# AN INTELLIGENT VISION SYSTEM ON A MOBILE MANIPULATOR

***M. H. Korayem*, V. Ehtemam, R. Sabzevari, M. Madani and V. Azimirad***

*Robotic Research Laboratory, College of Mechanical Engineering*
*Iran University of Science and Technology*
*P.O. Box 16765-163, Tehran, Iran*
*hkorayem@iust.ac.ir*

*Corresponding Author

**(Received: February 26, 2008 – Accepted in Revised Form: May 9, 2008)**

**Abstract** This article will introduce a robust vision system which was implemented on a mobile manipulator. This robot has to find objects and deliver them to pre specified locations. In the first stage, a method which is named color adjacency method was employed. However, this method needs a large amount of memory and the process is very slow on computers with small memories. Therefore since the previous methods, had used statistical method for object detection, the samples for the connectionist were extracted by this method. The obtained neural network is very robust against light changes and can detect objects very quickly. This neural network has the advantage of being extremely simple to implement, and astonishingly quick in practice.

**Keywords** Vision, Neural Network, Mobile Robot, Error

**چکیده** این مقاله یک سیستم بینایی مقاوم را معرفی می‌کند که روی یک ربات متحرک بازودار نصب شده است. ربات ماموریت یافتن اشیاء و انتقال آنها به جایگاهای مشخص شده را بر عهده دارد. برای بینایی ربات ابتدا روشی به نام "همسایگی رنگ ها" به کار گرفته شده است که نیازمند حافظه کامپیوتری با حجم بالاتری است و چون از روشی آماری برای شناسایی اشیاء استفاده می‌کند، باید از آنها نمونه برداری کامل انجام گیرد. برای عملکرد مقاوم روشی بهتر مبتنی بر شبکه عصبی نیز به کار گرفته شده است که در برابر تغییرات نور بسیار مقاوم بوده و نیز می‌تواند اشیاء را سریعا شناسایی نماید. همچنین قابلیت پیاده سازی آسان و سرعت عملکرد بالا از مزایای روش پیشنهادی است.

## 1. INTRODUCTION

In recent years, many scientists and students have their robots compete in robotic competitions in variety of ways. Micromouse maze contest, robocup robot soccer, and AAAI robot competition are the most famous contests that are held each year [1]. The sweeper mobile robot was designed to detect, pick and deliver three different objects to predetermined places as shown in Figure 1 [2]. To function effectively, the mobile robot must first be capable of purposefully move in its environment. Hence, a MC-310 USB webcam provides robot with the surrounding information. Once the image is acquired and processed, the robot starts picking the objects and taking them into predefined locations.

Recently, there has been a tremendous amount of development in color recognition. Color recognition can be fulfilled in a variety of methods such as Color Profile [4], Spatial-Color Joint Probability [3], Similarity of Color Histograms [6] and Histogram Threshold [5], and state-of-the-art technologies based on Neural Networks [7,8].

In this paper, first the design and manufacturing of sweeper is presented, then we take a glance at previous methods and algorithms presented for processing and analysis of images. Afterward, a method based on color adjacency will be introduced. This method is fast when we have enough memory. To remedy this problem, suggesting a second method which is based on neural network and has the advantage of being extremely easy to implement on a robot in order to have the highest performance.

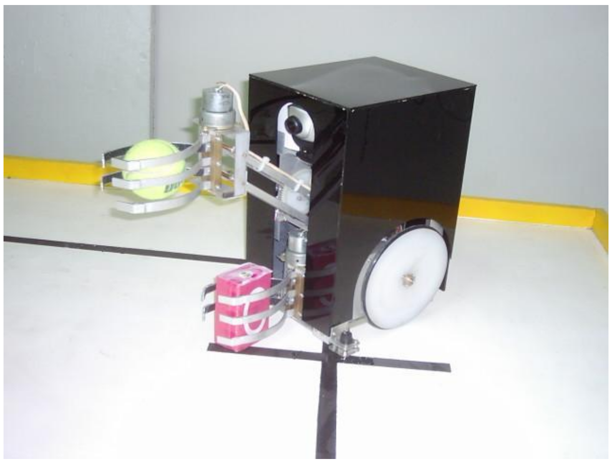


**Figure 1**. The sweeper mobile robot.

## 2. DESIGN AND MANUFACTURING

The differentially driven motors, sensor and vision system and also using two manipulator make sweeper the most powerful intelligent mobile robot in doing the expected tasks, with high mobility, following the line, fast detecting different objects, picking two objects simultaneously and placing them in specialized area as was necessary in robocontest.

