A Novel Implementation of G-Fuzzy Logic Controller Algorithm on Mobile Robot Motion Planning Problem

Seyyed Mohammad Reza Farshchi
Department of Artificial Intelligence
Islamic Azad University, Mashhad Branch, Iran
Tel: 98-091-5308-1125 E-mail: Shiveex@Gmail.Com

Seyyed Aliye Nezhad Hoseini & Fateme Mohammadi
Department of Image Science & Robotics
Sadjad University, Mashhad, Iran
Tel: 98-091-5157-5067 E-mail: Hosseininezhad@Sadjad.ac.ir

Abstract

This paper presents a new algorithm for global path planning to a goal for a mobile robot using GA and fuzzy Algorithms. A genetic algorithm is used to find the optimal path for a mobile robot to move in a dynamic environment expressed by a map with nodes and links. Locations of target and obstacles to find an optimal path are given in an environment that is a 2-D workplace. Each via point (landmark) in the net is a gene which is represented using binary code. The number of genes in one chromosome is function of the number of obstacles in the map. Therefore, we used a fixed length chromosome. The generated robot path is optimal in the sense of the shortest distance. The fitness function of genetic algorithm takes full consideration of three factors: the collision avoidance path, the shortest distance and smoothness of the path. The specific genetic operators are also selected to make the genetic algorithm more effective. The simulation results verify that the genetic algorithm is high effective under various complex dynamic alien environments.

Keywords: Genetic algorithm, Fuzzy logic, Obstacle avoidance, Trajectory planning

1. Introduction

The path planning problem of a mobile robot can be stated as: given (starting location, goal location, 2-D map of workplace including static obstacles), plan a collision-free path between two specified points in satisfying an optimization criterion with constraints (most commonly: shortest path). The path planning problem is computationally very expensive. Although a great deal of research has been performed to further a solution to this problem, conventional approaches tend

to be inflexible in response to:

- Different optimization goals and changes of goals
- Uncertainties in an environments and
- Different constraints on computational resources.

A review of the existing approaches for solving path planning problem is provided in (Acar & Choset, 2001) Many methods have been reported to generate an optimal path such: dynamic programming and distance transform methods. In the Dynamic programming method if the start point is P_s, goal point is P_g and sub-goal point is P_i, the path generation method is how to determine a sequence of sub-goals picking out the sub-goals from their set P_i (i=1,2,…,g-1). We must calculate all possible paths and select one has the smallest cost value. Very large computing power is required, especially in environments that have many sub-goals. In the distance transform method, the task of path planning is finding a path from the goal location back to the start location.

The path planning procedure covers the environment with a uniform grid and the propagate distances through free space from the goal cell. The distance wave front flows around obstacles and eventually through all free space in the environment. For any starting point within the environment representing the initial position of the mobile robot, the shortest path to the goal is traced by walking downhill via the steepest descent path. However, ambiguity of optimal paths exists where there exist two or more cells (nodes) to choose with the same least

distance transform. The above two methods require a very large computation power in case of environment that have large number of sub-goals and obstacles.

For most real world control and signal processing problems, the information concerning design, evaluation and realization, can be classified in two categories: numerical information obtained from sensor measurements and linguistic information obtained from human experts. Neural control is suited for using numerical data pairs (Agarwal & Flato, 2002) but the neural network controller only uses numerical data, and cannot utilize linguistic rules determined from expert drivers; whereas the fuzzy controller of (Akella & Peng, 2004) only uses linguistic rules, and cannot utilize sampled data. Therefore, both pure neural and fuzzy approaches are not sufficient for a successful control. In Aronov & Sharir, 1997) a general method for combining both numerical and linguistic information into a common framework has been proposed. Fuzzy logic can describe complex systems with linguistic rules (Amato & Bayazit, 1998). One of the most important applications of fuzzy logic is in controller design (Geraerts & Overmars, 2005) Fuzzy logic controllers (FLCS) convert the linguistic control strategy into an automatic control strategy. Experiences show that the fuzzy logic controller yields results sometimes superior to those obtained by conventional control algorithms. Successful design of a rule based fuzzy control system depends on several factors such as choice of the rule set, membership functions, inference mechanism, and the defuzzification strategy. Selection of an appropriate rule set is a computationally expensive combinatorial optimization problem. Sometimes for fuzzy controllers, rules are derived from human experts who have acquired their knowledge through experiences. However, experts may not always be available; even when available extraction of an appropriate set of rules from the experts may be tedious, time consuming, and process specific.

This paper focuses its attention to solving the problem of path tracking in the framework of genetic fuzzy logic. In fact, the adoption of such an approach leads to the use of fuzzy modules whose design is simple, rapid, inexpensive, and easily maintained because the rules can be linguistically interpreted by a human expert. For these reasons, over the latest decade, fuzzy logic and genetic algorithm has been widely used for mobile robots control to handle both general navigation problems, and specific motion control problems, such as, e.g., parking problem (Gray, 1998). Among the papers addressing path tracking control, (Hsu & Kindel, 2002) uses very few and simple fuzzy rules to adjust the gains of a linear controller, whereas (Pearl, 1984) develops a complex adaptive fuzzy system that learns from experiments while a human is driving the vehicle. The technique proposed in (Kavraki & Latombe, 1993) builds instead a number of rules to directly assign the steering and speed commands to the vehicle. Following the preliminary work in (LaValle & Kuffner, 2001) this paper combines a velocity controller with a simple fuzzy system that limits on-line the advancing speed of the vehicle so as to allow following an assigned path in compliance with the holding kinematic constraints. To accomplish this task, the linguistic variables curve and distance are introduced, that give information concerning the relevant geometric characteristics of the path ahead the robot. The implemented fuzzy rules are inspired by the behavior of a human driver who modulates the speed on the basis of the ahead knowledge of the path to follow, such as, e.g., slow down while approaching a narrow bend. The developed approach can be easily tailored to different vehicles of unicycle-like kinematics since it directly uses the velocity and acceleration limit values; more over, it can be used with paths assigned by any means (e.g., by a trace in the environment, by a sequence of set points, or by an analytical expression). Compared to traditional search and optimization methods, such as calculus-based and enumerative strategies (Nilsson, 1982) the genetic algorithm is a powerful search algorithm based on the mechanism of natural selection and uses operations of reproduction, crossover, and mutation on a population of strings at finding an optimum path in very large workspace.

