Optimal Path Planner for Mobile Robot in 2D Environment

Valeri KROUMOV
Department of Electronic Engineering, Okayama University of Science
1-1 Ridai-cho, Okayama, 700-0005, Japan

Jianli YU
Department of Mathematics and Physics, Henan University of Science and Technology
Luoyang, 471003, China

and

Hiroshi NEGISHI
QUICK VALUE Co.,
699 Irinaka, Bizen City, Okayama 705-0023, Japan

ABSTRACT

The problem of path planning for the case of a mobile robot moving in an environment filled with obstacles with known shapes and positions is studied. A path planner based on the genetic algorithm approach, which generates optimal in length path is proposed. The population member paths are generated by another algorithm, which uses for description of the obstacles an artificial annealing neural network and is based on potential field approach. The resulting path is piecewise linear with changing directions at the corners of the obstacles. Because of this feature, the inverse kinematics problems in controlling 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. Simulation results show the validity of the proposed algorithm.

Keywords: Optimal Path Planning, Mobile Robots, Genetic Algorithm, Artificial Annealing, Potential Field.

1. INTRODUCTION

Path planning is a fundamental issue in the field of mobile robot 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, not less important issue is to compute *realizable* and, if possible, *optimal path*, bringing the vehicle to the final position. This paper addresses the optimal path planning and proposes a path planning which employs a genetic algorithm (GA) [1].

Many methods for path planning of a mobile robot have been proposed in the recent years [2]–[14]. Recently, applications of GAs to path planning have been recognized [7]–[11]. GA is a search strategy using a mechanism analogous to evolution of life in nature. It is widely recognized that GA works even for complex problems such that the traditional algorithms cannot find a satisfactory solution.

Some of the previously proposed path planners based on GAs have drawbacks which lead to costly computations and could not fully exploit the abilities and benefits of using GAs. For example the motion planner proposed in [11] uses a binary string with fixed length for path coding which is not easy to construct, and it seems that the speed of evolution in GA slows down when there is an obstacle restricting the area the optimal path goes through. In many path planners GAs are entirely involved in the planning process [8], [9] which seems to increase the computational cost.

This paper proposes an algorithm for generating optimal in length path for differential drive robots. The proposed algorithm is based on GAs but the GAs is involved only in the optimization process. The actual path generation is performed by a fast algorithm based on the potential field approach [12], which is not trapped by local minima. 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.

The paper is organized as follows. In the next section an algorithm based on potential field approach [14] is briefly presented. We use this algorithm for the paths generation through the evolutionary process of the GA. Section 3 is central part of the paper and proposes an algorithm for optimal path generation based on GA. In Section 4 we show the effectiveness of the proposed algorithm by presenting some simulation results. In the final section we are discussing the results and some plans for future developments. The Appendix shows the approach to local minima problem solution used here.

The proposed here algorithm solves the local minimum problems and generates optimal path in relatively small number of calculations. Only the assumptions made in this work are that there are finite number of stationary polygonal or oval obstacles with finite number of vertices, and that the robot polygon also has a finite number of vertices. 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 [13].

2. ALMOST OPTIMAL PATH PLANNER [14]

In this section we briefly present the fast path planning algorithm which we proposed in [14]. For brevity we will call it $A+$ path planner. This algorithm generates optimized in length paths

but, as it can be seen from the following explanation the algorithm *does not guarantee* that the generated path is optimal. This 'almost' optimal algorithm is used for producing populations for the proposed in this paper planner based on GA.

Obstacles description

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

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

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 $f_{H_m}(\cdot)$ has the form:

$$f_{H_m}(x) = \frac{1}{1 + e^{-x/T_{H_m}}}, \tag{3}$$

where T_{H_m} is the pseudotemperature, and the induced local field (x) of the neuron is equal to I_O for Eq. (1) or equal to I_{H_m} in the case of Eq. (2). The pseudotemperature decrease is given by

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

and

$$T_{H_m}(t) = \frac{\beta_m}{\log(1+t)}. \tag{5}$$

respectively.

