International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 19 (2018) pp. 14273-14281

Scara Robot Path Planning Through flood fill Algorithm

[1]Julián Esteban Herrera-Benavides, [2]Cesar Giovany Pachon-Suescun, [3]Robinson Jimenez-Moreno
[1]Mechatronics engineer, Department of Mechatronics Engineering, Nueva Granada Military University,Bogotá, Colombia
[2]Research Assistant, Department of Mechatronics Engineering, Nueva Granada Military University,Bogotá, Colombia
[3]Professor, Department of Mechatronics Engineering, Nueva Granada Military University,Bogotá, Colombia,
Email id: julian-herrera1@upc.edu.co, u3900259@unimilitar.edu.co, robinson.jimenez@unimilitar.edu.co

Abstract

This document presents one of the most important algorithms in the mobile robot path planning (Flood Fill) to define the trajectory of a Scara robot, which is simulated in the Matlab software. The deduction presented in this document is intended to be applied to the principal robots that can be found in the industry, generating paths that aims to reduce the displacement, taking into account the presence of obstacles and physic restrictions. The results obtained show that it is possible to calculate the way to any point of the space work, as long as it is physically possible.

Keywords: *Flood fill, Homogeneous matrix, Scara robot, Orthogonality, coordinate axes, Inverse kinematics, Direct kinematics, Rigid body.*

INTRODUCTION

One of the most important topics in robotics is to determine the trajectory [1-2] that a robot must follow so that it can perform a specific task. In the case of mobile robots, those that do not have a fixed link, the Flood Fill algorithm is usually implemented [3-4]. Algorithm that allows the planning of trajectory from one place to another, taking into account that it must avoid obstacles.

The Flood Fill algorithm is a great tool that not only seeks to avoid obstacles, it calculates the shortest path possible taking into account that the displacements of the mobile agent are made by vertical and horizontal movements. In [5-7], it is possible to find the detailed explanation of how this algorithm works and how it has been implemented for the Micro mouse competitions, in which it is desired to calculate trajectories that solve a labyrinth in the shortest possible time.

On the other hand, it is possible to find robots that have links that are fixed with respect to a surface. These robots, in the simplest cases, are analyzed through direct and inverse kinematics, methods that involve the homogeneous transformation [8] to determine the location of the final actuator [8-9] and trigonometric relations, for the simplest robots, to determine the generalized coordinates that lead the system to the desired location. Among the most representative robots are the Cylindrical, Anthropomorphic, Spherical and Scara. Each of them has the advantage of having an inverse kinematics which is analytic, i.e. its solution is direct and comes from an equation that is analyzed from trigonometric relationships.

An example that illustrates the above is shown in [10], where the path planning of a food manipulator robot is calculated by analyzing the direct and inverse kinematics of this. The direct kinematics allows the programmer to obtain the position of the final link with respect to the reference plane, while with the analysis of the inverse kinematics, it allows to determine the value that the generalized coordinates must have to reach the desired point.

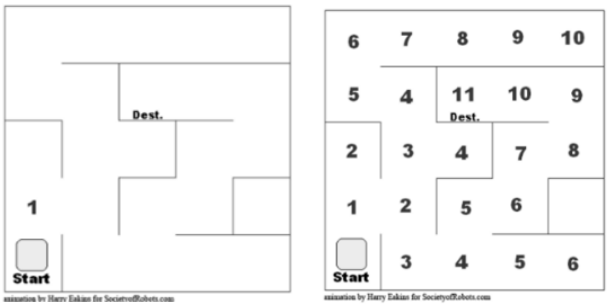
This document explains the way in which the Flood Fill algorithm, the direct kinematics and the inverse kinematics of the fixed robots can be combined to produce an algorithm that determines the values that the generalized coordinates must take to transfer to the final actuator, in the shortest way, to the desired point, taking into account that the displacements of the final actuator are given by translations, which, by movement, can only have one component in any of the three coordinate axes. This algorithm makes the planning of the system's trajectory avoiding that it collides with obstacles. At the same time, it takes into account the physical constraints found in the mechanism, such as the maximum displacements in the rotational links.

This article is divided into 5 sections. Section 1 explains the flood fill algorithm and the modification that is made to implement it in the Scara robot. Section 2 shows the tools used in robotics and analyzes how it is possible to discretize the workspace of a robot, in which the modified flood fill algorithm is implemented. Section 3 implements the modified flood fill algorithm in the Scara robot. Section 4 shows the results and section 5 analyzes the conclusions of the investigation.

THEORETICAL FRAMEWORK

The Flood Fill algorithm is developed in a matrix which represents a continuous environment where the mobile agent is located. In said matrix, each cell is a position in which the robot can be housed. The only restriction that exists to be able to reach any cell is to avoid trespassing the borders that have been referenced as an obstacle.