A number of results in the literature show the application of GA to robotic path planning. Among the results, Khoogar and Parker (Overmars, 2005) considered the path planning problem in a plane using a planar robot. Ram et al. Applied GA to a mobile The obtained results demonstrate the effectiveness of the proposed technique. Robot navigation problem in a 2-D space with stationary obstacles. In (Nieuwen, 2004) Toogood et al. Developed a path finding method for stationary avoidance obstacles by applying GA to a 3-D robot manipulator. Most researches of robotic path planning in the literature focus on the static environment with known obstacles, which have become a more mature stage. However, researches of robotic path planning in changing circumstances are still hot spots of robotic path planning. In this paper, the novel genetic fuzzy algorithm will be applied to generate a dynamic path tracking for mobile robot in the unknown environment. The planning can achieve satisfactory results and speed of convergence. This shows that our genetic fuzzy algorithm has a strong environmental adaptability and be effective under various complex dynamic environments.

This paper is organized as follows. Section II presents a brief description of path planning problem. Section III indicates the resolutions based on GA. Section IV displays the results of simulations. Section V gives the concluding remarks.

2. Motion Planning

2.1 Formal Description of Platform

We construct the navigation system on a robot system. It is a four-wheeled robot, has two independently driven wheels, two omni-directional wheels, the hardware mainly consisted of controller module, interface module, ultrasonic sensors, the actuators, photo-electricity encoder and radio-communication module. We choose this robot, because the mobile robot is steered by the velocity difference between these two wheels and it can climb up and down over speed bumps and slopes. The outdoor mobile robot requires ability to navigate in a dynamic and potentially dangerous environment due to obstacles. We add a laser range finder to detect obstacles and build a map of the environment. A camera that can pan, tilt, zoom-in and zoom-out, is installed to send the image data to the control station. The picture of our robot are shown in figure 1.

2.2 Definition of dynamic motion planning problem

A natural extension to the basic path planning problem is planning in dynamic environments, in which besides stationary obstacles, also moving obstacles are present. It will be the main topic of this article. Just as with planning in static environments. In a dynamic environment for successful steering of a mobile robot to the destination, two important problems should be solved. One is real-time identification of the moving obstacles. The other requests that the shortest and safe path is generated dynamically (Ghrist, 2005). We assume the former issue may be resolved by the higher level visual processing system. This research in this paper focuses on the latter one. The aim is to put forward a dynamic path planning scheme in the unknown environment, and requests that path planning satisfy the following conditions (Vanden, 2004) as depicted in figure 2.

- The path should not occur collision with any obstacles
- The path should be as short as possible
- The path curves as smooth as possible

A path through a dynamic environment (in literature often called a trajectory) is defined as a continuous function $\pi : \tau \to C$, parametrized by time. The path planning problem in dynamic environments is to find a (collision-)free path between a given start configuration *s* and goal configuration *g*. Formulated in terms of the configuration time space $C\tau$, that is finding a path π such that $\pi(0) = s$ and $\pi(T) = g$, and $\forall(t \in [0,T] ::< \pi(t), t > C\tau_{free})$, where *T* is the duration (or arrival time) of the path.

Many of the methods discussed above can be applied without much difficulty on configuration time spaces. However, as time marches forward, additional constraints are put on the validity of paths through the configuration time space. Obviously, paths should be monotonically increasing in the time dimension in order to prevent unrealistic time travel. Note that the above definition of a path in a configuration time space already enforces this. This may not be enough, however, as it still allows paths demanding the robot to travel with nearly infinite velocity. Often, therefore, the velocity of the robot is bounded to a maximum (say vmax), which imposes a constraint on the slope of paths (with respect to the time dimension) through the configuration time space:

$$\forall(t_1, t_2 \in [0,T] :: t_1 < t_2 : d(\pi(t_1), \pi(t_2)) \le v_{\max}(t_2 - t_1))$$

where $d : C^2 \to \Re$ is a distance metric on the configuration space *C*.

3. Algorithm design

3.1 Path coding

In the genetic searching algorithm, the coding technique is a very important aspect. The length of binary strings from the parameter sets made up of the via-points of a path, as well as the size of the search space, determines the computational time for a given fitness function. In order to shorten the length of the binary strings, we adopt a unique coding method. In fact, a chromosome represents a candidate solution of the problem, i.e., a rule set for the fuzzy logic controller. This approach transforms the two-dimensional data to one-dimensional data as shown in figure 3. In figure 3 (a), the set of node points G_i is located at equal distance along the straight line from the initial position to the goal position. If the straight line is treated as *x* axis, then the set of the node points x_i is located at equal distances along the *x* axis. Then g_i becomes the search space for each via-point of the path and the via-point candidates. Are specified by one-dimensional data (Aronov, 1997). To further reduce the size of data in each search space and thereby accelerate the speed of genetic searching algorithm, the dynamic allocation of starting points at each via-point is used, which reduces the search space dramatically as shown in Fig. 3 (b). In figure 3 (b), the origin of the *i*th perpendicular search line is the knot point in the *x* axis. In figure 4, the coding structure using the reducing dimensions method is shown.

