A Very Low Cost Distributed Localization and Navigation System for a Mobile Robot

Riccardo Cassinis[1], Paolo Meriggi, Maria Panteghini
Department of Electronics for Automation
University of Brescia
Via Branze 38
25123 Brescia
Italy

Abstract – This paper presents one of the first applications of a novel message-oriented middleware, DCDT, that has been recently developed at University of Brescia in collaboration with Politecnico of Milano.

This simple and low-cost experimental application is aimed at detecting and localizing a mobile robot in indoor environments using a commercial off-the-shelf webcam and a couple of active markers.

The most important point is that the system takes advantage of already existing computing and communication resources, thus limiting the cost of the additional equipment to a negligible amount. DCDT, on the other hand, reduces the burden of coordinating the various tasks and of establishing communication links to a simple and straightforward task.

Some experimental results are included, that show good accuracy and reliability, despite the low resolution and quality of the used devices.

Index Terms – Mobile Robotics, Localization, Low Cost, DCDT, Distributed Systems, Active markers.

1. Introduction

Taking advantage of the pervasive and ubiquitous diffusion of digital devices, we recently introduced a novel approach [1] to mobile robotics. According to this approach, a mobile robot can be seen as the acting portion of a distributed system, embedded in a certain environment.

In contrast to what we called the "monolithic approach" of common mobile robotics design, our goal is the realization of light, simple and therefore low-cost mobile devices, that only carry on board actuators and sensors that are needed for a reliable exploitation of their tasks, embedding most of the remaining portion of sensors and processing units into the environment, using whenever possible resources that are already available for other reasons.

Of course, this approach needs the presence of a reliable and stable communication layer. This common layer among all the devices consists in a recently developed message oriented middleware, Device Communities Development Toolkit (DCDT); DCDT is in fact responsible for the delivery of messages throughout the system and, via a suitable API, manages each single task running on the processing units.

In this paper we describe one of the first real experiences that used DCDT in mobile robotics in order to realize a small device community: the experiment, designed and developed at the Advanced Robotics Lab of the University of Brescia, basically consisted in navigating a blind mobile robot (carrying a couple of active markers) via the localizations offered by a webcam and a PC, using the laboratory LAN as the communication means.

The low cost of the hardware used, the ease of development and the modularity of the Device Communities approach attained, make the system very appealing for several robotics applications, especially in the consumer market. Despite the economy of the system, however, the experimental results are very promising, especially if seen in the perspective of a more extensive use of DCDT devices interacting with each other.

2. Description of the system

The system we have designed and implemented is depicted in figure 1 on the left: an omnidirectional sensorless mobile robot, MARMOT, that only carries onboard a small microcontroller unit, is connected via a serial line

[1] Corresponding author: riccardo.cassinis@unibs.it

(soon to be replaced by a wireless IEEE 802.1 link) to a PC that runs a suitable DCDT module, performing computations required to drive the robot along pre-defined paths to reach assigned set-point.