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 free element in the equation expressing the shape of the obstacle. When the obstacle has elliptic (circular) shape, I_{H_m} is expressed as:

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

and the description network has two neurons in the middle layer. Examples of descriptions of some more complicated objects and additional comments are given in [14].

The description of the obstacles by the shown here network has an advantage that it can be used for parallel computation of the path, thus increasing the speed of the path generation. This description can be easily extended for the 3-D case. The extension can be done by adding an additional input for the z (height of the obstacles) dimension to the obstacle description neural network. It is shown in [12] that the calculation time is linearly proportional to the number of the obstacles' vertices and decreases linearly with parallelizing the calculations.

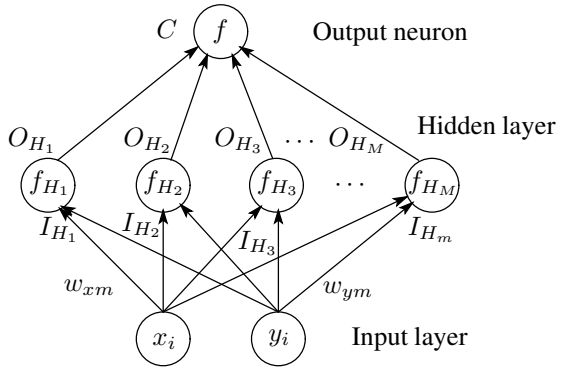

Figure 1: Obstacle description network

"Almost" optimal path planning algorithm

In this section an algorithm based on the background given in the previous section is briefly explained. 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 goal, K is the number of the obstacles, and C is obtained through Eq. (1).

The idea is to minimize Eq. (8) which means to obtain an "almost" optimal in length path, that does not collide with any of the obstacles. Differentiating the Eq. (8) and performing further reductions and calculations can be summarized in the following path planning algorithm [14]:

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 the Appendix) and, if necessary, perform elimination of local minima.
3. At $t = 0$ the coordinates of the points of the initial path (straight line) $(x_i, y_i;\ \ i = 2, \ldots, N-1)$ are assigned as

$$\begin{aligned} x_i &= x_0 + i(x_{N-1} - x_0)/(N-1), \\ y_i &= (y_{N-1} - y_0)(x_i - x_0)/(x_{N-1} - x_0) + y_0, \end{aligned} \tag{11}$$

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

Note 1 *The obstacles dimensions here are enlarged by the robot's polygon dimensions [13].*

Step2: Path generation

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

$$\begin{aligned}\dot{x}_i &= -2\eta_1 w_l(2x_i - x_{i-1} - x_{i+1}) \\ &-\eta_1 w_c \sum_{k=1}^{K} f'((I_O)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{xm}^k\right); \\ \dot{y}_i &= -2\eta_1 w_l(2y_i - y_{i-1} - y_{i+1}) \\ &-\eta_1 w_c \sum_{k=1}^{K} f'((I_O)_i^k)\left(\sum_{m=1}^{M} f'_{H_m}((I_{H_m})_i^k) w_{ym}^k\right), \\ i &= 2, \ldots, N-1. \end{aligned} \quad (12)$$

2. For the points (x_i, y_i) situated outside the obstacles, then instead of Eq. (12) use the following equations:

$$\begin{aligned}\dot{x}_i &= -\eta_2 w_l(2x_i - x_{i-1} - x_{i+1}); \\ \dot{y}_i &= -\eta_2 w_l(2y_i - y_{i-1} - y_{i+1}), \end{aligned} \quad (13)$$

   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.

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

Step 3: Test for convergence

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

$$d = \sum_{i=2}^{N-1} \{[x_i(t+p) - x_i(t)]^2 + [y_i(t+p) - y_i(t)]^2\}^{1/2}. \quad (14)$$

- 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.

