# Obstacle Evasion Algorithm for Clustering Tasks with Mobile Robot

César Giovany Pachón Suescún[1], Carlos Javier Enciso Aragón[1], Marco Antonio Jinete Gómez[1], and Robinson Jiménez Moreno[2(✉)]

[1] Universidad Piloto de Colombia, Bogotá, D.C, Colombia
{cesar-pachon1,carlos-enciso,marco-jinete}@upc.edu.co
[2] Universidad Militar Nueva Granada, Bogotá, D.C, Colombia
robinson.jimenez@unimilitar.edu.co

**Abstract.** This paper presents a proposal of obstacle evasion oriented to mobile robots in clustering tasks. For this case, polar coordinates are set for the movement of the mobile, the possible obstacles in the path are determined and imaginary boundaries are generated in each possible obstacle in order to delimit the path of the mobile between them. The algorithm developed under the Netlogo programming environment makes it possible to perform evasion and reach the clustering point efficiently.

**Keywords:** Clustering · Obstacle evasion · Mobile robot · Netlogo

## 1 Introduction

Clustering algorithms allow to perform different tasks of pattern recognition in data clustering, for instance, in [1], clustering validation techniques using cancer datasets are presented. However, clustering techniques are not biased only to datasets hence they are now being projected to navigation applications. For example, in [2], path-clustering techniques based on spatiotemporal restrictions are performed.

The analysis of trajectories and clustering of them has strong applications in systems of video surveillance [3], e.g., to determine trends in a particular path that can determine human-vehicle intersections and avoid accidents. Because of this, clustering applications can be found for vehicle [4] or aircraft trajectories [5].

Another application of clustering techniques in path planning is given in mobile robotics, where there are developments that seek to generate trajectories based on obstacle evasion to go from one point to another, as is the case of [6], also this is the objective proposed in [7], but in the latter case, a fuzzy clustering technique is used to delimit boundaries that will set the possible paths of the robotic mobile.

Algorithms such as those presented in [8, 9], which are based on a matrix system from the images captured in order to generate a path, generate a relatively high computational cost if they are implanted in programming languages not optimal for the use of matrices. In addition, they may be affected with the resolution in which the image is captured, since the larger the image, the longer the processing time. There are projects such as those implemented in [10] which use a global camera, or as seen in [11] that use


J.C. Figueroa-García et al. (Eds.): WEA 2017, CCIS 742, pp. 84–95, 2017.
DOI: 10.1007/978-3-319-66963-2_9

a local camera, causing in this way that the strategy for the path planning has a dependence of the location of the camera. Investigations focused on the clustering of objects [12], in which the clustering zone depends more on the initial location of the objects to be grouped, can cause them to lose their usefulness in real applications where objects are needed in a particular zone.

Many of the simulations that exist in both the path planning and in clustering of objects usually obviate situations that can occur in an implementation in a real environment, as is the method of interaction with the medium [13], the intercommunication schemes between robotic agents [14] and collision with obstacles. In this project it was sought to take into account this type of situations, for which it was necessary to determine a procedure that was implemented in the general algorithm, which focused on the avoidance of obstacles for clustering tasks with easy adaptability to systems with global and local camera, scalability to a real environment and, because it has been developed in a generic way, it may allow its validation in multi-robot environments and collaborative applications.

This paper is developed in three sections, the first presents the materials and the methods used, the second presents the analysis of results and the final section gives the conclusions reached.

## 2 Materials and Methods

This paper discusses the development of an algorithm of evasion and clustering for a single mobile robotic agent, which must perform the path planning for its displacement from an object to the final location, avoiding obstacles in the path. The algorithm is developed in the NetLogo simulation environment, in order to verify its correct operation and to be able to have a future collaborative working environment, which is the strength of such simulation software, to work with multiple agents of the same or different class.

