9

Neural Networks Based Path Planning and Navigation of Mobile Robots

Valeri Kroumov[1] and Jianli Yu[2]
[1]*Department of Electrical & Electronic Engineering, Okayama University of Science, Okayama*
[2]*Department of Electronics and Information, Zhongyuan University of Technology, Zhengzhou*
[1]*Japan*
[2]*China*

1. Introduction

The path planning for mobile robots is a fundamental issue in the field of unmanned vehicles control. 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, equally important issue is to compute a *realizable* and, if possible, *optimal path*, bringing the vehicle to the final position.

Although humans have the superb capability to plan motions of their body and limbs effortlessly, the motion planning turns out to be a very complex problem. The best known algorithm has a complexity that is exponential to the number of degrees of freedom and polynomial in the geometric complexities of the robot and the obstacles in the environment (Chen & Hwang (1998)). Even for motion planning problems in the 2-dimensional space, existing complete algorithms that guarantee a solution often need large amount of memory and in some cases may take long computational time. On the other hand, fast heuristic algorithms may fail to find a solution even if it exists (Hwang & Ahuja (1992)).

In this paper we present a fast algorithm for solving the path planning problem for differential drive (holonomic)[1] robots. The algorithm can be applied to free-flying and snake type robots, too. Generally, we treat the two-dimensional known environment, where the obstacles are stationary polygons or ovals, but the algorithm can easily be extended for the three-dimensional case (Kroumov et al. (2010)). The proposed algorithm is, in general, based on the potential field methods. The algorithm solves the local minimum problem and generates optimal path in a relatively small number of calculations.

The paper is organized as follows. Previous work is presented in Section 2. In Section 3 we give a definition of the map representation and how it is used to describe various obstacles situated in the working environment. The path and the obstacle collisions are detected using artificial annealing algorithm. Also, the solution of the local minima problem is described there. In Section 4 we describe the theoretical background and the development of a motion

[1] In the mobile robotics, the term *holonomic* refers to the kinematic constraints of the robot chassis. The holonomic robot has zero nonholonomic kinematic constraints, while the nonholonomic one has one or more nonholonomic kinematic constraints.

planner for a differential drive vehicle. Section 5 presents the algorithm of the path planner. In Section 6 the calculation time is discussed and the effectiveness of the proposed algorithm is proven by presenting several simulation results. In the last section discussions, conclusions, and plans for further developments are presented.

2. Previous work

Comprehensive reviews on the work on the motion planning can be found in Latombe (1991). In this section we concentrate on motion planning for moving car-like robots in a known environment.
Motion planners can be classified in general as:

1) complete;

2) heuristic.

Complete motion planners can potentially require long computation times but they can either find a solution if there is one, or prove that there is none. Heuristic motion planners are fast but they often fail to find a solution even if it exists.
To date motion planners can be classified in four categories (Latombe (1991)):

1) skeleton;

2) cell decomposition;

3) subgoal graph;

4) potential field.

In the skeleton approach the free space is represented by a network of one-dimensional paths called a *skeleton* and the solution is found by first moving the robot onto a point on the skeleton from the start configuration and from the goal, and by connecting the two points via paths on the skeleton. The approach is intuitive for two-dimensional problems, but becomes harder to implement for higher degrees of freedom problems.
Algorithms based on the visibility graph (VGRAPH) (Lozano-Pèrez & Wesley (1979)), the Voronoi diagram (O'Dúnlaing & Yap (1982)), and the *silhouette* (Canny (1988)) (projection of obstacle boundaries) are examples of the skeleton approach. In the VGRAPH algorithm the path planning is accomplished by finding a path through a graph connecting vertices of the forbidden regions (obstacles) and the generated path is near optimal. The drawback of the VGRAPH algorithm is that the description of all the possible paths is quite complicated. Actually, the number of the edges of the VGRAPH is proportional to the squared total number of the obstacles vertices and when this number increases, the calculation time becomes longer. Another drawback is that the algorithm deals only with polygonal objects. Yamamoto et al. (1998) have proposed a near-time-optimal trajectory planning for car-like robots, where the connectivity graph is generated in a fashion very similar to that of Lozano-Pèrez & Wesley (1979). They have proposed optimization of the speed of the robot for the generated trajectory, but the optimization of the trajectory itself is not enough treated.
In the cell decomposition approach (Paden et al. (1989)) the free space is represented as a union of cells, and a sequence of cells comprises a solution path. For efficiency, hierarchical trees, e.g. octree, are often used. The path planning algorithm proposed by Zelinsky (1992) is quite reliable and combines some advantages of the above algorithms. It makes use of quadtree data structure to model the environment and uses the distance transform methodology to generate paths. The obstacles are polygonal-shaped which yields a quadtree

with minimum sized leaf quadrants along the edges of the polygon, but the quadtree is large and the number of leaves in the tree is proportional to the polygon's perimeter and this makes it memory consuming.

In the subgoal-graph approach, subgoals represent *key* configurations expected to be useful for finding collision-free paths. A graph of subgoals is generated and maintained by a global planner, and a simple local planner is used to determine the reachability among subgoals. This two level planning approach is first reported by Faverjon & Tournassoud (1987) and has turned out to be one of the most effective path planning methods.

The potential field methods and their applications to path planning for autonomous mobile robots have been extensively studied in the past two decades (Khatib (1986), Warren (1989), Rimon & Doditschek (1992), Hwang & Ahuja (1992), Lee & Kardaras (1997a), Lee & Kardaras (1997b), Chen & Hwang (1998), Ge & Cui (2000), Yu et al. (2002), Kroumov et al. (2004)). The basic concept of the potential field methods is to fill the workspace with an artificial potential field in which the robot is attracted to the goal position being at the same time repulsed away from the obstacles (Latombe (1991)). It is well known that the strength of potential field methods is that, with some limited engineering, it is possible to construct quite efficient and relatively reliable motion planners (Latombe (1991)). 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 (Khosla & Volpe (1988); Rimon & Doditschek (1992); Sun et al. (1997)). Another problem with the existing potential field methods is that they are not so suitable to generate optimal path: adding optimization elements in the algorithm, usually, makes it quite costly from computational point of view (Zelinsky (1992)). Ideally, the potential field should have the following properties (Khosla & Volpe (1988)):

1. The magnitude of the potential field should be unbounded near obstacle boundaries and should decrease with range.
2. The potential should have a spherical symmetry far away from the obstacle.
3. The equipotential surface near an obstacle should have a shape similar to that of the obstacle surface.
4. The potential, its gradient and their effects on the path must be spatially continuous.

The proposed in this paper algorithm is partially inspired by the results presented by Sun et al. (1997) and by Lee & Kardaras (1997a), but our planner has several advantages:

- the obstacle descriptions are implemented directly in the simulated annealing neural network—there is no need to use approximations by nonlinear equations (Lee & Kardaras (1997b));
- there is no need to perform a learning of the workspace *off-line* using backpropagation (Tsankova (2010));
- there is no need to perform additional calculations and processing when an obstacle is added or removed;
- a simple solution of the local minimum problem is developed and the generated paths are conditionally optimized in the sense that they are piecewise linear with directions changing at the corners of the obstacles.

