LOW COMPLEXICITY GRAPH BASED NAVIGATION AND PATH FINDING OF MOBILE ROBOT USING BFS

Prabin Kumar Panigrahi
School of Computer Engineering
KIIT University, Patia
Bhubaneswar-751024, India
prabinprakash1@gmail.com

Hrudaya Kumar Tripathy
School of Computer Engineering
KIIT University, Patia
Bhubaneswar-751024, India
hktripathyfcs@kiit.ac.in

ABSTRACT

Path-finding is a fundamental problem, in mobile robotics which involves finding an optimal collision-free path from the source node to the destination node. Before the robot traces out the required path, the first step to be carried out is, exploring the environment. Localization plays a major role in this case. This paper adopts an RFID (Radio Frequency Identification) based localization technique. A set of RFID(IC) tags arranged in a grid structure in an equidistant manner are used for the purpose of tracing the current co-ordinate/location of the robot. After exploring the environment one virtual map is generated which contains the location of source, destination, obstacles and landmarks. From the map one graph is generated which is composed of a set of vertices that indicates the cells of the grid and a set of edges which indicates the free path in the environment reachable from the source. After analyzing different searching algorithms we found Breadth First Search (BFS) algorithm to be more effective, because it finds the shortest path from the source node to each node in the graph. We improved the BFS algorithm so that the optimal collision free path from source to destination node is generated. We have implemented our techniques in a simulated environment. We have used MATLAB and JAVA for the simulation purpose. Finally we have demonstrated the result of the implementation through examples.

Categories and Subject Descriptors

G.4 [**MATLAB**]: Language Constructs and Features – *graphical user interface, animation.*

D.3.2 [**Java**]: Language Constructs and Features – *object oriented programming, control structures, polymorphism.*

General Terms

Algorithms, Experimentation.

PerMIn '15, February 26 - 27 2015, Kolkata, West Bengal, India
Copyright 2015 ACM 978-1-4503-2002-3/15/02…$15.00
http://dx.doi.org/10.1145/2708463.2709068

Keywords

Path finding, Navigation, Localization, Breadth First Search, RFID.

1. INTRODUCTION

The word Robot is coined from the Czech word robota, which means "forced labor". A robot is an agent which forces the environment. A robot can be defined as an agent that carries out its tasks automatically or with a minimum of external impulse rather than are creation of life itself. It is a smart machine.

Basically there are two types of robots. First one is the fixed-based robotic manipulators. These types of robots are fixed at a specific position. However these types of robots like robotic arm can move with great speed and accuracy to perform repetitive tasks like spot welding and painting. However these types of robot have one major disadvantage, i.e. "lack of mobility". There can the birth of another category of robot called "mobile robot". The fixed-based manipulator have the limited capability of motion, however the mobile robot can travel anywhere in the environment. A mobile robot is an automatic machine which has the capability of movement in any given environment. Mobile robots have been widely applied in the fields of path patrolling, environmental searching, objects tracking, tour guiding or even house cleaning, etc. These type of robots are intelligent. They perform their task without any intervention of human user. Normally the fixed - base robotic manipulators are programmed to perform repetitive tasks with limited use of sensors, however mobile robots are less structured in their operation and use more sensors. In case of mobile robot the supervisor maintains the current location of the robot. Hence localization plays an important role in mobile robots.

There are various technologies available for the localization of mobile robot. Many authors have used Radio Frequency Identification (RFID) technology for mobile robot localization. RFID uses a set of IC tags for getting the current co-ordinate of the robot through the help of antenna. RFID technology found to be more effective and more promising as it produces the accurate location of the robot. Hence we have used the RFID scheme for the localization of the robot.

Navigation is very much essential in mobile robotics. Through navigation, the robot explorers the environment. However before navigation another thing should be kept in mind is path-finding. To reach the goal, the robot has to find the shortest path from source to destination. The shortest path problem can be categorized in various ways like: single-source shortest-paths problem(find a shortest path from source node to each reachable nodes in the environment), Single-destination shortest-paths problem(find a shortest path to a destination node from every node in the environment),Single-pair shortest-path problem(find a shortest path from a node u to a node v in the given environment), All-pairs shortest-paths problem(trace out a shortest path for each pair of nodes in the environment) [9]. There are various shortest path algorithms available, like BFS, A*, Dijkstra's algorithm, Bellman-Ford, Floyd-Warshall algorithm etc. There are different ways to generate the shortest path. Lucia Vacariu et al.[7] suggested a hardware and software implementation of mobile robot path planning using the BFS algorithm.

We have used a uniformed search algorithm called Breadth First Search (BFS) algorithm. Basically BFS algorithm finds a shortest path from the source node to each nodes in the graph. Our aim is to find the shortest path from the source node to the destination (goal) node. After generating the virtual world the robot explores the environment and finds all possible paths from source to goal along with its path cost. This is demonstrated through example. This path information is stored in the memory of the robot. The robot uses FIND-SHORTEST-PATH algorithm which in turn uses the BFS algorithm for the generation of an optimal shortest path from source to destination. Finally the devised shortest path is stored in the memory of the robot so that it can navigate to the goal in that path.

2. LOCALIZATION OF MAP

Localization is the process of tracing out the current position or co-ordinate of the robot i.e. the (x, y) co-ordinate value through the virtual world (considered as map). This is very much important for the robot while exploring the environment designed in a grid structure. We focus in the grid based environment where we fix the grid size to be 5 x 5. Each grid is a square one. In this paper placement of source, destination, and obstacles is dynamic. Here each time the robot is reinforced to learn the environment, i.e. each time it explores the environment.

