Fast Prototyping Exercise 2

Mean Shift Segmentation

Sabiha Barlaskar

917932501

1. Acquired a 512X512 gray image and rescaled it to 64X64 for fast execution.
   original = double(imread(inpImg));

2. Initialized the mean to a set of pixels and intensity values.
   mean_val = [i,j,original(i,j)];
3. Calculated the weight using Gaussian kernel, with bandwidth hr as the parameter which is the range domain, where I computed the difference between the mean and the current pixel values(i1,j1) and the intensities (Original(i1,j1).
   weight = exp(-1 * ((mean_val(1) - i1)^2 + (mean_val(2) - j1)^2 + (mean_val(3) -

   original(i1,j1))^2)/25);

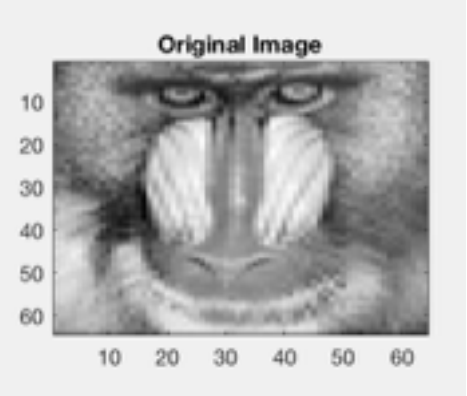
4. Using this weight I calculated the new mean values
   numerator = numerator + weight * [i1,j1,original(i1,j1)];
   denominator = denominator + weight;
   mean_new = numerator / denominator;

5. Calculated the shift in the new mean from the old mean and iterated this process until the mean values converged. Since using Gaussian kernel, the algorithm converges in an infinite number of steps which goes to negative values, so I have taken a threshold of 0.1, below which if there are norm mean shift values, then the algorithm will terminate

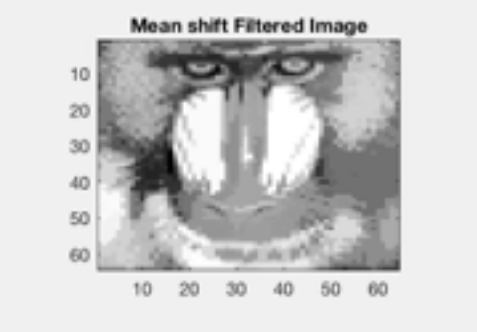
```
mean_shift = mean_new - mean_val;
norm(mean_shift);
if(norm(mean_shift)<0.1)
        iter = 0;
end
mean_val = mean_new;
new_image(i,j) = mean_val(3);
```

On an average of 12 iterations were required for converging
Observations:

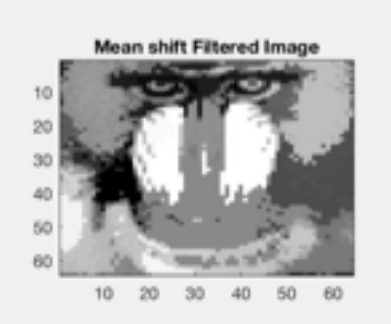
Here are some of the observations with different hr values



Original image resized to 64X64 from 512X512



Mean shift image with hr = 5



Mean shift image with hr = 8

Mean shift image with hr = 10

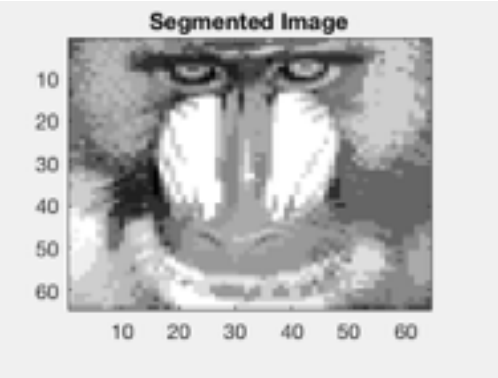
I saw that performing the mean shift filtering with increasing hr values, the algorithm directs the points toward increase in density which explains the details with high intensity values are prominent like the eyes and nose while details of fur are smoothed out.

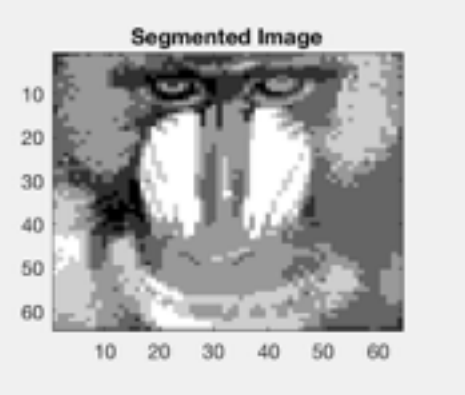
6. For segmentation, after removing the mean shift values below the threshold, there were pixels with very slight differences between them, for example 132.4 and 132.3. Therefore in

order to group them in single clusters, I have taken a bin size of 10 and divided the pixel values by 10. After dividing I took the floor of the values, which gave me pixel points with same values. For example floor of 132.4 and 132.3 will be 132.

s_image(x1,y1) = (floor(new_image(x1,y1)/10))*10;

On increasing the bin size, I found that the details smoothed out a little further because the difference between the pixel values increased and thus the density increased for the pixels. Here are the observations:



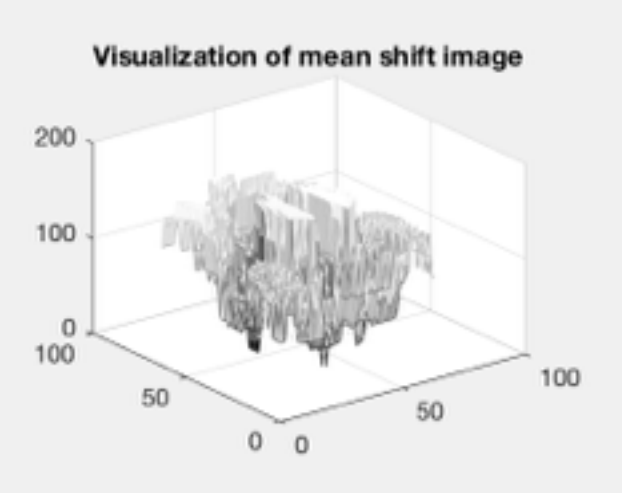


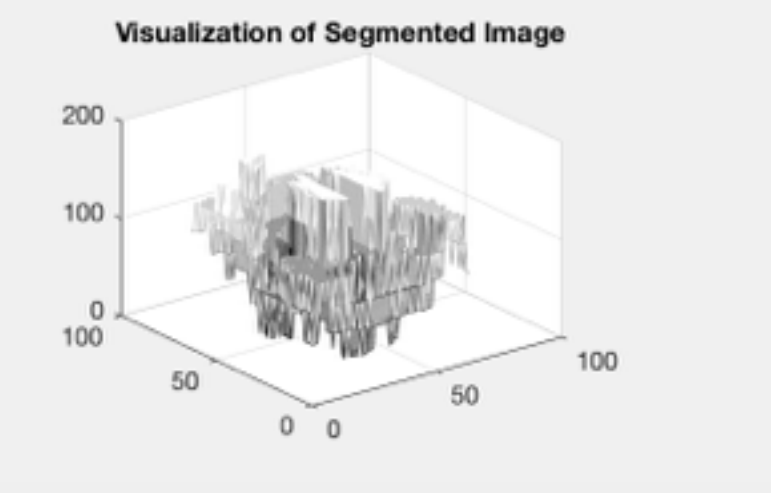
Bin size = 10

Bin Size = 30

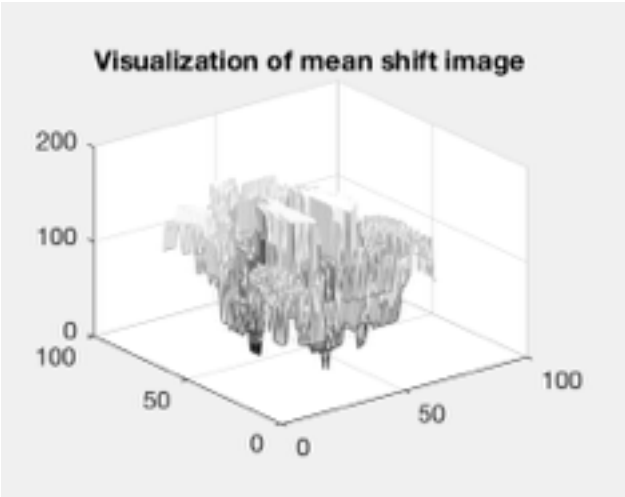
Visualizations of the mean shift and the segmented image

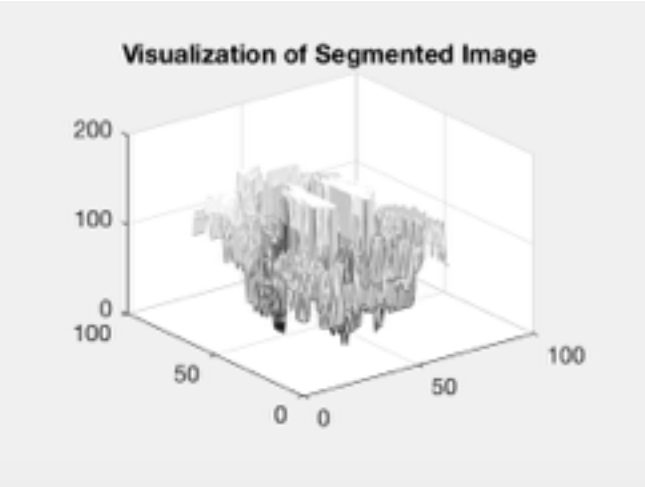
For hr = 5, Bin Size = 30





For hr = 5, Bin Size = 10

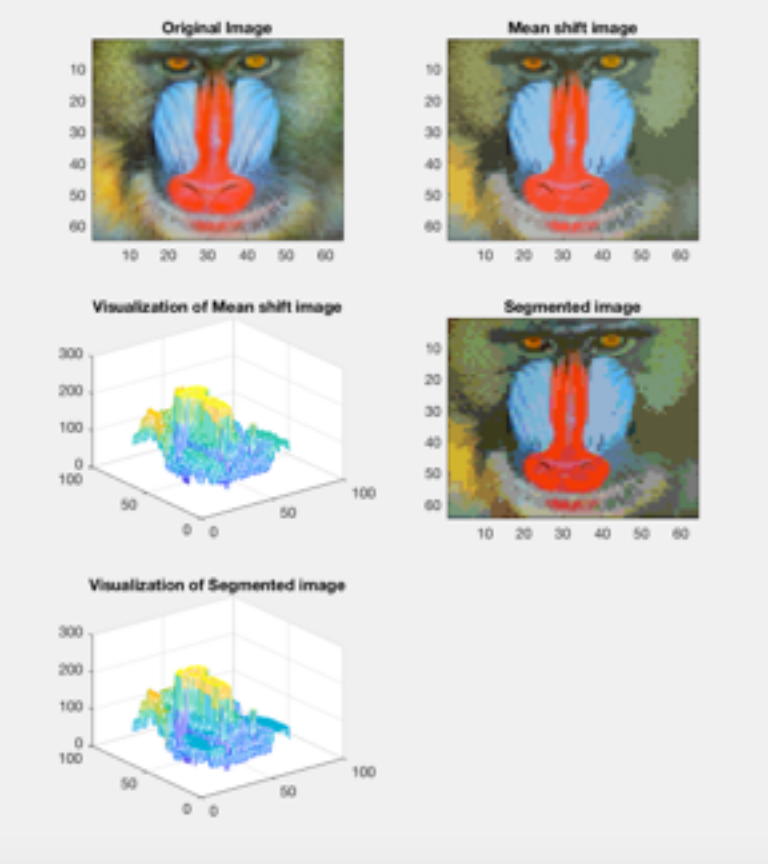




For colored Image, I followed a similar approach, but I took the intensities for all the RGB for calculating the new weighted mean

weight = exp(-1 * ((mean_val(1) - i1)^2 + (mean_val(2) - j1)^2 + (mean_val(3) - original(i1,j1,1))^2 + (mean_val(4) - original(i1,j1,2))^2 + (mean_val(5) - original(i1,j1,3))^2)/100);