The algorithm allows parallel calculation (Sun et al. (1997)) which improves the computational time. The experimental results show that the calculation time of the presented algorithm depends linearly on the total number of obstacles' vertices—a feature

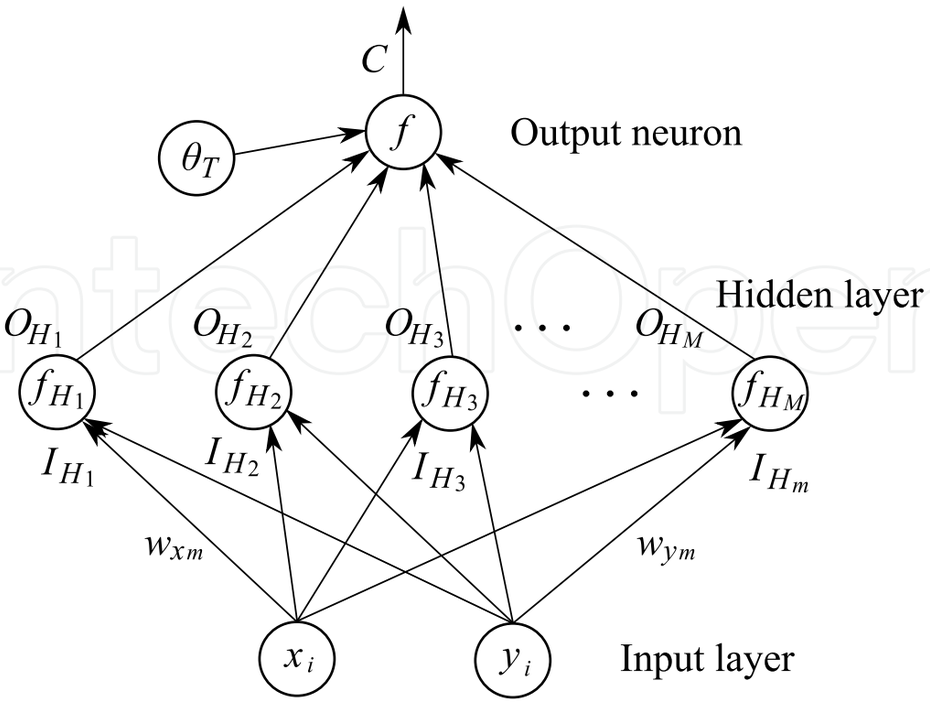


Fig. 1. Obstacle description neural network

which places it among the fastest ones. The only assumptions made in this work are that there are a finite number of stationary oval or polygonal obstacles with a finite number of vertices, and that the robot has a cylindrical shape. The obstacles can be any combination of polygons and ovals, as well. 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 introduced formally by Lozano-Pèrez & Wesley (1979).

3. Environment description

3.1 Obstacles

Every obstacle is described by a neural network as shown in Fig. 1. The inputs of the networks are the coordinates of the points of the 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:

$$C = f(I_0) = f\left(\sum_{m=1}^{M} O_{H_m} - \theta_T\right), \tag{1}$$

where I_0 takes a role of the induced local field of the neuron function $f(\cdot)$, θ_T is a bias, equal to the number of the vertices of the obstacle decreased by 0.5. The number of the neurons in the hidden layer is equal to the number of the vertices of the obstacle. O_{H_m} in eq. (1) is the output of the m-th neuron of the middle layer:

$$O_{H_m} = f(I_{H_m}), \qquad m = 1, \ldots, M, \tag{2}$$

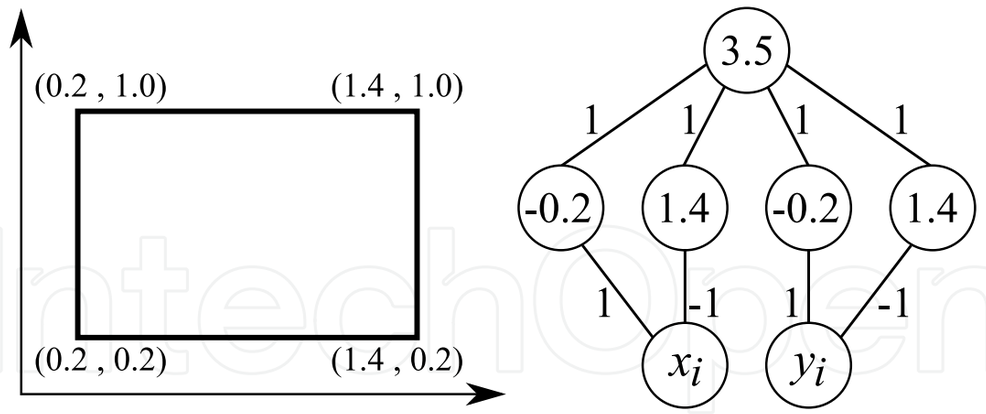


Fig. 2. The annealing network for a rectangular obstacle

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

$$f(x) = \frac{1}{1 + e^{-x/T}}, \tag{3}$$

where T is the pseudotemperature and the induced local field (x) of the neuron is equal to I_0 for eq. (1) or equal to I_{H_m} in the case of eq. (2). The pseudotemperature decreasing is given by:

$$T(t) = \frac{\beta_0}{\log(1+t)}. \tag{4}$$

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

$$I_{H_m} = w_{xm}x_i + w_{ym}y_i + \theta_{H_m}, \tag{5}$$

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 free element in the equation expressing the shape of the obstacle.
The following examples show descriptions of simple objects.

Example: 1. *Description of polygonal obstacle (see Fig. 2).*

The area inside the obstacle can be described with the following equations:

$$\begin{aligned} x - 0.2 > 0; \qquad & -x + 1.4 > 0 \\ y - 0.2 > 0; \qquad & -y + 1.0 > 0. \end{aligned}$$

From these equations and eq. (5):

$$\begin{aligned} w_{x1} &= 1 \quad w_{y1} = 0 \quad \theta_{H_1} = -0.2 \\ w_{x2} &= -1 \; w_{y2} = 0 \quad \theta_{H_2} = 1.4 \\ w_{x3} &= 0 \quad w_{y3} = 1 \quad \theta_{H_3} = -0.2 \\ w_{x4} &= 0 \quad w_{y4} = -1 \; \theta_{H_4} = 1 \end{aligned}$$

i.e. in the middle layer of the network, the activating functions become

$$\begin{aligned} I_{H_1} &= w_{x1}x_i + w_{y1}y_i + \theta_{H_1} = x_i - 0.2 \\ I_{H_2} &= w_{x2}x_i + w_{y2}y_i + \theta_{H_2} = -x_i + 1.4 \\ I_{H_3} &= w_{x3}x_i + w_{y3}y_i + \theta_{H_3} = y_i - 0.2 \\ I_{H_4} &= w_{x4}x_i + w_{y4}y_i + \theta_{H_4} = -y_i + 1.0. \end{aligned}$$

Hence, the free elements in the equations describing the obstacle are represented by the biases θ_{H_i} and the weights in the equations for I_{H_i} are the coefficients in the respective equations.

