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

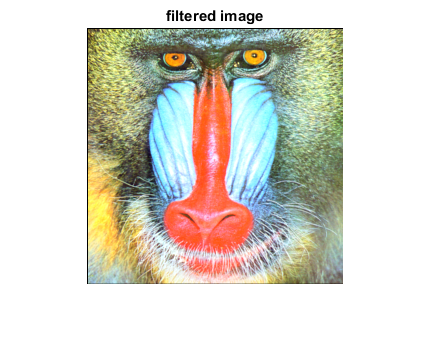
end

end

Original image:



Filtered image:



Segmented image:



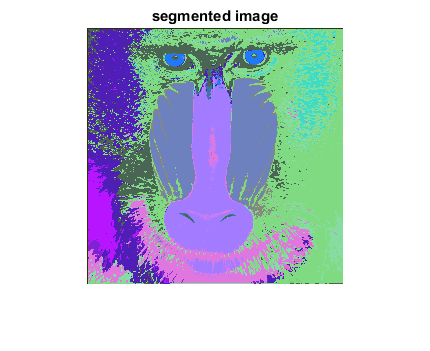
Spatial bandwidth = 0.8

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:



Here, spatial bandwidth=0.5

Number of clusters=24

Number of iterations=269