2.1 Technology used for localization of mobile robot in grid

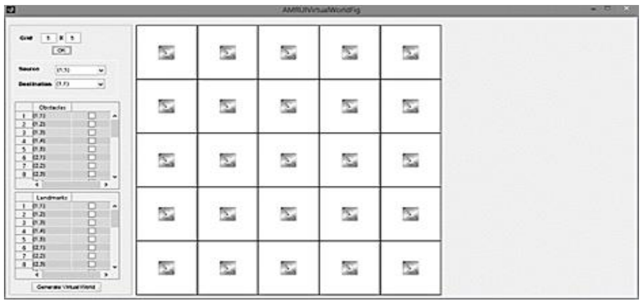
Different technologies are available for the localization of the mobile robot on grid structure designed for virtual world/environment. However we found the **Radio Frequency IDentification (RFID)** technology to be more effective in comparison to others. Yung-Fu Hsu *et al*.[1] discussed an RFID based localization model using the passive RFID tags. Sunhong Park *et al*.[2] described the robot system named UBIRO (UBIquitous RObot) which was developed based onan electric wheelchair for the elderly and disabled persons as a mobility device which has RFID as its part for orientation. Byoung-Suk Choi *et al*.[3] suggested the use of RFID sensor fusion technique for mobile robot localization. Ali Asghar*et et al*.[4] proposed a novel system which consists of RFID carpet and several peripherals for interpreting the sensor data. Dirk Hähnel *et al*.[5] discussed on the usage of RFID technology in improving the localization of the mobile robot. Wail Gueaieb *et al*.[6] adopts the RFID technology and utilized it in the 3D space.

Generally RFID tags are referred to as landmarks. It captures the signals by the help of RFID reader for determining the position of the robot and its direction. The position information (i.e. current co-ordinate) is sent to the robot for further processing.

2.2 Radio Frequency Identification (RFID) System

Generally a RFID system consists of a set of RFID (IC) tags, a RFID reader, an antenna(s) and software like middleware. Basically there are three types of RFID tags available: (1) active (2) passive (3) semi-passive tags. Concerning the cost parameter passive RFID tags are the cheapest one.

We proposed a fixed based 5 x 5 grid structure having each cell placed with a passive RFID tag. The RFID tags are equidistance to each other. Each tag is fixed in the center of the corresponding cell and contains the co-ordinate value to provide the position data to the mobile robot. The above figure (Figure 1.) demonstrates the placement of RFID tags.



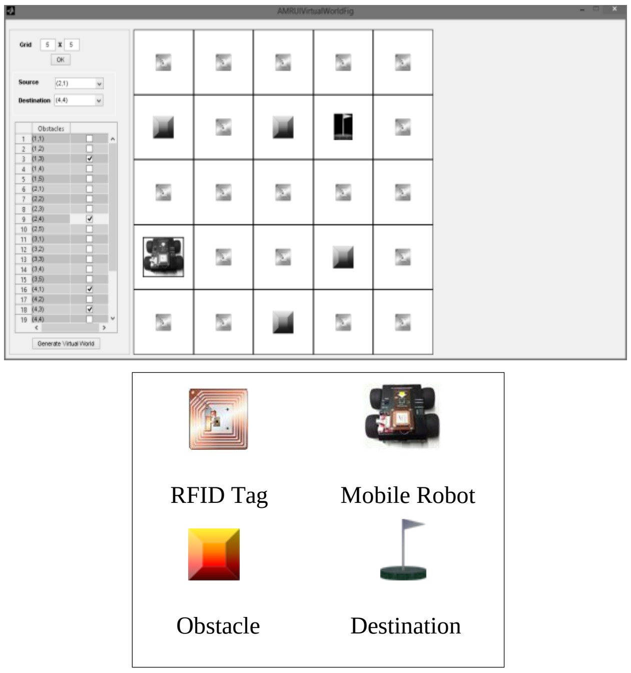
Grid 5 x 5

Source (1,1)

Destination (1,1)

Figure 1. Demonstration of RFID tag placement

For reading the data from the RFID(IC) tag a RFID reader is installed below the mobile robot. For localizing the robot, the RFID reader reads the coordinate value of the RFID tags, when the robot navigates to a position over a tag. Now the robot's absolute position is estimated without considering the obstacles. The following figure (Figure 2.) shows the placement of the robot, obstacles and identification of source and goal.



Grid 5 x 5

Source (2,1)

Destination (4,4)

	Obstacles	
1	(1,1)	☐
2	(1,2)	☐
3	(1,3)	✓
4	(1,4)	☐
5	(1,5)	☐
6	(2,1)	☐
7	(2,2)	☐
8	(2,3)	☐
9	(2,4)	✓
10	(2,5)	☐
11	(3,1)	☐
12	(3,2)	☐
13	(3,3)	☐
14	(3,4)	☐
15	(3,5)	☐
16	(4,1)	✓
17	(4,2)	☐
18	(4,3)	✓
19	(4,4)	☐

Figure 2. Demonstration of virtual world

The above figure is the result of the simulation. However the original positioning is shown in the following figure (Figure 3.).

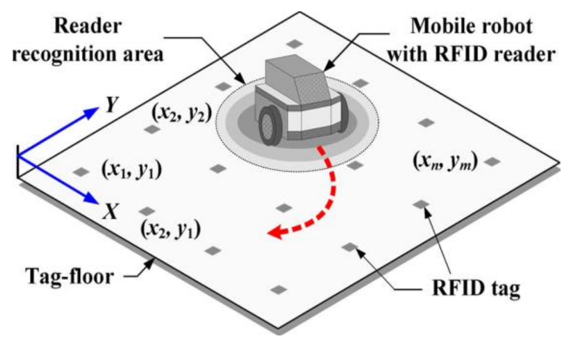


Figure 3. Distribution of RFID tags in floor for mobile robot localization based on the RFID system [3].