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

Every obstacle is described using a neural network as shown in Figure 1. The output neuron is described by Eq. (1), the neuron function $f(t)$ and $f_{H_m}(t)$ are same as Eq. (3) and the pseudotemperatures are as in Eq. (4) and (5).

One of the important advantages of above path-planning is that it allows parallelism in the calculations of the neural network outputs, which leads to increasing the speed of the calculations. The generated path is semi-optimal and is further optimized by applying genetic algorithm in the foolowing section.

3. PATH OPTIMIZATION USING GENETIC ALGORITHM

In the context of applying GA into optimal path generation problem formulation, an appropriate chromosome structure will greatly enhance the computation process. Instead of the commonly-used binary version [2], the chromosome is formed as sequence of path points as shown below.

1. **Collection of initial paths**

The population pool consists of n randomly generated initial paths using the path planning algorithm presented in the previous section. Every initial path has random number of points. The population pool is generated as follows:

Let the start position of the robot is (x_1, y_1), and the goal position is denoted as (x_N, y_N). The first path of the population r_1^1 is generated starting from straight line between the start and the goal and the distance between every two neighboring points of the path is equal.

$$r_1^1 = \{(x_1, y_1), (\bar{x}_{12}^1, \bar{y}_{12}^1), \ldots, (\bar{x}_{1j}^1, \bar{y}_{1j}^1), \ldots, (x_N, y_N)\} \quad (15)$$

The other paths in the population pool are generated by random selection of a point from the initial (straight line) path r_1^1 and placing that point randomly inside the working area thus constructing an initial path consisting of two straight lines.

$$r_i^1 = \{(x_1, y_1), (\bar{x}_{i2}^1, \bar{y}_{i2}^1), \ldots, (\bar{x}_{ij}^1, \bar{y}_{ij}^1), \ldots, (x_N, y_N)\} \quad (16)$$

where $(\bar{x}_{ij}^1, \bar{y}_{ij}^1)$ belong to paths $i = 2, 3, \ldots, n$ and j take values $j = 2, 3, \ldots, N-1$.

Then from the collection of initial paths the first generation can be calculated by applying the $A+$ algorithm from the previous section, i.e.

$$\begin{aligned} R_i^1 &= \{(x_1, y_1), (x_{i2}^1, y_{i2}^1), \ldots, (x_{ij}^1, y_{ij}^1), \ldots, (x_N, y_N)\} \\ (i &= 1, 2, 3, \ldots, n) \end{aligned} \quad (17)$$

i.e. $r_i^1 \xrightarrow{A+} R_i^1$. We express the collection of path of the first generation with

$$\varphi^1 = \{R_1^1, R_2^1, \ldots, R_n^1\}.$$

2. **Parent selection.** A roulette-wheel-selection technique applies. The parent selection procedure operates as follows:

(a) sum the fitness of all chromosomes in the population

$$F_s = \sum_{i=1}^{n} f_i^1 = \sum_{i=1}^{n} \frac{1}{L_i^1} \quad (18)$$

where

$$L_i^1 = \sum_{j=1}^{N-1} [(x_{i(j+1)}^1 - x_{ij}^1)^2 + (y_{i(j+1)}^1 - y_{ij}^1)]^{\frac{1}{2}} \quad (19)$$

(b) generate a random number $0 < p \leq 1$

(c) based on the probability

$$P_i^1 = f_i^1 / F_s$$

return the first population chromosome, for which $p > P_i^1$, and in this way reproduce $2m$ paths $(2m < n)$.

