THE BULLETIN OF OKAYAMA UNIVERSITY OF SCIENCE No. 35 A pp. 207-213 (1999)

Path Planning for Car-like Robots

Jianli YU*, Valeri KROUMOV** and Hiroyuki NARIHISA***

*Graduate School of Engineering
**Department of Electronic Engineering
***Department of Information & Computer Engineering
Okayama University of Science
1-1 Ridai-cho, Okayama 700-0005, Japan

(Received November 4, 1999)

The problem of path planning is studied for the case for a mobile car-like robot moving in an environment filled with obstacles which shapes and positions are known. An algorithm based on a description of the obstacles using a neural network is proposed, which allows to construct an almost optimal path. The path is piecewise linear with changing directions at the corners of the obstacles. The proposed algorithm is a significant improvement of the potential field algorithms because it finds an optimal path without being trapped in local minimums and the calculation speed for the proposed algorithm is comparatively fast. The algorithm solves the navigation problem in very complex environments such as polygonal mazes. Simulation results show the effectiveness of the proposed algorithm.

1 Introduction

In this paper we propose an algorithm for solving the path planning-problem for a polygonal robot in a two-dimensional known environment, where the obstacles are stationary polygons. The proposed algorithm is in general based on the potential field methods. It is well known that the strength of these methods is that, with some limited engineering, it is possible to construct quite efficient and relatively reliable motion planners [1]. But the potential field methods are usually incomplete and may fail to find a free path, even if one exists, because they can get trapped in a local minimum [2, 3, 4]. Another problem with the potential field methods is that they are not so much suitable to generate optimal path.

Path planning for car-like robots is a fundamental issue in the field. In the recent years, the navigation problem in known [1, 4] and unknown [5, 6, 7, 8] environments was often addressed in literature. The purpose of the path planner is to compute a path from the start position of the vehicle to the goal to be reached. The primary concern of path planning is to compute *collision-free* paths. Another important issue is to compute the *optimal path* bringing the vehicle to the final position.

Lozano-Pèrez and Wesley [9] have proposed an algorithm (called VGRAPH algorithm) close to the optimization approach, in which the path planning is accomplished by finding a path through a graph connecting vertices of the forbidden regions (obstacles). Their algorithm is not based on the potential field methods but we refer it because it is concerning the optimization problems. Hocaoğlu and Sanderson [10] have proposed an evolutionary path planner for multidimensional paths, which is much more effective than several existing algorithms. The drawback in above works is that the description of the all possible paths is quite complicated. Because of this, even when the number of the obstacles increases only by one, the rearrangement of the description becomes quite troublesome and the computational cost seems to be comparatively high.

The proposed in this paper algorithm is a significant improvement of the proposed one by Sun *et al*[4]. It solves the local minimum problems and generates optimal path in relatively small number of iterations. The assumptions made in this work are that there is finite number of stationary polygonal obstacles with finite number of vertices, and that the robot polygon also has a finite number of vertices. In order to reduce the problem of path planning to that of navigating a point, the obstacles are enlarged by the robot's polygon dimensions to yield a new set of polygonal obstacles. This "enlargement" of the obstacles is a well-known method introduces formally

by Lozano-Pèrez and Wesley [9].

The paper is organized as follows. In the next section we give a definition of the map representation. In Section 2 we describe the theoretical background for development of the algorithm. Section 3 is the central part of this paper and describes the proposed algorithm for a path planner. In Section 4 we show the effectiveness of the proposed algorithm by presenting two simulation results. In the final section we are discussing the results and some plans for future developments.

2 Preliminaries

In this section some preliminary results about the description of the obstacles map and the calculation of optimal path are given[4]. The section consists of two subsections. First we show how the obstacles are described and what kind of potential function is chosen. Next, the path planner description is presented briefly.

2.1 Obstacles description

Every obstacle is described by a neural network as shown in Figure 1. The inputs of the network are the coordinates of a point of a path. The output neuron is described by the following expression, which is called *a repulsive penalty function (RPF)* and has a role of repulsive potential.

$$C = f(I_O) \tag{1}$$

where I_O is the output of the hidden layer and takes a role of the induced local field of the neuron function $f(\cdot)$.

