Entire code and report - Assignment 3

by Yash Shah

Submission date: 02-Sep-2018 09:32PM (UTC+0800)
Submission ID: 991587069
File name: turnitin.txt (11.38K)
Word count: 1463
Character count: 9302

```
%% MyMainScript

tic;

%% Your code here

    myHarrisCornerDetector();

toc;

function [] = myHarrisCornerDetector()

S = load('../data/boat.mat');

I=S.imageOrig; %load the original image

% min(min(I)) = 0 so no need of reshifting

I = (I)/(max(max(I))); % make the intensity range from 0 to 1

% initialize parameters for tuning

sigma_1 = 1; % sigma_1 is the weight gaussian

sigma_2 = 1; % sigma_2 is the gaussian filtering for original image to get Ix,Iy
```

```
% filter the image before extracting the derivatives

I = imgaussfilt(I,sigma_2);

[Ix,Iy] = imgradientxy(I);

% copmute the Ix^2, Iy^2 ,IxIy matrices before computing the structure

% tensor

Ix2 = Ix.^2;

figure;

imshow(mat2gray(Ix)),colorbar;

title('Ix');

Iy2 = Iy.^2;

figure;

imshow(mat2gray(Iy)),colorbar;

title('Iy');
```

```
Ixly = Ix.*Iy;

% these lines filter Ix2 Iy2 Ixly by another gaussian which is supposed to

% be the weights for when we constrcut strcuture tensors for pixels, thus

% after this step we dont need to worry about the weights we can just add

% the matrices at the neighbourhood of each pixel

g = fspecial('gaussian', [6 6] ,sigma_1);

Ix2 = imfilter(Ix2,g,'same');

Iy2 = imfilter(Iy2,g,'same');

Ixly = imfilter(Ixly,g,'same');

[X,Y] = size(I);

eigen1 = double(zeros(X,Y)); % smaller eigen value

eigen2 = double(zeros(X,Y)); % larger eigen value

Result = double(zeros(X,Y)); % cornerness measure
```

```
k = 0.15; % tuning parameter for measuring

for i=1:X-1

    for j=1:Y-1

        if(i == 1 || j == 1)

            resIx2 = 0;

            resIy2 = 0;

            resIxIy = 0;

            for a=0:1

                for b=0:1

                    resIx2 = resIx2 + Ix2(a+i,b+j);

                    resIy2 = resIy2 + Iy2(a+i,b+j);

                    resIxIy = resIxIy + IxIy(a+i,b+j);

                end
```

```
end

tensor = double(zeros(2,2));

tensor(1,1) = reslx2;

tensor(1,2) = reslxly;

tensor(2,1) = reslxly;

tensor(2,2) = resly2;

%tensor;

e = eig(tensor);

%min(e);

%max(e);

eigen1(i,j) = min(e);

eigen2(i,j) = max(e);
```

```
            Result(i,j) = (eigen1(i,j)*eigen2(i,j)) -
k*(eigen1(i,j)+eigen2(i,j))*(eigen1(i,j)+eigen2(i,j));
        else
            resIx2 = 0;
            resIy2 = 0;
            resIxIy = 0;
            for a=-1:1
                for b=-1:1
                    resIx2 = resIx2 + Ix2(a+i,b+j);
                    resIy2 = resIy2 + Iy2(a+i,b+j);
                    resIxIy = resIxIy + IxIy(a+i,b+j);
                end
            end
```

```
tensor = double(zeros(2,2));

tensor(1,1) = reslx2;

tensor(1,2) = reslxly;

tensor(2,1) = reslxly;

tensor(2,2) = resly2;

%tensor;

e = eig(tensor);

%min(e);

%max(e);

eigen1(i,j) = min(e);

eigen2(i,j) = max(e);

Result(i,j) = (eigen1(i,j)*eigen2(i,j)) -
k*(eigen1(i,j)+eigen2(i,j))*(eigen1(i,j)+eigen2(i,j));
```

```
        end

    end

end

Corners = (Result > 0.35);

max(max(Result));

% display all the results

 figure;

 imshow(mat2gray(eigen1));

 title('Minimum eigen value'),colorbar;

 figure;

 imshow(mat2gray(eigen2));

 title('Maximum eigen value'),colorbar;

 figure;
```

```
 imshow(mat2gray(Result)),colorbar;

 title('Cornerness measure');

 figure;

 imshow(mat2gray(Corners)),colorbar;

 title('Thresholded Cornerness measure');

 figure;

 imshow(mat2gray(Corners+I)),colorbar;

 title('Cornerness measure superimposed on image');

%% Mean-Shift Segmentation

%

% We implemented the Mean-Shift algorithm by first downsampling the image

% (gaussian smoothing with ₩sigma=1 followed by resizing to half the size

% along both dimensions) and then scaling it to the range [0,1].
```

```
%
%% Parameter tuning
%
% cntr = 0;
% for hc=0.005:0.005:0.1
%     for hs=5:5:30
%         res = myMeanShiftSegmentation(img, hc, hs);