3. **Crossover Operation** One-point crossover is used as a genetic operation to perform evaluation. It works as follows:

(a) Construct m pairs of paths from the generated $2m$ paths above:

$$R_{u_s}^1 \leftrightarrow R_{v_s}^1, (u_s, v_s = 1, 2, \ldots, 2m; s = 1, 2, \ldots, m)$$

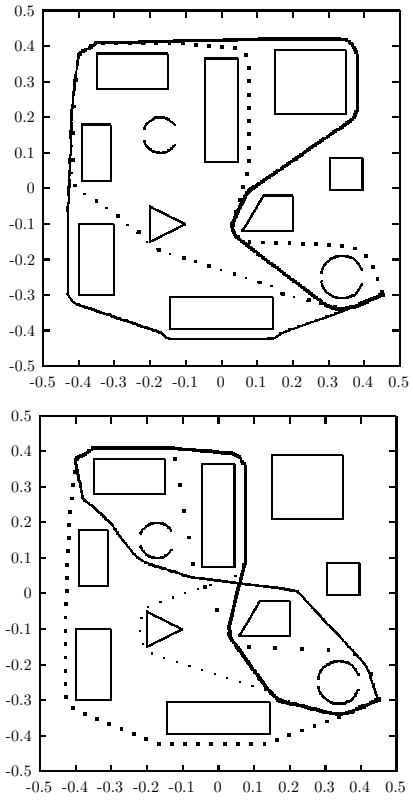

Figure 2: Example of initial population

Where

$$R^1_{u_s} = \{(x_1, y_1), (x^1_{u_s 2}, y^1_{u_s 2}), \ldots, (x^1_{u_s j}, y^1_{u_s j}), \ldots, (x_N, y_N)\},$$
$$R^1_{v_s} = \{(x_1, y_1), (x^1_{v_s 2}, y^1_{v_s 2}), \ldots, (x^1_{v_s j}, y^1_{v_s j}), \ldots, (x_N, y_N)\}$$

(b) Generate m numbers at random: $q_s (q_s < N)$.
(c) Cross the paths $R^1_{u_s} \leftrightarrow R^1_{v_s}$ at q_s-th point:

$$\{(x_1, y_1), (x^1_{u_s 2}, y^1_{u_s 2}), \ldots, (x^1_{u_s q_s}, y^1_{u_s q_s}), (x^1_{v_s q_s + 1}, y^1_{v_s q_s + 1}), \ldots, (x_N, y_N)\},$$
$$\{(x_1, y_1), (x^1_{v_s 2}, y^1_{v_s 2}), \ldots, (x^1_{v_s q_s}, y^1_{v_s q_s}), (x^1_{u_s q_s + 1}, y^1_{u_s q_s + 1}), \ldots, (x_N, y_N)\}$$

Let

$$r^2_u = \{(x_1, y_1), (x^1_{u_s 2}, y^1_{u_s 2}), \ldots, (x^1_{u_s q_s}, y^1_{u_s q_s}), (x^1_{v_s q_s + 1}, y^1_{v_s q_s + 1}), \ldots, (x_N, y_N)\},$$
$$r^2_v = \{(x_1, y_1), (x^1_{v_s 2}, y^1_{v_s 2}), \ldots, (x^1_{v_s q_s}, y^1_{v_s q_s}), (x^1_{u_s q_s + 1}, y^1_{u_s q_s + 1}), \ldots, (x_N, y_N)\}$$

As a result $2m$ paths are obtained: $r^2_k (k = 1, 2, \ldots, 2m)$.
(d) Generate $2m$ paths using $A+$ algorithm: $r^2_k \xrightarrow{A+} \bar{R}^2_k$.

4. **Survival strategy.** An elitist strategy is used to fix the potential best number loss by copying the best member of each generation into the succeeding generation, i.e. if R_{i_j} are $n-2m$ paths shortest in length in $\varphi^1 = \{R^1_1, R^1_2, \ldots, R^1_n\}$Cthen the collection of path of the second generation is:

$$\varphi^2 = \{R^2_1, R^2_2, \ldots, R^2_n\} = \{R^1_{i_1}, R^1_{i_2}, \ldots, R^1_{i_{n-2m}}, \bar{R}^2_1, \bar{R}^2_2, \ldots, \bar{R}^2_{2m}\}.$$

5. **Termination criteria.** The steps 2.–5. are repeated until the shortest path fulfils requirements or the shortest path of several generations in succession do not change.

