SELF-ORGANIZING MOBILE ROBOTS BASED ON MULTI-AGENT COORDINATION TECHNIQUES IMPLEMENTED WITH AERIAL VISION AND COMMUNICATION GATEWAY BETWEEN WIFI AND RF

Cynthia Daniela Briones Valencia
Universidad Autónoma de Guadalajara
danycdbv@gmail.com

Zandor Alfredo Machaen Terriquez
Universidad Autónoma de Guadalajara
zandor92@gmail.com

Luis Martin del Castillo
Universidad Autónoma de Guadalajara
luismartin.925@gmail.com

Gustavo Alejandro Torres Blanco
Universidad Autónoma de Guadalajara
gustavo.blanco@edu.uag.mx

Resumen

Este artículo presenta el desarrollo de robots móviles que poseen la capacidad de búsqueda y recuperación de obstáculos en un entorno de laberinto. El algoritmo incorporado en los robots fue diseñado con base en principios de coordinación y autoorganización, es decir, un grupo de agentes autónomos coordinan sus acciones para buscar y recuperar obstáculos del entorno a través de la cooperación. Para ello, se diseñaron dos tipos de agentes, organizadores y operadores. Los organizadores tratan de coordinar las acciones de los operadores, y estos últimos, tratan de recuperar todos los obstáculos en el medio ambiente. Cinco robots de cuatro ruedas fueron construidos desde cero utilizando Arduino Uno para los operadores, y Arduino Nano y NXP i.MX53 Quick Start

Boards para los organizadores. Además, se utilizó una cámara aérea (fijada al techo) para proporcionar percepción visual a los robots. La comunicación se realizó a través de una pasarela entre el canal de 8bit RF y WiFi, para los operadores y los organizadores, respectivamente.

Palabras Claves: Autoorganización, robots móviles, visión computacional.

Abstract

This paper presents the development of mobile robots that have the abilities of search and retrieval of obstacles in a maze-like environment. The algorithm embedded in the robots was designed based upon principles of coordination and self-organization, i.e., a group of autonomous agents coordinate their actions in order to search and retrieve obstacles from the environment through cooperation. To do this, two types of agents were designed, organizers and operators. Organizers try to coordinate the actions of the operators, and these last, try to retrieve all obstacles in the environment. Five four-wheeled robots were built from scratch using Arduino Uno for the operators, and Arduino Nano plus NXP i.MX53 Quick Start Boards for the organizers. Also, an aerial camera (attached to the ceiling) was used to provide visual perception to the robots. The communication was made through a gateway between 8bit channel RF and WiFi, for the operators and organizers respectively.

Keywords**:** *Computer vision, mobile robots, self-organization.*

1. Introduction

The solution of agent societies problems is a very complex task, it requires coordination techniques, policies and diverse communication standards. Fulfilling these requirements guarantees that the system will accomplish the objectives for which it was designed. This can be accomplished through the contribution of the individual knowledge of each of the members within the system because it allows to leverage the abilities and knowledge of each member that comprises the system. So, the general purpose of solutions like the one presented in this document is to divide complex tasks into single and simple sub-tasks, which will

reduce the computing order on space and time in each member of the system. Thus, these kinds of techniques improve the performance when a society of agents tries to achieve a global objective [Dimopoulos, 2006], [Chen, 2005], [Turner, 2013].
The self-organization principle states that the agents have a partial knowledge of the environment and can define, in an individual manner, the rules that regulate the interaction between members. It is believed that these interactions improve the performance of the agents within emerging environments. Therefore, they have abilities to sense and modify the environment and communicate with the rest of the agents within the system. Regardless of the environment configuration, the agent local interactions are performed to integrate all this knowledge in order to completely learn the current environment [Di Marzo, 2005], [Mataric, 1993]. Some methodologies and rules within software engineering have defined the coordination, scalability and redundancy in order to ensure the fulfillment of the given tasks [Yoshida, 2007], [Gomez-Sanz, 2006], [Criado, 2013].
The proposed approach in this document is to work with a decentralized system, [Muñoz, 2005], [Jacyno, 2013] which allows, unlike [Peng, 2013], to eliminate the need of relying on a leader to assign the tasks to each one of the agents. The design of the society is based on local interactions, through direct communication by message passing that allows the agents to determine their actions. Moreover, the whole idea is based principally in studies developed by Marco Dorigo at Iridia Labs. [Brambilla, 2012].
This paper is structured as follows. The description of the components of the system and the definition of the abilities of the agents is presented in section 2. The functions that define the interactions of each agent are described in section 3. The implementation and the results of the proposal appear in section 5. Finally, the conclusions and the future work are mentioned in section 6.