%         imwrite(res, ['../images/trial_',num2str(cntr),'.png']);
%         cntr = cntr+1;
%     end
% end
%
%%
```

```
% All resultant images were compared visually. We chose the best output by

% looking for meaningful and distinct (i.e. less interpolation between

% segments) segments, and found the following results:

%%

% variables for displaying images

myNumOfColors = 200;

myColorScale = repmat((0:1/(myNumOfColors-1):1)',1,3);

disp('Original image');

tic;

img = im2single(imread('../data/baboonColor.png'));

% smoothen the image

img = imgaussfilt(img, 1);

% downsample image
```

```
img = imresize(img, 0.5);

% image has been rescaled to [0,1] range

% so hc has to be adjusted accordingly

figure; imagesc(img); title('Original image');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc; tic;

disp('Tuned output, h_c=0.05, h_s=20');

res = myMeanShiftSegmentation(img, 0.05, 20);

figure; imagesc(res); title('Tuned output, h_c=0.05, h_s=20');

colormap(myColorScale);

daspect ([1 1 1]);
```

```
axis tight; axis off;

% save image

save(['../images/tunedBaboon.mat'], 'res');

toc; tic;

disp('Output for h_c=0.2, h_s=20');

res = myMeanShiftSegmentation(img, 0.2, 20);

figure; imagesc(res); title('Output for h_c=0.2, h_s=20');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc; tic;

disp('Output for h_c=0.005, h_s=20');

res = myMeanShiftSegmentation(img, 0.005, 20);
```

```
figure; imagesc(res); title('Output for h_c=0.005, h_s=20');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc; tic;

disp('Output for h_c=0.05, h_s=50');

res = myMeanShiftSegmentation(img, 0.05, 50);

figure; imagesc(res); title('Output for h_c=0.05, h_s=50');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc; tic;

disp('Output for h_c=0.05, h_s=5');
```

```
res = myMeanShiftSegmentation(img, 0.05, 5);

figure; imagesc(res); title('Output for h_c=0.05, h_s=5');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc;

%% Observations

%

% We can clearly see (from the output images generated by using

% neighborhood parameter values) that the chosen values are optimal. High

% values of h_c and h_s favour intermixing of segments (diffused boundaries)

% whereas lower values don't produce any significant changes.

function res = myMeanShiftSegmentation(img, hc, hs)
```

```
% Parameters

% #iterations till 'assumed' convergence

num_iters = 20;

% amount by which we move towards the peak everytime

step_size = (hc*hs)^2/(hc^2+hs^2);

% prepare data structure for storing pixel-wise information

[H,W,C] = size(img);

ds = zeros([H,W,C+2]);

ds(:,:,1:C) = img;

ds(:,:,C+1) = repmat(1:W, [H,1]);

ds(:,:,C+2) = repmat((1:H)', [1, W]);

ds = reshape(ds, [H*W, C+2]);

% allocate space for storing calculation results
```

```
num_nn = floor(sqrt(H*W));

diff = zeros([H*W, C+2, num_nn]);

fx = zeros([H*W, num_nn]);

sumfx = zeros([H*W, 1]);

grad_c = zeros([H*W, C]);

grad_s = zeros([H*W, 2]);

% waitbar for showing #iterations completed

h = waitbar(0, "Iterations completed");

for i=1:num_iters

    % find nearest neighbors

    idx = knnsearch(ds, ds, 'K', num_nn);

    % fill the diff matrix

    diff(:,:,:) = ds - permute(reshape(ds(idx', :)', [C+2, num_nn, H*W]), [3 1 2]);
```

```
% find the kernel weights for each pixel

fx(:,:) = exp(-squeeze(sum(diff(:, 1:C, :).^2/hc^2, 2))).*exp(-squeeze(sum(diff(:,
C+1:C+2, :).^2/hs^2, 2)));

% sum up the weights for all neighbors

sumfx(:,:) = sum(fx, 2);

% find the gradient for color domain

grad_c(:,:) = -squeeze(sum(reshape(fx, [H*W, 1, num_nn]).*diff(:,1:C,:)/hc^2, 3));

grad_c(:,:) = grad_c./sumfx;

% find the gradient for spatial domain

grad_s(:,:) = -squeeze(sum(reshape(fx, [H*W, 1, num_nn]).*diff(:,C+1:C+2,:)/hs^2, 3));

grad_s(:,:) = grad_s./sumfx;

% update the pixel data structure

ds(:,1:C) = ds(:,1:C) + step_size*grad_c;
```

```
    ds(:,C+1:C+2) = ds(:,C+1:C+2) + step_size*grad_s;

    waitbar(i/num_iters, h);

end

close(h);

res = reshape(ds(:,1:C), [H,W,C]);

end
```

