Real Path Planning based on Genetic Algorithm and Voronoi Diagrams

Facundo Benavides* Gonzalo Tejera* Martín Pedemonte* Serrana Casella*
fbenavid@fing.edu.uy gtejera@fing.edu.uy mpedemon@fing.edu.uy scasella@fing.edu.uy
*Instituto de Computación
Facultad de Ingeniería - UdelaR
Montevideo, Uruguay

***Abstract*—In the context of Mobile Robotics, the efficient resolution of the Path Planning problem is a key task. The model of the environment and the search algorithm are basic issues in the resolution of the problem. This paper highlights the main features of Path Planning proposal for mobile robots in static environments. In our proposal, the path planning is based on Voronoi diagrams, where obstacles in the environment are considered as the generating points of the diagram, and a genetic algorithm is used to find a path without collisions from the robot initial to target position. This work combines some ideas presented by *Roque* and *Doering*, who use Voronoi diagrams for modelling the environment, and other ideas presented by *Zhang et al*. who adopt a genetic algorithm for computing paths on a regular grid based environment, considering certain quality attributes. The main results were probed both in simulated and real environments.**

***Index Terms*—Autonomous Mobile Robots, Path Planning, Voronoi diagrams, Genetic Algorithms.**

I. Introduction

The autonomous mobile robotics has been developed primarily to enable high-level tasks to be performed by machines without human control. Therefore, robots should be able to move properly in the real world, and consequently, the path planning becomes one of the most important problems to be solved in the design of autonomous mobile robots. It usually involves modeling of geometric areas, incomplete knowledge, treatment of uncertainty in measurements, avoiding moving obstacles, unpredictability and kinematics of bodies, handling multiple robots or goals. However, the effective resolution of this problem can result in saving of working time, reduced concern about the robot and investments of funds and often is usually the basis for the development of other skills. In this context and for these reasons, the path planning is a very active area of research where an impressive range of approaches have been proposed.

Generally, this task consist in finding optimal or quasi-optimal paths from an initial state (origin) and a final state (target) following a certain criteria for evaluating the optimality of the paths. The most frequently used criteria are traveled distance, energy saving, smoothness and path's safety.

In this regard, and in order to provide robust, flexible, tolerant to changes in the environment, reliable and computationally efficient solutions, many different approaches have been proposed to address this problem, including neural networks [1], reinforcement learning [2], bioinspired methods [3], [4], graphs [5], [6], [7], evolutionary algorithms [8], [9], [10], [11], [12], [13], [14], among others [15], [16], [17].

This paper presents a proposal for path planning in static environments. Our approach is based on Voronoi diagrams (*VD*), where obstacles in the environment are considered as the generating points of the diagram, and a genetic algorithm is used to find a collision-free path between the initial and target position of the robot. This work combines ideas presented by *Roque and Doering* [6], where *VD* are used to represent the environment, running a shortest path algorithm on the Voronoi points, and the ideas presented by *Zhang et al*. [14], which uses a genetic algorithm for computing paths on a regular grid based environment, considering certain quality attributes (length, safety and smoothness).

The main motivation for this combination is to provide, to the path planning algorithm based on genetic algorithm, an environment model that is computationally more efficient and better adapted to the context of mobile robots than regular grids. The results obtained in a large set of simulated experiments show that our proposal improves both the quality as well as the execution time. Finally, the best instances of paths were executed in a real environment making our approach promising for practical applicability in mobile robotic systems.

The paper is organized as follows. Section II present a briefly introduction to Voronoi diagrams and evolutionary algorithms, respectively. A description of the approach to model the environment and the application of *GA* is carried out in Section III while Section IV describes the case studies, presenting a comparative analysis among the proposal and previous. Finally, we close with a discussion and some directions in Section V.

II. Background

A. Voronoi Diagrams

Voronoi diagrams are one of the main structures in the computational geometry area, commonly used as visibility graphs and for finding networks collision-free paths, they are also one of the most common techniques for building trajectory maps [5].

Given S a set of geometric entities in $\mathcal{R}^d$ and $\|.\|$ a distance metric, the Voronoi diagram for S is the partition of $\mathcal{R}^d$ in the maximum number of regions, such that the points belonging to each Voronoi region are closer to a single entity of S than to the other ones [7].

