Collision-free Path Planning for Mobile Robots Based on Extended A* Algorithm

Matous POKORNY[1]

[1]Department of Circuit Theory, Czech Technical University, Technická 2, 166 27 Praha, Czech Republic

pokormat@fel.cvut.cz

Abstract. *Collision-free path planning algorithm for a mobile robot for Eurobot competition is described in this paper. Extended A* algorithm implements new methods of represenatation of the robot in a map. The robot is represented by the real shape and dimensions, not only as one point. Sofisticated checking of the collision-free path is used. This approach was implemented and tested in the robot software. The result is more reliable robot motion in a limited space.*

Keywords

Eurobot competition, path planning, A* algorithm, nonholomic robot, mobile robot, collision-free path.

1. Introduction

The Eurobot competition [1] is a project, which can participate student teams with their robots. The two autonomous mobile robots compete each other on the one playground. During this competition they try to complete the prepared tasks. The task is usualy finding and moving the objects, it is another topic every year. Main meaning of the Eurobot competition is getting and sharing experinces between teams and support science and technology. We participate in Eurobot competition with our university student team Flamingos [2].

Path planning for the mobile robots using in this competition has its specification. The playground has dimension 2 m × 3 m, each robot has dimension approximately 30 cm × 30 cm. Many obstacles are on the playground. It is not much space for the robot motion, which should be precise and quickly (one match takes 90 sec.), therefore the path planning algorithm should determine time-shortest collison-free path from start to goal point in real-time.

Positions of the hard-placed obstacles are known before start of the match. Positions of other variable obstacle are detected by the sensor system of the robot during the match. The oponent robot is the most important and moving obstacle on the playground. It is advantageous to save the positions of all obstacles. They are drawn simply in to the map, which is saved as grid of the square cells. Each cell holds information about concrete place on the playground, for example, obstacle flag, path flag, start point flag and goal point flag.

Usual A* algorithm is the very suitable for a simple mobile robot with the circlular shape. Robot is represented by the point (X_r, Y_r) in the map, fig. 1 (a). Circle around the obstacle is drawn in the map and represents safe surroundings, in which must not be a path (referent point of the robot). The A* algorithm works with the map and checks the expanded map cells for a obstacle flag. This is the principle of the simple collision-free path planning. We assume a general nonholomic mobile robots with the differencial chassis and polygonal shape. If we use simple approach described above, safe suroundings (radius of the rotation), fig. 1 (b), is too big and motion of the robot in limited space on the playground is not possible.

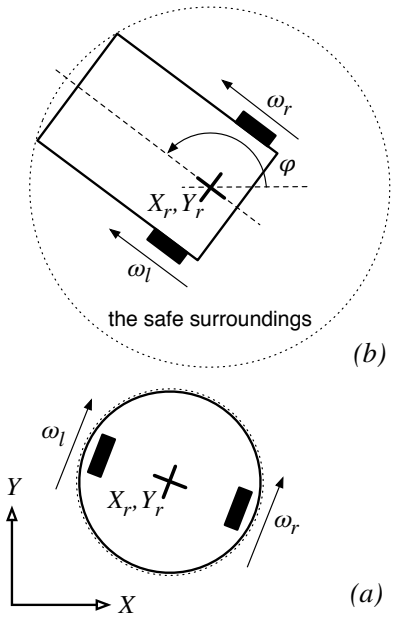

Fig. 1. Simple mobile robot with the circular shape (a) and general mobile robot with the polygonal shape (b).

We should choose another approach. We have complete robot software with simple A* algorithm. Many of the specific processes and dependencies are implemented in

this complex software. We should choose approch that combines functionality and implementation, because extensive changes mean a lot of time of debugging. It is short period for development a new solution. The rules of competition are issued in October and the Eurobot competition takes place in May. Simple and reliable solution is preferred.

The robot is represented by real shape, dimensions and angle of rotation in the map. The bitmap mask rotated to the required angle represents robot in the memory. Work with the mask and its rotation is very similar to the method that is described in [4] and [5], where using neural network. The map is represented by the grid with a square-shaped cells-grid of square cells, which hold information like obstacle, start, goal and path. An approach described in [6] is used in the similar robot and surroundings with limited space, but the map is represented by velocity profiles. In [7] force field is used as map representation, which is filled with data from two laser finders Hokuyo. Implemented algorithm determines two forces which push out the robot from the obstacles and force robot to turn. The action of first force is on the robot nose and the action of second force is on the robot tail. This method is great, but is based on another principles. It means to change robot conception and software completly. We assume extended A* algorithm described below.

