Final Exam

National Taiwan Normal University

Robot Vision

Marks

_____ out of 100

Without Answers

Examiners: 包傑奇

Date: 30th May 2022

Time: 14:20 - 17:00

Exam Room: Discord https://discord.gg/CuanKnp5GM

Notes:

- Attempt all questions.
- This is an *open* book and *open* Internet examination. Use of books, notes, laptops, and computers **with Internet connectivity** is *permitted*.
- This exam must be **your own** work. Communication with others via messenger apps, email, phone is strictly **forbidden**.
- Show your work to receive full marks. You must show your reasoning, intermediate steps/calculations to reach the answer.
- Some of the questions may not be solvable, that is it may be impossible to calculate the requested information. In this case, say so in your answer and explain why.
- Submit your exam by printing it to a pdf file (File -> Print or Ctrl-P in most browsers) and then sending the pdf file to the instructor in a private chat message.

Name: 隋嘉銘

Student ID: 61075074H

Submit

Python Programming

This section covers topics in python programming for robot vision.

[1] Opcodes and Program

Given are two list of numbers `opcodes` and `program`.

Implement a function `count_opcodes` that returns the number of times a program executed an opcode listed in `opcodes`.

```
count_opcodes([66,13,0,37,55,35,64,41,30,79,80],[35,94,27]) => 1
count_opcodes([71,75,48,38,29,41,37,0,80,47,96,94,97,48,16,75,93,57,99],[14,55,62,78,79]) => 0
count_opcodes([78,59,83,8,72,34,39,26,66,41,8,97,96,26,5,44],[50,7,72,8,52]) => 3
count_opcodes([81,3,47,4,31,50,46,67,18,99,100,90,16,72,8,5,9],[22,35,50,10,4]) => 2
```

8 marks

```
def count_opcodes( program, opcodes ):
```

```
def count_opcodes(lst1,lst2):

    count = 0

    for i in lst1:
        for j in lst2:
            if i == j:
                count +=1

    return count
```

q1.py

```python
def count_opcodes(lst1,lst2):

    count = 0

    for i in lst1:
        for j in lst2:
            if i == j:
                count +=1

    return count


print(count_opcodes([66,13,0,37,55,35,64,41,30,79,80],[35,94,27]))
print(count_opcodes([71,75,48,38,29,41,37,0,80,47,96,94,97,48,16,75,93,57,99],[14,55,62,78,79]))
print(count_opcodes([78,59,83,8,72,34,39,26,66,41,8,97,96,26,5,44],[50,7,72,8,52]))
print(count_opcodes([81,3,47,4,31,50,46,67,18,99,100,90,16,72,8,5,9],[22,35,50,10,4]))
```

問題 輸出 偵錯主控台 終端機

```
Windows PowerShell
著作權 (C) Microsoft Corporation。保留擁有權利。

安裝最新的 PowerShell 以取得新功能和改進功能！https://aka.ms/PSWindows

PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam> conda activate base
PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam> & C:/ProgramData/Anaconda3/python.exe c:/Users/user
1
PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam> & C:/ProgramData/Anaconda3/python.exe c:/Users/user
1
0
3
2
PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam>
```

[2] Worst-case Runtime Performance (`program`)

The program `count_opcodes` is called with parameter `program` (length 2000) and parameter `opcode` (length 500). The executionn takes 1.6930 seconds.

5 marks

What is the expected runtime of the `count_opcodes` program if the length of parameter `program` is changed to 8000 numbers.

Expected Worst-case Runtime: 0.09049558639526367

```python
import time
import random
def count_opcodes(lst1,lst2):

    count = 0

    for i in lst1:
        for j in lst2:
            if i == j:
                count +=1

    return count

parameter_program = []
parameter_opcode = []
for i in range(8000):
```

```
    parameter_program.append(random.randint(1,100))


for i in range(500):

    parameter_opcode.append(random.randint(1,100))

start_time = time.time()

count_opcodes(parameter_program,parameter_opcode)

end_time = time.time()

print("It spend",end_time-start_time,"s" )
```

q2.py > ...

```
import time
import random
def count_opcodes(lst1,lst2):

    count = 0

    for i in lst1:
        for j in lst2:
            if i == j:
                count +=1

    return count

parameter_program = []
parameter_opcode = []
for i in range(8000):

    parameter_program.append(random.randint(1,100))


for i in range(500):

    parameter_opcode.append(random.randint(1,100))

start_time = time.time()

count_opcodes(parameter_program,parameter_opcode)

end_time = time.time()

print("It spend",end_time-start_time,"s" )
```

