International Journal of Recent Engineering Research and Development (IJRERD)
ISSN: 2455-8761
www.ijrerd.com || Volume 07 – Issue 07 || July 2022 || PP. 01-08

A Pseudooptimal Navigation Path Planning Algorithm for Service Robot Environments

Fredy Martínez[1], Angélica Rendón[2]
[1]*Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia*
[2]*Facultad de Ingeniería, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia*

Abstract: Typical service robot environments are characterized by being dynamic, with human interaction, and restricted for the use of certain types of sensors due to their operation in indoor environments. These characteristics make the path planning for this type of task a complex problem and at the same time of priority solution. Navigation in such environments turns out to be a fairly simple activity for a person or a pet in their daily activity, but it also turns out to be an NP-complete computational problem quite difficult to solve. This type of problem is non-deterministic and with a polynomial-time computational solution. Many strategies have been formulated to solve this problem, but as such, it remains an open engineering problem. We propose to use an Ant Colony Optimization (ACO) algorithm to define in a short time a pseudo-optimal path considering the shortest possible distance. To simplify the scheme and turn it into a real application, we decompose the navigation environment into regions to move the robot along them. The proposed scheme is evaluated for a real environment and a robot with different obstacle configurations. In all cases, the scheme was able to find suitable solutions without high computational costs.

Keywords: Ant colony optimization, navigation, optimization, path planning, pheromone, pseudo-optimal.

1. Introduction

Planning the movement of robots is a quite complex problem, even more so when we consider dynamics in the environments. The most basic problem is to make a robot move from one point to another within an environment. This problem becomes even more complicated if such a solution requires using the most efficient path, which is usually the shortest. For this reason this kind of problems are usually attacked with uninformed search strategies.

The Ant Colony Optimization (ACO) algorithm has been proposed many times as a motion planning strategy for robots [1, 2, 3, 4, 5]. In most cases the algorithm is defined specifically to find the shortest path in a known environment. This is given the very nature of the algorithm. It is common to find in these applications that the navigation environment is divided into squares in which the robot fits perfectly. This approach, however, is difficult to fulfil in reality, since the dimensions of the robot and its on board sensors make it difficult to locate it in the grid (and quite expensive). In addition, the robot is required to have an excellent odometry, which can also generate movement errors.

However, this is not the only kind of robot in which the designer seeks to optimize a path. This strategy has also been used in robotic arms, submarine movement, automated guided vehicle, or welding robots. In these applications the kinetic, dynamic and energetic equations are rewritten in such a way that the problem becomes a search, and then the ACO is applied to look for the solution [6, 7, 8, 9]. In this kind of applications the algorithm has also been quite successful.

A particularly interesting case, and of much current interest is the multi-robot. This is a multi-objective optimization problem that has the characteristic of being intimately linked to the structure of the ACO, so the algorithm could well be emulated in physical form with many robots. However, the typical strategy is also to use off-line the ACO to find the shortest path, and then transfer this information to the group of robots [10, 11, 12].

ACO has also mixed with other movement planning strategies for robots [13, 14, 15, 16, 17]. The algorithm has been mixed with potential fields, tabu search, genetic algorithms, particle swarm optimization, and even fuzzy logic. In these cases the alternative strategy is used as heuristics to define the possible paths under the conditions of the task, and the ACO performs the best optimization. These strategies tend to be more real than the traditional grid, but the selected heuristics inherit their problems to the strategy, as for example problems of local minima.

We propose a path planning strategy for a robot in a dynamic observable environment using ACO for the selection of the most optimal path within the possible paths. Instead of using grids or heuristics, and thinking of applying the strategy on a real robot navigating a real environment, we decompose the navigation environment

into regions, and force the robot to navigate from region to region. This idea reduces the complexity of the problem by reducing the possible paths, and facilitates the movement of the robot by defining the regions according to their size and topology of the environment.