2. Methods

The maze-like environment is integrated by a finite number of obstacles, five mobile robots (agents) that are randomly distributed throughout the environment:

operator agents and organizer agents, three and two members, respectively, which must collaborate in order to achieve the following three main objectives: obtain a maze free of obstacles, organize obstacles outside the maze, and organize themselves into a formation.

The physical environment and the agents may be observed in figure 1. The maze had to be adjusted to the range of vision of the aerial camera, so the robots were to turn at 45 degrees through the perpendicular aisles. This was because a robot with a forklift was not able to completely turn to perform routes at right angles.

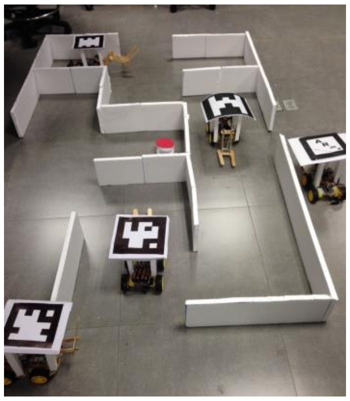

Figure 1 Maze configuration, the robots have a tag, the obstacle is a red dot.

Organizer Agent

These agents' duty is to coordinate the operators when they leave the environment and to relocate the obstacles that are already outside of the maze, their main functions are:

- Detect an agent outside the maze.
- Relocate obstacles towards the exit.
- Maintain the control of the agent's exit.
- Inform the agents that they may start the formation.

Operator Agent

Agents in charge of removing the obstacles that are immersed within the environment and evacuate through the nearest exit available, their main capabilities are the following:

- Detect obstacles in the environment.
- Remove obstacles from the environment.
- Request the organizing agent to relocate the obstacles away from the exit.
- Receive input notifications from the organizing agent.
- Initiate the formation outside of the maze.
- Search for maze exit.
- Avoid obstacles.

Mobile Robots

Five four-wheeled robots were used in this project, each of them built with the following components:

- Four 5 V DC motors, bridged in pairs.
- One H-Bridge to control the two pairs of motors.
- A forklift to grasp the obstacles in the maze.
- A unique tag with different patterns for aerial vision control.
- A 5 V servomotor for the forklift.

The two types of mobile robots were implemented using two different microcontroller architectures. First, the operator agents were built using Arduino Uno microcontrollers based on ATmega328P with the following extra components:

- A multichannel RF for wireless SPI protocol.
- Eight AA batteries to feed H-Bridge and Arduino Uno.

Secondly, the organizer agents were built using an i.MX53 QSB based on 32 bits ARM Cortex-A8 with the following extra components and software:

- A WiFi USB dongle.
- Arduino nano to control H-Bridge through serial communication (i.MX53 doesn’t have GPIO ports).
- GNU/Linux Yocto based operating system.
- JADE framework to run agents.

Gateway Between WiFi and RF Communication Protocols

A Laptop PC with GNU/Linux operating system was used to create a gateway through WiFi and RF. WiFi interface was used to create a connection with the organizers, and serial communication with an Arduino Nano was used to create connections with the operators.

Maze

The maze was built with polystyrene sheets. It was made only to present a visual representation of the maze to human observers. The maze is hardcoded in the aerial vision system, so all mobile robots control is made on the laptop PC through artificial vision.

Aerial vision System

An Android smartphone camera, connected through WiFi using IP streaming based application, was used to obtain visuals from the environment. The vision system is made up of the following software components:

- OpenCV library:
  - ✓ ARma framework for robot tags recognition.
  - ✓ PatternDetector Library from George Evangelidis [Evangelidis, 2017].
  - ✓ Color recognition module for obstacle detection.
- JADE framework for the agent's container.
- A JADE agent that acts as the interface to the WiFi to RF gateway.
- A Java based application interface to Arduino Nano SPI master using
- USB serial communication between the PC and the Arduino Nano.
- A Java based application to perform all coordination algorithms for therobots. Thus, it sends actions to all robots in the environment.
- Path planning is performed using A* algorithm.
- Additionally, the vision system is responsible of the following actions:
- Identify every robot in the environment through pattern recognition.
- Provide vision to all robots, in order to tell them where to move.
- Recognize the obstacles in the environment.

