A Flexible and Dynamic Mobile Robot Localization Approach

C. Peñaranda, V. Julian, J. Palanca, V. Botti

Departamento de Sistemas Informáticos y Computación (DSIC).
Universitat Politècnica de València.
Camino de Vera s/n. 46020 Valencia, Spain

Abstract. The main goal of this paper is to provide an approach to solve the problem of localization in mobile robots using multi-agent systems. Usually, the robot localization problem has been solved in static environments by the addition of the needed sensors in order to help the robot, but this is not useful in dynamic environments where the robot is moving through different rooms or areas. The novelty of this dynamic scenario is that each room is composed of external devices that can enter or exit the system in a dynamic way and report the position where the robot is. In this way, we propose a multi-agent system using the SPADE MAS platform to improve the location of mobile robots in dynamic scenarios. To do this, we are going to use some of the advantages offered by the SPADE platform such as presence notification and subscription protocols in order to design a friendship network between sensors/devices and the mobile robots.
keywords: Multi-Agent System, Mobile Robots, Open Systems.

1 Introduction

Current mobile robots are very complex machines which include a lot of components that in some circumstances are difficult to synchronize. For this reason, it is quite common for mobile robot applications that the level of complexity gets increased due to a good coordination and/or synchronization of the different elements for sensorization and actuation. This problem complicates the achievement of their objectives. One of the most well-known problems is the positioning problem of a mobile robot inside a building. This is a recurrent problem, which has been resolved with greater or lesser success in recent years.

In this paper, we employ the humanoid robot NAO[1], which has been developed by Aldebaran Robotics. The robot has a coordinates system to know its own position. The problem is that the robot is not very accurate when it performs several movements. In such situations it is very common for the robot to have a significant deviation between its actual position and the robot's belief of its position according to its sensors. Besides, when it performs several complex tasks, which are composed of different movements along different rooms,

[1] www.ald.softbankrobotics.com

the robot gets big mistakes due to its lack of knowledge about its real position, as we can see in Fig 1. Our goal is to provide an agent-based framework that gives a mobile robot, which is moving through different areas or rooms inside a building, a flexible and dynamic way to obtain its correct position.

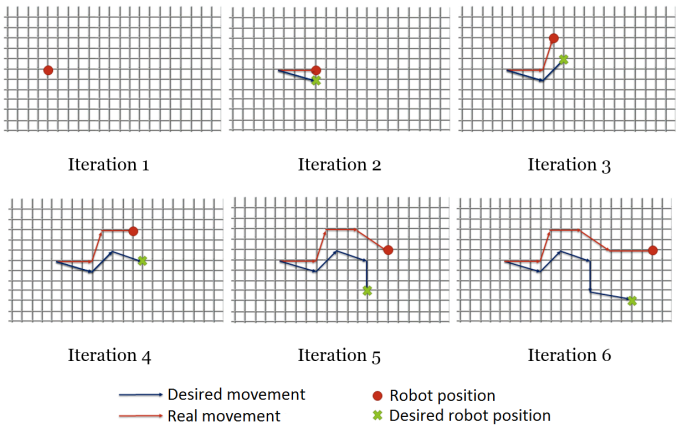

Fig. 1. NAO robot movement.

This location problem may appear due to two causes. The first one is due to the type of floor used. Each material exerts a kinetic friction between the robot and the ground and, depending of this kinetic friction, the robot movement is more or less accurate. The second one depends on the components that the robot employs for its movements. These components usually make little errors that may affect the robot movement and may make it believe that it is in a different position than the real one. This problem is even greater when a robot is moving between different rooms with different floors, lighting, objects. The precise adjustment of the own robot's sensors is not possible and, therefore, the use of external sensing devices (such as cameras) is mandatory. In this sense, this paper proposes a multi-agent based framework using the SPADE platform[2] to improve the location of the robots in dynamic scenarios using external devices that can enter or exit the system in a dynamic way. The goal of these devices is to help the robot for a more accurate calculation of the robot position.