The output neuron input is:

$$I_O = \sum_{m=1}^{M} O_{H_m} + \theta_T \tag{2}$$

where θ_T is a bias, equal to the negative number of the vertices of an obstacle decreased by 0.5. This choice makes the value of I_O become less than 0.5, which assures small pseudotemperature for the output neuron function. O_{H_m} in equation (2) is the output of the m-th neuron of the middle layer and

$$O_{H_m} = f(I_{H_m}), \qquad m = 1, \ldots, M \tag{3}$$

where I_{H_m} is the input of the m-th neuron of the middle layer and has a role of induced local field of the neuron function. The neuron function $f(\cdot)$ has the form:

$$f(x) = \frac{1}{1 + e^{-x/T}} \tag{4}$$

where T is the pseudotemperature and the induced local field of the neuron x is equal to I_O for equation (1) or equal to I_{H_m} in the case of equation (3). The pseudotemperature is given as:

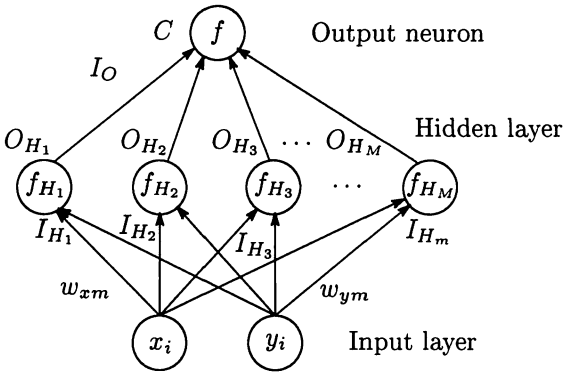

Fig. 1: Obstacle description network

$$T(t) = \frac{\beta_0}{\log(1+t)} \tag{5}$$

Finally, I_{H_m} is given by the activating function

$$I_{H_m} = w_{xm} x_i + w_{ym} y_i + \theta_{H_m} \tag{6}$$

where x_i and y_i are the coordinates of i-th point of the path, w_{xm} and w_{ym} are weights, and θ_{H_m} is a bias, which is equal to the x or y coordinates of the vertices of an obstacle. In other words the number of the neurons in the hidden layer is equal to the number of the vertices of an obstacle, and every obstacle is described with a network as in Figure 1.

Example 1: Figure 2 shows a description of rectangle obstacle. The vertices have coordinates (0.2, 0.2), (0.8, 0.2), (0.8, 0.7), and (0.2, 0.7). The obstacle is described by the following equations.

$$x - 0.2 > 0 \qquad -x + 0.8 > 0$$
$$y - 0.2 > 0 \qquad -y + 0.7 > 0$$

All the weights are chosen as shown in the figure.

2.2 Path planning

The state of the path is described by the following energy function.

$$E = w_l E_l + w_c E_c \tag{7}$$

where w_l and w_c are weights ($w_l + w_c = 1$), E_l depicts the squared length of the path

$$E_l = \sum_{i=1}^{N-1} L_i^2 = \sum_{i=1}^{N-1} ((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2), \tag{8}$$

and E_c is given by the expression.

$$E_c = \sum_{i=1}^{N} \sum_{k=1}^{K} C_i^k \tag{9}$$

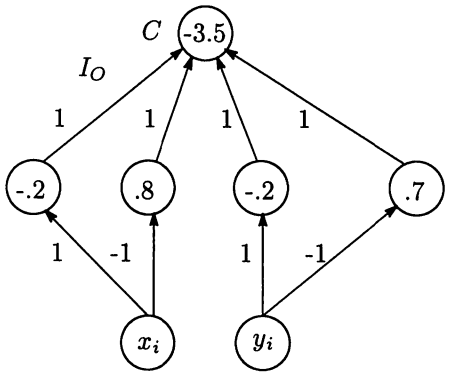

Fig. 2: Description network for a rectangle

where N is the number of the points between the start and goal and K is the number of the obstacles.

