

**e-Yantra Robotics Competition Plus**

**(eYRC+ Pilot)**

**<Please enter your team id here>**

|  |  |
| --- | --- |
| **Team leader name** | **Dhiraj Gehlot** |
| **College** | **K.J Somaiya College of Engineering** |
| **e-mail** | **dhiraj.gehlot94@gmail.com** |
| **Date** |  |

**Scope of the Task** (7)

1. Describe the algorithm used for solving path planning in this task.

<Teams should write in their own words a description of algorithms used in this task.

You can also draw some diagrams/figures, flowcharts to illustrate the algorithm used.

Answer format: Text

Word-limit: 100 words>

**Camera and Image Processing** (3)

Write down the answers to the following questions. For this part use first image (*test\_image1.png*) in *"Task2\_Practice/test\_images”* folder*.*

1. What is the resolution (size) of the test image?
2. What is the position of the Start point and the End point in the grid in the test image?

(Please refer to the *Task2\_Description.pdf* for the definitions of Start point and End point and answer in (x,y) form, where the x-axis is oriented from left to right and the y-axis is oriented from top to bottom)

1. Draw four shortest paths from the Start point to the End point (you may draw it manually if you desire). An example is shown below:

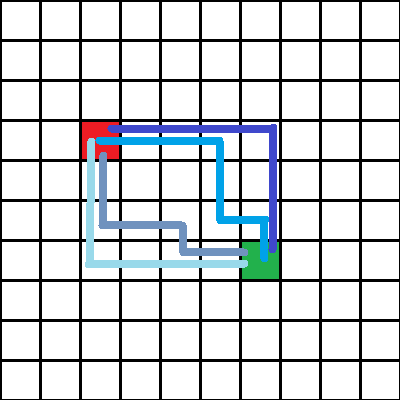


Figure: Example solution with four shortest paths drawn

<

Answer format:

* Answer to question 2 in bulleted form
* Answer to question 3 in bulleted form
* Image pasted for question 4

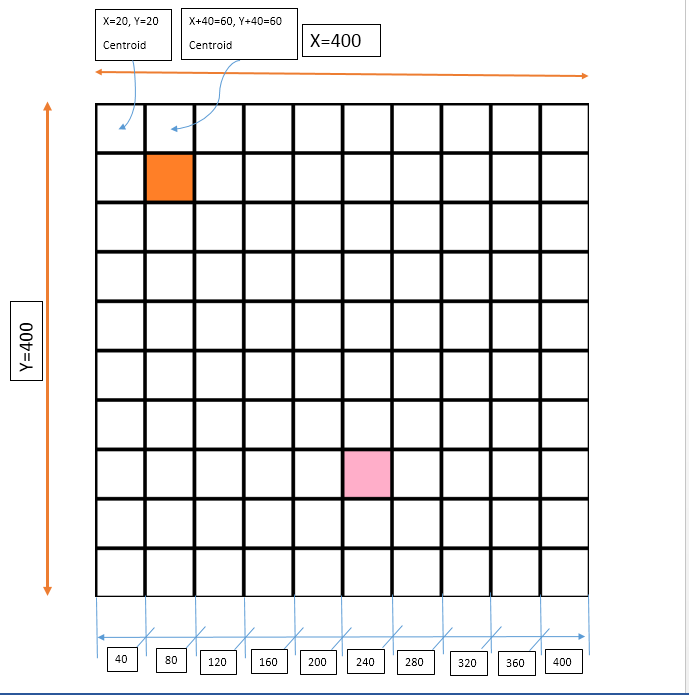
>

**Software used**  (10)

Write down the answers to the following questions. For this part use first image in *"Task2\_Practice/test\_images”* folder*.*

1. Write a function in python to open the image and return an image with a grid of **n** equally spaced horizontal and vertical red lines(RGB values (255, 0, 0)). You are required to write a function *draw\_grid(filename,n)* which takes two arguments:
   1. filename: color image
   2. n: number(integer datatype) of equally spaced horizontal and vertical lines

Output of program should be the image with the specified red grid drawn on it.



''' This code take image from user and return grid image

Syntax : draw\_grid(img,n);

where img : original image on which we want grid

n : Number of equally spaced line

Alogrithm:

1. Take image input

2. Call draw\_grid function with "image" and number of grid "n" as parameter

3. In function divide image of x\*y pixel with n to get x/n and y/n in order to get x and y line coordinates where line will start.

4. Lets take example of 400\*400 image then for 10 lines we get x/n=40 ,y/n =40 , for 10 lines in 400\*400 we have to leave 40 pixels distance in x and y;

5. Draw line from (x=x/n,0) to (x=x/n,400) for first vertical ine, here (0,0) to (0,400) then (40,0) to (40,400) and so on till (400,0) to (400,400)

6. Draw line from (x,y/n) to (400,y/n) for first horizontal ine with above logic

7. return image

'''

import numpy

import cv2

def draw\_grid(img,n):

x,y,c= img.shape; # get number of pixels in x,y of original image;

m=x/n; # get x co-ordinate spaing for vertical line to start

n=y/n; # get y co-ordinate spaing for horizontal line to start

p = 0

while p<y:

cv2.line(img,(p,0),(p,400),(0,0,255),2) # draw vertical line

cv2.line(img,(0,p),(400,p),(0,0,255),2) # draw horizontal line

p += m

return(img) # reurn grid image

img = cv2.imread('test\_image5.png') # image input

gridimage= draw\_grid(img,10); # call grid function

cv2.imshow('grid',gridimage); # display grid image

cv2.waitKey(0)

cv2.destroyAllWindows()

1. Write a function space\_map**(**img**)** in python to detect the layout of the grid as shown in the test image (Figure 1) below. Function space\_map**(**img**)**takes a test image as input and returns a 10x10 matrix called “grid\_map” of integers with values either 0 or 1. Each square must be identified as either navigable space(0), or obstacle(1). The Start and End points are considered as obstacles for this question. An example is shown in Figure 2 below.

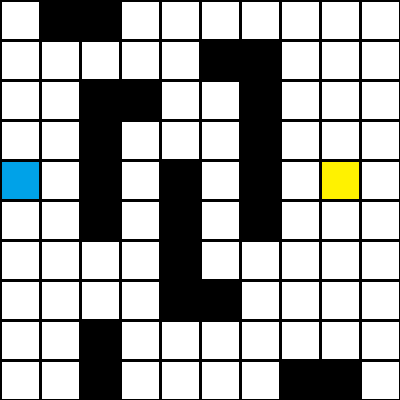
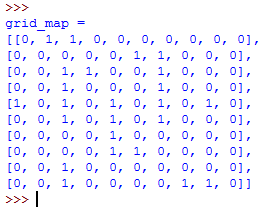
 

Figure 1: Example Test Image Figure 2: Example output

<Answer format:

Use the snippet given below by adding your code after the comment: #add your code here. Inline comments are mandatory to explain the code>

**def** path\_detect**(**img**):**

'''

img-- input color image stored as file

result— output binary image

'''

#add your code here

**return grid\_map**