The number of alleles (containing integer values) in a chromosome is equal to the number of distinguished antecedent clauses in the rules. In our problem definition, there are two antecedent variables Φ and x and θ and one consequent variable θ. The number of term sets corresponding to Φ, x and θ are m, n and l, respectively. Therefore, there are m×n alleles in a chromosome, one for every possible combination of input fuzzy sets associated with the input variables Φ and x. Also the number of rules will be equal to $m \times n$. So a chromosome can be represented as:

$$Chromosome = [g_1, g_2, ..., g_{m \times n}]$$

The allele value at each location in a chromosome contains either the label of an output linguistic value to be used for a given rule or zero. In other words, if g_i represents the allele at position i, its nonzero value gives the consequent part (i.e., the label of the corresponding fuzzy set on the output variable) of the rule which corresponds to the ith location of the chromosome. A chromosome containing an allele value zero at the ith position (i.e., g_i=0) indicates that the rule set represented by the chromosome has not selected any rule with the ith antecedent clause.

3.2 Rule base representation in a chromosome

Suppose the number of term sets corresponding to Φ, x and θ are7,5 and 7, respectively. Then, the control rules are of the form:

$$if\ (\Phi\ is\ \{VVL, VL, L, Z, H, VVH\}\ and\ \{x\ is\ \{VL, L, Z, H, VH\}\ then$$
$$(\theta\ is\ \{VVL, VL, L, Z, H, VH, VVH\}$$

Therefore, there are 35 (=7×5) alleles in a chromosome. The allele value is 1 when θ is *VVL*, 2 for *VL*, and so on.

3.3 Fitness function

Selection of fitness function should take into account the security of the path, the shortest path, and the smoothness of the path. However, the security of the path is primary factor. In view of these, the fitness function is composed of three sub-functions:

1-Sub-function of path length

We could describe path length with integral. The sub-function is as follows:

$$fit1 = \sum_{i=1}^{n} (P_i, P_{i+1}) \tag{1}$$

Where (P_i, P_{i+1}) is the distance between $P_i = P(x_i, y_i)$ and $P_{i+1} = P(x_{i+1}, y_{i+1})$.

2-Sub-function of path security

The condition of collision avoidance is stated as follows: if the distance between every via-point of the path and the obstacles is expressed as $|P_i S_j|$ for any path, then we get:

$$fit2 = \begin{cases} \min \sum_{i=1}^{n} \sum_{j=1}^{m} (|P_i S_j|) / S_d & if\ \min(|P_i S_j|) > S_d \\ 0 & otherwise \end{cases} \tag{2}$$

Where $P_i = P(x_i, y_i)$ is the position of every via points of the path, $S_j = S(x_j, y_j)$ is the position of each obstacle, i=1,2,...,n, j=1,2,...,m, s_d=r_0+r_k and r_0 is the radius of mobile robot, r_k is the radius of the number k obstacle.

3-Sub-function of smoothness

In this section, the smoothness of path planning is depicted. The path will be considered to be smooth, when mobile robot moves from the present point $P_i = P(x_i, y_i)$ to the next point $P_i = P(x_{i+1}, y_{i+1})$ only forward not circuitously. The sub-function is expressed as:

$$fit3 = \frac{1}{\sum_{1}^{m-1} \left(\frac{(y_i - y_{i+1})}{(x_i - x_{i+1})} - \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)} \right)} \tag{3}$$

4. Genetic algorithm implementation

For generating fuzzy rules bases, we use the self organized genetic algorithm-based rule generation (SOGARG) method (Geraerts, 2005). The SOGARG method consists of three stages. The first stage attempts to extract a rule set with a view to enabling the system to be controlled in the vicinity of the set point. In the second stage, this rule set is enhanced and modified to account for the entire input space. Finally, the last stage fine tunes the rule set through modification of existing rules and/or deletion of redundant rules and/or addition of new rules. This method has used different objective functions and different mutation schemes for different stages which are consistent with the objectives of different stages.

The initial point somewhere on a grid (xy -plane) at a given angle Φ. The objective is to navigate the robot from initial position (x_i, Φ_i) to the desired location.

4.1 Generated rule set

For generating rule base with three fuzzy sets that we, we use membership functions shown in figure 5-7 respectively for input variables Φ , x and output variable θ . In our rule set, each of the input and output linguistic variables has three linguistic value or fuzzy sets: L, Z and H. We use overlapped isosceles triangles as membership function. All membership functions have equal base lengths. This is possibly the most natural and unbiased choice for the membership functions. We use seven fuzzy subsets VVL, VL, L, Z, H, VH and VVH. The membership functions are overlapped isosceles triangles but with unequal base lengths as shown in correspond figure.

4.2 Genetic operation

4.2.1 Initialization of population operator

Initial population, as the beginning of optimization, may be produced by algorithm itself. N refers to the scale of population, namely random produce path $R_j = \{j = 1,2,N\}$, the individual can be chosen by being in proportion to fitness value. The operation is described as following:

- To calculate the individual adaptive value.
- To calculate the probability of choosing copy.
- To calculate the expected replication number.
- To round the expected replication number to the nearest integer and then get the actual replication number.