The cells of the matrix must be created in such a way that the robot can be located in the center of these. So it can be deduced that the matrix, which represents the workspace, is created according to the size of the mobile agent. The obstacles are located on the walls of the cells, which can be seen in Figure 1.a. In order to perform the trajectory planning of the mobile agent, the first thing to do is assign a value equal to zero to the initial position of the robot (start). The other cells must contain a value equivalent to the number of steps necessary to be reached by the robot (see Figure 1b).



a) b)

Figure 1. a) Matrix representation of a space. b) Filling the matrix [6].

Suppose that there is a world whose matrix representation is given by Figure 1a. The start of the robot is in "Start" and its first movement can only be vertical, due to the obstacles that surround it. Once the robot generates its first movement, it marks the position of said cell with a value equal to 1, because it is its first movement.

If this is repeated iteratively taking into account the other obstacles, a value can be assigned to each cell of the matrix as illustrated in Figure 1.b).

After eleven iterations, it is noted that the "destination" cell has been reached. This means that it is time to find the trajectory from the value assigned to each cell of the matrix. Figure 2 illustrates this step.

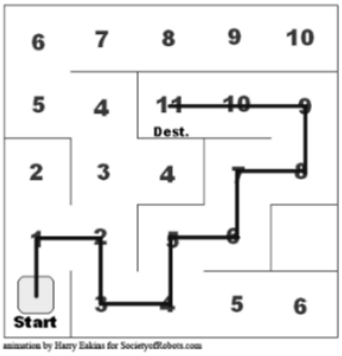


Figure 2. Solved trajectory of the Flood Fill.

In order to solve trajectories of a fixed robot, such as the Scara robot, it is necessary to modify the Flood Fill algorithm in order to allow to evaluate the obstacles as information inside the cells of the matrix. In other words, the obstacles will not be included in the ends of the cell, but as a data that is stored internally.

In Figure 3a, a world is presented where the red object must reach the green object, avoiding colliding with the blue objects. If the position of each object is discretized and enclosed in cells, it is possible to create a matrix that contains information about the obstacles, the robot and the desired point (See Figure 3b).

a) b)

Figure 3. Continuous world and its discrete representation.

The positions of the robot are stored with a value of -2, that of obstacles with a value of -1 and that of the destination with a value of -3.

To solve the path of the robot (red object), all the values -2 are eliminated from the matrix except one that will be used as reference (See Figure 4).

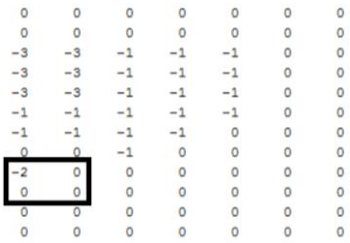


Figure 4. The ends of the robot in the matrix are eliminated except the reference value used.

Now, it is proceeded to perform the filling of the matrix in the same way as explained in the Flood Fill, except that now the collisions must be evaluated by an intersection of the ends of the robot in cells marked with the value of -1.

After 19 iterations it is observed in Figure 5 that the robot comes into contact with cells marked with the value of -3, which suggests that the desired point has been reached. All the cells that have remained with a value of zero represent a position to which the reference point cannot reach, since the ends of the robot would collide with obstacles or leave the matrix.

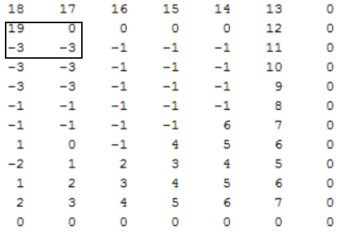


Figure 5. Arrival of the robot to the desired point.

Next, it is proceeded to determine the trajectory of the robot in the same way as it was done in the Flood Fill, resulting Figure 6.

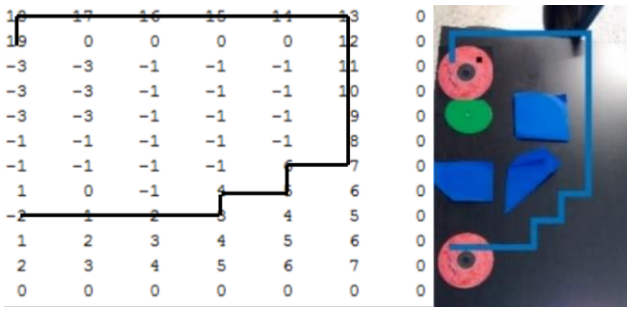


Figure 6. Solution to the trajectory of the mobile robot.

The steps implemented, which led to Figure 6, are indispensable for understanding the trajectory planning of a fixed robot. From the previous algorithm, it is observed that the first step was to discretize the workspace in a matrix, which contains the location of the robot, the obstacles and the desired position. The following section explains how to do it for fixed robots.