The following part of the paper is arranged in this way. Section 2 presents preliminary concepts and problem formulation. Section 3 illustrates the design profile and development methodology. Section 4 we present the preliminary results. And finally, in Section 5 we present our conclusions.

2. Problem Formulation

A set of *n* robots is defined in a $\boldsymbol{W}$ workspace. Let $\boldsymbol{W} \subset \mathbb{R}^2$ be the closure of a contractible open set in the plane that has a connected open interior with obstacles that represent inaccessible regions. Let $\partial \boldsymbol{W}$ denote the boundary of $\boldsymbol{W}$.

Let $\boldsymbol{O}$ be a set of obstacles, in which each $O \subset \boldsymbol{O}$ is closed with a connected piecewise-analytic boundary that is finite in length. Furthermore, the obstacles in $\boldsymbol{O}$ are pairwise-disjoint and countably finite in number. Let $E \subset \boldsymbol{W}$ be the free space in the environment, which is the open subset of $\boldsymbol{W}$ with the obstacles removed.

Consider that the free space *E* is divided into a set of *s* connected regions such that (Eq. 1):

$$\bigcup_{n=1}^{s} r_n = E \qquad (1)$$

The decomposition of into regions is done according to the movement needs of the robots, but in such a way that each region *r* is small in front of *E*, and large in front of the size of the robots, in this way each robot can be considered as a moving point. See Fig. 1 for an example. Decomposition can be done with some geometric strategy, such as visibility graphs, approximate cell decomposition, exact cell decomposition or Voronoi diagrams.

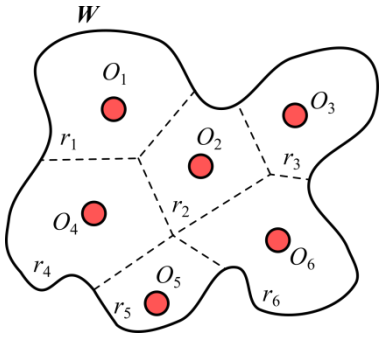

Figure 1: Decomposition of the environment in regions. Diagram of Voronoi created in an environment from six points representing obstacles

Each region can be represented as a node of a graph, therefore the path between a region *i* and another region *j* can be represented by a graph as the connection of the nodes (regions) connected from *i* to *j* (Fig. 2). We want to optimize the selection of the route between the regions *i* and *j* by means of an Ant Colony Optimization algorithm (ACO). In such sense the ants represent the movement of the robots, and using pheromone deposits, the probability that an ant *k* located in node *i* will choose to go to another node *j* is given by (Eq. 2):

$$p_{ij}^{k} = \begin{cases} \dfrac{\left(\tau_{ij}^{k}\right)^{\alpha}\left(\eta_{ij}^{k}\right)^{\beta}}{\sum_{l \in N_i^k}(\tau_{il}^{k})^{\alpha}(\eta_{il}^{k})^{\beta}} & if\ j \in N_i^k \\ 0 & if\ j \notin N_i^k \end{cases} \qquad (2)$$

Where:

- τ_{ij}^{k} = Pheromone levels.
- η_{ij}^{k} = Heuristic information (application dependent).
- α, β = Weights pondering the importance of pheromone and heuristic values (application dependent).

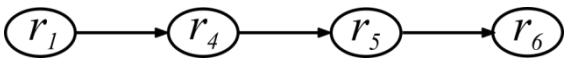

Figure 2: Graph with the navigation path between regions r1 and r6 shown in Fig. 1

As in real ants, the higher the level of pheromone on a path, the greater the probability that the ants will take the path again. The summation in the denominator considers all possible choices (regions or nodes) in the N_i^k set when the ant is in node *i*. When $\beta = 0$, $\left(\eta_{ij}^k\right)^\beta = 1$, then the probability only depends on the pheromone levels. On the other hand, when $\alpha = 0$, the probability only depends on the heuristic values (the ant moves towards the nearest node).

