Barbara Siemiątkowska
Institute of Fundamental Technological Research,
Polish Academy of Sciences,
Świętokrzyska 21,
00-049 Warsaw, Poland

PLANNING THE MOTION OF A TEAM OF ROBOTS

Abstract

In this paper the following problem is addressed: knowing positions of mobile robots and locations of their goals, find collision free paths leading from each robot to its goal. It is assumed that the robots can move with different speeds. The map of the environment has a form of an occupancy grid. State of each cell of the grid is determined with one of three labels: *occupied, free or unknown.* The robots' trajectories are planned using a modified version of value iteration algorithm. The experimental results demonstrate high computational efficiency of the proposed algorithm of path planning. It is shown that the costs of computations depend approximately linearly on the number of robots as well as on the number of their goals.

1. INTRODUCTION

Multi-robot systems have recently become very popular. It is believed that there are several complex and time-consuming tasks that are difficult or inappropriate for a single mobile robot to deal with. However, a team of cooperating robots could fairly easily accomplish them. Examples include transport of objects, surface cleaning, exploration of an unknown environment, surveillance. Usually, complex tasks are decomposed into subtasks, that may be accomplished by the individual robots or by small teams of them. In any case, there is a need for efficient navigation techniques that would be uniformly suitable for the individual robots, the teams or robots, and the groups of robotic teams, all sharing a common workspace. In this paper we consider a method of path planning that was originally developed for a single mobile robot [4][5], and that maintains its computational efficiency when applied to multiple robot navigation problems. The discussed method of path planning relies on the occupancy grid based representation of the workspace configuration. It is assumed that the environment is at least partially known. State of each cell of the grid can be marked with one of three possible labels: *occupied*, free and *unknown*. The goal of the path planning task is to find collision-free sequences of subsequent neighboring unoccupied grid cells leading from locations of each robot to their corresponding goals, so that the number of traversed cells would be minimal. To solve the task formulated in such a way, we implement a variant of the diffusion algorithm (also known as value iteration in dynamic programming world). The same method is used for global path planning that takes into account only locations of the known static obstacles and for local conflict resolution (or collision avoidance), in which case the other moving robots are considered as well. The remaining part of this paper is organized as follows. Section 2 presents general ideas of our approach to planning a path for a single robot. Its application in the multiple robot case is described in Section 3. Experimental results are reported in Section 4, and followed by conclusions.

2. PATH PLANNING FOR A SINGLE ROBOT

Very often the path-planning problem is defined as follows: knowing the robot's position and the goal position find the continuous and collision free path leading from the robot to the goal. In the described method a map of the robot's environment is represented as a grid of cells. Each cell is labeled either as *free* (traversable), *occupied* (by an obstacle) or *unknown.* In additional to the occupancy grid, another two-dimensional array of cells called diffusion array is maintained in order to determine a direction of the robot's motion after the path has been planned. Each cell of the global grid corresponds to a cell in the diffusion array. The method of value iteration (diffusion method) consists of three stages:

2.1 Initialization

$$c_{ij} = \begin{cases} -1 & if \quad c_{ij} \quad is \quad occupied \\ 0 & if \quad c_{ij} \quad is \quad free \\ R & if \quad c_{ij} \quad is \quad a \quad goal \end{cases} \tag{1}$$

Initially, all cells of the diffusion array which correspond to a region free from the obstacles are set to 0, the cells that represent parts of obstacles are set to -1, and a cell which represents the goal position is set to R (a very large value).

2.2 The diffusion process: update loop

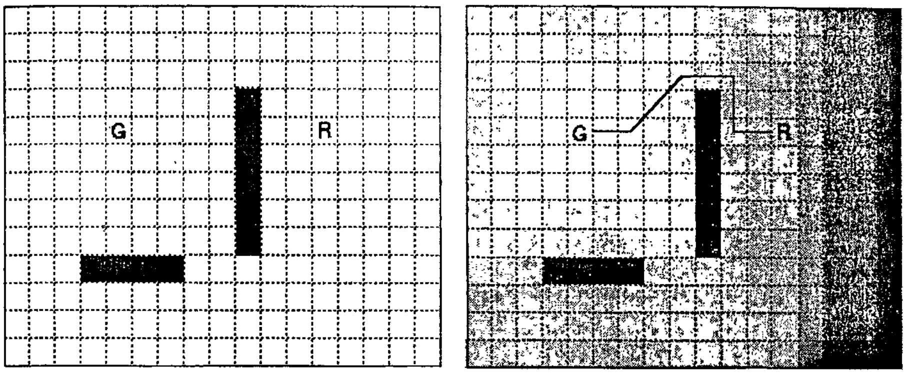
Values of all cells are updated using the following rule:

$$C_{ij} = \begin{cases} C_{ij}^{old} & gdy \quad C_{ij}^{old} < 0 \\ \max\limits_{\substack{k=-1,0,1 \\ l=-1,0,1}}(c_{i+k,j+l}^{old} - dist(C_{ij} - C_{i+k,j+l})) & w \quad przeciwnym \quad przypadku \end{cases} \tag{2}$$

where dist(c_{ij},c_{kl}) is the distance between cells c_{ij} and c_{kl}.
The update rule iterates until stability, that is until $c_{ij}^{new}=c_{ij}^{old}$ for every cell.