DISCRETE REPRESENTATION OF THE OPERATIONAL ENVIRONMENT OF A FIXED ROBOT

Before analyzing the discretization of the operating environment of the robot, the robotics tools that are implemented in this process are mentioned.

The homogeneous matrix, symbolized as T_E^W, allows finding the location of a point, which is seen from the coordinate system *E*, with respect to the base coordinate system *W*. The homogeneous matrix internally contains information about the orientation and location of system *E* with respect to *W*. This tool is quite useful to determine the direct kinematics of the mobile robot. In addition, it allows finding the location of the links with respect to the base system, in which the fixed link is located.

To determine the location of a point, which is seen from the system E, with respect to the system W, it is necessary to do the operation of equation 1.

$$\boldsymbol{P_w = T_E^w \cdot P_E} \qquad (1)$$

Equation 1 implements the homogeneous matrix to transfer the points of the coordinate system *E* to the coordinate system *W*. If the points represent the shape of the links, it is possible to use this same relationship to be able to represent all the points with respect to the base system.

Figure 7 shows the coordinate axes of each link for the Scara robot. It is known that the points, which represent each link, are seen with respect to the origin of each link. So to transfer all the points with respect to the base system, it is necessary to use the homogeneous matrices and implement equation 1.

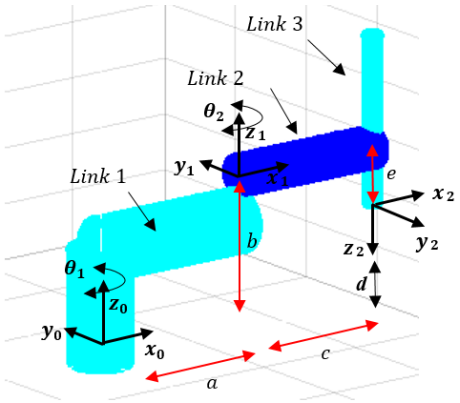


Figure 7. Scara robot and coordinate systems.

From Figure 7 it is also possible to observe the type of movement that each link experiences. Two rotational and one prismatic links are available, the combination of these three articulations allows the final actuator to be located at a point in space, so the generalized coordinates are:

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\theta_1} \\ \boldsymbol{\theta_2} \\ \boldsymbol{d} \end{bmatrix}$$

To locate all the points that represent the shape of each link, with respect to the base system, equations 2 to 4 must be implemented.

$$\boldsymbol{p_{link1_0} = T_1 \cdot p_{link1}} \qquad (2)$$

$$\boldsymbol{p_{link2_0} = T_2 \cdot p_{link2}} \qquad (3)$$

$$\boldsymbol{p_{link3_0} = T_3 \cdot p_{link3}} \qquad (4)$$

Link 1 is only rotating with respect to the z axis of system "0". So the homogeneous matrix one obtains the form presented in equation 5.

$$\boldsymbol{T_1} = \begin{bmatrix} \boldsymbol{c(\theta_1)} & \boldsymbol{-s(\theta_1)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{s(\theta_1)} & \boldsymbol{c(\theta_1)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \qquad (5)$$

For link 2, its rotation with respect to the z-axis of the coordinate system "1", the rotation of link 1 with respect to the coordinate system "0" and the distance at which the coordinate system "1" is with respect to "0" are taken into account,

resulting equation 6.

$$T_2 = T_1 \cdot \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

The homogeneous transform 2, equation 6, indicates that there is a rotation on the z_0 axis, then a translation to the system 1 and finally a rotation on the z_1 axis. If the final link is analyzed, equation 7 is obtained:

$$T_3 = T_2 \cdot \begin{bmatrix} 1 & 0 & 0 & c \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -e-d \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\pi & -s\pi & 0 \\ 0 & s\pi & c\pi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

In this case it is assumed that *d* moves positively in the negative direction of z_1.

Once all the homogeneous transformations are obtained, it is possible to locate all the points with respect to the "0" system. To determine the location of the final end of link 3, to which the final actuated is adjusted, equation 4 should be implemented taking into account that the point no longer refers to the shape of the prismatic link, but to the location of the final actuator with respect to the coordinate system 2. Resulting in equation 8.

$$p_{final\ actuator_0} = T_3 \cdot p_{final\ actuator} \quad (8)$$

It is assumed that the path to be calculated will avoid obstacles, therefore, it is necessary to include the obstacles and determine the respective homogeneous matrices. In this case it is assumed that the obstacles are seen with respect to the coordinate system "0", so it is not necessary to find a homogeneous matrix (see equation 9).

$$p_{obstacles_0} = p_{obstacles} \quad (9)$$

Now that it is possible to obtain the points with respect to the "0" system, it can be entered any value to the generalized coordinates and graph the results in a software, and thus see the operating environment of the robot and its location in it. The result is shown in Figure 8.

