Mono-Camera Robotic System for Tracking Moving Objects

Diana Gornea, Dan Popescu, Grigore Stamatescu, Radu Fratila
Department of Automatic Control and Industrial Informatics
University ”Politehnica” of Bucharest
grigore.stamatescu@upb.ro

***Abstract*—The paper presents a mobile platform that, using a single wireless camera, tracks in real-time a predefined object. For simplicity, object recognition is based on shape and color analysis in the HSV spectrum. Platform positioning towards the object is done on two planes: centering the image on the center of the object and approaching the object within a set distance interval. Platform movement is controlled locally, using an Arduino Uno development board, and image processing is done centrally by a general computing system that runs Matlab. Wireless communication is implemented between the two.**

Keywords — real time image processing; object recognition; color segmentation; HSV color space, object tracking.

I. Introduction

The aim of this paper is to implement a mobile robotic platform which, using image analysis, recognizes a spherical object in the scene and positions itself so as to pursue it. Video information can be acquired from a camera mounted directly on the mobile robot, in which case the robot motion induces camera motion, or the camera can be fixed in the workspace, so that the motion can be observed from a stationary viewpoint [1]. Such procedure is usually a time consuming process, because of the large volume of data that a video stream contains.

The object to be tracked should be identified in consecutive video frames, and the association is the more difficult, as the objects are moving fast, relatively to the frame frequency. If the object changes orientation over time, this will also add to the complexity of the problem. To perform video tracking, an algorithm analyzes the sequential image frames and provides output movement (in different representations) between frames of the object tracked. [2]

Typically, before starting the tracking, it is recommended to introduce a pre-processing stage, in order to reduce noise in the image, applying smoothing filters. Then, after the object has been recognized, using various techniques, there is a stage of interpreting its movement, followed by a decision conveyed towards a different system.

A motion control system based on artificial vision consists of three main modules: computer-based image processing module, motricity control module and communication module. There are several possibilities for data processing in an image-based movement control system: a) on a computing equipment external to the mobile part and data communication to the mobile part, such as presented by Taylor et al. [3], b) on a computing embedded equipment, located on the mobile part, as in the example of Cherubini et al. [4] and c) partially mobile and partially on the fixed equipment, ensuring communication between the two parts, as in the present situation. In the latter case, a part of the intelligence of the system is integrated on the movable platform, and another part, which involves high amount of calculation data is carried out by a fixed unit. The wireless communication module is necessary here as well, like in the first case, for the same reasons, but fewer data are transferred. Also, there are fewer centralized calculations. The main disadvantage is the dependency on the fixed block, which makes possible only tracking on short distances. This latter method was chosen for this paper, considering the flexibility offered and the ease of changing the control and image processing programs.

Wang et al. [5] illustrates an algorithm for environment mapping and tracking of moving objects based on images received from a video camera, using a motion model obtained using a Kalman state estimator. Initially, the system starts from a number of known features; then other features of the environment are detected, until the whole space is mapped. Any new object detected in motion is identified by a certain feature and tracked. Implementation is done, unlike the present case, only on a central processing system.

De Luca et al. [6] also presents a tracking system based on a motion model constructed from images, using a robot with 3 degrees of freedom, capable of rotating on the OZ axis, having a mobile camera, with 1 degree of freedom, which can also rotate on the OX axis. In contrast, the robot presented in this article is simpler, being only capable of translational motion and having a fixed camera.

In addition, the only sensor used in this case is a video camera, unlike the mobile robot presented by Segura et al. [7] that is equipped with broadband wireless sensors, using for location distance estimation by calculating the time of arrival for the signals.

As can be seen, in the specialty literature there are similar examples, but the present solution has, compared to those, advantages like: simplicity, flexibility, reduced energy consumption and lower construction costs.

II. The Proposed Architecture

The main functions of the platform are defined:

- to acquire and process images;
- to recognize from images, according to certain criteria, the object to be tracked;

- to locate and track by a specific law the identified object, providing real time control for the motors;
- to communicate the data.