VD have been usually applied to mobile robotics considering the $\mathcal{R}^2$ space, the Euclidean distance between points and the nearest neighbour principle. Given $P = \{p_1, p_2, ..., p_n\}$ a set of points not aligned in the plane and $d(p_i, p_j)$ the Euclidean distance between points p_i and p_j, the Voronoi region $R(p_i)$ generated by the point p_i is defined by Equation 1:

$$R(p_i) = \left\{p \in \mathcal{R}^2; d(p, p_i) \leq d(p, p_j), \forall j \neq i\right\} \quad (1)$$

The points p_i are called Voronoi generators while the set of all Voronoi regions $R(p_1), R(p_2), ..., R(p_n)$, is called Voronoi diagram of *P*. In addition, a frontier between two regions and an intersection of three or more edges are called Voronoi edge and Voronoi vertex, respectively. Figure 1 presents a diagram where the regions are shown in shades of gray, while the edges and the generating points are shown in black.

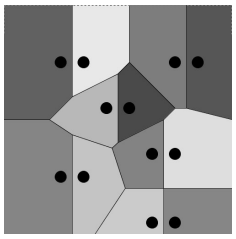


Figure 1: Voronoi diagram.

Voronoi diagrams have two interesting properties in the context of path planning with mobile robots:

1) every Voronoi edge belongs to the bisector of the line segment determined by the two generating points of the regions that determine the edge.
2) every Voronoi vertex is located exactly at the circumcenter of the polygon defined by the generating points of the regions that determine the vertex.

These properties make the Voronoi diagrams a good alternative for computing maps of paths with maximum security. Another advantage it is that these diagrams could be easily generalized, defining non-points generators with geometric shapes representative of real world obstacles. Therefore, it would be possible to define a generator with a line segment to represent a wall or with a polygon to represent the orthogonal projection onto the plane of another robot, and other types of obstacles that often occur in real environments.

B. Evolutionary Algorithms

Evolutionary algorithms (*EAs*) are stochastic search techniques inspired in the natural process evolution of species guided by the principle of survival of the fittest [18], [19]. *EAs* iteratively evolve a population of individuals representing candidate solutions of the optimization problem. The evolution process involves the probabilistic application of operators to find better solutions.

The initial population is usually randomly generated. Each iteration of the algorithm is divided in four stages. In the first stage, every individual in the population is associated with a fitness value that measures the quality of the candidate solution. Later, the solutions are selected from the population based on their fitness value, usually giving higher priority to higher quality solutions. In the third stage, new solutions are constructed applying evolutionary operators to the selected solutions. Typically, the evolutionary operators used are the crossover (recombination of parts of two individuals) and the mutation (random changes in a single individual). Finally, in the fourth stage, the new population is created, by replacing the worse adapted individuals with solutions generated in the iteration. This process is repeated until the population converges to good quality solutions, where the best individual has a good chance of representing an optimal solution or near-optimal.

One of the most classical and widely known types of *EA* are: Genetic Algorithms (*GAs*).

III. Our proposal

In this work we combine ideas from two different previous works. On the one hand, *Roque and Doering* [6] used *VD* to represent the environment, considering a single attribute for path planning (distance traveled using a shortest path algorithm), because security is guaranteed by the properties of the geometric construction used. On the other hand, *Zhang et al.* [14] used a genetic algorithm on a regular grid based environment model that considered several quality attributes (distance, security and smoothness). Our approach combines both ideas considering quality attributes not taken into account by *Roque and Doering*, and using a better representation of the environment that the one used by *Zhang et al.*

The rest of this section describes the major aspects of the environment representation and the genetic algorithm used to find the paths.

A. World modeling

The world model consists in a regular grid in which each cell can be a free zone or represent an obstacle. A Voronoi diagram is generated from the set of points located in the center of each obstacle. Thus, the center of each cell that represents a barrier will also be a Voronoi generator. Figure 2 shows a typical scenario modeled according to this representation.

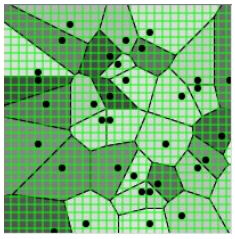


Figure 2: A world model example.