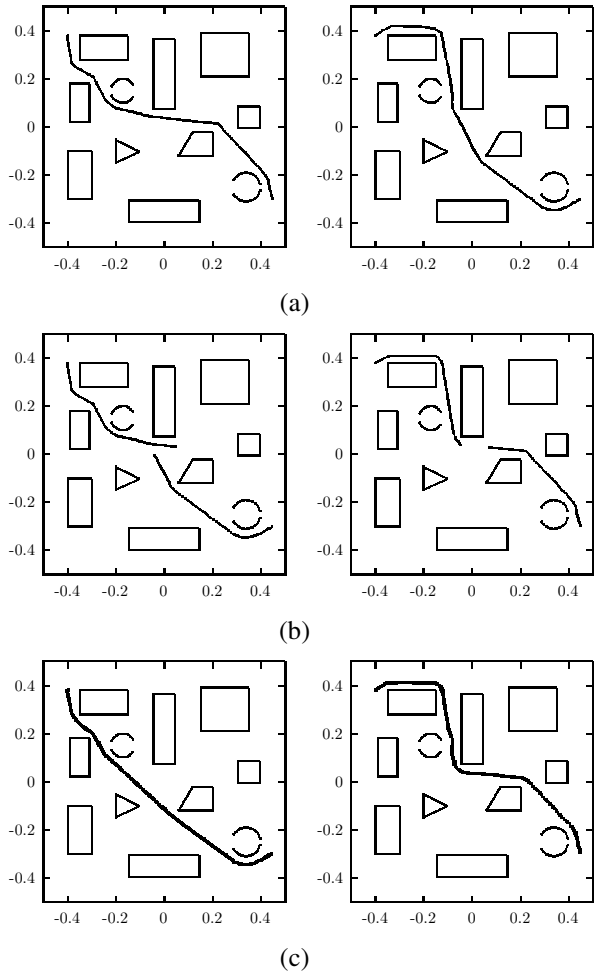

Figure 3: The optimal path (part of the 3-rd generation). (a) parents; (b) crossover operation, (c) the final paths

4. SIMULATION RESULTS

To show the effectiveness of the proposed in this paper algorithm, simulation results are given in this section. The initial population of ten paths is shown in Fig. 2. In the upper part the first five paths and in the lower part of the figure the remaining five paths are shown. The number of the path points is generated randomly in the range between 60 and 120 points. The lengths of the paths, the number of points in every path, parents selection results, crossover points, and the next generation members for ev-

#	L_i	pts	rep.	cros	q_s	r_i^2	R_i^2
R_1^1	1.319	86	2	R_1^1	25	r_1^2	$\bar{R}_1^2$
R_2^1	1.581	92	2	R_5^1		r_2^2	$\bar{R}_2^2$
R_3^1	1.319	92	0	R_2^1	70	r_3^2	$\bar{R}_3^2$
R_4^1	1.957	98	0	R_5^1		r_4^2	$\bar{R}_4^2$
R_5^1	1.490	98	3	R_1^1	59	r_5^2	$\bar{R}_5^2$
R_6^1	1.278	86	1	R_6^1		r_6^2	$\bar{R}_6^2$
R_7^1	1.175	86	0	R_5^1	56	r_7^2	$\bar{R}_7^2$
R_8^1	1.889	98	0	R_2^1		r_8^2	$\bar{R}_8^2$
R_9^1	1.489	110	0	line	R_7^1		R_9^2
R_{10}^1	1.590	80	0		R_6^1		R_{10}^2

Table 1: Parameters for the first generation. (L_i: length of the i-th path, 'pts': number of points in a path, 'rep': reproduction, 'cros': crossover pairs, q_s: crossover point)