The projection of the algorithm starts from the initial idea of an agent that has a camera and, through image processing, locates objects with known dimensions in the work area, obtaining their polar coordinates with respect to the agent. This is the starting point of the algorithm, i.e. it starts from the base that the position of the objects with respect to the agent is known as well as the coordinates of the point of clustering. The agent must generate a strategy to move from each object to the point of clustering, then go to another object and return to the point of clustering and thus to finish the groupable objects. In Fig. 1, it can be seen the agent, the objects represented as a magenta square and the clustering area as a green square.

The initial problem that arises in the development of the algorithm is how to go from a point A (current location of the mobile) to a point B (location of the object), and then from the point B to a point C (clustering location) (Fig. 1). Because the polar coordinates are known at each of the points (B and C), where the agent must go, the agent would only need to rotate an angle $\alpha$ and move a distance $h_1$ to move to point B (see Fig. 2).

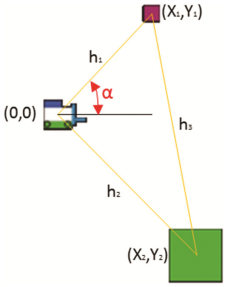


**Fig. 1.** Work area.

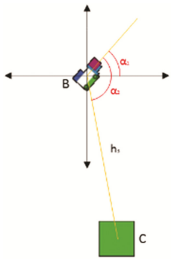


**Fig. 2.** Translation from point A to point B

Now it is desired to go from point B to point C, but the origin of the system and the orientation of the plane changed, therefore, it must be calculated the new position of C with respect to the new point of origin that is now located in B, taking into account the angle at which the agent is rotated.

For this calculation, the initial positions of A, B, C can be taken. Observing in Fig. 1, the coordinates of A are (0, 0), but to calculate the coordinates of B and C, which are $(X_1, Y_1)$ and $(X_2, Y_2)$ respectively, it is necessary to perform the conversion of the known polar coordinates to rectangular ones, (see Fig. 3).

$$x_n = h\cos(\propto)$$
$$y_n = h\sin(\propto)$$

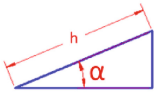


**Fig. 3.** Conversion from polar coordinates to Cartesian

Once these values are known, a subtraction is performed between the coordinates of C and those of B, with Eq. (1).

$$NC = (X_2, Y_2) - (X_1, Y_1) \tag{1}$$

In this way, the coordinates of B and those of C can be taken as (0, 0) and as NC, respectively. Now to calculate $h_3$, which refers to the displacement from B to C, the magnitude of NC is calculated in (2).

$$h_3 = |NC| \tag{2}$$

Part of what is sought in this algorithm is to determine the different possible paths and which of them is the shortest, both translational and rotationally. When the agent rotates, it has two possibilities, the first is to rotate to the right and the second to the left, it will always be sought to rotate to the side where the angular displacement is the shortest, so if it is desired to calculate the angle of rotation from B to C, it must be assumed that the orientation of the agent is at 0°. With the function Atan2, the shortest angle can be calculated from 0°, but NetLogo lacks this function, so a simple Atan and a conditional are used, in order to achieve the same results as with the Atan2. If the angle is greater than 180°, 360° is subtracted from this angle, otherwise, if it is minor, no changes are made. The angle $\propto_1$ is subtracted from the result obtained in order to know the total angle that must be rotated to be in orientation to C. After calculating $h_3$ and $\propto_2$, it can be proceeded to rotate and move the agent to C (see Fig. 4). In this way if there were a point D, and we wanted to go from C to D, it would be proceeded to perform the same procedure that was done from B to C.

**Fig. 4.** Translation from point B to point C

The way in which it was decided to address the problem of collisions was proposing imaginary borders around each object, where their dimensions will be at least the radius of rotation of the mobile agent to implement, plus an extra safety factor given in centimeters, which can vary from 0 to the value that is deemed convenient, in order to ensure that, at any point on the frontier to which the agent is addressed, it will not collide with the object in question. To better understand a case of evasion, it will be made an analysis based on the possible scenario that can be observed in Fig. 5.