Example: 2. *Circular shape obstacle.*

When an obstacle has a circular shape, I_{H_m} is expressed as:

$$I_H = R^2 - (x_i - P)^2 - (y_i - Q)^2, \tag{6}$$

where R is the radius and (P, Q) are the coordinates of the centre.

Example: 3. *Description of elliptical obstacles.*

When the obstacle has elliptic (circular) shape, I_{H_m} is expressed as:

$$I_H = a^2b^2 - (X - x_i)^2b^2 + (Y - y_i)^2a^2, \tag{7}$$

which comes directly from the standard equation of the ellipse

$$\frac{(X - x_i)^2}{a^2} + \frac{(Y - y_i)^2}{b^2} = 1,$$

with xy-coordinates of the foci $(-\sqrt{a^2 - b^2} + X, Y)$ and $(\sqrt{a^2 - b^2} + X, Y)$ respectively, and the description network has two neurons in the middle layer.

Note 1. *The oval shaped obstacles are represented by neural network having two neurons (see the above Example 2 and Example 3) in the middle layer.*

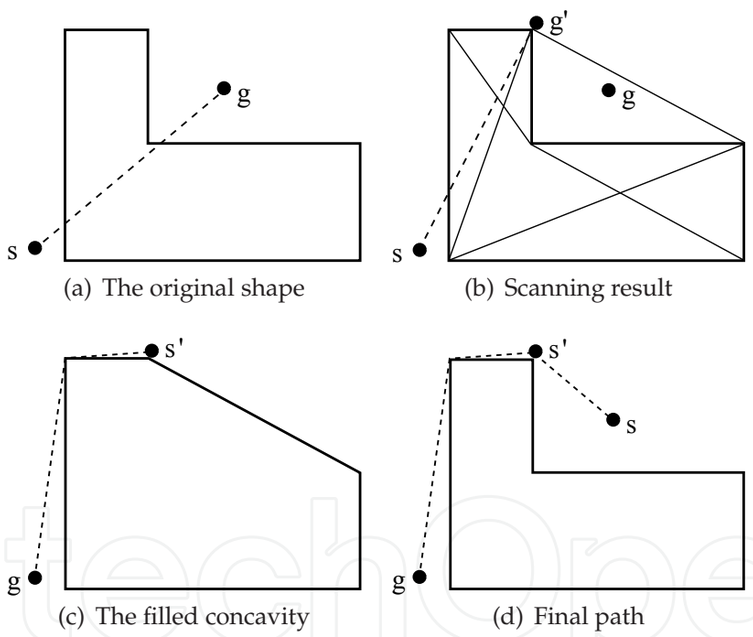
It is obvious from the above that any shape which can be expressed mathematically can be represented by the description neural network. This, of course, includes configurations which are a combination of elementary shapes.
The description of the obstacles by the shown here network has the advantage that it can be used for parallel computation of the path, which can increase the speed of path generation. As it will be shown later, this description of the obstacles has the superiority that the calculation time depends only on the total number of the obstacles' vertices.

3.2 The local minima problem

The local minima remain an important cause of inefficiency for potential field methods. Hence, dealing with local minima is the major issue that one has to face in designing a planner based on this approach. This issue can be addressed at two levels (Latombe (1991)): (1) definition of the potential function, by attempting to specify a function with no or few local minima, and (2) in the design of the search algorithm, by including appropriate techniques for escaping from local minima. However, it is not easy to construct an "ideal" potential function with no local minima in a general configuration. The second level is more realistic and is

addressed by many researchers (see e.g. Latombe (1991); Lozano-Pèrez et al. (1994) and the references there).
In the proposed in this chapter algorithm, the local minima problem is addressed in a simple and efficient fashion:

1. After setting the coordinates of the start and the goal points respectively (Fig. 3(a)), the polygonal obstacles are scanned in order to detect concavity (Fig. 3(b)). In the process of scanning, every two other vertices of a given obstacle are connected by a straight line and if the line lies outside the obstacle, then a concavity exists.
2. If the goal (or the start) point lies inside a concavity, then a new, temporary, goal (start) lying outside the concavity is set (point g' in Fig. 3(b)), and the initial path between the original goal (start) and the new one is set as a straight line. The temporary goal (start) is set at the nearest to the "original" start (goal) vertex.
3. Every detected concavity is temporarily filled, i.e. the obstacle shape is changed from a concave to a nonconcave one (see Fig. 3(c)). After finishing the current path-planning task, the original shapes are retained so that the next task could be planned correctly, and the temporary goal (start) is connected to the original one by a straight line (Fig. 3(d)).



(a) The original shape (b) Scanning result (c) The filled concavity (d) Final path

Fig. 3. Concave obstacle (the local minima solution)

The above allows isolating the local minima caused by concavities and increases the reliability of the path planning.

4. Optimal path planner

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

$$E = w_l E_l + w_c E_c, \tag{8}$$

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

$$E_l = \sum_{i=1}^{N-1} L_i^2 = \sum_{i=1}^{N-1} [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2], \tag{9}$$

and E_c is given by the expression.

$$E_c = \sum_{i=1}^{N} \sum_{k=1}^{K} C_i^k, \tag{10}$$

where N is the number of the points between the start and the goal, K is the number of the obstacles, and C is obtained through eq. (1).
Minimizing eq. (8) will lead to obtaining an optimal in length path that does not collide with any of the obstacles. In order to minimize (8) the classical function analysis methods are applied. First, we find the time derivative of E:

$$\begin{aligned} \frac{dE}{dt} = & \sum_{i=1}^{N} \left[w_l \left(\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i} \right) + w_c \sum_{k=1}^{K} \frac{\partial C_i^k}{\partial x_i} \right] \dot{x}_i \\ & + \sum_{i=1}^{N} \left[w_l \left(\frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i} \right) + w_c \sum_{k=1}^{K} \frac{\partial C_i^k}{\partial y_i} \right] \dot{y}_i \end{aligned} \tag{11}$$

Setting

$$\begin{aligned} \dot{x}_i &= -\eta \left[w_l \left(\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i} \right) + w_c \sum_{k=1}^{K} \frac{\partial C_i^k}{\partial x_i} \right], \\ \dot{y}_i &= -\eta \left[w_l \left(\frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i} \right) + w_c \sum_{k=1}^{K} \frac{\partial C_i^k}{\partial y_i} \right] \end{aligned} \tag{12}$$

we can rewrite the above equation as

$$\frac{dE}{dt} = -\frac{1}{\eta} \sum_{i=1}^{N} (\dot{x}_i^2 + \dot{y}_i^2) < 0 \tag{13}$$

where η is an adaptation gain. It is obvious that, when $\dot{x}_i \to 0$ and $\dot{y}_i \to 0$, E converges to its minimum. In other words, when all points of the path almost stop moving, there is no collision and the path is the optimal one, i.e. eq. (13) can be used as a condition for termination of the calculation iterations.
Now, from equations (12),

$$\begin{aligned} \frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i} &= -2x_{i+1} + 4x_i - 2x_{i-1}, \\ \frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i} &= -2y_{i+1} + 4y_i - 2y_{i-1} \end{aligned} \tag{14}$$