4.2.2 Crossover operator

The crossover operator combines parts of two different chromosomes to create two new ones. There are many ways to do the crossover operation, but for the chromosome formed by a node sequence in the path, only single-point and multi-point cross have factual significance, and there is no difference in essence between them. So, the single-point cross means is adopted in this paper. Two individuals are randomly selected, and then choose a crossing point to cross according to a certain crossover probability; replace the individual of the father with individual of the offspring after cross, a new population would be produced.

4.2.3 Mutation Operator

Mutation is to randomly choose a node and replace it with a node that is not included in the path. Mutation is served as a key role to diversify the solution population. Therefore, it is not necessary that a solution is better after it is mutated. Mutation operation is illustrated in figure 8. The mutation probability of each element in the population is P_m (generally within 0.001~0.3). In this paper, three kinds of way are adopted as the way of mutation. They are increasing a point, decreasing a point and removing a point. Mutation and crossover are co-used abstemiously, aiming at avoiding over-mature to lose vital genetic information. But mutation often keeps lower probability for fear that should destroy the individual fabric of next generation.

4.2.4 Smooth Operator

The smooth operators choose feasible path of greater corner, and then randomly inserted three new path nodes, if these three new nodes constitute a viable segment of the line, retain new nodes, and delete the original nodes. Smooth operation is shown in figure 9.

5. Simulation experiment

The algorithm has been implemented for both free-flying and articulated robots with six degrees of free domain in a desired workspace. We performed experiments in different environments and the results indicate that the

method achieves interactive performance. In this section we describe a number of experiments in detail, and compare our method with the straightforward approach as well.

5.1 A Preliminary experiment

Our method was tested in the building floor scene in left side of figure 10. The scene has dimensions of 8 (length) by 5 (width) by 2 (height) units of length. In the scene two moving obstacles A and B are moving. A moves along an H-shaped trajectory and B along a rectangular trajectory. The velocity of both moving obstacles is 1 unit of length per unit of time. The positions of the moving obstacles at the start time are shown in the figure. The motions of both obstacles are cyclic, i.e. They move endlessly.

As the robot we used a free-flying table, which has a radius of 0.5 units of length. It has to move through some narrow passages (having a width of 0.6 units of length) from s in the lower-right room to *g* in the left room. The distance between two configurations of the robot is measured as the euclidean distance plus the amount of rotation times the robot's radius. Its velocity under this distance measure is bounded by 1 unit of length per unit of time.

The construction of the road-map stopped when a predefined set of query configurations was connected by the road-map. The road-map is shown in right side of figure 9 and consists of 198 vertices and 478 edges.

The shortest path in the road-map between the start and the goal configuration is 15.82 units of length. So if no moving obstacles would be present 15.82 units of time are necessary to complete the path. However, moving obstacle A will move through the passageway in the opposite direction of the robot, so the robot must make a detour to avoid this obstacle.

This example problem may look simple, but it certainly is not. The stationary obstacles in the scene strongly confine the maneuverability of the robot, and the environment contains many narrow passages. Moreover, the moving obstacles 'sweep clean' the passageways in the environment, so the robot really has to 'flee' into other rooms to avoid collisions.

The running time of the algorithms directly dependent on the choice of the value of the principal time step Δt. In this experiment we chose Δt to be 0.07, so that the collision checking resolution with respect to the moving obstacles is exactly corresponding to the resolution in which the road-map was collision checked with respect to the stationary obstacles.

The algorithm was run on a 3GHz Pentium IV with 1 GByte of memory. For the problem described above, it returned a path in only 0.46 seconds of computation time. The path takes 35.14 units of time to traverse, and indeed the robot must make quite a detour to avoid the moving obstacles (see figure 11). It more than once 'flees' into a room at the side of the passageway. Obviously, the path is collision-free with respect to both the static and the moving obstacles.

The computed path is near-time-optimal over the road-map. Yet, it takes 35.14 units of time to traverse, while this would be 15.82 units of time without obstacles. We call these numbers the path length and the road-map distance respectively, and the ratio between them the delay-factor. It gives an indication of the 'difficulty' of the problem. For this experiment the delay-factor is 35.14 / 15.82 = 2.22.

5.2 Varying the quantities

In this section we explore the effect on the performance of our method of gradually increasing major quantities, among which the delay-factor of the problem and the size of the road-map. For these experiments, we use the environment and the quantities of the above experiment. Only the quantity under consideration is varied.

The difficulty of a problem is always hard to measure, let alone varying it *ceteris paribus*, i.e. Without changing other quantities. We use the delay-factor as a measure for the difficulty of a problem. In the above experiment the delay-factor was 2.22.

We vary the delay-factor over the experiments by tuning the velocity and the behavior of the moving obstacles. The other quantities are kept equal; we use the same road-map over the experiments and do not change the value of Δt.

The performance of our method not only depends on the delay-factor, but also on whether an obstacle interferes with the robot in the beginning of the path, when little probes are active, or near the goal configuration, when many probes are active. To cover a large range of possibilities, we performed 55 experiments with different behavior of the moving obstacles.

One would expect that the running time grows exponentially with the delay-factor, because the more time it

takes to reach the goal, the wider the interval tree grows. The A* nature of our algorithm though moderates this effect as it focuses the search toward the goal.

Also, for larger delay-factors the space gets more confined. This means that there is less maneuverability for the robot, which has a moderating effect on the width of the interval tree (and hence the running time) as well.

