Contents

- MyMainScript
- Harris Corner Detector

MyMainScript

```
tic;
```

Harris Corner Detector

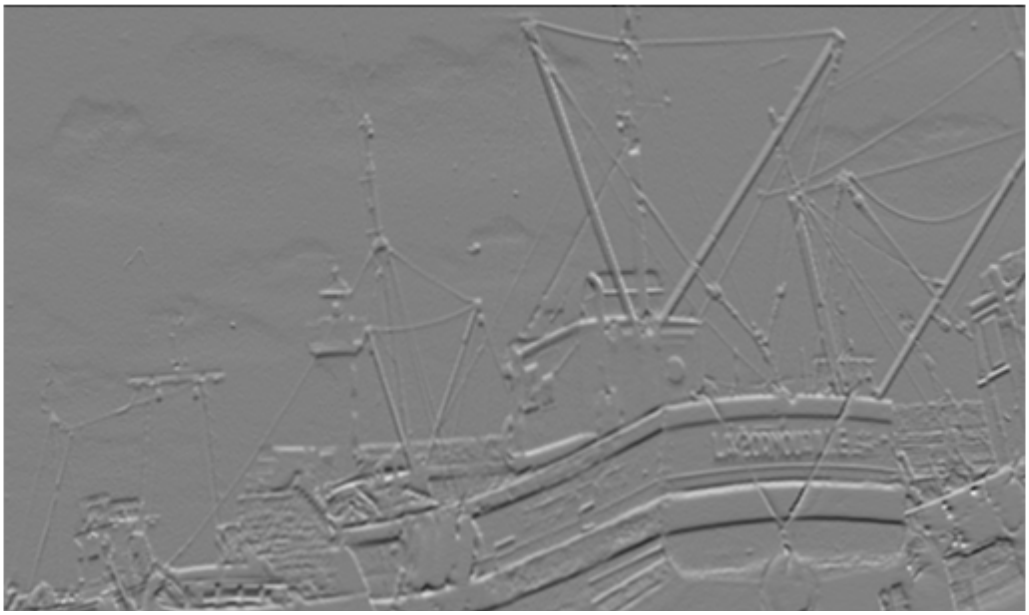
```
image  = load('../data/boat.mat');
inputImage = reScale(image.imageOrig);

figure, imshow(inputImage);
title('OriginalImage');
myHarrisCornerDetector(inputImage, 1, 0.15, 10);
toc;
```

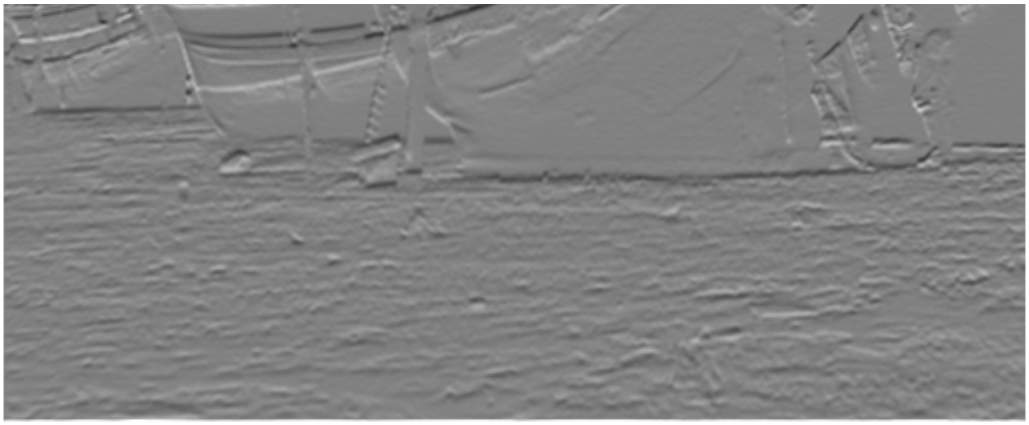
```
Elapsed time is 3.891328 seconds.
```

```
sigma = 1
k = 0.15
size of patch = 10
```