#	L_i	pts	rep.	cros	q_s	r_i^2	R_i^2
R_1^2	1.743	98	0	R_2^2	38	r_1^3	$\bar{R}_1^3$
R_2^2	1.427	86	1	R_7^2		r_2^3	$\bar{R}_2^3$
R_3^2	1.693	98	0	R_5^2	45	r_3^3	$\bar{R}_3^3$
R_4^2	1.505	92	0	R_9^2		r_4^3	$\bar{R}_4^3$
R_5^2	1.396	86	1	R_6^2	29	r_5^3	$\bar{R}_5^3$
R_6^2	1.266	86	2	R_8^2		r_6^3	$\bar{R}_6^3$
R_7^2	1.319	92	1	R_6^2	33	r_7^3	$\bar{R}_7^3$
R_8^2	1.685	98	1	R_{10}^2		r_8^3	$\bar{R}_8^3$
R_9^2	1.176	86	1	line	R_9^2		R_9^3
R_{10}^2	1.279	86	1		R_6^2		R_{10}^3

Table 2: Parameters in the second generation

#	L_i	pts	rep.	cros	q_s	r_i^2	R_i^2
R_1^3	1.345	92	1	R_1^3	60	r_1^4	$\bar{R}_1^4$
R_2^3	1.338	86	0	R_9^3		r_2^4	$\bar{R}_2^4$
R_3^3	1.411	98	0	R_6^3	40	r_3^4	$\bar{R}_3^4$
R_4^3	1.368	86	0	R_5^3		r_4^4	$\bar{R}_4^4$
R_5^3	1.189	60	2	R_5^3	31	r_5^4	$\bar{R}_5^4$
R_6^3	1.341	86	1	R_8^3		r_6^4	$\bar{R}_6^4$
R_7^3	1.327	86	0	R_9^3	39	r_7^4	$\bar{R}_7^4$
R_8^3	1.311	86	1	R_8^3		r_8^4	$\bar{R}_8^4$
R_9^3	1.176	86	2	line	R_9^3		R_9^4
R_{10}^3	1.266	86	1		R_5^3		R_{10}^4

Table 3: Parameters in the third generation

ery generation are summarized in Tables 1–3. The pair matching can be found also in these tables.

The path lengths in every generation and the final results from the simulation are depicted in Table 4. The horizontal rows represent the individual path lengths and the last column gives the total sum of the lengths in every generation. It can be seen from the table, that the path length decreases gradually before the optimal solution is found. For the given example the optimal path was reached in four generations and we have observed from more than one thousand simulations for randomly generated environments with 5–30 obstacles, that in most of the cases the optimum is found in 3–5 generations. This may be because for every path generation we use the $A+$ algorithm which produces almost optimal paths. In other words, the GA "collects" pieces of "almost" optimal paths until the optimal one is found and it can be said that this is the key for the decreased number of generations.

The simulation finishes at the fourth generation, and the generated optimal path and its formation are shown in Fig. 3. The path $\bar{R}_5^4$ is the optimal one.

In the course of operation, the "immigrant" method is adopted: the longest path in parents path collection is substituted by a new path by generating it using $A+$ starting from straight line initial condition. In this way the speed of convergence is accelerated.

In addition, we observed that for given configuration of obstacles, in many cases, several suboptimal paths with fitness values very close to the optimal fitness (at most 0.05% difference) might exist (see Table 4) and they are quite different from each other. Insertion of a new obstacle creates a new optimal path which is based on the suboptimal paths and it generation is very fast. This might be because the GA maintains the diversity of a population to some extent. Of course, this is not always the case. If an environmental alternation drastically changes an optimal path, and the current population does not cover features of a new optimal path, finding a new optimal path is basically same as finding it from scratch.

