Path finding and guidance algorithms for autonomous mobile robots in a tree structure network

S Nanu¹, G Belgiu², A-M Dan¹, A Berdich¹ and A Anistoroaei¹
¹Politehnica University of Timisoara, Department of Automation and Industrial Informatics, Timisoara, Romania
²Politehnica University of Timisoara-Romania, Department of Machine Manufacturing Technology, Timisoara, Romania

E-mail: sorin.nanu@upt.ro

Abstract. Considering an autonomous mobile robot that has to navigate indoor and outdoor from a certain point towards another certain point in a tree structure, the paper is giving a solution for finding the path and navigation. The mathematical model is proposed for a number \( n_{DP} \) of departure/destination points (DP), a number \( n_{IP} \) of intersection points (IP) and connections between them in a tree structure. Based on mathematical model, the problems solved and presented in this paper are i) to find a path from any departure point to any destination point, ii) to generate an algorithm for guidance of an autonomous robot to follow the path. Both algorithms were tested on Matlab Simulink® environment. An implementation of algorithms in a pseudocode is presented. The algorithms were adapted to be implemented on a microcontroller based board. The advantage of algorithms and implementation is simplicity. The outcome is less memory consumption and increased computing and execution speed.

1. Introduction
In the context of developing the autonomous robots for transportation, used in industry and agriculture, together with the great need of automation, in order to increase the efficiency, the path finding and guidance are of great interest. Manufactories, warehouses, large industry workshops transport are places where autonomous mobile robots can be used for unattended transportation.

There are many solution for orientation and storing maps in indoor environments, presented in different paperwork. One way to store a map is by scanning a range and keep data in specified polar coordinates [1]. Other robots orientate themselves by photoelectric scanning of the area and based on some landmarks they know where they are and where to go [2]. Another way is by defining a finite state machine [3], and robot can perceive it’s state (by sensors) and a set of possible actions. It chooses an action and receive from the environment a new state. This system was used on Sony Aibo social dog robot.

A bidirectional search graph is based on another algorithm which returns a sequence of vertices that are visited to reach the destination. Per iteration, expand each path in the frontiers with their neighbouring nodes until the frontiers intersect [4]. Also on graphs are based A star algorithm and Dijkstra. They are used for multiple mobile robots path planning algorithms [5-7].

Another solution uses Petri networks in robots guidance [8]. In [9] is used an algorithm, somehow similar to that presented in this paper, that uses a compass sensor and linear encoder.
to guide robots indoor and to avoid obstacles also. A different principle of guiding mobiles is [10] where GPS is used.

2. Mathematical model of the track
The solution presented in paper provides the movement planning of an autonomous robot that runs on a predefined track using line follower, in order to go from a certain point to another. The robot has to recognize the path and the intersection.

The track is formed of Departure/Destination Points (DP), Intersection Points (IP) and connections between them, that represents the available paths, in a structure of a tree, as in example in figure 1. The example is not reducing the degree of generality.

Figure 1. Example of tree structure of track.

The structure is given a main arbitrary direction, in this example, towards north. The mobiles can depart or arrive in any DPs, following in any direction the connections specified by paths from departure DP towards destination DP, through intermediate IP. In a network there are \( n_{DP} \) Departure/Destination Points and \( n_{IP} \) Intersection Points.

It is also restricted that every path enters and leaves the DPs and IPs only to/from rectangular directions (called north, east, south, west).

The objective of this paper is to describe the algorithm for determining:

\( a) \) The sequence of all intermediate IPs the mobile has to follow from departure to destination

\( b) \) The guidance of mobile in every IP, i.e. which direction to exit from an intersection, relative to entering point.

In order to model the Intersection Point (IP), the direction of entering/leaving points are coded arbitrary but consecutively, as in figure 2. Considering main reference orientation towards north, east, north, west and south entering and leaving points are coded "1", "2", "3" respectively "4".

Description of all connections between DPs and IP, that means all the paths the mobile can follow, is made through two matrices.

Figure 2. Model of the intersection point.
2.1. Model of the Departure/Destination Points (DP)

DP Matrix (relative to previous example, presented in figure 3) has two columns and n_{DP} lines.

Column 1 represents the IP connected to each DP (DP_{j} corresponds to line j), DP_{j}, 1) is the IP connected to DP_{j}.

Column 2 represents the code of entering direction from DP_{j} into Intersection Point DP_{j}, 1). e.g. From DP2, the path enters IP1 from direction 3.

```
IP Dir
1 1  DP1 enters IP1 from direction 1
1 3  DP2 enters IP1 from direction 3
5 2  DP3
6 2  DP4
3 1  DP5
4 1  DP6
4 2  DP7
5 1  DP8
6 3  DP9
7 3  DP10
7 2  DP11 enters IP7 from direction 2
```

Figure 3. DP Matrix.