An important characteristic of the ACO is the percentage or rate of evaporation from the pheromone ρ. This percentage of evaporation from node *i* to node *j* is determined as (Eq. 3):

$$\tau_{ij} \longleftarrow (1-\rho)\tau_{ij} \quad / \quad 0 \leq \rho < 1 \qquad (3)$$

After evaporation occurs, the new pheromone level is updated with additional pheromone deposits placed in the path by the ants that use it. The update is given by (Eq. 4):

$$\tau_{ij} \longleftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \quad / \quad \Delta\tau_{ij}^k = \frac{1}{C^k} \qquad (4)$$

C^k is the associated cost or reward of ant *k* for choosing the path. According to the algorithm, based on the update of the pheromone level in a finite time there will only be one path between nodes *i* and *j*, and this will be the one with the lowest cost.

The objective of this research is to find the shortest route between an arbitrary initial position (region *i*) and the target point (region *j*), advancing in $\boldsymbol{W}$ avoiding obstacles even when they move (dynamic environment). We want the shortest route, and this will be the characteristic to optimize with the ACO.

3. Methodology

We propose the use of an ACO to select the navigation path in an observable environment. The navigation environment has been segmented into regions according to the size of $\boldsymbol{W}$ and the location of the obstacles. The idea of the segmentation process is to divide the navigable free space *E* into regions in such a way that the regions become desired locations for the robot. Therefore, the size of the regions must be small compared to the environment. In addition, we want the robots to be able to move smoothly within these regions, so we select a region size considerably larger than the area the robot occupies on the ground. According to these definitions, the r_n regions defined in *E* can each be represented as a navigation node, i.e. the navigation path selection problem has a total of *n* nodes that can represent the position of the robot at any time.

The goal is to find the shortest path between the initial node and the target node. Consequently, for the design of the solution to the problem the size of the navigation environment loses importance, and the existing distance between nodes becomes more important. The total length of the path is selected as the cost or reward function associated with each possible solution. This total length is defined as the sum of the distances between the nodes that make up the path. We implement the algorithm for the navigation environment in our laboratory. We have a square space of 4.2 m x 4.0 m. On this environment we have configured different navigation environments by placing several obstacles in different locations. Our robot is the ARMOS TurtleBot 1 robotic platform with a ground plane size of 55.8 cm x 32.5 cm [18] (Fig. 3).

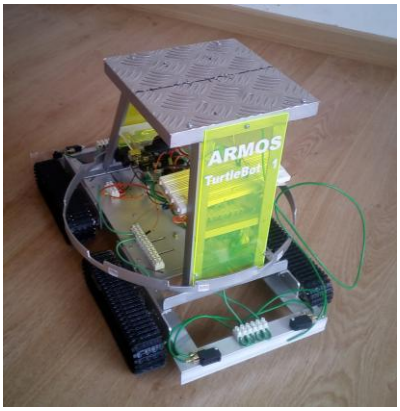

Figure 3: ARMOS TurtleBot 1 robotic platform. The robot has four tracks, each driven by a DC motor. It has WiFi communication capacity, and a control unit commanded by a Cortex A53 processor at 1.2 GHz

There is a great variety of options for regions and their sizes, it is not fixed the number of nodes adjacent to a given node *i*. The regions are evenly distributed along *E*, but the distance between two regions (nodes) is always different, and is measured with respect to the centroid of the region. Fig. 4 shows an example according

to the segmentation performed in Fig. 1. The distances are measured in centimeters, however in the algorithm are normalized in the range of 0 to 1. In the simulations the robot was considered as a point, with the ability to move in any direction.