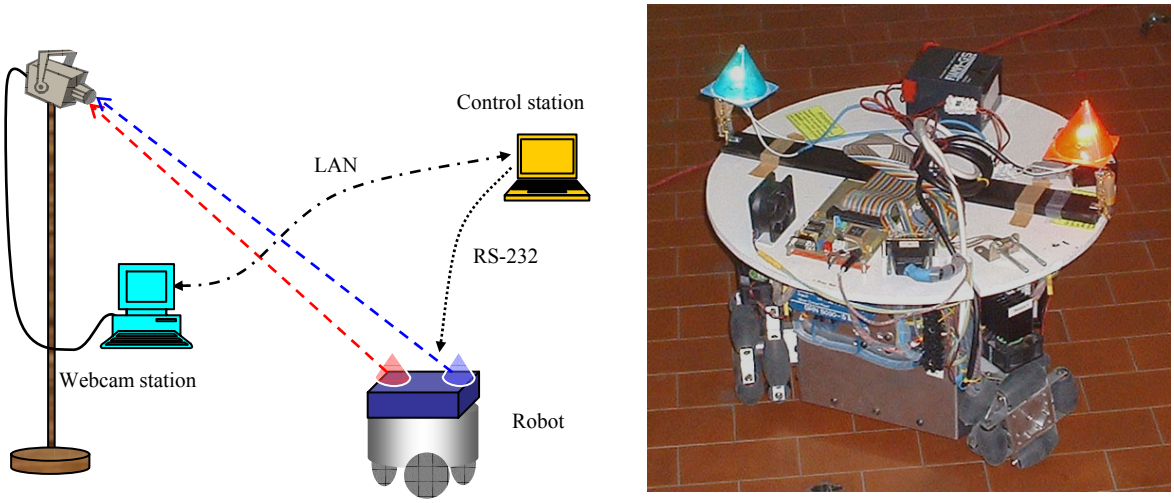

Fig. 1. Schematic representation of the system (on the left). The omnidirectional robot MARMOT (Mobile Advanced Robot for Multiple Office Tasks), equipped with the two active markers (on the right).

MARMOT (Fig. 1 on the right) is an omnidirectional robot designed and realized at Advanced Robotics Laboratory (ARL) of the University of Brescia. It is built with three Mecanum wheels [2] (recently used also in [3]). These wheels are driven by three stepping motors, controlled by a Motorola 68HC11 Microcontroller, that receives motion commands via an RS-232 serial link.

Two active markers were placed on the top of the robot, and are used to visually localize the robot. They were realized using two 12V halogen bulbs enclosed in small cones of colored plastic film. The chosen colors are red and blue, in order to maximize their distance in the RGB color space, hence enhancing the probability to correctly detect the markers.

Robot localization is exploited by means of a commercial off-the-shelf webcam: Philips ToUcam Pro PCVC740K, priced well under € 100, with a resolution of 640x480 pixels 256 colors and a frame rate up to 30 fps. This camera is connected via a USB link to a second PC, where another DCDT Member processes incoming images and provides the required stream of coordinates. Both computers are common PC connected with each other and to the Internet over a 100 Mbit/s Ethernet LAN.

On both PC stations, appropriate DCDT Members run graphical user interfaces, providing the users an easy access to the system; the most important interface of the two is indeed the one on the Webcam station, through which the user may define the set-points sequence to be sent to the robot.

3. An Application of DCDT - Devices Communities Development Toolkit

The middleware DCDT [1] has been designed and realized to allow the joint use of several interconnected devices, which could interact each other in a transparent way, regardless of the physical communication means used.

By exploiting the abstraction features offered by Object Oriented Programming, DCDT provides the user a suitable Application Programming Interface (API) to easily embed software agents into active and independent modules. Modules, called *Members*, are software objects whose methods are executed in a concurrent way by threads, and communicate with each other by sending and receiving messages according to a publish/subscribe mechanism. Members can then be divided basically into two categories: user members, whose methods are programmed by the user, and system members, which perform those background functionalities (just like dispatching messages, contacting and keeping the communications with other devices, etc.).

Communication among different devices is carried out in a completely transparent way through suitable software objects, called Channels, which are abstractions of the underlying physical means actually used.

All the modules and the objects required by the architecture, are contained within the main object, called Agora. The whole set of the interacting members (i.e. *agents*) on the same or different Agorae is referred to as the *DCDT Community*.

As reported in Fig. 2, in this experiment the DCDT community is composed by four user members: two on the Webcam Station (in Agora W), and two on the Control Station (in Agora C). On each Agora, one member is functional to the robot localization and motion (i.e. *Webcam Member* on the Webcam Station and *Control Member* on the Control Station in 0), while the other is the graphical user interface (GUI).

