Localization of a Mobile Autonomous Robot Based on Image Analysis

F. Ribeiro, *Member, IEEE*, L.Tosini and G. Lopes

***Abstract*— This paper introduces an innovative method to solve the problem of self localization of a mobile autonomous robot, and in particular a case study is carried out for robot localization in a RoboCup field environment.**

The approach here described is completely different from other methods currently used in RoboCup, since it is only based on the use of images and does not involve the use of techniques like Monte Carlo or other probabilistic approaches.

This method is simple, acceptably efficient for the purpose it was created, and uses a relatively low computational time to calculate.

I. Introduction

ROBOCUP is an international joint project to promote AI, robotic and related fields. The challenge is quite amazing, and the people involved in this project have thrown another amazing challenge which consists of developing a fully autonomous humanoid robots team able to challenge for a win against the human world champion in soccer by the year 2050 [1].

This aim is very difficult because today there are a lot of open problems in the field of autonomous robotic and the humanoid league is still far away to reach the level needed to challenge a human team.

One of the major open problems today is the robot field localization using only the data available from the robot mounted sensor, for example vision system or odometry. Localization is the process by which a mobile robot determines its own position and orientation with respect to a reference system within a certain environment. This is an essential capability for autonomous physical agents in several application domains, which are often characterized by structured or semi-structured surroundings. A localization algorithm which relies on an initial estimate and required to be quite an accurate estimate of the real pose is known as performing a local localization; on the other hand, operating without an initial estimate is known as global localization. Local localization is an easier task, but it is not sufficient for the applications of our interest, because the robot position in an absolute coordinate system is needed in order to integrate high level routines like a team strategy.

Another problem that involves the robot localization is named "kidnapped problem", when the robot after committing a foul is moved by the referee into another position, or in other words the robot is replaced in an unknown position within a known environment. Therefore all the localization algorithms used today, need a small amount of time to estimate the new robot position in order to obtain the same samples from the sensors.

This work proposes a very different solution from the method that other teams are currently using. Today's trend consists of using a localization system that uses data gathered by different types of sensors, and using them to achieve reduced data noise due to the use of different data type. For example, it is impossible to obtain good results using odometry alone because the precision of this method decreases very quickly with time and besides should a robot hit a wall (or another robot) counts will be lost and the odometry is not able to recover from this situation.

The method described in this paper is based only on the data acquired from a vision system. This is possible because the image acquired has a low amount of noise and the vision feasibility is high, since the image is updated all the time (a frame is grabbed every 20 milliseconds) describing always a real representation of the game field.

II. BACKGROUND

Most vision systems used today in RoboCup are similar because the solution found is very smart and less complicated than a stereoscopic vision system. The vision system is composed by a normal camera oriented towards an hyperbolic mirror that is placed over the camera, so the image obtained is omni directional [2], permitting to see the entire game field at the same time.

[1] Manuscript received September 14, 2006. This work was supported in part by the Portuguese Ministry for Science and Superior Education by FCT (Fundação para a Ciência e Tecnologia) under Grant POSI (Programa Operacional de Quadro Comunitário de Apoio pertinente) - Project POSI/ROBO/43892/2002.

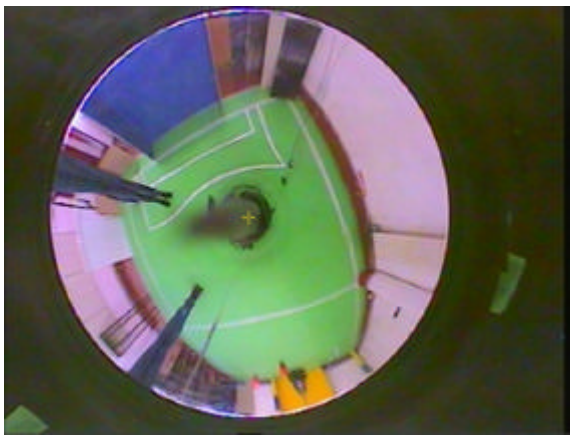

Fig. 1. Example of an omni-directional image

After the image being grabbed by a robot's computer mounted frame grabber, and ready to be processed by the vision system, this module tries to gather information from the image just by analyzing it. One example of information to acquire is the robot position within the field.