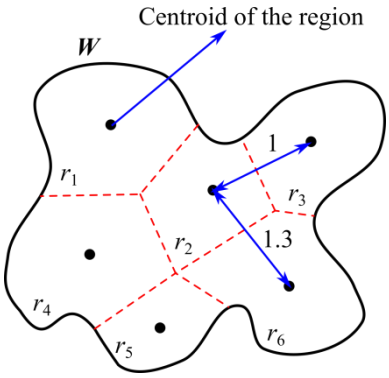

Figure 4: Regions and their centroids according to the segmentation in Fig. 1. Region 1 has two adjacent regions (r_2 and r_4), while region 4 has three adjacent regions (r_1, r_2, and r_5). The distance between the centroids of r_2 and r_3 is 1, while the distance between the centroids of r_2 and r_6 is 1.3

We use different obstacle configurations in the same environment, including the one in which there are no obstacles. Fig. 5 shows one of these configurations with five obstacles. The figure also shows the segmentation into regions and the location of the initial (green point region) and target (red point region) regions. These regions were selected arbitrarily, and changed with each new environment configuration. In this case we applied Voronoi from the four vertices of the obstacles, and then eliminated the intersection of the regions with the obstacles making the edges of the obstacles become the new boundaries of the regions. In total we create 20 regions, and label them sequentially from top to bottom and from left to right.

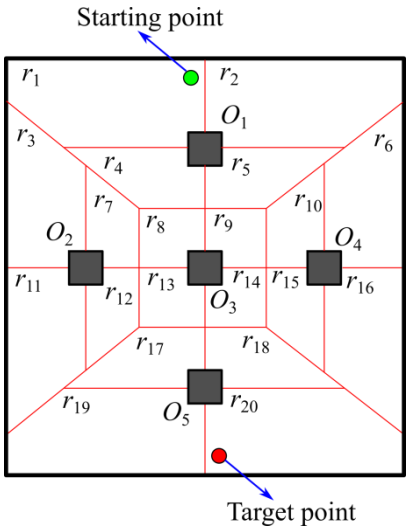

Figure 5: One of the configurations used for the evaluation of the proposed strategy. The starting point was arbitrarily selected at the top (green) and the target point at the bottom (red). According to the obstacles, the environment was decomposed in 20 regions

We apply the ACO algorithm to find the shortest path. Fig. 6 shows a flowchart of the implemented algorithm. As such it is a classic Ant Colony Optimization algorithm, first we configure the starting node (region), and then we search the following nodes to find the next node with the lowest cost. Then it verifies if the found node is the target node, if it is not it keeps moving from the new position always looking for the next node with the lowest cost. With this strategy at some point it will find the target node, in that moment it must begin to optimize the path. For it begins to play with the update of the value of pheromone, if the path is not optimized the logic makes that the pheromone evaporates and the value is updated according to the deposits of pheromone, and the whole algorithm is restarted until it finds an impossible way to improve or finish the iterations programmed in the algorithm.