Mechanical design is done in the SOLIDWORKS package and we used AVR Micro Controller for processing the data given from sensors and controlling the motors of the wheels and manipulators and following trajectory.

The aim was to design a mobile manipulator with the capacity of pick and place. So the gripper with 5 fingers is designed for grabbing objects. The grippers enhance robot to carry 2 objects simultaneously and to pick the objects up from various heights in its work space easily.

To consume less energy and becoming light, the chassis and manipulators are made of aluminum, two step motors are used to drive the mobile robot, so the robot could move differentially and to increase the torque, a gearbox with a ratio of 4:1 was designed for this unit. A One DC motor drives gripper.

The manipulator of robot is designed in such a way which keeps the gripper horizontally in all position. It helps the robot to carry or to take the objects with higher accuracy.

As mentioned, the sweeper is intelligent programming robot. Five electronic boards are used in this robot as an internal control unit, these are containing: Micro Controller Board, Communication Board, DC Motor Driver, Step Motor Driver and Infra Red Sensor Board.

By tracking a trajectory specified by line, the robot can find a predetermined path and get to the correct location and position to take a photo from area and start detecting objects as the next mission.

## 3. PREVIOUS WORKS

**3.1. Detecting Objects Using Color Histograms** A color Histogram is constructed by first choosing a discrete partition of color space. Each color axis is divided into a number of regular intervals, and then the 3D intersections of these intervals are referred to as color bins. Given an image M, the color histogram $H_M$ is an array $(m_1, m_2,\ldots,m_n)$, where $m_j$ is the number of pixels that fall into the jth color bin. Different methods presented that use color histograms to detect objects, such as color histogram back-projection [9]. This is a method by which a known target object can be located in an image. Given an image of the scene, I, with histogram $H_I$ and an image of the target object, T, with histogram $H_T$ a ratio histogram, R, is constructed.

$$R(i) = \min\left(\frac{H_T(i)}{H_I(i)},1\right) \qquad (1)$$

The ratio histogram values are back-projected onto the image by the value of the color bin into which that pixel falls. The image is smoothed, the peak found, and thereby the target object located [4].

**3.2. Color Profiles** In this type of object detection system, the system learns by example how characteristic of each discrete color is a class of objects using two sets of training images, one set is of images known to contain at least one target object, while the other set is of images known to be free of the target objects. Using the training images a lookup table is constructed, in which each discrete color is assigned a value which approximates the probability that pixels with this discrete color are part of a target object. This table is called the Color Profile. To detect the presence of target objects in a new image the color profile values are back projected onto the image and the image is threshold. A suitable threshold is calculated automatically and the system can recognize if the target objects are not able to be distinguished by the Color Profile method [4].

### 3.3. Lookup-Table Based Methods (LUTs)

This method uses lookup tables indexed by color vector (usually with red, green and blue components), and each cell contains information whether the indexed color belongs to the object or to the background. The segmentation step is preformed by successive inquires of pixel colors indexing the table cells. Our method is similar to this method. Color lookup-tables play a particular role in the domain of color object detection.

McNaughton [10] used lookup-tables for color vision in robots used in RoboCup Contest.

**3.4. Neural Network Approaches** Stoksik, et al designed a color recognition system which was based on artificial neural network, whose generalization capabilities enabled the system to overcome limitations of traditional algorithm or microprocessor based systems [7]. They described a new design using a combination of an artificial neural network to replace the database and an expert system to interpret the color. Their back propagation neural network had three input neurons plus 35 hidden layer neurons and 10 output neurons. The three input neurons are for the three input signals (R, G and B) from a probe. The ten output neurons were each designated red, green, blue, brown, grey, purple, yellow, orange, white and black. More than one output neuron could be activated at any time, so colors such as light greenish-blue would be represented by outputs on the white, green and blue output neurons. They obtained some erroneous results for some lighter and darker shades of the representative colors. To alleviate this problem, they decided to train lighter and darker shades of the representative colors by partially activating the white or black output unit in the training set.