2.2. Model of the Intersection Points (IP)

IP Matrix (example relative to this example is presented in figure 4) has n_{IP} x n_{IP} size. Its non-zero element in position IP(k,l) shows a direct connection between IP_{k} and IP_{l}. A zero element in position IP(k,l) shows that there is no direct connection between IP_{k} and IP_{l}.

```
IP =
0 0 4 0 0 0 0  |  IP1 enters IP3 from direction 4
0 0 2 4 4 0 4  |  IP2 from IP2 to IP1
2 4 0 0 0 0 0  |  IP3 enters IP1 from direction 2 and enters IP2 from direction 4
0 3 0 0 0 0 0  |  IP4
0 2 0 0 0 1 0  |  IP5
0 0 0 3 0 0 0  |  IP6
0 1 0 0 0 0 0  |  IP7 from IP7 to IP1
```

Figure 4. IP Matrix.

The value of IP(k,l) element specifies the entering direction into IP_{i} (column) from IP_{k} (line).

These two matrices describes completely both the direct connections between DPs and IPs and the entering directions in all IPs (from DP or from other IP).

3. Algorithm for path finding

The objective of the algorithm is to find the sequence of IPs to follow by robot in order to run from departure point (named DP_{i}) to destination point (named DP_{j}). The input of the algorithm are the departure point and destination point. The output of the algorithm is a row vector called roadVector() whose elements are in order the indexes of IPs to follow from departure to destination.

The steps followed in this algorithm are:
a) From DP matrix, corresponding to line i, column 1, the index of first IP (let it be IP<sub>k</sub> and called IP<sub>first</sub>) is extracted and is included as first element of roadVector(). In the same manner, the IP that is the last IP in row is that one that is connected to destination point DP<sub>j</sub>. Let it call IP<sub>last</sub>

b) From IP matrix, corresponding to k line, all IPs that are connected directly with IP<sub>k</sub> and are possible next IP are seen as non-zero elements. They are included in a vector called possibleIP(). In this stage of development of algorithm, vector possibleIP() contains all IPs connected directly to IP<sub>k</sub>

c) If possibleIP() vector contains IP<sub>last</sub> element, the algorithm is concluded and roadVector= (IP<sub>first</sub>, IP<sub>last</sub>)

d) If possibleIP() vector contains only one element, then the search will be done further after this element.

e) If possibleIP() vector contains more than one element, the first element of possibleIP() vector is searched identically as in step 3.2. The new possibleIP1() vector is generated and the search is done in the same manner.

f) The difference between the second and first elements is that from the possibleIPi() vector are removed

   i) the IPs searched before, (they are saved in a vector visitedIP()), in order to avoid repeated

   search on the same IP,

   ii) the IPs connected directly to a DP that are different from destination point DP<sub>j</sub>, saved in

   deadEnd() vector,

   iii) the IPs that are already included in roadVector

Consequently, starting from the second step, the visitedIPi() vector is formed of all possible IPs in the current line after removing a), b), c) elements.

g) The tree structure of path assures that there is a unique solution for a given pair of DP<sub>i</sub> and DP<sub>j</sub>.

h) The roadVector() is available for using in the guidance procedure.

4. Algorithm for guidance

The robot, on its way from departure point up to destination point, has to pass through intersection points. Every IP has four enter/exit directions, and the robot must be told which direction has to follow since it enters the IP. Consequently, the robot can perform only three types of movement: rotation to the right (coded as 1), rotation to the left (coded as 3), move straight forward, (coded as 2).

The principle of computing the movement is presented with the aid of figure 5. In the picture there are three IP that will be crossed by robot. The robot goes from IP<sub>k-1</sub> to IP<sub>k</sub> and further on towards IP<sub>k+1</sub>. The problem is when robot detects the intersection, which way to go?

In this case, he has to rotate to the left. The movement is computed in this way:

The movement code is obtained as a difference from the code of exit direction minus the code entering direction. If the difference is a negative number, value 4 will be added. In this way, the result is always 1 (rotate to the right), 2 (move straight ahead) or 3 (rotate to the left).

In this example, exit direction code = 2, enter direction code = 3. Movement code = 2-3+4=3 that means rotate to the left. The formula is general valid.

The guidance algorithm uses the mathematical model of the track (IP matrix and DP matrix) together with roadVector() and provides a vector of movement for each IP named movementVector(), whose size is equal to size of roadVector().

Figure 5. Relative to movement of robot in IPk
The steps of the algorithm are:

a) Take the index of the first IP ($IP_1$) from roadVector() and determine from DP matrix the enter direction code and from IP matrix the exit direction code. This is obtained by finding the entering direction to $IP_1$ from next IP in roadVector(), $IP_2$.
b) Make the difference between exit direction code and enter direction code, adding 4 if result is negative.
c) Add the result in movementVector().
d) Repeat for next IP until end.

5. Algorithm implementation

For implementation, a pseudocode was generated, that calculates the angle that robot has to rotate in each intersection.