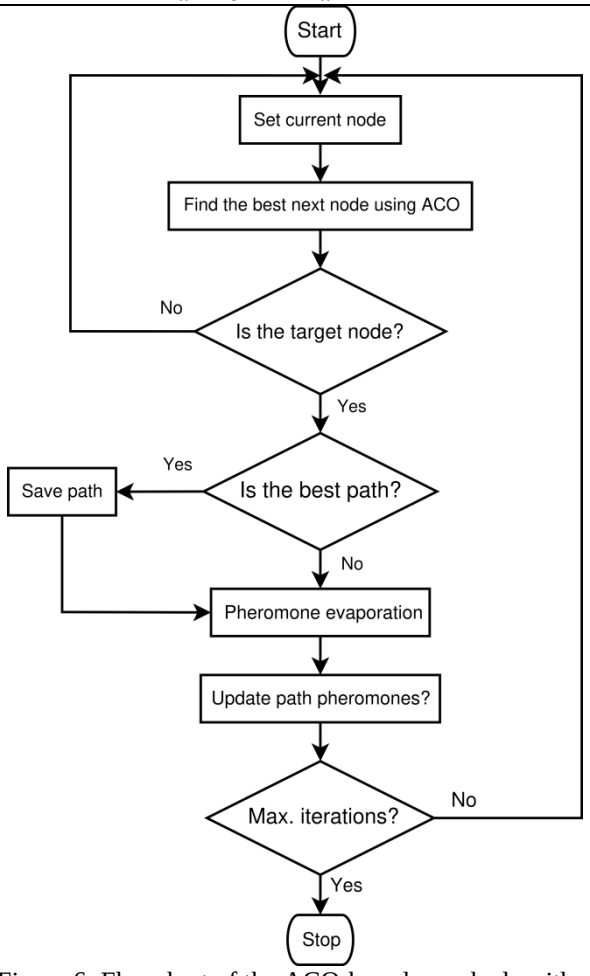

Figure 6: Flowchart of the ACO-based search algorithm

To evaluate the behaviour of the strategy in a dynamic environment we play with the obstacles. That is to say, after establishing an environment configuration and finding an optimal path with our algorithm based on ACO, we move the obstacles and apply again with the new environment configuration without altering the current pheromone value.

4. Results and Discussion

In the algorithm ants travel from the initial region to the target region evaluating all possible paths. We assume that they return using the same path and depositing pheromone on their return. They deposit more pheromone at short distances than at long distances and only along the path used. Each ant decides autonomously which region to visit according to the level of pheromone on the path and the distance to the nearest region.

For example, for the case shown in Fig. 5 we measured in the laboratory and constructed the distance matrix as shown in Fig. 7 [19]. The distances were measured in centimetres between the centroids of each region, and only for the neighbouring regions of each region. For example, from region 1 it is only possible to move to regions r_2, r_3 and r_4. For the other distances an infinite value was assigned to make the trip to them impossible. In the same way we did for the distances of a region to itself (diagonal of the matrix).

```
ant_colony.py ✖   envir09.py ✖

import numpy as np

from ant_colony import AntColony

distances = np.array([[np.inf, 184,    124,    81,     np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [184,    np.inf, np.inf, np.inf, 81,     124,    np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [124,    np.inf, np.inf, 115,    np.inf, np.inf, 77,     np.inf, np.inf, np.inf, np.inf, 130,    np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [81,     np.inf, 115,    np.inf, 106,    np.inf, 70,     67,     np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, 81,     np.inf, 106,    np.inf, 115,    np.inf, np.inf, 67,     70,     np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, 124,    np.inf, np.inf, 115,    np.inf, np.inf, np.inf, np.inf, 77,     np.inf, np.inf, np.inf, np.inf, np.inf, 130,    np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, 77,     70,     np.inf, np.inf, np.inf, 58,     np.inf, np.inf, 77,     np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, 67,     np.inf, np.inf, 58,     np.inf, 72,     np.inf, np.inf, np.inf, 62,     np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, np.inf, 67,     np.inf, np.inf, 72,     np.inf, 58,     np.inf, np.inf, np.inf, 62,     np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, np.inf, 70,     77,     np.inf, np.inf, 58,     np.inf, np.inf, np.inf, np.inf, np.inf, 77,     np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, 130,    np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 77,     np.inf, np.inf, np.inf, np.inf, 115,    np.inf, 124,    np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 77,     np.inf, np.inf, np.inf, 77,     np.inf, 58,     np.inf, np.inf, np.inf, 70,     np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 62,     np.inf, np.inf, np.inf, 58,     np.inf, 72,     np.inf, np.inf, 67,     np.inf, np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 62,     np.inf, np.inf, np.inf, 72,     np.inf, 58,     np.inf, np.inf, 67,     np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 77,     np.inf, np.inf, np.inf, 58,     np.inf, 77,     np.inf, 70,     np.inf, np.inf, np.inf, np.inf],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, 130,    np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 77,     np.inf, np.inf, 115,    np.inf, 124],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 115,    70,     67,     np.inf, np.inf, np.inf, np.inf, 106,    81,     np.inf],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 67,     70,     115,    106,    np.inf, np.inf, 81],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 124,    np.inf, np.inf, np.inf, np.inf, np.inf, 81,     np.inf, np.inf, 184],
                      [np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, np.inf, 124,    np.inf, 81,     184,    np.inf]])

ant_colony = AntColony(distances, 100, 20, 2000, 0.95, alpha=1, beta=1)
shortest_path = ant_colony.run()
print "shorted_path: {}".format(shortest_path)
```