A. Image acquisition

The video stream is acquired using a single video camera, mounted on the mobile platform. Using a mono camera sensor differentiates this solution from other options like stereo vision or using external video cameras. General requirements for the camera are:

- high resolution image for as accurate as possible shape recognition;
- quality optical sensor with high sensitivity for colors;
- wide viewing angle for covering a larger visual field;
- high frequency wireless transmission capability for acquired frames;
- reduced size and weight, in order to be easily mounted on the platform;
- low power consumption, since the power will be supplied by batteries;
- low cost.

B. Image processing

Sorting blobs by color

Regarding the implementation of object recognition by color, we opted for the HSV color space, RGB representation being more affected by shadows and lighting variations (does not recognize same color in different lighting conditions, which is a significant disadvantage). This is the reason why, for color recognition application, the V component is eliminated. Although, at this moment, color recognition is done only by H component, the S and V components are retained as well, for flexibility and future developments.

Identify the object

To begin with, the minimum area less than which the connected regions of pixels with value "1" will be ignored, is set as an empirical percentage of 0.2% of the total area of the image. After setting this area, blobs having an area $A < A_{\min}$ will be removed.

Further on, the noise from the image is removed. The outline is partially restored on restricted portions, eliminating small gaps, using a morphological closing operation, using a disk shaped structuring element with radius of 3.The operation of morphological closing represents the erosion of a dilation of a binary image. The erosion of a binary image *B* with a structuring element (Fig. 1) produces a new binary image having pixels with value "1" in all locations (*i,j*) of the origin of the structuring element in which that structuring element perfectly overlaps the input image *B*, and pixels with value "0"otherwise. Dilation of a binary image *B* with a structuring element produces a new binary image having pixels with value "1" in all locations (*i,j*) of the origin of the structuring element in which that structuring element intersects the input image *B*, having at least one pixel valued "1" overlapping a pixel valued "1" of the image, and pixels with value "0" otherwise.

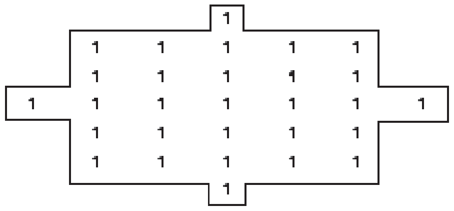

Fig. 1. The structuring element.

Each of the remaining blobs are labeled and their following features are extracted: area, centroid, minimum radius, maximum radius, smallest rectangle containing the region. In the second phase, from the selected blobs, only those with eccentricity:

$$e = \sqrt{1 - \frac{r_{\min}}{R_{\max}}}$$

less than a threshold of 0.2 will be retained, empirically established (for the ideal case, the circle, eccentricity is *e=0*), where $r_{\min} = \min_{1 \le k \le K_0} \{D(g, p_k\}$ is the minimum radius, defined by minimum distance from centroid G to the points on the outer edge p_k with $k \in \overline{1, K_0}$ and $R_{\max} = \max_{1 \le k \le K_0} \{D(g, p_k\}$ is the maximum radius, defined by maximum distance from the centroid of the object to the outer edge.

Finally, the centroid is extracted and the object's radius is computed as (1):

$$R = \sqrt{\frac{Area}{\pi}} \tag{1}$$

C. Locating the object

Before starting to track the object, it is necessary to calibrate the system for measuring the distance to the object, by interpreting its height when it is at a known distance.

The method of locating the object is:

- for locating in the field of view:
  - calculate the i_G coordinate of the centroid of the object;
  - define displacement *d* as the difference between the coordinates of the image's centroid and the coordinates of the object's centroid;
  - because movement is done only on the horizontal axis of the plane (not in height), calculate only the difference in the *x* coordinates, as follows: $x_{\text{image}} - x_{\text{object}} = d$; the result, *d*, will represent the offset in pixels;
  - a negative result would be equivalent to a shift to the right of the object, and a positive result will mean a shift to the left of the object;
- for locating into the depth of field:

- learn the size (height - 2R) of the object at a certain known distance *d* (*F'E* in Fig. 2);

- the triangles ΔABC and $\Delta AB'C'$ from the real scene are similar with the triangles from the image plane ΔAFE , and $\Delta AF'E'$ respectively;

