A vision-based external localization and automatic evaluation system for mobile robots localization

Juan I. Alonso-Barba*, Alejandro Jiménez-Picazo†, Ismael García-Varea‡, Jesus Martínez-Gómez§
Email: JuanIgnacio.Alonso@uclm.es*, ajimenez@dsi.uclm.es†, Ismael.Garcia@uclm.es‡, Jesus.Martinez@uclm.es §
Computing System Department
University of Castilla-La Mancha
Albacete, Spain

Abstract— **In this paper we present a system of external localization that enables localization of mobile robots in an arbitrary real space. The purpose of the system is to perform a real-time calculation of the current position and orientation of a mobile robot in different working environments (e.g. the different scenarios of the RoboCup competition). A real application of the system is the automatic evaluation of a specific localization algorithm without the need for human intervention. The system has been developed with the capability to manage an arbitrary number of cameras, and it has been tested in a real environment for different mobile robots and localization algorithms. As a result, a considerable saving of time in the development and validation of a localization algorithm has been achieved. In addition, the proposed system provides an error-free automatic evaluation system of localization algorithms for mobile robots thus doing away with manual evaluations.**

Index Terms— **Vision-based external localization, automatic evaluation of systems, mobile-robot localizationVision-based external localization, automatic evaluation of systems, mobile-robot localization**

I. Introduction

One of the main problems in developing localization algorithms for mobile robots is how to evaluate the performance of such algorithms. This evaluation consists of comparing the pose (position x, y and orientation θ in a 2D space) obtained by the localization algorithm to be tested and the current pose of the robot in a real environment.

There exists two types of localization methods:

- **Active localization (external localization): An external system is used to estimate the pose of the robot, for example a GPS device. In [8] an external localization system using landmarks is proposed.**
- **Passive localization (self localization): The robot has to be able to calculate its location without the help of any external devices. Some examples of self localization methods are: Kalman filters [13], [12], [14], Markov models [10], [11], Monte Carlo methods [15], among others.**

For autonomous robot challenges (i.e. Robocup competitions) only passive methods can be used. In these cases, during the development of self localization algorithms, it is worthwhile and helpful to have an automatic and error-free evaluation process at one's disposal.

Typically, in the absense of an automatic evaluation system, the assessment of a localization algorithm is carried out manually. That is, every time a pose estimation is required, the human operator has to measure and record the current pose in order to systematically compare them afterwards.

Manual evaluation of a localization algorithm entails different problems:

- **The localization results can be biased by errors in the manual measurements.**
- **It is difficult, even unfeasible, to make a comparison with respect to other localization algorithms. For example, when a predefined robot tour is to be tested.**
- **The development time for new localization algorithms is substantially greater as the robot has to be stopped for each manual measurement.**

All these drawbacks can be avoided if an automatic evaluation system is used [9]. Obviously, we believe that each research group has their own automatic evaluation system for the localization algorithms they have proposed. Despite this, and to the best of our knowledge, no generic automatic evaluation system has been proposed for this purpose in the literature.

In this paper we present a vision-based framework to develop an external localization system for mobile robots, as well as an automatic evaluation system for passive localization algorithms.
Automatic evaluation is performed by means of real-time computation of the current localization of the robots.

The remainder of this paper is organized as follows. In section II a description of a generic system architecture is presented. Section III describes a case study consisting of the specific system we have implemented in our laboratory, as wells as the experiments we carried out to test the system. Section IV presents the main conclusions and future developments.

II. General system description

In this section a description of an external localization system is given.
In general, an external localization system must be able to:

- **Manage an arbitrary number of cameras connected to the system.**
- **Detect any kind of mobile robots.**
- **Establish a communication procedure between the system and the robots.**

Taking into account these requirements whilst aiming to facilitate the development and increase the modularity of the system, we propose to divide the system into the following modules: 1) camera management module; 2) robot detection module; 3) communication module; 4) image processing module (optional).