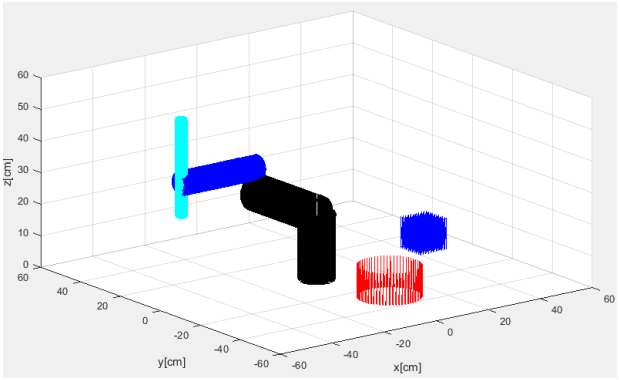


Figure 8. Operating environment of the robot.

The next step is to discretize the operating environment in a matrix that contains the values of the starting point, the obstacles and the end point with the values -2, -1, -3, respectively.

For the above, a new coordinate system must be created that is located at one end of the robot workspace, taking into account that the orientation of the plane directs the coordinate axes towards the robot origin, as illustrated in Figure 9.

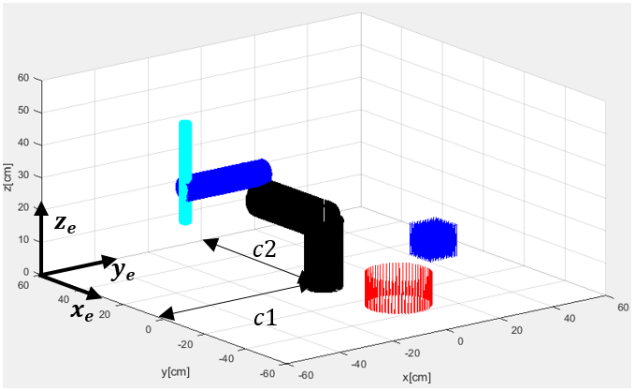


Figure 9. Creation of a coordinate system at one end of the robot's operating environment.

The homogeneous matrix that relates the "0" system to the "e" has been written in equation 10.

$$T_0 = \begin{bmatrix} c\left(\frac{\pi}{2}\right) & -s\left(\frac{\pi}{2}\right) & 0 & 0 \\ s\left(\frac{\pi}{2}\right) & c\left(\frac{\pi}{2}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & c_1 \\ 0 & 1 & 0 & -c_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

The location of the system "e" in the matrix must be the position (1,1,1), and the location of the system "0" will be determined with the relationship that exists between a cell and the displacement unit implemented. Figure 10 allows to see the relationship between the cells of the matrix that represents the

discretized space, and the displacement in the system "e".

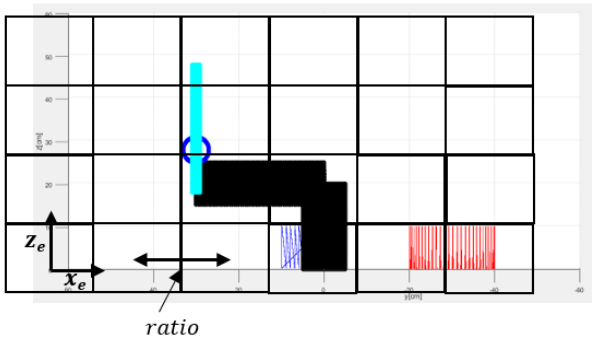


Figure 10. Relationship between the system "e" and the matrix that represents the discretized space.

For the three axes, there are equations 11-13:

$$(f-1)\cdot relation = x_e \quad (11)$$

$$(c-1)\cdot relation = y_e \quad (12)$$

$$(h-1)\cdot relation = z_e \quad (13)$$

The variables "f", "c" and "h" are the row, the column and the height in the matrix, respectively. For this case, the rows increase in the direction of the $\boldsymbol{x_e}$ axis, the columns in the direction of the $\boldsymbol{y_e}$ axis and the height in the direction of the $\boldsymbol{z_e}$ axis.

