Journal of Mechatronics, Automation and Identification Technology
Vol. 1, No.3. pp. 5 - 9

Localization of Mobile Robots Using Machine Vision and QR Codes

Svetislav ĆIRIĆ, Nemanja JANKOVIĆ, Nenad JOVIČIĆ

Department of Electronics, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia
Department of Electronics, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia
Department of Electronics, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia
Institute for Public Health of Vojvodina, University of Novi Sad, Futoška 12, 21000 Novi Sad, Serbia
ciricsvetislav@gmail.com

***Abstract*— Due to the increase in applications of mobile robots there is growing need to support their autonomy with good localization. As the camera takes its place as one of the most important sensors in robotics a great number of algorithms for image processing is being developed every day. This paper focuses on detecting multiple robots on the field with three cameras set in the pre-defined places. The detection and identification is done by using QR codes via OpenCV functions library. The designed localization provides a robust algorithm for detecting and following mobile robots which enhances their autonomy with the ability to dynamically decide on the strategy of movement and paves the way for future applications of the combination of QR codes and machine vision.**

***Keywords*— localization, robots, machine vision, QR codes, OpenCV**

I. INTRODUCTION

Today, robotic systems have many applications in industry, medicine, military and other fields, and are finding more and more applications in everyday life. With the development of technology there is a growing need for autonomy of robotic systems, especially mobile robots. Mobile robots are capable of moving in their environment and are not fixed to one physical location. These robots find many applications in commerce and industry. They are used by hospitals and warehouses for transferring materials and goods. They are being developed on a great number of universities in the world and almost everyone has one or more laboratories doing research in this field. Autonomy of movement is provided by the knowledge of the environment itself and correct localization. Localization of mobile robots can be carried out in various ways, for example using encoders, machine vision, lasers and GPS [1].

In this paper localization of a mobile robot is realized using machine vision and QR codes. Primary idea is to detect a predefined shape in the image and to calculate the distance of the object from a camera based on that, which will result in the position of the object on the field. As the use of QR codes is increasing with each day, many algorithms for their detection have been developed. Typical QR code detection algorithms need the QR code to take at least 30% of the image. Because of that, it is required to use an algorithm that can recognize a QR code on greater distances from a camera. As it is possible to use more than one camera for detection, an algorithm is needed to decide the position of the robot in case of detection by multiple cameras.

In the second section of this paper the localization setup and the hardware that was used will be presented. After that, an overview of the algorithms used and developed will be given in section three. The section four will provide information about the QR codes that were used and give an insight to the software implementation. The way localization was tested and results are presented in the fifth section. Localization described in this paper is developed for *Eurobot* competition, an international competition in robotics.

II. SETUP AND USED HARDWARE

The field on which the robots are moving is 3m long and 2m wide. The outline of the field and setup of the system is schematically presented in Fig. 1.

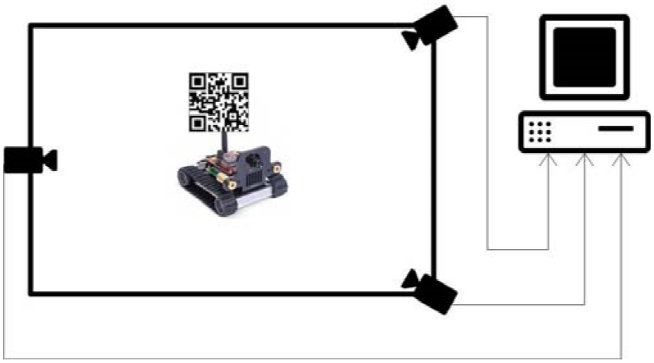

Fig. 1 - Sensor positions along the edges of the field

Three cameras are placed on the edges of the field that will detect QR codes that are placed on the beacon on top of the robots. Images acquired are transferred to a computer that processes them and decides on the position. For this localization three Logitech HD Webcams C270

have been used. The specifications of these cameras that are the most important for this purpose are [2]:

- Focal length: 4mm
- Resolution: 1280 x 720 pixels
- Field of view (FOV): 60º
- Pixel size: 2.8µm

As FOV is 60º, the cameras need to be directed so that they cover the whole field as shown on the Fig. 2.