Most RoboCup teams use as localization systems the image acquired to determine or help finding the position of the robot on the field. A case study was carried out based on three RoboCup localization systems.

Brainstorm TriBots: This team localization method uses a particle filter system that relies on features found on the image [3]. The features are points on field lines, which can be easily and reliably recognized under natural light conditions. The features are detected along rays radially arranged around the omni-directional image centre. Using particle filter methods permit to estimate the position of the robot very precisely on a 16x10 meters large outdoor field. For example driving and turning the robot (using omni-directional drive) for 30 seconds across the field with a speed of 1.5 m/s, the self-localization deviates only approximately 20 cm on average from the reference path (with maximal deviation of 50 cm).

Milan RoboCup Team: This team use a localization system named MUREA (MUlti-Resolution Evidence Accumulation) and it consists of a mobile robot localization method for known 2D environments [4]. The localization space is divided into sub regions (cells) and then it applies an evidence accumulation method, where each perception votes those cells that are compatible with the map of the environment. In order to reduce the complexity of working with a fine grid, they have adopted a multi-resolution scheme. They start applying evidence accumulation on a coarse grid, with few large cells, then they select and refine (i.e., divide into smaller cells) those cells that have collected highest votes. The main characteristics of MUREA are the system capacity to integrate sensor data coming from different sensors and the system capacity to be both a global and a local localization system, depending on the availability of a global estimate. The results of this method are very good, the average localization errors is less than 10 cm for position and about 1° for orientation [5][6].

Philips Team: Actually the localization system of this team is very simple, because it is based only on odometry. It is a very fast method but also very noisy, due to the odometry sensors that are inaccurate when a wheel slips or bounces against an obstacle. This method is based on sensors mounted on the wheels that measure the space covered by the robot and by another sensor that records the robot pose. By joining this information it is possible to estimate the robot position, however the robots do not have a global position but just a local one.

All methods here described are different but with some degree of success. The Brainstorm TriBots team uses vision as the main sensory source to acquire data which is processed by a particle filter system. Even though they obtain good performance results, the time needed to calculate the robot position can be high due to the system complexity and this disadvantage can limit its acceptance in some teams [7].

The system used by the Milan RoboCup team is innovative and it seems to be very efficient. The idea of joining much data acquired by different sensors together with the use of a multi resolution method is good, because should there be not enough time to calculate the exact robot position, it is possible to find a non optimal location.

The method used by the Philips RoboCup Team is the simplest one, it is easy to implement since it is based on odometry but it is a very noisy method.

Also, techniques known as "Monte Carlo" are very popular in RoboCup because it is a well know technique, reliable and not so time consuming like other kinds of approaches. To really understand why the approach of this work is different from the "Monte Carlo" method, it is important to illustrate the "Monte Carlo" technique.

A. Monte Carlo Localization

Monte Carlo localization, from here onwards MCL, is a probabilistic method based on Bayesian Filtering and it calculates the probability density of the robot position, or belief, and iteratively propagates this probability density using motion and sensor information. Usually the motion data is obtained from the robot odometry but the sensor information depends on which type of sensor is used.

It is usual in RoboCup to use odometry to supply motion data and the camera to supply sensor information. The belief about the robot position is represented with a set of discrete points in the C-space of the robot, and these points are called samples. These samples are updated over time so the belief about the robot position is updated. To each sample corresponds also a weight that is the probability of the robot is occupying that sample. A Sample with a low weight corresponds to a low probability of the robot occupying that

position, while a sample with a high weight corresponds to most probable robot position. To update the belief the knowledge of two conditional densities, called motion model and sensor model, is needed. The motion model is a probabilistic representation of the robot's kinematics which describes a posterior density over possible following robot's poses.