The idea is to minimize equation (7) which means to obtain an optimal in length path, which does not collide with any of the obstacles. In order to minimize (7) the classical function analysis methods are applied. First, we find the time derivative of E:

$$\frac{dE}{dt} = \sum_{i=1}^{N}(w_l(\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i}) + w_c\sum_{k=1}^{K}\frac{\partial C_i^k}{\partial x_i})\dot{x}_i + (w_l(\frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i}) + w_c\sum_{k=1}^{K}\frac{\partial C_i^k}{\partial y_i})\dot{y}_i) \quad (10)$$

Letting

$$\dot{x}_i = -\eta(w_l(\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i}) + w_c\sum_{k=1}^{K}\frac{\partial C_i^k}{\partial x_i})$$

$$\dot{y}_i = -\eta(w_l(\frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i}) + w_c\sum_{k=1}^{K}\frac{\partial C_i^k}{\partial y_i}) \quad (11)$$

we can redefine the minimization problem as

$$\frac{dE}{dt} = -\frac{1}{\eta}\sum_{i=1}^{N}(\dot{x}_i^2 + \dot{y}_i^2) < 0 \quad (12)$$

where η is a scaling factor. Now, from equations (11),

$$\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i} = -2x_{i+1} + 4x_i - 2x_{i-1}$$

$$\frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i} = -2y_{i+1} + 4y_i - 2y_{i-1} \quad (13)$$

and

$$\frac{\partial C_i^k}{\partial x_i} = \frac{\partial C_i^k}{\partial (I_O)_i^k}\frac{\partial (I_O)_i^k}{\partial x_i}$$

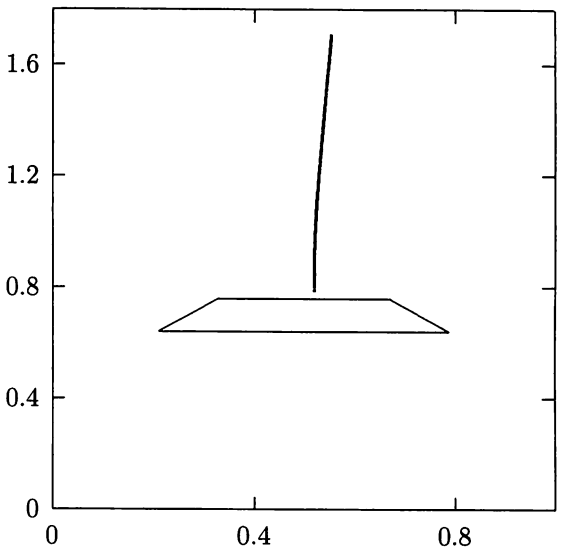

Fig. 3: Trapezoidal obstacle (Example 2): the pseudotemperatures are equal to each other, i. e. $\beta_m = 0.5$

$$= \frac{\partial C_i^k}{\partial (I_O)_i^k}(\sum_{m=1}^{M}\frac{\partial (O_{H_m})_i^k}{\partial (I_{H_m})_i^k}\frac{\partial (I_{H_m})_i^k}{\partial x_i})$$

$$= f'((I_O)_i^k)(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k)w_{xm}^k)$$

$$\frac{\partial C_i^k}{\partial y_i} = \frac{\partial C_i^k}{\partial (I_O)_i^k}\frac{\partial (I_O)_i^k}{\partial y_i}$$

$$= \frac{\partial C_i^k}{\partial (I_O)_i^k}(\sum_{m=1}^{M}\frac{\partial (O_{H_m})_i^k}{\partial (I_{H_m})_i^k}\frac{\partial (I_{H_m})_i^k}{\partial x_i})$$

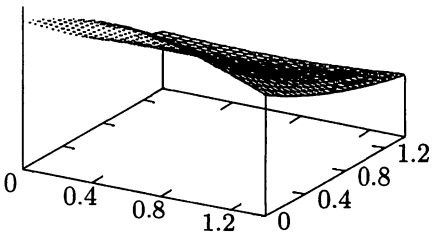

Fig. 4: Form of the RPF for Example 2

$$= f'((I_O)_i^k)(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k)w_{ym}^k). \quad (14)$$

This leads to the final form of the function:

$$\begin{aligned}
\dot{x}_i &= -\eta(2w_l(2x_i - x_{i-1} - x_{i+1}) \\
&+w_c\sum_{k=1}^{K} f'((I_O)_i^k)(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k)w_{xm}^k)) \\
\dot{y}_i &= -\eta(2w_l(2y_i - y_{i-1} - y_{i+1}) \\
&+w_c\sum_{k=1}^{K} f'((I_O)_i^k)(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k)w_{ym}^k)) \quad (15)
\end{aligned}$$

where f' is given by the following expressions

$$\begin{aligned}
f'(\cdot) &= \frac{1}{T}f(\cdot)(1 - f(\cdot)) \\
f'_{H_m}(\cdot) &= \frac{1}{T}f_{H_m}(\cdot)(1 - f_{H_m}(\cdot)) \quad (16)
\end{aligned}$$

In equations (15) the first member in the rights side is for the path length optimization and the second one is for the obstacle avoidance.