Figure 7: Defined distance matrix for the documented environment

The pheromone deposit matrix is defined of the same size as the distance matrix. This matrix is initialized with small values all of equal value. The configuration of this example environment is shown in the lower part of Fig. 7. The variables used in the function are in order:

- Distance matrix: Square matrix of distances.
- Number of ants running per iteration: 100
- Number of best ants who deposit pheromone: 20
- Number of iterations: 2000
- Rate at which pheromone decays: 0.95
- Alpha (exponent on pheromone): 1
- Beta (exponent on distance): 1

The algorithm spent just under a minute finding the shortest path as shown in Eq. 5. We have represented this response in the navigation environment in Fig. 8, and as a graph in Fig. 9. The path uses six nodes (regions): r_1, r_4, r_8, r_{13}, r_{17} and r_{19}. The total distance is 359 cm.

$$shorted\_path: ([(1,4),(4,8),(8,13),(13,17),(17,19)], 359.0) \qquad (5)$$

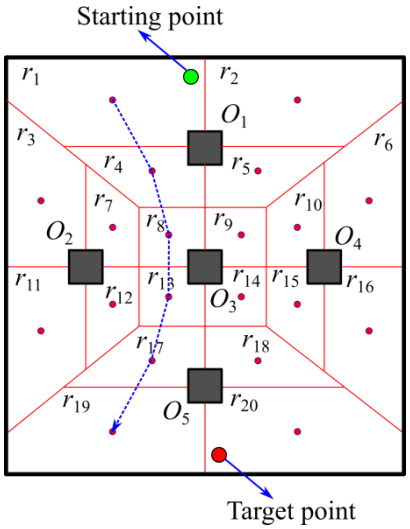

Figure 8: Navigation path selected by the proposed algorithm drawn on the real environment in laboratory

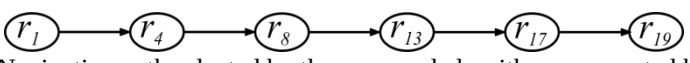

Figure 9: Navigation path selected by the proposed algorithm represented by a graph

We tested the algorithm with diverse configurations of environments modifying the amount and position of the obstacles but maintaining the location of the starting points and target, therefore the initial and target regions were always the same [20]. The algorithm in each case was able to identify the shortest path between the two regions. The simulation time is basically the same in each case. The computational cost in all cases is very

similar.

The last tests with the strategy sought to automate the generation of regions from photographs of the environment and image processing. The final objective is to make the strategy autonomous and independent of the configuration of the environment.

5. Conclusion

In this paper we propose a strategy of quasi-optimal selection of the shortest path for a robot in an observable environment. The strategy uses as optimization scheme the Ant Colony Optimization (ACO) algorithm. To make viable the strategy on real navigation environments, in particular those facing a service robot, we propose to decompose the free space of the environment in a finite number of regions by means of some geometrical strategy, in particular, considering the location of the obstacles. This decomposition must be done in such a way that the regions will be small with respect to the environment, thus the navigation path is more precise. The results of the strategy under laboratory conditions demonstrate its success in determining the shortest path. Future research may include other additional criteria to the algorithm, such as the dangerousness of the path or the advantages of a path as a source of energy. It is also necessary to evaluate much more complex environments and automate the strategy.

References

[1] H. Chen-Chien, H. Ru-Yu, K. Wen-Chung and L. Shih-An, Fpga-based path planning using improved ant colony optimization algorithm, *IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin 2015)*, 2015, pp. 443–444.

[2] R. Rashid, N. Perumal, I. Elamvazuthi, M. Tageldeen, M. Khan and S. Parasuraman, Mobile robot path planning using ant colony optimization, *2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA 2016)*, 2016, pp. 1–6.