2.3 Determining the direction of motion

If c_{ij} corresponds to the current position of the robot, then the motion direction is indicated by that of the neighboring cells which value c_{kl} is the largest. Figure 1 illustrates the idea of path planning. Figure 1a shows the occupancy grid based map of an environment. Black rectangles represent known obstacles, R corresponds to the initial location of the robot, and G is the goal. Figure 1b presents the result of running value iteration algorithm. The lighter the shade of the grid the larger the value of the corresponding cell of the diffusion array. Cells that cannot be traversed are depicted in black. The presented method of the path planning has the following practical advantages [4][5]: it does not suffer from local minima problems (dead-ends are not a problem here); the situation when the goal or the robot is surrounded by the obstacles (and thus the path does not exist) can be easily recognized; it is easily parallelizable, so in case of an appropriate implementation, it would be possible to plan paths at frequencies comparable to reactive navigation. The efficiency of described method strictly depends on the implementation. Usually update loop requires a number of multiplication proportional to N^4 where NXN is the dimension of the diffusion array. But in fact, the following modifications make it possible to significantly speed up the diffusion process:



a) b)

FIGURE 1: *Path planning for a single robot*

1. Because the current c_{ij} value of each cell is computed upon information provided by its neighbors only, it is more efficient to address the neighbors directly rather than by index. If the address of the value c_{ij} is known then the address of the neighbors can be obtainedimmediately: if *address*(c_{ij})=*adr* then *address*(c_{i+1j})=*adr+N* and *address*(c_{ij+1})=*adr+1* (fig. 2)

2. Buffering addresses of cells that are going to be updated in the next iteration provide additional speed-ups of the process.

I-1,J+1	I,J+1	I+1,J+1
I-1,J	I,J	I+1,J
I-1,J-1	I,J-1	I+1,J-1

adr-N+1	adr+1	adr+N+1
adr-N	adr	adr+N
adr-N-1	adr-1	adr+N-1

FIGURE 2: *Part of an array and addresses*

Table 1 presents the average times of path planning for different dimensions of the grid-based maps. The second and the third rows of the table show the results published in [1]. The fourth row corresponds to the results obtained with the method discussed in this paper. All tests were conducted on a Pentium 200 MHz PC.

Number of cells	64X64	128X128	256X256
The method described in[1] – t_0	2.2s	31.7s	502.52s
Modification of the method decribed in [1] – t_1	0.55s	2.36s	9.89s
My implementation t_2	0.003s	0.014s	0.042s
$\frac{t_0}{t_2}$	733.3	2264.2	11964.7
$\frac{t_1}{t_2}$	183.3	168.5	235.4

TABLE 1 *Comparison of different implementation methods*

3. PATH PLANNING FOR A TEAM OF ROBOTS

Motion planning for multiple mobile robots can be defined as follows [2] for any robot find a path such that: the planned path is free from the obstacles; for each robot R_i at any time *t* does not exist a robot R_j, where $i <> j$ which collides with R_i . Collision between robots exists when distance($R_i(t)$,$R_j(t)$) < *safe_dist*. Parameters *safe_dist* depends on size of robots. Typically, in multi-robot systems each robot has its own copy of the global map and its own module of global path planning and obstacle avoidance. When the environment is known, robots are operating in the same area and the target is common for the team, then each robot can generate its own collision free path based on the same, common diffusion array. Figure 2 presents schematically the idea of path planning method for two robots R_1 and R_2 that have been assigned to get to the same target.