Updating the robot's position according to the kinematics only does not take into account errors given by odometric inaccuracy and possible collisions of the robot with obstacles. Therefore, usually a random noise term is added to the values given by the odometry. Usually the noise term is modelled with Gaussian zero centred random variables, dependent on both amount of translation and rotation.

The sensor model describes the probability for taking certain sensor measurements at certain poses: and as introduced before it strongly depends on the particular sensor used. The localization algorithm basically consists of three steps:

1) All particles are moved according to the motion model of the last kinematics measurement.
2) The weights of the particles are determined according to the observation model for the current sensor reading.
3) A re-sampling step is performed: high probability particles are multiplied, low probability ones are discarded.

Finally, the process repeats from the beginning.

The samples generated at the beginning of the algorithm are randomly generated over all possible positions, but this can be a disadvantage because at the beginning all samples have the same weight and therefore an accurate localization is difficult to achieve. Besides, in a "kidnapped situation" the robot needs some time to gather some data in order to supply a position.

Regarding the sensor model, many approaches have been studied and applied, and the simplest one with the hypothesis of using a camera like sensor is now explained.

The image grabbed from the vision system is analyzed and all the colour transitions are memorized, usually only the kind of transition (for example lines, corners or score posts) and the angle of the transition detection are used and when the whole image has been analyzed the information obtained is applied to all samples. Should the sample match with the information obtained from the sensor, the weight of that sample is increased, otherwise the weight is decreased.

When all the samples have been analyzed it is necessary a normalization of all weights, in order to get the sum equal to one, because the weight represents probability of one event, after this the algorithm proceeds like explained before.

As illustrated in this section the MCL is a reliable, easy and well know method to solve the problem of autonomous agent localization. This approach is not used by Minho Team for two main reasons: the first is due to unavailable odometric system and so the motion model is quite difficult to obtain. Using only the command sent to the motion system, for example "turn right 90° and proceed for 10 meters" without a feedback is possibly imprecise due to the noise and robot motion inaccuracy (wheels slipping). The second reason is due to the processing time needed by the MCL approach in a "kidnapped" situation.

III. Localization Algorithm

This section illustrates a new proposal for the self localization robot challenge, based on the analysis of an omni-directional image acquired from the robot vision system.

The algorithm can be divided into five consecutive sequential steps:

1) The image is analyzed to find the colour transitions that represent lines or goals.
2) The detected points are processed to find the position and direction of the goals.
3) The robot absolute direction is calculated.
4) The detected points are post-processed in order to apply the consecutive steps.
5) The absolute robot position is calculated.

A. First Step – Image Analysis

To start, a grabbed image is analyzed along radial virtual lines from the image centre outwards. The angle between two consecutive lines and the space between two different analyzed points lying in the same line are parametrically input beforehand. For this example, a value of three degrees between the lines and analyzing one pixel every two is the right trade-off between processing time and accuracy.

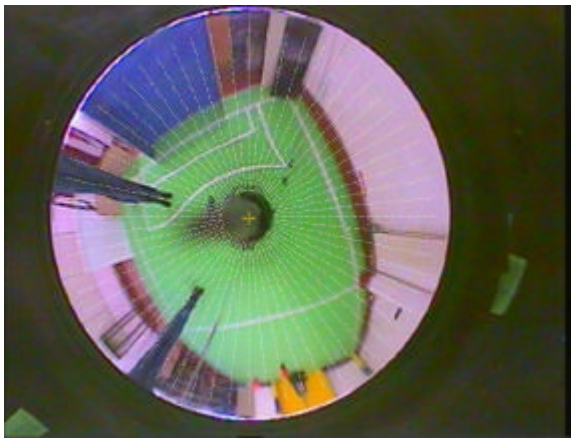

Fig. 2. Virtual lines drawn over the omni-directional image

The image is analyzed using the intersection points between the virtual lines and the entities on the field, starting from the lines at zero degrees and proceeding until all the lines have

been analyzed. The scan of each line starts from the point closer to the image centre and proceeds outwards.