問題 輸出 偵錯主控台 終端機

```
Windows PowerShell
著作權（C） Microsoft Corporation。保留擁有權利。

安裝最新的 PowerShell 以取得新功能和改進功能！https://aka.ms/PSWindows

PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam> conda activate
PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam> & C:/ProgramDat
It spend 0.09049558639526367 s
PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam>
```

[3] Worst-case Runtime Performance (`opcode`)

The program `count_opcodes` is called with parameter `program` (length 2000) and parameter `opcode` (length 500). The executionn takes 1.6930 seconds.

6 marks

What is the expected runtime of the `count_opcodes` program if the length of parameter `program` is changed to 8000 numbers **and** the length of the parameter `opcodes` is changed to 2000 numbers.

Expected Worst-case Runtime: 0.36172032356262207

```
import time
import random
def count_opcodes(lst1,lst2):
```

```
    count = 0

    for i in lst1:
        for j in lst2:
            if i == j:
                count +=1

    return count

parameter_program = []
parameter_opcode = []
for i in range(8000):

    parameter_program.append(random.randint(1,100))


for i in range(2000):

    parameter_opcode.append(random.randint(1,100))

start_time = time.time()

count_opcodes(parameter_program,parameter_opcode)

end_time = time.time()

print("It spend",end_time-start_time,"s" )
```

q3.py > ...

```
import time
import random
def count_opcodes(lst1,lst2):

    count = 0

    for i in lst1:
        for j in lst2:
            if i == j:
                count +=1

    return count

parameter_program = []
parameter_opcode = []
for i in range(8000):

    parameter_program.append(random.randint(1,100))


for i in range(2000):

    parameter_opcode.append(random.randint(1,100))

start_time = time.time()

count_opcodes(parameter_program,parameter_opcode)

end_time = time.time()

print("It spend",end_time-start_time,"s" )
```

問題 輸出 偵錯主控台 終端機

```
PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam> & C:/ProgramDa
It spend 0.36172032356262207 s
PS C:\Users\user\Desktop\碩士課程\機器人視覺\final_exam>
```

Seam Carving

This topic includes analysis and implementation of smart zoom using the seam carving algorithm.

[4] Minimum Energy Seam

Given below is a gradient magnitude image. The value for each pixel is the magnitude of the gradient for this pixel in the original image.

15 marks

Show the minimum energy seam from top to bottom in the gradient image below:

Table shows the magnitude of gradient

24	19	73	48	79	24	65	91	68	94	56	81
47	84	89	78	58	64	26	58	77	73	17	98
78	70	97	26	95	86	26	57	76	89	42	21
84	18	19	62	34	33	66	50	25	89	36	83
69	55	15	54	23	96	97	92	67	89	85	24
66	48	13	64	34	89	97	51	33	97	52	28
69	52	95	51	59	75	22	38	94	63	76	52