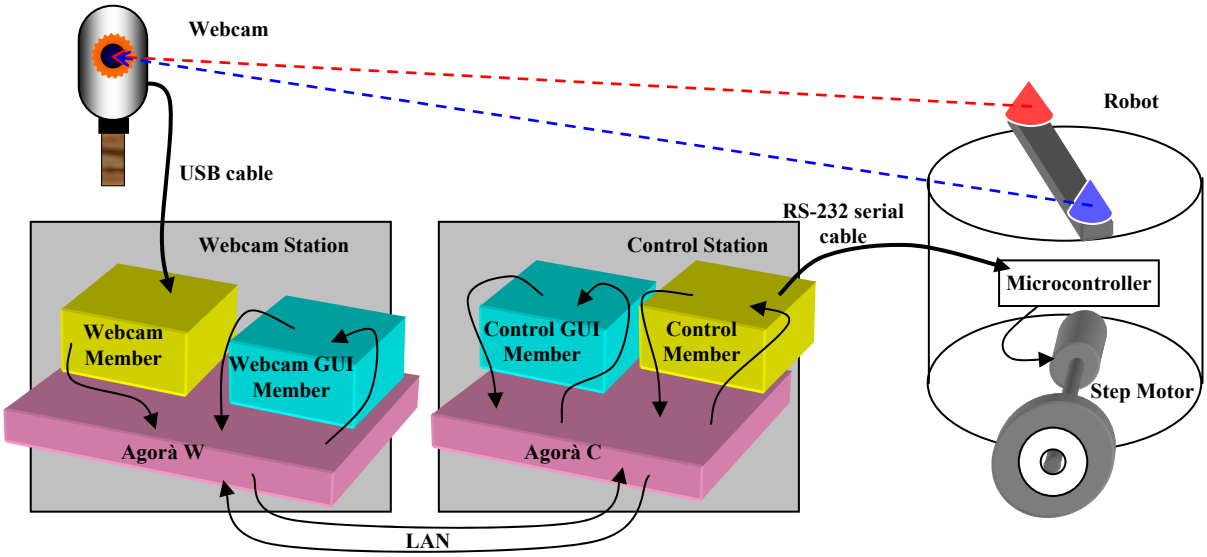

Fig. 2. Schematic representation of the webcam experiment

Note that some members could interact directly with other devices: Control Member on Agora C sends directly motion commands to the microcontroller unit via serial link (since a DCDT extension for the 68HC11 microcontroller is not available yet) and the Webcam Member on Agora W acquires directly the images from the USB camera by mean of suitable OS driver.

The interactions among user Members in this experiment could be easily sketched according the following pattern

1. The Webcam Member acquires the image and calculates the position of the centroid and the orientation of the segment between the two lights (when possible), providing the whole community with the requested coordinates, according to the process that will be described in next section.
2. The GUI Member on Agora W waits for the initial position of the robot to be received and the presence of the Control GUI on Agora C. When it receives a suitable signal from Agora C, it prompts for the setpoint sequence to be reached by the robot, and then starts sending each single setpoint to be reached, waiting for a message from Control Member on Agora C, reporting the conclusion of the path.
3. The Control Member on Agora C continuously receives the actual coordinates of the robot and adjust the motion command sent to the Microcontroller in order to reduce the Euclidean distance between them. When such distance falls within a predefined constant radius (e.g. 20 mm), the message of *Setpoint Reached* is shared among the communities, triggering the GUI Member on Agora W to send the next setpoint.

The GUI Member on Agora C during the experiment has been simply devoted to the visualization of the message traffic, mainly for debugging purposes, for its native goal should be the direct management of the robot motion by a suitable interface panel, in case of communication failure with the Agora W.

4. Recognition Algorithm

The recognition algorithm was developed as an M.Sc. thesis work [4] and basically consists in 5 steps: acquisition, RGB conversion, thresholding, identification and transformation.