For every analyzed pixel a function to filter the pixel colour is used. This function must be set-up before every test or game in order to have a very precise colour classification. This function can differentiate between the two goals colour, the lines and the field, and must be set-up properly for accurate results.

If during the same scan line a series of consecutive points is found, yellow, blue or white, this series of points is recorded until the next analyzed position is of different colour. Should the number of consecutive points be greater than a threshold, the closest point to the image centre is taken if it is a score post, or the medium position if it is a line. A structure is created in memory for every interesting point containing the coordinate and the line angle to which the point belongs.

B. *Second Step – Score Posts Detection*

After a complete scan of all lines on the image, some post processing procedures are necessary to obtain the goals representative points. This procedure marks the yellow and blue points belonging to the same line, because they probably indicate a corner and not a goal. Afterwards, if on the same line two or more interesting points are detected, only the closest to the image centre is taken.

After the whole image scan and the post processing phase is completed, a structure in memory is built where for every relevant point detected it contains the image point coordinate and its angle.

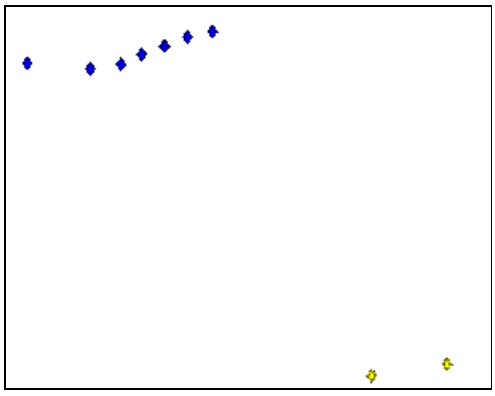

Fig. 3. Relevant detected points for the goals

The next step is identifying the goals position by calculating the average of all coordinates detected of the same colour, to obtain just one point for each goal.

If one or two points are available, the algorithm calculates the distance and the angle of the points according to the image centre (robot's centre); to obtain the distance of the point a distance function is used, which supplies the Cartesian distance in pixels distant on the image. This function uses only the Cartesian distance in pixels and not the angle because the mirror used is circular and therefore the distance is not a function of the angle but only a distance in pixels. By now, the algorithm supplies for every goal detected the distance and the angle according to the robot's position.

C. *Third step – Absolute robot's direction*

The absolute robot direction calculation is the next step and this is usually sorted out by the localization algorithm. A simple method is used to find the robot direction and this is based only on the captured image. The omni-directional image obtained by the camera can be transformed in a planar one, like the following picture illustrates.

Fig. 4. Planar view of robot camera

It is possible to see on this image vertical dotted lines that divide the image in 45° degrees sectors and the actual robot direction is represented by the centre line.

To obtain the robot direction it is necessary that both goals are visible, and this happens over 90% of the time. If both goals are visible, the Cartesian distance between each goal centre is calculated as well as the distance between the vertical centre line and the nearest goal (in this example the blue goal is the nearest one).

Let's call the *total_dist* the distance between the two goals and the *post_dist* the distance between the nearest goal and the vertical centre line. The robot direction is given by:

$$180 : total\_dist = direction : goal\_dist \qquad (1)$$

$$direction = \frac{180 \cdot goal\_dist}{total\_dist} \qquad (2)$$

D. *Fourth step – Post processing of detected line points*

During the image scan the points belonging to the field lines were also recorded.

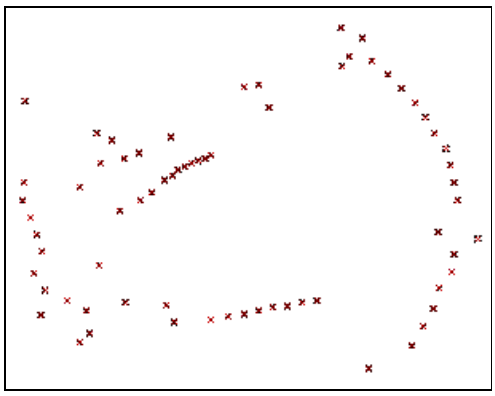

Fig. 5. Relevant detected points for field lines

The lines perception is complex due to mirror geometry. The lines appear like curves and traditional algorithms to find lines in an image are not suitable here. To avoid this problem, to each point detected is applied the distance function used for the goals, obtaining an image where the detected points appear to be in an almost straight line.