- thus, the new distance *D* at which the object is located can be computed as (2):

$$D = \frac{FE}{F'E} \cdot d \tag{2}$$

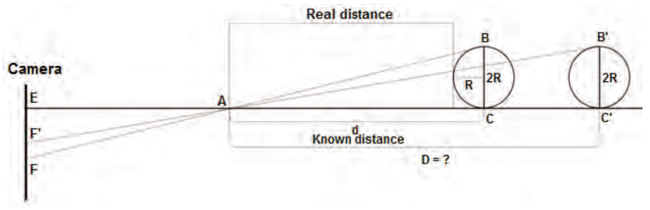

Fig. 2. Geometry used for finding the distance to the object.

D. Tracking the object

Regarding the issue of tracking the object, this lies in finding a way to transform the data received from the image localization module - data that represents an offset, and the height of the object respectively, expressed in pixels - into a control law for the voltage for the motors. There is a need for a system initialization which includes object identification – fixing the set point [8].

Independent control of each motor is achieved:

- for the left-right movement it is required to control the motors differentially on the two sides, i.e. a different control voltage for the two channels;

- in this way, the process model becomes of type MISO;

- thereby, simultaneous adjustment of the direction and speed can be provided–in order to turn left/right the voltage on one motor is increased, and the other motor also changes its revolution in order to compensate for the speed error.

III. Implementation

For the physical implementation of the project architecture, the image acquisition part is performed by an Edimax IC-3115W wireless video camera, the computing part is achieved by a central processing unit, using Matlab, the mobile platform control part is managed by an Arduino Uno development board, and communication between these modules is done via Bluetooth. Fig. 3 shows the functional block diagram of the system.

The mobile platform controlled by Arduino board [9], on which the video camera is mounted, is represented by the A4WD1model of Lynxmotion company, which includes: aluminum frame, chassis, wheels with special tires for good adherence, with 12cm diameter and 6 cm thickness, and 4 DC motors which provide mobility to the platform (Fig 4).

The nominal supply voltage for the motors is 12V DC and is provided through batteries.

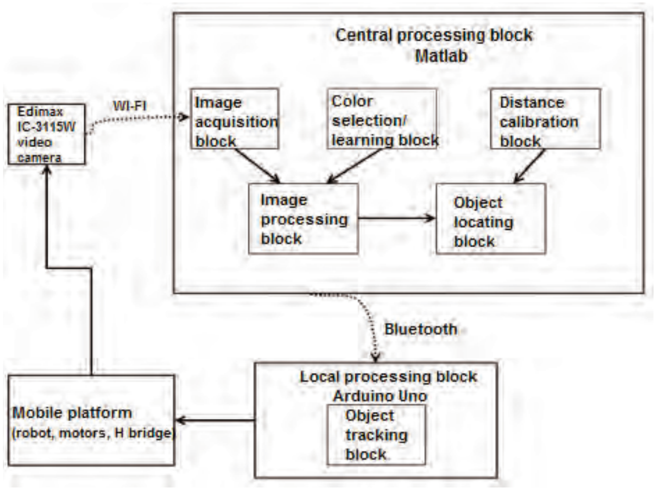

Fig. 3. Block diagram of the complete system architecture.

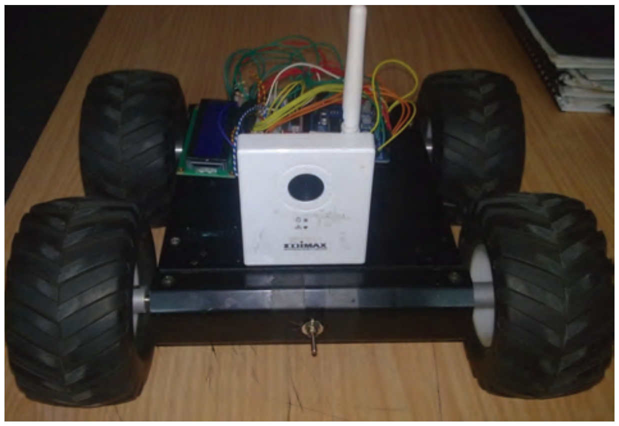

Fig. 4. The robot.