Question1

August 2018

1

Findings

optimal parameters :- σ 1 = σ 2 = 1, k = 0.15 and threshold parameter = 0.35.

The tuned output is in images folder where all the asked images have been

provided in the report folder there are 4 folders. One of them corresponds to tuned output the rest show how deviating from the tuned output the results get worse, particularly when we decrease k or threshold parameter more and more edges start showing up as corner, so we have set a high k so that we get less edges wothout compromising the corners.

Mean-Shift Segmentation

We implemented the Mean-Shift algorithm by first downsampling the image (gaussian smoothing with ₩sigma=1 followed by resizing to half the size along both dimensions) and then scaling it to the range [0,1]. The optimum parameter values found are: num_iteration = 20, Gaussian kernel bandwidth for the color feature = 0.05, Gaussian kernel bandwidth for the spatial feature = 20

Contents

Parameter tuning

Observations

Parameter tuning

```
cntr = 0;

for hc=0.005:0.005:0.1

    for hs=5:5:30

        res = myMeanShiftSegmentation(img, hc, hs);

        imwrite(res, ['../images/trial_',num2str(cntr),'.png']);

        cntr = cntr+1;

    end

end
```

All resultant images were compared visually. We chose the best output by looking for meaningful and distinct (i.e. less interpolation between segments) segments, and found the following results:

```
% variables for displaying images
```

```
myNumOfColors = 200;

myColorScale = repmat((0:1/(myNumOfColors-1):1)',1,3);

disp('Original image');

tic;

img = im2single(imread('../data/baboonColor.png'));

% smoothen the image

img = imgaussfilt(img, 1);

% downsample image

img = imresize(img, 0.5);

% image has been rescaled to [0,1] range

% so hc has to be adjusted accordingly

figure; imagesc(img); title('Original image');

colormap(myColorScale);
```

```
daspect ([1 1 1]);

axis tight; axis off;

toc; tic;

disp('Tuned output, h_c=0.05, h_s=20');

res = myMeanShiftSegmentation(img, 0.05, 20);

figure; imagesc(res); title('Tuned output, h_c=0.05, h_s=20');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

% save image

save(['../images/tunedBaboon.mat'], 'res');

toc; tic;

disp('Output for h_c=0.2, h_s=20');
```

```
res = myMeanShiftSegmentation(img, 0.2, 20);

figure; imagesc(res); title('Output for h_c=0.2, h_s=20');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc; tic;

disp('Output for h_c=0.005, h_s=20');

res = myMeanShiftSegmentation(img, 0.005, 20);

figure; imagesc(res); title('Output for h_c=0.005, h_s=20');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc; tic;
```

```
disp('Output for h_c=0.05, h_s=50');

res = myMeanShiftSegmentation(img, 0.05, 50);

figure; imagesc(res); title('Output for h_c=0.05, h_s=50');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;

toc; tic;

disp('Output for h_c=0.05, h_s=5');

res = myMeanShiftSegmentation(img, 0.05, 5);

figure; imagesc(res); title('Output for h_c=0.05, h_s=5');

colormap(myColorScale);

daspect ([1 1 1]);

axis tight; axis off;
```

toc;

Original image

Elapsed time is 0.093338 seconds.

Tuned output, h_c=0.05, h_s=20

Elapsed time is 79.566733 seconds.

Output for h_c=0.2, h_s=20

Elapsed time is 80.835150 seconds.

Output for h_c=0.005, h_s=20

Elapsed time is 86.300257 seconds.

Output for h_c=0.05, h_s=50

Elapsed time is 82.222428 seconds.

Output for h_c=0.05, h_s=5

Elapsed time is 80.574358 seconds.

Observations

We can clearly see (from the output images generated by using neighborhood parameter values) that the chosen values are optimal. High values of h_c and h_s favour intermixing of segments (diffused boundaries) whereas lower values don't produce any significant changes.

Published with MATLAB® R2018a

Entire code and report - Assignment 3

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27