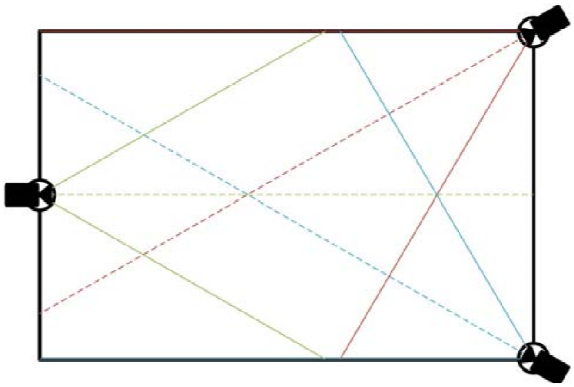

Fig. 2 Direction of the cameras

III. Overview of the Used Algorithms

A. *Algorithm for QR code detection*

QR code (Quick Response Code) is a type of twodimensional, matrix barcode [3]. Barcode is an optical label that contains information about the object on which it is put and which can be read by a machine. QR code contains four standardized coding modes for storing data. The amount of data that can be stored depends on mode which is used.

For detecting the position of a QR code on the image a special insignia is being used shown on the Fig. 3 called FIP (Finder Pattern). There are three FIPs on the QR code located in the corners of the QR code.

Measurements were performed using in house measurement setup, shown if Fig. 3a it consists of a sterile test tube filled with testing material, test tube holder and Hewlett and Packard 4194A Impedance/gain-phase analyzer connected to PC. As can be seen from Fig. 3b sensor is placed inside of solution in test tube and connected to Impedance analyzer.

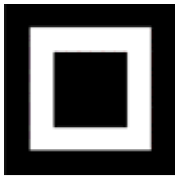

Fig. 3 Finder pattern (FIP)

Algorithm used for detection of the QR codes is based on the face detection algorithm using *Haar* characteristics cascade classifiers developed by Paul Viola and Michel Jones [4]. This algorithm represents the approach of training the cascade function using machine learning with a large amount of positive and negative images. Cascade function is later being used for detection in other images. Training the *Haar* cascade classifier was done according to the paper [5].

B. *Algorithm for detecting position*

Algorithm for detecting position is based on calculating the distance of the detected object from the lens of the camera and transforming the acquired data to the unique coordinate system of the field. Coordinate system of each camera is placed so that the camera lens represents the point (0, 0) of the coordinate system, x axis is in the direction of the camera and y axis is directed right from the camera as shown on the Fig. 4. The coordinate system of each camera and their relation to the field coordinate system is shown of Fig. 5.

Fig. 4 Camera coordinate system

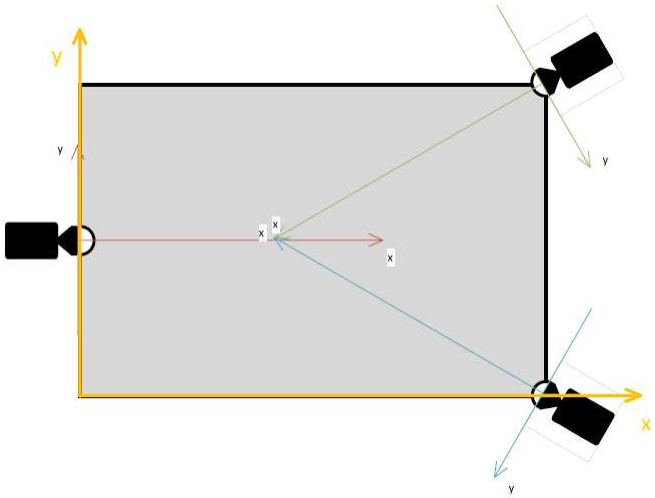

Fig. 5 Coordinate system orientations

The equation used to determine the distance of the object from the lens is derived from equation of the biconvex lens shown on Fig. 6:

$$d_0 = \frac{f h_0 h_{sp}}{h_{ip} h_s}$$

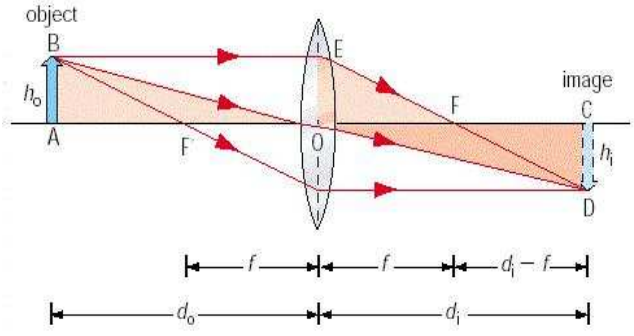

Fig. 6 Biconvex lens

where d_o is distance of the object from the lens, f is the focal length, h_o is the height of the object, h_{sp} is the height of the sensor in pixels, h_{ip} is the height of the object in pixels and h_s height of sensor. By doing this the x coordinate of the object is determined. Similarly, we can calculate the y coordinate. In this case we know the distance of the object from the lens and the unknown variable is the y coordinate itself and the height of object in pixels is calculated as the distance of the middle of the object from the center of the image. Now, it is necessary to transform the coordinate from the cameras' coordinate systems to the coordinate systems of the field. The first camera's coordinate system can be transformed into the field coordinate system with the following transformation equations:

$$x = x'$$

$$y = y' + a$$

where the coefficient is equal to the half of the fields width. For the other to this is not the case. The relation between coordinate systems can be represented by following equations [7]:

$$x = ax' + by' + c$$

$$y = bx' - ay' + d$$

Starting with these equations it is easy to determine the unknown coefficients for each of the three cameras by knowing two pairs of coordinates. The resulting coefficients are:

$$a_2 = -\frac{\sqrt{3}}{2}; b_2 = -\frac{1}{2}; c_2 = 3000mm; d = 0$$

$$a_3 = -\frac{\sqrt{3}}{2}; b_3 = -\frac{1}{2}; c_3 = 3000mm; d = 2000mm;$$

C. Decision algorithm

As the possibility of false detection exists and with it the multiple detection of the object on one or more cameras it is needed to develop a decision algorithm. Decision algorithm must either make a decision on the position of the object or return information that the object was not found or that there was an error in detection. The algorithm steps are following:

- It is first needed to form decision pairs. Pairs are formed by looking at all combinations of positions from two different cameras. The arithmetic mean of the x and y coordinates is kept.
- When these pairs are formed it is first needed to check if the consensus between three cameras has been reached. That is achieved by going through the decision pairs in order to find the two pairs whose difference in position is less than the pre-defined threshold. New value is being set by taking the mean values of the x and y coordinates of the two pairs.
- If there was no consensus, a pair is being looked for that has the smallest difference in detected positions. The smallest difference in detection must also be less than the pre-defined threshold. If the pair is found that satisfies the conditions its position is being taken as the position of the object.
- If there was no pair that satisfies the conditions in the previous step, the next step is the decision based on the information from one camera. First, the region covered by only one camera are checked. In the case that only one object was detected by a camera and if that object is in the region covered only by that camera, that position is taken as the position of the object. If not, the next step is to find the position that is the closest to the previous known position. If this position satisfies a predefined threshold, that position is taken as the object position. If not, the object was not detected. This step can be accessed only two times in a row as the uncertainty of the robot's position becomes too high to be sure of the detected position.

IV. Realization of the Localization

First thing to do is to determine the look of the QR codes and their placement on the beacon. As it is needed to detect QR code from the distance of 3.61m (field diagonal) the minimal dimensions of the FIPs had to be determined. It turns out that because of the quality of the cameras the minimal height of the FIP is 50mm. This proves to be a problem since there is no way to place 3 of them on the beacon.

As only one FIP is needed for detection of the robot this problem is easily overcome by placing only one. As it is not enough for the implementation of the whole QR code, a different approach had to be taken in order to keep the information about the robot on the beacon. As there can be only four robots on the field at one time it is enough to represent them with only two bits. It can be done with placing the black and white areas on the both sides of the FIP, where the white area represents bit 0 and the black bit 1. Placing the same combination of bits on the both sides of the FIP it is secured that there can be no false detection in case the identification from the other beacon overlaps the beacon on which the FIP is detected, or there the false identification due to bad lighting or a shadow. The identification codes that were used are 10mm wide which leaves 5mm space between the FIP and the identification so that the detection algorithm can run properly. The final look of the beacons is shown in the Fig. 7.