2. Extended A* algorithm

The extended A* algorithm works with the map, where the robot is represented by its real shape, real dimensions and its angle of rotation. This algorithm extends simple A* algorithm to using bitmap masks, which represent robot in the map. The masks initialization and free expanded cells checking based on using the masks are described below, other parts are same for both algorithms. It it assumed from reader that the principle of A* algorithm is known.

Algorithm 1 Initialization of the masks

INPUT: Shape and dimensions of the robot
OUTPUT: Bitmap masks allocated in memory m_rot, m_0PI and m_PI4

1: Create vector pattern of mask v_m_0PI for angle 0π based on robot real shape and dimensions
2: Create vector pattern of mask v_m_PI4 by rotation v_m_0PI by angle $\pi/4$
3: Allocate memory for three masks
4: Convert arcs between v_m_0PI and v_m_PI4 to cells of the rotation mask m_rot
5: Convert v_m_0PI and v_m_PI4 to cells of the direct motion masks m_0PI and m_PI4
6: m_rot = m_rot $\cup$ m_0PI $\cup$ m_PI4

Robot is represented by the masks in the map. The mask is set of relative coordinates related to reference point. Two kind of the masks are used, direct motion mask and rotation mask. Angle of rotation of the mask is increased by $\pi/4$, because the surroundings of eight cells is used. Sixteen masks are used in all, eight direct motion masks and eight rotation masks. The mask is in the memory in optimized format to minimalize memory usage. They share common parts together. Three masks are saved really in the memory, alg. 1. The masks are created once during the path planning algorithm initialization. In the first step, vector patterns are created based on the real shape and dimensions and then are converted to bitmap mask based on the map resolution. Rotation mask is composed from two direct motion mask rotated by $\pi/4$ and four arc, which symbolize the rotation. Using of this mask assumes that obstacles have minimal diameter of 50 mm (cell 25 mm $\times$ 25 mm).

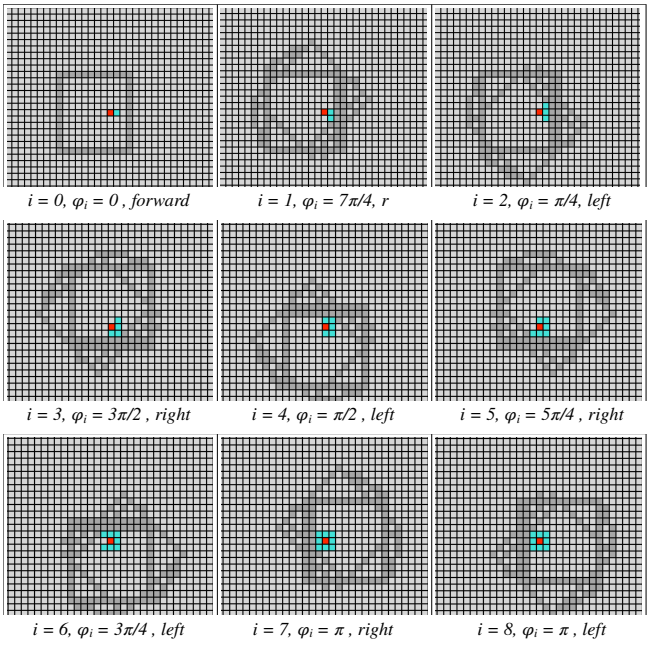

Fig. 2. Algorithm expands the cell (red squere) and check opened cells (turquoise squere), if they are available by the masks (dark grey square).

If the robot is represented by one cell in the map, the expansion is same for all cells. Every expanded cell is tested tested for an obstacle flag. If we use extended A* algorithm, the cells are expanded differently, depending on the angle of rotation and direction of movement. Mask and transformation are determined and then it is tested for an obstacle flag in alignment mask in the map.