2.3 Discussion

One of the important advantages of above algorithm is that it allows parallelism in the calculations of the neural network outputs, which leads to increasing the speed of the calculations. Unfortunately, as a common problem of the potential field algorithms, some difficulties arise in adjustment of the pseudotemperatures in the equation (4). As it is shown in the examples below, in some cases the algorithm cannot produce optimal in length path and even can generate paths which might be unrealizable.

Example 2: A typical example is when the shape of the obstacle is not a regular polygon. One example is the trapezoidal shape as shown in figure 3. In this case the penalty function changes as shown in figure 4 and as a result of this it appears that, from the point of view of the algorithm, the path is the shortest one and does not collide with the obstacle (figure 3). In this case there are no points lying inside the obstacle, but in fact the path does not avoid the obstacle. This is because the RPF increases faster for long edges, and forms a maximum at the longest side. After decreasing the value of the pseudotemperature for the long edge a perfect avoidance is done (see figures 6 and 5). As shown in figure 5, after the decreasing the initial value of the pseudotemperature for the long edge, the maximum of the RPF is moved inside the surface of the RPF and the repulsion of the points of the path becomes as in figure 6.

Example 3: Another problem, which arises when the above algorithm is used, is that in many

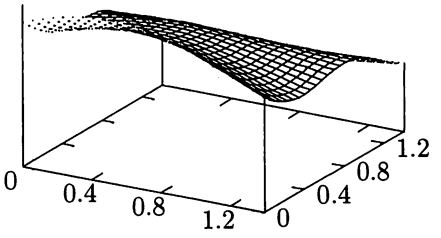

Fig. 5: The growth of the RPF with modified pseudotemperatures for Example 2

cases the calculation may not give an optimal path. A typical example is shown in figure 7. This usually happens because even when the path does not collide with any obstacle the minimization procedure for the repulsive part of equation (15) continues. Finally, this might lead to producing of trajectories, which are sometimes almost unusable or even unrealizable. In addition, it often happens that the condition for finishing of the calculations might not be satisfied easily, because of which the number of the iterations increases drastically. Actually, after all the points of the path leave the obstacles, there is no need to continue the calculations for the repulsion of the points. This is shown in the next section.

3 Algorithm

In this section a modification of the algorithm shown in Section 2 is proposed. The calculations for the path are conceptually composed by the following steps:

1. Initial step

   (a) Let the start position of the robot is (x_0, y_0), and the goal position is denoted as (x_{N-1}, y_{N-1}).

   (b) At $t = 0$ the coordinates of the points of the initial path (straight line) $(x_i, y_i;\ i = 1, 2, \ldots, N-2)$ are assigned as

$$\begin{aligned}
x_i &= x_0 + i(x_{N-1} - x_0)/(N-1) \\
y_i &= (y_{N-1} - y_0)(x_i - x_0)/(x_{N-1} - x_0) + y_0 \quad (17)
\end{aligned}$$

   i. e. the distance between the points is equal.

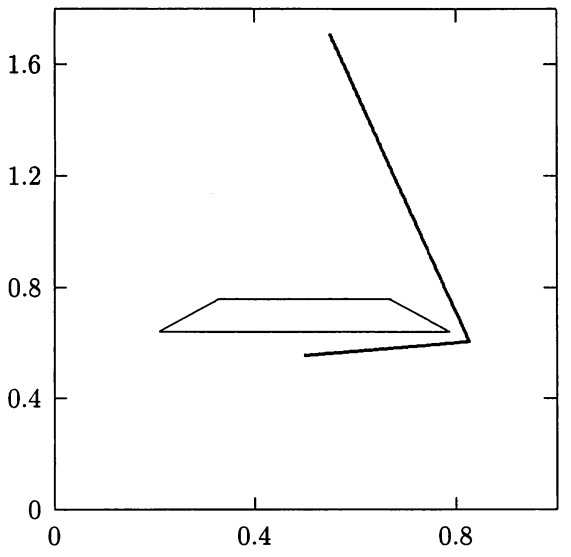

Fig. 6: Perfect obstacle avoidance for Example 2: the initial pseudotemperatures are modified as $\beta_1 = 0.5$, $\beta_2 = 0.5$, $\beta_3 = 0.2$, $\beta_4 = 0.5$