The camera management module is used to access and gather data from all the cameras connected to the system. The number of cameras and their localization define the space

where the system can detect the robots. The system may use an arbitrary number of visual cameras connected to a computer that analyses the captured images to calculate the position and the orientation of the robots. The main advantage of this system is that web cameras are very cheap and the real space that can be covered by the system increases linearly with the number of cameras. Thus, the scalability of the system is only limited by the number of cameras that our computer can manage at the same time. If only one camera is used, it will not be possible to find the robot in a big area, but if too many cameras are used a very powerful computer will be needed to work with the data in real time. The same happens with the resolution of the cameras: a high resolution allows the system to find the robot easily but this is computationally expensive.

The objective of the robot detection module is to detect the position (x, y) and the orientation θ of the robot in one specific image frame. After analysing some image processing libraries and some different algorithms to detect and track objects based on shape detection [7], colour detection [17] or mark detection [1], [2] we suggest the use of one based on mark detection, but any could be used. Mark detection libraries can easily detect any kind of robot simply by placing a specific mark on the robot in question. Moreover, it is possible to use different marks to detect different robots or more than one mark on the same robot to increase the precision of the system.

The communication module is used to send and receive localization data between the system and the robot. This module allows the system to compare the results obtained from the system with the ones sent by the robot. The result of this comparison is used to develop an autolocalization algorithm.

The image processing module is optional and only used for debug purposes. This module is mainly used to graphically represent the information that the system is using. The main task of this module is to draw, in a user interface (UI), the pictures received from the cameras, the results of the robot detection module and the robot autolocalization information if it is available.

III. A CASE STUDY: THE SIMD EXTERNAL LOCALIZATION AND AUTOMATIC EVALUATION SYSTEM

In this section we present the system that we have developed and implemented in our research laboratory (SIMD[1]), following the requirements and the structure set out in the previous section.

The global architecture of the system is depicted in Fig. 1.

A. Camera management module

To manage the cameras we used the free software library `VideoInput` [6][2]. This library is based on `OpenGL` [16] and `DirectX` [3][3].

Our system manages a total of four cenital cameras, which are connected to the computer running the application. In our case, four cameras are enough to cover all the environment in which the robots can freely move. `VideoInput` allows the use of up to twenty cameras simultaneously.

[1]Data Mining and Intelligent Systems Lab. (www.dsi.uclm.es/simd)

[2]`VideoInput` library has been developed only for MS-Windows based systems

[3]`DirectX` Microsoft DirectX is a collection of application programming interfaces (APIs) for handling tasks related to multimedia, especially game programming and video, on Microsoft platforms

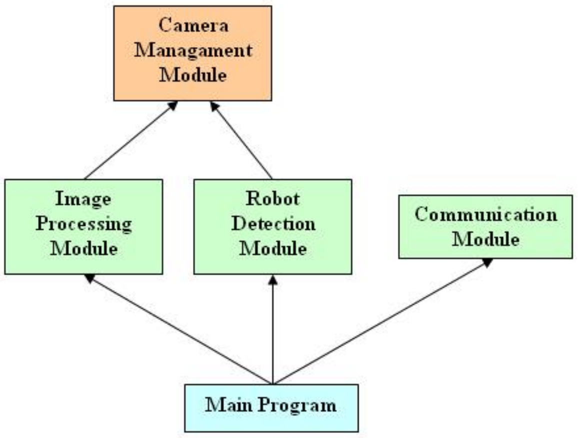

Fig. 1. System architecture

The `VideoInput` library has to be configured according to a set of parameters which define the image resolution, the colour space (RGB/BGR), the video format (NTCS/PAL/SECAM), and whether the images are stored flipped or not. All these parameters have to be correctly defined by the user/programmer according to the specific hardware (cameras) used.

B. Robot detection module

For this module we have used the open source mark detector library `ARToolKitPlus` [18]. `ARToolKitPlus` is a software library that can be used to calculate camera position and orientation relative to special physical markers in real time, and vice versa.

In our case we are interested in obtaining the position and orientation of a physical mark, which will be placed in the middle and highest part of the robot, from one (or more) fixed camera(s).

`ARToolkitPlus` works as follows: when a physical mark is detected the confidence measure (from 0 to 1, this value giving a measure of the confidence of the system seeing the mark) and the model view matrix of the mark are returned. The model view matrix describes the position and orientation of the mark in the 3D camera coordinates reference system, which is automatically converted to the 2D screen/image coordinates system. That is, the center of the image (position (0,0)) corresponds to the axis (0,0,Z) in the camera coordinates reference system. A graphical description is shown in Fig. 2. The orientation is also computed with respect to the (0,0,Z) axis.