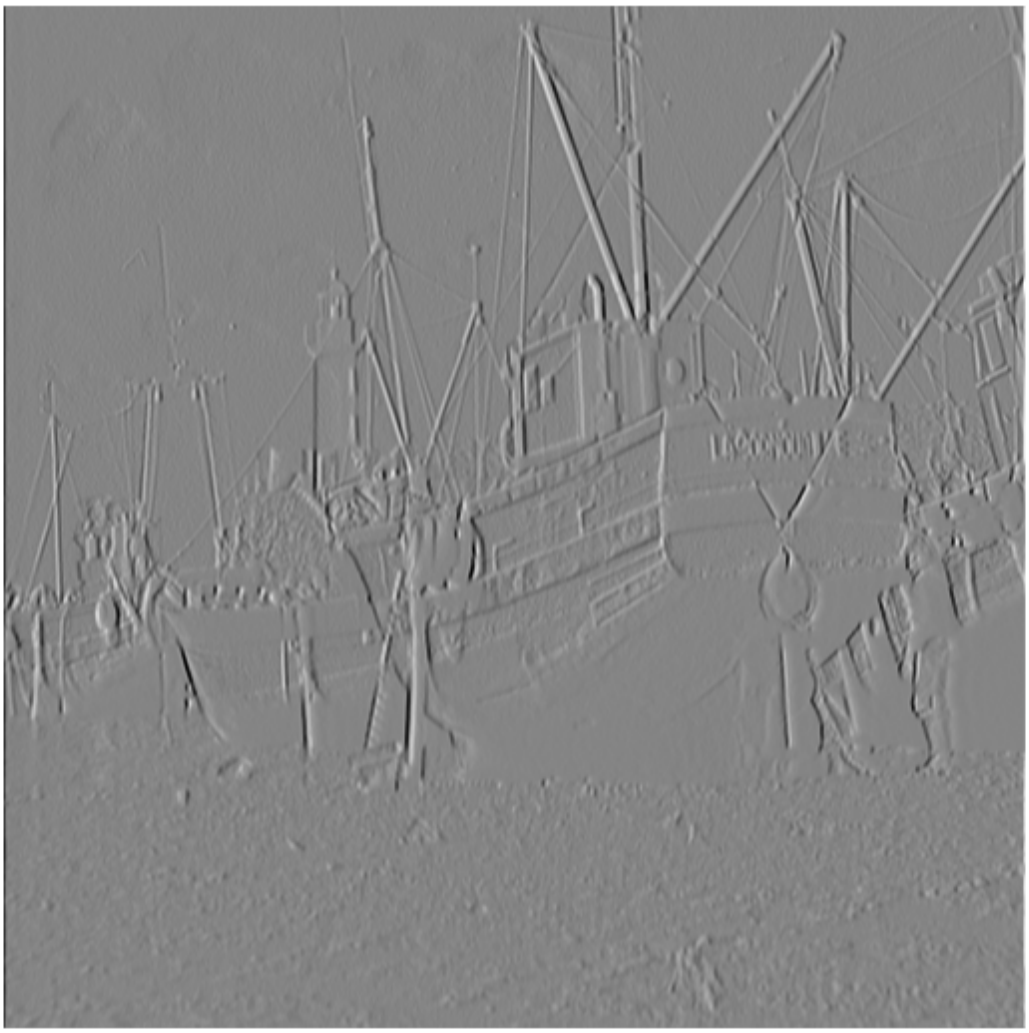
OriginalImage

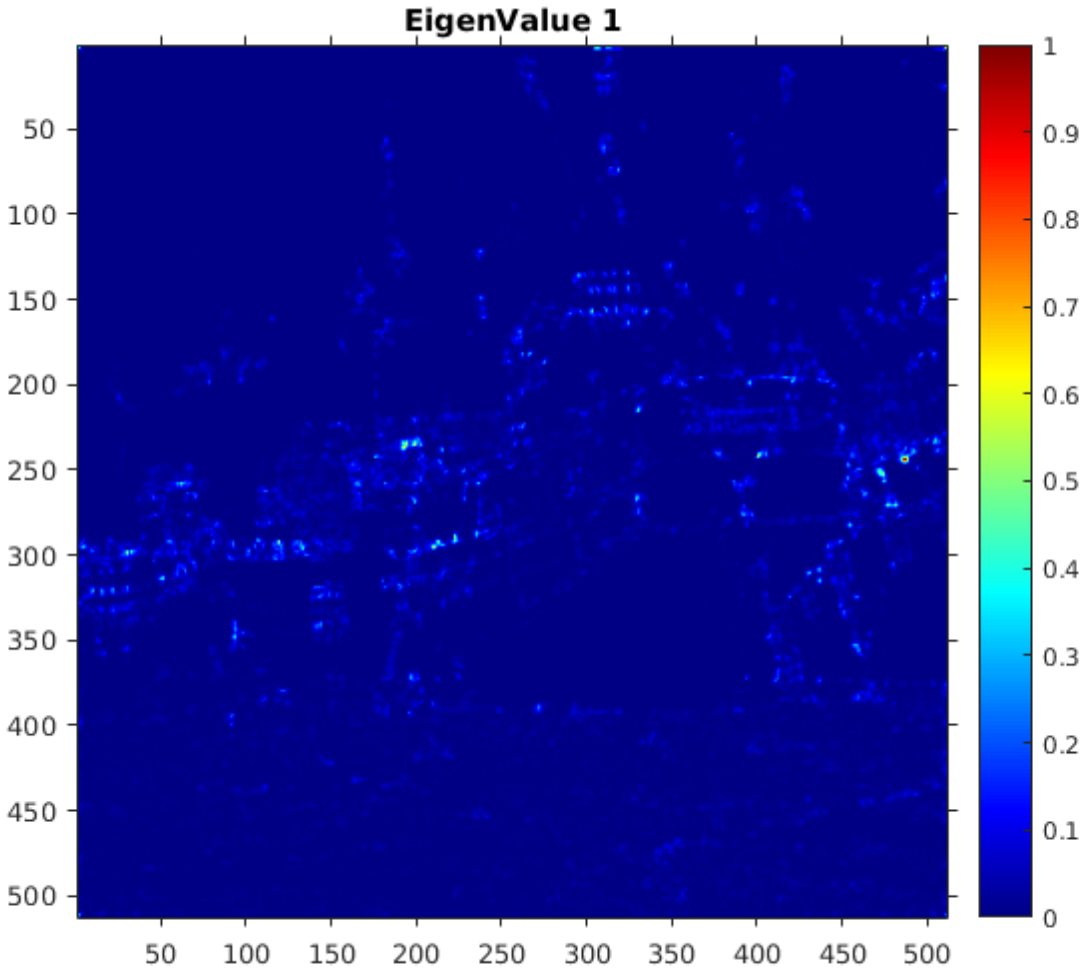
Y Derivative

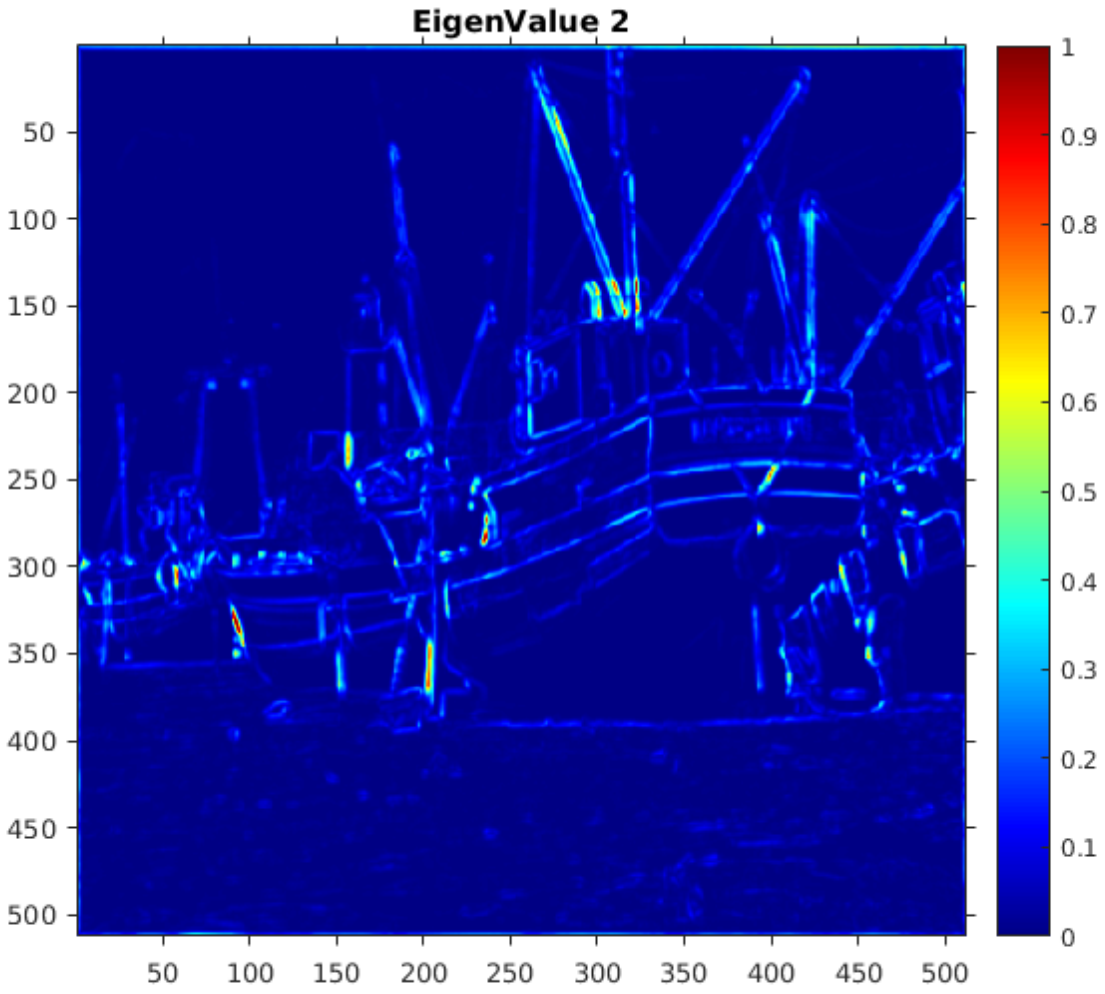


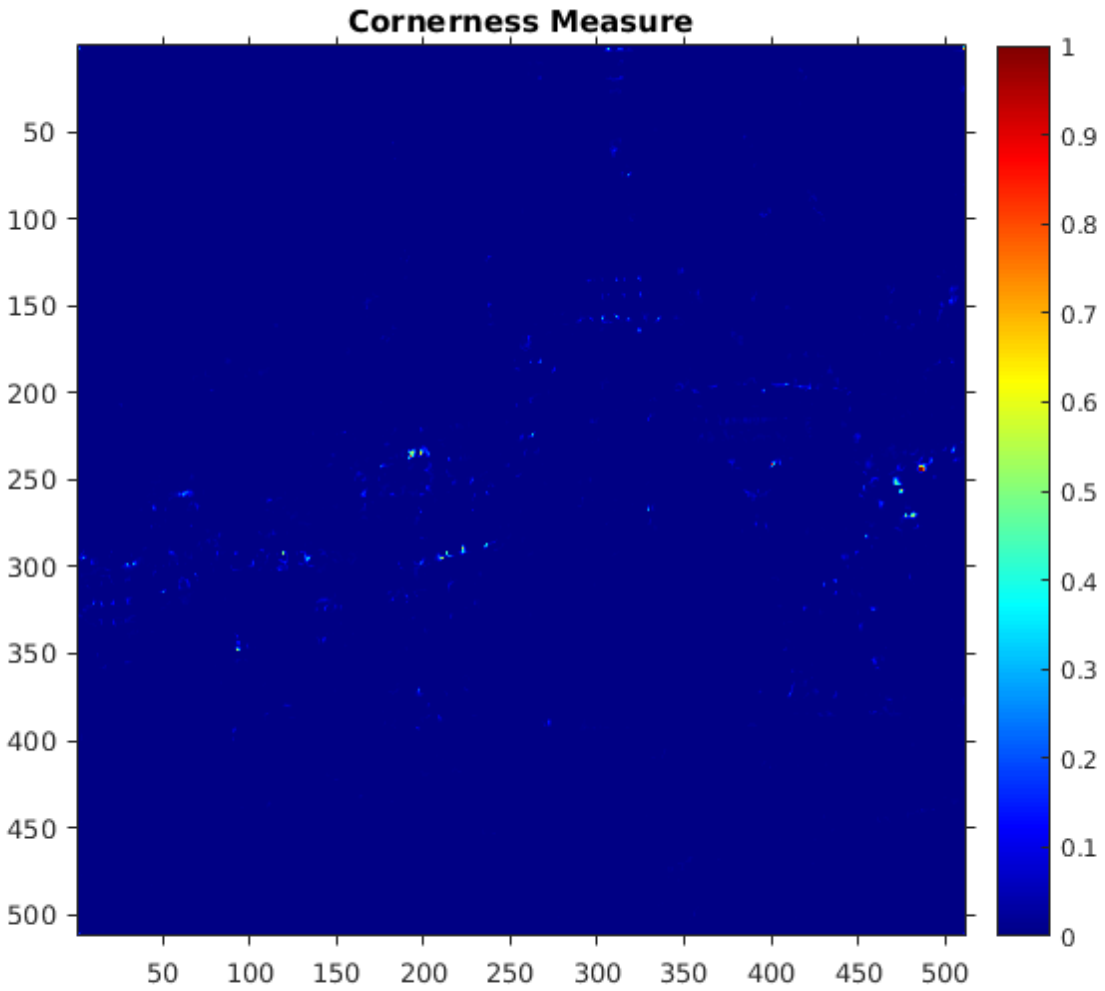


X Derivative




EigenValue 1
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


EigenValue 2
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0



Published with MATLAB® R2015b

Contents

- MyMainScript
- Baboon Color

MyMainScript

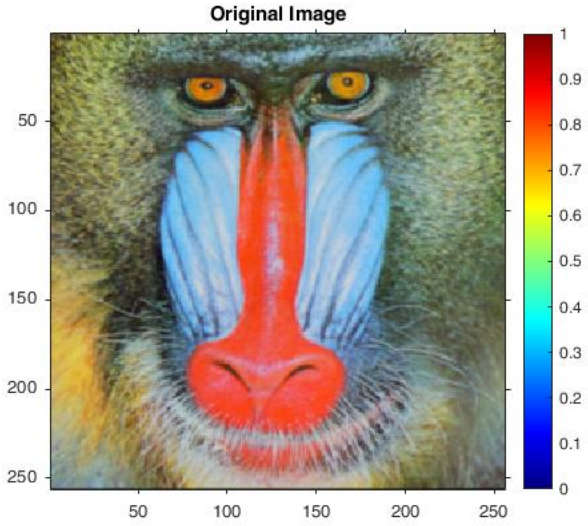
```
tic;
```

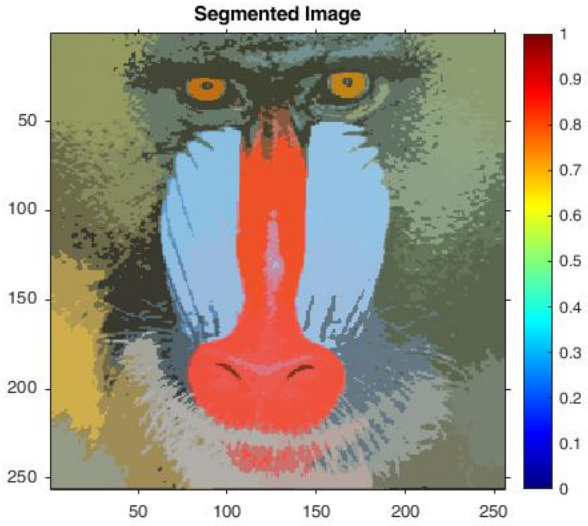
Baboon Color

```
imageOrig = imread('../data/baboonColor.png');
imageOrig = imgaussfilt(imageOrig, 1.0);
imageOrig = imresize(imageOrig, 0.5);

h1 = 250;
h2 = 250;
rate = 1.0;

segmentImage = myMeanShiftSegmentation(imageOrig, h1, h2, rate);
myDisplayTwoImage(imageOrig, segmentImage);

toc;
```

```
Elapsed time is 503.220639 seconds.
```

```
Gaussian kernel bandwidth for color feature : 250
Gaussian kernel bandwidth for spatial feature : 250
Number of iterations : 15
Number of nearest neighbours : 1500
```





Published with MATLAB® R2018a

Problem 1

myDisplayImage.m

```
function myDisplayImage(img,title1)
   figure;
   imshow(img);
   colormap(jet(200));
   axis on;
   colorbar;
   title(title1);
end
```

myHarrisCornerDetector.m

```
function resultHCD = myHarrisCornerDetector(inputImage, sigma, k, patch)

filterForGradient = fspecial('sobel');
Ix = imfilter(inputImage, filterForGradient);
Iy = imfilter(inputImage, filterForGradient');

wSize = patch/2;
windowFilter = fspecial('gaussian', 2*wSize+1, sigma);
Ixx = imfilter(Ix.^2, windowFilter);
Iyy = imfilter(Iy.^2, windowFilter);
Ixy = imfilter(Ix.*Iy, windowFilter);
[hsize, vsize] = size(inputImage);
eigenValues = zeros([hsize vsize 2]);
resultHCD = zeros([hsize vsize]);

for i = 1:hsize
   for j = 1:vsize
      resultHCD(i,j) = det(Ixx(i,j).*Iyy(i, j) - Ixy(i,j).^2) - k*(Ixx(i,j)+Iyy(i, j)).^2;
      eigenValues(i,j,:) = eig([Ixx(i,j) Ixy(i,j); Ixy(i, j) Iyy(i, j)])';
   end
end

figure,imshow(reScale(Ix));
title('Y Derivative');
```

```
figure,imshow(reScale(Iy));
title('X Derivative');
myDisplayImage(reScale(eigenValues(:,:,1)),'EigenValue 1');
myDisplayImage(reScale(eigenValues(:,:,2)),'EigenValue 2');
myDisplayImage(resultHCD,'Cornerness Measure');
end
```

myMainScript.m

```
%% MyMainScript

tic;
%% Harris Corner Detector
image  = load('../data/boat.mat');
inputImage = reScale(image.imageOrig);

figure, imshow(inputImage);
title('OriginalImage');
myHarrisCornerDetector(inputImage, 1, 0.15, 10);
toc;
```

reScale.m

```
function output = reScale(inputImage)
   minVal = min(min(inputImage));
   maxVal = max(max(inputImage));
   output = (double(inputImage)-minVal)/(maxVal-minVal);
end
```

Problem 2

myDisplayImageColor.m

```
function myDisplayImageGrey(img,title1)
   imshow (uint8(img));
   title(title1);
         colormap(jet(200));
         axis on;
         colorbar;
```

myDisplayTwoImage.m

```
function myDisplayTwoImage(img1,img2)
   figure('Position', [100, 100, 1200, 600]);
   subplot(1,2,1);
   myDisplayImageColor(img1,'Original Image');
   subplot(1,2,2);
   myDisplayImageColor(img2,'Segmented Image');
end
```

myMainScript.m

```
%% MyMainScript

tic;
%% Baboon Color
imageOrig = imread('../data/baboonColor.png');
imageOrig = imgaussfilt(imageOrig, 1.0);
imageOrig = imresize(imageOrig, 0.5);

h1 = 250;
h2 = 250;
rate = 1.0;

segmentImage = myMeanShiftSegmentation(imageOrig, h1, h2, rate);
myDisplayTwoImage(imageOrig, segmentImage);
```

```
toc;
```

myMeanShiftSegmentation.m

```
%% Mean Shift Segmentation
function segmentedImage = myMeanShiftSegmentation(img, h1, h2, rate)
	[len, wid, c] = size(img);

	colorMatrix(:,:,:) = double(img);

	% Preparing the distance feature
	distanceMatrix(:,:,1) = double(repmat((1:wid), len, 1));
	distanceMatrix(:,:,2) = double(repmat((1:len)', 1, wid));

	% featureMatrix is len * wid * 5 dimensional matrix, with 1:3 channels being of color
	% 4:5 channels being for distance
	featureMatrix(:,:,1:c) = colorMatrix;
	featureMatrix(:,:,c+1:c+2) = distanceMatrix;

	% Flattening out the featureMatrix to shape (len * wid) * 5
	columnFeatureMatrix = reshape(featureMatrix, len*wid, c+2);

	% Storing the gradients of the points
	gradientMatrix = zeros(size(columnFeatureMatrix));

	diagH1 = diag(h1*ones(c, 1));
	diagH2 = diag(h2*ones(2, 1));

	for t=1:15
		% Finding the nearest neighbours over which the gradient would be computed
		knnIndexes = knnsearch(columnFeatureMatrix, columnFeatureMatrix, 'K', 1500);
		for i=1:len*wid
				knnPoints = columnFeatureMatrix(knnIndexes(i, :), :);
				colorWeights = mvnpdf(knnPoints(:, 1:c), columnFeatureMatrix(i,
1:c), diagH1);
				distanceWeights = mvnpdf(knnPoints(:, c+1:c+2),
columnFeatureMatrix(i, c+1:c+2), diagH2);

				finalWeights = colorWeights .* distanceWeights;
```

```
                meanPoints = sum(bsxfun(@times, knnPoints, finalWeights), 1) /
sum(finalWeights);
                gradientMatrix(i, :) = meanPoints;
            end
            % Gradient ascent
            columnFeatureMatrix = columnFeatureMatrix*(1-rate) + rate*gradientMatrix;
        end
        segmentedImage = reshape(uint8(columnFeatureMatrix), [len, wid, c+2]);
        segmentedImage = segmentedImage(:,:, 1:c);
end
```