The proposed approach is based on the use of multi-agent technology (MAS). Intelligent agents offer a great opportunity to solve different types of distributed problems. For that reason, we are going to use a Multi-Agent System (MAS) to help the robot to know its real position when it performs several movements inside a building. The MAS provides the distribution, dynamism and flexibility needed to solve this problem. Besides, the proposed solution can be used to help

[2] https://pypi.python.org/pypi/SPADE

any mobile robot to know their position and also to allow new devices that know the robot position to be added or removed.

The rest of the paper is organized as follows. Section 2 describes the developed MAS framework. Section 3 shows the experiments and the final performance achieved using the proposed MAS. Section 4 presents a discussion in this topic. Finally, some conclusions and future work are presented.

2 Multi-Agent System proposal

As commented before, the main contribution of this paper is to provide an agent-based framework that allows us a flexible and dynamic way to obtain the correct position of a mobile robot which moves through different areas or rooms inside a building. The framework has been implemented using the SPADE platform [2]. The use of an agent-oriented approach allowed us to obtain a flexible and distributed solution where different sensing devices can be added or deleted in the MAS in a transparent way. The platform follows FIPA and XMPP/Jabber standards [2] and it is also the first platform based on XMPP instant messaging.

Agents developed in SPADE are implemented using Python, which is one of the most used programming languages for the development of applications for the NAO robot. Also, the SPADE platform incorporates different features that are interesting for this work, such as: a transparent use of the publish and subscribe event protocol following the PubSub protocol (which is supported by the platform) and the use of presence notification that allows the system and the different agents to know, in real-time, which agents are connected in the platform.

The use of the SPADE platform allows us to use it as a component that connects the different agents that control sensing devices, called *device agents*, with the agent that controls the robot, which is called the *robot agent*. The device agents can be used to manage the different sensing devices which are placed in the environment. Thus, each sensing device can have a different way to obtain and calculate the position of the robot, depending on its physical sensor component and its location algorithm. Next section describes in more detail this kind of agent.

2.1 Device agent

The device agent has been designed as a set of different behaviors. The first one is the **Service Behavior** which activates the services offered by the device and periodically publishes information about the robot position through these services (this is done using the PubSub protocol[3]). Note that, the number of activated services depend on the number of sensors that the device incorporates. That is, a device agent can be an aggregation of different sensors available through the

[3] The PubSub protocol allows agents to subscribe to events published by other agents. Agents receive the information that another agent publishes if they are subscribed to the event.

same device. Finally, for each offered service, the device agent also includes a **Position Calculation Behavior** which calculates the robot position using the available data of a specific sensor (camera, sonar, etc.) and its own algorithm.

For camera sensors, the calculation of the robot position is done following an algorithm which searches the robot position in the environment, using the OpenCV[4] library. The developed algorithm depends directly on the device position and the employed sensor. So each device agent can have a different **Position Calculation Behavior** for each sensor. To ease the position calculation, the environment has been divided into fields (identified by two coordinates x and y), which reduces drastically the number of possible positions of the robot in the experimentation environment.

2.2 Calculating position with OpenCV

Different algorithms that detect the robot position have been developed depending on the different devices that we use. In this paper, we used two types of devices: VGA cameras and infrared cameras that detect the depth of the environment. Scenario was divided into fields where different sensing devices can locate the robot position.

First, a vertical search is performed to detect the position x of the robot. In this search the algorithm obtains the position in which the robot gets the largest area. This initial part is similar in VGA and infrared cameras. Then, the algorithm performs a horizontal search that detects the position y. This algorithm is different in VGA and infrared cameras because Infrared cameras perform an in-depth search and easily find the robot position. But, VGA cameras look for the robot from the camera's closest position to the farthest one and stores the position in which the robot appears first, if the area that gets exceeds a certain threshold. This threshold has been set to improve the accuracy of these cameras. An example of the use of this algorithm is shown in Fig. 2, where the darker lines are the x and y position that have been detected by a normal camera.