Pomerleau chose a multi-layer perceptron Autonomous Land Vehicle in a Neural Network (ALVINN) with a single hidden layer for mobile robot guidance [12]. The input layer of his network consists of a 30 × 32 unit "retina" which receives images from a sensor. Each 960 units in the input retina is fully connected to the hidden layer of 4 units, which in turn is fully connected to 30-unit-ouput layer. ALVINN learns to guide mobile robots using the back propagation training algorithm. He presented real time training "on-the-fly" techniques to train his network. He applied his network on a CMU Navlab which is a modified Chevy van equipped with forward and backward facing color cameras and scanning laser range finder for sensing the environment.

## 4. COLOR ADJACENCY METHOD

In this method, we define color classes as all colors used in one object with a few different lighting conditions [13]. Color classes values are defined during training process; then we check each pixel of object to determine that this pixel belongs to which of these color classes, by checking color class values and adjacent color values of all colors of a color class. We use a color look-up table (LUT) to store all color classes information. This lookup-table is a 4 dimension array, Color Array structure is [R,G,B,Referred Object], the set of color values a pixel may take on, is RGB = $[0..255]^3$, the set of ordered triplets of integers between 0 and 255 inclusive. The size of $|RGB| = 255^3 = 16777216$ and the number of our objects is 3, we assumed background as one object, consequently size of this array is $255^3 \times 3$, we used this big array in order to improve random access speed. Only few cells of this array are used, because there are few color members in our color classes, but random access to these few colors in this type of data structure is very fast. Each color array's cell contains number of times that one color is happened in referred object.

In training process we manually select a patch of an image that is inside of our desired object. Then for each color of this image patch we increment the value of this color-object cell in color array. This only takes few minutes to train the system for each environment with different lighting condition. Taking image patches of our object is done by simply selecting square patches out of image by mouse, shape of image patch is not important because we don't need exactly all colors that happened in one object; so only one square out of a circular shape object like a ball is enough. Image pixel value distribution in color array for one of our objects is like sample shown in Figure 2.

Each dot in Figure 2 is a color array cell with value other than 0. It means that this color is happened in our background one or more times. In Figure 3, it's shown for Object 1 that in this case was a tennis ball. After applying this routine on

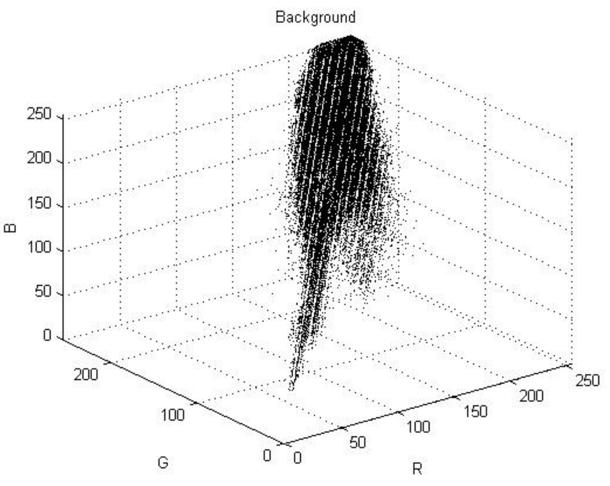


**Figure 2**. Background pixel value distribution in RGB space.

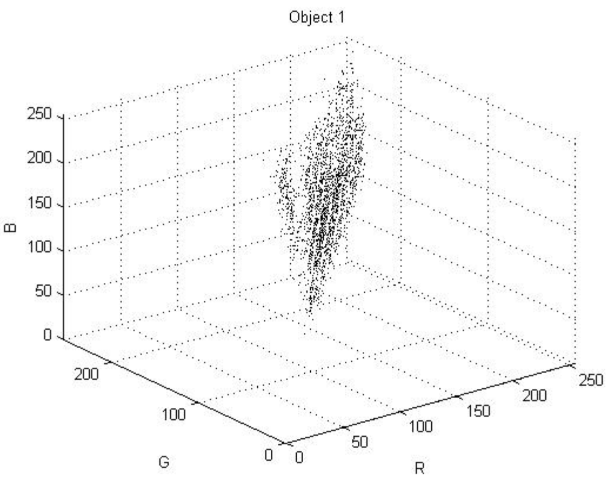


**Figure 3**. Object 1 pixel value distribution in RGB space.

few different images, we gathered all training information we need. So the system was trained.