and

$$\frac{\partial C_i^k}{\partial x_i} = \frac{\partial C_i^k}{\partial (I_0)_i^k}\frac{\partial (I_0)_i^k}{\partial x_i} = \frac{\partial C_i^k}{\partial (I_0)_i^k}\left(\sum_{m=1}^{M}\frac{\partial (O_{H_m})_i^k}{\partial (I_{H_m})_i^k}\frac{\partial (I_{H_m})_i^k}{\partial x_i}\right)$$
$$= f'((I_0)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{xm}^k\right);$$
$$\frac{\partial C_i^k}{\partial y_i} = \frac{\partial C_i^k}{\partial (I_0)_i^k}\frac{\partial (I_0)_i^k}{\partial y_i} = \frac{\partial C_i^k}{\partial (I_0)_i^k}\left(\sum_{m=1}^{M}\frac{\partial (O_{H_m})_i^k}{\partial (I_{H_m})_i^k}\frac{\partial (I_{H_m})_i^k}{\partial x_i}\right)$$
$$= f'((I_0)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{ym}^k\right). \tag{15}$$

This leads to the final form of the function:

$$\dot{x}_i = -2\eta w_l(2x_i - x_{i-1} - x_{i+1}) - \eta w_c \sum_{k=1}^{K} f'((I_0)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{xm}^k\right);$$
$$\dot{y}_i = -2\eta w_l(2y_i - y_{i-1} - y_{i+1}) - \eta w_c \sum_{k=1}^{K} f'((I_0)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{ym}^k\right), \tag{16}$$

where f' is given by the following expressions:

$$f'(\cdot) = \frac{1}{T} f(\cdot)[1 - f(\cdot)]$$
$$f'_{H_m}(\cdot) = \frac{1}{T} f_{H_m}(\cdot)[1 - f_{H_m}(\cdot)]. \tag{17}$$

In eq. (16) the first member in the right side is for the path length optimization and the second one is for the obstacle avoidance.
One of the important advantages of the above path-planning is that it allows parallelism in the calculations of the neural network outputs (see Section 6), which leads to decreasing the computational time. The generated semi-optimal path can be optimized further by applying evolutionary methods (e.g. genetic algorithm). Such approach leads to an optimal solution but the computational cost increases dramatically (Yu et al. (2003)).

5. The path-planning algorithm

In this section an algorithm based on the background given in Sections 3 and 4 is proposed. The calculations for the path are conceptually composed by the following steps:

Step 1: Initial step

1. Let the start position of the robot is (x_1, y_1), and the goal position is denoted as (x_N, y_N).
2. Check for concavities (see Section 3.2.) and, if necessary, reassign the goal (start) position.
3. At $t = 0$ the coordinates of the points of the initial path (straight line) $(x_i, y_i;\ \ i = 2, 3, \ldots, N-1)$ are assigned as

$$x_i = x_0 + i(x_N - x_1)/(N-1),$$
$$y_i = (y_N - y_1)(x_i - x_1)/(x_N - x_1) + y_0, \tag{18}$$

i. e. the distance between the x and y coordinates of every two neighboring points of the path is equal.

Note 2. *It is assumed that the obstacles dimensions are enlarged by the robot's polygon dimensions (Lozano-Pèrez & Wesley (1979)).*

Step2: 1. For the points (x_i, y_i) of the path which lie inside some obstacle, the iterations are performed according to the following equations:

$$\begin{aligned}\dot{x}_i &= -2\eta_1 w_l(2x_i - x_{i-1} - x_{i+1}) - \eta_1 w_c \sum_{k=1}^{K} f'((I_0)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{xm}^k\right);\\ \dot{y}_i &= -2\eta_1 w_l(2y_i - y_{i-1} - y_{i+1}) - \eta_1 w_c \sum_{k=1}^{K} f'((I_0)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{ym}^k\right) \qquad (19)\\ & i = 2, 3, \ldots, N-1.\end{aligned}$$

2. For the points (x_i, y_i) situated outside the obstacles, instead of eq. (19) use the following equations:

$$\begin{aligned}\dot{x}_i &= -\eta_2 w_l(2x_i - x_{i-1} - x_{i+1});\\ \dot{y}_i &= -\eta_2 w_l(2y_i - y_{i-1} - y_{i+1}),\end{aligned} \qquad (20)$$

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.

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

Step 4: Test for convergence

Calculate the difference d of the path lengths at time t and time $(t+p)$, i.e.

$$d = \sum_{i=2}^{N-1} \{[x_i(t+p) - x_i(t)]^2 + [y_i(t+p) - y_i(t)]^2\}^{1/2}. \qquad (21)$$

- If $d < \varepsilon$ then the algorithm terminates with the conclusion that the goal is reached via an "optimal" path. Here ε is a small constant, say $\varepsilon = 0.1$.
- If $d \geq \varepsilon$, then GO TO step 2.

Here eq. (19) is almost the same as eq. (16) with the difference that instead of only one RPF, different functions, as explained below, are used for every layer of the neural network. Every obstacle is described using a neural network as shown in Fig. 1. The output neuron is described by eq. (1), the neuron function $f(\cdot)$ is the same as in eq. (3) and the pseudotemperature is as in eq. (4).
The hidden layer inputs are as in eq. (1) but the outputs O_{H_m} now become