[3] P. Joshy and P. Supriya, Implementation of robotic path planning using ant colony optimization algorithm, *International Conference on Inventive Computation Technologies (ICICT 2016)*, 2016, pp. 1–6.

[4] C. Honglei, W. Zongzhi, K. Rongxue and Y. Chao, Global path planning for explosion-proof robot based on improved ant colony optimization, *Asia-Pacific Conference on Intelligent Robot Systems (ACIRS 2016)*, 2016, pp. 36–40.

[5] S. Huangfu, S. Tang, B. Song, M. Tong and M. Ji, Robot path planning based on improved ant colony optimization, *International Conference on Robots & Intelligent System (ICRIS 2018)*, 2018, pp. 25–28.

[6] J. Li, T. Dong, Y. Li and Y. Hao, Study on robot path collision avoidance planning based on the improved ant colony algorithm, *8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC 2016)*, 2016, pp. 540–544.

[7] J. Xing, K. Junfeng, Z. Jingjing and Y. Xiang, Trajectory planning of a six-dof robot based on a hybrid optimization algorithm, *9th International Symposium on Computational Intelligence and Design (ISCID 2016)*, 2016, pp. 148–151.

[8] H. Yihu, G. Yiming and Z. Zhao, Research on the path planning of hair-insertion robot arm based on ant colony optimization, *37th Chinese Control Conference (CCC 2018)*, 2018, pp. 5191–5195.

[9] Y. Shan, Study on submarine path planning based on modified ant colony optimization algorithm, *IEEE International Conference on Mechatronics and Automation (ICMA 2018)*, 2018, pp. 288–292.

[10] H. Wei and X. Xinying, Immune ant colony optimization network algorithm for multi-robot path planning, *IEEE 5th International Conference on Software Engineering and Service Science*, 2014, pp. 1118–1121.

[11] X. Yanqin, L. Mingshan, Z. Yuan, W. Rui and Z. Wenbo, Ant colony based on cat swarm optimization and application in picking robot path planning, *7th IEEE International Conference on Software Engineering and Service Science (ICSESS 2016)*, 2016, pp. 162–165.

[12] Y. Li, Y. Li, Y. Guo and K. Cai, Cooperative path planning of robot swarm based on aco, *IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2017)*, 2017, pp. 1428–1432.

[13] W. Jingyu, C. Junfeng, C. Shi and X. Yingjuan, Double heuristic optimization based on hierarchical partitioning for coverage path planning of robot mowers, *12th International Conference on Computational Intelligence and Security (CIS 2016)*, 2016, pp. 186–189.

[14] S. Haiming, A study of welding robot path planning application based on genetic ant colony hybrid algorithm, *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC 2016)*, 2016, pp. 1743–1746.

[15] W. Qing-Quan, A novel method for real-time globally optimized path planning in a dynamic environment based on ant colony system algorithm, *13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP 2016)*, 2016, pp. 53–58.

[16] X. Qi-Lei, C. Man-Man and Z. Lei-Hong, The robot path planning based on ant colony and particle swarm fusion algorithm, *Chinese Automation Congress (CAC 2017)*, 2017, pp. 411–415.

[17] W. Hui, W. Zheng, Y. Lijun, W. Xueying and L. Chaoda, Ant colony optimization with improved potential field heuristic for robot path planning, *37th Chinese Control Conference (CCC 2018)*, 2018, pp. 5317–5321.

[18] F. Martínez, TurtleBot3 robot operation for navigation applications using ROS, *Tekhnê*, 2021, 18(2): 19-24.

[19] A. Rendón, Operational amplifier performance practices in linear applications, *Tekhnê*, 2019, 16(1): 57-68.

[20] L. Bobadilla, F. Martinez, E. Gobst, K. Gossman, SM LaValle, Controlling wild mobile robots using virtual gates and discrete transitions, *American Control Conference (ACC 2012)*, 2012, pp. 743-749.