MOBILE ROBOT NAVIGATION IN DYNAMIC ENVIRONMENTS USING PATH PLANNING ALGORITHMS WITH FOCUSING HEURISTIC

Gheorghe LAZEA, Radu ROBOTIN,
Sorin HERLE, Florin COTOFAN

Technical University of Cluj-Napoca, Department of Automation
26-28 Baritiu Str., Cluj-Napoca
E-mail: {gheorghe.lazea; radu.robotin; sorin.herle; florin.cotofan}@aut.utcluj.ro

Abstract: This paper describes a version of D* algorithm with focusing heuristic for mobile robot path planning in dynamic environments. The aim is to determine the optimal path to the target using a map able to reflect the changes in the environment, detected using the sensors system. The optimality of the trajectory may be considered from path length point of view, time required to reach the goal position, or energy consumption.

Keywords: mobile robot, path planning, navigation algorithm, focusing heuristic.

1. INTRODUCTION

Navigation is the science (or art) of directing the course of a mobile robot as it traverses the environment (land, sea, or air). Inherent in any navigation scheme is the desire to reach a destination without getting lost or crashing into anything (McKerrow, 1995).
The navigation module comprises three sub-modules: mapping, planning and driving,, as presented in figure 1.

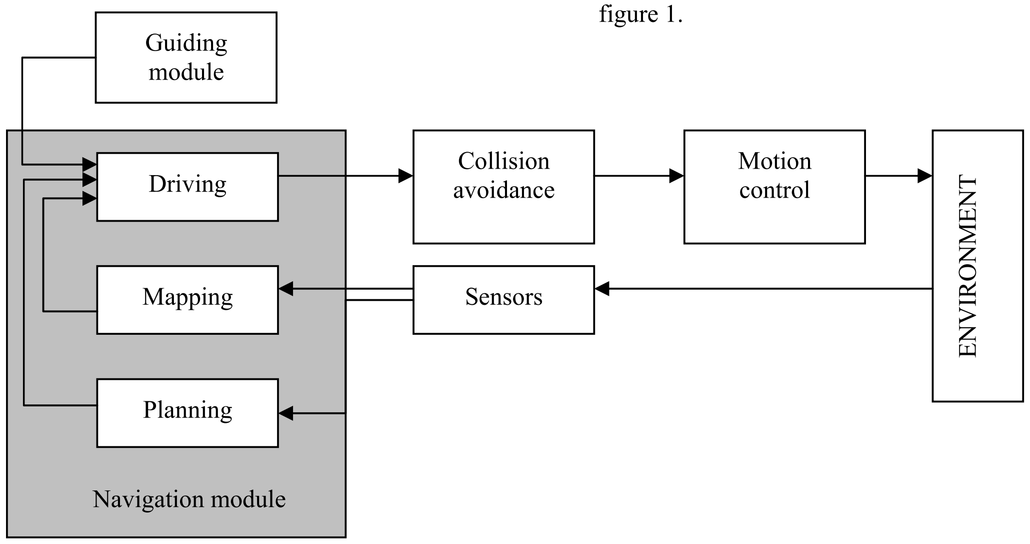


Fig.1. System architecture and mobile robot – environment information flow.

The ability to cross dynamic environments is necessary for any real robot. From technical point of view, the presence of dynamic obstacles increases the difficulty of motion planning. In this case, the shortest path from the length point of view is not necessarily the shortest form time expenditure point of view. Therefore optimality must involve both time and path length. Moreover, if the energy consumption is another term involved in the optimality equation, special attention must be paid when velocity and acceleration vectors are generated. The security of the mobile robot is another important issue. Most of the time, a "safe" trajectory tends to be inefficient form the length point of view, therefore the aim is to generate trajectory without loosing too much efficiency.

