Vision Guided Shortest Path Estimation Using Floodfill Algorithm for Mobile Robot Applications

Atul Patel, Anupam Dubey, Ajitesh Pandey, Siddharth Dutt Choubey

Abstract--**In a number of applications, the problem of determining the optimum path occurs, to be specific we shall look at the case of finding the optimum path for mobile robots. This paper presents a shortest path estimation scheme utilizing the overhead video capturing device, for mobile robot path planning. The presented scheme applies the floodfill technique to evaluate shortest path on the maze of dimension (7x7 junction wise). The overhead camera takes continuous snapshot of the maze and accordingly the path planning and computations are done. All the complex calculations and shortest path evaluations are accomplished using MATLAB®. The simulation results that have been taken shows, the mobile robot travels successfully from starting point and reaching its goal point. All obstacles that are located in its way have been avoided and shortest path estimation is done successfully.**

Index Terms-- **Adaptive algorithm, Computer simulation, Image processing, MATLAB, Object detection, Robot control, Shortest path problem, Software algorithms**

I. Introduction

Mobile robot applications often require recursive traversal in a continuously changing environment conditions between predefined starting and destination points. Real environments where these kind of mobile robots [1] have to operate are dynamically varying by nature. From the point of view of mobile robot navigation [2], it means that unexpected obstacles can appear and disappear. The nature and density of the forthcoming obstacles is usually unknown or too difficult to model. Thus in our work, an overhead camera is used to detect obstacles density and their locations on maze. A mobile robot [4] [5] [6] in a surrounding environment has to fulfill its assignment as fast and optimally as possible. This means choosing paths between source and target points that are most likely unblocked and where the robot does not spend too much time maneuvering between obstacles is our primary concern. It is often of interest to consider the minimum length path. Alternative formulations of a minimum cost path are also sometimes important. In particular, the minimum time path needs not necessarily be the same thing as the minimum length path. A common case in which these two can differ occurs when the maximum robot speed is a function of the curvature of the path, as is the case with many synchronous-drive vehicles. As a consequence a minimum-length path may involve lower robot speed than a longer but lower curvature (and hence faster) alternative path.

In this work a software simulation model has been proposed for two driven wheels mobile robot path planning that can navigate in dynamic environment with static distributed obstacles. The simulation model is constitute using MATLAB® and the results are also examined on the same.

In Section II, basic structure of the work is justified and its intellect in path planning field. Proposed algorithm, Simulation results and related important experimental aspects are described in Section III. In Section IV, conclusions and future works are discussed.

II. Basic Structure

Software simulation model for vision based shortest path estimation is carried out in MATLAB [3], where all the complex computations are done. Simulation starts with the snap of the maze, to monitor number of obstacles (red block of dimension 10x10x10 cm^3) and their relative position on the arena. Detection of obstacle in the mazes' vicinity is done using the ALGORITHM [A] depicted in section III. Images snap of the maze looks like one shown in Fig. 1, this snap is continuously captured using an overhead video capturing and accordingly processed in MATLAB. Starting position is shown with green marking and destination point is described by blue marking on the maze. Obstacles shown are randomly distributed in the maze, keeping in mind that at least one path is available between starting and destination point. The presented scheme is an orientation-based approach, i.e. robot would likely take the path that it is facing, so as to avoid time taken to make a left or a right turn.

Atul Patel is currently pursuing Bachelors degree in Electronics and Communication Engineering in Gyan Ganga Institute of Technology and Sciences, Jabalpur, India. E-mail: atul8255@gmail.com

Anupam Dubey is currently pursuing master's degree program in Information Technology in IPM, Kanpur, India. E-mail: Dubey.anupam@gmail.com

Ajitesh Pandey is currently pursuing Bachelors degree in Electronics and Communication Engineering in Oriental Engineering College, Jabalpur, India. E-mail: ajitesh.9.pandey@gmail.com

Siddharth Dutt Choubey is currently pursuing ME in System Software in SRIT Jabalpur, India. E-mail: Siddharth.choubey@gmail.com