2.2.1 *Calculation for position estimation*

For localizing the mobile robot, the major part is position estimation i.e. identification of the current co-ordinate of the robot. Many authors have used different techniques for identifying the current location of the robot. Sunhong Park *et al.*[2] suggested a polar co-ordinate based localization model. Polar coordinates (r,θ) are defined in terms of Cartesian coordinates (x,y) by,

$$x = r\cos\theta \tag{1}$$

$$y = r\sin\theta \tag{2}$$

$$r = \sqrt{(x^2 + y^2)} \tag{3}$$

$$\theta = \tan^{-1}(y/x) \tag{4}$$

Where, r: radial distance from origin, θ: counter clockwise angle from the positive X-axis. When the robot detects a new IC tag Q(x_2, y_2), as shown in Figure 4., the current location of the robot P($x_{current}$, $y_{current}$) can be calculated as follows,

$$x_{current} = r\cos\theta' + x_2 \tag{5}$$

$$y_{current} = r\sin\theta' + y_2 \tag{6}$$

Where θ' denotes an incident angle against the newly detected IC tag.

Byoung-Suk Choi*et et al.* [3] adopts a novel approach for estimating the state of the mobile robot at a time k. This state can be calculated as,

$$\hat{P}(k) = [\hat{x}(k)\ \hat{y}(k)]^T = [x'(k) + a(k)\ y'(k) + b(k)]^T \tag{7}$$

In case of real robot suppose the antenna in the robot detects multiple tags as Tag1 (x_1, y_1), Tag2 (x_2, y_2) …Tag1 (x_n, y_n).The position of the robot can be estimated by using the following equations

$$x_{estimate} = (x_1+x_2+\ldots+x_n)/n, \tag{8}$$

$$y_{estimate} = (y_1+y_2+\ldots+y_n)/n, \tag{9}$$

Where n is the detected number by antenna.

In our system the RFID tags are placed in an equidistance manner i.e. 116.6 pixels to each other. The robot can move only in 4 directions i.e. left, right, up, and down. Whenever the robot stays above the RFID tag, the tag sends the co-ordinate value by which the robot obtains the current location. In our case the robot can move from centre of a cell to the centre of another reachable cell i.e. it can travels 116.6 pixel in each of the four directions.

3. EXPLORATION OF PATH

Before finding the shortest path the robot should explore the virtual world. This means with the given source and goal co-ordinate the robot should be able to reach the goal in various possible ways. In our experiment we place the robot in the source

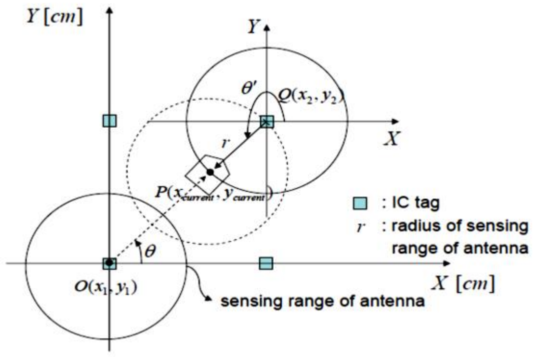
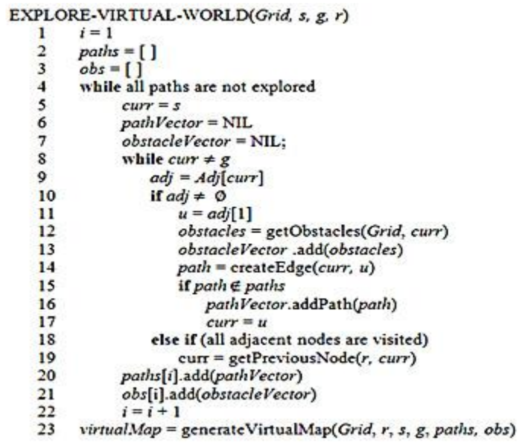


Figure 4. Polar coordinate at P [2]

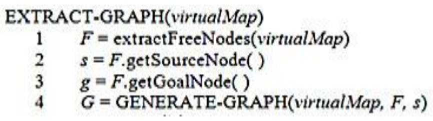
co-ordinate. The robot searches for all possible paths from the source to the goal.

While searching for the path it searches for obstacle by throwing light sensor to adjacent (left, right, up and down) cells. It can move only to the obstacle free adjacent cell. When it moves to a cell it stores the edge and its cost (1 here) in its memory. After reaching the goal the robot has the whole path and the corresponding total path-cost in its memory. After a path is devised the robot searches for the next possible path. This process is continued until the robot traces out all possible paths from source to goal node. This is the exploration stage where the robot has all information about the virtual world. Our algorithm for exploring the path is given below (Figure 5.(a)):



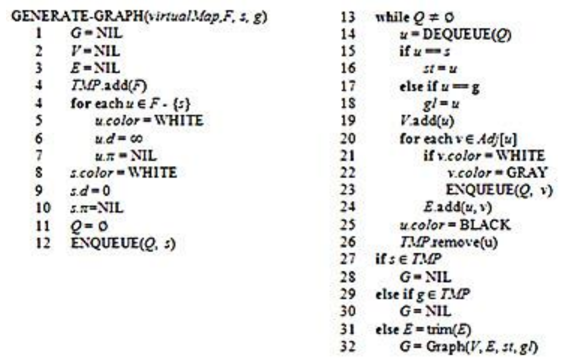
```
EXPLORE-VIRTUAL-WORLD(Grid, s, g, r)
    i = 1
    paths = [ ]
    obs = [ ]
    while all paths are not explored
        curr = s
        pathVector = NIL
        obstacleVector = NIL;
        while curr ≠ g
            adj = Adj[curr]
            if adj ≠ Ø
                u = adj[1]
                obstacles = getObstacles(Grid, curr)
                obstacleVector .add(obstacles)
                path = createEdge(curr, u)
                if path ∉ paths
                    pathVector.addPath(path)
                curr = u
            else if (all adjacent nodes are visited)
                curr = getPreviousNode(r, curr)
        paths[i].add(pathVector)
        obs[i].add(obstacleVector)
        i = i + 1
    virtualMap = generateVirtualMap(Grid, r, s, g, paths, obs)
```