2 PATH PLANNING

Considering that at a certain moment of time the environment structure is known, it may be represented in a map to be used by the path planner task to generate the mobile robot trajectory as a directed graph (a sequence of states, starting from initial state and ending in the goal state). This trajectory is optimal if the sum of all transitions (arc costs) is minimum with respect to all possible transitions in the graph. The states represent possible robot locations while arc cost represent the cost of mobile robot traverse from one state to the adjacent one. If, during the path traverse one or more arcs are found to be inaccurate (i.e. an obstacle initially known has disappeared or, a new obstacle has been detected) the remaining of the path must be re-planned.
There are several algorithms suitable for optimal search in a graph with variable arc costs. An example of such algorithm is A* algorithm (Winston, 1984) that uses an a-priori available map. The robot follows the initial planed path until it discovers a discrepancy between the map and the sensors extracted data. At this point a re-planning may be necessary, thus the algorithm may prove to be inefficient when the distance between the start point and the target point is large, due to the high number of necessary re-planning operations.

The D* algorithm (Stentz, 1994) may be used to generate optimal paths for a mobile robot using all the available a-priori information (given as a map) and sensor data. The map may be accurate, incomplete or even empty. The arc cost may change during the search, thus the algorithm is suitable for use in dynamic environments. The problem of mobile robot navigation may be defined as the problem of finding the optimal path in a directed graph, in which the arc labels represent the cost of traverse form the current state to the adjacent one. The robot sensors measure the arc costs in the vicinity of the robot and the information is included in the map. The robot starts at a particular state and moves across arcs (incurring the cost of traversal) to other states until it reaches the goal state, denoted by G. Every visited state except G has a backpointer to a next state denoted by b(X)=Y. D* uses backpointers to represent paths to the goal. The cost of traversing an arc from state to state is a positive number given by the arc cost function c(X,Y). If Y does not have an arc to X , then c(X,Y) is undefined. Two states X and Y are neighbors in the space if c(X,Y) or c(Y,X) is defined. In figure 2 is presented the graph representation of the current robot location X and its neighbors.

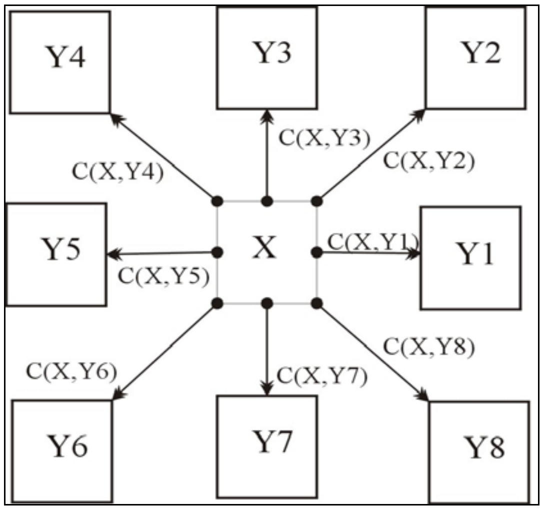


Fig. 2. Neighbors & cost function of a state X in the robot space

The algorithm consists mainly of two functions: Process_State and Modify_Cost. Function Process_State is used to compute optimal trajectories to the goal, while function Modify_Cost is used to propagate cost changes to the neighbouring states.

3. FOCUSING HEURISTIC ALGORITHMS

Starting form initial D* algorithm, a focussing heuristic algorithm may be implemented. It uses an OPEN list to propagate information about changes to the arc cost function and to calculate path costs to states in the space. Referring to figure 2 again, every state *X* has an associated tag *t(X)*, such that if *X* has never been on the list *t(X)=NEW*, if *X* is currently on the list *t(X)=OPEN*, and if *X* is no longer on the list *t(X)=CLOSED*. For each visited state *X*, D* maintains an estimate of the sum of the arc costs from *X* to *G* given by the path cost function *h(X)*. Given the proper conditions, this estimate is equivalent to the minimal cost from state *X* to *G*. For each state *X* on the OPEN list (i.e., *t(X)=OPEN*), the key function, *k(X)*, is defined to be equal to the minimum of *h(X)* before modification and all values assumed by *h(X)* since *X* was placed on the OPEN list. The key function classifies a state *X* on the list into one of two types: a RAISE state if $k(X)<h(X)$, and a LOWER state if $k(X)=h(X)$. The algorithm uses RAISE states on the OPEN list to propagate information about path cost increases and LOWER states to propagate information about path cost reductions. The propagation takes place through the

repeated removal of states from the list. Each time a state is removed from the OPEN list, it is expanded to pass cost changes to its neighbors. These neighbors are in turn placed on the OPEN list to continue the process.