If the algorithm expands cell in the map, angle of rotation is determined from the coordinates of n_{best} cell (the first node in priority list O, the best candidate to expansion) and currently being processed cell x_i. Direct motion mask m_0PI or m_PI4 is used based on the angle of rotation. Mask m_0PI is chosen for even multiples of $\pi/4$ and mask m_PI4 is chosen for odd multiples of $\pi/4$. The space in front of the robot is check by the direct motion mask in first step of the loop. It is tested, if cell from $N(n_{best})$ (set of the nodes, which neighbor with n_{best}) is available in the direction of movement.

Direct motion mask m_0PI (m_PI4) has basic orientation to the angle 0 ($\pi/4$). Relative coordinates of the mask is tranformed by the rotation matrix, which makes rotation of $\pi/2$, then the matrix elements have values of 0, ± 1. Combination of two masks m_0PI, m_PI4 and rotation matrix can be used for all eight directions.

The mask is set of integer coordinates related to the robot reference point (axis of rotation). Absolute coordinates of mask on the playground is determined during the test. Reference cell is x_i for m_0PI and m_PI4, n_{best} for m_rot. Obstacle flag is tested on these coordinates. If test is positive, position on this cell is declared as unavailable.

Algorithm 2 Extended A* algorithm, cells expansion and free cells checking

```
Create the masks and save it in to memory
...
repeat
    Choose n_best from O that f(n_best) ≤ f(n), ∀n ∈ O
    Remove n_best from O and put it in to C
    Go to end, if n_best = q_goal
    Determine robot angle of rotation φ_0 in n_best
    Choose direct motion mask m_0PI or m_PI4
    based on φ_0
    for i = 0 to 9 do
        Remove x_i from N(n_best), if it is not in C
        if x_i ∉ O then
            Transform choosen mask to required angle φ_i,
            i = 0 direct motion, (−1)^i < 0 rotation to the
            right, (−1)^i > 0 ratation to the left
            if M(x_i, φ_i) ∩ P = ∅ then
                Add x_i to O
            else
                Do not test alredy rotation to the left and to the
                right based on (−1)^i
            end if
        else if g(n_best) + c(n_best, x_i) < g(x_i) then
            Update backpointer of the node x_i to n_best
        end if
        Choose rotation mask m_rot for next loop step
    end for
until O is not empty
```

Rotation of the robot is represented by rotation mask mask_rot, which expresses rotation of $\pi/4$ (from O to $\pi/4$) and motion forward after the rotation. It is a complete assumed movement. Rotation to the right $\langle\varphi, \varphi - \pi\rangle$ and to the left $\langle\varphi, \varphi + \pi\rangle$ is tested alternately in the loop based on the i, fig. 2. Mask rotates by $\pi/4$, it is rotation to the neighboring cell in surroundings of eight cells. If any position during the rotation is not available, next position in this direction is not tested. The rotation mask is transformated to the required angle by rotation matrix as direct motion mask. Mask inverts around the x axis before transformation for odd multiples of $\pi/4$. One rotation mask and rotation matrix can be used for all eight directions.

3. Implementation and testing

Described algorithm is implemented as C library based on the simple A* algorithm. Path planner library is a part of the complex robot software created by Flamingos team. The robot software was build for two platforms, common PC like developer's notebook and industrial PC. We use PowerPC (400MHz, 128 MB RAM, 64 MB FLASH) industrial PC as main robot computer.

Algorithm was tested in the simulator on a PC and in the real robot. Parts of robot software comunicate together over the middleware ORTE (OCERA Real-Time Ethernet) [3], it allows very fast and flexible testing and debuging. Robot can move on the playgound and robot software runs on the PC, data from sensors and control signals are transmitted via WiFi over ORTE and display in simulator, especially information from the map. We need to see all information on the one place during the debugging of behavior of robot.

