array of 256 elements and each cell of the array has the new intensity of the corresponding intensity.For example intensity 0 may change to intensity 20. What we we do next is to change every pixel of the girl face image to the new intensity we calculated through mean-shift.

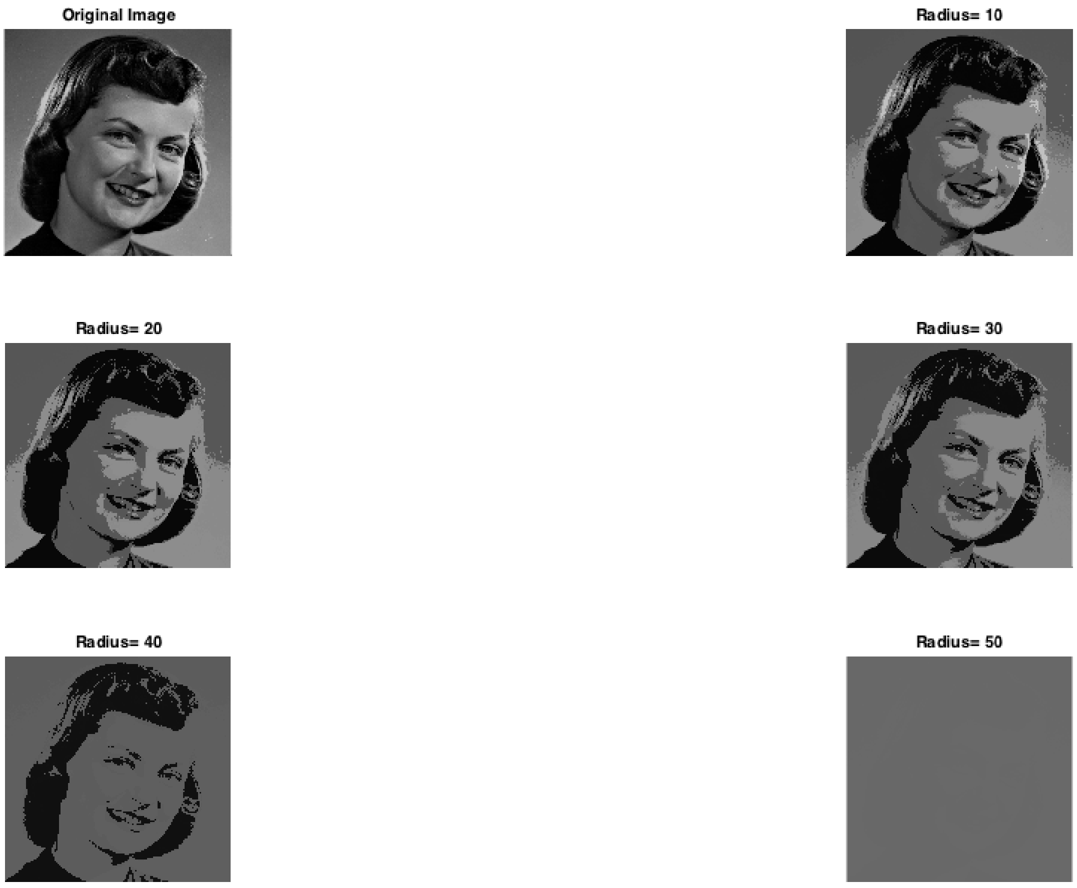


Figure 3.3: Mean Shift Results for different radius values

Bibliography

[1] M. Hinkley, *Photo courtesy of mary hinkley // copyright: © ucl media services. composite image courtesy of charles thomson, ucl cege.* .