%input data:
% IP – Intersection Point Matrix
% DP – Departure Point matrix
% uvect – Path vector (e.g. uvect=[1 1 3 5] is the Path DP1 → IP1 →IP3 → DP5 )
% Nuvect – Intersection Points Number in the Path

%output data:
% yvect – Direction vector for the Intersection Points in the Path
% Theta – Rotation angle of the Robot

%algorithm
Input_IP:=zeros(Nuvect);
Output_IP:=zeros(Nuvect);

% The input direction of every IP in the Path
Input_IP(1):=DP(uvect(1),2);
for i=2 to Nuvect
    Input_IP(i):=IP(uvect(i),uvect (i+1));
endfor

% The output direction of every IP in the Path
for i=1 to Nuvect-1
    Input_IP(i):=IP(uvect(i+2),uvect (i+1));
endfor
Output_IP(Nuvect):=DP(uvect(Nuvect+2),2);

% Direction vector in IP
for i=1 to Nuvect
    if yvect(i)<0 then
        yvect(i):=yvect(i)+4;
    endif
endfor

% Rotation angle of the robot in IP
for i=1 to Nuvect
    case yvect(i) of
        1: Theta(i):=-90
        2: Theta(i):=0
        3: Theta(i):=90
    endcase
endfor
For this code, a simulation program was written in Matlab.

6. Example
Considering the model in figure 1, departure point DP1 and destination point DP9, the path finding algorithm will provide the path in figure 6:
roadVector()= (1,3,2,5,6) that means the robot has to run in order through IP1, IP3, IP2, IP5, IP6.
The guidance vector is:
movement Vector()=(1, 2, 2, 3, 2) that means at first IP has to take right, at the second and third IP has to go straight ahead, at the fourth has to go to the left and straight ahead again.
   In figure 7 is presented the Matlab simulation.

![Figure 6. Example.](image)

7. Conclusions
Paper presents a solution for guiding a mobile robot on a predefined tree structure track when departure and destination points are provided. The model of the tree track and algorithms for track finding and guidance were meant for implementing in embedded systems. The algorithms were simulated in Matlab. For a structure with \( n_{\text{IP}} \) intersection points, \( n_{\text{IP}}^2/2 \) connections and \( n_{\text{DP}} \) departure/destination points, the EEPROM memory requested for model is \( n_{\text{IP}}^2 + 2n_{\text{DP}} \), while estimated RAM memory for variables is \( n_{\text{IP}}^3 \) [11]. This approach was aimed to offer a very simple solution from the memory and speed point of view in order to be implemented on a simple microcontroller with reduced resources. As the track structure is a tree, there are no loops and the solution is unique. We intend, and are working on a more general structure where the track has loops.

8. References
[1] Nguyen V, Gächter S, Martinelli A, Tomatis N, Siegwart R, 2007 A comparison of line extraction algorithms using 2D range data for indoor mobile robotics, Published online: 8 June 2007 © Springer Science+Business Media, LLC.
[2] Huang Z, Zhu J, Yang L, Xue B, Wu J, Zhao Z 2015 Accurate 3-D Position and Orientation Method for Indoor Mobile Robot Navigation Based on Photoelectric Scanning, IEEE Transactions on Instrumentation and Measurement, 64(9) 2518-2529.
[3] Perez X, Angulo C, Escalera S, Pardo D, 2010 Vision-based Navigation and Reinforcement Learning Path Finding for Social Robots, Barcelona, Spain.
[4] Fitzgerald, T., 2012 A System for Bidirectional Robotic Pathfinding Portland State Univ., Portland.

[5] Zhang Z, Zhao Z, 2014 A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm, 1,2 College of Computer and Information Engineering, Tianjin Normal University, Tianjin, China.

[6] Mahadevi S., Shylaja K. R., Ravinandan M. E. 2014 Memory Based A-Star Algorithm for Path Planning of a Mobile Robot, International Journal of Science and Research 3(6), 1351-1355.

[7] *** Path-finding with Dijkstra’s Algorithm
http://files.kipr.org/gcer/2011/proceedings/Fletch

er_Woodruff_E_Final.pdf, accessed on 24March 2021.

[8] Khosravi A, Danesh M, Khosravi M, 2013 Esigning A Robot’s Guidance Algorithm Through Petri Nets For Following A Mobile Target Using The Data Of Several Sensors, Bulletinul Institutului Politehnic from Iași, March 25.

[9] Mutlu L, Uyar E, 2012 Indoor Navigation and Guidance of an Autonomous Robot Vehicle with Static Obstacle Avoidance and Optimal Path Finding Algorithm, September 12-14, 2012. Sofia, Bulgaria.

[10] Simon D, El-Sherief H, 1994 Real-Time Navigation Using the Global Positioning System, SIG Technology, Summer http://academic.csuohio.edu/simond/pubs/SIGTechReview.pdf, accessed on 24March 2021.

[11] Nanu S, 2007 Educational Aspects in Control System Design Technology, Scientific and Technical Bulletin, Series: Electrotechnics, Electronics, Automatic Control and Computer Science UAV Arad, 4(1) 584-585.