Figure 12 gives the result of the experiments. The scatter-plot indicates that the running time actually grows more or less linear with the delay-factor. It turned out to be quite hard to find problems with a delay factor larger than 6 for which a solution still exists. The few problems we did find however, confirm the apparent linear relationship. Note that when the delay-factor is 1, the running time is nearly zero.

The size of the road-map influences the performance of our method. The more edges in the road-map, the more probes need to be sent and, hence, the interval tree grows wider. On the other hand, more possibilities become available in the road-map when the road-map gets larger. This can also have a positive effect on the performance, as this may lower the delay-factor of the problem.

To avoid strong deviations in the running times of the experiments as a result of the randomness involved in creating road-maps, we extend road-maps over the experiments. To be precise, in each experiment we add 100 vertices to the road-map of the previous experiment.

The number of edges grows more or less proportionally. The results are shown in figure 13 and table 1. The figure shows a linear relation between the road-map size and the running time. We see that for a too small road-map the running time actually gets higher. This is because the possibilities of the robot are too much restricted, which results in a high delay-factor of the problem (see table 1). The minimum running time is found for a road-map with 200 vertices (and 558 edges). Note that the resulting path is not substantially longer than the paths found in larger road-maps (see column ‘path length’).

Other quantities do not directly influence the performance of our method. The number of degrees of freedom of the robot, for instance, or the size of the environment, only influence the performance if they make the problem more difficult, or if they make the environment require a larger road-map. Also, some quantities, such as the number of moving obstacles or the complexity of the moving obstacles, cause collision-checks to become more expensive. They may as such only have an indirect relation to the performance of our method.

5.3 Visual simulation

In this simulation environment, selection adopts roulette wheel selection to save the optimal individuals, crossover adopts single point random crossovers and mutation employs single point random mutation. Within a square area of 5m×5m, gray circle represents the mobile robot, while static and dynamic obstacles are exemplified with other color circles. Crossover probability P_c is 0.62, mutation probability P_m is 0.1, smooth probability is set to 0.2, population size generation is 50, N (the number of evolution era) = 2000, the simulation results are shown in figure 14. In order to reflect the capability of genetic fuzzy system in complex dynamic environments, the numbers of static obstacles and dynamic obstacles vary.

Figure display that the dynamic path generation and collision avoidance for moving obstacles have been accomplished under such dynamic environment, where there are four or six static obstacles respectively and one moving obstacle. Figure 14 illustrate that the dynamic path generation and collision avoidance for moving obstacles have been accomplished under such dynamic environment, where there are five static obstacles and two moving obstacles.

5. Conclusion

This paper presents a dynamic path planning method based on genetic fuzzy algorithm for mobile robot under an alien environment. The real coding, fitness function and specific genetic operators are designed to accelerate the convergence of the algorithm and improve the accuracy of operation. The genetic algorithm used in this paper can achieve satisfying path planning results under an alien dynamic environment. The simulation results show that the genetic algorithm has strong adaptability of dynamic and alien environments and verify the proposed method is high effective.

References

Acar, E., Choset. H., & Atkar. D. (2001). Complete sensor-based coveragewith extended-range detectors: A hierarchical decomposition in terms of critical points and Voronoi diagrams. *In Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, P.1305–1311.

Agarwal, P., Flato, E., & Halperin, D. (2002). Polygon decomposition for efficient construction of Minkowski sums. *Computational Geometry: Theory and Applications*, P.39–61.

Akella, S. & Peng. J. (2004). Time-scaled coordination of multiple manipulators. *In Proc. IEEE Int. Conf. on Robotics and Automation*, P.3337–3344.

Amato, N., Bayazit, O. L. Dale, Jones, C., & Vallejo, D. (1998). OBPRM: An obstacle-based PRM for 3D workspaces. *In Proc. Workshop on Algorithmic Foundations of Robotics*, P.155–168.

Aronov, B., & Sharir, M. (1997). On translational motion planning of a convex polyhedron in 3-space. *SIAM Journal on Computing*, P.1785–1803.

Geraerts, R., & Overmars, M. (2005). Sampling and node adding in probabilistic road-map planners. *Journal of Robotics and Autonomous Systems*, P.406–412.

Ghrist, R., O'Kane, J., & LaValle, S. (2005). Computing pareto optimal coordinations on road-maps. *International Journal of Robotics Research*, P.997–1010.

Gray, A., (1997). *Modern Differential Geometry of Curves and Surfaces with Mathematica*, P. 40–42. CRC Press, Boca Raton, FL, 2nd edition.

Halperin, D., & Sharir, M. (1996). A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment. *Discrete Computational Geometry*, P.121–134.

Hart, P., Nilsson, N., & Rafael, B. (1998). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, P.100–107.

Hsu, D., Kindel, R., Latombe, J., & Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, P.233–255.

Kavraki, R., Latombe, J., Motwani, R., & Raghavan, P. (1995). Randomized preprocessing of configuration space for fast path planning. *In Proc. ACMSymp. on Theory of Computing*, P.353–362.

LaValle, S., & Kuffner, J. (2001). Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions*, P. 293–308.

LaValle, S., Branicky, M., & Lindemann, S. (2004). On the relationship between classical grid search and probabilistic road-maps. *International Journal of Robotics Research*, P.673–692.

Lin, M., & Manocha, D. (2004). Collision and proximity queries. *Handbook of Discrete and Computational Geometry*. CRCPress, Boca Raton, FL, 2nd edition.

Lozano-Pérez, T., & Wesley, M. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, P.560–570.

Nieuwen huisen, D., & Overmars, M. (2004) Useful cycles in probabilistic road-map graphs. *In Proc. IEEE Int. Conf. on Robotics and Automation*, P.446–452.