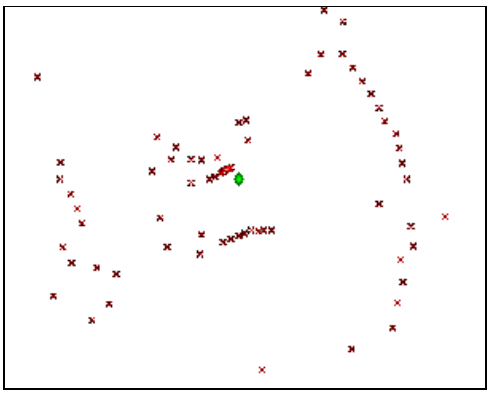

Fig. 6. Points of the lines after using the distance function

The last algorithm step calculates the absolute robot direction, allowing image rotation independence, in order to obtain a representation of those points. Applying the two formulas:

$$\begin{cases} x_{rot} = x_{orig} \cdot \cos(a) - y_{orig} \cdot \sin(a) \\ y_{rot} = x_{orig} \cdot \sin(a) + y_{orig} \cdot \cos(a) \end{cases} \quad (3)$$

The following image is obtained.

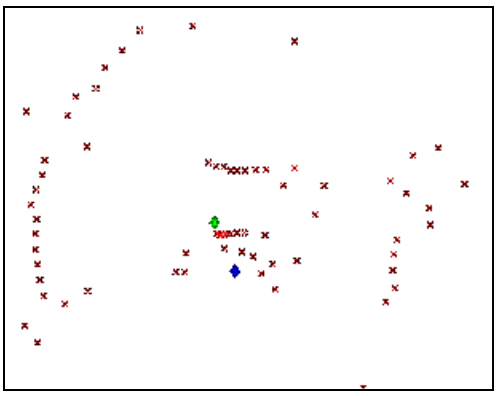

Fig. 7. The rotated image with the points detected

The green diamond represents the robot's position and the blue star represents the blue goal. Next, it is carried out the absolute robot position detection, addressed in the next step.

E. Fifth step – Robot Absolute Position

The solution is based on a pattern-matching problem between the data obtained from the image analysis and a field model stored in memory. This can be a time consuming approach but usually the relevant number of points found on the image are only between 40 and 60 and so the time needed is quite short.

Both representations have to be in the same scale. It was decided to use a matrix to describe the model field, where each cell corresponds to a 5 by 5 cm field area. Each unit has three different values; one to describe the presence of line and two for the different goals. A different value is used for cells that do not represent lines or goals.

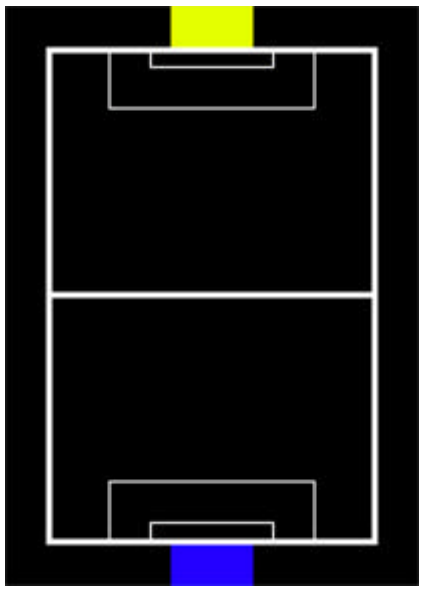

Fig. 8. Model of the game field

The data gathered from the camera has to be converted in this representation but that is simple because the values are

already in memory. For each detected point, the structure contains the point coordinate expressed as distance from the unknown robot position and its translation in a grid representation.