Image acquisition

The video stream acquisition program is implemented in Matlab, using the Image Acquisition toolbox. We mainly relied on the *imaqfind*, *imaqhwinfo*, *videoinput* functions. Video stream loading is performed via the GUI interface, by calling the *start_video* function. Image resolution of *640x480* pixels is used, since it was proven to be the most convenient for our application requirements. Execution flow of the acquisition program is described in the diagram from Fig. 5 and the image processing is described in Fig. 6.

Color learning

The function *process_image*, which performs object recognition from the scene, receives as parameters the input image and the color thresholds (H, S, V components) from the graphical user interface. The input image consists of one frame grabbed from the video stream at a time. The desired color of the object can be set in two ways:

a) choose one of the conventional colors available;
b) learn a particular color*:*

- if option “Learn color” is used, a message window will be displayed, alerting the user to position the object in the center of the image;

- color learning consists of applying a mean-based filter on a neighborhood of 5x5 pixels of the center of the image captured immediately after closing the alert window.

Note that the distance calibration step cannot be done before the color selection step, because height learning implies recognizing the object.

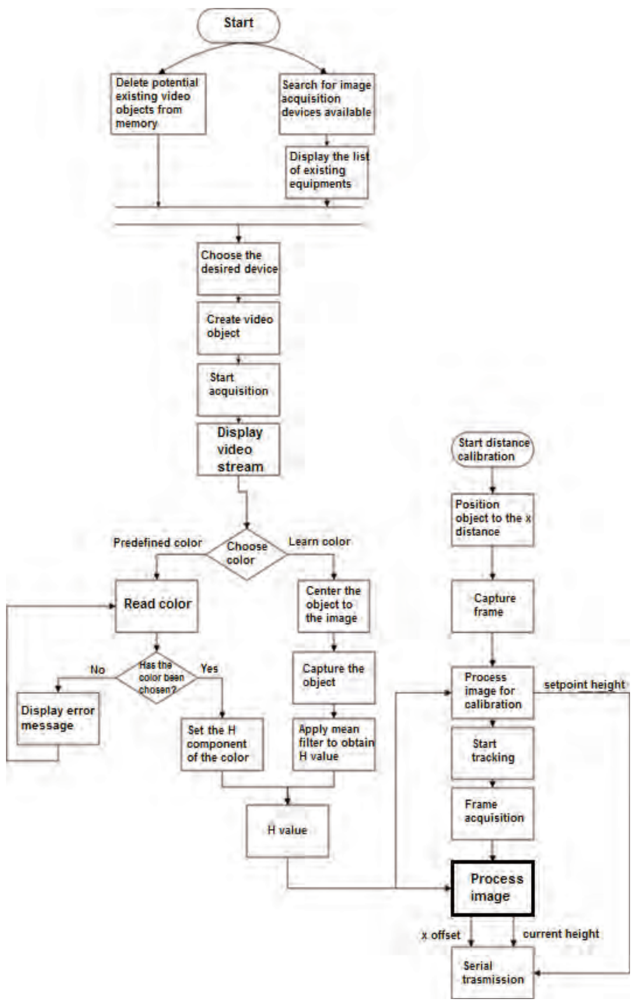
Start
Delete potential existing video objects from memory
Search for image acquisition devices available
Display the list of existing equipments
Choose the desired device
Create video object
Start acquisition
Display video stream
Predefined color
Choose color
Learn color
Read color
Has the color been chosen?
No
Yes
Display error message
Set the H component of the color
Center the object to the image
Capture the object
Apply mean filter to obtain H value
H value
Start distance calibration
Position object to the x distance
Capture frame
Process image for calibration
setpoint height
Start tracking
Frame acquisition
Process image
x offset
current height
Serial trasmission

Fig. 5. Execution flow of Matlab application.

The image processing function will be called for every acquired frame in part and consists of image binarization based on object color, eliminating forms with a smaller area than the one set, noise removal, features extraction and decision whether it is the object to be tracked or not, and finally communicating the needed data: radius and offset.

Image binarization, based on color is achieved decomposing the image into components H, S and V and applying the desired color threshold, as follows:

```
hsv = rgb2hsv(im);
h = hsv(:,:,1);
s = hsv(:,:,2);
v = hsv(:,:,3);
bwh = (h>hval(1))&(h<hval(2));
bws = s>sval;
bwv = v>vval;
bw = bwh.*bws.*bwv;
```

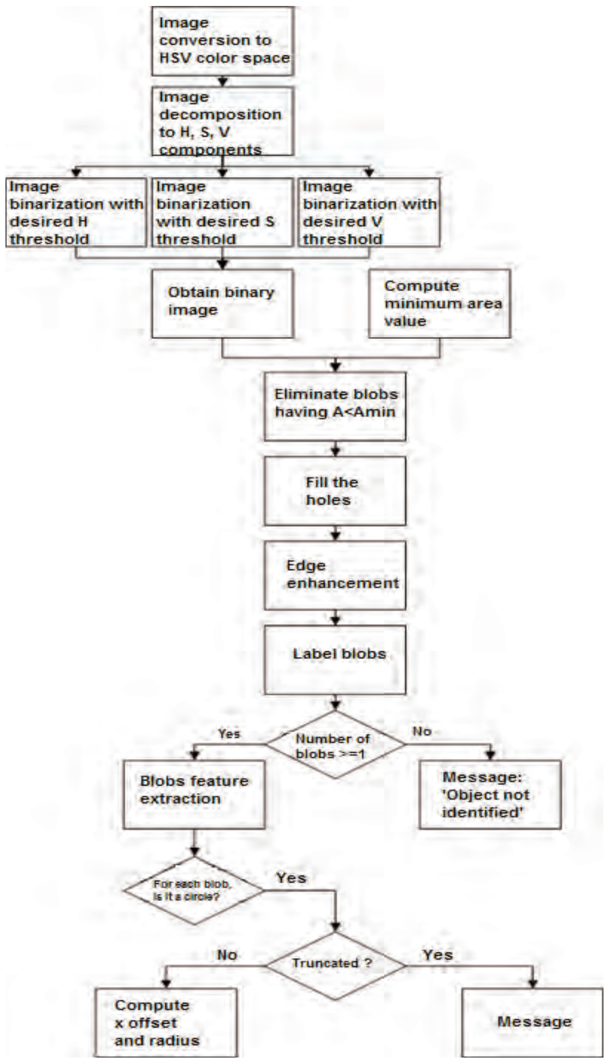
Image conversion to HSV color space
Image decomposition to H, S, V components
Image binarization with desired H threshold
Image binarization with desired S threshold
Image binarization with desired V threshold
Obtain binary image
Compute minimum area value
Eliminate blobs having A<Amin
Fill the holes
Edge enhancement
Label blobs
Number of blobs >=1
Yes
No
Blobs feature extraction
Message: 'Object not identified'
For each blob, is it a circle?
Yes
Truncated ?
No
Yes
Compute x offset and radius
Message

Fig. 6. Execution flow of the image processing program.

Object tracking

The Sabertooth motor driver board can receive from Arduino two PWM input signals with duty cycle values between 51 and 75 to rotate the motors in forward direction with decreasing speed and values between 75 and 102 to rotate them backwards with increasing speed. In other words, setting the duty cycle at 75 is equivalent with stopping the motors, and the lower this value is, the higher the forward speed. If the image processing program does not identify any object in the input image, the parameters offset and distance needed for tracking will be 0, therefore the robot will not move, the duty cycle for the two motors will be 75.