C. Communication module

Wireless communication is used to establish and manage communication between the system and the robots. Wireless networks are very common today and most robots include a wireless interface. We use standard TCP sockets for communication, so the system should be compatible with any robot.

The operation of the system is as follows. The robot sends the result of its localization algorithm to the system. Then, the system analyzes and compares the received data with the value returned by the robot detection module.

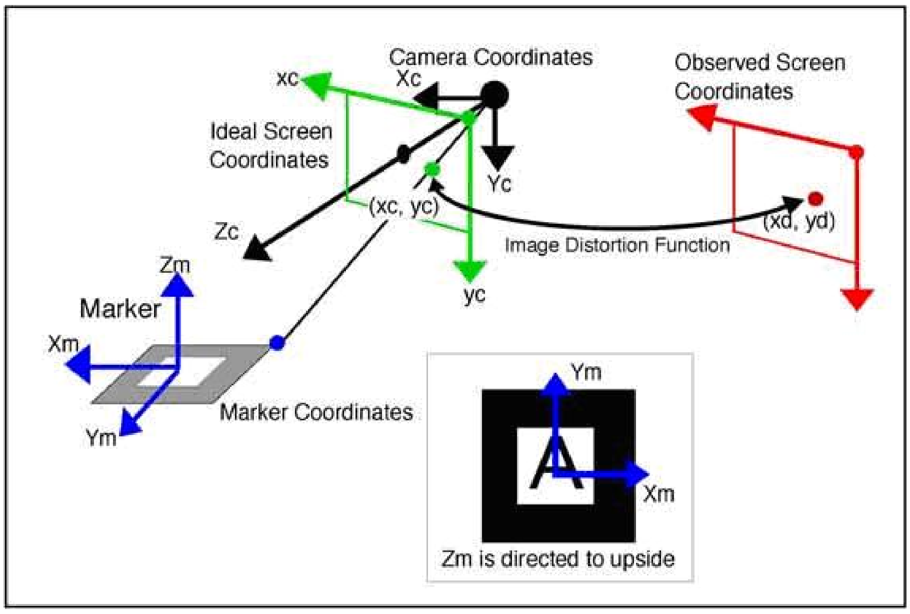

Fig. 2. Coordinates system of Artoolkit+

D. Image processing module

For the image processing module we have used the `OpenCV` library [4]. `OpenCV` is a well-known and widely-used open source Computer Vision library. We chose this library because it provides an easy way to show and modify the images taken by the cameras [5].

The main use of this module is to show the images taken by the cameras in a user interface. It also represents the current position of the detected robot (given by the robot detection module) with a colour mark over it. Furthermore, if the robot has sent its localization data to the communication module it is also represented with another colour mark.

Figure 3 shows an example of the output of the module. The module shows the images received by the four cameras and a color mark is drawn on the estimated position of the robot.

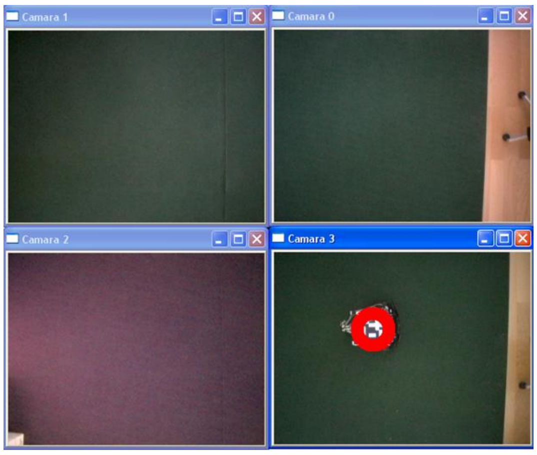

Fig. 3. SIMD laboratory picture

E. Scenario Overview

In this section we describe the real scenario where the system is working. The area where the system is able to detect robots is defined by the cameras connected to the system. We use four FireWire cameras with a resolution of 320x240 pixels and 24bits of colour. The cameras were installed on the ceiling of our laboratory and form a perpendicular angle with it (Fig. 4). The area covered by the cameras is about 4x3 m^2. To combine the localization data obtained from all the cameras it is necessary to know the distance between the cameras connected to the system.

