SIT789 – Robotics, Computer Vision and Speech Processing

Pass Task 2.1: Image filtering and edge detection

Objectives

The objectives of this lab include:

- Practising with different kinds of filters used in image filtering and getting experiences with the use of image filters
- Practising with the edge detection using the Sobel filters and the Canny edge detector

---

Tasks

1. Image filtering with OpenCV

First, you need to copy the supplied image empire.jpg (in OnTrack) into your working directory. You then launch your jupyter notebook and navigate to your working directory. In the first task, you are to convert the image in empire.jpg to grayscale and then filter the grayscale image using a 5x5 averaging kernel defined as,

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Intuitively, this kernel calculates the filtered value for a pixel by averaging the intensity values of the neighbouring pixels of that pixel. The following code can be used for image filtering with the above kernel *K*.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('empire.jpg') #load image
img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

avg_kernel = np.ones((5,5),np.float32) / 25 #kernel K defined above
avg_result = cv.filter2D(img_gray,-1,avg_kernel) # 2nd parameter is always set to -1

plt.imshow(img_gray, 'gray')
```

In week 2, you have studied different linear filters. Your task here is to investigate common linear separable kernels including the **Gaussian** and **corner** kernels (week 2 handouts). You are to apply those kernels on the grayscale image img_gray and write the filtered results into image files.

Note

1. To clearly see the effect of the filters, you should save the results into files and view them using image viewer software, e.g., MS Paint.
2. Kernels can be represented by 2D matrices and you can find examples of defining 2D matrices in part 2 of task 1.2.

Also in week 2, you have studied non-linear filters, e.g., **median filter**, **bilateral filter**. To experiment with those filters, you are given an image with synthetic-noise in empire_shotnoise.jpg file (in OnTrack). You need to download this image file, convert it to grayscale and apply median and bilateral filter on the grayscale version using the following code.

```
#Testing median filter
img_noise = cv.imread('empire_shotnoise.jpg')
img_noise_gray = cv.cvtColor(img_noise, cv.COLOR_BGR2GRAY)

ksize = 5 # neighbourhood of ksize x ksize; ksize must be an odd number
med_result = cv.medianBlur(img_noise_gray, ksize)

plt.imshow(med_result, 'gray')
```

```
#Testing bilateral filter
rad = 5 #radius to determine neighbourhood
sigma_s = 10 #standard deviation for spatial distance (slide 21 in week 2 handouts)
sigma_c = 30 #standard deviation for colour difference (slide 21 in week 2 handouts)
bil_result = cv.bilateralFilter(img_noise_gray, rad, sigma_c, sigma_s)

plt.imshow(bil_result, 'gray')
```

Now, you are to apply the Gaussian filter on img_noise_gray, observe its result and compare the result of the Gaussian filter with that of median filter and bilateral filter.

2. Edge detection

2.1. Edge detection using Sobel kernels

As presented in week 2 handouts, we can use image gradients to detect edges. In particular, we can calculate the horizontal and vertical derivatives of img_gray using the following code.

```
D_x = np.float32([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]) / 8
der_x = cv.filter2D(img_gray, cv.CV_32F, D_x) # CV_32F is used to store negative values

plt.imshow(der_x, 'gray')
```

```
D_y = np.float32([[-1, -2, -1], [0, 0, 0], [1, 2, 1]]) / 8
der_y = cv.filter2D(img_gray, cv.CV_32F, D_y) # CV_32F is used to store negative values

plt.imshow(der_y, 'gray')
```

To get the edges of img_gray, we can calculate the gradient magnitude of img_gray as follows

```
import math

height, width = img_gray.shape
mag_img_gray = np.zeros((height, width), np.float32) # initialise the gradient magnitud
e image with 0s

for i in range(0, height):
    for j in range(0, width):
        square_der_x = float(der_x[i, j]) * float(der_x[i, j])
        square_der_y = float(der_y[i, j]) * float(der_y[i, j])
        mag_img_gray[i, j] = int(math.sqrt(square_der_x + square_der_y))

plt.imshow(mag_img_gray,'gray')
```

2.2. Edge detection with Canny edge detector

You have seen how to detect image edges using the Sobel filters. Now, you practise with the Canny edge detector. Take img_gray as an example, edges of this image can be detected using the Canny edge detector as follows,

```
minVal = 100 # minVal used in hysteresis thresholding
maxVal = 200 # maxVal used in hysteresis thresholding
Canny_edges = cv.Canny(img_gray, minVal, maxVal)

plt.imshow(Canny_edges, 'gray')
```

You can vary the values of minVal and maxVal to see their impact on the quality of edge detection.

Submission instructions

1. Complete the supplied answer sheet with the results of the Gaussian, corner, median, and bilateral filters (Section 1), and the horizontal and vertical derivative images, gradient magnitude image, and the result of the Canny edge detector (Section 2).
2. Compare the edge detection results by the Sobel filters and the Canny edge detector, and describe your observations in the Discussion section in the supplied answer sheet.
3. Submit the answer sheet (in pdf format) to OnTrack.
4. Submit your code (in .py format) to OnTrack.