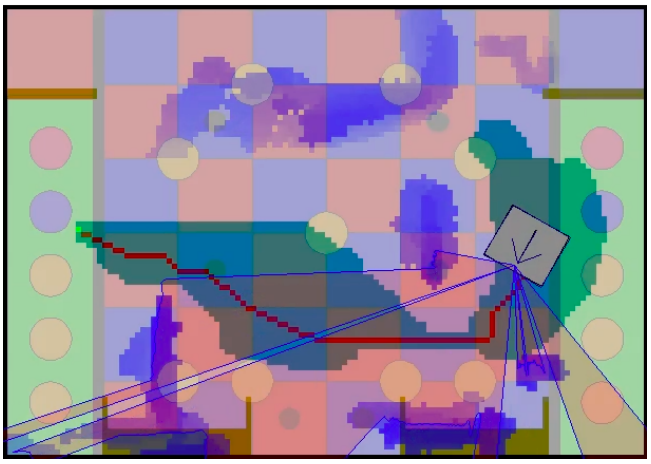

Fig. 3. Screenshot from the simulator, robot moves on the playground and software runs on PC. Red line shows the found path, dark turquoise color shows expanded cells, obstacles are blue.

Algorithm was tested in the real robot on the playground and robot software runs on the PowerPC industrial computer. Many obstacles on playground are complex space for testing. Robot draw all obstacles in the map during the movement based on data from the laser range finder Hokuyo. The obstacles can be moving and gradually forget in the map (shade of blue in the simulator), fig. 3.

The tests showed that the robot can move withnout problems in very limited space. Algorithm results and smooth movement affects resolution of the map (cell 25 mm $\times$ 25 mm) and forget time of the obstacles (5 sec.). If algorithm runs on PowerPC, the extended A* algorithm is ten times slower than the simple A* algorithm in the worst case. There needs to be done additional optimization.

4. Conclusion

Extended A* algorithm based on simple A* algorithm was described. Real shape, dimensions and angle of rotation of the robot are represented by bitmap mask, which is used for checking for free space in the surroundigs of the robot. Extended A* algorithm was implemented in software for robot and tested in the PC simulator and in the real robot. Tests showed that the robot can move reliably in limited space on the playground. Further work is to optimize the computational complexity on the main robot computer. We should consider to change the principles of robot software and the algorithm for more complex solution.

Acknowledgements

The work described in the paper was supervised by Ing. Michal Sojka, Ph.D., Department of Control Engineering, FEE CTU in Prague and was created within project Eurobot student team Flamingos. This project was supported by Ministry of Education of the Czech Republic under project 1M0567 (CAK). The work has been supported by the grant No. SGS12/143/OHK3/2T/13 of the Czech Technical University in Prague and also has been supported by the research program No. MSM 6840770012 of the Czech Technical University in Prague (sponsored by the Ministry of Education, Youth and Sports of the Czech Republic).

References

[1] The Eurobot competition website, `http://www.eurobot.org/eng/`, March 2012.

[2] The Flamingos robotic team website, `http://flamingos.felk.cvut.cz/`, March 2012.

[3] The ORTE project website, `http://www.ocera.org/download/components/WP7/orte-0.3.1.html`, March 2012.

[4] Simon X. Yang and Max Q.H. Meng. Real-Time Collision-Free Motion Planning of Mobile Robot Using a Neural Dynamics-Based Approach. In *IEEE Transactions On Neural Networks*, 2003, vol. 14, no. 6, p. 1541 – 1552.

[5] C. H. Xuefu Xiang, Jiye Zhang. An efficient system for nonholonomic mobile robot-path planning. In *Proceedings on Intelligent Systems and Knowledge Engineering ISKE2007*, 2007.

[6] K. O. Arras, J. Persson, N. Tomatis, and R. Siegwart. Real-time obstacle avoidance for polygonal robots with a reduced dynamic window. In *Proceedings of the 2002 IEEE International Conference on Robotics Automation*. Washington, DC, 2007, p. 3050 – 3055.

[7] Y. K. Hiroaki Seki and M. Hikizu. Real-time obstacle avoidance using potencial field for a nonholonomic vehicle. In *J. Silvestre-Blanes, editor, Factory Automation*. 2010, p. 523-542.

[8] POKORNÝ, M. Collision-free Trajectory Planning for Mobile Robots. Diploma thesis, FEE CTU in Prague, 2012.

About Authors...

Matouš POKORNÝ

was born 5.1.1986 in Prague. He graduated Master degree in Sensors and Instrumentation at FEE CTU in Prague. He worked on thesis Collision-free Trajectory Planning for Mobile Robots. He is member of the university robotic team Flamingos, which participates in the Eurobot competition since 2010.