Using equations 9 and 10, it is possible to transfer the points, which represent obstacles, to the coordinate system "e" (see equation 14).

$$\boldsymbol{p}_{obstacles_e} = \boldsymbol{T_0}\cdot\boldsymbol{p}_{obstacles_0} \quad (14)$$

For the case of the links and the terminal point of the robot, there are equations 15-18:

$$\boldsymbol{p}_{link1_e} = \boldsymbol{T_0}\cdot\boldsymbol{T_1}\cdot\boldsymbol{p}_{link1} \quad (15)$$

$$\boldsymbol{p}_{link2_e} = \boldsymbol{T_0}\cdot\boldsymbol{T_2}\cdot\boldsymbol{p}_{link2} \quad (16)$$

$$\boldsymbol{p}_{link3_e} = \boldsymbol{T_0}\cdot\boldsymbol{T_3}\cdot\boldsymbol{p}_{link3} \quad (17)$$

$$\boldsymbol{p}_{final\ actuator_e} = \boldsymbol{T_0}\cdot\boldsymbol{T_3}\cdot\boldsymbol{p}_{final\ actuator} \quad (18)$$

Then, by means of equations 11 to 13, the position where each point is located in the matrix can be determined, i.e. since all the points are already transferred to the "e" system, it can be used the calculated coordinates to enter said data in equations 11 to 13, and thus find the equivalent to the row, the column and the height in the matrix. The positions in the matrix must be an integer, so the results obtained must be approximated to an integer value.

Once obtained the equivalent positions of the obstacles and the final actuator in the matrix, it is proceeded to mark the cells with the values -1 and -2 respectively. The value wanted to reach is chosen randomly by marking any cell with the value of -3. To see the result of this step, it is necessary to show how it has been the matrix, but, because it is dealing with a three-dimensional one, it is not possible to show the values that each cell has. To solve this, the values of row, column and height are used as coordinates to generate a graph that adequately shows the data. The black objects are the obstacles, the red object is the starting point and the green one is the point that is wanted to reach (see Figure 11).

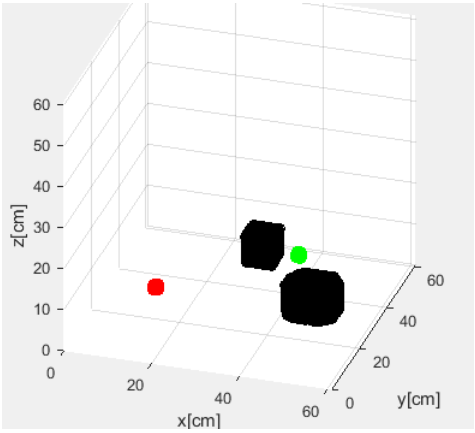


Figure 11. Operating environment of the discretized robot.

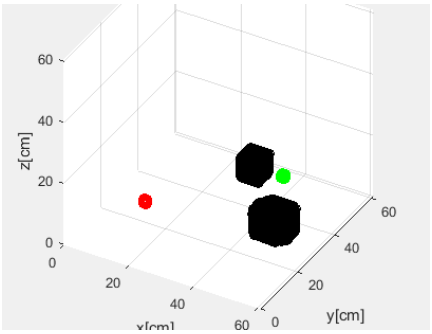
Figure 11 is the discrete representation of the environment shown in Figure 8. At this time, the same data as is needed in the modified flood fill is available, so the next step is to fill the matrix.

Before continuing, it is necessary to bear in mind that robots like Scara have the ability to reach a point in two different ways, since their inverse kinematics gives two possible solutions. This means that at the time of filling the matrix it is not only necessary to take into account the number of movements necessary to reach a cell, but the way in which the robot has arrived. The type of arrival can be differentiated by the angle formed by link 2 with respect to 1. If the angle θ_2 is

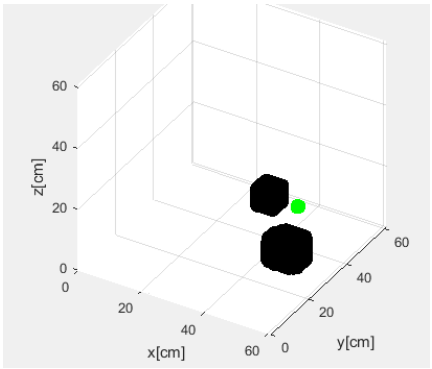
greater than or equal to 0° and less than or equal to 180° , it will be taken as the arrival solution 1 and if it is greater than or equal to 180° and less than or equal to 360°, it will be taken as the arrival solution 2. At points where the angle is 0 or 180 the solutions are equal.

MODIFIED FLOOD FILL APPLIED TO SCARA ROBOT

Keeping in mind the existence of two solutions for the same point, it is chosen to implement two matrices. The value that these two matrices take for the state in which the robot is found in Figure 8 is as shown in Figure 12.



a)



b)

Figure 12. a) Matrix for arrival solution 1. b) Matrix for arrival solution 2.

Figure 12a has the green point that represents the start of the robot (start), because the link two meets an angle that is between 0 and 180 degrees. For this reason Figure 12b, only has the points that represent the obstacles and the desired position.