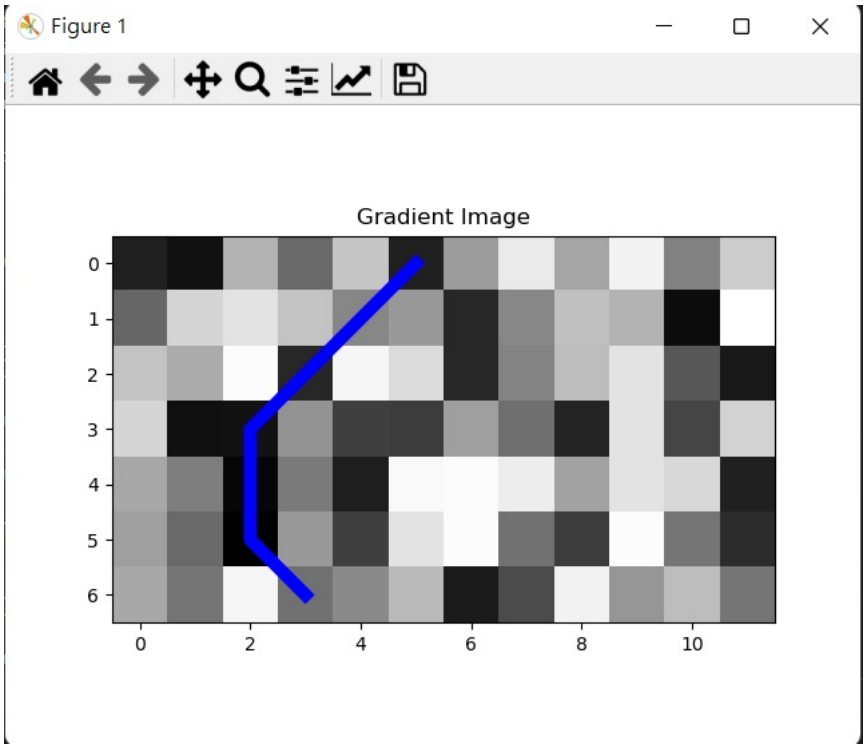
```
import matplotlib.pyplot as plt
import numpy as np
import cv2
grad = np.array([[24,19,73,48,79,24,65,91,68,94,56,81],
                 [47,84,89,78,58,64,26,58,77,73,17,98],
                 [78,70,97,26,95,86,26,57,76,89,42,21],
                 [84,18,19,62,34,33,66,50,25,89,36,83],
                 [69,55,15,54,23,96,97,92,67,89,85,24],
                 [66,48,13,64,34,89,97,51,33,97,52,28],
                 [69,52,95,51,59,75,22,38,94,63,76,52]])
def findMinEnergySeam( grad ):
  height, width = grad.shape
  #print(grad.shape)
  carve = np.zeros((height, width))

  max = None
  carve[0,:] = grad[0,:]
  for y in range(1,height):
    for x in range(width):
      emin = None
      for tx, ty in [ [x-1, y-1], [x, y-1], [x+1, y-1] ]:
        if (tx >= 0) and ( tx < width ) and (ty >= 0 ) and (ty < height ):
          #print('tx',tx,'ty', ty, 'emin', emin, 'g[y,x]', grad[y,x], 'g[ty,tx]', grad[ty,tx] )
          if ( emin is None ) or ( grad[y,x] + grad[ty, tx] < emin ):
            emin = grad[y,x] + grad[ty, tx]
            #print('Setting emin', emin)
      carve[y,x] = emin
      if max is None or emin > max:
        max = emin
  return carve
carve = findMinEnergySeam(grad)
def findPath( carve ):
  height, width = carve.shape
  minIndex = None
  min = None
  for x in range(width):
    if min is None or carve[height-1, x] < min:
      minIndex = x
      min = carve[height-1, x]

  cx = minIndex
  path = [ [ height-1, cx]]

  for y in range( height-2, -1, -1 ):
    min = None
    minX = None
    for dx in [ cx-1, cx, cx+1 ]:
      if ( dx >= 0 ) and ( dx < width ):
        if min is None or carve[y,dx] < min:
          min = carve[y,dx]
          minX = dx
    cx = minX
    path.append([ y, cx ] )
  return path
print("the minimum energy",carve)
minPath = np.array(findPath(carve))
fig = plt.figure()
```

```
ax1 = fig.add_subplot( 1,1,1 )
ax1.set_title( "Gradient Image")
ax1.imshow(grad,cmap = "gray")
ax1.plot( minPath[:,1], minPath[:,0], 'b-', linewidth=7 )
plt.show()
plt.close()
the minimum energy
[[ 24. 19. 73. 48. 79. 24. 65. 91. 68. 94. 56. 81.]
 [ 66. 103. 108. 126. 82. 88. 50. 123. 145. 129. 73. 154.]
 [125. 117. 175. 84. 153. 112. 52. 83. 134. 106. 59. 38.]
 [154. 88. 45. 88. 60. 59. 92. 76. 82. 131. 57. 104.]
 [ 87. 73. 33. 73. 56. 129. 130. 117. 92. 114. 121. 60.]
 [121. 63. 28. 79. 57. 112. 189. 118. 100. 164. 76. 52.]
 [117. 65. 108. 64. 93. 109. 73. 71. 127. 96. 104. 80.]]
```