2.3 Robot agent

The robot agent also incorporates a set of different behaviors. The first one is the **Service Search Behavior** which is in charge of collecting information periodically about the different available services (offered by the available device agents). If one service is found, the robot agent sends a subscription request to the found service. Each subscription activates in the robot a new behavior named **Reception Information Behavior** that is in charge of collecting the necessary information in order to calculate the robot position. Finally, the robot agent also incorporates the **Movement Behavior** which is responsible for moving the robot around the environment.

Thanks to the presence notification feature of the SPADE platform, the robot agent can know the current status of each device agent, knowing whether the

[4] http://www.opencv.org/

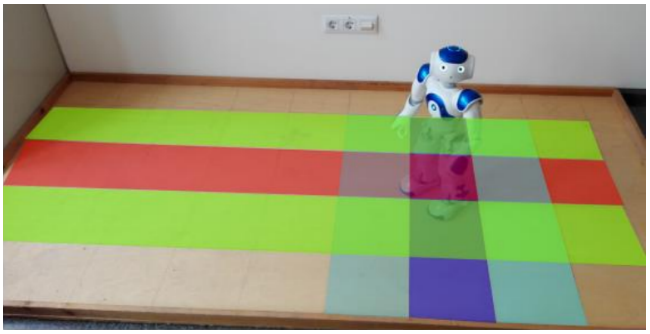

Fig. 2. Robot positioning with VGA camera.

device agent is connected or not to the platform. Thereby, the robot agent can modify its subscription related to this agent. This allows the robot agent to have the subscription list always updated with the device agents that are currently available in the environment.

2.4 Trust model

One of the problems that the robot agent can have in order to use the information given by the device agents is the reliability of that information. Sensors can have some percentage of error in the calculation of the robot position. Moreover, this error can increase due to the changing conditions of the environment. According to this, a trust model has been incorporated to the proposal. This model allows the robot agent to assign a level of trust for each device agent, that can be adapted during the execution of the system. Note that the robot also has a self-trust value, because it has an internal coordinates system that is also used as a sensor device. The problem is that this system is not very accurate and leads the robot to make errors in its movements. The trust model on each agent is used by the robot agent to calculate its position, as explained in the next section.

Initially, a new device agent will have the highest possible trust value. This value can be decreased or increased depending on the information about the robot position given by the device to the robot agent. As the robot moves, the robot agent can adjust the trust value assigned to each device agent following the Figure 1. Since the environment is divided in a grid, the robot agent calculates the difference between the calculated position and the position returned by the device and adjusts the trust value depending on the difference obtained in previous steps and the last difference obtained. On the one hand, if the difference obtained is 0 it increases the trust value in λ when the previous difference is also 0. When the previous difference is more than 0 it increases the trust value in θ. On the other hand, the trust value decreases θ if the current difference is 1 and the previous is more than 0 or λ if the current difference is more than 1. Note

that the trust value can not be increased more than its maximum value (1) or be decreased less than its minimum value (0).

Previous difference	Current difference	Trust modification
0	0	+ λ
>0	0	+ θ
0	1	None
>0	1	- θ
-	>1	- λ

Table 1. Summary of the values to adjust the trust model.

2.5 Position calculation

The **Movement Behavior** commented in section 2.3 is responsible for the movement of the robot around the environment depending of the information obtained in the **Reception Information Behavior**. First, the agent robot calculates the robot position following the Equation 1, that depends of the information returned by each device agent (p), the trust value of each device agent (β) and a factor (α) that takes into account the arrival time of the returned information. This factor will be larger the less time has passed from the reception of the information. Then, the robot only stops when the result of this calculation matches with the desired position. If the calculated position is not the target position, the robot will try again to move to the desired position until it reaches the target position.