Nilsson, N. (1980). *Principles of Artificial Intelligence*. Tioga Publishing Company.

Overmars, M. (2005). Path planning for games. *In Proc. Int. Game Design and Technology Workshop*, P.29–33.

Pearl, J. (1984). *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley.

Roslina, I., Rasimah, C. M. Y., & Azizah, J. (2008). Computer Games Playing Activities: Habits of University Teknologi Malaysia Student. *Paper presented at the International Conference on IT and Multimedia (ICIMU), Bangi, Selangor*.

Rubijesmin, A. L. (2007). Understanding Malaysian students as gamers: Experience. Paper presented at *the Proceedings of the 2nd International Conference on Digital interactive Media in Entertainment and Arts Perth, Australia.*

Siméon, T., Leroy, S., & Laumond, J. (2002). Path coordination for multiple mobile robots: a resolution complete algorithm. *IEEE Transactions on Robotics and Automation*, P.42–49.

The CGAL project homepage. http://www.cgal.org/.

The CORE library homepage. http://www.cs.nyu.edu/exact/core/.

Vanden Berg & Overmars, M. (2004). Road-map-based motion planning in dynamic environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, P.1598, 1605.

Vanden Berg, Nieuwen huisen, D., Jaillet, D., & Overmars, M. (2005). Creating robust road-maps formation planning in changing environments. *In Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, P.2415–2421.

Table 1. Results for various road-map sizes

#Vertices	#Edges	Road-map distance	Path length	Delay factor	Running Time
100	246	17.15	43.40	2.53	0.67
200	558	16.52	25.58	1.55	0.33
300	944	16.52	25.48	1.54	0.62
400	1376	16.52	25.48	1.54	1.16
500	1806	14.70	25.48	1.73	1.88
600	2236	14.70	25.48	1.73	2.31
700	2698	14.70	25.48	1.73	2.48
800	3156	14.70	25.48	1.73	2.85
900	3668	14.21	24.64	1.73	3.17
1000	4158	14.21	24.64	1.73	3.90

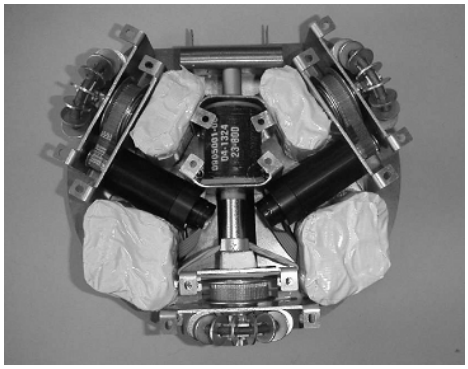

Figure 1. Picture of our robot

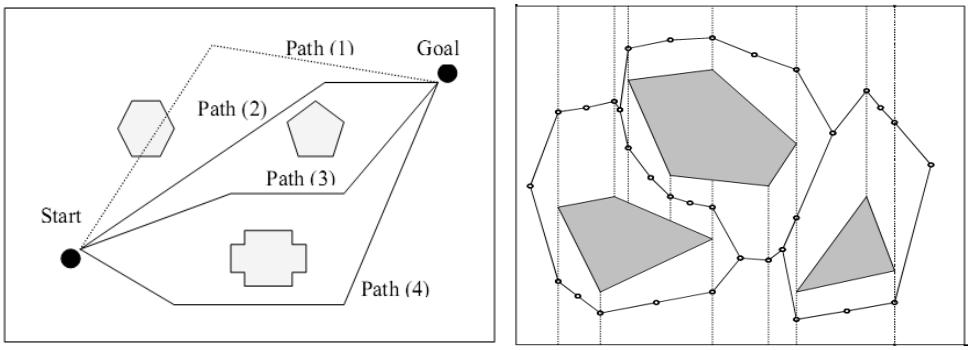

Figure 2. (a) An example path that is collision-free with respect to the stationary obstacles. (b) The discretized "road-map-time" space, which is a complex of two-dimensional grids along each of the edges, forms the search space of the problem.