The next step is to perform the filling of the matrices by evaluating the collisions of the robot with the obstacles. In the case of the modified flood fill algorithm, a movement is generated in all directions and it is evaluated whether the new position is an obstacle or if the new position of the robot's perimeter is located in an obstacle. If there are no collisions, the cell is marked with the steps that were taken to reach it. For the Scara Robot, the same thing must be done, only now there is not a square figure. In order to determine if there is a collision, the inverse kinematics of the Scara robot and equations 11 to 13 and 15 to 17 must be implemented. With equations 15 to 17, the robot is represented with respect to the coordinate system "e" and by means of equations 11 to 13, the cells occupied by the robot inside the matrix can be obtained.

Before continuing, Figure 4 should be reviewed again. This represents the initial state of the matrix for a workspace. If two iterations are generated in the displacement of the robot, the result shown in Figure 13 is obtained.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
-3	-3	-1	-1	-1	0	0
-3	-3	-1	-1	-1	0	0
-3	-3	-1	-1	-1	0	0
-1	-1	-1	-1	-1	0	0
-1	-1	-1	-1	0	0	0
1	0	-1	0	0	0	0
-2	1	2	0	0	0	0
1	2	0	0	0	0	0
2	0	0	0	0	0	0
0	0	0	0	0	0	0

Figure 13. Second movement of the mobile robot in the situation of Figure 4.

It can be seen that there are some cells of the matrix that have not acquired the steps that the robot has given, however, there is a cell to which the number 2 has not been associated. This cell can be seen inside the black square that represents the mobile robot. It is not possible to assign a value to this cell because the perimeter of the robot comes into contact with a cell that has the value -1, which means that the mobile agent would collide with an obstacle. In order to sort all these steps in an algorithm and then edit it to create the one of the Scara robot, a pseudocode has been made that explains the modified Flood Fill algorithm.

Step 1: Initialize an integer variable with a value equal to 0.

Step 2: Locate the cells that contain the value of the variable. Only for the first search, the cell with the value -2 must be found. If no cell with the desired value is found, the program must be terminated.

Step 3: Once the cells are located, a displacement must be generated, from one cell, in all the senses in each one of them. Only the cells that do not experience a collision and that have a value equal to zero must be marked with the value $(variable + 1)$. If the new position is the cell with the value -3 or the perimeter of the robot in that new position comes into contact with the cell marked -3, without creating a collision, it means that the solution has been reached and that the iterations must be eliminated, going to step 5.

Step 4: After having evaluated all the cells with the value of the

variable, this is increased by a value of 1 ($variable = variable + 1$) so that in the next iteration the new cells are searched. Then go back to step 2.

Step 5: It must be located in the position in which the end of the loop was determined and saves that position in a variable that stores the trajectory by means of the values row, column and height (f, c, h). Then, it must be moved to a cell that has been marked with a lower value than the current one and save that position in the variable that stores the trajectory. This must be repeated until reaching a cell that contains the value of 1.

In the case of the Scara robot, it must be remembered that two matrices are available and that each of them represents the movement of the robot with the "arrival solution 1" and the "arrival solution 2". It is known that these matrices represent the same point only when the generalized coordinate θ_2 equals 0 or 180 degrees. The discretization of the environment and the displacement of the robot will make it often impossible to obtain these two values and therefore the two matrices cannot be related. To avoid this, a range of values is considered in which the generalized coordinate will cause a point to be represented in the same way by the two matrices. For this case, a range of 10 degrees in a positive and negative direction is considered. The algorithm for the Scara robot is as follows:

Step 1: Initialize an integer variable with a value equal to 0.

Step 2: Locate the cells of the "solution 1" matrix that contain the value of the variable. Only for the first search the cell with the value -2 must be found. If no cell with the desired value is found, proceed immediately to step 4.

Step 3: Once the cells are located, in the matrix "solution 1", a displacement must be generated, from one cell, in all the directions in each one of them. Only the cells that do not experience a collision and that have a value equal to zero must be marked with the value ($variable + 1$).

If the evaluated cell has been marked, the value of the generalized coordinate θ_2 is examined. If this value is in the implemented range, it is determined what value the "solution 2" matrix has in that position. If the value is zero and no collision is experienced, the same position is marked in the "solution 2" matrix.

If the new position is the cell with the value -3 and a collision was not generated, it means that the solution has been reached and that the iterations must be eliminated, going to step 6.

Step 4: Locate the cells of the "solution 2" matrix that contain the value of the variable. Only for the first search the cell with the value -2 must be found. If no cell with the desired value is found and just before step 2 a position was not found either, the program must be terminated.

Step 5: Once the cells are located, in the matrix "solution 2", a displacement must be generated, from one cell, in all the directions in each of them. Only the cells that do not experience a collision and that have a value equal to zero must be marked with the value ($variable + 1$).