States are sorted on the OPEN list by a biased $f(\circ)$ value, given by $f_b(X, R_i)$, where X is the state on the OPEN list and R_i is the robot's state at the time X was inserted or adjusted on the OPEN list. Let $\{R_0, R_1, \ldots, R_N\}$ be the sequence of states occupied by the robot when states were added to the OPEN list. The value of the bias function is given by:

$$f_B(X, R_i) = f(X, R_i) + d(R_i, R_0) \qquad (1)$$

where $f(\circ)$ is the estimated robot path cost given by:

$$f(X, R_i) = h(X) + g(R_i, R_{i-1}) \qquad (2)$$

and $d(\circ)$ is the accrued bias function given by:

$$\begin{aligned} d(R_i, R_0) &= g(R_1, R_0) + g(R_2, R_1) + \\ &\ldots + g(R_i, R_{i-1}) + \varepsilon \end{aligned} \qquad (3)$$

if $i > 0$. If $i = 0$, $d(R_0, R_0) = 0$.

The function *g(X,Y)* is the focussing heuristic, representing the estimated path cost from *Y* to *X*. The list states are sorted by increasing $f_b(\circ)$ value, with ties in $f_b(\circ)$ ordered by increasing $f(\circ)$, and ties in $f(\circ)$ ordered by increasing $k(\circ)$. Ties in $k(\circ)$ are ordered arbitrarily. Thus, a vector of values $\langle f_B(\circ), f(\circ), k(\circ) \rangle$ is stored with each state on the list. Whenever a state is removed from the OPEN list, its $f(\circ)$ value is examined to see if it was computed using the most recent focal point. If not, its $f(\circ)$ and $f_b(\circ)$ values are recalculated using the new focal point and accrued bias, respectively, and the state is placed back on the list. Processing the $f_b(\circ)$ values in ascending order ensures that the first encountered $f(\circ)$ value using the current focal point is the minimum such value, denoted by $f_{\min}$. Let k_{val} be its corresponding $k(\circ)$ value. These parameters comprise an important threshold for D*. By processing properly-focussed $f(\circ)$ values in ascending order (and $k(\circ)$ values in ascending order for a constant $f(\circ)$ value), the algorithm ensures that for all states *X*, if $f(X) < f_{\min}$ or ($f(X) = f_{\min}$ and $h(X) < k_{val}$), then $h(X)$ is optimal. The parameter val is used to store the vector $\langle f_{\min}, k_{val} \rangle$ for the purpose of this test. If R_r represents current robot position, the function $r(X)$ is defined and returns the robot position when the state *X* was removed from the OPEN list.

4. IMPLEMENTATION

The algorithm consists primarily of three functions: Process_State, Modify_Cost, and Move_Robot. Process_State computes optimal path costs to the goal, Modify_Cost changes the arc cost function and enters affected states on the OPEN list, and Move_Robot uses the two functions to move the robot.

The auxiliary routines are:

- *MIN(a,b)* returns the minimum of the two scalar values *a* and *b*;
- *LESS(a,b)* takes a vector of values $\langle a_1, a_2 \rangle$ for a and a vector $\langle b_1, b_2 \rangle$ for b and returns TRUE if *a1<b1* or (*a1=b1* and *a2<b2*);
- *LESSEQ* takes two vectors *a* and *b* and returns TRUE if *a1<b1* or (*a1=b1* and *a2<=b2*);
- *COST(X)* computes $f(X, R_r) = h(X) + GVAL(X, R_r)$ and returns the vector of $\langle f(X, R_r), h(X) \rangle$ values for a state *X*;
- *DELETE(X)* deletes state *X* from the OPEN list and sets *t(X)=CLOSED*;
- *PUT_STATE(X)* sets *t(X)=OPEN* and inserts *X* on the OPEN list according to the vector $\langle f_b(X), f(X), k(X) \rangle$;
- *GET_STATE* returns the state on the OPEN list with minimum vector value (NULL if the list is empty).

The above-presented algorithm was implemented on a mobile robot, equipped with 8 range finding sensors. We have used client-server architecture, with P2OS operating system on robot's microcontroller, and Saphira routines for the client on a PC workstation. The software was implemented using Microsoft Visual C++, and Saphira OS functions using multithread functions, this solution proved to be the most efficient way to connect to the Saphira core. The core itself runs as a high priority thread, with 100ms cycle time.

