CS 663: Assignment 3

Group Members:- Rishabh Shah 150050006
Anmol Mishra 150010041
Shriram SB 150050099

Notes :-

- Since none of our team members were able to register in Turnitin, we are attaching our code and (fast to generate images).
- The published part contains output for Q2 and scripts for both the parts.
- Individual ***question/report*** directories contain in-depth report for each question
- We also have attached boat.mat file for Q1.

Table of Contents

MyMainScript

```
load('../data/boat.mat');
inputImage = mat2gray(imageOrig);
sigma_grad = 1;
sigma_weights = 3;
k = 0.05;
[derivatives, eigenvalues, cornerness] =
 myHarrisCornerDetector(inputImage, sigma_grad, sigma_weights, k);
```

Plotting the derivative images

```
tic;
myNumOfColors = 256;
myColorScale = [[0:1/(myNumOfColors-1):1]' ,[0:1/
(myNumOfColors-1):1]', [0:1/(myNumOfColors-1):1]'];

h = figure;
subplot(1, 2, 1), imagesc(mat2gray(derivatives(:,:,1))),
 title('Derivative along x-axis');
daspect([1 1 1]);
colormap(myColorScale); axis tight; colorbar;

subplot(1, 2, 2), imagesc(mat2gray(derivatives(:,:,2))),
 title('Derivative along y-axis');
daspect([1 1 1]);
colormap(myColorScale); axis tight; colorbar;

waitfor(h);

toc;
```

Plotting the eigenvalues

```
tic;

myNumOfColors = 256;
myColorScale = [[0:1/(myNumOfColors-1):1]' ,[0:1/
(myNumOfColors-1):1]', [0:1/(myNumOfColors-1):1]'];

h = figure;
subplot(1, 2, 1), imagesc(mat2gray(eigenvalues(:,:,1))), title('First
 Eigenvalue');
```

```
daspect([1 1 1]);
colormap(myColorScale); axis tight; colorbar;

subplot(1, 2, 2), imagesc(mat2gray(eigenvalues(:,:,2))), title('Second
 Eigenvalue');
daspect([1 1 1]);
colormap(myColorScale); axis tight; colorbar;

waitfor(h);

toc;
```

Plotting cornerness

```
tic;
myNumOfColors = 256;
myColorScale = [[0:1/(myNumOfColors-1):1]' ,[0:1/
(myNumOfColors-1):1]', [0:1/(myNumOfColors-1):1]'];

h = figure;
subplot(1, 1, 1), imagesc(mat2gray(cornerness)), title('Cornerness');
daspect([1 1 1]);
colormap(myColorScale); axis tight; colorbar;

waitfor(h);

toc;
```

Published with MATLAB® R2018a

```
function [img_derivative, eigenvalues, cornerness] =
 myHarrisCornerDetector(inputImage, sigma_grad, sigma_weights, k)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

img = inputImage;
[img_x, img_y] = imgradientxy(img);         %Gradient along x and y
 directions
grad_kdim = round(3*sigma_grad);            %Size of gradient
 smoothening kernel
grad_kernel = fspecial('gaussian', [grad_kdim grad_kdim], sigma_grad);
img_x = conv2(img_x, grad_kernel, 'same');
img_y = conv2(img_y, grad_kernel, 'same');

img_derivative = cat(3, img_x, img_y);      %Storing the derivatives
 for output

%Calculating gradient products for constructing structure tensor
img_xx = img_x .^ 2;
img_yy = img_y .^ 2;
img_xy = img_x .* img_y;

%Smoothening using the weights i.e. gaussian window in Harris
weights_kdim = round(3*sigma_weights);
weights_kernel = fspecial('gaussian', [weights_kdim weights_kdim],
 sigma_weights);
pixel_xx = conv2(img_xx, weights_kernel, 'same');
pixel_yy = conv2(img_yy, weights_kernel, 'same');
pixel_xy = conv2(img_xy, weights_kernel, 'same');

[num_rows, num_cols] = size(img);
eigenvalues = ones(num_rows, num_cols, 2);
cornerness = ones(num_rows, num_cols);

%For each pixel, constructing the structure matrix,
%to determine its eigenvalues and cornerness
for i = 1:num_rows
    for j = 1:num_cols
        struct_tensor = [pixel_xx(i,j) pixel_xy(i,j); pixel_xy(i,j)
 pixel_yy(i,j)];
        eigenvalues(i,j,:) = eig(struct_tensor);
        cornerness(i,j) = det(struct_tensor) - k *
 trace(struct_tensor) * trace(struct_tensor);
    end
end
end
```

Published with MATLAB® R2018a

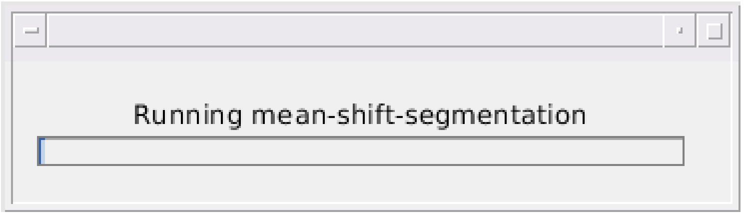
Table of Contents

MyMainScript

```
tic;
```

Preparing Input parameters

```
f = waitbar(0,'PreparingInput');
downsample_factor = 0.25;
h_s = 35;
h_r = 35;
stopping_threshold = 0.01;
waitbar(0.01, f, "Running mean-shift-segmentation");
```