Fig. 7 FIPs with identification

Localization is realized using the *C++* programming language with the *OpenCV* programming library version 2.4.10. For this application the libraries used are imgproc, *highgui* and *objdetect.*. Library *imgproc* was used for the basic functions for manipulating and processing images. Library *highgui* is used to provide an easy interface for display, loading and saving images from disk or a camera and to input commands from a mouse or a keyboard. The *objdetect* library contains functions necessary for object

detection. In this library the face detection algorithm previously explained is implemented that is being used for FIP detection. The function in which the algorithm is implemented is called *detectMultiScale*. The input to this function is trained cascaded classifier, and the output is the structure which contains positions and sizes of the detected objects. [7]

Because the process of detection takes a lot of processing time all images taken from the cameras are cut in half. So that we have an image 1280 pixels wide and 360 pixels high which cuts the detection time in half allowing a higher framerate. The cutting is done by cutting out upper and lower quarter of the image.

The acquisition time of the images must be synchronized as well. To avoid taking images from different point in time, all three images are taken at the same time and then processed one by one.

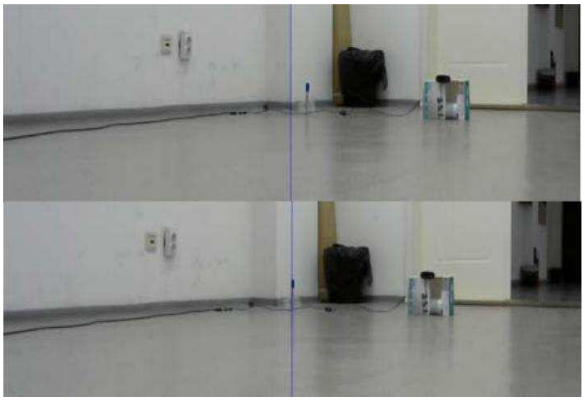

Fig. 8 Calibration of the cameras

As the operating system automatically decides the ID of each of the cameras, a system had to be devised to identify the camera based on its position. Also, the cameras direction has to be set properly to avoid errors. For that purpose, the initialization process was introduced. During this process, cameras are turned on one by one and they are identified by the number of FIPs placed in front of them using the same detection algorithm. After that the screen is introduced with the blue line in the middle of the image. That blue line has to match the calibration marker set on the the edge of the field where the y axis of the camera intersects the edge. The calibration process is shown on the Fig. 8. After this the system has been initialized and the localization can start working.

V. Testing and Results

A. Illustrating results

For illustrating the results two methods were used. First is the image from the cameras. In this case images from all three cameras are merged and displayed as one. On them, the detected FIPs are marked with a circle around them with color depending on the identification. In the upper left corner, the coordinates are displayed in the same color as the circle around the FIP. This method is show on the Fig. 9.

Other method is drawing a path on the field. Path is drawn on the image whose width to height ratio is the same as the fields' by connecting the detected positions with a straight line. Every robot has its own color, like in the previous example. This method is shown on Fig. 10.

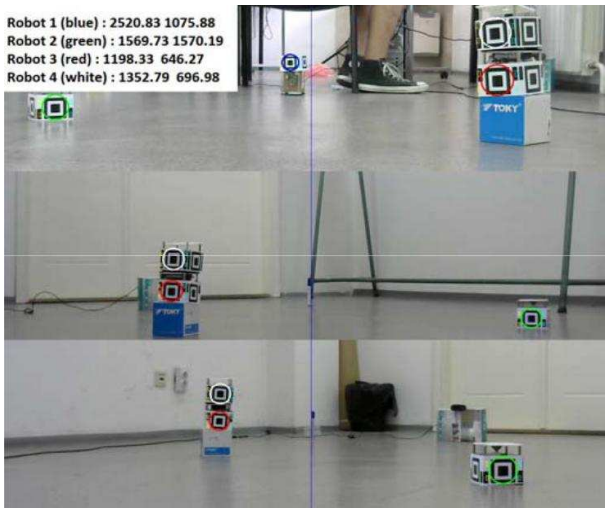

Fig. 9 First method of displaying results

B. Testing

To demonstrate the robustness of the localization, all four beacons are placed on the field at different heights. The result is shown on Fig. 9. This proves useful as the height of the beacon plays no importance to the detection algorithm.