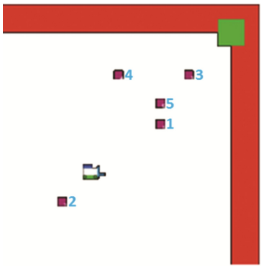


**Fig. 5.** Example of evasion

If it is wanted to go to object 3 and then take it to the clustering area, the agent would have to be able to evade the other objects, which would become obstacles for the agent. The first step to achieve this goal is to identify possible obstacles. The first parameter is

if any of the objects is a distance greater than the target, then this is discarded as a possible obstacle. The next parameter refers to the maximum angle that the objects could be with respect to the objective, in order to discard those that exceed that angle in later calculations since these objects would not affect in the trajectory. In Fig. 6 it is possible to analyze a case in which an object is as close as possible to the agent, without hindering its rotation. If the agent wants to move to another point, the maximum angle that must be rotated to avoid colliding or that the object may obstruct the rotation of the agent after translation is 90°, therefore, this angular value is assigned as the maximum that objects can be relative to the target, so that they can be considered as possible obstacles.

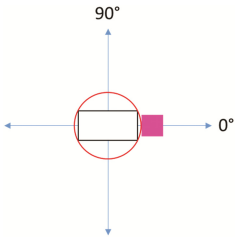


**Fig. 6.** Minimum angle to avoid an obstacle.

To calculate the angle between the target and each of the obstacles, this will be equal to the scalar product between the two vectors, divided by the product of their modules, and from this result, the arc cosine is calculated in (3).

$$\cos^{-1}\left(\frac{\overrightarrow{obj}\cdot\overrightarrow{obs}}{\left|\overrightarrow{obj}\right|\left|\overrightarrow{obs}\right|}\right) \tag{3}$$

Where $\overrightarrow{obj}$ refers to the target vector and $\overrightarrow{obs}$ to the obstacle vector.

Only objects that become possible obstacles in the path, the calculation of their borders is made. In Fig. 7, it can be seen that objects 1, 4 and 5 have points which represent the vertexes of their boundary.

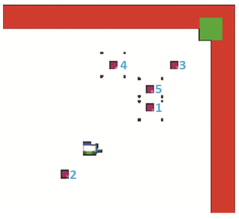


**Fig. 7.** Boundaries

In this case, the objects 5 and 1 share borders, therefore, when sharing borders are considered both as a single obstacle, and a new boundary is defined which will take into

account for its dimensions the maximum and minimum points between boundaries of 5 and 1.

In Fig. 8a a special case is presented in which object A and B share boundaries but do not share with object C. In this case a new boundary is formed between the merging of A and B, this new resulting boundary shares boundaries with C (see Fig. 8b), therefore, in the algorithm designed, once two obstacles are merged it is proceeded to compare if this union shares a border with another obstacle, if so, they are combined again. Consequently, when C shares boundaries with the fusion between A and B, they generate a new one, and, what were initially 3 obstacles, are now treated as one (see Fig. 8c). This boundary treatment will avoid complex inter-object calculations of escape routes.

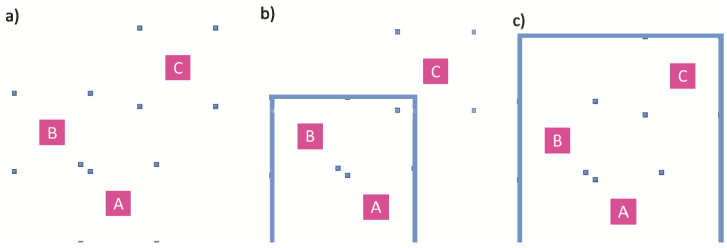


**Fig. 8.** Boundary union.

Once the final boundaries are set, which for reasons of simulation it is obviated to have to draw them, it is proceeded to calculate which are the possible obstacles that remained, in this case are 2 (object 4 and the union between 5 and 1), after this it is proceeded to calculate if the angle $\propto$ is contained between the maximum and minimum angles that form the boundaries of each obstacle with respect to the agent, (see Fig. 9).