Otherwise, the higher the values of the parameters (offset and/or the ratio of the set point radius and the current radius), the lower the duty cycle values, namely:

- motor control voltage range is divided into 24 speed increments; they are distributed 4 for orientation and 16 for advancement;
- the left-right speed increment is subtracted from the 75 value of the duty cycle related to the PWM signal of the right or left motor, depending on the sign of the *offset* variable;
- the current distance to the object, considering that the implicit calibration distance is 20cm, will be:

$$dist = \frac{SetpointRadius \cdot 20}{CurentRadius}$$

- the proportionality factor for speed increment setting for moving forward is calculated as:
- the minimum area that the tracked object can have, which comes from the minimum area under which objects are not longer identified (of 0.2% of the image resolution) is: $A_{\min} = 0.002 \cdot 640 \cdot 480 = 614.4$ pixels;
- from this minimum area results the minimum current radius that the object can have, as:

$$r_{\min} = \sqrt{\frac{A_{\min}}{\pi}} \cong 14 \text{ pixels;}$$

- the maximum radius that the object can have is half of the smallest dimension of the image resolution, meaning 240 pixels;
- from this possible extremes of the current radius, the maximum distance at which the object can be computed, is:

$$dist_{\min} = \frac{240 \cdot 20}{14} \cong 343 \text{ cm,}$$

  which means that, for an initial distance calibration of 20cm, the object can be recognized from an up to approximately 3,4m distance;
- the maximum distance error that the algorithm will try to compensate is of: $\varepsilon = 343 - 20 = 323$cm, benefiting from maximum 16 speed increments. Thus, the proportionality factor is: $P \cong 0.05$, therefore, the current speed increment for forward movement will be: $tr_d = 0.05 \cdot (dist - 20)$

This speed increment for forward movement is subtracted from the 75 value of each duty cycle.

The localization algorithm for tracking the object is the following:

- calibrate the distance to the object: acquire the frame containing the object properly positioned at desired distance; recognize the object from this frame; compute its height (diameter), from the area information extracted;
- transmit the height the object has at this known distance;
- compute the centroid of the entire image as $\frac{M_2}{2}$, where M_2 represents the number of columns the image has;
- for the identified object, compute the x coordinate of its centroid as (3):

$$i_G = \frac{1}{A} \sum_{i=0}^{M_1-1} \sum_{j=0}^{M_2-1} iB(i,j) \tag{3}$$

  where A the total area of the binary image, M_1 and M_2 represents the number of rows and columns, respectively, of the binary image, and *B(i,j)* is the pixel from the binary image;
- compute the offset in pixels between the object and the image center as:

$$\frac{M_2}{2} - i_G$$

- transmit this offset (signed integer number of pixels) to the motion control module;
- during the current movement, for the object identified in each video frame, compute the new height;
- communicate this new height to the motion control module.

The object tracking algorithm is the following:

- read the number of pixels of the offset between the object and the image's center and the distance to the object, received as inputs from the localization module;
- if the offset is a negative value, then, for orientation, only the right motor will be commanded, and if it is a positive value, then only the left motor will be commanded;
- compute a law that converts the offset to a voltage command (4)

  - a simple control law, of type P, is used as follows: if to a maximum offset (of $\frac{M_2}{2}$, where M_2 represents the number of columns of the image) corresponds the maximum value of control voltage that can be applied to a motor for orientation, $\frac{u_{\max}}{24} tr_0$, then to the current offset corresponds a control voltage value of:

$$u = \frac{offset \frac{u_{\max}}{24} tr_0}{\frac{M_2}{2}} \tag{4}$$

  - tr_0 was chosen empirically, as a result of practical experiments.

  - a large proportionality factor leads to higher amplifications at setpoint shifting, therefore to more abrupt movements of the platform.

- the computed voltage value is interpreted as a value of the speed increment and then as a value of the duty cycle of the PWM control signal;
- compute a law that transforms the setpoint distance to the object to voltage command (5):

  - compute the maximum distance from which the object can be seen, given the fact that the recognition module has set a minimum area less than which the object is no longer identified.

- the maximum error distance is calculated as the difference between the maximum distance at which the object can be seen, and the setpoint distance.

- if this maximum error ε_{max} corresponds to the maximum voltage allocated to advancement, then the current error ε corresponds to a voltage value of: $u = \frac{\varepsilon \cdot \frac{V_{max}}{24} \cdot r_d}{\varepsilon_{max}}$ (5);

- the computed voltage value is interpreted as a value of the speed increment and therefore as a value of the duty cycle of the PWM command signal;
- the computed command for advancement is supplied for both motors, same value (linear displacement);
- if the platform reached the desired distance to the object and the object is centered in the image, the robot stops.