$$\hat{P}_{\{x,y\}} = \frac{\sum_{i=0}^{d} \alpha_i \cdot \beta_i \cdot P_{i\{x,y\}}}{\sum_{i=0}^{d} \alpha_i \cdot \beta_i} \tag{1}$$

A general view of the above commented agents can be seen in Fig. 3. Figure also shows the exchange of messages between the two agent classes. From a perspective of the MAS, agents are initially registered in the Agent Management System (AMS). After this initial step, the device agent registers the offered services in the Directory Facilitator (DF).Then, the robot agent tries to search the available services and subscribes to them following the PubSub standard. Finally, the device agent periodically publishes information about the robot position; the robot agent receives this information in order to determine the correct position of the robot.

3 Experimentation

Before starting the different tests, the robot movement was verified in different environments with different floors and we concluded that for these experiments

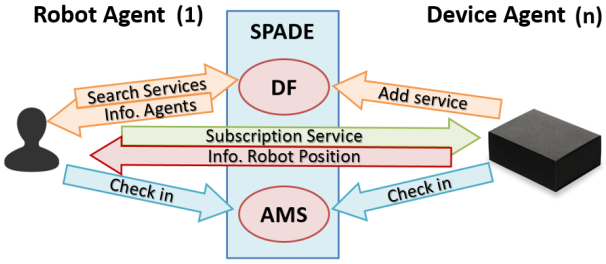

Fig. 3. General scheme of the proposed system.

the kinetic friction was lower in the wooden floor. Therefore, a scenario with wooden floor was developed and was divided into fields. Then different sensing devices where installed in the scenario to locate the robot position. Specifically, these sensing devices were VGA cameras and infrared cameras that can detect the depth of the environment. Besides, the OpenCV library has been used to calculate the robot position following the algorithm explained in section 2.2.

In order to evaluate the developed system, five experiments have been designed using different cameras as sensing devices. The first experiment evaluates how the number of available devices affect in the calculation of the robot position. To evaluate this aspect the experiment increases in each test the number of cameras in a room. In this experiment we have employed cameras that always know the real position of the robot without any error. In the experiment the robot performs different routes in order to measure the error made by the robot in its movements. The second experiment is similar to the previous experiment. In this experiment we increase the number of cameras used in a room, but, in this case, we use cameras with an error threshold in the calculation of the position. The third experiment evaluates how the trust model affects in the proposed MAS. Besides, this experiment allows us to show how the trust assigned to each device agent is adapted during the execution of the system. The fourth and fifth experiments analyze the results obtained when the robot is moving through different rooms of a building in a real and in a simulated scenario.

Note that the euclidean distance was used to calculate the error in each experiment using the field where the robot was and the field where the robot was supposed to be. Besides, each experiment has been repeated ten times, and both the average error obtained in each movement of the robot and the average standard error have been calculated and showed in each figure.

3.1 First experiment using cameras without error

In this experiment, the robot performs different routes to measure the error made using different cameras which know the real robot position (without error). Each

route is performed several times by the robot increasing the number of available cameras in each repetition.

The first route is a simple movement where the robot tries to do a circle (Fig. 8, path *a*). The results obtained can be seen in Fig. 4. As we can see, the robot accumulates the obtained error when it moves without the help of the cameras, achieving a maximum error value of 0.8 fields. The error is reduced until a value of 0.5 fields when we introduce a new camera (37.5% of improvement than without cameras). Finally, the error is completely eliminated introducing a second camera.

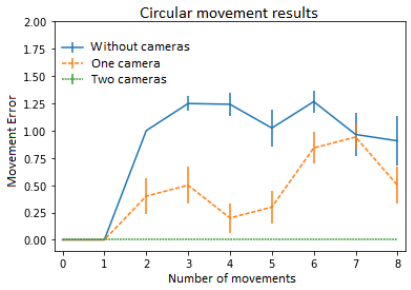

Fig. 4. Circular movement results.

