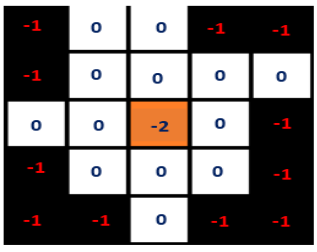
e-Yantra Robotics Competition Plus
(eYRC+ Pilot)

Team leader name	**Dhiraj Gehlot**
College	**K.J Somaiya College of Engineering**
e-mail	**dhiraj.gehlot94@gmail.com**
Date	**29-12-2014**

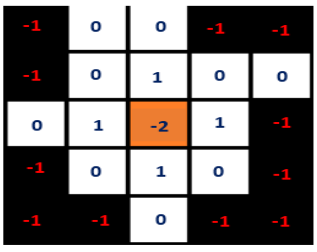
Scope of the Task (7)

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

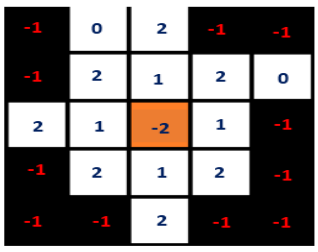
- In given task we have to figure out shortest path between start and end point.
- Take input image , find out start and end point using color mask and make a grid map of image using gridmap() which is 2d array with obstacles = -1, Start =-2 and navigable path=0.
- Add '1' to neighboring block of start point in grid map, as neighboring block are at distance '1' as shown in figure below. This is done by using startone() function
- Add '2' to neighboring of block assigned as '1', these block will be at distance of '2' from start point, keep doing this by incrementing 'count' ,and assign it to block until you get end point.
- Number should be assigned only to navigable path which are represented by '0' in grid map and not to obstacles and block which has number assigned previously.
- For routing, start from end point and store all points tracing from end point back to start point, for tracing start with neighboring point of end point and search for 'shortest length-1' point in neighboring of end point, store that point in array.
- Keep doing this trace until you get start point, co- ordinates in array gives you route in reverse order, make reverse of array and return that array as shortest route.



Calling gridmap function at start



Calling startone function



Calling scan and send_co_ordinates function

Test_image5 as example

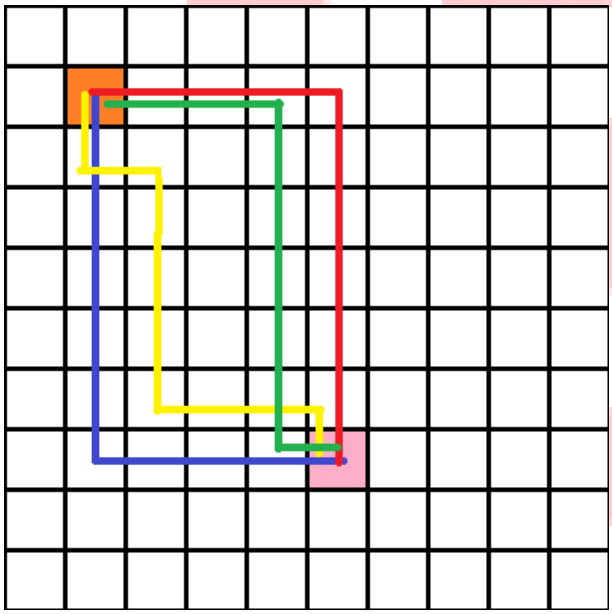
Camera and Image Processing (3)

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 of test_image1 is 400*400
- Start = (2,2) , End= (6,8)



Solution 4.