$$O_{H_m} = f_{H_m}(I_{H_m}), \quad m = 1, \ldots, M \qquad (22)$$

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

$$f_{H_m}(I_{H_m}) = \frac{1}{1 + e^{-I_{Hm}/T_{Hm}(t)}} \qquad (23)$$

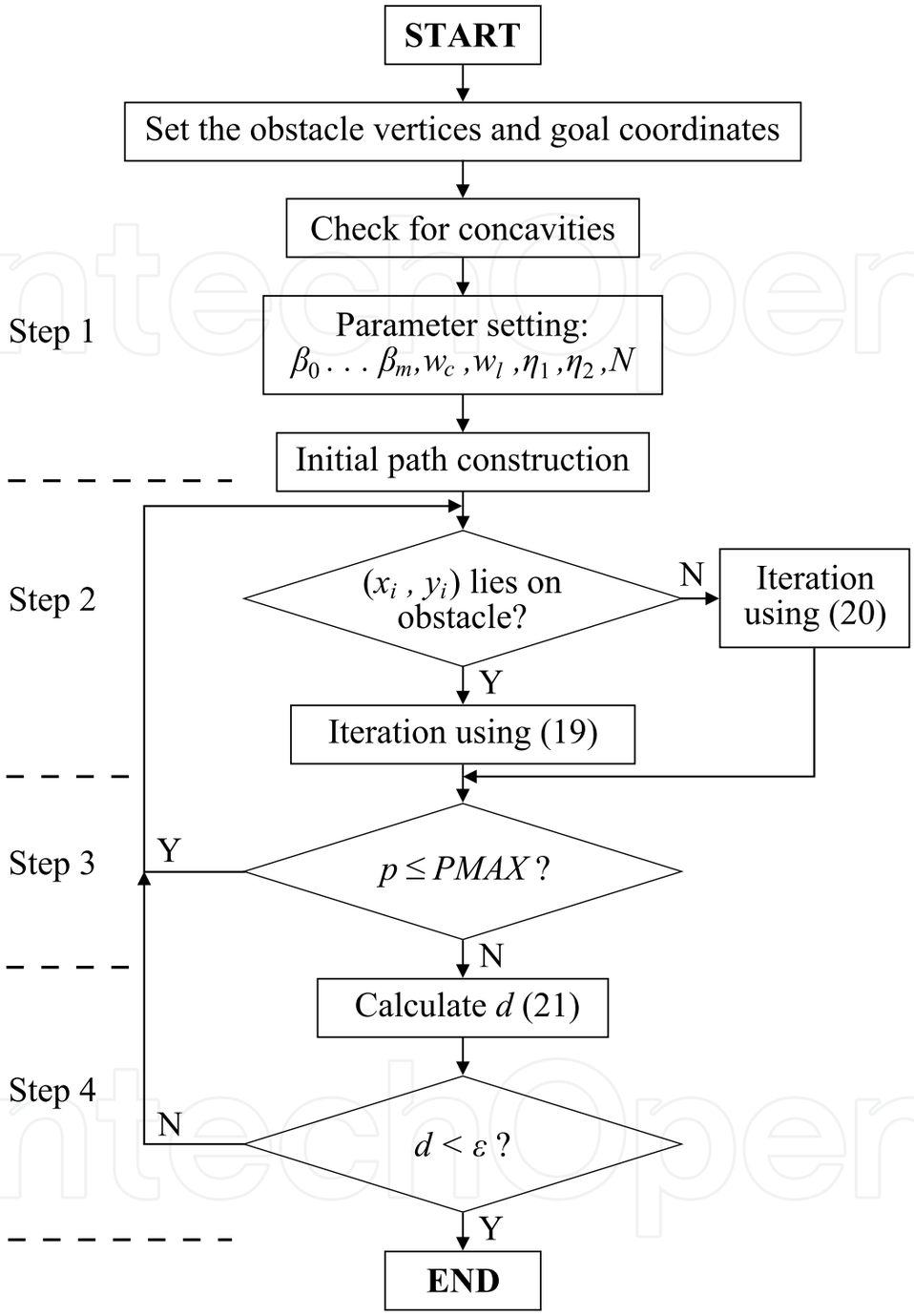


Fig. 4. Block diagram of the algorithm

and

$$T_{H_m}(t) = \frac{\beta_m}{\log(1+t)} \tag{24}$$

Finally, I_{H_m} is given by the activating function (5). Equations (3) and (23) include different values of pseudotemperatures, which is one of the important conditions for convergence of

the algorithm and for generation of almost optimal path. The block diagram of the algorithm is shown in Fig. 4.

6. Experimental results

To show the effectiveness of the proposed in this paper algorithm, several simulation examples are given in this section.

6.1 Calculation speed evaluation

The simulations were run on ASUS A7V motherboards rated with 328 base (352 peak) SPEC CFP2000 and 409 base (458 peak) SPEC CINT2000 on the CPU2000 benchmark. The environment shown in Fig. 5 has been chosen for the benchmark tests. In the course of the speed measurements the number of the vertices was increased from 20 to 120 with step of 20 vertices. The final configuration of the obstacles inside the benchmark environment with three generated paths is shown in Fig. 5. To compare the run-times, 12 path generations have been performed randomly. After deleting the lowest and the highest path planning times, the average of the remaining 10 values has been adopted.

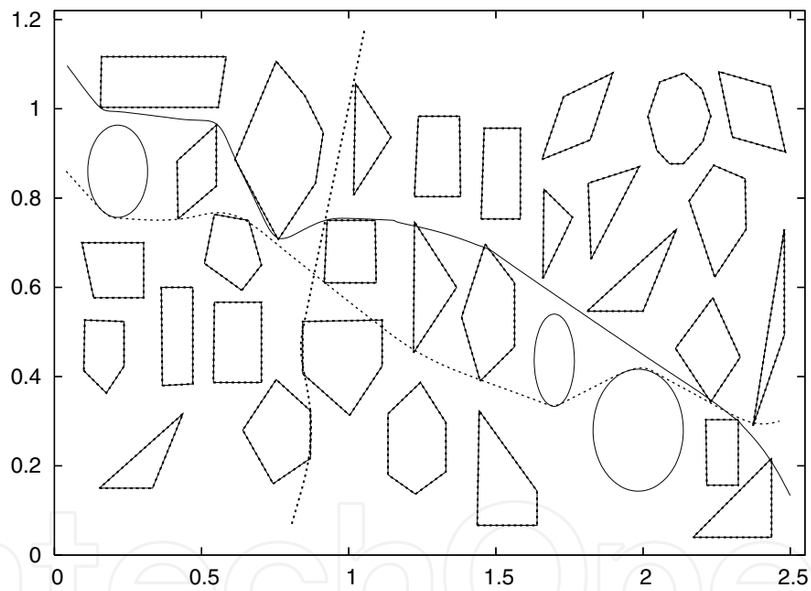


Fig. 5. The benchmark environment with three generated paths

Figure 6 shows the speed change depending on the total number of vertices of the obstacles. It is clear that the speed changes linearly with increasing the number of the vertices (cf. Lozano-Pèrez & Wesley (1979)).

As it was explained at the end of Section 4 the algorithm allows parallelizing the path computation. To calculate the speedups we ran parallel simulations for the same benchmark environment on one, two, and three processors. Figure 7 shows how the speed increases with increasing the number of the processors. For the case of two processors the simulation was run on two ASUS A7V boards, and for the case of three processors a Gigabyte 440BX board was added as a third processor. The communication during the parallel calculations was established by an Ethernet bus based network between the computers.

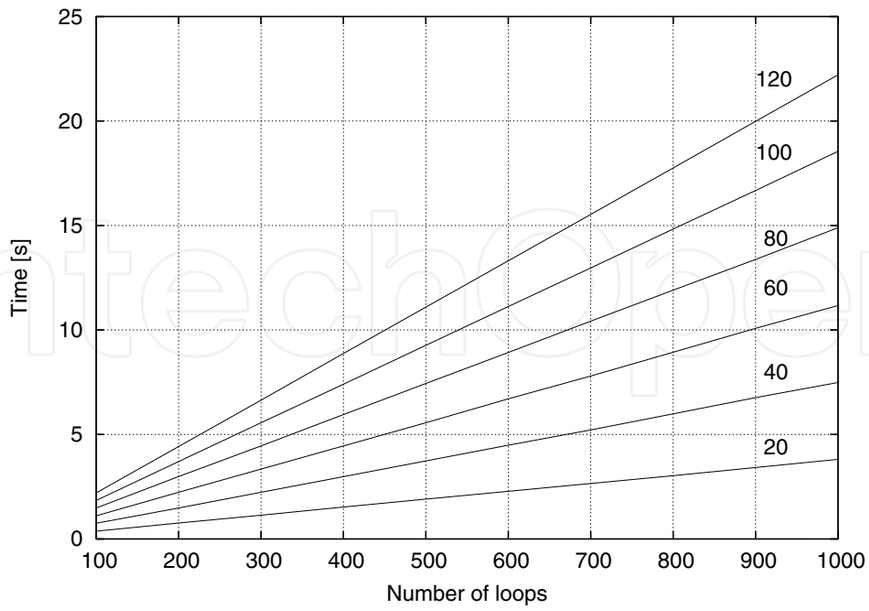


Fig. 6. Computation time depending on the number of the loops (100–1000) and number of the vertices (20–120)