- Detect when a robot is able of lift an obstacle.

The pattern detector algorithm is as follows: first, detect pattern corner in order to establish pattern's position in every captured frame. Then, normalize the ROI for every detected pattern and compare it with loaded patterns. Once a pattern match with a known one, estimate de transformation between camera's cardinal system and the pattern rotation. Finally, use the transformation matrix for pattern location, rendering, etc. [Evangelidis, 2017].

In the following section, the description of the algorithms for the operators and organizers are presented. Then, the implementation issues and results are detailed.

The algorithms described in the following paragraphs were analyzed for completeness and soundness. All algorithms are complete because there are only if-else statements that covers up every choice in the space. Concerning to the soundness, it is affected by correct pattern detection and reliable communication, because of the hardware used in the implementation it can be said that there are not sound. But, in a future work it will be interesting to perform a study, concerning distributed systems and image processing algorithms, which was not the main focus of this research.

Operator Agent Functions

Forward: Function that allows the agent to move inside the maze until it finds a barrier that hinders it, which should be analyzed to determine which action to perform afterwards:

```
If(exit_found&&bring_obstacle)Then
    Send request to the organizer to get out
Else if(exit_found &&!bring_obstacle)Then
    Avoid Exit
Continue forward
Else if(barrier_found)Then
    Identify type of barrier
Else
    Continue forward
End if
```

Identify Barriers: This function allows the agent to determine the actions that it must take according to the type of barrier that it finds within the environment:

```
If(obstacle)Then
    If(bring-obstacle)Then
        Avoid obstacle
    Continue forward
Else
        Get obstacle
        Find Exit
End if
Else if(wall) Then
    Avoid barrier
    Continue forward
Else if(agent)Then
    Stop for a time t
    Continue Forward
End if
```

Exit the Maze: The operator gets ready to start formation when it receives a notification to exit the environment by the organizer.

```
Request approval to get out of the labyrinth
If(request_approved) Then
    Place obstacle out of the labyrinth
    Wait for notification of organizing agent to get out
If(notification_received) Then
    Gets out of the labyrinth
    Start formation
End if
Else
    Find another exit
End if
```

Start Formation: Function that allows the agent to integrate into the formation once it is out of the environment:

```
If (agents_in_formation == 0) Then
Start formation
Else
    Request re-organization of the agents
    Integrate into formation
End if
```

Functions of the Organizing Agent

Exit Authorization: Once the organizer receives an exit request from an operator agent, it will analyze if there is still a request in the formation in order to send an answer to the received request.

```
If (number_of_elements <= 3) Then
        Send approval
If (detecta_obstaculo) Then
        Get the obstacle
Relocate the obstacle
End if
Else
        Reject exit.
End if
```

Relocate obstacle: Allows the organizer to detect an obstacle outside of the maze in order to relocate it with the rest of the obstacles that are already outside.

```
If(obstacle_ouside != 0) Then
Move obstacles that are outside
End if
Relocate obstacle
Return to initial position
        Send notification to the operator agent to start the formation.
```

3. Results

The simulation can be divided into three major phases: obstacle search, obstacle removal and outskirts formation. Every phase its explained in the following paragraphs. Before explaining phase's results, a discussion of some characteristics of the implementations is needed.

The most complex location is the one that is most distant to all the robot operators because the calculation of the roads is more complex. Moreover, time is not relevant, because the goal is not to finish in a time limit, but note that the mobile robots exhibit, through its search and retrieval capabilities of obstacles in a maze environment, cooperative behaviors. If there were more obstacles to be removed, as the system is parallel, it should work if there are the same number of obstacles than robots. If there were more obstacles than robots, it should not complete the scenario.

Obstacle search: First, the robots try to find obstacles in the environment, performing a random search controlled by the aerial vision system. The vision system sends movement commands to adjust the travel of every robot.

When an obstacle is found, the robot approaches it, then it lift its forklift in order to retrieve the obstacle. Then, it communicates with an organizer robot to request to exit the maze, figure 2 shows these actions.