[5] Minimum Energy and Convolution

The seam carving algorithm returns the following path for an image I_1. The minimum energy seam for image I_1 is given below:

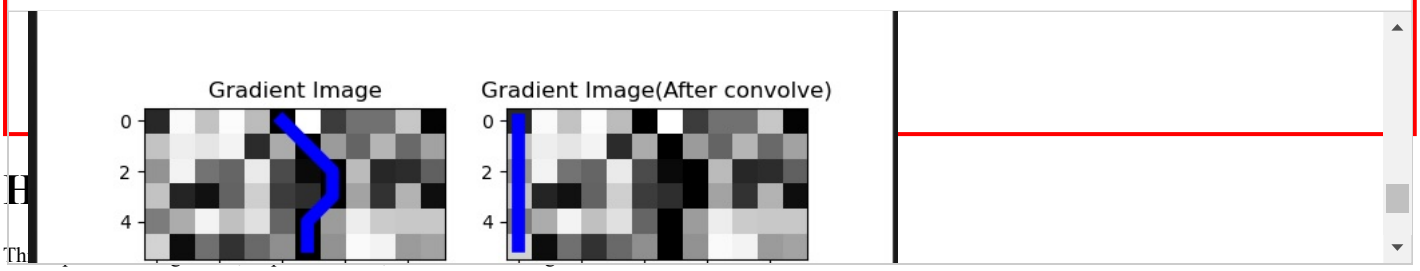
[[5, 6], [4, 6], [3, 7], [2, 7], [1, 6], [0, 5]]

Image I_2 is the image resulting from applying the following convolution kernel to image I_1

0.0	0.0	0.0
0.0	0.0	0.5
0.0	0.0	0.0

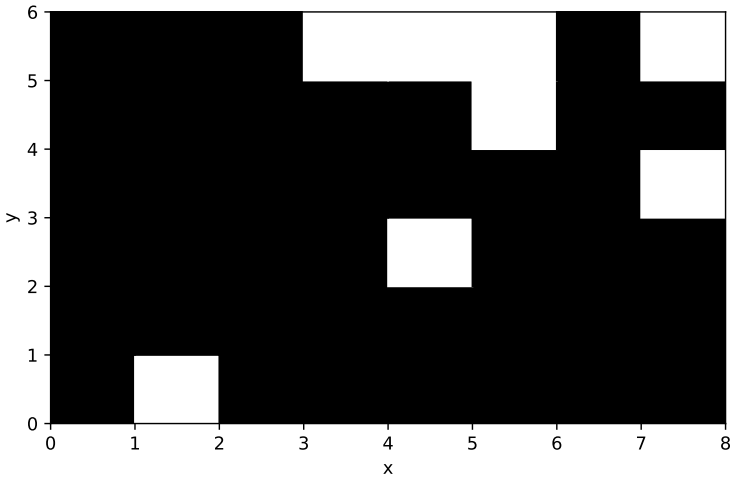
5 marks

Show the new path that will be returned when applying the seam carving to image I_2



[6] Hough Transform

Given is the following edge map of an image as a gray-scale bitmap. The pixel values range from 0 (black) to 255 (white).



Show the resulting Hough space for this input image in the figure below. Use the line-intercept model for lines $y=a*x+b$ with a between -2 to 2 in steps of 1 and b between 0 to 6 in steps of 1. Ignore points that lie outside of the given Hough space.

11 marks

Hough Space:

	-2	-1	0
0			
1			
2			
3			
4			
5			
6			

Show your work here ...

[7] Hough Line Algorithm

Given is the following sample image.

In this question, you will investigate the effect of blurring on the number of lines found in the image using the Hough transform.

Use the OpenCV library to implement the following program.

1. Read the image into your program,
2. apply the Canny edge detector with parameters min=150 and max=500,
3. apply the HoughLines algorithm with a resolution of 5.0 deg for theta. and a resolution of 3.0 for rho. Set the threshold parameter to 100.