The calculation speed remains as a problem which solution does not seem to be that easy. The $A+$ algorithm itself allows generation of semi-optimal paths in real-time, but the optimization process based on GA takes a lot of computational time. The reason is that in the population generation step the $A+$ algorithm is used at least eight times and this increases the calculation time drastically—e.g. when the total number of the vertices rises to 120 the computational time grows up to 220 s. Parallelizing the computations using computers connected via Ethernet decreases the speed greatly and this might be one of the possible solutions, but to reach reasonable computational time a network of eight computers is needed—a hardware cost which is hardly acceptable.

5. CONCLUSIONS

In this paper we have proposed a path planner for differential drive mobile robots moving in known environment. The proposed algorithm is based on GA and the resulting path is optimal in length. 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

R_1^1	R_2^1	R_3^1	R_4^1	R_5^1	R_6^1	R_7^1	R_8^1	R_9^1	R_{10}^1	L_1
1.319	1.581	1.320	1.957	1.491	1.279	1.176	1.889	1.489	1.590	15.092
R_1^2	R_2^2	R_3^2	R_4^2	R_5^2	R_6^2	R_7^2	R_8^2	R_9^2	R_{10}^2	L_2
1.744	1.427	1.693	1.506	1.397	1.266	1.316	1.685	1.176	1.279	14.492
R_1^3	R_2^3	R_3^3	R_4^3	R_5^3	R_6^3	R_7^3	R_8^3	R_9^3	R_{10}^3	L_3
1.345	1.337	1.411	1.368	1.189	1.341	1.327	1.311	1.176	1.266	13.072
R_1^4	R_2^4	R_3^4	R_4^{41}	R_5^4	R_6^4	R_7^4	R_8^4	R_9^4	R_{10}^4	L_4
1.408	1.175	1.401	1.183	**1.173**	1.269	1.190	1.291	1.176	1.189	12.456

Table 4: The path lengths in every generation (the shortest path in bold)

the required goal orientation θ is met.

The algorithm finds an optimal path without being trapped in local minima. It is applicable to free-flying robots and snakes, too. The algorithm is an extension of the one proposed by us in [14].

Because the GAs are adaptive they can work even if the environment is time-varying, one of future directions of our research is to adopt the proposed here algorithm for environments with sudden changes.

Since the $A+$ algorithm used for population generation can be directly adopted for 3D space, at present we are evaluating the GA in 3D environment and the results of that research will be reported in subsequent paper. Decreasing of the calculation time is another problem to be worked out—a matter which we are pursuing currently.

ACKNOWLEDGEMENTS

This research was partially sponsored by the Department of Electronic Engineering and the Graduate School of Okayama University of Science.

The authors are grateful to Professor Hiroyuki Narihisa from the Department of Information & Computer Engineering, Okayama University of Science for his helpful comments.

Finally, the authors would like to thank to the anonymous reviewers for the fruitful suggestions and remarks—this helped a lot in improving the quality of the manuscript.

A. THE LOCAL MINIMA PROBLEM

The local minima remains 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 [15]: (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. Therefore, second level is more realistic and is addressed by many researchers (see e.g. [15, 16] and the references there).

In the proposed in this paper algorithm, the local minima problem is addressed in a simple and efficient fashion:

1. After setting the goal position, the polygonal obstacles are scanned for concavities (fig. 4(b)).

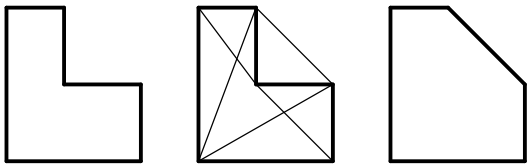

Figure 4: Concave obstacle: (a) the original shape, (b) scanning result, (c) the filled concavity

2. If the goal (start) lies inside a concavity, then a new goal (start) liyng outside the concavity is set, and the path between the original goal (start) and the new one is set as straight line. The new goal (start) is set at nearest the vertex measured from the start (goal).
3. Every detected concavity is temporarily filled, i.e. the obstacle shape is changed from concave to nonconcave one (see fig. 4(c)). After finishing the current task, the original shapes are retained so that the next task be planned correctly.