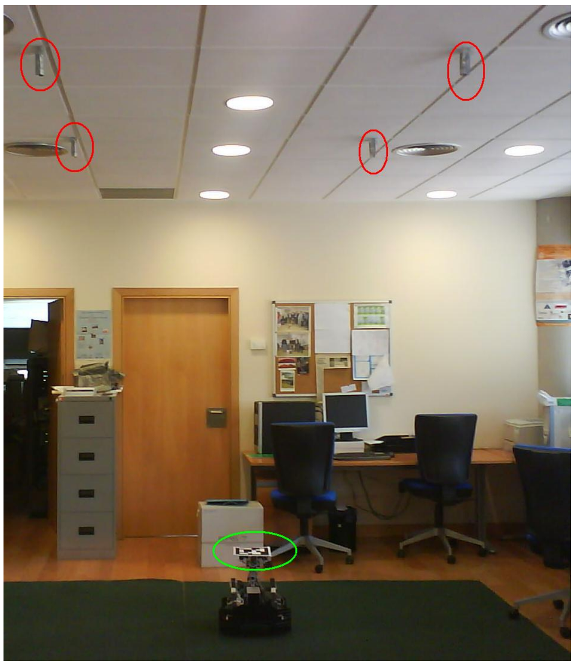

Fig. 4. Real scenario. In the picture, the camera positions are highlighted with red circles, and the physical mark of the robot with a green circle.

In our case we chose a square distribution and thus we cover a large square area combining the pictures taken from all cameras. The system takes the point on the ground vertically bellow the first camera as the origin of the reference system (see Fig. 5).

The mark detection module returns the mark positions in cm but this position is local to each camera.

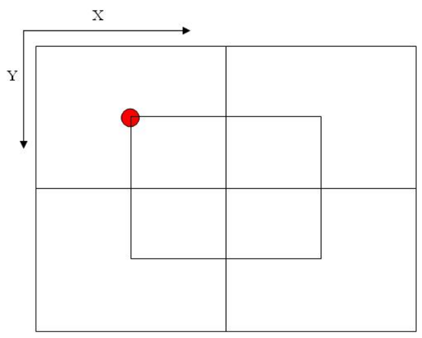

Fig. 5. Initial reference system

It is possible to modify the global origin and move it to any place in the reference system. To do this it is only necessary to put one mark at the new origin and to store the coordinates $< x, y >$ of the mark in the system. For example, if we want to set the origin of the reference system in the upper left corner, we need to place a mark at this site, take the position returned by the system and store it in the origin correction variable defined in the system (see Fig. 6).

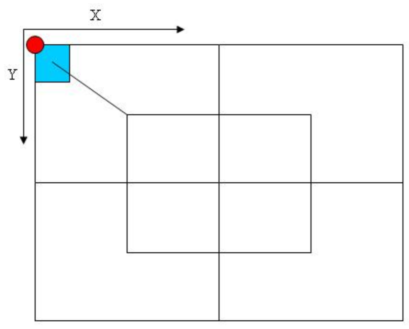

Fig. 6. Reference system based on a fixed point

F. Experimental results

First of all we adjusted the parameters of the camera (focus, brightness, saturation, etc.) and selected the right size of mark to get good results according to the distance between the ceiling and the floor. In our case, marks of 19x19cm were enough to detect them correctly.

To test our system, we drew a grid which divides the floor into 20x20cm squares and we set the initial position (0,0) in the top left corner of the grid. We started using only one mark, and we moved it around all the corners of the grid. We checked that the value returned by the system corresponded to the position in the real world. Next, we put four different marks at random corners of the grid several times and checked that all of them were correctly detected. In all the cases, the algorithm detected the correct mark and the error was always less than 2 cm.

In addition, three random tours for two different robots (an ER1 and an ERSP Scorpion) were used to test the system. Every second the current position was estimated by our system and was stored together with the images taken by the camera. Later, we manually checked the correspondence between the positions estimated by the system and the images taken for each point, showing that all the estimated positions corresponded with the real ones.