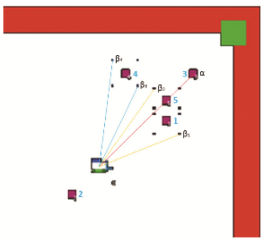

**Fig. 9.** Obstacles identification.

In this case only the union between 5 and 1 interfere with the trajectory of the agent, as 4 does not interfere, it is not considered as an obstacle. In case there are two or more obstacles that interfere in the path, only the one closest to the agent will be taken into account. Once the obstacle that interferes is identified, the evasion algorithm is performed. In this phase, 4 possible ways of evading the obstacle are established, these forms depend on the position of the object with respect to the agent (see Fig. 10).

- For case 1, it must be sought which is the lower corner of the obstacle boundary that is closest to the target.
- For case 2, it must be sought which is the upper corner of the obstacle boundary that is closest to the target.
- For case 3, it must be sought which is the left corner of the obstacle boundary that is closest to the target.
- For case 4, it must be sought which is the right corner of the obstacle boundary that is closest to the target.

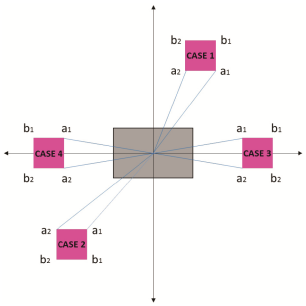


**Fig. 10.** Evasion cases.

Once the case is detected, it is proceed to move the agent to this point, in this example it can be seen that it is the case 1, and the lower corner closest to the target is the lower right, (see Fig. 11a).

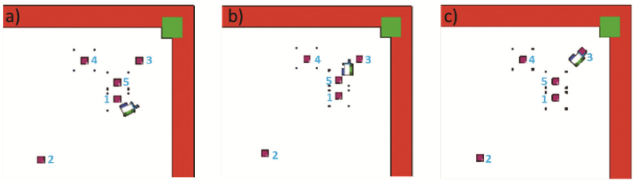


**Fig. 11.** Trajectory of evasion

Subsequently, the agent is moved to the corresponding point b, in this case $b_1$, as can be seen in Fig. 11b.

Once this point has been reached, it is necessary to calculate again whether there may be obstacles between the agent and the target, if this is the case, the same procedure for identifying and avoiding obstacles must be repeated. Since this is not the case, it is calculated the displacement to be made by the agent to the target, and move to this target point, (see Fig. 11c).

Finally, it is proceeded to calculate the path to the clustering zone, in which it must be also taken into account if there may be obstacles, if they exist, the evasion is done under the same algorithm, but since this is not the case, once the angle of rotation and

the distance to be displaced have been calculated, the agent is taken along with the target to the clustering zone, (see Fig. 12).

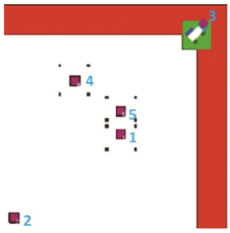


**Fig. 12.** Displacement to the clustering zone.

Subsequent to this, in order to avoid having to evade a high number of obstacles, in the algorithm once the first object is taken to the clustering zone, the agent will bring the other objects from the closest to the farthest.

The case presented above refers if the agent was first for a specific object, but finally what is sought with this algorithm is to determine the shortest joint path, taking into account both linear and angular displacements. Therefore, an equation must be proposed in which the different possibilities of paths are calculated, and then compared and go through the shortest path, which is shown in (4).

$$Trajectory = Tra_{ag\rightarrow ob_i} + Tra_{ob_i\rightarrow za} + 2 * \left( \left( \sum_{n=1}^{numob} Trs_{za\rightarrow ob_n} \right) - Trs_{za\rightarrow ob_i} \right) \quad (4)$$

Where:

- $Tra_{ag\rightarrow ob_i}$ is a path from the agent to the object *i*.
- $Tra_{ob_i\rightarrow za}$ is path from object *i* to the clustering zone.
- *numob* refers to the number of objects in the environment.
- $Trs_{za\rightarrow ob_n}$ is the unobstructed path from the clustering zone to an object n.
- $Trs_{za\rightarrow ob_i}$ is the path without obstacles from the clustering zone to the object *i*.

