

**e-Yantra Robotics Competition Plus**

**(eYRC+ Pilot)**

**Team ID: eYRC+#2447**

|  |  |
| --- | --- |
| **Team leader name** | **Jayant Solanki** |
| **College** | **J.K. Institute of Applied Physics & Technology** |
| **e-mail** | **jayantjnp@gmail.com** |
| **Date** | **December 27, 2014** |

**Scope of the Task** (7)

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

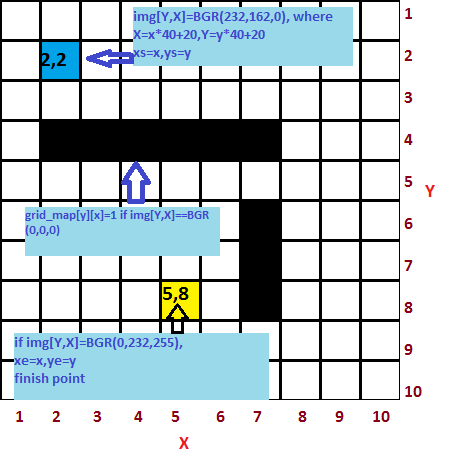
<Teams should write in their own words a description of algorithms used in this task.

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

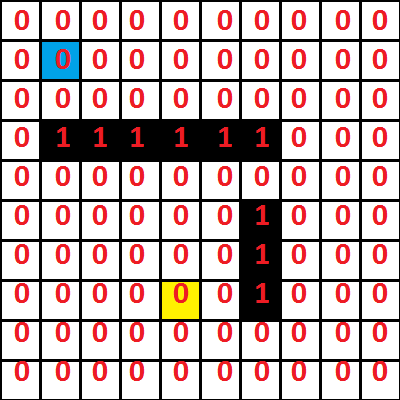
Answer format: Text

Word-limit: 100 words>

* Generate the grid map of the current image
* Initialize zero filled 10x10 grid\_map
* process image to get the pixel coordinates (X,Y) of the Centre of cell nodes
* grid\_map[y][x]=1 if img[Y][X]==black color,
* start=(xs,ys), finish=(xe,ye) for img[Y][X] == blue and yellow color
* return grid\_map

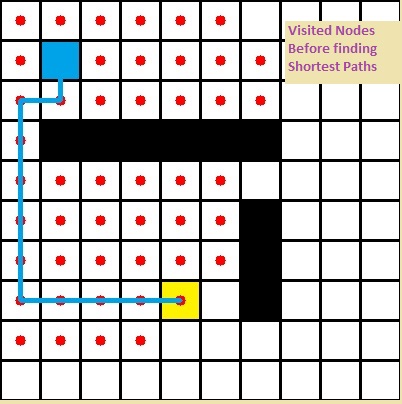
****

**Figure1.1 showing coordinates description and pixel color detection**

****

**Figure1.2 showing grid\_map representation**

* Compute shortest path
* Create priority queue which stores each visited cell's minimum cost and its coordinates.
* Repeat rest steps until current node==finish node
* visit each neighbor of current cell except obstacles
* if neighbor’s visiting cost is less or not visited previously then push it into heapq along with its new minimum cost
* Link parent node to neighbor node.
* Pop first element from heapq
* if element equals finish node, return route cost and path