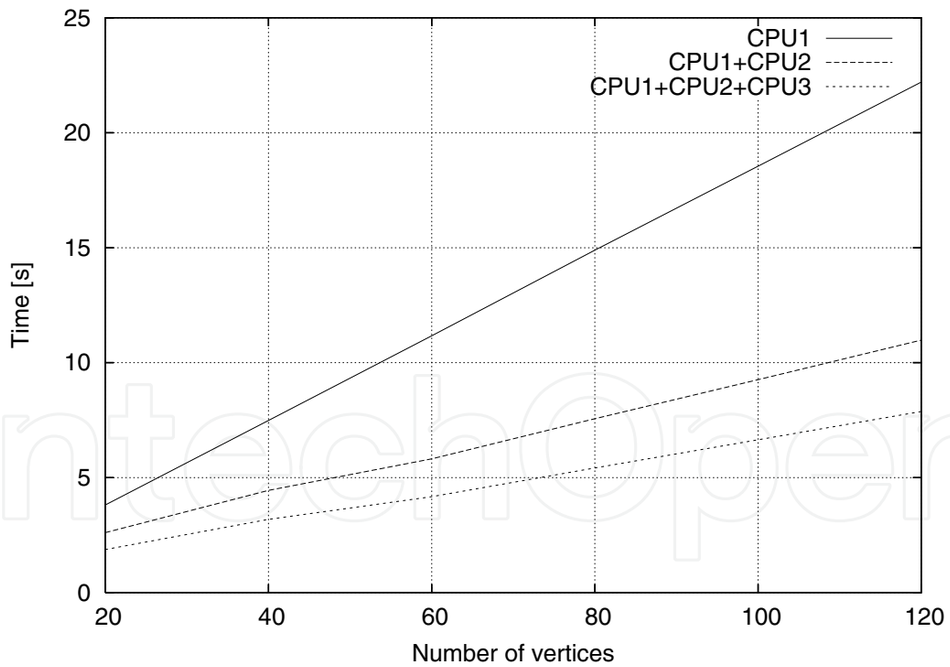


Fig. 7. Run-time t for different number of processors for 1000 loops depending on the number of vertices

It is possible to further increase the calculation speed by introducing adaptive adjustment of the parameters η_1 and η_2. For example, the following adaptive adjusting law increases the speed about three times (see Fig. 8):

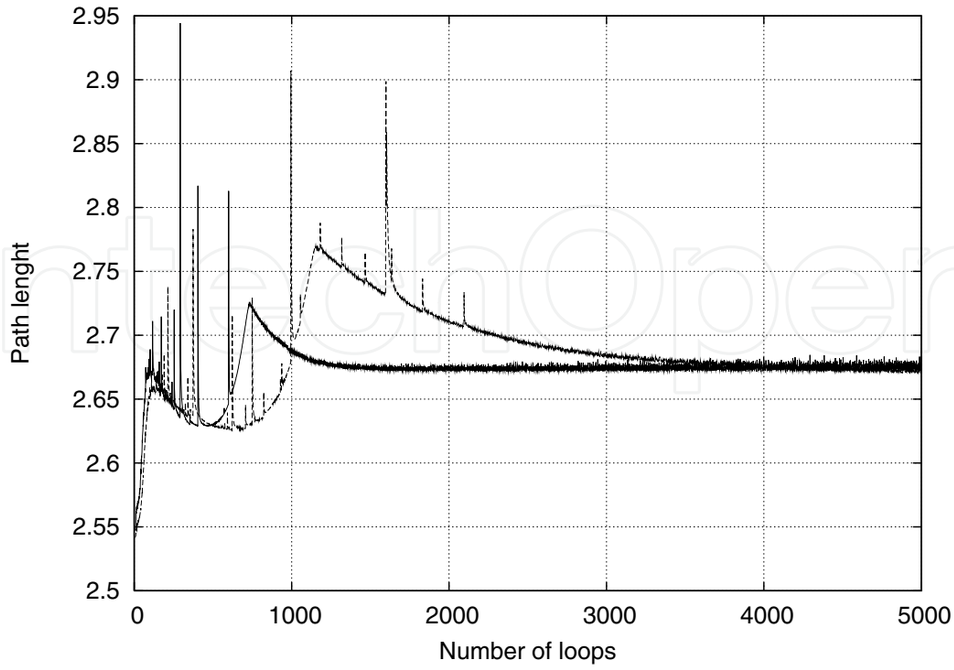


Fig. 8. Convergence speed of the algorithm depending on the way of setting η_1 and η_2: the slowly converging line is for the case when constant values are used, and the fast converging one depicts the convergence when adaptive adjustment is applied

$$\begin{aligned} \dot{\eta}_1(n+1) &= -\eta_1(n) + \delta\,\mathrm{sgn}[(d(n)-d(n-1))(d(n-1)-d(n-2))], \\ \dot{\eta}_2(n+1) &= -\eta_2(n) + \delta\,\mathrm{sgn}[(d(n)-d(n-1))(d(n-1)-d(n-2))] \end{aligned} \tag{25}$$

where δ is a small constant.

In the course of path calculations, a limited number of obstacles has impact on path generation—mostly the obstacles the initial, straight line path intersects with, contribute to path forming. Then, while performing *Step 1* of the algorithm, if all obstacles which do not collide with the initial path are ignored and excluded from the calculations, it is possible to drastically decrease the calculation time. A good example is the path in vertical direction in Fig. 5. When all obstacles are considered, the total number of vertices is 135 and RPF for all obstacles would be calculated. However, if only the four obstacles which collide with the initial straight line are used, the number of vertices decreases to only 17 and the expected calculation time will be approximately 8 times shorter. After finishing the path generation for the decreased number of obstacles, the original environment should be retained and a final correction of the path shape should be performed. Naturally, this final correction would require only few additional calculation loops.