The relationship between the variables that determine the number of rows, columns and the side of each cell reproduces the relationship between the size of the robot, obstacles and the total area where the robot can move. So, the representation and the algorithms consider the robot as a single particle (point). To some extent, this relationship can be established as a simplified

implementation of the proposal of Wein et al. [7], regarding the model of the environment and more specifically, how to consider the robot as a point without loss of proportionality between the areas occupied by different objects and the total area of the environment.

B. Genetic Algorithm for path planning

The objective pursued with the implementation of the *GA* is to find a path free from obstacles, specially considering the following attributes: length, smoothness and safety. To this end, the *GA* should be able to properly explore the search domain in the early stages, considering the restrictions imposed by the environment and avoiding premature convergence. In the final stages, the *GA* can benefit from the knowledge acquired during the search process in order to reach a final result as close as possible to the optimum.

1) GA encoding: Each individual represents a path consisting in a variable length sequence of free cells, which begins with the cell where the robot is originally located and ends with the cell defined as the path destination.

It is important to note that this representation, as in [14], is valid for individuals that represent both feasible and unfeasible (no obstacle-free) paths. This aspect is relevant because the fitness function has to be able to compute fitness for unfeasible solutions. Although it may involve a higher computational effort, working on genetically richer populations can contribute to not bias the search process in early stages. The unfeasible paths are composed of unfeasible and feasible sections. The latter are important and should not be lost because they can, at least potentially, contain valuable information that could be helpful for building good solutions. In summary, incorporating the management of unfeasible solutions in the *GA* tries to avoid falling into local optima at the expense of increasing the effort of the search process.

2) Fitness function: The fitness function measures the quality of a path, ensuring that unfeasible paths have always a lower value than a feasible path.

Feasible paths: The Equations [2, 3, 4 and 5] presents the fitness function for feasible paths considering the length, safety and smoothness of the path. In this case, *maxPathLength*, *minPathLength*, *maxPathSafety* and *avePathSmooth* values are considered as upper and lower bounds and average for length, safety and smoothness, respectively.

$$d_{term} = \frac{maxPathLength - pathLength}{maxPathLength - minPathLength} \tag{2}$$

$$sa_{term} = \frac{pathSafety}{maxPathSafety} \tag{3}$$

$$sm_{term} = \frac{pathSmooth + avePathSmooth}{avePathSmooth} \tag{4}$$

$$f_f = w_d * d_{term} + w_{sa} * sa_{term} + w_{sm} * sm_{term} \tag{5}$$

Unfeasible paths: The Equations [2, 6, 7 and 8] presents the fitness function for unfeasible paths considering the length, the rate of infeasibility (the ratio between the number of unfeasible edges and total number of edges) and the degree of infeasibility (the ratio between the sum of the lengths of the unfeasible sections and the total length of the path). In this case, *maxPathCrossDepth* value is considered as an upper bounds for cross depth weight of the paths.

$$ur_{term} = 1 - pathUR \tag{6}$$

$$cdw_{term} = \frac{maxPathCrossDepth - pathCrossDepth}{maxPathCrossDepth} \tag{7}$$

$$f_u = w_d * d_{term} + w_{ur} * ur_{term} + w_{cdw} * cdw_{term} \tag{8}$$

3) Genetic Operators: This subsection describes the genetic operators applied to individuals during evolution.

Crossover: The crossover operator is applied to each individual of the population (pw_{f1}) with probability p_c and consist in drawing another individual from the population (pw_{f2}), drawing a cell or cross position (i) and generating two new children using the genetic information of the parents (pw_{f1} and pw_{f2}), as it is shown in Equations 9 and 10. Figure 3a presents an example of application of the crossover operator.