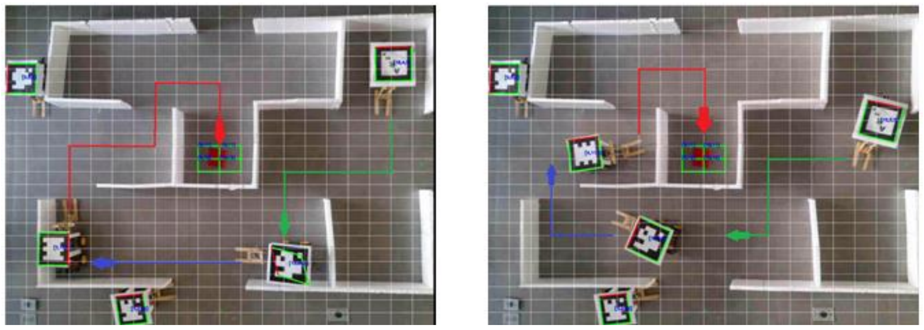
Figure 2 Obstacle search, the obstacle is identified by the aerial system.

Obstacle removal: Secondly, the robots that does not retrieve any obstacle receive a request from operator to exit the maze. Meanwhile, the robot that retrieved the obstacle tries to get out of the maze, aided by the vision system using an A* algorithm to generate the path to the nearest exit with an organizer robot. figure 3 shows these actions.

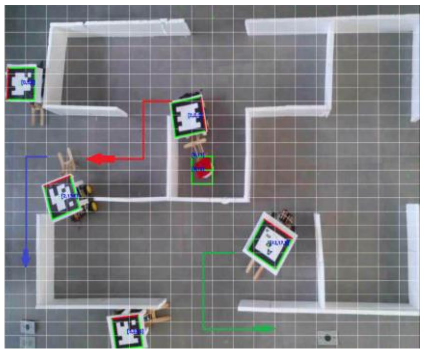
Figure 3 Obstacle removal, the nearest robot lifts the obstacle in order to remove it from the maze.

Outskirts formation: Thirdly, when the retriever robots gets to the exit, the organizer robot tells him to put down the object, removes the obstacle and tells the retriever to start the formation. The retriever robot exits the maze. figure 4 shows these actions.

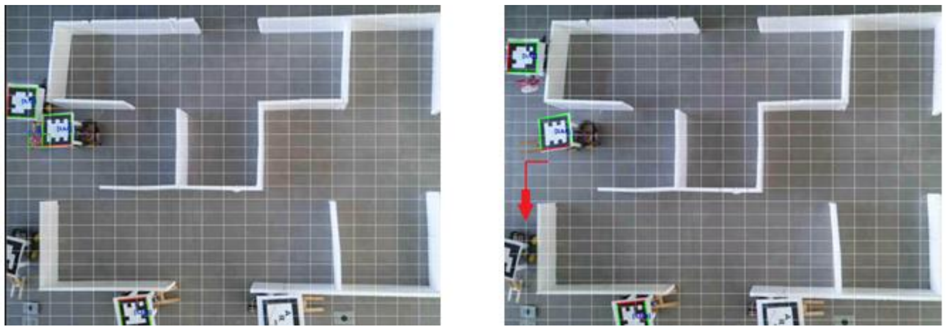
Figure 4 Organizer robot job, the organizer robot tells the retriever to leave the obstacle and begin with the formation.

Finally, the retriever robot accommodates in a formation on the outskirts of the maze. Meanwhile, the other robots keep searching for obstacles in the environment. figure 5 shows these actions.

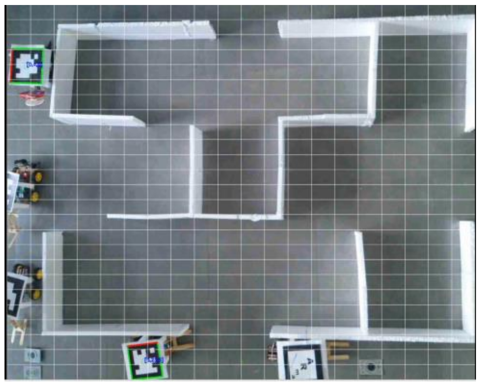
Figure 5 Formation, the retriever accommodates in a formation.