6.2 Simulation results

Figure 9 shows additional results of path planning simulation in an environment with concaved obstacles. It can be seen that the local minimums do not cause problems in the path generation. In these simulations the number of the points between the start position and the goal was set to 200. In both examples the values of the coefficients in equations of the algorithm were not changed. In fact, the authors have performed many simulations using different environments, in which the obstacles were situated in many different ways. In all

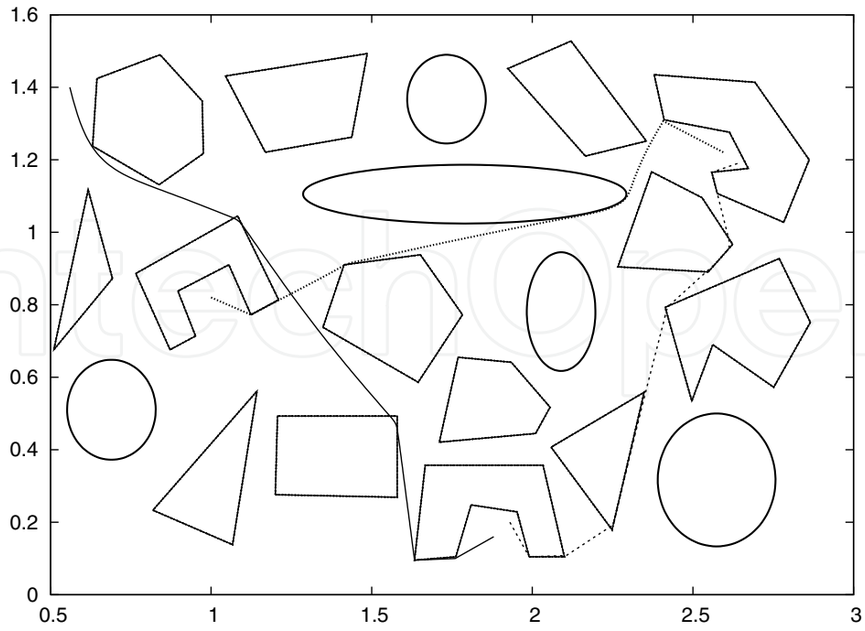


Fig. 9. Simulation example (environment including concave obstacles)

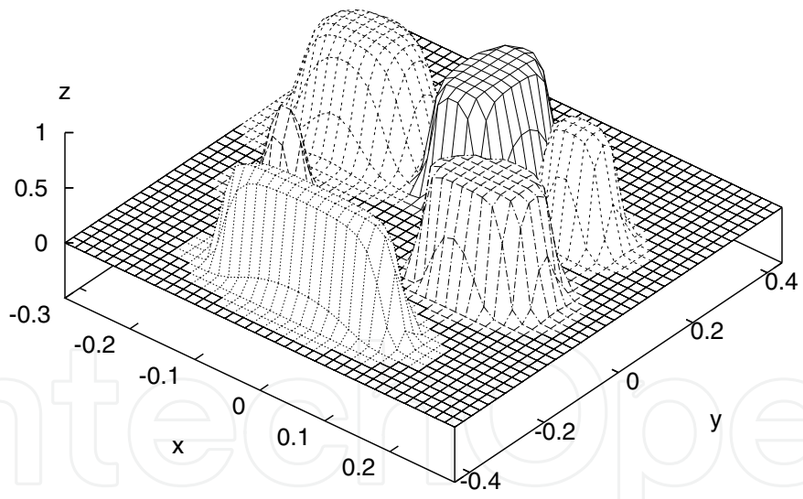


Fig. 10. Shape of the RPF in the course of simulation

the simulations the paths were successfully generated and the time for the calculations agreed with the calculation speed explained above. Figure 10 shows the potential fields of obstacles in the course of a simulation.

We have developed software with a graphic user interface (GUI) for the simulation and control of differential drive robots in a 2D environment (See Fig. 11). The environment allows the user to easily build configuration spaces and perform simulations. This environment is available at *http://shiwasu.ee.ous.ac.jp/planner/planner.zip*. The algorithm was successfully applied to the

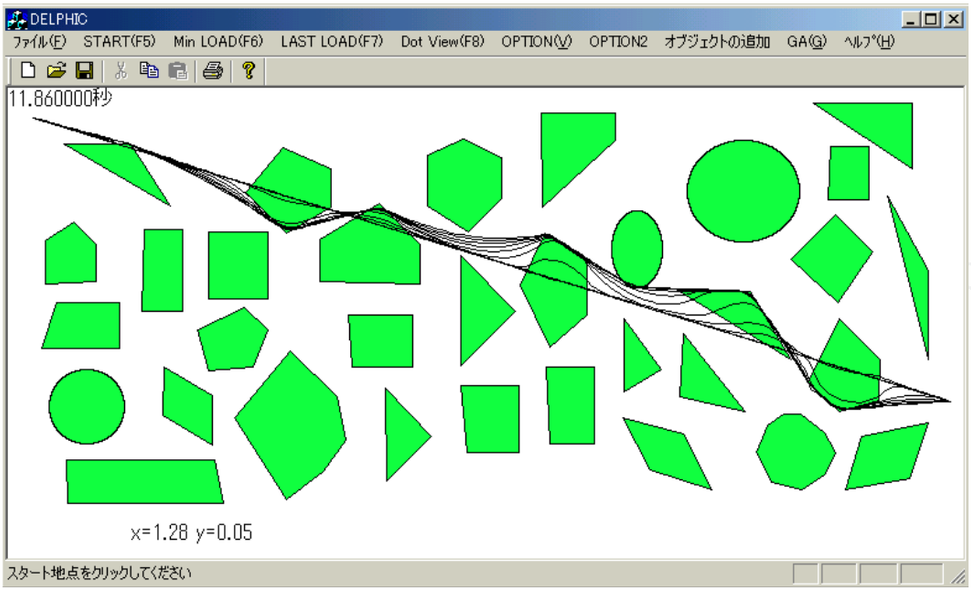


Fig. 11. GUI environment for simulation and control of differential drive robots

control and navigation of "Khepera" (Mondada et al. (1993)) and "Pioneer P3-DX" robots.

7. Discussion and conclusions

In this paper we have proposed an algorithm which guarantees planning of near optimal in length path for wheeled robots moving in an *a priori* known environment. We are considering wheel placements which impose *no* kinematic constraints on the robot chassis: castor wheel, Swedish wheel, and spherical wheel.

The proposed algorithm is a significant improvement of the potential field algorithms because it finds an almost optimal path without being trapped in local minima and the calculation speed for the proposed algorithm is reasonably fast. Because the only assumptions made are that the obstacles are stationary polygons and ovals, the algorithm can solve the navigation problem in very complex environments. For description of the obstacles the algorithm uses only the Cartesian coordinates of their vertices and does not need memory consuming obstacle transformations (cf. Zelinsky (1992); Zelinsky & Yuta (1993)). The consumed memory space is equal to twice the number of the coordinates of the path points and some additional memory for keeping the parameters of the description networks.

Because the generated paths are piecewise linear with changing directions at the corners of the obstacles, the inverse kinematics problems for the case of differential drive robots are simply solved: to drive the robot to some goal pose $(x,\ y,\ \theta)$, the robot can be spun in place until it is aimed at $(x,\ y)$, then driven forward until it is at $(x,\ y)$, and then spun in place until the required goal orientation θ is met.

It becomes clear from the course of the above explanations that the effectiveness of the proposed algorithm depends on the development of some mechanism for choosing the initial values of the pseudotemperatures (β_m) or properly adjusting the pseudotemperatures in order to generate potential fields satisfying the properties given in Section 2. Applying of an adaptive adjusting procedure to the coefficients β_m similar to that for adjusting of η_1 and η_2 (see. eq. (25)) is one possible solution. Moreover, there is no need to perform continuous adjustment throughout the whole path generation—adjustment in the first few steps gives

good approximations for the initial values of the β_m coefficients. Strict solution of this problem is one of our next goals.
The proposed algorithm can be easily extended for the 3-D case (Kroumov et al. (2010)). The extension can be done by adding an additional input for the z (height of the obstacles) dimension to the obstacle description neural network. Adding of z coordinate and employing a technique for path planning, similar to that proposed in Henrich et al. (1998), would allow further adoption of the algorithm for planning of paths for industrial robots. Development of such application is another subject of our future work.