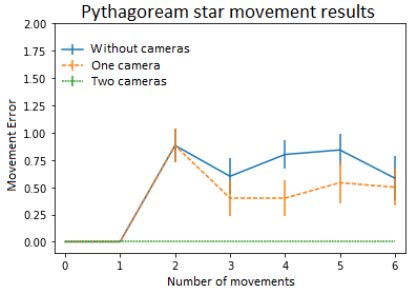

Fig. 5. Pythagorean star movement results.

Other route used is a movement that simulates a star with 5 points, showed in the path *b* of Fig. 8 (Pythagorean Star). This route contains fewer movements than the previous route, but their movements contain higher angles which make the robot movement harder. In Fig. 5, we can see the results of this experiment, which are quite similar than in the previous route. The robot accumulates an error without the help of the device agents until it achieves an error of 0.6 fields. This error is reduced when a new camera is introduced in the system. It finally achieves an error of 0.5 fields (16.67% of improvement than without cameras). In this case the error has not been significantly reduced compared to the previous route. Finally, with two cameras the error is completely eliminated.

Finally, route *c* from Fig. 8 was tested. In this case the robot performs a greater number of movements. The obtained results are very similar to those obtained in the previous routes as can be seen in Fig. 6. Without using sensing devices, the robot accumulates an error of 1.4 fields. The error is reduced until it reaches a value of 1.2 fields (14.29% better than without cameras) when we introduce the first camera. Finally, the error disappears again when two cameras are used.

3.2 Second experiment using cameras with error

The use of cameras that always know the real robot position without error is not possible in a real scenario. Devices can produce errors in their calculations

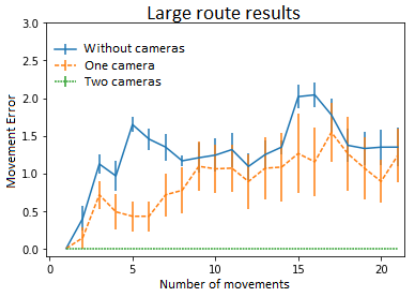

Fig. 6. Large route results.

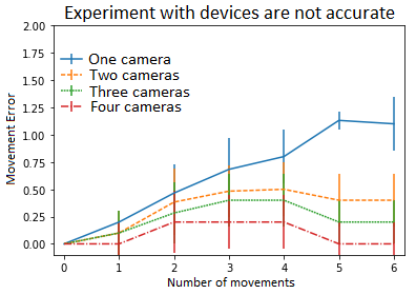

Fig. 7. Results of the experiment with non ideal cameras

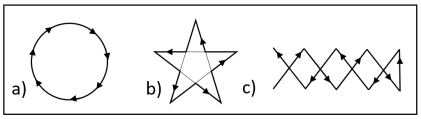

Fig. 8. Different routes used in the experiments.

due to several reasons, such as: the sensors are not effective enough or because different changes have been introduced in the environment that hinder sensor effectiveness. So, we decided to perform new experiments where the number of cameras have an associated error. As in the previous experiments, the number of cameras is increased in each test. In this case, the number of analysed routes has been reduced. The robot only performs the movements that simulate the Pythagorean Star because we consider that is the route with the highest probability of error during the movements of the robot. Moreover, we used cameras with a random error between 10% and 30% in their calculation process of the position of the robot.

Fig. 7 shows the results obtained in this experiment. We can see the error obtained by the robot while the number of cameras gets increased. The robot obtains worse results when a new device agent, which has an associated error, is introduced in the experiment. Specifically, the robot gets an error of 1.1 fields (the robot got an error 0.6 fields without the help of the device agents). However, the error is reduced when the number of cameras is increased until it performs the movement without error with 4 cameras. As the cameras introduce some kind of error, the robot does not reach the desired position in all of the intermediate steps in many of the tests (using 4 cameras), but the final accumulated error is low enough (only 0.2 fields).