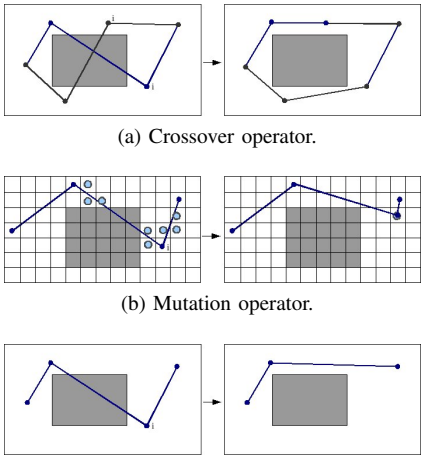
$$pw_{s1} = \{[pw_{f1,1}, ..., pw_{f1,i}], [pw_{f2,i+1}, ..., pw_{f2,n}]\} \tag{9}$$

$$pw_{s2} = \{[pw_{f2,1}, ..., pw_{f2,i}], [pw_{f1,i+1}, ..., pw_{f1,m}]\} \tag{10}$$

Mutation: The mutation operator is applied to each individual with probability p_m and consists in drawing a cell of the path, pw_i, and replaces it with another cell belonging to the section defined by the predecessor pw_{i-1} and successor pw_{i+1}. Figure 3b presents an example of application of the mutation operator.

Smooth: An ad-hoc smooth operator is used to provide a mechanism for removing cells from paths. This operator allows building shorter and softer paths and can help to factibilize infeasible solutions. It also has another positive effect that is the elimination of redundant cells (cells belonging to the line segment determined by the cells immediate predecessor and successor). The operator is applied to each individual, with probability p_s and consists in drawing a cell from the path, pw_i, and removing it. Figure 3c presents an example of application of the ad-hoc smooth operator.

4) Selection: In order to be consistent with previous decisions and preserve genetic diversity in the early stages of the evolution process, we used a moderately elitist selection operator. While this mechanism ensures that the 5% of best individuals are selected for the next generation, the rest of the population is drawn without taking into account the associated fitness value and thus enabling the selection of feasible and unfeasible individuals with equal probability.



(a) Crossover operator.

(b) Mutation operator.

(c) Smooth operator.

Figure 3: Example of genetic operators application.

5) Initialization of the Population: Based on the ideas of the related works, the initial population in our work is generated as follows:

1) Generate a Voronoi diagram from the set of points of the centers of the obstacles in the environment.
2) Identify the regions R_s and R_e that contain the origin and destination points of the path planning, respectively.
3) Compute shortest paths for each pair of vertices (v_s, v_e) on the underlying graph (vertices and edges of the Voronoi diagram), where v_s and v_e belong to the set of vertices of the regions R_s and R_e, respectively. We used an ad-hoc implementation of *Dijkstra's* algorithm [20] to compute the shortest paths.
4) Draw a random number of paths from the set of paths generated in step 3.
5) For each path selected in step 4, calculate the sequence of cells that represents it on the underlying regular grid.
6) Eliminate a random number of cells from each sequence of cells calculated in step 5.

Although step 6 involves the loss of information, this effect should be compensated during the evolution process by the application of the crossover operator. The elimination of cells from the set of paths aims to remove redundant cells and generate an initial population with individuals representing shortest and softer paths, although the paths may be less secure and even unfeasible.

6) Termination conditions: To stop running the algorithm the following termination criteria were defined:

Convergence: This criterion determines when the evolution does not provide best individuals or the improvement is not significant enough to continue to invest resources in it. Is determined by controlling the maximum number of evolutionary steps without register the best individual variations nor significant variations in the average population fitness.

The following values were used: {Minimum number of evolutions=5ev, Maximum number of changes with no change in the fittest individual=5ev, Minimum percentage of improvement in the average fitness of the population=0.08%}.

Maximum number of evolutionary steps: Is used as a preventive mechanism that guarantees the end of the algorithm. The number of steps was set at 155ev.

IV. Experimental results

This section describe simulated and real test environments, presenting the platforms used for experimentation, and the results obtained, both in assessing the *GA* as to compare them against previous proposals.

A. Platforms

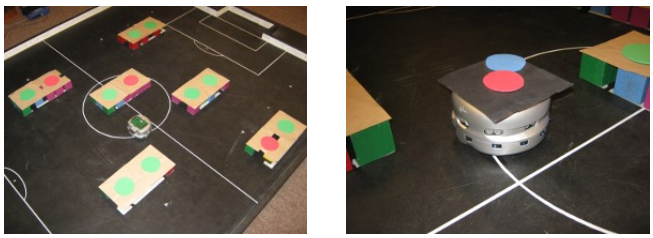
This section details the technical specification of hardware and software tools and the arena used for the experiments.