8. Acknowledgement

This work is partially supported by the Graduate School of Okayama University of Science, Okayama, Japan.
The authors would like to thank Mr. Yoshiro Kobayashi, a graduate student at the Graduate School of Okayama University of Science, for his help in preparing part of the figures. We are thankful to Ms Z. Krasteva for proofreading the final version of the manuscript.

9. References

Canny, J. F. (1988). *The Complexity of Robot Motion Planning*, The MIT Press, Cambridge, MA, U. S. A.

Chen, P. C. & Hwang, Y. K. (1998). Sandros: A dynamic graph search algorithm for motion planning, *IEEE Transactins on Robotics & Automation* 14(3): 390–403.

Faverjon, B. & Tournassoud, P. (1987). A local approach for path planning of manipulators with a high number of degrees of freedom, *Proceedings of IEEE International Conference on Robotics & Automation*, Raleigh, New Caledonia, U. S. A., pp. 1152–1159.

Ge, S. S. & Cui, Y. J. (2000). New potential functions for mobile robot path planning, *IEEE Transactins on Robotics & Automation* 16(5): 615–620.

Henrich, D., Wurll, C. & Wörn, H. (1998). 6 dof path planning in dynamic environments—a parallel on-line approach, *Proceedings of the 1998 IEEE Conference on Robotics & Automation*, Leuven, Belgium, pp. 330–335.

Hwang, Y. K. & Ahuja, K. (1992). Potential field approach to path planning, *IEEE Transactins on Robotics & Automation* Vol. 8(No. 1): 23–32.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *International Journal of Robotics Research* 5(1): 90–98.

Khosla, P. & Volpe, R. (1988). Superquadratic artificial potentials for obstacle avoidance and approach, *Proceedings of IEEE International Conference on Robotics & Automation*, Leuven, Belgium, pp. 1778–1784.

Kroumov, V., Yu, J. & Negishi, H. (2004). Path planning algorithm for car-like robot and its application, *Journal of Systemics, Cybernetics and Informatics* Vol. 2(No. 3): 7.

Kroumov, V., Yu, J. & Shibayama, K. (2010). 3d path planning for mobile robots using simulated annealing neural network, *International Journal on Innovative Computing, Information and Control* 6(7): 2885–2899.

Latombe, J. C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers, Norwell.

Lee, S. & Kardaras, G. (1997a). Collision-free path planning with neural networks, *Proceedings of 1997 IEEE International Conference on Robotics & Automation*, Vol. 4, pp. 3565–3570.

Lee, S. & Kardaras, G. (1997b). Elastic string based global path planning using neural networks, *Proceedings of 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'97)*, pp. 108–114.

Lozano-Pèrez, T., Mason, M. T. & Taylor, R. H. (1994). Automatic synthesis of fine motion strategies for robots, *International Journal of Robotics Research* 3(1): 3–24.

Lozano-Pèrez, T. & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles, *Communications of the ICM* Vol. 22(No. 10): 560–570.

Mondada, F., Franzi, E. & Ienne, P. (1993). Mobile robot miniaturization: a tool for investigation in control algorithms, *Proceedings of ISER3*, Kyoto, Japan.

O'Dúnlaing, C. & Yap, C. K. (1982). A retraction method for planning the motion of a disk, *Journal of Algorithms* Vol. 6: 104–111.

Paden, B., Mees, A. & Fisher, M. (1989). Path planning using a jacobian-based freespace generation algorithm, *Proceedings of IEEE International Conference on Robotics & Automation*, Scottsdale, Arizona, U. S. A., pp. 1732–1737.

Rimon, E. & Doditschek, D. E. (1992). Exact robot navigation using artificial potential fields, *IEEE Transactins on Robotics & Automation* Vol. 8(No. 5): 501–518.

Sun, Z. Q., Zhang, Z. X. & Deng, Z. D. (1997). *Intelligent Control Systems*, Tsinghua University Press.

Tsankova, D. (2010). Neural networks based navigation and control of a mobile robot in a partially known environment, *in* A. Barrera (ed.), *Mobile Robots Navigation*, InTech, pp. 197–222.

Warren, C. W. (1989). Global path planning using artificial potential fields, *Proceedings of IEEE International Conference on Robotics & Automation*, pp. 316–321.

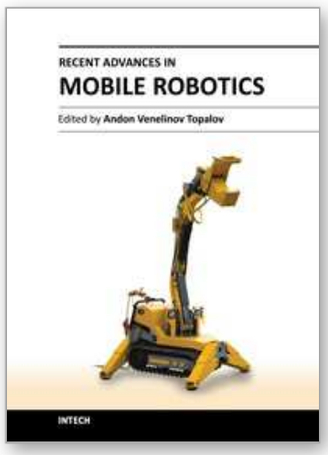
Yamamoto, M., Iwamura, M. & Mohri, A. (1998). Near-time-optimal trajectory planning for mobile robots with two independently driven wheels considering dynamical constraints and obstacles, *Journal of the Robotics Society of Japan* 16(8): 95–102.

Yu, J., Kroumov, V. & Narihisa, H. (2002). Path planning algorithm for car-like robot and its application, *Chinese Quarterly Journal of Mathematics* Vol. 17(No. 3): 98–104.

Yu, J., Kroumov, V. & Negishi, H. (2003). Optimal path planner for mobile robot in 2d environment, *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Vol. 3, Orlando, Florida, U. S. A., pp. 235–240.

Zelinsky, A. (1992). A mobile robot exploration algorithm, *IEEE Transactins on Robotics & Automation* Vol. 8(No. 6): 707–717.

Zelinsky, A. & Yuta, S. (1993). Reactive planning for mobile robots using numeric potential fields, *Proceedings of Intelligent Autonomous Systems 3 (IAS3)*, pp. 84–93.



Recent Advances in Mobile Robotics

Edited by Dr. Andon Topalov

ISBN 978-953-307-909-7
Hard cover, 452 pages
Publisher InTech
Published online 14, December, 2011
Published in print edition December, 2011

Mobile robots are the focus of a great deal of current research in robotics. Mobile robotics is a young, multidisciplinary field involving knowledge from many areas, including electrical, electronic and mechanical engineering, computer, cognitive and social sciences. Being engaged in the design of automated systems, it lies at the intersection of artificial intelligence, computational vision, and robotics. Thanks to the numerous researchers sharing their goals, visions and results within the community, mobile robotics is becoming a very rich and stimulating area. The book Recent Advances in Mobile Robotics addresses the topic by integrating contributions from many researchers around the globe. It emphasizes the computational methods of programming mobile robots, rather than the methods of constructing the hardware. Its content reflects different complementary aspects of theory and practice, which have recently taken place. We believe that it will serve as a valuable handbook to those who work in research and development of mobile robots.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen