2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE).
Mexico City, Mexico. September 5-7, 2018

Optimal trajectories generation for autonomous navigation tasks in mobile robots

Viridiana Y. Hernández Márquez
Automatización y Control
UPT
Hidalgo, México
1631007@upt.edu.mx

Rafael S. Núñez Cruz
Automatización y Control
UPT
Hidalgo, México
rafael.nunez@upt.edu.mx

Juan M. Ibarra Zannatha
Control Automático
CINVESTAV
CDMX, México
jibarra@ctrl.cinvestav.mx

Carlos Enríquez Ramírez
Automatización y Control
UPT
Hidalgo, México
carlos.enriquez@upt.edu.mx

***Abstract*—In this work we propose to implement a route generation algorithm during the process of creating autonomous maps by means of a mobile robot. The problem to solve is to automatically obtain the optimal route in an unknown environment to explore and map the entire environment. The main contributions of this work are: the formulation of the optimization problem, the analysis of the proposed image to implement the evaluation of the objective function and the comparison of different optimization techniques applied to this problem in experimental tests. The platform used for the implementation is the open source robot called TurtleBot2, the proposed algorithm was implemented in a ROS node (Robot Operating System) that works in parallel to the mapping and odometry programs already included in this operating system.**

***Index Terms*—Trajectory Generation, Heuristic Optimization, Genetic Algorithms, Mobile Robots.**

I. Introduction

The aim of robots is to help or replace people in hazardous tasks [1] or repetitive tasks or those that require high precision [2]. Initially the field of application of robots was restricted to manufacturing processes [3], today the work of robots has expanded to a large number of areas including medicine [4], security [5], inspection, etc. These new fields of application involve more complicated challenges because the work area is usually changing, sometimes unknown, in addition to a variety of environments such as water, land and air.

If it is desired that a robot replace a person within its own environment, it will be necessary for the robot to have the same abilities of a human. This is a complicated task that requires solving different problems such as environment representation [6], [7], automatic generation of trajectories [8], detection of obstacles [9], communication between agents [10], etc.

Sometimes the environment in which the robot develops is so complicated that it is necessary the intervention of a person to make decisions at appropriate times to avoid accidents or errors in operation, this type of operation is known as teleoperation [11]. There are situations in which it is not possible to teleoperate a robot either due to environmental limitations (underwater or closed environments) or limitations of communications due to natural disasters.

Autonomous robotics, on the other hand, tries to provide the robots with the necessary capabilities to solve the problems they face on their own. This may involve the use of advanced learning techniques [12]. This work focuses on the problem of autonomous navigation for maps creation in closed indoor environments through the autonomous trajectories generation.

Currently is possible to find many works on this topic as in [13], also in [14], another approach found in [15] and [16]. In this work it is proposed that the mobile robot be able to complete the following tasks: 1. Identify zones of interest to search in an image (map) and areas to be avoided in closed indoor environments. 2. Generate an optimal trajectory for the robot, based on vision systems and inertial sensors. The autonomous trajectories generation is approached as an optimization problem with restrictions. The objective is to minimize the distance between the points in the proposed trajectory and the unknown areas on the map.

The mapping process is iterative, the proposed algorithm starts with an initial partial map which is used to build the optimal trajectory, and then the robot is moved along this trajectory after that new points are added to the map from the new location. The process is completed when there are no more unknown locations in the map. The comparison of different optimization techniques applied to this problem is shown, including the random search method, the hill climbing method and genetic algorithms. Comparison results of the algorithms in experimental tests are shown, implemented on the open source robot called TurtleBot2.

II. Autonomous navigation problem

As mentioned above there are many ways to solve the problem of autonomous navigation, it is proposed to address it as an optimization problem for which it is necessary to clearly define the decision variables, the search space, the cost function and the restrictions.

The purpose of the algorithm is to generate the trajectories (paths) that allow the robot to navigate in the entire area in which it is located to generate a complete map of the surrounding environment.

The path of the robot P is described by a polynomial parametric function of degree five defined by the following equations:

$$\begin{aligned} x(t) &= a_0^x + a_1^x t + a_2^x t^2 + a_3^x t^3 + a_4^x t^4 + a_5^x t^5 \\ y(t) &= a_0^y + a_1^y t + a_2^y t^2 + a_3^y t^3 + a_4^y t^4 + a_5^y t^5 \end{aligned} \quad (1)$$

978-1-5386-7033-0/18/$31.00

The six parameters of the equations 1 are calculated by the next six conditions: the trajectories start and end at rest so the robot moves continuously, the initial position q_0 and orientation are calculated by the current pose of the robot, the intermediate point q_1 and the end point q_2 are defined by an angle θ_k and a distance r_k with respect to the initial point as shown in figure 1.

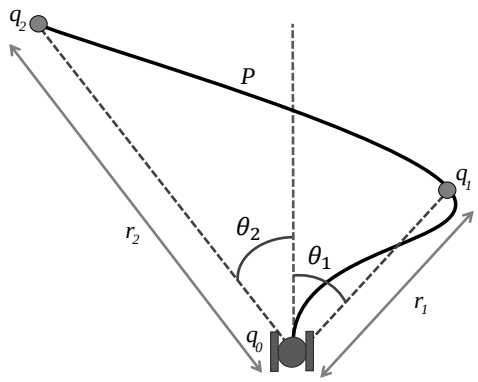

Fig. 1. Trajectory parameters.

The first four conditions are fixed so the vector of decision variables is calculated by the parameters used to define the position of the points q_1 and q_2:

$$\xi = \begin{bmatrix} \theta_1 \\ \theta_2 \\ r_1 \\ r_2 \end{bmatrix} \tag{2}$$

The search space of an optimization problem is the domain of the function to be optimized, in this case the domain is the set of values used to find the optimal solution, each set is defined by an upper and lower limits as follows:

$$\begin{aligned} \theta_1 &\in (-\phi_1, \phi_1) \\ \theta_2 &\in (-\phi_2, \phi_2) \\ r_1 &\in (\ 0\ \ , l_1) \\ r_2 &\in (\ 0\ \ , l_2) \end{aligned} \tag{3}$$

Where ϕ_k is the limit for the angle θ_k and l_k is the limit for the distance r_k.

From this section set notation will be used to describe the regions of the current map and to formulate the objective function.

The set of all the points that are part of the map is called M. In robotics a map is usually a description of an area which classifies every position $m \in M$ into three sets: the free space F, the obstacles O and the unknown area U. These definitions imply that $M = F \cup O \cup U$.

The task assigned to the robot is to explore a closed indoor area to build a complete map. A map is completed when all the points that can be reach by the robot belong to F and all the points with which the robot can collide belong to O.

The mapping process is iterative, the robot should move to different areas to classify the points using distance sensor or equivalent. The robot can not classify points behind an obstacle unless it moves around them.

Considering the mapping process, the optimal trajectory will be the one that drives the robot over a set of positions P which allows the robot to map unknown areas.

Based on the previous statement one might be tempted to define the optimization problem as the minimization of the distance between the current position of the robot and the unknown areas, however not every point in the map can be mapped, for instance all the points behind a wall can not be reached so they can not be mapped.

A new set should be defined to describe the points in U that can be mapped. These points are the ones in the border between F and U, denoted by B, considering the definition in equation 4.

$$B = \{u \in U \mid (\exists f \in F)[\ \|f - u\| \leq 1]\} \tag{4}$$

Because every point in M is a vector with coordinates (x, y), the operator $\|\cdot\|$ is used to represent the distance between two points using the euclidean norm.

It is proposed that the aptitude A_ξ of a trajectory defined by ξ is measured by the number of points in B that are in the vicinity of the points in P.

The vicinity set Λ_k is defined as all points in M whose distance to some point in P is less than or equal to λ_k, which defines the maximum distance allowed. This set is defined as follows:

$$\Lambda_k = \{m \in M \mid (\exists p \in P)[\ \|p - m\| \leq \lambda_k]\} \tag{5}$$

A graphical description of all the sets defined in this section is shown in figure 2.

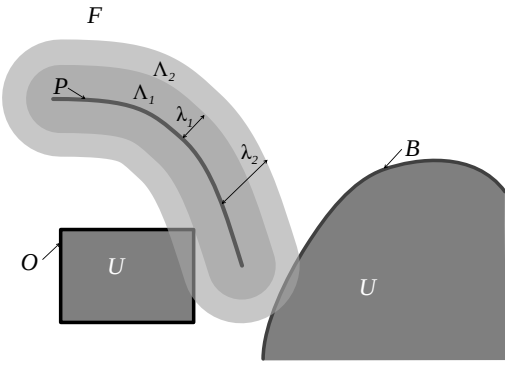

Fig. 2. Set definitions inside the map.

Considering all the previous definitions, the problem is formulated as a maximization of the aptitude A of a trajectory, calculated as the number of elements $\mathrm{n}(\cdot)$, in B that also belongs to the difference $\Lambda_2 \setminus \Lambda_1$. Aptitude or cost function is described by the following equation:

$$A = \mathrm{n}(\{b \mid b \in B \cap (\Lambda_2 \setminus \Lambda_1)\}) \tag{6}$$

As shown in figure 2 we consider $\lambda_2 > \lambda_1$, points in set Λ_1 are excluded from the cost function calculation because

the unknown areas may contain obstacles that have not been mapped yet and getting to close to the unknown zone may lead into a collision.

There are some restrictions that a feasible trajectory P must hold for security reasons. Three functions are defined to ensure that the robot will not collide to the obstacles.

It is important that all the points in a feasible trajectory P are located inside the free area F, this means that $P \subset F$. Therefore the level of transgression to this restriction R_1 is calculated as the number on elements in P that are not in F:

$$R_1 = \mathrm{n}(\{p \mid p \in P \setminus F\}) \tag{7}$$

It is also important that the elements in trajectory P are far enough to the obstacles to avoid collisions, the minimum distance is λ_1. Therefore the level of transgression to this restriction R_2 is calculated as the number of elements in O that are also in Λ_1:

$$R_2 = \mathrm{n}(\{o \mid o \in O \cap \Lambda_1\}) \tag{8}$$

It is inconvenient that the robot navigates too close to the border B since it could encounter obstacles that have not been mapped yet. Therefore the level of transgression to this restriction R_3 is calculated as the number of elements in B that are also in Λ_1:

$$R_3 = \mathrm{n}(\{b \mid b \in B \cap \Lambda_1\}) \tag{9}$$

It is possible to see that the cost function in equation 6, as well as the restriction functions in equations 7 8 9 depends on the path P which in turn depends on parameter vector ξ so finally the optimization problem can be formulated as follows:

$$\begin{aligned} &\underset{\xi}{\text{Maximize}} \quad A(\xi) \\ &\text{subject to:} \\ &\qquad R_1(\xi) \leq \ 0 \\ &\qquad R_2(\xi) \leq \ 0 \\ &\qquad R_3(\xi) \leq b_m \end{aligned} \tag{10}$$

Such that parameter vector ξ belongs to the search space defined by equation 3. The value b_m defines the upper limit value allowed for restriction R_3.

III. Optimization problem implementation

The platform used to implement the autonomous navigation task is a commercial robot called TurtleBot2 driven by a RaspberryPi3b computer running ROS Kinetic. A desktop computer is also used to monitor the progress of the navigation task as well as for visualize the current obtained map.

The robot uses a Microsoft Kinect as main sensor in order to measure the distance of the surrounding points to the robot.

By using the information from the Kinect the robot can obtain a partial map of the surrounding area by using a ROS program called *Gmapping* which in turn uses the distance to the objects and the measurement of the odometry to calculate the position of the objects in the map.

The map obtained by this process is called an occupancy grid [17] which is a vector that describes the occupancy probability of each point in the map in the range $(0, 100)$, an unknown point is labeled as -1.

The proposed algorithm creates a gray scale image based on the information in the occupancy grid. The obstacles O are drawn in black ($g = 0$), the free area F es drawn in withe ($g = 255$) and the unknown area U is drawn in gray ($g = 128$).

The set of borders B is calculated by finding all the pixels in U which are next to a pixel in F using an eight directions connectivity criterion. Figure 3 shows the process of finding the pixels in B, these points are drawn in dark gray ($g = 90$).

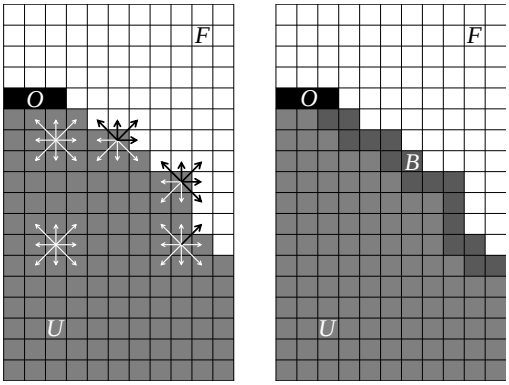

Fig. 3. Connectivity criterion to define border set B

Figure 3 shows connectivity of pixels in U to pixels in F with black arrows in the left image, pixels found in B are show in dark gray in right image.

The vicinity sets Λ_1 and Λ_2, are calculated by using the well known *dilate* algorithm, this process is intended to be used on a binary image, so a new image is created where the elements of P are drawn in withe ($g = 255$) and all the other pixels are drawn in black ($g = 0$), the new image has the same dimensions as the original map.

The dilate algorithm is implemented by replacing the value of each pixel with the maximum value in a mask centered in each pixel, the size of the mask λ define the dilation degree.

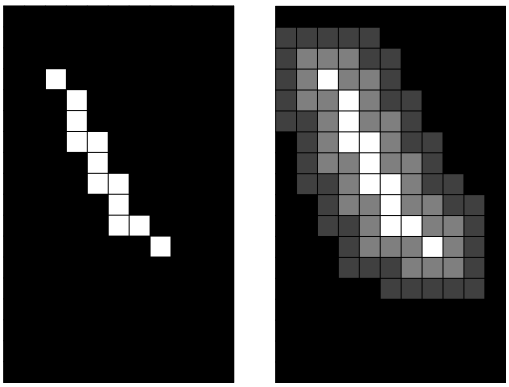

Fig. 4. Dilatation process used to find vicinity sets Λ

Figure 4 shows the original image in the left, pixels added by using dilation process are shown in gray in the left image, using $\lambda = 1$ and $\lambda = 2$, the gray scale values are only for

demonstration purposes, the points added to set Λ and the original points in P will have the same value.

The discrete map represented by an image obtained by using the processes described in this section is shown in figure 5 as the discrete version of the map shown in figure 2.

In the actual implementation sets P and Λ_k are saved in different images apart from the original map, the reason is that each pixels in M belongs to only one of the sets F, O, U and B, this means that these sets are disjoint, so it is possible to describe the membership to any of these sets by a different gray scale in the same image without ambiguity, however any position in M can also be in P or Λ_k so is not possible to draw these sets in the same image. Figure 1 and 5 shows all the sets in the same image only for demonstration purposes.

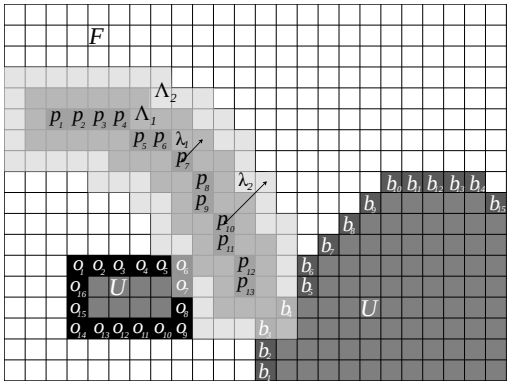

Fig. 5. Discrete map.

Considering the images defined in this section is possible to evaluate the membership of an element in M by comparing the gray scale values of the pixels in the corresponding images. By using this method is possible to evaluate the expressions in the problem formulation of the equation 10.

IV. Optimization methods

Classical optimization methods require to use the information of the derivative of the cost function. In this case the cost function defined by equation 6 represents the number of elements in a set, which is calculated using the numerical method described in section III, this means that there is no explicit function of the derivative.

This kind of optimization problems can be solved by using the approach of the heuristic methods, usually a heuristic is intuitive technique used to get an approximate solution to a problem when classical methods failed or are very slow.

When using heuristic methods there is no need to guarantee that the objective function is convex, differentiable or even continuous because this methods rely only in function evaluations. The tradeoff is that heuristics can not guarantee asymptotic convergence.

This section provides a brief description of the three heuristic techniques used to solve the optimization problem previously formulated.

A. Blind search method

The blind search method can be described as a sequence of iteration based on two operations: solution generation, which is the program that produces a candidate solution in the search space using random processes, the update procedure is the rule used to replace the current accepted solution by the candidate solution when it has a better cost function.

The method ends when some criterion is met, usually a maximum number of evaluations or when the solution is not improved any further over the last iterations.

This method has the advantage that the probability to find the optimal solution increases over each iteration, the disadvantage is that the convergence is slow and is possible that the error is big when not enough evaluations of the cost function are used. This method sometimes is called uninformed search because it does not use any information of the domain of the function.

B. Hill climbing method

The hill climbing method is an extension of the blind search but the difference is that the solution generator will not produce a single candidate solution but a collection of solutions around the current accepted solution.

Another difference is that the search space is contracted each iteration using some deterministic rule, for instance the limits of the intervals can be moved closer to the accepted solution based on some factor.

In the update procedure the accepted solution is replaced by the best solution among all the candidate solutions generated during each iteration. The idea behind the hill climbing method is to approximate the gradient of the function numerically.

The behavior of this method is described by two parts: the exploration stage which occurs during the first iterations: the search space is big enough to cover all the feasible solutions. The approximation stage occurs during the last iterations, when the search space is getting smaller, the purpose of this stage is to increase the accuracy of the accepted solution.

The main advantage of the hill climbing method over the blind search is that it produces more accurate results when the cost function is smooth. It has the disadvantage that the algorithm can get stock in local optimal solutions.

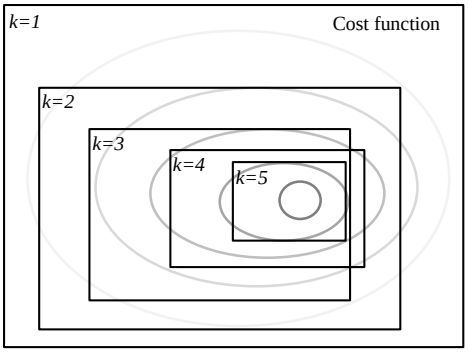

Fig. 6. Search space contraction.

C. Genetic algorithm

A genetic algorithm (GA), proposed by John Holland [18], is based on the natural selection process, the cost function is associated to the aptitude of each solution, its applications include solving multi objective optimization problems [19], machine learning, pattern recognition, prediction, etc.

The solutions are coded in the form of binary chains. GA are population methods, this means that several solutions are created over all the search space, each one of them moves to the optimal solution independently.

In a GA a solution represents an individual, iterations represents generations which create a new offspring by using the information of the previous population by using mathematical operators inspired in natural processes:

- Evaluation: This tool shows the relation between the phenotype and genotype. In this process the binary chains are converted to solutions to evaluated the cost function.
- Selection: This step represents the competition of the individuals to be able to reproduce, the process involves the comparison of the individual's aptitude to choose the better ones to create a population of parents.
- Crossover: This process generates the offspring by combing the chains of the parents selected in the previous step.
- Mutation: This tool is inspired in the random changes that occurs in biological organisms, it is implemented by randomly change the value o some genes using a probabilistic distribution function.

At the end of each generation the population of the offspring replace the population of the previous individuals.

An advantage of GA over the previous methods is the parallel exploration which usually end up finding the optimal solution with high accuracy. The disadvantage is that the performance of the method depends on the problem and the proper selection of the parameters for the implementation.

V. Comparison of optimization methods

The efficiency comparison of the different algorithms implemented were carry on in real tests on the commercial robotic platform called TurtleBot2 using a RaspberryPi3b embedded computer running Ubuntu 16.04 Xenial and ROS Kinetic.

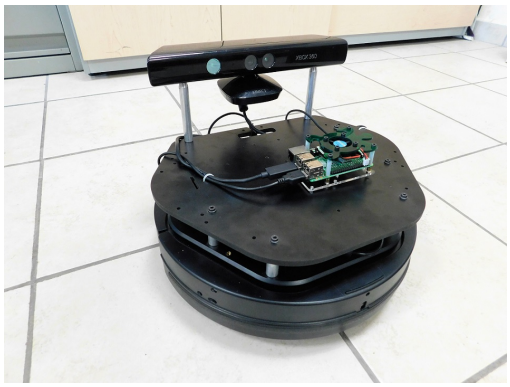

Fig. 7. Experimental platform

The use of random variables in heuristic techniques implies that every time the algorithms is executed a different solution will be obtained. On way to measure the performance of an heuristic method is to execute the program several times and calculate the average solution and the variance of the results.

The objective of the mapping task is to explore all the zones in a closed area, the task will be completed when the set B, as defined by equation 4, is empty. Then it is proposed that the efficiency measurement of the algorithms is $n(B)$ after a number of iterations is executed in each algorithm.

After 30 experiments performed, the GA results with a lower average since the variance of the graph shows a trend to five, which is less compared to the hill climbing algorithm which has an average of eight, or the blind search with an average of ten, for this reason it is known that the genetic algorithm is the best in this comparison, see figure 8.

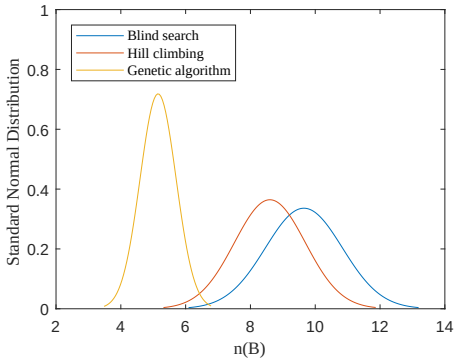

Fig. 8. Comparison using independent executions

It is necessary to remember that the mapping process is a repetitive sequences of steps alternating between trajectory generation and motion, this means that a suboptimal trajectory chosen at some point would affect all the following process.

There are many factors than can induce error in the mapping process, for instance, a delay in the mapping generation, changes in illumination, poor approximation of the objects position or odometry, etc. This sources of error are independent from the optimization algorithm used.

Then a second efficiency measurement is proposed, all the algorithms will be executed with the same information from the sensors of the robot, the best trajectory will be followed by the robot and we will compare the algorithms based on how many times each algorithm produce the best solution during a single mapping process.

In figure 9 the results of the second experiment are observed, which shows the average number of times that one algorithm was better than another during the same iteration and the same conditions in the execution of the mapping. In this case it is observed that the GA was selected an average of nine times during the execution of the mapping, which indicates that it is better in comparison with the hill climbing that was selected twice and the blind search that was only selected once.

The results shown in figures 8 and 9 were obtained by using 30 experiments of each algorithm. The comparison were made

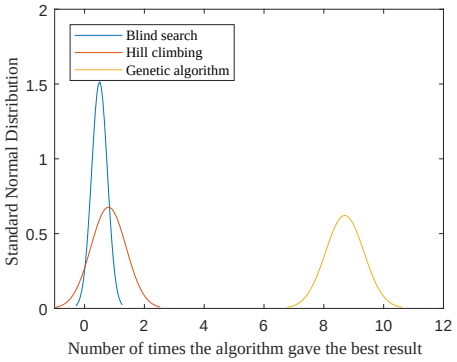

Fig. 9. Comparison using the same execution

using a maximum number of evaluation $M_{eva} = 100$ for all the algorithms, in the case of the hill climbing method, the evaluations were divided in $N_{it} = 10$ number of iterations using $N_{ev} = 10$ number of evaluations per iteration.

The genetic algorithm were implemented using binary tournament selection, two points crossover with rate $C_r = 0.9$ and uniform mutation with rate $M_r = 0.9$.

VI. Conclusion and Future Works

The results of both tests show that the algorithm with the lowest number of points in the border and the one that generates better results in more occasions is the GA. However, overlaps can be observed in probability functions, which means that sometimes one of the other algorithms can be better, although it is unlikely.

Sometimes the robot lost its location due to faults of the sensors, so the map was overloaded and routes were generated in places different to the position of the robot, which caused it to collide. On the other hand, this same problem could also prevent the map from being updated, so the algorithm was not able to find suitable routes, causing the robot to stay spinning.

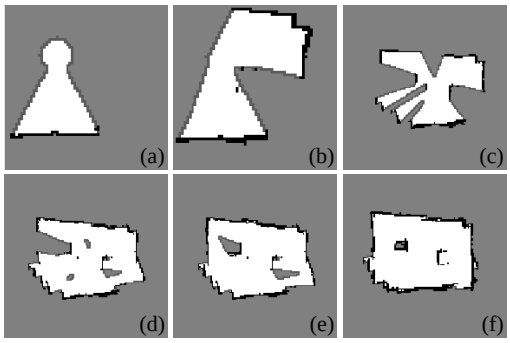

Fig. 10. Creation of the map

Figure 10 illustrates the creation of the map in six images, in figure 10.a it is shown the initial map and position of the robot, this is taken as starting point to generate the trajectories, figures 10.b-10.e shown intermediate points that complete the map, which are added as the robot makes decisions to rotate or move through the free zone, finally figure 10.f shows the completed map after several iterations of this process.

As future work we intend to implement other algorithms such as the rapidly exploring random trees [20], potential fields [21] and Simplex method of Nelder-Mead [22], which will allow us to perform a new analysis and observe its behavior through different perspectives in the solution of these problems.

References

[1] J. Trevelyan, W. R. Hamel, and S. C. Kang, "Robotics in hazardous applications," *Springer Handbook of Robotics*, 2016.

[2] M. Chui, J. Manyika, and M. Miremadi, "Four fundamentals of workplace automation," *McKinseyQarterly*, November 2015.

[3] G. Graetz and G. Michaels, "Robots at work," *Discussion paper series*, vol. IZA DP No. 8938, March 2015.

[4] R. Nakadate and M. H. J. Arata, "Next-generation robotic surgery - from the aspect of surgical robots developed by industry," *Center for Advanced Medical Innovation*, 2015.

[5] P. L. na, J. Blanco, K. Bunda, A. Cruz, A. Ramirez, and A. Bandala, "Swarm algorithm implementation in mobile robots for security and surveillance," *TENCON 2014 - IEEE Conference*, pp. 1–5, 2014.

[6] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Cambridge University Press 2014*, vol. 33, pp. 463–497, 2015.

[7] W. Yu, E. Zamora, and A. Soria, "Ellipsoid slam: a novel set membership method for simultaneous localization and mapping," *Autonomous Robots*, vol. 40, pp. 125–137, January 2016.

[8] L. Capito, P. P. no, O. Camacho, A. Rosales, and G. Scaglia, "Experimental comparison of control strategies for trajectory tracking for mobile robots," *Int. J. Automation and Control*, vol. 10, pp. 308–327, 2016.

[9] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," *Springer International Publishing Switzerland*, pp. 455–472, 2017.

[10] J. C. Elizondo, G. Ramírez, E. Rodriguez, and J. R. Martínez, "Multi-robot exploration using self-biddings under constraints on communication range," *IEEE Latin America Transactions*, vol. 14, February 2016.

[11] R. M. Pierce, "Increasing transparency and presence of teleoperation systems through human-centered design," *Publicly Accessible Penn Dissertations 1947*, January 2015.

[12] H. Duan and P. Qiao, "Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning," *International Journal of Intelligent Computing and Cybernetics*, vol. 7, pp. 24–37, 2015.

[13] Y. Quinonez, I.Tostado, and C. Burgueno, "Aplicación de técnicas evolutivas y visión por computadora para navegación autónoma de robots utilizando un turtlebot 2," *RISTI [online] n.spe3*, pp. 93–105, 2015.

[14] Y. Liang, F. Hong, Q. Lin, S. Bi, and L. Feng, "Optimization of robot path planning parameters based on genetic algorithm," *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 529–534, 2017.

[15] M. S. Ganeshmurthy and G. R. Suresh, "Path planning algorithm for autonomous mobile robot in dynamic environment," *2015 3rd International Conference on Signal Processing*, pp. 1–6, 2015.

[16] B. Song, Z. Wang, and L. Sheng, "A new genetic algorithm approach to smooth path planning for mobile robots," *Assembly Automation*, vol. 36, pp. 138–145, 2016.

[17] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[18] J. H. Holland, "Complex adaptive systems," *Daedalus, JSTOR*, vol. 121, pp. 17–30, 1992.

[19] C. Coello, D. V. Veldhuizen, and G. Lamont, "Evolutionary algorithms for solving multi-objective problems," *Genetic Algorithms and Evolutionary Computation*, 2002.

[20] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," *Iowa State University*, 1998.

[21] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," *IEEE International Conference on Robotics and Automation, Sacramento*, vol. 2, pp. 1398–1404, 1991.

[22] P. Wang and T. Shoup, "Parameter sensitivity study of the neldermead simplex method," *Advances in Engineering Software*, vol. 42, pp. 529–533, July 2011.