1) Simulated environment:

- Hardware: {CPU=Intel Pentium Dual Core E2140 - 1.60GHz, RAM=2GB DDR}
- Software: {OS=GNU/Linux - Xubuntu 2.6.31-22, Programming Language=Java 1.6, GA Library=JGAP 3.3.3, Graph Library=JGrapht 0.8.1, Voronoi Library=Quickhull3d 1.4, Visor Processing 1.0.7}

2) Real environment:

- Arena: {An 1680x1680 milimeters square area over a FIRA Mirosot field [21]}
- Vision System: {Doraemon [22]}
- Robot: {KheperaIII + KoreBot II [23]}
- Resources and media: {available online [24]}



(a) Scenario. (b) Robot in action.

Figure 4: Moving in a real environment.

B. Parameters settings and used scenarios

An extensive configuration analysis has been carried out for calibrating the parameters of the proposed *GA*. The parameters studied were population size, mutation, crossover and smooth operator probabilities, on 18 cases repeated 10 times each. The range of values considered were: 170-40 for the population size, 0.02-0.01 for the mutation probability, 0.62-0.50 for the crossover probability and 0.50-0.25 for the smooth operator probability. The best parameters values determined using several different scenarios and finally used in this work were: {ps=142, p_m=0.0139,p_c=0.51 and p_s=0.33}.

Two different studies were conducted to evaluate the algorithm proposed in this work. The first one was performed in order to evaluate the quality of the obtained solutions and the performance of the algorithm. The second study was performed in order to compare our proposal with the previous work [14]. For the first study were considered 10x10

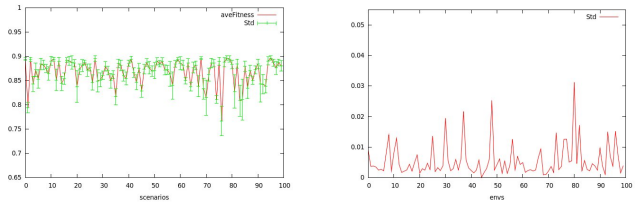
dimension scenarios where the amount and location of the obstacles were determined at random. For the second one, it was considered a scenario identical to the one used in the previous work to allow a fair comparison.

Figure 5a presents the average fitness and standard deviation on the 50 independent runs for 10x10 size scenarios. From these results, it is possible to draw the following conclusions:

1. the algorithm does not converge prematurely, but manages to evolve quality solutions. Similarly, this result is repeated for different sizes of scenarios.

2. whereas the generation of the scenarios was carried out at random and the maximum fitness value is 1, we can conclude that for a large number of situations reach high quality solutions.

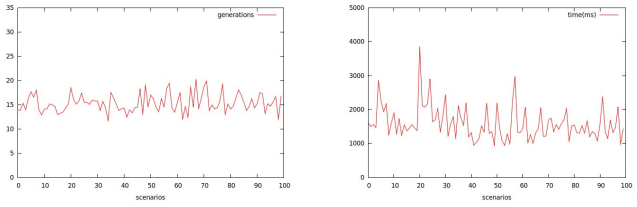
Figure 5b shows the standard deviation of the average fitness obtained for each of the scenarios. The standard deviation remained bounded by low values in most cases, showing that the algorithm has a very good stability to globally converge to high quality solutions.



(a) Average fitness. (b) Standard deviation.

Figure 5: Average fitness and standard deviation.

Figure 6a shows the number of generations required to reach the stop condition while Figure 6b shows the execution time required for solving each of the scenarios. It can be concluded that the algorithm is extremely stable in the number of generations required for solving each scenario, as well as in the time required for solving each scenario.



(a) Generations. (b) Execution Time.

Figure 6: Evolution Generations & Execution Time.

C. Comparative analysis

In order to compare our proposal with the previous work, we studied the quality of the solutions obtained and the performance of the algorithms. .

Table I presents the results of 1000 independent runs of the *GA* based on the Digital Potential Field method (*DPF*) and *VD*.