Figure 3 presents an typical potential wheel problem. The grey obstacles are initially known while the black obstacle is unknown. As figure 3 shows, the grey part in the unknown obstacle is detected by the robot's sensors, the path is considered closed and the

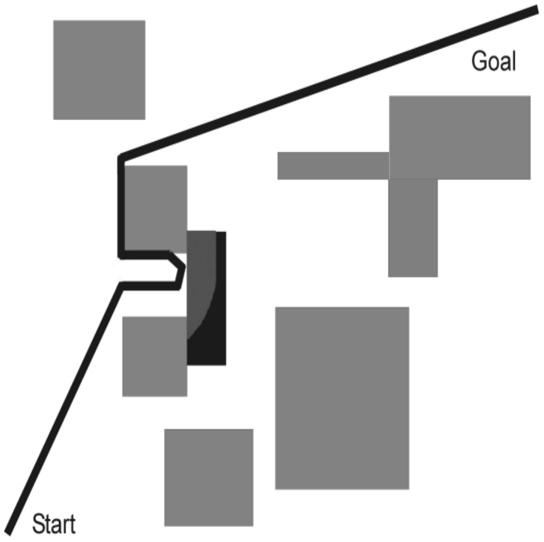
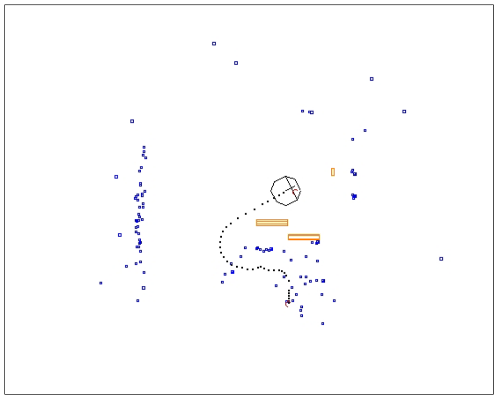
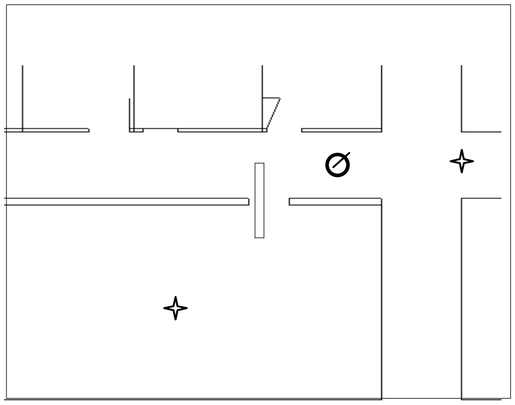


Fig. 3. Trajectory for a typical potential wheel problem.



a)



b)

Fig. 4. Real experiment using Pioneer 2 mobile robot. (a) Saphira navigation display screenshot; (b) environment structure.

trajectory is re-planned. Figure 4.a shows a real experiment using the environment in figure 4.b. No a-priori information was available, the initial trajectory was planned with an optimistic map and the robot had to discover all the obstacles in the environment.

5. CONCLUSIONS

The presented algorithm may handle the mobile robot navigation in dynamic environments, working with a complete spectrum of a-priori information, ranging from a detailed and accurate map, to the absence of information.
The further work will be conducted in order to decrease the response time thus allowing mobile robot cooperation in a multi-agent environment.

REFERENCES

McKerrow, P.J. (1995). *Introduction in Robotics*, Addison-Wesley Pub. Company.
Winston, P.H. (1984). *Artificial intelligence*, Addison-Wesley Pub. Company,.
Stentz, A. (1994). *Optimal path planning for partially known environments*, Available from: http://www.frc.ri.cmu.edu/~axs/ Accessed: 2000-12-1,.
Latombe, J.-C.(1991). *Robot Motion Planning*, Kluwer Academic Publishers, ISBN 0-7923-9206-X, Boston MA.
*** (1998). *Saphira User Manual*, ActivMedia Robotics.

This research was sponsored by CNCSIS in contract no A 34.970/2003, code 39/1194, “Mobile robot navigation in dynamic environments”