e-Yantra Robotics Competition Plus
(eYRC+ Pilot)
<Please enter your team id here>

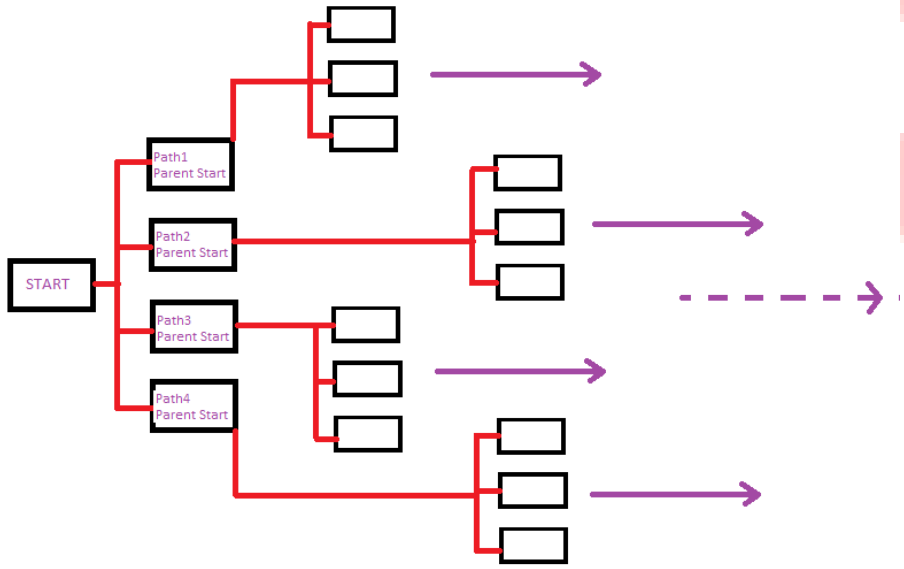
Team leader name	**Suvrat Shankar Chaturvedi**
College	**JSS Academy of Technical Education Noida**
e-mail	**suvrat_chaturvedi@yahoo.com**
Date	**22-12-2014**

Scope of the Task (7)

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

   Algorithm of the task:

   - Algorithm used for solving the planning in the task is the Dijkstra's Algorithm. We start with the start point and look for the all possibilities.
   - Now we create new paths from the existing path keeping into consideration the parent of the new points created.
   - In this way the paths are made and the path first to reach the end point is the shortest path.
   - If there are more than one shortest path this algorithm gives the path which first reaches the end point but more than one path can also be obtained.



Scope of the task:

Artificial intelligence and Robotics: Path as instructed will be followed by the machine and robot. Enables to identify the shortest path.

Example: Robots in outer space planets identify shortest path between start and destination.

Camera and Image Processing (3)

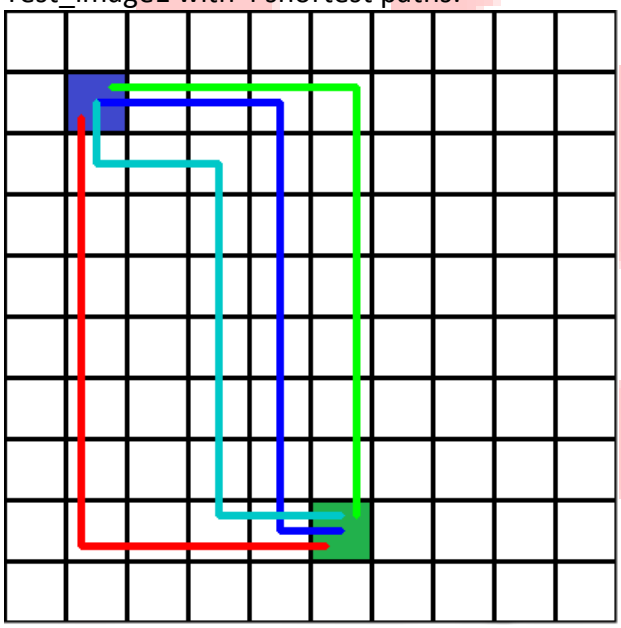
Write down the answers to the following questions. For this part use first image (*test_image1.png*) in *"Task2_Practice/test_images"* folder.

2. What is the resolution (size) of the test image?
3. 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)
4. 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:

ANSWERS:

- Resolution(size) of the test images in 400x400
- Position of Start and End Point are: In pixels: start = (60,60), end = (220,340)
  In single digit: start = (2,2), end = (6,9)

- Test_image1 with 4 shortest paths:



Software used (10)

Write down the answers to the following questions. For this part use first image in *"Task2_Practice/test_images"* folder.

5. 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:
   a. filename: color image
   b. 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.

```
def draw_grid(filename,n):
    '''
    filename-- input color image stored as file
    n-- integer from 1 to 10
    returns img-- the image with the red grid (having specified number of
    lines) drawn on it
    '''
    #add your code here
    w, h, c = img.shape
    v=h/(n+1)
    for z in range(1,n+1):
        cv2.line(img,(z*v,0),(z*v,w),(0,0,255),2)
    v=w/(n+1)
    for z in range(1,n+1):
        cv2.line(img,(0,z*v),(h,z*v),(0,0,255),2)
    return(img)
```

6. 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.

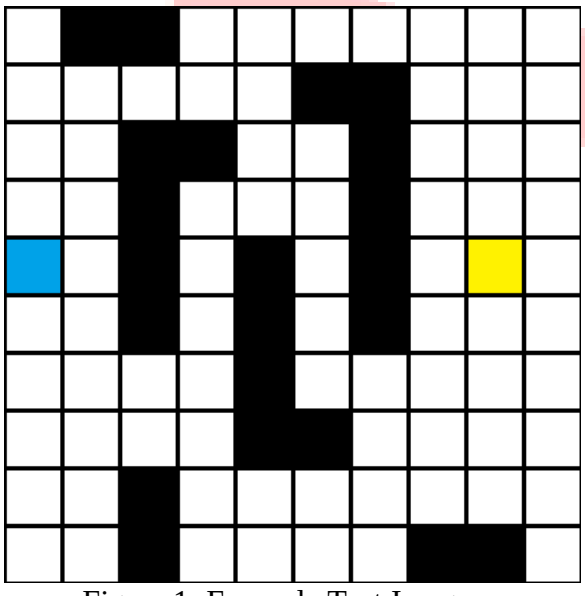


Figure 1: Example Test Image

```
>>>
grid_map =
[[0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 0, 0, 0],
[0, 0, 1, 1, 0, 0, 1, 0, 0, 0],
[0, 0, 1, 0, 0, 0, 1, 0, 0, 0],
[1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
[0, 0, 1, 0, 1, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 0, 0, 0, 0, 1, 1, 0]]
>>>
```

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
    grid_map =[]
    for i in range (20,400,40):
        row = []
        for j in range (20,400,40):
            b,g,r = img[i,j]
            if b == 255:
                row.append(0)
            elif b != 255:
                row.append(1)
        grid_map.append(row)
    print grid_map
    return grid_map
```