In the equation it is initially assumed that the agent will be moved to get an object ***i***, therefore, to take it to the clustering zone, it must be calculated the path to this object, and then from the object to the clustering zone. Whenever it is referred to trajectory, it is taken into account the displaced both linearly and angularly. As it is wanted to take all objects to this area, after carrying the first one, it should go from the closest to the farthest, therefore a summation of the paths from the zone of clustering up to each of the objects is proposed assuming that there are no obstacles, since it will go from the nearest to the farthest, to this sum it is subtracted the value of the trajectory without obstacles of the clustering zone until the object ***i***, since it has already gone for that object, this result is multiplied by two because the path of the object n to the clustering zone must also be taken into account.

This calculation will be done by varying ***i*** from 1 to the number of objects. From all the results, it is calculated which is the smallest and therefore the value that had ***i*** in that

result will be the object by which the agent must first be displaced in order to perform the shortest path.

## 3 Analysis and Results

For the validation of the algorithm, an interface was designed in which different variables related to the work area of the clustering task can be entered, (see Fig. 13).

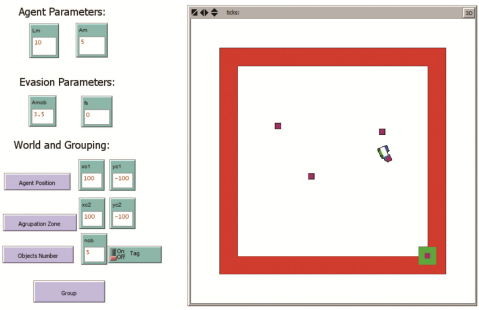


**Fig. 13.** NetLogo interface. (Color figure online)

In the parameters of the agent, it must be entered half the length and half the width of the mobile agent, since with these values the radius of rotation will be calculated and the boundaries will be established. In the evasion parameters, half the width of the objects is entered and a wanted extra safety factor to be added to the size of the boundaries. For the practical case, an ultrasound sensor that directionally detects the distance to the obstacle and this safety factor, delimits the maximum approach of the mobile to the object.

For the execution of the program, the initial position of the agent must be set, besides the zone of the surroundings where it is desired to group the objects, and finally the number of objects that are desired, they will come out with random coordinates inside the work area (area delimited by the red frame Fig. 13).

The operation of the algorithm can be seen in [15, 16], where, in a first situation, the agent groups up to 10 objects, and in the other, how the safety factor can cause the boundaries of the objects to merge and consequently the agent has a different path.

In all the tests except 3 specific situations, it was possible to group all the objects in the zone set, the situations that were presented are:

- The initial objective is contained within the boundaries of obstacles
- The agent is located at the boundary of the obstacles
- If a safety factor is entered high enough to cause the agent to be unable to move properly within the work area.

In order to check the speed of execution of the calculation of the trajectory according to the set evasion and clustering method, it is proceeded to propose four different scenarios in which the position of the clustering zone is changed in order to verify if this

change can significantly affect the execution times, or these are due to some other factor, (see Fig. 14).

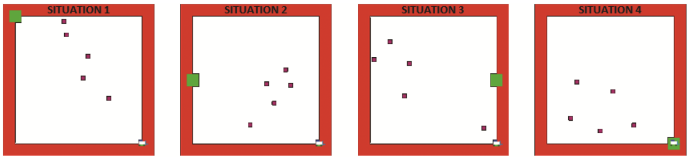


**Fig. 14.** Test scenarios

In order to accurately set the execution times, taking into account that in each test the positions of the objects change randomly, five different tests were performed for the same number of objects in each scenario. It should be noted that the tests were performed on a computer that does not operate the simulation in real-time, which can cause that the time increases if it is implemented in an embedded system due to its hardware limitations. In Table 1, the results obtained in each of the four situations can be observed, by varying the number of objects to be grouped.