**Figure 2.1 : visited nodes highlighted by red dots and shortest path shown as blue zig-zag line**

**Flow chart of above stated algorithm**

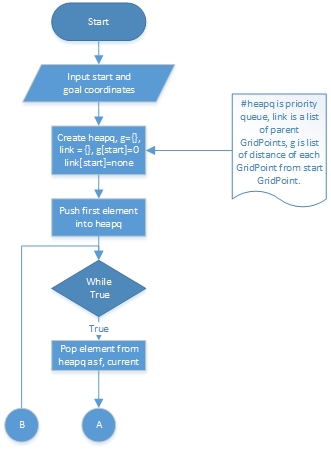
1. **Algorithm overview**

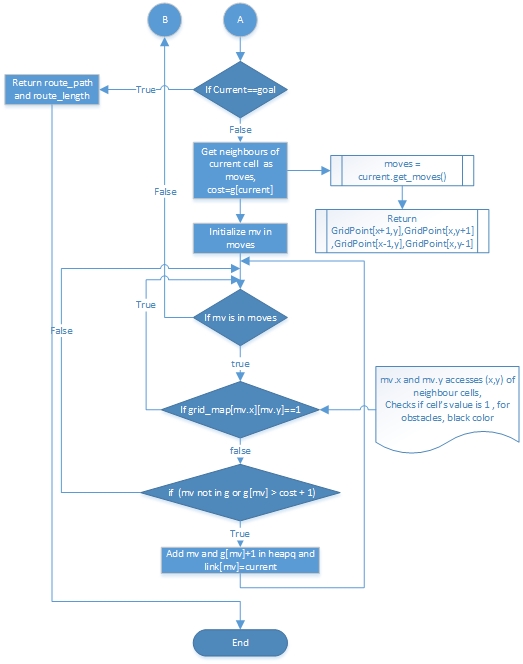
****

1. **grid\_map generator overview**

****

1. **shortest-path generator overview**

****

****

**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?

Answer to question 2

* 400\*400 pixels

1. What is the position of the Start point and the End point in the grid in the test image?

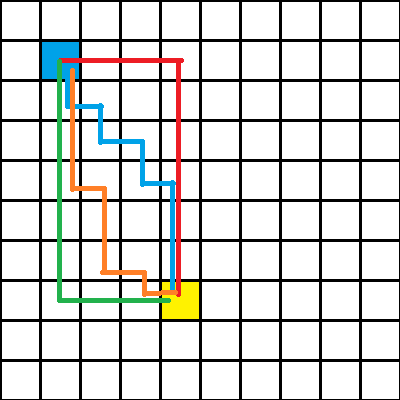
(Please refer to the *Task2\_Description.pdf* for the definitions of Start point and End point and answer in (x,y) form, where the x-axis is oriented from left to right and the y-axis is oriented from top to bottom)

Answer to question 3

* Position of start point: (2,2)
* Position of end point: (5,8)

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:

Answer to question 4



**Figure: test\_image1.png 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.

<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** 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

'''

img=cv2.imread(filename) ##getting input image

line\_width=400/(n-1)##width between 2 consecutive parallel lines

for x in range(0, n): ##drawing lines

X=x\*line\_width

for y in range(0,n):

Y=y\*line\_width

##vertical lines

cv2.line(img,(Y,X),(Y,400),(0,0,255), 2)#lines in red color ##horizontal lines

cv2.line(img,(X,Y),(400,Y),(0,0,255), 2)#lines in red color

**return(**img**)**

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.

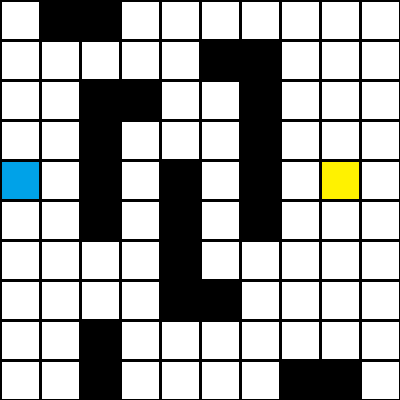
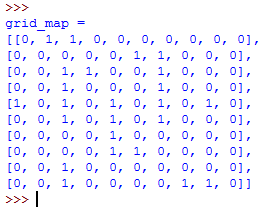
 

Figure 1: Example Test Image Figure 2: Example output

<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** space\_map**(**img**):**

'''

img-- input color image stored as file

result— output grid\_map

'''

grid\_map= [ [ 0 for i in range(10) ] for j in range(10) ]# initializing #zero filled 10x10 matrix

for x in range(0, 10):

X=x\*40+20

for y in range(0,10):

Y=y\*40+20

#img[Y,X] is pixel at the center of each cell

if img[Y,X,0]!=255 or img[Y,X,1]!=255 or img[Y,X,2]!=255: #detecting obstacle, if pixel color is not white then mark it as obstacle

grid\_map[y][x]=1 #marking obstacles with value 1

continue

**return** grid\_map