If the position has been marked, the value of the generalized coordinate θ_2 is examined. If this value is in the implemented range, it is determined what value the "solution 1" matrix has in that position. If the value is zero and no collision is experienced, the same position is marked in the "solution 1" matrix.

If the new position is the cell with the value -3 and a collision was not generated, it means that the solution has been reached and that the iterations must be eliminated, going to step 6.

Step 6: After having evaluated the value of the variable, the variable is incremented by a value of 1 ($variable = variable + 1$) so that in the next iteration the new cells are searched. Then, return to step 2.

Step 7: It should be located in the position and the matrix in which the end of the loop was determined, and save that position in a variable that stores the trajectory through the values row, column, height and "type of solution" (f, c , h, T). Then you must move to a cell that has been marked with a lower value than the current one and save that position in the variable that stores the trajectory. This must be repeated until reaching a cell that contains the value of 1.

If in the process of displacement a cell containing a lower value than the current one is not found, it means that it is time to start decreasing in the same position, but in the opposite matrix, so the variable "T" that stores the type of solution must indicate it. This variable can take values such as "1" or "2". Indicating with the first value that refers to the "solution 1" matrix.

Steps three and five should examine whether it is possible to reach a point by evaluating whether the robot hits an obstacle. In order to determine this, the following must be done:

It is wanted to know if it is possible to mark the cell (f_n, c_n, h_n) in the two matrices. The first step is to move that location, which is seen from the matrix, to a point in system "0".

By means of equations 11 to 13, equations 19-21 are obtained:

$$(f_n - 1) * relation = x_{en} \quad (19)$$

$$(c_n - 1) * relation = y_{en} \quad (20)$$

$$(h_n - 1) * relation = z_{en} \quad (21)$$

By means of the homogeneous transformation $\boldsymbol{T_0}$, the point seen from the "0" plane is obtained (See equation 22):

$$\begin{bmatrix} x_{0n} \\ y_{0n} \\ z_{0n} \\ 1 \end{bmatrix} = (\boldsymbol{T_0})^T * \begin{bmatrix} x_{en} \\ y_{en} \\ z_{en} \\ 1 \end{bmatrix} \quad (22)$$

Obtain a function that determines the inverse kinematics of the robot, implementing as arguments, the desired point and, as return values, the generalized coordinates for the two possible solutions. It must be indicated at the same time if there is an error with that position for solution 1 or solution 2. This error can be related to the limits that the link has when rotating (See equation 23).

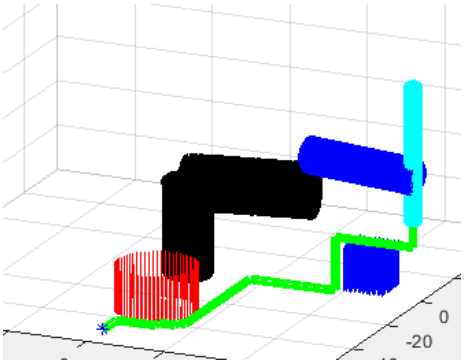
$$[\theta_1, \theta_2, d, err_1, \theta_{12}, \theta_{22}, err_2] = invk(x_{on}, y_{on}, z_{on}) \quad (23)$$

If the algorithm is in step 3, it must use the solution (θ_1, θ_2, d). If there is no error, it is proceeded to use equations 15 to 17, taking into account that homogeneous matrices must have such generalized coordinates. Then, using equations 11 to 13, all the values (f, c, h) are found to then approximate them. The result obtained is the position that the robot is occupying in the matrix. If any position contains the value -1, it means that it would be a mistake to locate the final actuator in the (f_n, c_n, h_n) position. It is therefore not possible to mark the (f_n, c_n, h_n) cell of the "solution 1" matrix. If the function that determines the inverse kinematics detects an error just in the position to be evaluated, it means that it is not possible to mark this position in the matrix either. The errors can contemplate not only locations outside the reach of the robot, but angles that by physical restrictions are not possible to reach.

For step 5, only the generalized coordinates in the homogeneous matrices should be modified by the variables $(\theta_{12}, \theta_{22})$ and take into account that it is dealing with the matrix "solution 2".

RESULTS

Some of the solutions that can be obtained by implementing said algorithm are shown in Figure 14. The displacements are plotted with green and red lines. Those that are green represent a point that must be reached by the arrival solution 1, while the red lines indicate that you must arrive through the arrival solution 2. The asterisk is the point wanted to reach at the end of the entire trajectory.