Results for hr = 10, bin size = 30



Code for mean shift segmentation in gray image

```
inpImg = '/Users/sabihabarlaskar/Documents/MATLAB/Segmentation_Data/gray/
Baboon.bmp';

mean_shift_gray_segment(inpImg)
function mean_shift_gray_segment(inpImg)
%read the image
  original = double(imread(inpImg));
  figure;
%Resize the image
  num_pixels = size(original,1);
  scale_factor = 64/num_pixels;
  original = imresize(original, scale_factor);
  subplot(3,2,1);
  colormap('gray');

  imagesc(original);
  title("Original Image");
%Initialize a new matrix
  new_image = zeros(64,64);

  for i = 1:64
   for j = 1:64
   iter = 25;
%Initialized the mean with pixel values and intensity
   mean_val = [i,j,original(i,j)];
%loop for convergence
    while(iter>0)
      numerator = 0;
      denominator = 0;

      for i1 = 1:64
        for j1 = 1:64
%Calculated the weight using Gaussian Kernel and bandwidth h
          weight = exp(-1 * ((mean_val(1) - i1)^2 + (mean_val(2) - j1)^2 + (mean_val(3) -
original(i1,j1))^2)/25);
          numerator = numerator + weight * [i1,j1,original(i1,j1)];
          denominator = denominator + weight;

        end
```

```
      end
      %New mean
      mean_new = numerator / denominator;
     % Calculated the shift in the mean compared to the o
      mean_shift = mean_new - mean_val;
      norm(mean_shift);
    %Threshold for convergence
      if(norm(mean_shift)<0.1)
        iter = 0;
      end
      mean_val = mean_new;


      iter = iter - 1;
    end
   new_image(i,j) = mean_val(3);
   end

  end

  subplot(3,2,2);
  colormap('gray');
  imagesc(new_image);
  title("Mean shift Filtered Image");
  subplot(3,2,3)
  mesh(new_image);
  title("Visualization of mean shift image");
  s_image = zeros(64,64);
  %Segmentation
 for x1 = 1:64
    for y1 = 1:64
       %s_image(x1,y1) = floor(new_image(x1,y1)/10)
       s_image(x1,y1) = (floor(new_image(x1,y1)/10))*10;
    end
  end

  subplot(3,2,4);
  imagesc(s_image);
  title("Segmented Image");
  subplot(3,2,5);
  mesh(s_image);
  title("Visualization of Segmented Image");
end
```

Code for mean shift segmentation in colored image

```
inpImg_color = '/Users/sabihabarlaskar/Documents/MATLAB/Segmentation_Data/
BaboonRGB.bmp';

colored_mean_shift_segment(inpImg_color)
function colored_mean_shift_segment(inpImg_color)
  original = double(imread(inpImg_color));
  num_pixels = size(original,1);
  scale_factor = 64/num_pixels;
  original = imresize(original, scale_factor);
  subplot(3,2,1);
  %colormap('gray');
  imagesc(uint8(original));
  title("Original Image");
  new_image = zeros(64,64,3);
  s_image = zeros(64,64,3);
  for i = 1:64
   for j = 1:64
   iter = 25;
   mean_val = [i,j,original(i,j,1),original(i,j,2), original(i,j,3)];
    while(iter>0)
      numerator = 0;
      denominator = 0;

      for i1 = 1:64
        for j1 = 1:64
          weight = exp(-1 * ((mean_val(1) - i1)^2 + (mean_val(2) - j1)^2 + (mean_val(3) -
original(i1,j1,1))^2 + (mean_val(4) - original(i1,j1,2))^2 + (mean_val(5) - original(i1,j1,3))^2)/
100);
          numerator = numerator + weight *
[i1,j1,original(i1,j1,1),original(i1,j1,2),original(i1,j1,3)];
          denominator = denominator + weight;

        end

      end
      mean_new = numerator / denominator;
      mean_shift = mean_new - mean_val;
      norm(mean_shift);
      if(norm(mean_shift)<0.1)
        iter = 0;
      end
```

```
        mean_val = mean_new;


        iter = iter - 1;
      end
     new_image(i,j,1:3) = mean_val(3:5);

     s_image(i,j,1:3) = (floor(new_image(i,j,1:3)/30))* 30;
     end

    end


    subplot(3,2,2);
    imagesc(uint8(new_image));
    title("Mean shift image");

    subplot(3,2,3);
    mesh(new_image(:,:,1));
    title("Visualization of Mean shift image");
    subplot(3,2,4);
    imagesc(uint8(s_image));
    title("Segmented image");
    subplot(3,2,5);
    mesh(s_image(:,:,1));
    title("Visualization of Segmented image");

end
```