Figure 5. (a) EXPLORE-VIRTUAL-WORLD algorithm.



```
EXTRACT-GRAPH(virtualMap)
    F = extractFreeNodes(virtualMap)
    s = F.getSourceNode( )
    g = F.getGoalNode( )
    G = GENERATE-GRAPH(virtualMap, F, s)
```

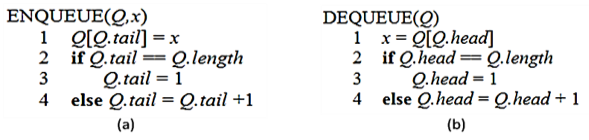
Figure 5. (b) EXTRACT-GRAPH algorithm.



```
GENERATE-GRAPH(virtualMap, F, s, g)
    G = NIL
    V = NIL
    E = NIL
    TMP.add(F)
    for each u ∈ F - {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
    s.color = WHITE
    s.d = 0
    s.π = NIL
    Q = Ø
    ENQUEUE(Q, s)
    while Q ≠ Ø
        u = DEQUEUE(Q)
        if u == s
            st = u
        else if u == g
            gl = u
        V.add(u)
        for each v ∈ Adj[u]
            if v.color = WHITE
                v.color = GRAY
                ENQUEUE(Q, v)
            E.add(u, v)
        u.color = BLACK
        TMP.remove(u)
    if s ∈ TMP
        G = NIL
    else if g ∈ TMP
        G = NIL
    else E = trim(E)
        G = Graph(V, E, st, gl)
```

Figure 5. (c) GENERATE-GRAPH algorithm.

The idea used in the EXPLORE-VIRTUAL-WORLD algorithm is that, initially the robot will look for obstacle free adjacent cells and traverse to any of these cell if and only if the cell is not visited by from the current cell. If the adjacent cell is visited than the robot will look for other free adjacent cells. If all of the adjacent cells are visited then the robot will move back to the previous node and will search for the adjacent cell. These steps are repeated until the goal node is reached. When the robot reaches the goal. The explored path is added to a array and the robot searches for the next path. After all possible paths are explored a virtual map is generated. EXTRACT-GRAPH procedure (Figure 5.(b).) find out the source and goal nodes and generates the graph using the GENERATE-GRAPH procedure (Figure 5.(c).). It uses the Queue data structure. In a queue insertion takes place in the tail of the queue and deletion takes place in the head of the queue. The procedures ENQUEUE(Q,x) (Figure 6.(a).) and DEQUEUE(Q) (Figure 6.(b).) performs the insertion and the deletion operation in the queue Q respectively.



```
ENQUEUE(Q,x)
    Q[Q.tail] = x
    if Q.tail == Q.length
        Q.tail = 1
    else Q.tail = Q.tail + 1
(a)

DEQUEUE(Q)
    x = Q[Q.head]
    if Q.head == Q.length
        Q.head = 1
    else Q.head = Q.head + 1
(b)
```

Figure 6.(a) ENQEUE algorithm (b) DEQEUE algorithm [9]

We use a 5 x 5 grid based virtual world as shown in Figure 7.

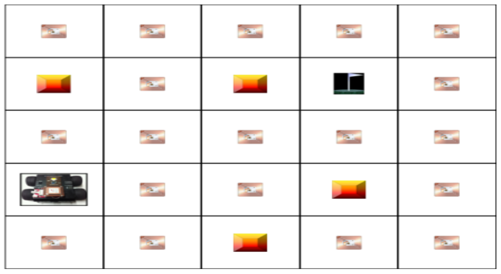
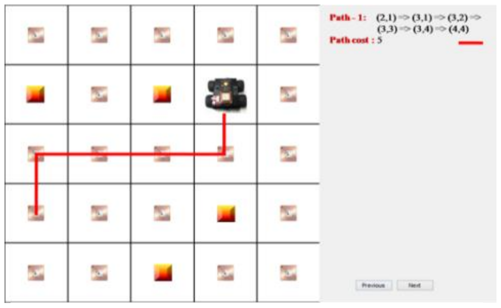


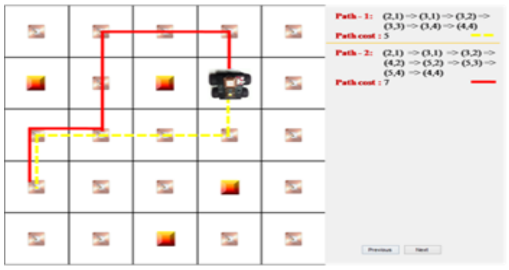
Figure 7. Virtual World

After the virtual world is generated the robot explores the paths (Figure 8., Figure 9., Figure 10. , Figure 11., and Figure 12.). There can be other paths as well. However we have taken five paths to demonstrate. The robot stops exploring paths when all possible paths are explored. After that it generates a virtual map of the virtual world. Finally the graph is generated from the virtual map. The path exploration is demonstrated in the following example.



