Name: MEHTA NIHAR NIKHIL

Roll: 13D100011

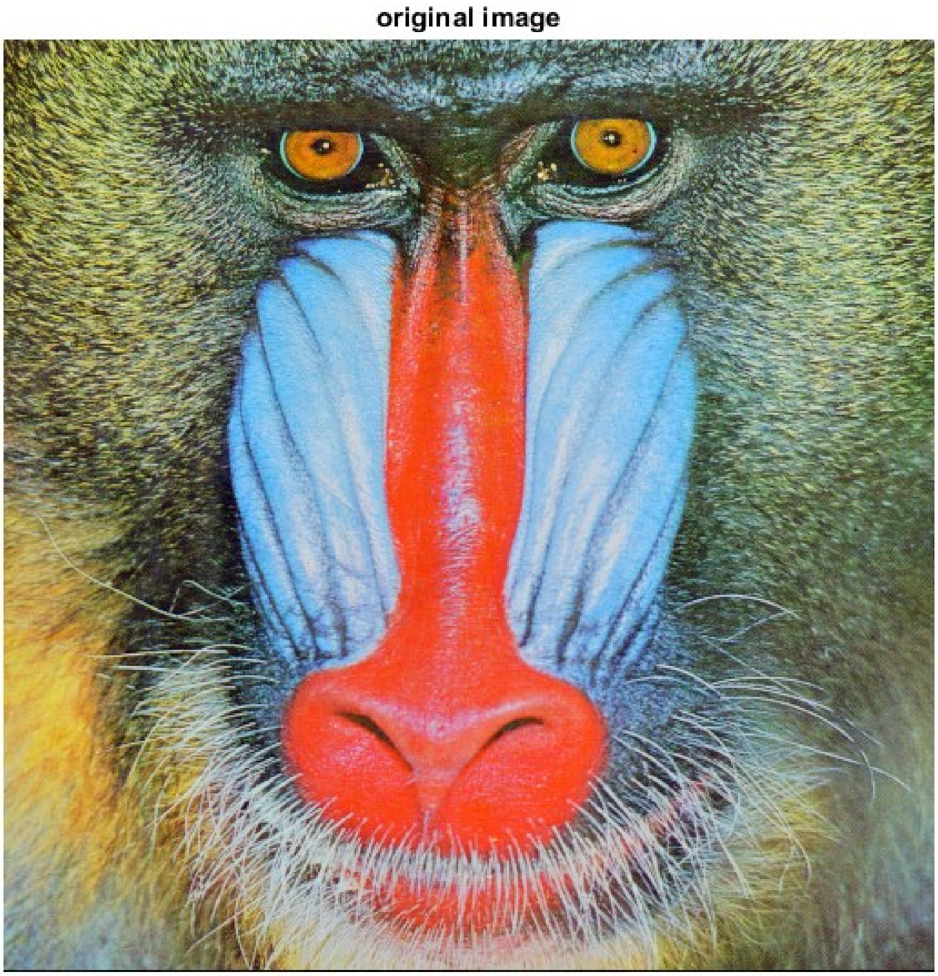
Code:

```
function [cluster_center]=myMeanShiftSegmentation(image_path,hs,hr)


    sigma=1; %Parameter for gaussian smoothing
    a=imread(image_path);
    figure,imshow(a), title('original image');

    I = im2double(a);

    %Gaussian smoothing
    kernelX = [[-1, 0, 1];
                [-1, 0, 1];
                [-1, 0, 1]];

    kernelY = [[-1, -1, -1];
                [0, 0, 0];
                [1, 1, 1]];


    kernel=exp(-0.5*((kernelX.^2 +
kernelY.^2)/(2*sigma^2)))/(2*sigma*sqrt(2*3.1415));

    I=apply_kernel(I,kernel);
   %Image shrinking
    I=myShrinkImageByFactorD(I,2);


    figure,imshow(I), title('filtered image');

    [x,y] = meshgrid(1:size(I,2),1:size(I,1));
    L = [y(:)/max(y(:)),x(:)/max(x(:))]; % Normalization

    C = reshape(I,size(I,1)*size(I,2),3);
    X = [L,C];
    X=X'; % 5 x 65536 vector [x y r g b]


    threshold=1e-2;

    [dims,num_points]=size(X);
    active_points=1:num_points;
    cluster_votes    = zeros(1,num_points,'uint16');

    visited=false(1,num_points);
    remaining_points=num_points;
    num_clusters=0;
    cluster_center=[];
    final_points=num_points;
    iter=0;
    while((remaining_points>0))
        iter=iter+1

        temp= ceil((remaining_points-1e-6)*rand);        %pick a random seed 
point
        point= active_points(temp);                  %use this point as start 
of mean
        mean= X(:,point);

        thisClusterVotes    = zeros(1,num_points,'uint16');
        cluster_members=[];


       count=0;
       %while(count<20) %Use this if stuck in the inner loop
        while(true) %Use the above if stuck here
           remaining_points


           list1=space_distance(X,mean);
           list2=intensity_distance(X,mean);

           %Choose points satisfying both bandwidths
           final_points=find((list1<hs) & (list2<hr));


thisClusterVotes(1,final_points)=thisClusterVotes(1,final_points)+1;

           mean_prev=mean;

           %Compute new mean

mean=gaussian_kernel(X(:,final_points),list1(final_points),list2(final_points
),hs,hr);

           %Add to the cluster
           cluster_members=[cluster_members final_points];

           %Keep a check of visited pixels
           visited(cluster_members)=true;

           %Convergence criteria
           if(norm(mean-mean_prev)<threshold)
              merge_check=0;
              for i=1:num_clusters
                  dist=norm(mean-cluster_center(:,i));
                  if ( dist<hs/2)
                     merge_check=i;
                     break;
                  end
              end
              if(merge_check>0)
                  cluster_center(:,merge_check)=
0.5*(mean+cluster_center(:,merge_check));
                  cluster_votes(merge_check,:)    =
cluster_votes(merge_check,:) + thisClusterVotes;

              else
                  num_clusters= num_clusters+1;
                        cluster_center(:,num_clusters)= mean;

                        cluster_votes(num_clusters,:)= thisClusterVotes;
              end
              break;

           end


            count=count+1;
        end


    active_points=find(visited==0);
    remaining_points=length(active_points);

    end


    num_clusters
    [val,data2cluster] = max(cluster_votes,[],1);

    cluster2dataCell = cell(num_clusters,1);

    for i = 1:num_clusters
        myMembers = find(data2cluster == i);
        cluster2dataCell{i} = myMembers;
    end
    clustMembsCell=cluster2dataCell;

    X=X';

    for i = 1:length(clustMembsCell)
% Replace Image Colors With Cluster Centers
        X(clustMembsCell{i},:) =
repmat(cluster_center(:,i)',size(clustMembsCell{i},2),1);
    end
        Ims = reshape(X(:,1:3),size(I,1),size(I,2),3);
% Segmented Image
        Kms = length(clustMembsCell);
        figure,imshow(Ims),title('segmented image');


end
```

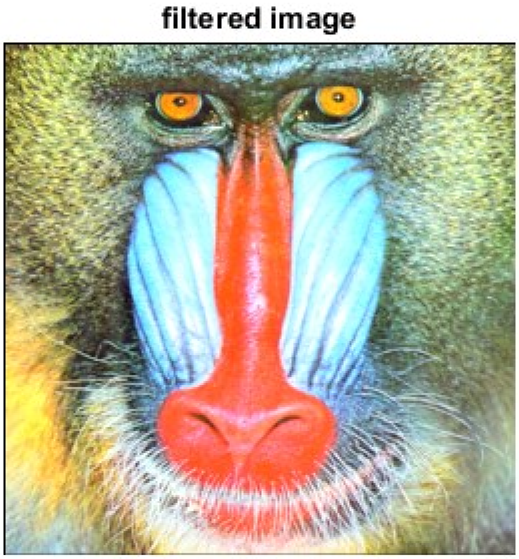
```
function out=gaussian_kernel(x,d1,d2,hs,hr)

    resolution = 1000; % resolution
    spatial = linspace(0,hs,resolution+1); % spatial
    range = linspace(0,hr,resolution+1); %range

    fun1 = exp(-(spatial.^2)/(2*hs^2));
    fun2 = exp(-(range.^2)/(2*hr^2));

    w1 = fun1(1,1:size(d1)).*(round(d1/hs*resolution)+1);
    w2=fun2(1,1:size(d2)).*(round(d2/hr*resolution)+1);

    w=w1+w2;

    w = w/sum(w); % normalize

    out = sum( bsxfun(@times, x, w ), 2 );
end

function list=space_distance(X,mean)

   list=sqrt((X(1,:)-mean(1,1)).^2+(X(2,:)-mean(2,1)).^2);

end


function list=intensity_distance(X,mean)

    list=sqrt((X(3,:)-mean(3,1)).^2+(X(4,:)-mean(4,1)).^2+(X(5,:)-
mean(5,1)).^2);

end

function [new_image]=apply_kernel(image,kernel)

[row,col,dim]=size(image);
[krow,kcol]=size(kernel);

new_image=zeros(row,col,dim);

midrow=floor((krow-1)/2);
midcol=floor((kcol-1)/2);

for i=1+midrow:row-midrow
    for j=1+midcol:col-midcol
        new_image(i,j,1)=sum(sum(kernel.*image(i-midrow:i+midrow,j-
midcol:j+midcol,1)));
        new_image(i,j,2)=sum(sum(kernel.*image(i-midrow:i+midrow,j-
midcol:j+midcol,2)));
        new_image(i,j,3)=sum(sum(kernel.*image(i-midrow:i+midrow,j-
midcol:j+midcol,3)));
    end
end
```

```
end
```

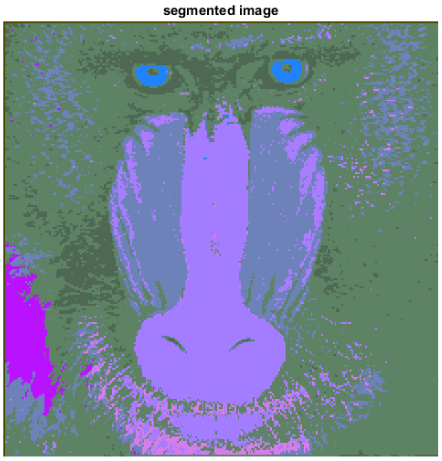
Original image:



Filtered image:



Segmented image:



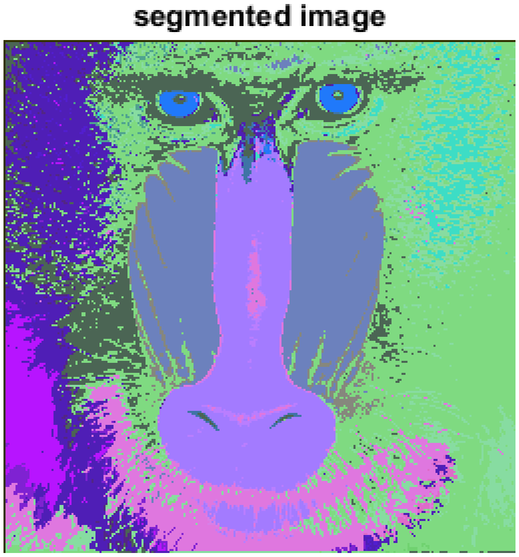
Spatial bandwidth = 0.8

Color bandwidth=0.1

Number of clusters=11

Number of iterations= 202

The above image is for a big spatial bandwidth. The segmented image for a smaller bandwidth is similar to this:


segmented image

Here, spatial bandwidth=0.5

Number of clusters=24

Number of iterations=269