**Table 1.** Processing time

| Situation | Objects Number | Time 1(ms) | Time 2(ms) | Time 3(ms) | Time 4(ms) | Time 5(ms) | Average Time(ms) |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 10 | 9 | 12 | 8 | 9 | 9,6 |
| 1 | 10 | 21 | 19 | 24 | 18 | 25 | 21,4 |
| 1 | 15 | 36 | 37 | 36 | 35 | 37 | 36,2 |
| 1 | 20 | 57 | 51 | 53 | 62 | 57 | 56 |
| 2 | 5 | 8 | 9 | 11 | 7 | 8 | 8,6 |
| 2 | 10 | 18 | 18 | 22 | 16 | 16 | 18 |
| 2 | 15 | 38 | 33 | 39 | 42 | 35 | 37,4 |
| 2 | 20 | 56 | 49 | 53 | 53 | 55 | 53,2 |
| 3 | 5 | 7 | 10 | 8 | 7 | 9 | 8,2 |
| 3 | 10 | 19 | 20 | 17 | 15 | 14 | 17 |
| 3 | 15 | 42 | 30 | 32 | 41 | 42 | 37,4 |
| 3 | 20 | 60 | 56 | 68 | 51 | 59 | 58,8 |
| 4 | 5 | 10 | 10 | 7 | 7 | 8 | 8,4 |
| 4 | 10 | 19 | 17 | 21 | 17 | 20 | 18,8 |
| 4 | 15 | 33 | 42 | 41 | 33 | 41 | 38 |
| 4 | 20 | 53 | 59 | 57 | 63 | 53 | 57 |

As it can be seen in Table 1, the average times for different scenarios and the same number of objects do not vary significantly from one situation to another. The algorithm speed presents an approximate linear behavior between the average time and the number of objects, obtaining in the present situations of Table 1 a correlation higher than 0.97 between the data with respect to a trend line generated.

## 4 Conclusions

In the simulations performed, 3 specific cases were presented in which the algorithm was not as expected. In order to avoid the second situation corresponding to the fact that the agent was confined within the boundaries of the obstacles, it is advised to locate the initial position of the agent at the border or outside of the work area. For the third situation, a safety factor is entered that is high enough to cause the agent not to be able to

move correctly in the work zone, so it is advisable that, according to the measures of the agent to be implemented, empirical tests are made to decide which the most appropriate safety factor is.

It is advised that, in case of an actual implementation, the clustering zone should be outside the work area, since it could present cases in which the accumulation of objects is such that interferes with the correct operation of the algorithm.

Observing in the simulations of the situations presented in Table 1, the execution times do not vary depending on the initial distance to which the agent is from the clustering zone, but of the obstacles that would be generated in the calculation of the most optimal path.

It should be noted that the algorithm is developed under conditions of reasonable use, including agent dimensions, number of objects and their size, safety factor, work area dimensions, among others. Therefore, setting appropriate ranges for the execution of the algorithm will depend on the user.

With the developed algorithm it is sought that this only needs the initial location of the objects with respect to the agent, causing in this way not to depend on the resolution of the image or on the image processing algorithm that previously would be implemented. In addition, it can be implemented with both global and local cameras, thus presenting greater adaptability to different types of clustering projects. It even allows the user to be the one to decide where they want the clustering zone, thus expanding their adaptation to real situations. If it is analyzed the processing times, it can be thought about the feasibility of adapting the algorithm to one with real-time image processing, thus allowing the development of human-machine interaction, multi-agent or collaborative algorithms.