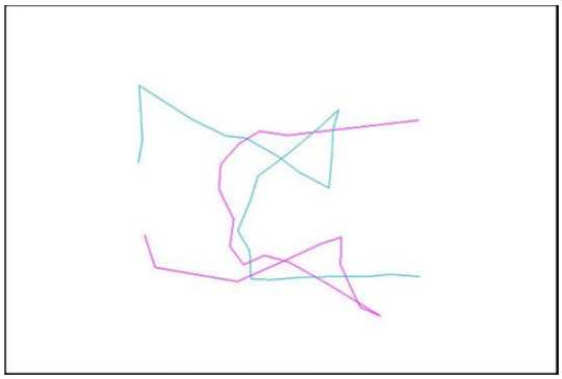

Fig. 10 Paths of two robots

Second test is devised to test the robot moving on the pre-defined path to measure the difference between detected and actual path. The result is shown on Fig. 11.

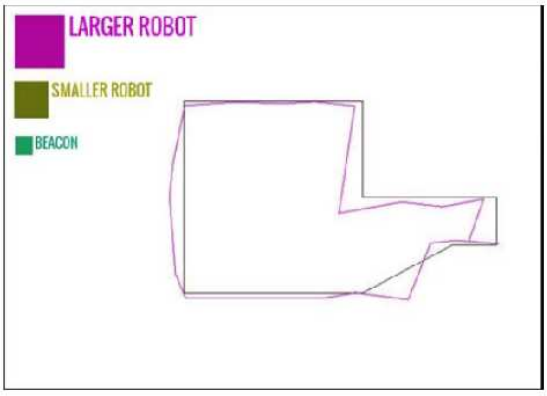

Fig. 11 Results of the drawing the path from detected positions

Next to the path the sizes of the robots have been shown to visually illustrate the error of detection with the robot whose position it is trying to detect.

From the results, the following conclusions can be drawn:

- In the worst case scenario, the localization makes an error of 100mm. This error manifests various factors. First is the field's imperfection. It primarily affects the transformation from one coordinate system to another. The second one is size of the beacon, because the detected position from the cameras on the opposite sides of it will differ 80mm from each other.
- Average time of one iteration is approximately 380ms which is roughly 2.5 frames per second and it varies based on the computer that is being used.
- Based on the number of detected FIPs in one frame that the number of false positive detections is relatively small and does not affect the decision process because of the robust identification process.

VI. Conclusion

In this paper the localization was presented which uses the advantages of the camera as a sensor in robots. Using the *OpenCV* library which provides access to heavily optimized and sophisticated algorithms for image processing satisfying results have been achieved in speed and precision of position detection. Taking into the account that the commercial web cameras were used, this paper demonstrates the perspective of computer vision as one of leading methods in mobile robot localization.

The next step in improving the results is improving the decision algorithm by including machine learning into the process to calculate the threshold which were in this paper acquired empirically. On the other side, intensive development of *OpenCV* library opens the possibility to use *GPGPU* in the detection process and speed up the whole system.

As this localization is primarily developed for *Eurobot* competition there is a possibility to find, through future work, further applications of QR code detection using machine vision.

Acknowledgement

Special thanks to all the members of the Eurobot team Enika for their support.

References

[1] Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011

[2] http://support.logitech.com/en_us/article/17556

[3] https://en.wikipedia.org/wiki/QR_code

[4] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[5] Belussi, Luiz FF, and Nina ST Hirata. "Fast QR code detection in arbitrarily acquired images."Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on. IEEE, 2011.

[6] https://msdn.microsoft.com/en-us/library/jj635757(v=vs.85).asp

[7] http://opencv.org/