 ISSN 1913-8989 E-ISSN 1913-8997

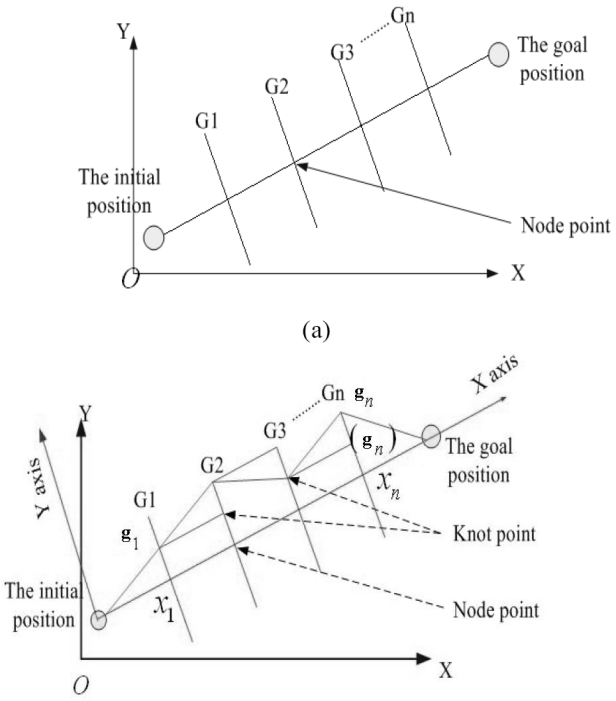

Figure 3. Conversion of search space

g_1	g_2	g_3	…	g_n

Figure 4. Coding structure

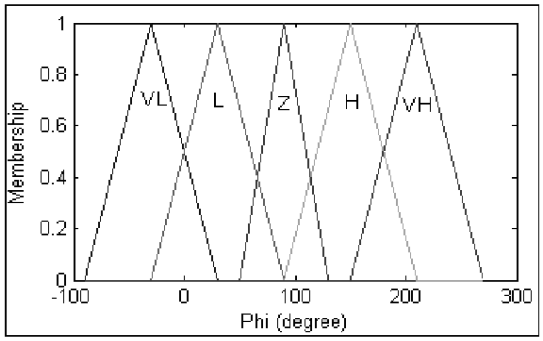

Figure 5. Membership function with 7 fuzzy sets

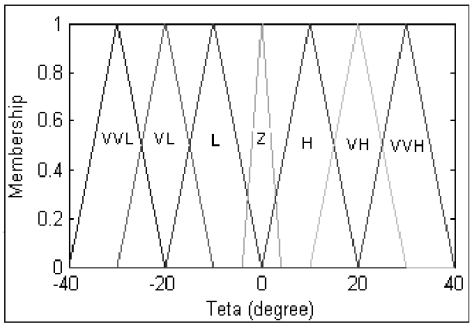

Figure 6. Membership function with seven fuzzy sets with the same base lengths

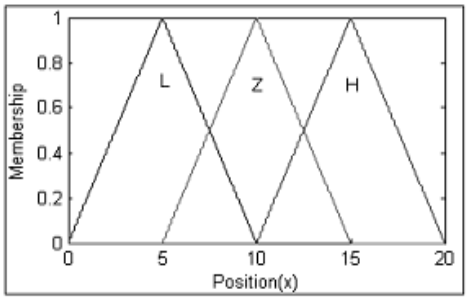

Figure 7. Membership function with three fuzzy sets

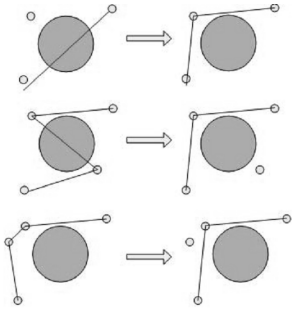

Figure 8. Mutation operation: increasing a point (higher), decreasing a point (medium), and removing a point (lower)

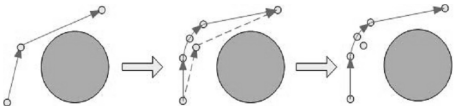

Figure 9. The smooth operation

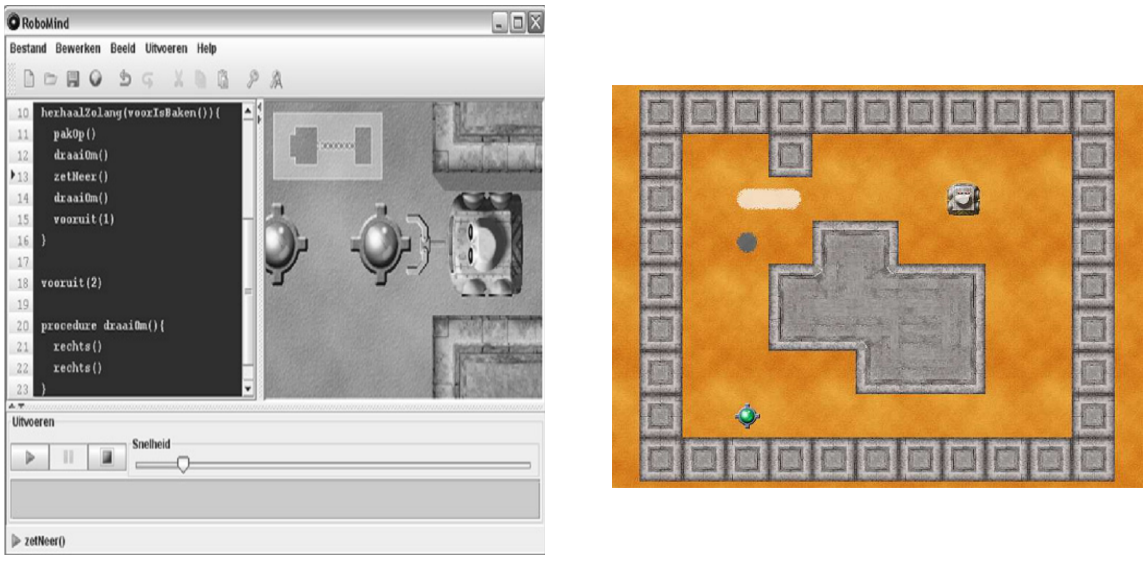

Figure 10. (a) A dynamic environment in which a table has to move from s to g avoiding the moving obstacles A and B (cylinders). The cylinders move cyclically along the dotted lines. (b) A road-map that is collision-free with respect to the stationary obstacles. The rotational degrees of freedom are not shown in the road-map