4. Discussion

The construction and implementation of the robots and the algorithms were not straightforward. The following list details most of the issues that the team solved:

1. Robot movement control through vision, caused by floor slipperiness.
2. Room illumination variation that affects vision.

3. Power issues caused by the battery packs.
4. Forklift design.
5. Integration of components, caused by power feeding issues.
6. RF Communication, caused by modules malfunction.
7. Robots free rotating wheels.

The solutions are listed as follows:

1. Diminish the distance traveled by the robots, and vision correction through a squared threshold in the virtual environment.
2. Blocking sunlight from the environment.
3. It was not resolved very well, in the future it is expected to use better performance battery packs.
4. Forklift design needs improvement, maybe using a 3D printer.
5. Power feed the H-bridge and motors independently of the microcontroller.
6. The issue was resolved by switching RF modules whenever there is a malfunction due to power issues.
7. Change free rotating wheels with an extra pair of DC motors and wheels.
8. It is important to note that the solutions need to be improve implementing more adequate methods, encountered in robotics theories. Most of the solutions presented here were made because of the pricing of more accurate components.

5. Conclusions

This paper detailed the coordination algorithms, the technologies for computer vision and communication gateway. The results were not as planned because of the several issues that the team faced in the construction and implementation of the robots. All major issues were explained in the corresponding section, and the solutions were not the best. But these can be taken as improvement opportunities to be prepared as future work.

The team has now the responsibility to improve the performance, first by trying to get more adequate materials and hardware, like switching forklift made up of wood with either 3D printed or acrylic ones. Switching the NXP i.MX53 that lack of GPIO ports with microcontrollers that did have ports. Improving the traction of the robots using a less slippery floor. Improving the maze with a more professional material than polystyrene, and so on. It is expected to fulfill these improvements as soon as possible.

6. Bibliography and References

[1] Chen, G., Yang, Z., He, H., and Goh, K. M., Coordinating multiple agents via reinforcement learning, Autonomous Agents and MultiAgent Systems, vol. 10, no. 3, pp. 273–328, 2005.

[2] Criado, N., Using norms to control open multi-agent systems, AI Commun., pp. 317–318, 2013.

[3] Brambilla, M., Ferrante, E. Birattari, M., Dorigo, M. Swarm Intell Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence, Springer, 7 (1), pp.1-41, 2013.

[4] Di Marzo, K. M., Serugendo, M., Gleizes, P., and Karageorgos, A., Self-organization in multi-agent systems, Knowl. Eng. Rev., vol. 20, no. 2, pp. 165–189, 2005.

[5] Dimopoulos, Y. and Moraitis, P., Multi-agent coordination and cooperation through classical planning, in 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 398–402, 2006.

[6] Evangelidis, G. Arma library: Pattern Tracking for Augmented Reality: https://sites.google.com/site/georgeevangelidis/arma at June 2017.

[7] Gomez-Sanz, J. J., and Pavon, J., Defining coordination in multi-agent systems within an agent oriented software engineering methodology, in Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 424–428, 2006.

[8] Jacyno, M., Bullock, S., Geard, N, Payne, T. R., and Luck, M., Self-organizing agent communities for autonomic resource management, Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems, vol. 21, no. 1, pp. 3–28, 2013.

[9] Mataric, M. J., Designing emergent behaviors: From local interactions to collective intelligence, in Proceedings of the Second International Conference on From Animals to Animats 2: Simulation of Adaptive Behavior: Simulation of Adaptive Behavior, pp. 432–441, 1993.

[10] Muñoz Salinas, R., Aguirre, R., García-Silvente, M., and Gomez, M., A multi-agent system architecture for mobile robot navigation based on fuzzy and visual behaviour, Robotica, vol. 23, no. 6, pp. 689–699, 2005.

[11] Peng, Z., Wen, G., and Rahmani, A., Leader-follower formation control of multiple nonholonomic robots based on backstepping, in Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 211–216, 2013.

[12] Turner, J. R., Multiagent systems as a team member, International Journal of Technology, Knowledge & Society, vol. 9, no. 1, pp. 73–90, 2013.

[13] Yoshida, T., Cooperation learning in multi-agent systems with annotation and reward, Int. J. Know. -Based Intell. Eng. Syst., vol. 11, no. 1, pp. 19–34, 2007.