MAEG5720

Computer Vision in Practice

Assignment 2

Date: 10th October 2021

Due Date: 18:30pm 26nd October 2021

Submission Guidelines

The folder you hand in must contain the following:

README.txt – contains anything about the assignment that you want to tell the TA, including a brief introduction of the usage of the code.

Code/ - directory containing all your code (**only .m files and image files allowed**) for this assignment, which is expected to have at least one .m file along with one .jpg or .png files.

Report.pdf – **only 1 document** showing the question number, result and comments.

Rename the folder as <your student ID>-Asgn2, and compress it into <your student ID>Asgn2.zip, and upload it to the blackboard system. (For example, 1155123456-Asgn2/ and 1155123456Asgn2.zip, pay attention to the name.)

Please read the guidelines CAREFULLY. If you fail to meet the deadline because of a submission problem on your side, marks will still be deducted.

The late submission policy is as follows:

- 1 day late: -20 marks

- 2 days late: -40 marks

- 3 days late: -100 marks

Pay attention to the format before. **10% deduction for every wrong format** (filename, function name, etc.).

MAEG5720

Computer Vision in Practice

(1) A box filter can be used for smoothing the image. Your job is to define 5x5 averaging filter.

i.e. $average\_filter = \frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$ to smooth an image.

a) The main.m is the main function of your program. In your main.m file, read an image and define the average filter in MATLAB and finally display the result. (10 points)

b) In MATLAB, you can define your algorithm in function using the following syntax [out] = function_name[arg1, arg2, …]. For Example

```
[out_image] = box_filter[input_image, filter];
```

And save the function in the same directory as box_filter.m. Then you can repeatedly call your function by passing the arguments.

Please write your own function (without using MATLAB conv2 and imfilter) to perform convolution of the image with the filter and then return the output image. You need to pad your image before taking the convolution, you may use MATLAB padarray function to do it (10 points)

c) Gaussian filter is one of the common filters we use to filter the image. You may define your own Gaussian filter

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

Use the function you defined in (b), shows and comment the effect of different kernel size on the image. (10 points)

d) Find the different of original image and the smoothened image and display it. (10 points)

Hints: you can cast the image into double by img2=double(img) for calculation. To display the image, you need to cast the imagine to uint8 by imshow(uint8(img));

e) Smoothing the image takes away the high frequency signal. Sharpening can be done by adding the high frequency signal to the original image. Please sharpen the image using the output in (d) (10 points)

MAEG5720

Computer Vision in Practice

(2) Edge detection of digital image can be performed using convolution. Prewitt and Sobel are two common edge detectors you can perform to a digital image using box filter as follow:

Prewitt Filter $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Sobel Filter $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

(a) Write you own function [out] = prewitt_filter(img, threshold) without using MATLAB ***imfilter*** function. You may use your box_filter function in Q1 or using the ***conv2*** function to do it. (10 points)

(b) Write you own function [out] = sobel_filter(img, threshold) without using MATLAB ***imfilter*** function. You may use your box_filter function in Q1 or using the ***conv2*** function to do it. (10 points)

(3) Write your own Canny-edge detector function [out] = canny[img, high_threshold, low_threshold, sigma]

(a) Read your image and convert it into gray scale image.

(b) Smooth the $image$ with Gaussian filter (with sigma value say 1.0) (10 points)

(c) Compute the G_x, G_y. You may use the functions in question 1 (10 points)

(d) Compute the $mag(i,j) = sqrt(Gx(i,j)^2 + Gy(i,j)^2)$ and $angle(i,j) = tan^{-1}(\frac{Gy(i,j)}{Gx(i,j)})$ (10 points)

(e) Perform non-maximum suppression. You may quantize the angle into 4 cases. To simply the calculate you can just compare the neighbouring pixel along edge normal. If it is not maximum among is neighbours, set it to zero. (10 points)

(f) Classify the pixels into strong edge and weak edge

$high_{threshold} = \max(mag) * high\_threshold$

$low_{threshold} = \max(mag) * high\_threshold * low\_threshold$

For each strong edge pixel, search the neighbouring weak edges and set to strong edge pixel .Search until no more weak edge found. You may need to define a search function such as

For all pixel(i,j) classified as strong edge, find_connected_weak_edge(mag, i, j);

Function[mag] = find_connected_weak_edge(mag, r, c)

```
for i=row-1:row+1
  for j=col-1:col+1
     if not out of bound
        if mag(i, j) is a weak_edge
          set it to strong edge
          find_connected_weak_edge(mag, i, j )
         end
      end
   end
```

Display the image before and after double threshold. You may try to adjust the threshold value for the best result. (20 points)