## References

1. Yu, Z., Kuang, Z., Liu, J., Chen, H., Zhang, J., You, J., Wong, H.S., Han, G.: Adaptive ensembling of semi-supervised clustering solutions. IEEE Trans. Knowl. Data Eng. **PP**(99), p. 1. doi:10.1109/TKDE.2017.2695615. 1 August 2017
2. Wu, H.R., Yeh, M.Y., Chen, M.S.: Profiling moving objects by dividing and clustering trajectories spatiotemporally. IEEE Trans. Knowl. Data Eng. **25**(11), 2615–2628 (2013). doi:10.1109/TKDE.2012.249
3. Bak, Ç., Erdem, A., Erdem, E.: Clustering motion trajectories via dominant sets. In: 2016 24th Signal Processing and Communication Application Conference (SIU), Zonguldak, 2016, pp. 601–604. doi:10.1109/SIU.2016.7495812
4. Besse, P.C., Guillouet, B., Loubes, J.M., Royer, F.: Review and perspective for distance-based clustering of vehicle trajectories. IEEE Trans. Intell. Transport. Syst. **17**(11), 3306–3317 (2016). doi:10.1109/TITS.2016.2547641
5. Mcfadyen, A., O'Flynn, M., Martin, T., Campbell, D.: Aircraft trajectory clustering techniques using circular statistics. In: 2016 IEEE Aerospace Conference, Big Sky, MT, 2016, pp. 1–10. doi:10.1109/AERO.2016.7500601
6. Shantia, A., Bidoia, F., Schomaker, L., Wiering, M.: Dynamic parameter update for robot navigation systems through unsupervised environmental situational analysis. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, 2016, pp. 1–7. doi:10.1109/SSCI.2016.7850238

7. Moreno, R.J., Lopez, D.J.: Trajectory planning for a robotic mobile using fuzzy c-means and machine vision. In: Symposium of Signals, Images and Artificial Vision - 2013: STSIVA - 2013, Bogota, 2013, pp. 1–4. doi:10.1109/STSIVA.2013.6644912
8. Mohammed, A.: Autonomous navigation of mobile robot based on flood fill algorithm. Iraq J. Electr. Electron. Eng. **12**(1), 79–84 (2016). E-ISSN 2078-6069
9. Burgos, D.A.T.: Planeamiento de trayectorias de un robot móvil. In: Enero 2006. [En línea]. http://tangara.uis.edu.co/biblioweb/tesis/2006/119245.pdf. [Último acceso: Junio 2017]
10. Murakami, K., Hibino, S., Kodama, Y., Iida, T., Kato, K., Naruse, T.: Cooperative soccer play by real small-size robot. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS, vol. 3020, pp. 410–421. Springer, Heidelberg (2004). doi: 10.1007/978-3-540-25940-4_36
11. Jiménez, F.J., Moreno, J.C., González, R., Rodríguez, F., Sánchez, J.: Sistema de visión de apoyo a la navegación de un robot móvil en invernaderos. In: XXIX Jornadas de Automática, 3–5 Septiembre. Tarragona, España (2008)
12. Gauci, M., Chen, J., Li, W., Dodd, T., Gross, R.: Clustering objects with robots that do not compute. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, pp. 421–428, 5 May 2014. E-ISSN 978-1-4503-2738-1
13. Chatty, A., Kallel, I., Gaussier, P., Alimi, A.M.: Emergent complex behaviors for swarm robotic systems by local rules. In: 2011 IEEE Workshop on Robotic Intelligence Informationally Structured Space (RiiSS), pp. 69–76, 11–15 April 2011. doi:10.1109/RIISS.2011.5945791
14. Kwon, J.W., Kim, J.H., Seo, J.: Consensus-based obstacle avoidance for robotic swarm system with behavior-based control scheme. In: 2014 14th International Conference on Control, Automation and Systems (ICCAS), pp. 751–755, 22–25 October 2014. doi:10.1109/ICCAS.2014.6987879
15. Suescún, C.G.P., Aragón, C.J.E., Gómez, M.A.J., Moreno, R.J.: Youtube, Junio 2017. [En línea]. https://www.youtube.com/watch?v=LfxV1McBJRY&t=30s. [Último acceso: Junio 2017]
16. Suescún, C.G.P., Aragón, C.J.E., Gómez, M.A.J., Moreno, R.J.: Youtube, Junio 2017. [En línea]. https://www.youtube.com/watch?v=HBLqLI7yKWo. [Último acceso: Junio 2017]