978-1-4673-1049-9/12/$31.00 ©2012 IEEE

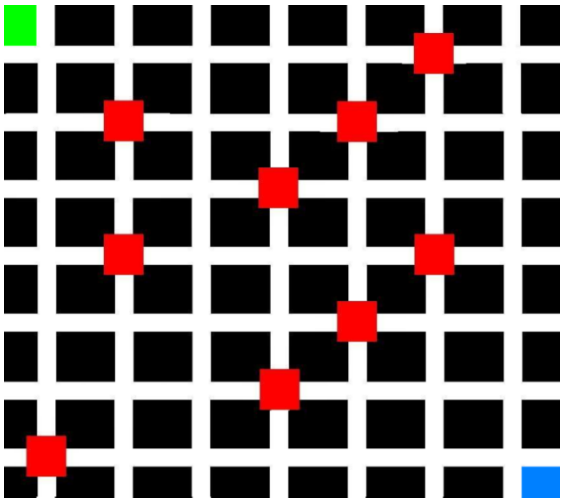

Fig. 1. Diagrammatic Image of the Arena with Obstacles.

III. Algorithms and Results

A. Obstacle detection using image snap

This part of the algorithm simply takes the snap of maze from overhead video capturing device and monitors the vicinity for the presence of obstacles. Monitoring is done by pixel-to-pixel scanning the snapped image with continuously judging the junctions in the maze for obstacles' locality. A pre-defined matrix [j of dimension 9*9] is taken with outer elements as 99 and remaining elements having a dummy value 64. Upon detection of an obstacle in the maze, corresponding element of the matrix [j] is assigned a highest value of 99, as soon in Fig.2 below.

```
[a b c]=size(rgb);
rgb1=rgb(:,:,1); rgb2=rgb(:,:,2); rgb3=rgb(:,:,3);
%extracting the RGB components in seperate Matrix
y=a;
x=b;
i=0;
o=0;
k=0;
j=[99,99,99,99,99,99,99,99,99 ;
   99,64,64,64,64,64,64,64,99 ;
   99,64,64,64,64,64,64,64,99 ;
   99,64,64,64,64,64,64,64,99 ;
   99,64,64,64,64,64,64,64,99 ;
   99,64,64,64,64,64,64,64,99 ;
   99,64,64,64,64,64,64,64,99 ;
   99,64,64,64,64,64,64,64,99 ;
   99,99,99,99,99,99,99,99,99];
for m=75:100:a
for n=75:115:b
if (rgb1(m,n)>200 && rgb2(m,n)>200 && rgb3(m,n)>200)
%Reading Junction points
i=i+1;
else % If Obstacle is present
o=o+1;
p=(n-75)/115;
p=p+2;
q=(m-75)/100;
q=q+2;
j(q,p)=99
end
end
end
```

```
j =

    99    99    99    99    99    99    99    99    99
    99    64    64    64    64    64    99    64    99
    99    64    99    64    64    99    64    64    99
    99    64    64    64    99    64    64    64    99
    99    64    99    64    64    64    99    64    99
    99    64    64    64    64    99    64    64    99
    99    64    64    64    99    64    64    64    99
    99    99    64    64    64    64    64    64    99
    99    99    99    99    99    99    99    99    99

>>
```

Fig. 2. Matrix [j] after applying 1st Algorithm.