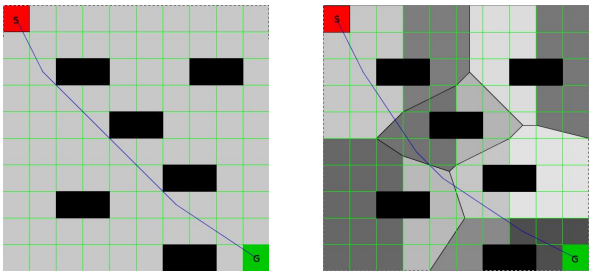
	GA over *DPF*	*GA* over *VD*
Environment generation (ms)	1632	1305
GA evolution (ms)	275.79	223
Average number of generations	12.51	15
Average *Fitness*	0.82990	0.83437
Standard deviation	0.01067	0.01376
Best *Fitness*	0.84583	0.85433
Deviation from the fittest	0.01267	0.01996
Average length	656.27	662.44
Average safety	253.58	291.19
Average smoothness	0.29111	0.29058

Table I: Numerical comparison.

The results obtained show that it is possible to maintain the quality of the solutions while significantly reducing the computation time. The *GA* based on *DPF* approach requires a 25.05% more time in the initial phase of building the environment. Also, when considering the quality of the solutions obtained, the *GA* based on *VD* achieves better solutions, finding safer paths with similar values for the length and smooth. It also shows a higher level of stability in the sense that converges to populations whose members are more similar.

On the other hand, Figure 7 presents graphically the solutions obtained by each of the variants implemented. Figure 7a to be seen as the resulting path is reasonably smooth and straightforward but, in comparison with Figure 7b that throws the proposal based on *VD*, is less secure.

Once the questions previously raised were answered and considering the practical application of both approaches in the control of a real robot able of performing multiple tasks in an environment, it should be studied the effect of assigning multiple tasks sequentially in different zones of the environment for each approach.



(a) *GA* over *DPF*. (b) *GA* over *VD*.

Figure 7: Visual comparison.

The environment must be generated for *DPF* each time a new path has to be calculated when the robot has changed its position because the coding of paths depends on the initial position of the robot. That is to say, if the robot has to perform a sequence of n tasks in k different zones of the environment, it should be generated k different environments, one for each different zone. On the contrary, the coding of paths is independent of the initial position of the robot in *VD* approach thereby the generation of the environment, which is

the most computationally intensive part of the algorithm, is performed only once.

The difference concerning the computational efficiency between the two proposals considering a static environment and multiple tasks is presented in Figure 8.

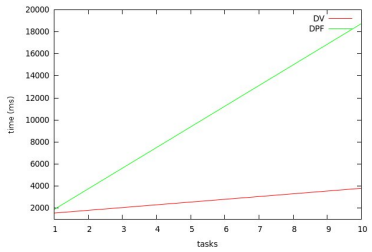


Figure 8: Multiple tasks execution time.

V. Conclusion and future work

In this work we tackled the path planning problem in static environments using a genetic algorithm based on Voronoi diagrams for representing the environment. The obtained results show that our proposal is promising from a methodological point of view and also considering its practical applicability in mobile robotic systems, due to its short execution time; our proposal improves both the quality as well as the computational time of the solutions compared with the proposal based on *DPF* [14]. This last aspect is not caused by a better computational efficiency of the algorithm, but also because the representation based on *VD* overcomes the limitation established by the representation based on *DPF*, which requires regenerating the environment every time the robot needs to plan a new path, having changed its original position.

On the other hand, we couldn't find a public implementation of the *Visibility Pathway* proposal presented by *Roque and Doering* [6], in order to compare results.

It should be noted that *Zhang et al.* [14], work on dynamic and real environments, and, along this line, the proposal presented by *Roque and Doering* [6] simulate dynamic environments obtaining response times appropriate for working with real robots.

Our future work will take into account the dynamic aspect, so that we can evaluate the applicability of this alternative to path planning in a more complex and realistic environments with further restrictions. Therefore, we shall consider the applicability of a multi-objective *GA* in order to balance the objectives of minimizing the distance, maximize safety and smoothness of solutions.

VI. Acknowledgment

The authors would like to thank to *Eduardo Grampín* and *Federico Andrade* for his useful advice that was instrumental to improve this paper.

References

[1] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot using neural networks," *IEEE Transaction on Neural Networks*, vol. 9, no. 4, pp. 589–600, Jul. 1998.