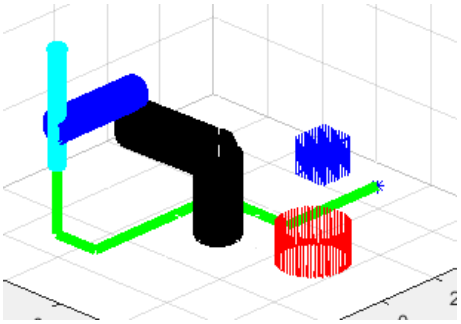
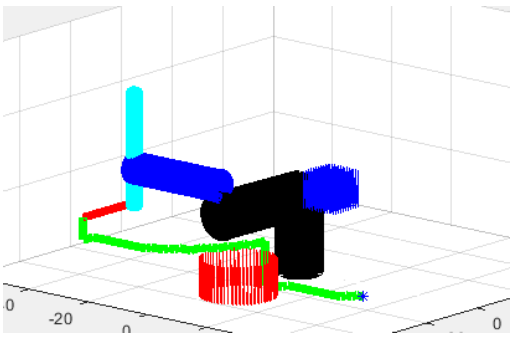


Figure 14. Path planning for the Scara robot.

An interesting situation occurs when the robot must change the location of the elbow to reach the destination. Then, in Figure 15, this situation is simulated by modifying the position of the blue obstacle.



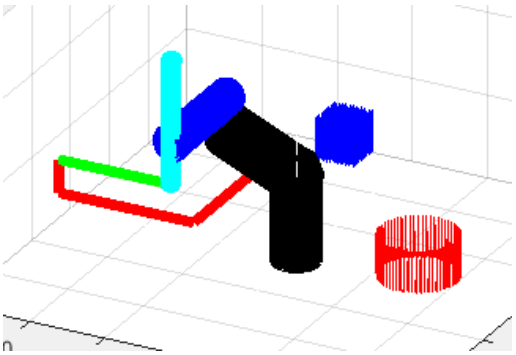


Figure 15. Path planning for the Scara robot.

CONCLUSIONS

Implementing the modified Flood Fill algorithm in trajectory planning of fixed robots, such as Scara, is a great tool that allows generating solutions that not only avoid obstacles, but also determine the shortest trajectory. Bearing in mind that the displacements of the final actuator are given by translations, which, by movement, can only have one component in any of the three coordinate axes.

The response time and the precision of the algorithm depend totally on the chosen size for the matrix that represents, in a discrete way, the operating environment of the robot. Greater precision necessarily involves an increase in the number of

cells that represent the workspace.

The results obtained from all calculated trajectories indicate that the Algorithm in all situations will get the robot to the desired point, as long as the required translation is physically possible, i.e. it does not require overcoming the physical restrictions of the robot and that collisions with obstacles do not occur.

The algorithm can be implemented in robots other than Scara. The logic allows to extend trajectory planning to every robot with an inverse analytical kinematics.

ACKNOWLEDGMENT

The research for this paper was supported by Davinci research Group of Nueva Granada Military University.

REFERENCES

[1] Pachón-Suescún, C.G., Aragón, C.J.E., Gómez, M.A.J. and Moreno, R.J., 2017, "Obstacle Evasion Algorithm for Clustering Tasks with Mobile Robot," In Workshop on Engineering Applications, Springer, Cham, 742, pp. 84-95. doi: 10.1007/978-3-319-66963-2_9.

[2] Milanés Hermosilla, D. and Castilla Pérez, A., 2016. "Generación de trayectorias para el brazo robótico (ArmX)," Ingeniería Electrónica, Automática y Comunicaciones, 37(3), pp.58-71.

[3] Luis, D., 2013, "Diseño e Implementación del Algoritmo Flood-Fill," UNAM Univ, Mexico.

[4] George, L., 2013, "Quantitative Comparsion of Flood Fill and Modified Flood Fill Algorithms," International Journal of Computer Theory and Engineering, 5(3), pp. 503-508.

[5] Ayad, M., 2016 "Autonomous Navigation of Mobile Robot Based on Flood Fill Algorithm," Electrical and Electronic Engineering, 12(1), pp. 79-84.

[6] IEEE CPP, "MICROMOUSE PROJECT," Available: http://www.cppieee.org/micromouse. Consult date: Sep 10, 2018.

[7] Zhang, H.M., Peh, L.S. and Wang, Y.H., 2014. "Study on flood-fill algorithm used in micromouse solving maze," In Applied Mechanics and Materials, 599, pp. 1981-1984.

[8] Pérez, M.A., Cuevas, E. and Zaldívar, D., 2015. Fundamentos de Robótica y Mecatrónica con MATLAB© y Simulink©. *México DF: Afaomega*.

[9] Reyes Cortés, F., 2012. *Matlab aplicado a robótica y mecatrónica* (No. 681.51 670.4272).

[10] López, J. Padilla, F. Luna, M. and Peña C., 2011, "Diseño e implementación de un brazo robótico educacional con aplicaciones culinarias," *Vector*, 6, pp. 45-53.