3.3 Study of the trust model

This experiment evaluates how the trust model affects in the proposed MAS. The robot agent assigns trust values to all the device agents following the table shown

in Figure 1. These trust values are used to calculate the robot position taking into account that the information provided by device agents will be more important if their trust is higher. So, a correct assignment of the trust to the different device agents is critical. Moreover, the experiment also shows how the trust assigned to each device agent is adapted according to changes in the environment.

The experiment has been executed using 4 cameras with an associated error shown in Figure 9. The error of camera 4 and camera 1 are exchanged in the movement number ten. As commented before, initially all the device agents (and also the robot agent) have the maximum trust assigned. Note that, as explained in section 2.4, the robot also has a self-trust value. This is because it has a coordinates system that is used as a sensing device too. The problem is that this system is not accurate when the robot makes mistakes in its movement, and that is why the robot keeps a trust value also for its own coordinates system.

As can be seen in the figure, this trust is continuously decremented in the case of cameras 4 and 3 and the robot agent. This is because they have a higher error assigned. However, when the robot reaches the movement number ten, their trusts begin to change, increasing the trust in the case of camera 4 until it reaches the maximum and decrementing the trust of the camera 1 until it reaches the minimum value.

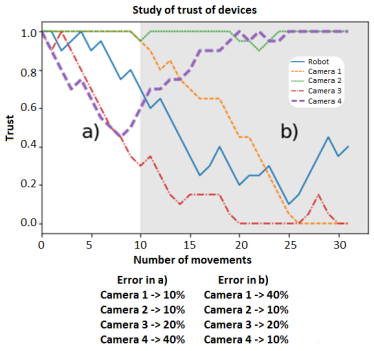

Fig. 9. Study of the trust model.

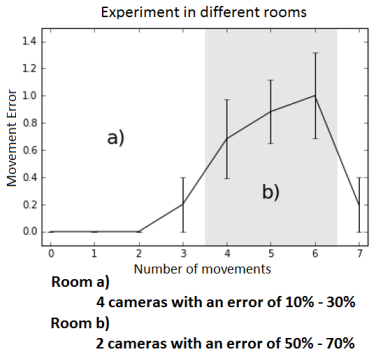

Fig. 10. Experiment with different rooms.

3.4 Fourth experiment with movements through different rooms

This experiment results can be seen in Figure 10. In this case, we want to analyse the results obtained when the robot is moving though different rooms and some device agents can appear or disappear. The experiment has been designed with two rooms. The first one is a room (a) with 4 cameras with an error between 10% - 30%. The second room (b) is a less informed room with 2 cameras with an error of 50% - 70%. We are trying to demonstrate how the robot is able to reach the desired position on the first room in most of the cases, but its error begins to

increase when the robot enters the second room. In addition, the accumulated error of the second room is reduced when the robot returns to the first room. As in previous experiments, in the figure we can see the error obtained in each movement and the standard mean error.

3.5 Fifth experiment with simulated scenario

In this experiment we moved the robot in a more complex and big scenario to validate how the distributed sensors may collaborate to help the robot to improve its location. This complex scenario has been simulated since we had no resources to build it as is. The scenario is again divided into squared fields which are ore units in the coordinates system. This scenario has also different rooms (a living room, a kitchen, a bathroom, ...) and some sensors distributed along the rooms (except the bathroom, for privacy reasons). These devices have also an error between 10% - 30% when they compute the robot's location. The experiment has moved the robot through the scenario following the numbers in Figure 11 in ascending order.