[2] K. C. Tan, K. K. Tan, T. H. Lee *et al.*, "Autonomous robot navigation based on fuzzy sensor fusion and reinforcement learning," in *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*. IEEE, Oct. 2002, pp. 182–187.

[3] R. Maurya and A. Shukla, "Generalized and modified ant algorithm for solving robot path planning problem," *3rd IEEE International Conference on Computer Science and Information Technology*, vol. 1, pp. 643–646, Jul. 2010.

[4] G. Liu, T. Li, Y. Peng *et al.*, "The ant algorithm for solving robot path planning problem," in *Proceedings of the Third International Conference on Information Technology and Applications*, vol. 2. IEEE, Jul. 2005, pp. 25–27.

[5] W. L. Roque, *Introducao a Técnicas de Planejamento de Trajetórias de Robôs Móveis*. -, 1996, vol. 1, pp. 125–150.

[6] W. L. Roque and D. Doering, "Trajectory planning for lab robots based on global vision and voronoi roadmaps," *Robotica*, vol. 23, no. 4, pp. 467–477, 2005.

[7] R. Wein, J. P. van den Berg, and D. Halperin, "The visibility–voronoi complex and its applications," *Computational Geometry: Theory and Applications*, vol. 36, no. 1, pp. 66–87, 2007.

[8] Z. Bi, Y. Yimin, and Y. Wei, "Hierarchical path planning approach for mobile robot navigation under the dynamic environment," in *The IEEE International Conference on Industrial Informatics*, Jul. 2008, pp. 372–376.

[9] X. Hu and C. Xie, "Niche genetic algorithm for robot path planning," in *Third International Conference on Natural Computation*, vol. 2. IEEE, Aug. 2007, pp. 774–778.

[10] L. Lei, H. Wang, and Q. Wu, "Improved genetic algorithms based path planning of mobile robot under dynamic unknown environment," in *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*. IEEE, Jun. 2006, pp. 1728–1832.

[11] M. Mansouri, M. Aliyari Shoorehdeli, and M. Teshnehlab, "Integer ga for mobile robot path planning with using another ga as repairing function," in *Proceedings of the IEEE International Conference on Automation and Logistics*. IEEE, Sep. 2008, pp. 135–140.

[12] X. Yan, Q. Wu, J. Yan *et al.*, "A fast evolutionary algorithm for robot path planning," in *2007 IEEE International Conference on Control and Automation*, Jun. 2007, pp. 84–87.

[13] L. Yanju, C. Yundong, Y. Yu, D. Wang, W. Xin, Y. Jun *et al.*, "A path planning study of autonomous mobile robot in dynamic environment," *3rd IEEE International Conference on Industrial Electronics and Applications*, pp. 1040–1043, Jun. 2008.

[14] Y. Zhang, L. Zhang, and X. Zhang, "Mobile robot path planning base on the hybrid genetic algorithm in unknown environment," in *ISDA '08: Proceedings of the 2008 Eighth International Conference on Intelligent Systems Design and Applications*, vol. 2. Washington, DC, USA: IEEE Computer Society, Nov. 2008, pp. 661–665.

[15] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion-Theory, Algorithms, and Implementation*. MIT Press, 2005.

[16] H. H. González-baños, D. Hsu, and J. claude Latombe, "Motion planning: Recent developments," Tech. Rep., 2005.

[17] J. Minguez, F. Lamiraux, and J. P. Laumond, *Motion Planning and Obstacle Avoidance*. Springer, 2008, ch. 35, pp. 827–852. [Online]. Available: http://www.springer.com/engineering/robotics/book/978-3-540-23957-4

[18] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Upper Saddle River,NJ, USA: Addison-Wesley Professional, 1989.

[19] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[20] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959. [Online]. Available: http://jmvidal.cse.sc.edu/library/dijkstra59a.pdf

[21] Mirosot rules. [Online]. Available: http://www.fira.net/

[22] J. Anderson and J. Baltes. (2002) Doraemon user manual. [Online]. Available: http://robocup-video.sf.net/

[23] K-Team. Khephera iii. [Online]. Available: http://www.k-team.com/

[24] F. Benavides. Home page. [Online]. Available: http://www.fing.edu.uy/∼fbenavid