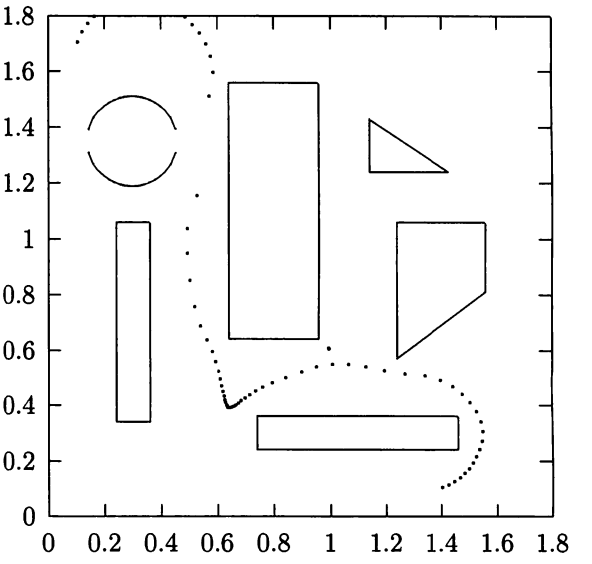

Fig. 7: Example 3: The path is not optimal

The obstacles here are enlarged by the robot's polygon dimensions[9].

2. For the points (x_i, y_i) of the path which lie inside some obstacle, the iterations continue according to the following equations:

$$\begin{aligned}
\dot{x}_i &= -\eta_1(2w_l(2x_i - x_{i-1} - x_{i+1}) \\
&+w_c \sum_{k=1}^{K} f'((I_O)_i^k)(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{xm}^k)) \\
\dot{y}_i &= -\eta_1(2w_l(2y_i - y_{i-1} - y_{i+1}) \\
&+w_c \sum_{k=1}^{K} f'((I_O)_i^k)(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{ym}^k)), \quad (18) \\
i &= 1, 2, \ldots, N-2
\end{aligned}$$

For the points (x_i, y_i) situated outside the obstacles, then instead of equations (18) use the following equations:

$$\begin{aligned}
\dot{x}_i &= -\eta_2 w_l(2x_i - x_{i-1} - x_{i+1}) \\
\dot{y}_i &= -\eta_2 w_l(2y_i - y_{i-1} - y_{i+1}) \quad (19)
\end{aligned}$$

i.e. for the points of the path lying outside obstacles, we continue the calculation with the goal to minimize only the length of the path.

Here equations (18) are almost the same as equations (15) with the difference that instead of only one RPF, different functions, as explained below, are used for every layer of the neural network. In equations (18), η_1 is a small enough so that the local minimums be continuously searched, while η_2 is chosen much bigger to allow decreasing of the number of steps for reaching shorter path.

3. Perform p times the calculations of step 2, i.e. find $x_i(t+p)$, $y_i(t+p)$ $(i = 1, 2, \ldots, N-2)$, where p is any suitable number, say $p = 100$.

4. Test for convergence

   Calculate the distance between the points $(x_i(t), y_i(t))$, and the points $(x_i(t+p), y_i(t+p))$ $(i = 1, 2, \ldots, N-2)$, i. e.

$$\begin{aligned}
d &= \sum_{i=1}^{i=N-2} ((x_i(t+p) - x_i(t))^2 \\
&+(y_i(t+p) - y_i(t))^2)^{1/2} \quad (20)
\end{aligned}$$

   - If $d < \varepsilon$ then the algorithm terminates with the conclusion that the goal is reached via an optimal path.
   - If $d \geq \varepsilon$, then GO TO step 2

   where ε is a small constant, say $\varepsilon = 0.1$.