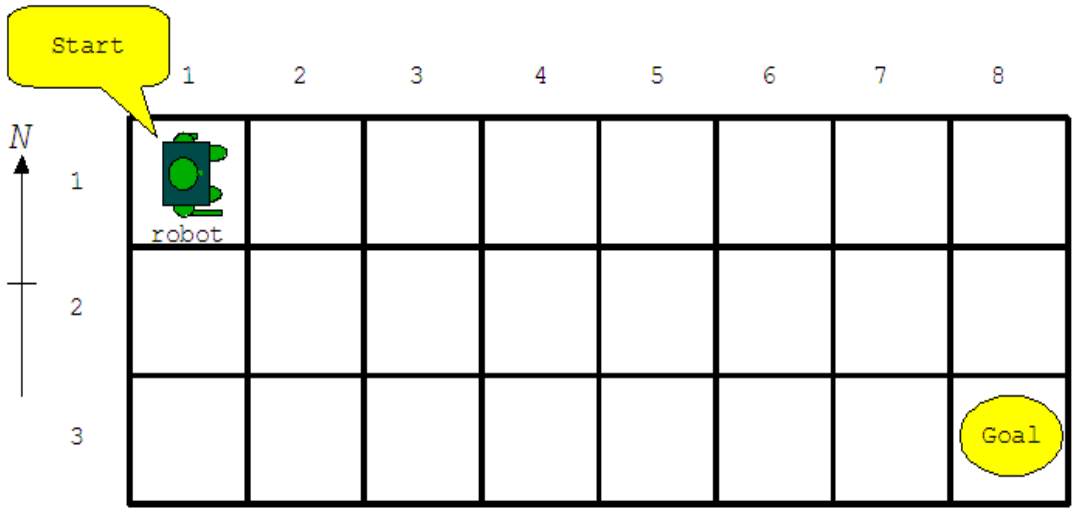

Figure 11. Pictures from the path.

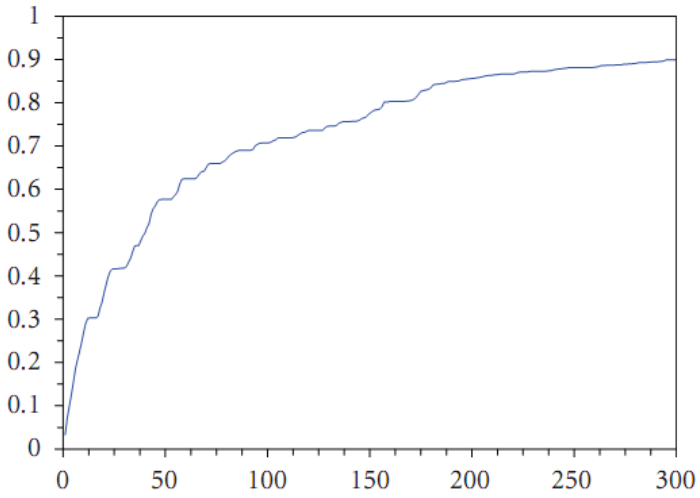

Figure 12. A scatter-plot of 55 experiments with different delay-factors gives

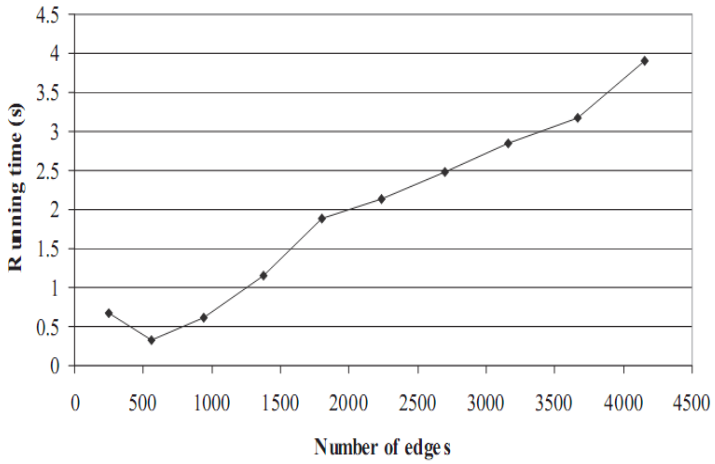

Figure 13. The running time for road-maps of various sizes

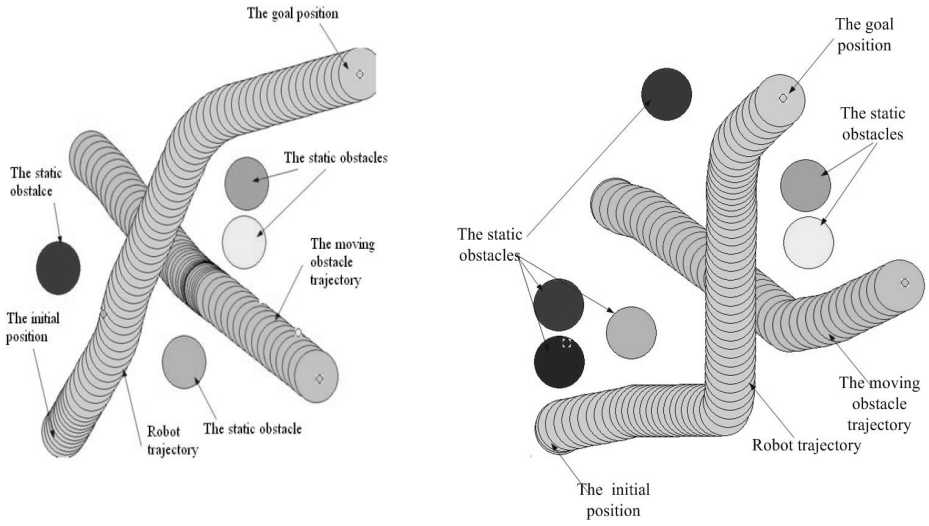

Figure 14. (left) Path generation while avoiding four static obstacles and one dynamic obstacle. (right) Path generation while avoiding six static obstacle