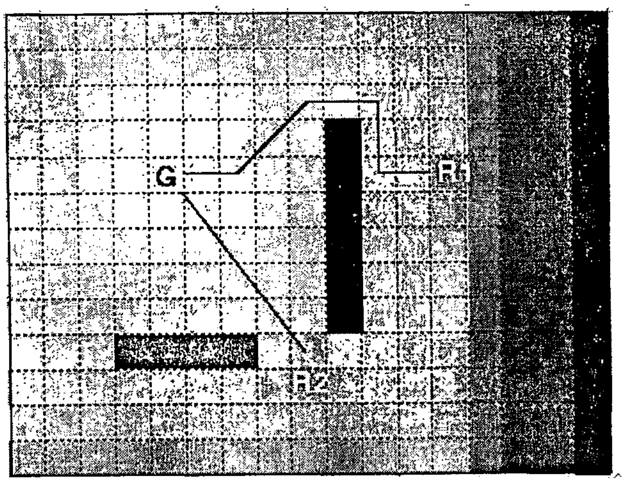


FIGURE 3 *Path planning for a pair of robots*

The same method can be used to avoid collisions between robots. When the cell *ij* is occupied by one of them robot, the corresponding c_{ij} value is temporarily set to -1, so this cell at the moment could not be accessed by another vehicle. When the robot leaves the cell c_{ij}, the original value is restored so it becomes traversable again. Figure 4 illustrates the idea of the method. Initial values of the diffusion array are shown in the Figure 4a, and 4b), 4c), 4d, illustrate the temporary modification of the array.

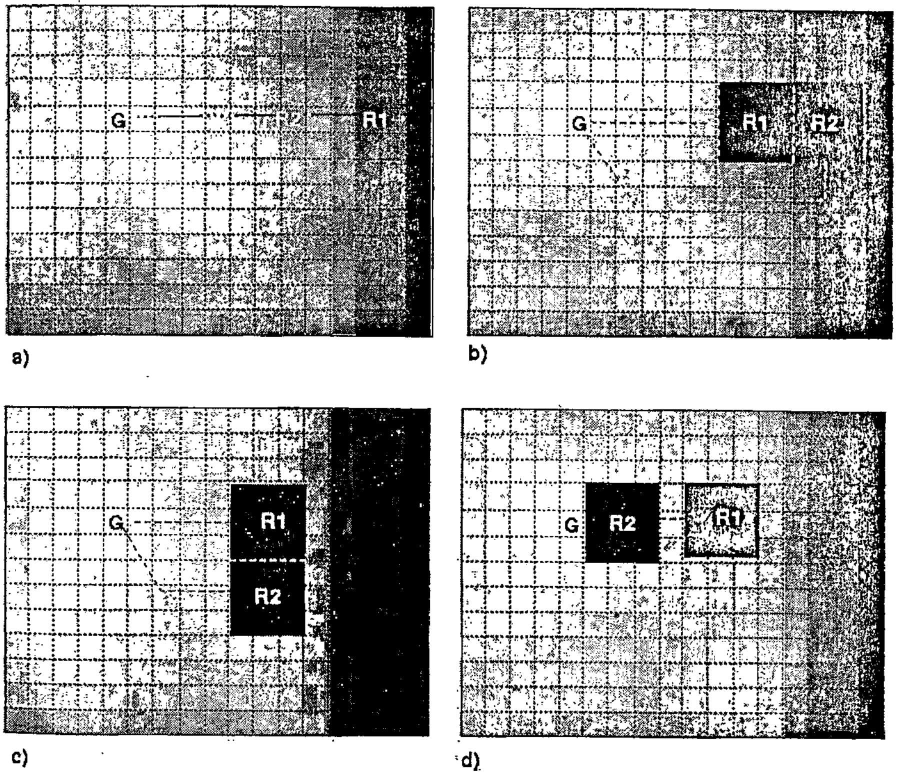
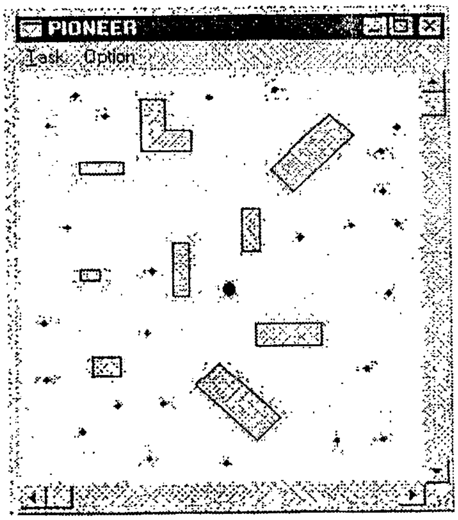


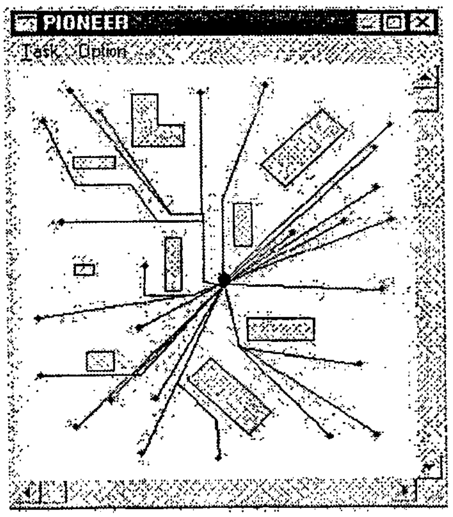
FIGURE 4 *Collision avoidance for a pair of robots*