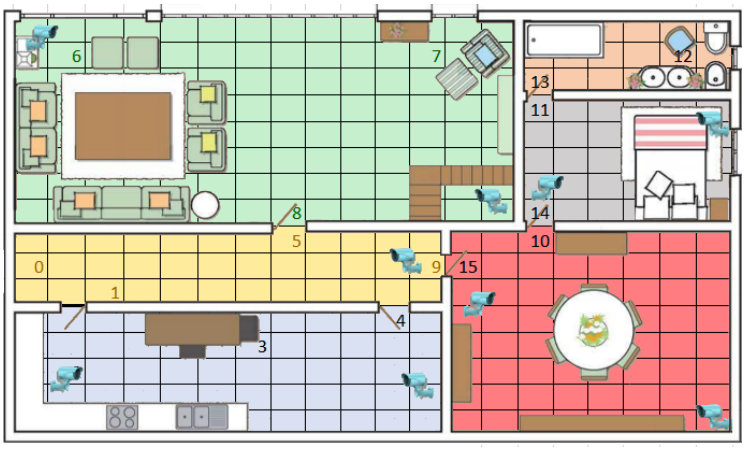

Fig. 11. Simulated scenario.

We are trying to demonstrate how the robot is able to reach the desired position in an scenario if we use external low-cost devices to help the robot to improve its location. In Figure 12 the error obtained at each movement is shown. We can see how the error kept low until the robot reached the movement number 12. At this movement the robot entered the bathroom, where there are no devices to help the robot. Then, when the robot exited this *blind* room, the error was reduced again to regular values and this low error was kept until the end of the experiment.

Another observable conclusion of this experiment is regarding the trust values of each experiment. The system presented in this work is supposed to decrease the trust in devices that are not providing correct enough values for the location of the robot. The experiment has been executed using 9 devices cameras with an error between 10% - 30% and the onboard location system of the robot. Since the cameras error is not too high, the trust values of each device are kept high during the robot movements (see Figure 13), while the trust of the onboard sensor gets decreased due to its low precision.

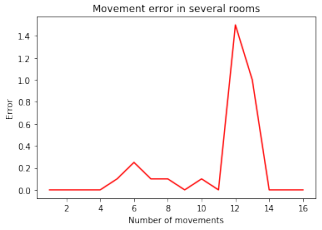

Fig. 12. Experiment in simulated scenario with different rooms.

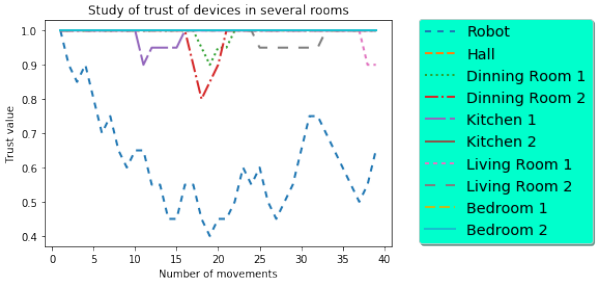

Fig. 13. Study of the trust model in different rooms.

4 Discussion

Several previous works have done different studies taking into account this problem. Specifically, they have focused on two approaches: the first one tries to locate the robot position using fixed devices that know the robot position in the environment [1,9,8]. These studies satisfy the main objective of locating the robot, but they are oriented to a single static room. Besides, the information given by the external devices is critical, hence the works that follow this approach do not tolerate possible failures of these devices such as incorrect operations or changes in the environment that difficult the detection of the robot (for example, a new obstacle between the device and the robot). The second approach satisfies the main objective using only onboard robot sensing [6,11]. The robot calculates its position depending on the detected objects of the environment and its distance from the detected objects using vision-based approaches [5,10]. This solution may not work correctly in a dynamic environment, because the robot can not calculate its global position when there are obstacles in the environment that are changing their position or even new obstacles appear. However, there are some works that add more advanced onboard sensing to the robots to make them able to improve its location by means of probabilistic approaches [3,4]. These works add advanced sensors like 2D laser range finders, which are not low-cost sensors and thus are out of our goals in this work. Finally there are vision-guided works [7] that propose a robot guidance using an external camera which allow robots

to avoid obstacles dinamically. However these works are not able of coordinating different external cameras or sensors to improve robot localization.