Finally, Fig. 7 presents a screenshot from the Matlab GUI of the application.

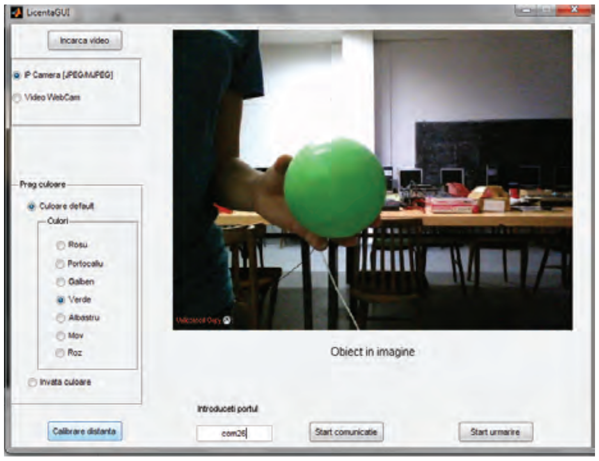

Fig. 9 Matlab GUI application screen shot.

IV. CONCLUSIONS

Throughout this paper the three sides of the problem were followed: recognition, localization and tracking, starting from the theoretical study phase, continuing with the general architecture and analysis, to the implementation phase. We have established an implementation method, an algorithm (taking into account the mathematical relationships identified), but also both hardware, and software implementation of each module, taking into account features and possibilities of technologies and devices used; implementation testing, interpreting the results and drawing several lines of future study. A number of concepts, technologies and devices already existing but from different areas, in one single functional application, succeeding their operation together.

This lead to a sufficiently robust method for recognizing mono-colored objects with spherical shape from an image. We have determined a way to locate an object in a three dimensional space, using information from a single sensor and existing geometric relations.

ACKNOWLEDGMENT

This work was supported by a grant of the Romanian Space Agency, “Multisensory Robotic System for Aerial Monitoring of Critical Infrastructure Systems” (MUROS), project number 71/2013.

REFERENCES

[1] F. Chaumette, S. Hutchinson, “Visual Servo Control. Part I: Basic Approaches”, IEEE Robotics and Automation Magazine vol. 13, pp.82-90, Issue 4, 2006.

[2] S. Kang , J. Paik, A. Koschan, B. Abidi, M.A. Abidi, “Real-time video tracking using PTZ cameras” Proc. of SPIE 6th International Conference on Quality Control by Artificial Vision, pp 103–111, Gatlinburg, TN, May2003.

[3] S.Taylor, K. Farinholt.,E. Flynn, E.Figueiredo, D.Mascarenas, E.Moro, G.Park, M.Todd, C.Farrar, “A mobile-agent-based wireless sensing network for structural monitoring applications”, Measurement Science and Technology, vol 20,14 pp,. 2009.

[4] A.Cherubini,F. Chaumette, G.Oriolo, “A position-based visual servoing scheme for following paths with non holonomic mobile robots”, International Conference on Intelligent Robots and Systems, Nice, France, pp.1648-1654, Sept, 22-26, 2008.

[5] Y.T. Wang, C.H. Sun, M.J. Chiou, “Detection of moving objects in image plane for robot navigation using monocular vision”, EURASIP Journal on Advances in Signal Processing 2012, 2012: 29**.**

[6] A. De Luca, M. Ferri, G. Oriolo, R.P. Giordano, “Visual Servoing with Exploitation of Redundancy: An Experimental Study”, Proc. IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, pp3231-3237, May 19-23, 2008.

[7] M.J.Segura, F.A. Cheein, J.M. Toibero, V. Mut, R.Carelli, “Ultra Wide-Band Localization and SLAM: A Comparative Study for Mobile Robot Navigation”, Sensors, vol 11, pp.2035-2055, Issue 2, 2011.

[8] D. Kragic, H. Christensen, “Survey on Visual Servoing for Manipulation”, Technical report, Computational Vision and Active Perception Laboratory, 2002.

[9] http://www.arduino.cc/