- **Acquisition** : During this step the image is acquired from the webcam and saved in a suitable memory area available to the Webcam Member. Because of the webcam features, the image is 640x480 pixels, coded according the YUV420P[2] format. Using active markers results very useful, since it is possible to tune the shutter speed to the amount of energy emitted by the lamps, greatly reducing the false positive detection.
- **RGB Conversion**: this step was found to be necessary because of the really complex calculations required

[2] also known as "I420", 12bpp, basic planar format, Y-U-V (Y/Cb/Cr) order

when using the plain Y-U-V image format natively provided by the webcam. Instead, working with RGB images and searching for red and blue blobs becomes simply a matter of thresholds, greatly reducing calculations to much easier comparisons.

- **Thresholding**: after the conversion, the whole image is then thresholded, leaving only four different colors: red and blue for those pixels with only one of the chromatic components over threshold, white for those ones with both red and blue components over threshold and black for the remaining pixels. The aim of this operation is to deliver an image where it is much easier (i.e. faster) to detect the region of interest (ROI). In average conditions, these ROIs are represented by a white blob surrounded by red or blue coronas, as in Fig.3.

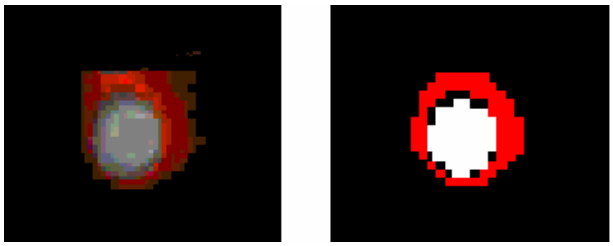

Fig.3. Difference between the image of the red light before and after thresholding

- **Identification** : In this phase the acquired image is processed according to a blob-growing algorithm, in order to characterize the red and blue ROIs. The detected ROIs (which could eventually be many more than two) are carefully analysed in order to reject false assignaments (e.g. leds, neon lights or sun reflections): for this reason, some morphological parameters of the ROIs are calculated:

1. max/min dimension ratio of white pixels blob
2. ratio of the number of blob's white pixels and the area of the smallest square containing the blob
3. difference between the height and width of the white blob
4. max/min ratio of colored pixels
5. amount of pixels with colors different from the blob's average coloring, which might be due to the other marker or from natural lightning.
6. amount of white and colored pixels in the image, that indicate the possible abnormal presence of enilghtment (for instance to the sun light, etc.)

If all these morphological parameters are within predefined limits, the coordinates of ROI centroid in the image space are calculated and delivered. Since the pattern we are searching is basically a small group of white pixels surrounded by colored ones, we calculate the centroids of the white blobs:

$$x_{centroid\_red} = \sum_{i=1}^{N} x_i \,, \qquad y_{centroid\_red} = \sum_{i=1}^{N} y_i$$

$$x_{centroid\_blue} = \sum_{i=1}^{N} x_i \,, \qquad y_{centroid\_blue} = \sum_{i=1}^{N} y_i$$

where N is the number of white pixels and x_i , y_i represents their position in image coordinates.

- **Transformation** : Image space coordinates $[x_F \;\; y_F]$ are transformed in real world Cartesian coordinates $[x_W \;\; y_W \;\; z_W]$ according to the following calculations sequence. First, we have to calculate the distorted coordinates $[x_D \;\; y_D]$ using their relationship with the image coordinates $[x_F \;\; y_F]$, derived from a theoretical model of the webcam that doesn't consider parameters like s_X, empirically chosen by the user, that models the difference in scale of the two axis of the camera's sensitive area and d'_X and d_Y:

$$x_D = d'_X (x_F - C_X) s_X^{-1}$$
$$y_D = d_Y (y_F - C_Y)$$
$$d'_X = d_X \frac{N_{CX}}{N_{CY}}$$

where d_Y is the distance between two adjacent CCD elements, N_{CX} and N_{CY} the number of sensor elements

along X and Y directions, C_X and C_Y the coordinates, in pixel, of the real centre of the image.

After that, we obtain the undistorted coordinates $[x_U \ y_U]$ from $[x_D \ y_D]$ using two parameters, D_X and D_Y, that come from the first-order radial distortion coefficient κ_1.

$$x_U = x_D + D_X$$
$$y_U = y_D + D_Y$$

where

$$D_X = x_D(\kappa_1 r^2 + \ldots)$$
$$D_Y = y_D(\kappa_1 r^2 + \ldots)$$

and

$$r = \sqrt{x_D^2 + y_D^2}$$

Now we can find the first relationship with the real-world coordinates writing these equations that model the theoretical perspective projection having as a fundamental parameter the focal length *f*:

$$x_C = \frac{x_U z_C}{f} \qquad y_C = \frac{y_U z_C}{f}$$

The triplet $[x_C \ y_C \ z_C]$ represents the world coordinates expressed in a camera-centered coordinate system. The third coordinate, z_C, is in effect a parameter of the system, being always constant: this is due to the limitations in the motion of the robot, that cannot move along the Z axis; thus, knowing the height of the lights from the ground, real world coordinates are calculated by a matrix multiplication in a 2D → 2D transformation. This is also the reason why we can localize and navigate the robot using a single camera.

The final step allow the delivering of coordinates $[x_W \ y_W \ z_W]$ in the Cartesian system chosen by the user:

$$\begin{bmatrix} x_W \\ y_W \\ z_W \end{bmatrix} = R^{-1}\left(\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} - T\right)$$

R and *T* are, respectively, the rotation matrix and the translation vector of the webcam.

All these steps, quite simple during runtime, require a very careful calibration of the system for a good accuracy; there are in fact many parameters related to the position and inclination of the webcam, its focal length, its radial distortion coefficient and CCD discrete acquisition, that may easily affect the transformation matrix, with the result of providing completely wrong coordinates.

Once the coordinates of the two lights in real cartesian space have been calculated, the position and orientation of the robot are computed, according to the following formulae:

$$x_{robot} = \frac{x_{red} + x_{blue}}{2} \qquad y_{robot} = \frac{y_{red} + y_{blue}}{2}$$

$$orientation_{robot} = arctg\left(\frac{y_{red} - y_{blue}}{x_{red} - x_{blue}}\right)$$

Eventually, a final minor correction is applied, that takes into account the distance between the lights, that is fixed and known.

5. Calibration of the system

As previously described, the system calibration is a mandatory procedure for the algorithm to properly work.

The calibration procedure we have used is based on the mathematical model referred to as *simplified photogrammetric*, introduced by R.Y Tsai [5], which is particularly adherent to the characteristics of the low cost camera used (webcam), taking into consideration all those parameters like radial distortion, real image center, etc.

The calibration procedure is really simple to work out, requiring to establish a one-to-one correspondence between a certain set of points (for instance belonging to a grid) in the real Cartesian space and in the camera

image space. A certain number of these correspondences, in fact, might be used to calculate the intrinsic and extrinsic parameters of the camera, which play a key role in the runtime transformations from the image space to the real Cartesian one.

Intrinsic parameters are represented by focal length, radial distortion coefficient, actual center of the pixel image, etc.

Extrinsic parameters are instead the rotation and translation of the camera with respect to the absolute coordinate system used as Cartesian reference.

According to some literature examples (i.e. [6]), we found that 11 is the least number of points required, even if a higher number is often used in order to achieve an accurate precision: in a high-precision experiment, regarding the localization on a plane close to the camera, Tsai evaluated about 60 different points.

In order to ease the calibration task and to reduce the required time, given the amount of different measurements to be evaluated, a suitable GUI has been realized.

6. Experimental results

The experiment was really succesfull, thanks to the precise localization algorithm that showed average errors under 1% of the distance between the robot and the webcam (measured in static conditions), allowing several trials to be performed correctly with low latencies both on the processing and transmission.

The choice of an active marker proved to be very good, with high stability in recognition performances, that were quite independent from environment lighting.