The idea beyond pattern-matching is simple and it consists of comparing a model representation and a partial representation of data belonging to this model to find the position of the pattern (data) inside the model representation.

Artificial vision pattern-matching is normally carried out using convolution or correlation but these approaches are very time-consuming. Using a matrix representation of both the model and the data permits to exponentially decrease the algorithm processing time needed.

In this work it is not necessary to use convolution to find the best match between the field model and the relevant points found. At this stage, the only secure information the algorithm has is the goals position, so the algorithm takes the closest goal and the corresponding points in the model. As an example, if the yellow goal is closer, the algorithm takes all points corresponding to the yellow rectangle on the model. Then, for each point of the rectangle (yellow or blue) the coordinates of the points detected are converted to fit on the grid. It is also checked in the model if the point is a line and in this case a counter is incremented.

When all the points detected have been checked, if the counter is greater than the previous value, the old value is replaced by the new one and the robot position is calculated according to the model. When all the points of the rectangle have been processed the optimal robot position has been found and it corresponds to the location where most points fit on the model.

IV. Tests and conclusions

This algorithm is still at prototype level, it has been implemented in Matlab, running on a Pentium-M 1400 with 512 Mb of RAM. Using an image of 640 x 480 pixels, the time needed by the algorithm is around 0.2 seconds. Running the algorithm on the RoboCup robot dedicated computer and written in C language, the time needed will be shorter, probably about 0.1 seconds, and this shows that the algorithm is acceptably fast to run in real time.

In the tests carried out the error was less than 1 meter, Even though the robot vision system was not properly calibrated and the distance function used was not the correct one for the new mirror used on the tests. For all these reasons, the team believes that, if these features are improved the algorithm will be much more precise. This is a new method to solve the particular problem of robot localization, the method is fast, robust, and need only data from the vision system.

In this paper the method is applied to a RoboCup robot. This is an optimal situation because a game field model is easy to obtain. This work can be applied to other indoor environment, but it will not be easy to implement in an outside environment where a complete model of the environment is almost impossible to obtain.

Acknowledgment

This work has been partly founded by the FCT (Fundação para a Ciência e Tecnologia) and the program POSI (Programa Operacional de Quadro Comunitário de Apoio pertinente) - Project POSI/ROBO/43892/2002.

The University of Minho, School of Engineering in Guimarães city, the Department of Industrial Electronics and the Algoritmi research centre deserve our acknowledgement for all the efforts.

References

[1] http://www.robocup.org.

[2] P. Lima, A. Bonarini, C. Machado, F. M. Marchese, C. Marques, F. Ribeiro, D. G. Sorrenti, “Omni-Directional Catadioptric Vision for Soccer Robots”, Special Issue of the Robotics and Autonomous Systems Journal, Elsevier, Amesterdão, Volume 36, nº 2, 31 de Agosto de 2001.

[3] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, Monte carlo localization: Efficient position estimation for mobile robots, AAAI-99.

[4] E. Menegatti, A. Pretto, E. Pagello A New Omnidirectional Vision Sensor for Monte-Carlo Localization Proc. of the 2004 Int. RoboCup Symposium CD-ROM (13 pages).

[5] Marcello Restelli, Domenico G. Sorrenti, Fabio M. Marchese - Evidence Accumulation Method for Mobile Robot Localization.

[6] E. Menegatti, A. Pretto, E. Pagello - Testing omnidirectional vision-based Monte-Carlo Localization under occlusion.

[7] F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, N. Pereira, Controlling Omni-Directional Wheels of a Robocup MSL Autonomous Mobile Robot, ROBOTICA’2004, Porto, Portugal, April 2004.

[8] A. Merke, S. Welker and M. Riedmiller, "Line Based Robot Localization under Natural Light Conditions", in European Conference on Arificial IntelligenceMachine Learning (ECAI) 2004, Workshop on Agents in real-time and dynamic environments.