Note 2 *For the special case when the start and the goal are situated inside the same concavity, the concavity is left untouched, i.e. the filling is not performed.*

References

[1] D. E. Goldberg, **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison-Wesley, 1989.

[2] L. Davis, **Handbook of Genetic Algorithms**, Van Nostrand Reinhold, 1991.

[3] P. C. Chen and Y. K. Hwang, "SANDROS: A Dynamic Graph Search Algorithm for Motion Planning", **IEEE Trans. on Robot. and Automat.**, Vol. 14, No. 3, 1998, pp. 390–403.

[4] Y. K. Hwang and K. Ahuja, "Potential Field Approach to Path Planning", **IEEE Trans. Robot. Automat.**, Vol. 8, Sept. 1992, pp. 23–32.

[5] D. K. Pratihar, K. Deb, A. Ghosh, "Fuzzy-genetic Algorithms and Time-optimal Obstacle-free path Generation for

Mobile Robots", **Engineering Optimization**, Vol. 32, No. 1, 1999, pp. 117–142.

[6] Y. K. Hwang and K. Ahuja, "Gross Motion Planning—A Survey", **ACM Comput. Surveys**, Vol. 24, No.3, 1992, pp. 219–291.

[7] C. Hočaoglu and A. C. Sanderson, "Planning Multi-paths Using Specification in Genetic Algorithms", **Proc. 1997 IEEE Int'l Conf. of Evolutionary Computation**, Nagoya, Japan, May 1996, pp. 378–383.

[8] H.-S. Lin, J. Xiao, and Z. Michalewicz, "Evolutionary Algorithm for Path-planning in Mobile Robot Environment", **Proc. 1st IEEE Conf. on Evolutionary Computation**, Orlando, FL, Vol. 1, June 1994, pp 211–216.

[9] T. Shibata and T. Fukuda, "Intelligent motion Planning by Genetic Algorithm with Fuzzy Critic", **Proc. 8th IEEE Int'l Symp. on Intelligent Control**, Chicago, IL, Aug. 1993, pp. 565–570.

[10] X. Y. Xu and G. Vukovich, "Fuzzy Evolutionary Algorithms and Automatic Robot Trajectory Generation", **Proc. 1st IEEE Conf. on Evolutionary Computation**, Orland, FL, Vol. 2, June 1994, pp 595–600.

[11] K. Sugihara and J. Smith, "Genetic Algorithms for Adaptive Planning of Path and Trajectory of a Mobile Robot in 2D terrains", **IEICE Trans. Inf. & Syst.**, Vol. E82-D, No. 1, Jan. 1999, pp. 309–316.

[12] V. Kroumov, J. Yu and H. Narihisa, Path Planner for Differential Drive Mobile Robots Using Annealing Neural Network, **Transactions of the Society of Instrumentation and Control Engineers, Japan** (*submitted*).

[13] 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, 1979, pp. 560–570.

[14] V. Kroumov, J. Yu and H. Narihisa, "Path planning for unmanned vehicles", **Proc. of the 1999 International Conference on Mechatronic Technology (IMCT'99)**, Pusan, Korea, Oct. 1999, pp. 667–672.

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

[16] T. Lozano-Pèrez M. T. Mason and R. H. Taylor, "Automatic Synthesis of Fine Motion Strategies for Robots", **International Journal of Robotics Research**, Vol. 3, No. 1, 1984, pp. 3–24.

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

[18] C. W. Warren, "Global Path Planning Using Artificial Potential Fields", **in Proc. IEEE Int. Robot. Automat.**, 1989, pp. 316-321.

[19] D. E. Doditshek: "Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations", **Proc. IEEE ICRA**, 1987, pp. 1–6.