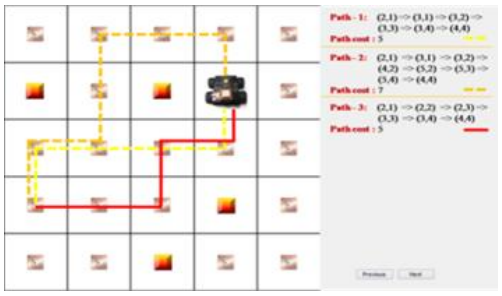
Path - 1: (2,1) => (3,1) => (3,2) => (3,3) => (3,4) => (4,4)
Path cost : 5

Figure 8. After the exploration of path-1



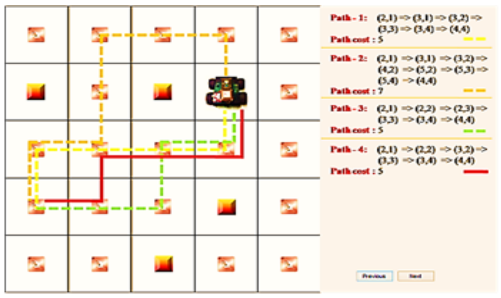
Path - 1: (2,1) => (3,1) => (3,2) => (3,3) => (3,4) => (4,4)
Path cost : 5
Path - 2: (2,1) => (3,1) => (3,2) => (4,2) => (5,2) => (5,3) => (5,4) => (4,4)
Path cost : 7

Figure 9. After the exploration of path-2



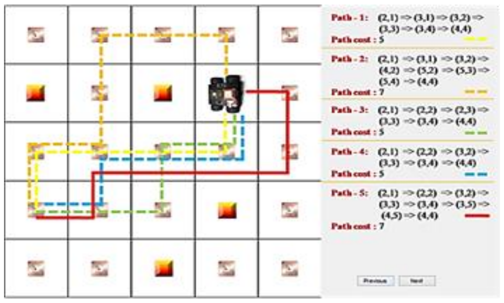
Path - 1: (2,1) => (3,1) => (3,2) => (3,3) => (3,4) => (4,4)
Path cost : 5
Path - 2: (2,1) => (3,1) => (3,2) => (4,2) => (5,2) => (5,3) => (5,4) => (4,4)
Path cost : 7
Path - 3: (2,1) => (2,2) => (2,3) => (3,3) => (3,4) => (4,4)
Path cost : 5

Figure 10. After the exploration of path-3



Path - 1: (2,1) => (3,1) => (3,2) => (3,3) => (3,4) => (4,4)
Path cost : 5
Path - 2: (2,1) => (3,1) => (3,2) => (4,2) => (5,2) => (5,3) => (5,4) => (4,4)
Path cost : 7
Path - 3: (2,1) => (2,2) => (2,3) => (3,3) => (3,4) => (4,4)
Path cost : 5
Path - 4: (2,1) => (2,2) => (3,2) => (3,3) => (3,4) => (4,4)
Path cost : 5

Figure 11. After the exploration of path-4

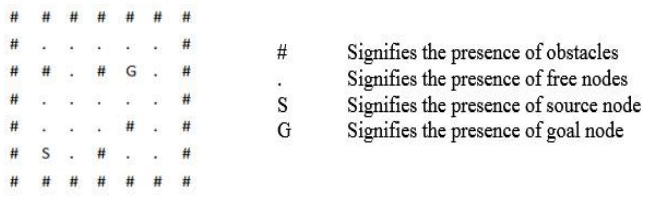


Path - 1: (2,1) => (3,1) => (3,2) => (3,3) => (3,4) => (4,4)
Path cost : 5
Path - 2: (2,1) => (3,1) => (3,2) => (4,2) => (5,2) => (5,3) => (5,4) => (4,4)
Path cost : 7
Path - 3: (2,1) => (2,2) => (2,3) => (3,3) => (3,4) => (4,4)
Path cost : 5
Path - 4: (2,1) => (2,2) => (3,2) => (3,3) => (3,4) => (4,4)
Path cost : 5
Path - 5: (2,1) => (2,2) => (3,2) => (3,3) => (3,4) => (3,5) => (4,5) => (4,4)
Path cost : 7

Figure 12. After the exploration of path-5

The virtual map is shown in the following figure (Figure 13.). The next phase is finding the shortest path from source to goal.

The following figure (Figure 14.) shows the resultant graph from the map. Here 1 is distance between any two nodes.



```
# # # # # # #
# . . . . . #
# # . # G . #
# . . . . . #
# . . . # . #
# S . # . . #
# # # # # # #
```

#	Signifies the presence of obstacles
.	Signifies the presence of free nodes
S	Signifies the presence of source node
G	Signifies the presence of goal node

Figure 13. Generated Virtual map

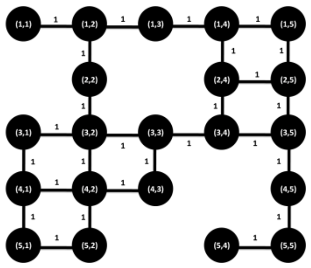


Figure 14. Resultant graph generated from virtual map.

4. GENERATION OF SHORTEST PATH

After exploring all possible paths the robot stores these paths and their costs in memory. The next step is to trace out the optimal shortest path from source to goal. This can be easily be found by choosing the path having low path cost. However this is not the optimal one. Hence we go for an optimal shortest path algorithm i.e. the Breadth-First-Search (BFS) algorithm.

Graph Theory plays a major role in mobile robot navigation. Graph is an important mathematical tool which covers a wide range of areas like operational research, chemistry, genetics, linguistics, electrical engineering, geography, sociology, architecture, computer science etc.[8]. Here we briefly explain the concept of graph and its constituents.