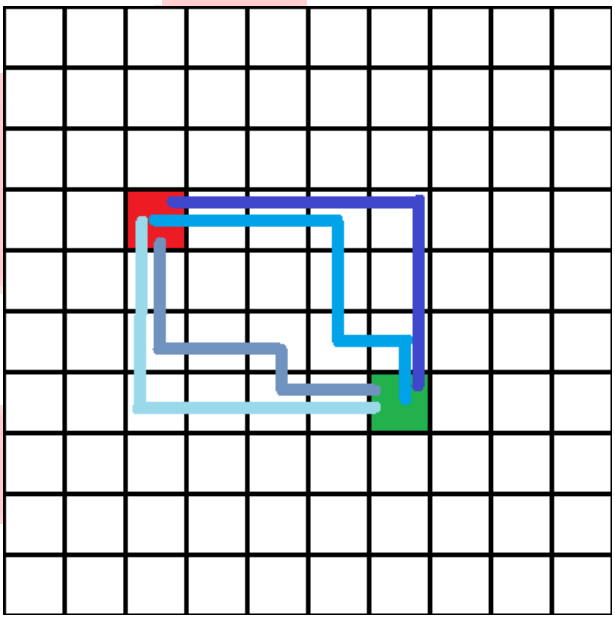


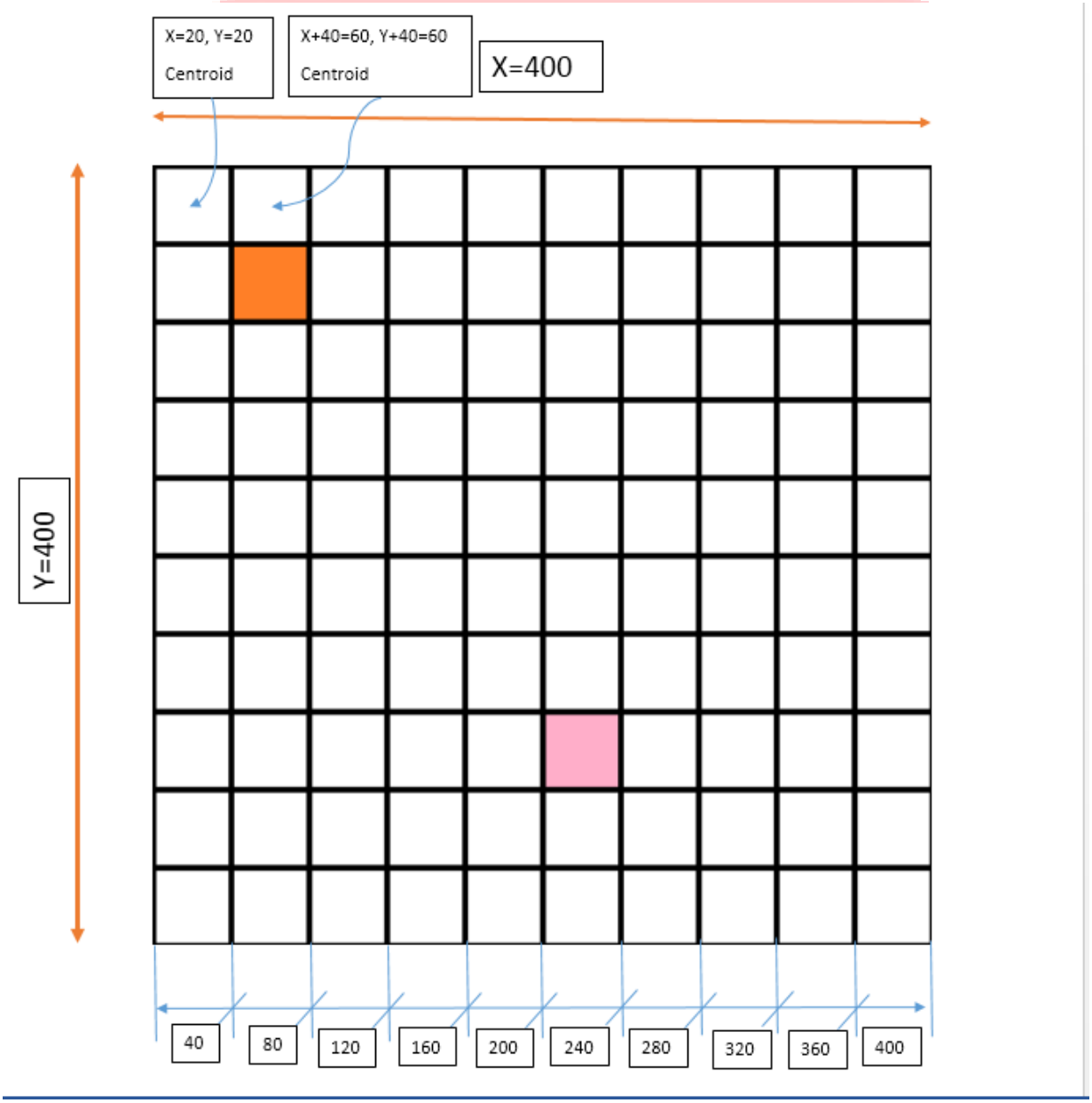
Figure: Example solution with four shortest paths drawn

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.



```
''' 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)                        # return grid image


img = cv2.imread('test_image1.png') # image input
gridimage= draw_grid(img,10);      # call grid function
cv2.imshow('grid',gridimage);      # display grid image

cv2.waitKey(0)
cv2.destroyAllWindows()
```

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.