Therefore, this work has proposed a hybrid approach which solves the problem of the robot positioning when it has to perform several tasks in different rooms of a building with changing conditions as new objects, failures of external devices, changes in the available sensorization, etc. Also, the proposed system tolerates and fixes possible deviations committed by the external devices, either due to an internal malfunction or due to obstacles that difficult the detection of the robot. These new features improve the previous solutions that were presented above.

5 Conclusions

In this paper, we have designed a MAS to help a mobile robot to know its real position when it performs several movements in a building with several rooms. According to this, we have proposed a new distributed, dynamic and flexible solution that can be used to help any mobile robot and which allows anyone to include any external device that can provide information about the robot position. We have reached different experiments to evaluate the proposed solution. We introduced the robot into an environment with perfectly accurate devices which know the exact robot position in the first experiment. The robot did different routes and we found that results improve as more devices are introduced into the MAS. This is because the robot agent believes to be in a consensus between the position returned by each device agent and the desired position.

The system has been also tested using not accurate devices that introduce some error. We tried to simulate a real environment, so we used different devices with an error between 10% and 30% in their returned information. We found that we can obtain the real position using four of these devices, finding an acceptable error of 0.2 fields in most cases. Besides, the system incorporates a trust model in order to improve the calculation of the real position. This has been evaluated with a study of the trust assigned by the robot agent to the device agents. The robot agent assigned more trust to device agents which are more precise and it adapted trust values according to changes in their accuracy.

According to the proposed evaluation, the implemented MAS can help to know the robot position in an environment with one or more rooms, with changes in the environment and with changes in the external devices used by the robot for the calculations of its position. Besides, this solution allows us to incorporate different device agents and appropriately adjusts their trusts. As future work, we want to test different types of sensing devices that the ones used in our tests. Then, we will study how the movement error of the robot evolves as the devices and the robot reach a consensus. Moreover, in this paper we used only one robot, but in future work we will use a bigger team of robots that cooperate to obtain a result that is beneficial for all.

Acknowledgements

This work was supported by the project TIN2015-65515-C4-1-R of the Spanish government.

References

1. Ignacio Fernández et al. Guidance of a mobile robot using an array of static cameras located in the environment. *Autonomous Robots*, 23(4):305–324, 2007.
2. Miguel Escrivá Gregori, Javier Palanca Cámara, et al. A jabber-based multi-agent system platform. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1282–1284. ACM, 2006.
3. A. Hornung, K. M. Wurm, and M. Bennewitz. Humanoid robot localization in complex indoor environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1690–1695, Oct 2010.
4. Armin Hornung, Stefan Oßwald, Daniel Maier, and Maren Bennewitz. Monte carlo localization for humanoid robot navigation in complex indoor environments. *International Journal of Humanoid Robotics*, 11(02):1441002, 2014.
5. Junichi Ido, Yoshinao Shimizu, Yoshio Matsumoto, and Tsukasa Ogasawara. Indoor navigation for a humanoid robot using a view sequence. *The International Journal of Robotics Research*, 28(2):315–325, 2009.
6. Francisco Martín Rico, Vicente Matellán Olivera, Rafaela González-Careaga, Pablo Barrera González, et al. Visual based localization for a legged robot. 2006.
7. P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade. Vision-guided humanoid footstep planning for dynamic environments. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 13–18, Dec 2005.
8. Eduardo Munera Sánchez, Manuel Muñoz Alcobendas, Juan Fco Blanes Noguera, et al. A reliability-based particle filter for humanoid robot self-localization in robocup standard platform league. *Sensors*, 13(11):14954–14983, 2013.
9. Daniel Pizarro, Manuel Mazo, Santiso, et al. Localization of mobile robots using odometry and an external vision sensor. *Sensors*, 10(4):3655–3680, 2010.
10. A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello. A visual odometry framework robust to motion blur. In *2009 IEEE International Conference on Robotics and Automation*, pages 2250–2257, May 2009.
11. Stephen Se, David G Lowe, and James J Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on robotics*, 21(3):364–375, 2005.