The overhead due to DCDT architecture resulted very small and absolutely acceptable for a correct navigation of the robot, proving that the ***device communities*** approach could actually find useful implementations. In fact, the overall low cost of the system, the use of means that are normally available in many office environments (such as personal computers, lan's, etc.), as well as the smooth motion of the robot and the ease of development of various *DCDT Members*, open new and interesting perspectives.

Moreover, from a timing performance point of view, on a common 1,1 GHz Pentium III computer, we reached with non-optimized executables the following results, that seem quite interesting, since they allow an average detection rate of up to 14 frames per second.

	Average time (ms)	**Maximum time (ms)**
Acquisition	10	10
RGB Conversion and Thresholding	25	30
Identification	10	20
Coordinate transformation	25	30
Total Time	70	90

Obviously, the system has also some limitations, basically related to the quality of the images and the time required by the five processing steps previously described: dealing with a commercial webcam, means low quality and resolution in the image. Also, the power of the active marker had to be limited, due to the fact that the robot is battery-driven and its energy supply is limited. The limited dynamics of the non-professional camera become another constraint, since the characteristics of the image acquired in function of the distance, may be very different, as reported in Fig. 4, and will eventually prevent a proper detection of the lights in the far field (about 8 meters and above from the webcam standpoint).

Fig. 4. Images of the red light at different distances from the webcam

For these reasons, the system may perform quite well only in indoor plain environments, where the external light (i.e. the sun) and white reflexes are almost absent. Moreover, the robot needs to move in a confined space (up to 8 m from the webcam) at a reasonable speed according to the camera frame rate and the processing power available. In our case, we found that an average speed of 500-800 mm per second could be quite precisely tracked by the system. It is obviously possible to use multiple cameras, as it will be done in future experiments.

7. Conclusions and future works

In this paper we presented a first simple implementation example of a recently developed middleware, called DCDT, to foster what we have named Community of devices, leading to the realization of feasible and low cost mobile robotics infrastructures.

Being only at the beginning of the research, there are many possible evolutions and way of improving the system:

- The realized code might be optimized with the result of faster execution and better timing performances;
- More sophisticated filtering approaches (such as Kalman or nonlinear techniques) could be implemented in order to enhance the detection of the colored blob centroid and reject false positives ;
- Expand the system with the presence of more than one camera covering a broader area and connecting the robot via IEEE802.11b wireless network;
- Test the improvement which can be achieved using cameras with greater dynamics;
- Enhance the robustness of the system by toggling on/off or otherwise modulating the emissions of markers in order to have the system recognize and properly localize the robot, for instance if in presence of false recognitions.

Nevertheless, the results achieved even in this early stage in term of precision reached, low cost, ease of implementation, are indeed promising and suggests several possible industrial applications.

References

[1] R. Cassinis, P. Meriggi, A. Bonarini, and M. Matteucci, "Devices Community Development Toolkit: An Introduction," presented at Eurobot '01, Lund, Sweden, 2001.

[2] G. Wampfler, M. Salecker, and J. Wittenburg, "Kinematics, Dynamics, and Control of Omnidirectional Vehicles with Mecanum Wheels," *Mechanics of Structures and Machines*, vol. 17, pp. 165-177, 1989.

[3] P. d. Pascalis, M. Ferraresso, M. Lorenzetti, A. Modolo, M. Peluso, R. Polesel, R. Rosati, N. Scattolin, A. Speranzon, and W. Zanette, "Golem Team in Middle-Sized Robots League," in *RoboCup-2000: RObot Soccer World Cup (IV)*, G. Kraetszchmar, Ed. Berlin: Springer Verlag, 2001, pp. 603-606.

[4] M. Cagno and M. Panteghini, "Un Sistema Distribuito per la Localizzazione di Robot Mobili dotati di Marker attivi," in *Department of Electronics for Automation*. Brescia, Italy: University of Brescia, 2002.

[5] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 4, pp. 323-334, 1987.

[6] H. Bakestein, "A Complete DLT-based Camera Calibration with Virtual 3D Calibration Object," in *Department of Mathematics and Physics,*. Prague: Charles University, 1999.