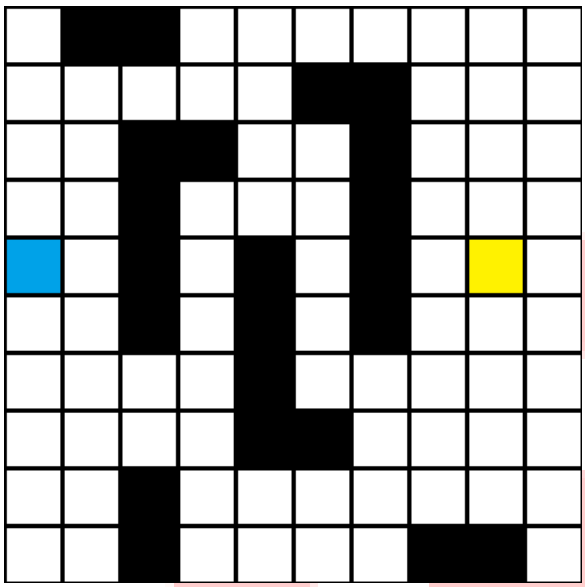


Figure 1: Example Test Image

```
>>>
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

```
''' This code take image from user and return grid map of image
Syntax : path_detect(img);
where img : original image on which we want grid map


Alogrithm:
1. Take image input
2. Call draw_grid function with "image" as parameter
3. get RGB parameters of image
4. get centroid of each image for given example 400*400 image we divide by 10 to get
40*40
5. Create array named g for storing grid map
6. Check for centroid pixel RGB value of each grid
7. If Pixel RGB value matches with white color RGB i.e 255 then it is white color
8. Assign '0' to white color pixels in grid map array else assign '1'
9 Move to next pixel by incrementing x value by 40 same is done for y value.
10. Print array returned by function using printList( ) function
'''
```

```
import numpy
import cv2

rows = 10 # rows for grid map array
cols = 10 # column for grid map array

g=[]      # array for grid map

for row in xrange(rows): g += [[0]*cols] # define array g with all element zero

'''          maxItemLength of item of an array          '''
def maxItemLength(a):
        maxLen = 0
        rows = len(a)
        cols = len(a[0])
        for row in xrange(rows):
            for col in xrange(cols):
                maxLen = max(maxLen, len(str(a[row][col])))
        return maxLen


''' function to print array sytematically as specified in theor question '''

def printList(a):
        if (a == []):
            # So we don't crash accessing a[0]
            print []
            return
        rows = len(a)
        cols = len(a[0])
        fieldWidth = maxItemLength(a)

        print "[ ",
        for row in xrange(rows):
            if (row > 0): print "\n  ",
            print "[ ",
            for col in xrange(cols):
                if (col > 0): print ",",
                # The next 2 lines print a[row][col] with the given fieldWidth
                format = "%" + str(fieldWidth) + "s"
                print format % str(a[row][col]),
            print "]",
        print "]"
```

```
def path_detect(img):
    x,y,z= img.shape; # get size of image
    n=10
    m=x/n;          # divide pixel value by 10 to scale x corodinate and store value of
pixel in array 'g'
    n=y/n;          # divide pixel value by 10 to scale y corodinate and store value of pixel
in array 'g'

    p = 20;         # start from centroid of first grid i.e (20,20) therefore initialize p=20
and q=20
    while p<x:
        q = 20;
        while q<y:
            if p<400 and q<400:
                b,c,d=img[p][q]; # get RGB value of image

                if b>240 and c>240 and d>240: # check RGB value ,if white then assign '0'
else '1'
                  g[p/40][q/40]=0;
                else:
                  g[p/40][q/40]=1;
            q=q+40;

        p=p+40;

    return g;       # return grid map array


img = cv2.imread('test_image1.png') # take input image
gridmap= path_detect(img);          # call function to get gridmap

printList(g);                       # print array 'g'

    cv2.waitKey(0)
cv2.destroyAllWindows()
```