A graph(G) is a composition of non-empty finite set of vertices(V) and edges(E) and is defined as,

G = (V, E), where V⊆ V(G) and E ⊆ E(G).

V(G) is the set of all vertices in the graph.

E(G) is set of all edges in the graph

A graph can either be represented as adjacency matrix or adjacency list. In our experiment we have used the adjacency list representation. The following example (Figure 15. & Table 1.) demonstrates a graph and it's representation.

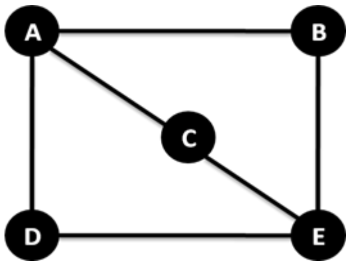


Figure 15. Graph G with 5 vertices & 6 Edges

Table 1.Adjacency list representation of Graph G

Vertex	Adjacent Vertices
A	B, C, D
B	A, E
C	A, E
D	A, E
E	B, C, D

The graph is constructed from the virtual map and is represented as adjacency list. The robot calls FIND-SHORTEST-PATH (a shortest path algorithm) algorithm which generates the shortest path in this graph. A shortest-paths problem, involves a weighted graph G = (V,E) having the weight function w : E → R which maps edges to real-valued weights. The weight w(p) of path p= <v0,v1,...,vk> is defined as the sum of the weights of its constituent edges [9]:

$$w(p) = \sum_{i=1}^{k}\left(w\left(v_{i-1,}\ v_i\right)\right) \tag{10}$$

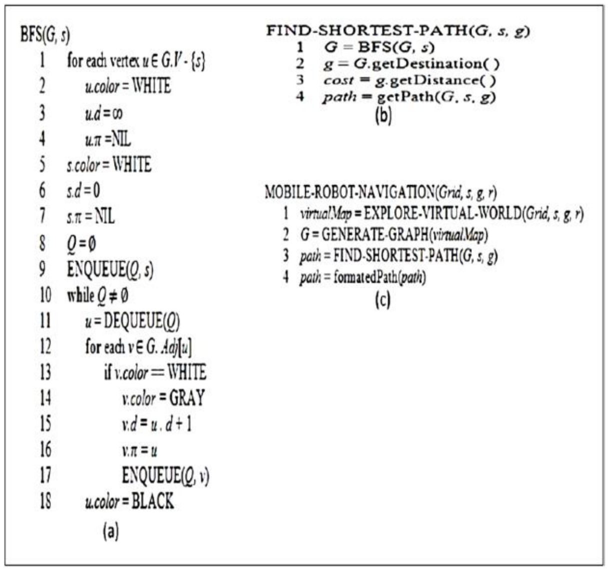
The shortest-path-weight $\partial(u,\ v)$ from u to v can be defined as follows,[9]

$$\partial(u,v) = \begin{cases} \min\{w(p) : u \sim v\}, & if\ there\ is\ a\ path\ from\ u\ to\ v \\ \infty, & otherwise \end{cases} \tag{11}$$

A shortest path from u to v is defined as any path p with weight $w(p) = \partial(u, v)$.

There exists several algorithms for finding the shortest path from source to goal. As explained previously these can be categorized in to four different categories: [9]

(i) Single-source shortest-paths problem
(ii) Single-destination shortest-paths problem
(iii) Single-pair shortest-path problem
(iv) All-pairs shortest-paths problem



```
BFS(G, s)
1  for each vertex $u \in G.V - \{s\}$
2      $u.color$ = WHITE
3      $u.d = \infty$
4      $u.\pi$ = NIL
5  $s.color$ = WHITE
6  $s.d = 0$
7  $s.\pi$ = NIL
8  $Q = \emptyset$
9  ENQUEUE($Q$, $s$)
10 while $Q \neq \emptyset$
11     $u$ = DEQUEUE($Q$)
12     for each $v \in G.Adj[u]$
13         if $v.color$ == WHITE
14             $v.color$ = GRAY
15             $v.d = u.d + 1$
16             $v.\pi = u$
17             ENQUEUE($Q$, $v$)
18     $u.color$ = BLACK
                (a)
```

```
FIND-SHORTEST-PATH(G, s, g)
1  G = BFS(G, s)
2  g = G.getDestination( )
3  cost = g.getDistance( )
4  path = getPath(G, s, g)
                (b)
```

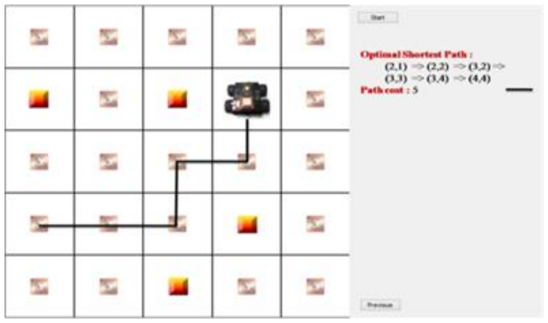
```
MOBILE-ROBOT-NAVIGATION(Grid, s, g, r)
1  virtualMap = EXPLORE-VIRTUAL-WORLD(Grid, s, g, r)
2  G = GENERATE-GRAPH(virtualMap)
3  path = FIND-SHORTEST-PATH(G, s, g)
4  path = formatedPath(path)
                (c)
```