Here again every obstacle is described using a neural network as shown in Figure 1, but we define different pseudotemperatures for every layer. The

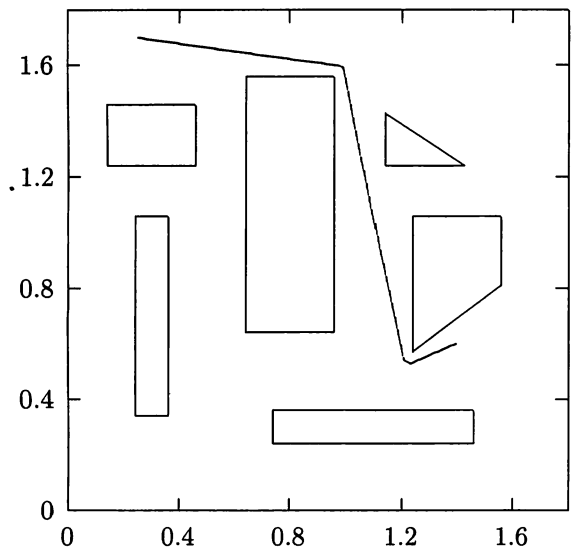

Fig. 8: Simulation 1: $\beta_0 = 0.4$, $d = 2.03331$, w_c=0.5,w_l=0.5,η_1=0.1, η_2=30, 3000 iterations, $x(0) = 0.25, y(0) = 1.7, x(N-1) = 1.4, y(N-1) = 0.6$

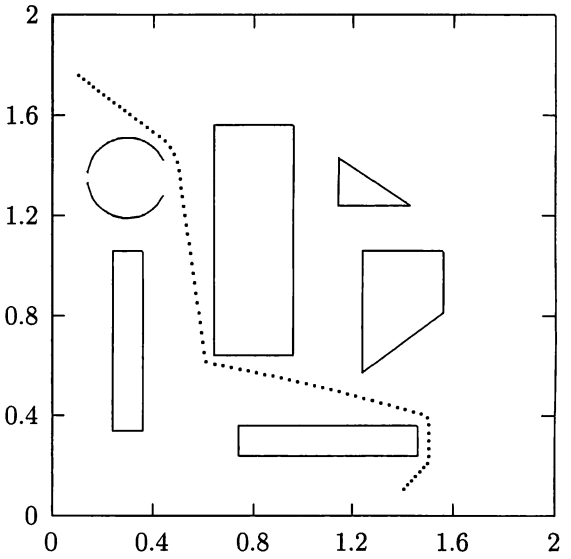

Fig. 9: Simulation 2: $\beta_0 = 0.4$, $d = 2.5$, $w_c = 0.5$, $w_l = 0.5$, $\eta_1 = 0.1$, $\eta_2 = 30$, 2850 iterations, $x(0) = 0.25$, $y(0) = 1.75$, $x(N-1) = 1.4$, $y(N-1) = 0.1$

output neuron is described again by equation (1), the neuron function $f(\cdot)$ is same as equation (4) and the pseudotemperature is as in equation (5). The hidden layer's inputs are as in equation (2) but the outputs O_{H_m} now become

$$O_{H_m} = f_{H_m}(I_{H_m}), \quad m = 1, \ldots, M \tag{21}$$

with I_{H_m} becoming the induced local field of the neuron function f_{H_m}:

$$f_{H_m}(I_{H_m}) = \frac{1}{1 + e^{-I_{H_m}/T_{H_m}(t)}} \tag{22}$$

and

$$T_{H_m}(t) = \frac{\beta_m}{\log(1+t)} \tag{23}$$

Finally, I_{H_m} is given by the activating function (6). Note, that the equations (4) and (22) have different pseudotemperatures which is one of the important conditions for convergence of the algorithm and generation of almost optimal path.

4 Simulation results

To show the effectiveness of the proposed in this paper algorithm, two simulation examples are given in this section. Figures 8 and 9 show the results of simulation of path planning in a complicated environment. The obstacles are shown in their original shapes, i.e. without "enlargement". In these simulations the number of the points between the start position and the goal was set to 80 (comparatively small number of points). In the both examples the values of the coefficients in equations of the algorithm were not changed and as shown in the figures the number of the iterations is small (approximately 3000 iterations). In fact, the authors have performed several simulations using different environments, in which the obstacles were situated in many different ways. In all the simulations the paths were successfully generated and the time for the calculations were much smaller compared to these for the original algorithm. Figure 10 shows the final shape of the RPF for the simulation environment.