4. EXPERIMENTS

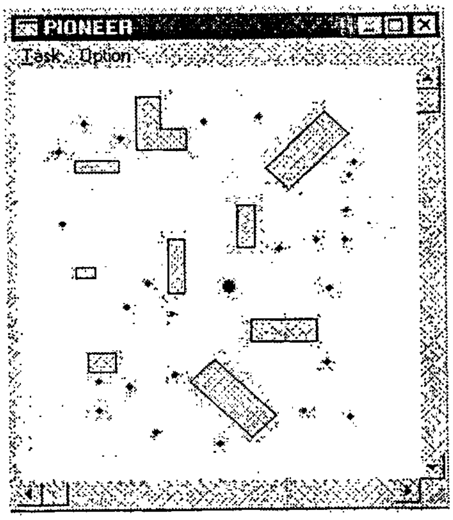
The discussed method has been tested in computer simulations for different sizes of diffusion arrays and different numbers of robots (from 20 to 100) and different number of goals. Figure 5a presents initial configuration of 25 robots (represented by diamonds) and a single goal (represented by circle). Planned paths are shown at figure 5b. Figure 5c and 5d present the intermediate and final configuration of the robots.



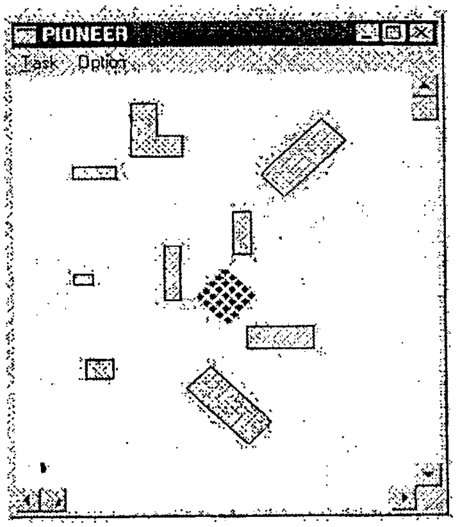
a)



b)



c)



d)

FIGURE 5 *Path planning for many robots and a single goal*

Table 1 in section 2 presents the average time necessary to plan paths for single robots. When the robots are assigned the common goal, the method is almost independent on their number. Table 2 presents comparison between classical method of path planning for multi-robot systems (in this method global path is planned independently for each robot) and the proposed method.

Number of robots	1	20	40	60	80	100
Described method – t_0	0.042s	0.053s	0.063s	0.074s	0.085s	0.095s
Classical approach – t_1	0.042s	0.84s	1.68s	2.52s	3.39s	4.2s
$\frac{t_1}{t_0}$	1.0	10.58	26.6	34.0	40.3	46.6

TABLE 2 *Path planning for different number of robots*

The system was tested in the cases of different number of goals. It is assumed that the dimension of the map is equal 256X256 cells. Table 3 presents average time necessary for path planning for a team of 50 robots and different number of goals.

Number of goals	1	10	20	30	40	50
Described method– t_0	0.07s	0.45s	0.86s	1.31s	1.93s	2.12s
Classical approach – t_1	2.1s	2.1s	2.1s	2.1s	2.1s	2.1s
$\frac{t_1}{t_0}$	31.3	11.3	4.7	1.6	1.08	1.0

TABLE 3 *Path planning for different number of goals*

CONCLUSIONS

This paper discusses a simple and fast method of path planning for individual robots, their teams and groups of teams. Experimental results prove its computational efficiency, and demonstrate approximately just linear increase of costs when adding new robots or new goals to the system. In case when the robots share a common goal, the time necessary to plan the paths for the whole team is almost the same as for a single robot.

ACKNOWLEDGEMENTS

Thanks to Artur Dubrawski for comments and suggestions regarding this paper.

REFERENCES

[1] Urdiales C., Bandera A., Cassanova J., Sanboval F: *Pyramidal Path Planning Algorithm For Autonomous Robots*, Proceedings of the SIRS'99, Coimbra, Portugal, July 1999, pp. 447 - 453.

[2] Chua L. And Young L.: *Cellular Neural Networks*, IEEE Trans. CAS, 35, 1988.

[3] Marchese F.: *Cellular automata in robot path planning*, LNCS 124, Springer-Verlag, Berlin, pp. 116 – 125, 1997.

[4] Siemiątkowska B.: *Cellular Neural Network for Path Planning*, Proceedings of the SIRS'94, Grenoble, France, July 1994, pp. 125 - 130.

[5] Siemiątkowska B., Dubrawski A.: *Cellular Neural Network for a Mobile Robot*, Rough Sets and Current Trends in Computing, Springer, Berlin June 1988, pp.147 - 155.