15 marks

Given the input image and algorithm as described above, the OpenCV HoughLines algorithm returns 8382 lines.

```
import matplotlib.pyplot as plt
import numpy as np
import cv2


img = cv2.imread("sample_image.png")
gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
rho = 3
theta = np.pi/180*5
threshold = 100
```

```
min=150
max = 500

masked_edge = cv2.Canny(gray,min,max)

line_image = np.copy(img)*0
lines = cv2.HoughLinesP(masked_edge,rho,theta,threshold,np.array([]))

print(len(lines))
```

[8] Hough Lines and Vertical Lines

Extend your solution to the previous question to count the number of vertical lines, that is lines with theta being between -5.0 and 5.0 in the image.

5 marks

There are [ ] lines with theta between -5.0 deg. and 5.0 deg.

Show your work here ...

[9] Gaussian Blur and Hough Lines

Use the Gaussian blur algorithm of the OpenCV library. Use a kernel size of 3 and a sigma of 0.

5 marks

Given the input image and algorithm as described above, the OpenCV HoughLines algorithm after application of a blurring algorithm returns 2886 lines.

```
import matplotlib.pyplot as plt
import numpy as np
import cv2


img = cv2.imread("sample_image.png")
gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
kernel_size = 3
blur_gray = cv2.GaussianBlur(gray,(kernel_size,kernel_size),sigmaX=0,sigmaY=0)
rho = 3
theta = np.pi/180*5
threshold = 100
min=150
max = 500

masked_edge = cv2.Canny(blur_gray,min,max)

line_image = np.copy(img)*0
lines = cv2.HoughLinesP(masked_edge,rho,theta,threshold,np.array([]))

print(len(lines))
```

Histogram of Oriented Gradient (HOG) Algorithm

This section covers topics using the Histogram of Oriented Gradient (HOG) algorithm.

[10] Sobel Edge Detection

Given is the following image.

	Block Col 0	Block Col 1	Block Col 2
Block Row 0	72 131 239 94 186 193 161 173 2	179 123 130 59 51 147 142 41 88	69 14 97 75 50 99 124 114 22
Block Row 1	153 46 27 197 79 10 28 67 48	186 151 239 138 25 76 122 15 28	131 24 142 215 159 216 70 121 41
Block Row 2	134 49 251 62 141 101 39 56 234	101 179 196 204 254 223 231 101 182	46 12 60 38 163 80 238 171 209

6 marks

Using the Sobel edge detector, calculate the **gradient direction** and **gradient magnitude** of the center pixel (pixel[4, 4]=25).

Gradient direction: 245.48892 degrees

Gradient magnitude: 0.5861401

```
img = np.array([[186,151,239],[138,25,76],[122,15,28]])


img = np.float32(img) / 255.0

gx = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=1)
gy = cv2.Sobel(img, cv2.CV_32F, 0, 1, ksize=1)


mag, angle = cv2.cartToPolar(gx, gy, angleInDegrees=True)

print("Gradient direction:\n",angle)
print("Gradient magnitude:\n",mag)
```

```
Gradient direction:
 [[  0.        0.        0.     ]
 [270.      245.48892 270.     ]
 [  0.      180.        0.     ]]
Gradient magnitude:
 [[0.         0.20784312 0.        ]
 [0.2509804  0.5861401  0.827451  ]
 [0.         0.36862746 0.        ]]
```

[11] HOG Feature Descriptor

Using one block (i.e., a 3 by 3 neighborhood), calculate the **HOG feature descriptor** (9 weighted angle values for 20 degree buckets between 0 to 180 degrees) of the center pixel.

Use the absolute of the angle to map it to a direction between 0 to 180 degrees.

Calculate the HOG feature vector without normalization for this question.

6 marks

HOG feature descriptor (without normalization)

{table}

Show your work here ...

[12] Normalized HOG Feature Descriptor

Normalize the HOG feature detector using a 3 by 3 block neighborhood, that is use a total of 9 blocks to normalize the HOG feature desriptor.

13 marks

HOG feature descriptor (with normalization)

Angle 0:20	
Angle 20:40	
Angle 40:60	
Angle 60:80	
Angle 80:100	
Angle 100:120	
Angle 120:140	
Angle 140:160	
Angle 160:180	

Show your work here ...