Compared to the original algorithm the proposed here is not so much sensitive to the values of the coefficients. It was confirmed by simulations that the coefficients should be chosen as follows. η_1 can take any value between 0.1 and 0.3. η_2 may have any value between 0.1 and 30, and as big η_2 is the shortest length of the path is reached faster. β_m may be successfully between 0.1 and 0.5.

It becomes clear from the above simulations that the practicability of the proposed algorithm depends on the development of some mechanism for choosing the initial values of the pseudotemperatures β_m. This is the subject of research now in progress.

5 Conclusions

In this paper we have proposed an algorithm, which guarantees a planning of near optimal in length path for car-like robots moving in *a priori* known environment. The proposed algorithm concerns only two-dimensional world, which is comparatively strong restriction. However, the description of the obstacles which we apply, allows easily extending the algorithm for three-dimensional environment. This will allow to construct more realistic paths.

The proposed algorithm is a significant improvement of the potential field algorithms because it finds an optimal path without being trapped in local minimums and the calculation speed for the proposed algorithm is comparatively fast. The algorithm solves the navigation problem in very complex environments such as polygonal mazes. This is because the only assumptions made are that the obstacles are stationary polygons with a finite number of vertices. The calculations for the optimization of the path and these for the obstacle avoidance are separated. Because of this, when there is no need to search for a short path the iterations can be stopped after the avoidance is completed.

There are several problems left to be solved. For example, when in the initial time, the start and the goal points lie exactly on a line which coincide with the axis of symmetry of an obstacle, the path collide with the obstacle. This is a common problem for most of the algorithms, based in potential field and there is a need to take special measures against such cases.

We have assumed throughout this paper that the robot performs exact motion. This assumption is not practical because the phenomenon of wheel slippage, which creates errors between the planned and the actual path executed by the robot. This assumption may be omitted if a feedback for comparison the actual position of the robot and calculated one is introduced and if a difference occurs a respective correction in the path be included. We are in process of experimental realization of the proposed algorithm using a "Khepera" car-like robot by which we are planning to solve the problems arising when not exact motion is performed.

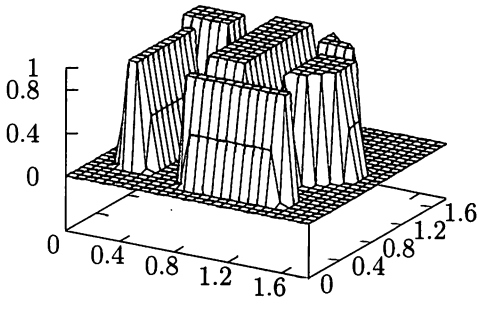

Fig. 10: Shape of the RPF for the simulation environment

References

[1] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[2] P. Khosla and R. Volpe, "Superquadratic Artificial Potentials for Obstacle Avoidance and Approach", In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1778–1784, 1988.

[3] E. Rimon and D. E. Koditschek, "Exact Robot Navigation Using Artificial Potential Fields, *IEEE Transactions in Robotics and Automation*, Vol. 8, No. 5, pp. 501–518, October 1992.

[4] Z. Q. Sun, Z. X. Zhang, and Z. D. Deng, *Intelligent Control Systems*, Tsinghua University Press, pp. 227–237, 1997.

[5] G. Foux, M. Heymann, and A. Bruckstein, "Two – Dimensional Robot Navigation Among Unknown Stationary Polygonal Obstacles", *IEEE Transactions on Robotics and Automation*, Vol.9, No.1, pp.96-102, February 1993.

[6] X. D. Yang, M. Yamamoto and A. Mohri, "Path Planning for Mobile Robot in Uncertain Workspace Using Obstacle Topology", *Journal of the Robotics Society of Japan*, Vol.13, No.8, pp.1130-1137, August 1995.

[7] K. Nagatani and S. Yuta, "Path and Sensing Point Planning for Mobile Robot Navigation to Minimize the Risk of Collision", *Journal of the Robotics Society of Japan*, Vol.15, No.2, pp.197-206, February 1997.

[8] B. Kreczmer, "Path Planning System for Car-like Robot", In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium, pp. 40–45, May 1998.

[9] T. Lozano-Pèrez and M. A. Wesley, "An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles", *Comm of the ICM*, Vol. 22, no. 10, pp. 560–570, 1979.

[10] C. Hocaoğlu and A. C. Sanderson, "Evolutionary Path Planning using Multiresolution Path Representation", In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium, pp. 318–323, May 1998.