Figure 16. (a) BFS algorithm [9], (b) FIND-SHORTEST-PATH algorithm, (c) MOBILE-ROBOT-NAVIGATION algorithm

similar to that of in BFS. The BFS algorithm accepts a graph G, and a source vertex s and finds the shortest distance from s to each vertex in the graph G. It also produces a "breadth-first tree", which contains all reachable vertices having s as the root. For any reachable vertex v from s, the simple path from s to v in the breadth-first tree corresponds to a "shortest path" from s to v in graph G, that is, a path which contains the smallest number of edges. The algorithm is applicable on both directed as well as undirected graphs. The breadth-first-search procedure BFS as shown in Figure 16.(a) assumes an adjacency list representation of the input graph G. Each vertex in graph G is defined with different attributes like color, distance (d), parent vertex(π). The color and predecessor of each vertex $u \in V$ is stored in the attributes u.color and u.π respectively. Any vertex u with no predecessor (the case where, u = s or u has not been discovered), is declared as u.π = NIL. The distance from the source s to vertex u computed by the algorithm is hold by the attribute u.d. The set of gray vertices are managed in a first-in, first-out queue Q by the algorithm. [9]

The FIND-SHORTEST-PATH algorithm as shown in Figure 16.(b) uses the BFS in an effective way. It takes a graph G (which is generated at the exploration phase), a source vertex (the robot's source position), and a goal vertex(the destination of the robot). The output is a path-sequence (i.e. from source to goal).

In the FIND-SHORTEST-PATH procedure at line-1 BFS is called with parameters G (graph) and s (source). From this updated graph the destination node is extracted in line-2. Line-3 obtains the path-cost from source to goal. Line-4 generates the path-sequence from source s to goal g in the graph G.



Optimal Shortest Path :
(2,1) => (2,2) => (3,2) =>
(3,3) => (3,4) => (4,4)
Path cost : 5

Figure 17. Navigation of the mobile robot in the devised shortest path from source (2, 1) to goal (4,4).

The robot calls the MOBILE-ROBOT-NAVIGATION procedure (Figure 16.(c)) to get the path-sequence. Line-1 explores the virtual world and returns a virtual map which is used in line-2 for generating the graph. In line-3 the FIND-SHORTEST-PATH procedure finds the shortest path from source s to goal u. Finally line-4 formats the path-sequence to proper format of robot co-ordinates and returns it which is sent to the robot. The robot navigates in the resultant path directly without looking for obstacles. We proof that the resultant path is the optimal shortest path. The result of MOBILE-ROBOT-NAVIGATION algorithm is shown in the following figure (Figure 17.).

5. RESULT ANALYSIS

As shown in Figure. 12, the paths 1, 3 and 4 shows the minimum path cost that is 5. Our algorithm FIND-SHORTEST-PATH also finds the shortest path with the path cost 5. In the previous case the robot has to explore all paths and then only the shortest path can be obtained. If each time the robot explores for all possible path then there will be high complexity. However FIND-SHORTEST-PATH algorithm is able to find the shortest path from any node (source) to any other reachable node (goal) without exploring all paths. Hence this algorithm is proved to be more effective. The following table (Table 2.) shows the paths and their cost obtained by the robot at the phase of exploration.

Table 2.Explored paths and their cost

Sl No.	Paths	Cost
1	(2,1)=>(3,1)=>(3,2)=>(3,3)=>(3,4)=>(4,4)	5
2	(2,1)=>(3,1)=>(3,2)=>(4,2)=>(5,2)=>(5,3)=>(5,4)=>(4,4)	7
3	(2,1)=>(2,2)=>(2,3)=>(3,3)=>(3,4)=>(4,4)	5
4	(2,1)=>(2,2)=>(3,2)=>(3,3)=>(3,4)=>(4,4)	5
5	(2,1)=>(2,2)=>(3,2)=>(3,3)=>(3,4)=>(3,5)=>(4,5)=>(4,4)	7

As shown in the above table, paths 1,3 and 4 have lost cost of 5 however paths 2 and 5 have high cost of 7. The FIND-SHORTEST-PATH algorithm return the path (2,1) => (2,2) => (3,2) => (3,3) => (3,4) => (4,4) with cost 5 which is the optimal shortest path.We also prove the correctness of our algorithm through the help of **cyclomatic complexity.**

5.1 Cyclomatic Complexity

The cyclomatic complexity is defined as,

$$V(G) = E - N + 2 \tag{12}$$

Where, V(G) is the cyclomatic complexity, E is the number of flow graph edges, N is the number of nodes.

In our experiment,

N = 6 and E = 5 hence V(G) = E - N + 2 = 5 - 6 + 2 = 1

For paths 2 and 5 the cyclomatic complexity is calculated as follows:

N = 8 and E = 7 =>V(G) = E - N + 2= 7 - 8 + 2 = 1

For paths 1, 3 and 4 the cyclomatic complexity is calculated as follows:

N = 8 and E = 7 =>V(G) = E - N + 2 = 5 - 6 + 2 = 1

In all these cases the cyclomatic complexity is same i.e. 1 but the path cost is different. However the following case (Figure 18.) shows that the cyclomatic complexity is not 1.

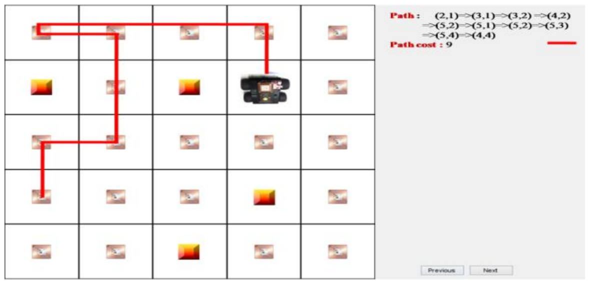

Path : (2,1)=>(3,1)=>(3,2)=>(4,2)
=>(5,2)=>(5,1)=>(5,2)=>(5,3)
=>(5,4)=>(4,4)
Path cost : 9