B. Floodfill Algorithm

The Algorithm described below is the shortest path determining part, which takes matrix [j] and scans it element by element, assigning each element with new set of values (k), based on the presence or absence of obstacles. This algorithm firstly, assigns destination point with least value i.e. zero and then numbers the elements corresponding to shortest path, during this process it also checks the neighboring elements. At the end of the algorithm, we get a new matrix [j], with flooding values between starting position and destination points.

```
j(8,8)=0 %% Destination position
k=0;
i=0;
trial=49;
while trial>0
trial=trial-1;
for m=2:8
for n=2:8
if (j(m,n) == k)
if ((m+1<9)&&j(m+1,n)~=99&&j(m,n)<j(m+1,n))
j(m+1,n)=k+1;
end
if ((m-1>=0)&&j(m-1,n)~=99&&j(m,n)<j(m-1,n))
j(m-1,n)=k+1;
end
if ((n+1<9)&&j(m,n+1)~=99&&j(m,n)<j(m,n+1))
j(m,n+1)=k+1;
end
if ((n-1>=0)&&j(m,n-1)~=99&&j(m,n)<j(m,n-1))
j(m,n-1)=k+1;
end
end
end
end
k=k+1;
end
```

```
j =

    99    99    99    99    99    99    99    99    99
    99    12    11    10    11    12    99     6    99
    99    11    99     9    10    99     6     5    99
    99    10     9     8    99     6     5     4    99
    99     9    99     7     8     7    99     3    99
    99     8     7     6     7    99     3     2    99
    99     7     6     5    99     3     2     1    99
    99    99     5     4     3     2     1     0    99
    99    99    99    99    99    99    99    99    99

>>
```

Fig. 3. Matrix [j] after applying 2nd Algorithm.

```
pathx =

     2     2     2     2     2     2     3     3     4     5     6     7     8

pathy =

     2     3     4     5     6     7     7     8     8     8     8     8     8

>>
```

Fig. 4. X and Y coordinates (Elements of the Matrix[j]) of the shortest path for arena depicted in Fig. 1.

Fig. 4 shows the image of the two position vectors, showing the coordinates of the shortest path in terms of elements of the matrix [j]. The same path vectors are then plotted on the graph to visualize the generated shortest path as shown in Fig. 5.

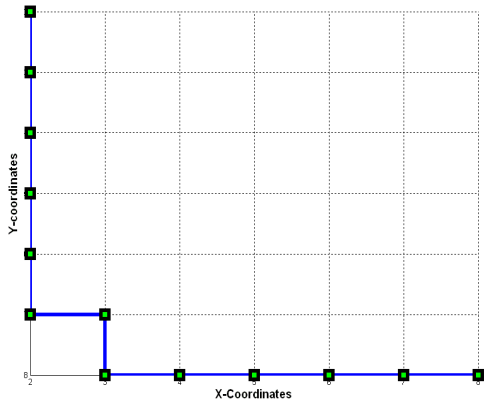

Fig. 5. Plot of Pathx and Pathy for Arena described in Fig. 1.

IV. Conclusion

Although the floodfill algorithm is specific to artificial intelligence, it finds applicability in any application requiring graph search, in our case it is employed to mobile robotics. This is due to the fact that it is a generic algorithm, applicable to any optimum path problems. Through this research, our proposed scheme seems like a new shortest path planning model which exhibits very attractive efficiency while eliminating the fault paths (path containing obstacles), which increases the functionality efficiency in paths feasibility examination.

V. Acknowledgment

The special thank goes to our helpful supervisors, Mr. Anupam Dubey. The supervision and support that he gave truly help the progression and smoothness of the proposed work. The work accomplished during the tenure would be nothing without the enthusiasm and imagination from him. The co-operation is much indeed appreciated. Proposed work could never have been attempted without our reference to and inducements from the works of others, whose commendable works are mentioned in references section.

VI. References

[1] Brooks, R.A., "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, vol. 2, no. 1, (1986): 14-23.

[2] Dudek, G., Milios, E., Jenkin, M. & Wilkes, D., "Map validation and self-location for a robot with a graph-like map", Robotics and Autonomous Systems, vol. 26, (1997): 159- 187.

[3] Gonzalez R.C, Woods R.E, Digital Image Processing, 2nd edition, Prince Hall India.

[4] Hongshe Dang; Jinguo Song; Qin Guo; "An Efficient Algorithm for Robot Maze-Solving", appeared in Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2010 2nd International Conference ,(26-28 Aug. 2010), 79 – 82.

[5] Mishra, S. Bande, P. "Maze Solving Algorithms for Micro Mouse", appeared in Signal Image Technology and Internet Based Systems, 2008. SITIS '08. IEEE International Conference,(Nov. 30 2008-Dec. 3 2008), 86 – 93.

[6] Nanjing, Jiangsu" An Efficient Algorithm for Robot Maze-Solving", appeared in 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetic(August 26-August 28).