Smoothen the image and downsample

```
img = imread('../data/baboonColor.png');
img = imgaussfilt(img, 1);
img = imresize(img, downsample_factor);
img = double(img);
```

Algorithm

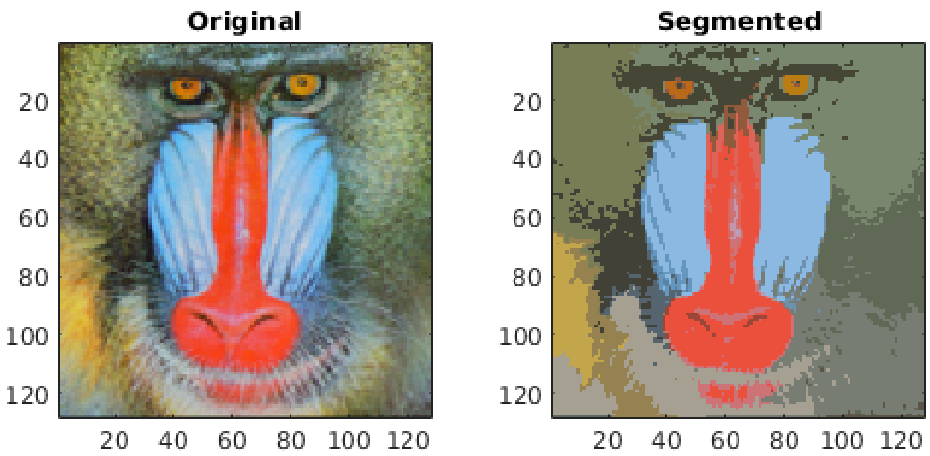
```
[final_img, num_iterations] = myMeanShiftSegmentation(img, h_s, h_r,
 stopping_threshold, f);
[unique_colors, ~, ~] = unique(reshape(uint8(final_img), [],
 3), 'rows');
fprintf('Terminated in %d iterations with %d segments\n',
 num_iterations, size(unique_colors, 1));
delete(f);
```

Terminated in 16 iterations with 25 segments

Plotting all images

```
h = figure;
subplot(1, 2, 1), imagesc(uint8(img)), title('Original');
daspect([1 1 1]);

subplot(1, 2, 2), imagesc(uint8(final_img)), title('Segmented');
daspect([1 1 1]);
```



End of script

```
toc;
```

```
Elapsed time is 146.171003 seconds.
```

Published with MATLAB® R2018a

myMeanShiftSegmentation Code

```
function [segmented_img,num_iterations] = myMeanShiftSegmentation(img,
 h_s, h_r, stopping_threshold, bar)
    % ---INPUT---
    % img               - input RGB image(may be downsampled) to be
 segmented
    % h_s               - spacial bandwidth parameter
    % h_r               - range(color) bandwidth parameter
    % stopping_threshold- value s.t if maximum movement of any pixel
 is
    %                     less than this value then the algorithm is
 said to converge
    % bar               - waitbar to display progress

    % ---OUTPUT---
    % segmented_img     - final RGB image with segmented colors
    % num_iterations    - Number of iterations till convergence


    % ---ALGORITHM---

    % Step 1: Make feature vectors
    [n_rows, n_cols, ~] = size(img);
    features = reshape(img, n_rows * n_cols, 3); %colors R,G,B
    [X, Y] = meshgrid(1:n_rows,1:n_cols);
    features(:,4) = reshape(Y, n_rows * n_cols, 1); %row number
    features(:,5) = reshape(X, n_rows * n_cols, 1); %col number

    % Step 3: Define Kernel
    kernel = @(x, y) exp(-sum((x-y).^2, 2));
    dist = @(x, y) sum((x - repmat(y,1)).^2, 2);
    knn = @(x, y, c) find(dist(x, y) < c);

    % Step 4: Mean Shift Algorithm
    num_iterations = 0;
    while num_iterations < 20
        num_iterations = num_iterations + 1;
        waitbar(double(num_iterations)/20, bar, "Running iteration " +
 int2str(num_iterations));
        err = 0;

        for i=1:n_rows * n_cols
            temp_features = horzcat(features(:,1:3)/h_r ,
 features(:,4:5)/h_s);
            % Step 4.1: knnsearch on feature space to find nearest
 neighbours to pt i
            indices = knn(temp_features, temp_features(i,:), 1);

            % Step 4.2: Calculate weighted mean
            weights = kernel(temp_features(indices,:),
 repmat(temp_features(i,:), length(indices), 1));
```

```
            weights = weights/sum(weights);
            weights = repmat(weights,1,5);
            mean = sum(features(indices,:) .*  weights);

            % Step 4.3: Calulate the shift
            mean_shift = mean - features(i,:);
            err = max(err, sum(mean_shift.^2));

            % Step 4.4: update the point
            features(i,:) = mean;

        end
        % disp(err);
        if (err <= stopping_threshold)
            break
        end
    end

    % Step 5: Transform Final Features to RGB image
    segmented_img = reshape(features(:,1:3), n_rows, n_cols, 3);
end
```

Published with MATLAB® R2018a