Figure 18. A case showing V(G) > 1

As shown in the above figure(Figure 18.) the robot starts navigation from (2,1), navigates to cells (3,1), (3,2), (4,2), (5,2), (5,1). However it detects an obstacle in cell (4,1), and there is no path to move forward from (5,1) as it is one of the boundary co-ordinate. Hence the robot returns back to cell (5,2) where it finds only one free cell i.e. (5,3) and navigates to that cell. From (5,3) only one free cell is present i.e. (5,4), hence the robot navigates to cell (5,4). Now from (5,4) there are two free cells to navigate, (5,5) and (4,4). Robot can choose any of them. Let the robot chooses cell (4,4) and navigates to that cell and stays there, because the goal is reached. In this case, N = 9 and E = 9

Hence the cyclomatic complexity;

$$V(G) = E - N + 2 = 9 - 9 + 2 = 2$$

Our algorithm FIND-SHORTEST-PATH suggests that V(G) of the resultant path is 1 and that of the new path is 2. Again the path cost of the shortest path algorithm is found to be 5 where as that of the new path is 9.

This proves that our algorithm FIND-SHORTEST-PATH is the optimal one. Normally the cyclomatic complexity is high when the nodes (vertices) are repeatedly visited. In this case for each navigation path-cost is one. Hence if the robot moves from a node say u to another node say v, and returns back to the parent node i.e. u, then the path-cost will be counted as two, however the number of nodes is same i.e. two. The cyclomatic complexity will be greater than 1 if and only if number of edges traversed is greater than or equal to the total number of vertices. Mathematically,

$$V(G) > 1, \text{ if } |E| \geq |V|$$

Where E is the total number of edges traversed in the devised (including repetition),

V is the set of vertices in the devised path.

6. CONCLUSION

In this research we assume the grid to be fixed 5 x 5. However the obstacles are dynamic. Now a day's landmark-based-navigation for mobile robots are widely used [10]. We have planned to include landmarks in the virtual world. Basically landmarks are required for tracing out the checkpoints. Suppose, we the human being are placed in an unknown area let's say in a forest and are required to find out a way to outside the forest. In this situation what we basically do is, we make marks on certain objects while searching for the goal, so that whenever we get back to that point again, we trace out the visited path. This reduces the number of searching, which in turn reduces time. The same concept is applied in robotics.

When landmarks are included in the virtual world, the robot has to visit all landmarks. Hence in our future work we will improve the FIND-SHORTEST-PATH algorithm so that the shortest path from source to goal node by visiting all the landmarks will be obtained. Our idea is to use an algorithm for finding the nearest neighbor landmark (checkpoint) at each iteration so that the optimal shortest path will be generated.

In this current research we have focused on the path planning of a mobile robot in a grid-based single room environment. However there can be situations in which they may need to travel in a locations like, hospital, college, building etc. Hence our future plan is to develop an algorithm which will help the robot in finding the shortest path in a multi-room environment. This means the source my in a room, where as the goal may in another room. The robot will navigate in the explored shortest path without colliding any boundary wall or obstacles.

In this research we have focused on a single mobile robot navigation. Our research can lead to the development of autonomous vehicle which will be helpful in university campus. This type of vehicle don't need external driver. Only specifying the source and goal location the robot (vehicle) will reach the target in a shortest path. We can also develop autonomous electric wheel chair which will be very much helpful to blind people.

7. REFERENCES

[1] Yung-Fu Hsu, Chun-Hao Huang, Way-Ren Huang, and Woei-Chyn Chu, "An Improved Minimum-Cost Path finding Algorithm for Mobile Robot Navigation", Journal of Advances in Computer Network, Vol. 1, No. 3, September 2013.

[2] Sunhong Park and Shuji Hashimoto, "Indoor localization for autonomous mobile robot based on passive RFID", Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetic, Bangkok, Thailand, February 21 - 26, 2009.

[3] Byoung-Suk Choi, Joon-Woo Lee, Ju-Jang Lee and Kyoung-Taik Park "A Hierarchical Algorithm for Indoor Mobile RobotLocalization Using RFID Sensor Fusion", IEEE Transactions On Industrial Electronics, Vol. 58, No. 6, June 2011.

[4] Ali Asghar Nazari Shirehjini, Abdulsalam Yassine, and Shervin Shirmohammadi,, "An RFID-Based Position and OrientationMeasurement System for Mobile Objects in Intelligent Environments", IEEE Transactions On Instrumentation AndMeasurement, Vol. 61, No. 6, June 2012.

[5] Dirk Hihnel, Wolfram Burgard, Dieter Fox, Ken Fishkin, Matthai Philipose, "Mapping and Localization with WID Technology",Proceedings of the 2004 IEEE International Conference on Robotics & Automation, New Orleans, LA April 2004.

[6] Wail Gueaieb, and Md. Suruz Miah, "An Intelligent Mobile Robot Navigation Technique Using RFID Technology", IEEE Transactions On Instrumentation And Measurement, Vol. 57, No. 9, September 2008.

[7] Lucia Vacariu, Flaviu Roman, Mihai Timar, Tudor Stanciu, Radu Banabic and Octavian Cret, "Mobile Robot Path Planning Software and Hardware Implementations", Proceedings of the 3rd European Conference on Mobile Robots, EMCR 2007, September 19-21, 2007, Freiburg, Germany.

[8] Robin J. Wilson Introduction to Graph Theory, Fourth Edition, Addison Wesley Longman Limited, 1996. pp. vii.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C Stein Introduction to Algorithms, Third edition, The MIT Press, Cambridge, MA, 2009, pp. 589-644.

[10] Stephen Marsland,Ulrich Nehmzow, Tom Duckett, "Learning to select distinctive landmarks for mobile robot navigation", ELSEVIER, Robotics and Autonomous Systems 37 (2001), pp. 241–260.