IV. Conclusions and future work

In this paper, we have presented an external localization and automatic evaluation system for robot localization algorithms. This system considerably reduces the time needed to develop a generic self-localization algorithm and it avoids the need to make manual measurements in order to evaluate its performance. In this way the system avoids typical human and manual measurement errors.

We have presented a common architecture to implement an external localization system based on web cameras. The system architecture proposed follows a modular definition. This makes it possible to develop each module independently, providing a flexible, easily modifiable and a scalable system.

In our case, we have implemented a system that can work with up to twenty cameras simultaneously. The system is able to detect any kind of element that the Robot Detection Module can find and estimate its position in real space using a mark detection library. Communication with the robot is performed using standard TCP sockets, so any robot that has a network interface is able to exchange localization data with the system.

To sum up, we have implemented a complete and robust external localization system that defines a general framework for developing and automatically evaluating self-localization algorithms to be used in mobile robots.

For the future we have in mind the modification of the robot detection module in order to allow the use of cameras placed in any position of the scenario, not only on the ceiling.

References

[1] "ARToolKit: Augmented Reality Tracking Library, http://www.hitl.washington.edu/artoolkit/."
[2] "ARToolKitPlus: Augmented Reality Tracking Library (phase 2), http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php."
[3] "DirectX 9.0 SDK, http://www.microsoft.com/downloads/details.aspx?FamilyID=77960733-06e9-47ba-914a-844575031b81&DisplayLang=en."
[4] "OpenCV. Intel Open Source Computer Vision Library. http://opencvlibrary.sourceforge.net/."
[5] "Programming computer vision applications, http://www.site.uottawa.ca/~laganier/tutorial/opencv+directshow/cvision.htm."
[6] "VideoInput, devenloped by Theodore Watson, http://muonics.net/school/spring05/videoInput/."
[7] R. L. Achtman, R. F. Hess, and Y.-Z. Wang, "Sensitivity for global shape detection," *Journal of Vision*, vol. 3, no. 10, pp. 616–624, 10 2003. [Online]. Available: http://journalofvision.org/3/10/4/
[8] R. Baczyk, A. Kasinski, and P. Skrzypczynski, "Vision-based mobile robot localization with simple artificial landmarks," in *Procs. of 7th IFAC Symp. on Robot Control*, S. P. E. Baczyk R., Kasinski A., Ed., Wroclaw, Poland, 2003, pp. 217–222.
[9] B. Bonev, M. Cazorla, F. Martín, and V. Matellán, "Portable autonomous walk calibration for 4-legged robots," *Applied Intelligence*, pp. 1–12.

[10] D. Fox, W. Burgard, and S. Thrun, “Active markov localization for mobile robots,” pp. 195–207, 1998.

[11] ——, “Markov localization for mobile robots in dynamic environments,” *Journal of Artificial Intelligence Research*, vol. 11, no. 391-427, p. 27, 1999.

[12] L. Freeston, “Applications of the kalman filter algorithm to robot localisation and world modelling,” *School of Electrical Engineering and Computer Science, The University of Newcastle, June*, 2002.

[13] P. S. Maybeck, *The Kalman filter: An introduction to concepts*. Springer-Verlag New York, Inc., 1990.

[14] R. R. Negenborn, “Kalman filters and robot localization,” Master’s thesis, Institute of Information and Computer Science, Utrecht University, Utrecht, Netherlands, 2003. [Online]. Available: citeseer.ist.psu.edu/negenborn03kalman.html

[15] T. Rofer and M. Jungel, *Vision-based fast and reactive Monte-Carlo localization*, 2003.

[16] D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison-Wesley Professional, August 2005. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0321335732

[17] I. Ulrich and I. R. Nourbakhsh, “Appearance-based obstacle detection with monocular color vision,” in *Proceedings of the AAAI National Conference on Artificial Intelligence, Austin, TX*, 2000, pp. 866–871.

[18] D. Wagner and D. Schmalstieg, “ARToolKitPlus for pose tracking on mobile devices,” in *Proceedings of the 12th Computer Vision Winter Workshop (CVWW’07)*, 2007.