To detect objects, we examine the color of each pixel of our sample image that we want to detect objects on it by referring to its equivalent color array cell for each object, and then we check which of these 3 cells have greater value inside [13]. For better understanding assume that we want to detect that color A with [R,G,B] attribute belongs to which of our three objects (obj1, obj2 and obj3). C is our color array, and F(R,G,B) that detect object is defined as:

$$F(R, G, B) = \begin{cases} \text{obj1} & \text{if } (C[R, G, B,1] > C[R, G, B,2]) \text{ and } (C[R, G, B,1] > C[R, G, B,3]) \\ \text{obj2} & \text{if } (C[R, G, B,2] > C[R, G, B,1]) \text{ and } (C[R, G, B,2] > C[R, G, B,3]) \\ \text{obj3} & \text{if } (C[R, G, B,3] > C[R, G, B,1]) \text{ and } (C[R, G, B,3] > C[R, G, B,2]) \\ \text{No Obj} & \text{if } C[R, G, B,1] + C[R, G, B,2] + C[R, G, B,3] = 0 \end{cases}$$

We evaluate each pixel by function F(R,G,B), if it returned NoObj, it means this color belongs to none of our three objects, then we check its adjacent colors in this function.

We define 3 types of adjacent colors for color (r,g,b) in RGB Color Space Coordinate system as below:

**4.1. Linear Adjacent** In RGB coordinate system each color is a point like (r,g,b), Linear adjacent (LACs) of one point are:

For i = r – Lr to r + Lr

(i,g,b)

Union

For i = g - Lr to g + Lr

(r,i,b)

```
Union

For i = b – Lr to b + Lr

(r,g,i)
```

In which Lr defines linear adjacency range.

This adjacent is 3 lines in RGB coordinate parallel to R, G and B Axis. That point (r,g,b) is in the middle of each of these lines. It will gives us 3*(2*Lr+1) color, that only one of the components of these colors is different with our primary color.

**4.2. Cubic Adjacent** Cubic adjacent colors (CACs) of (r,g,b) are:

```
For i = R-Cr tp R + Cr

For j = G-Cr to G + Cr

For k = B-Cr to B + Cr

(i,j,k)
```

Cr defines cubic color adjacency range, which is a cube in color space with our primary point in the center of the cube.

**4.3. Luminance Adjacent Color** Luminance adjacent colors (IACs) are:

```
For i = Lrlow to Lrhigh

(r + I, g + I, b + i)
```

Which lrlow and lrhigh are lower band and upper band of luminance adjacent range.

If we checked color of one pixel by function F(R,G,B) and it retuned NoObj, then we check its luminance adjacent by F(R,G,B) if any of these adjacent return a value other than NoObj, we assume that our sample point (r,g,b) is the object returned by function F applied to one of its adjacent, if all of luminance adjacent returned NoObj in evaluating with function F, then we check linear adjacent colors, if no object detected from these colors then we evaluate cubic adjacent colors.

LACs are colors that are different with our primary color (r,g,b) in illumination property of color in HSI color space, so by checking LACs we can find objects with the same color as our sample training images but with different brightness. LACs and CACs are different in hue or saturation property of color. The cubic adjacent set contain linear adjacent set, but we evaluate linear adjacent colors before cubic adjacent colors, so if the object is detected, we will not check cubic adjacent, this way will improve speed in some cases that color properties of objects or light have a little difference with training samples. If all sets of adjacent colors return NoObj, so this color doesn’t belongs to any object.

We do this evaluation for color of each pixel of captured image, and store results of evaluation in an integer array with the same dimension as the captured image. We store 0 for NoObj, 1 for obj1, 2 for obj2, and 3 for obj3. To remove noises from result array, a median filter was applied. Now we know that pixel with (x,y) coordinate in captured image belongs to which object.

## 5. DETECTING POSITION OF OBJECTS

For finding position of objects we scan result array bottom-up line by line, if we found a pixel that belongs to an object, we use classic region growing algorithm to find all other pixels of this object and then we calculate number of pixels of this object and (x,y) of center of this object. Form (x,y) of center of object we can find angle and distance of object to camera, we used simple mapping table to map pixel distances to cm distances.

## 6. COLOR ADJACENCY SEARCH RESULTS

We developed training and testing application in C#, so we could easily select a patch on a picture and train it as one of our objects. We trained about 192401 colors from 1533403 pixels. These pixels are taken out of 12 images with 640x480 dimensions. It takes about 10 minute to train the system. Statistical information about training data for 2 objects and the background is shown in Table 1.

In Table 1, Trained Colors shows number of colors of RGB color space that is evaluated as Object(x) after color adjacency object detection; Trained Pixels are the total number of pixels in image patches that are indicated as Object(x); Correct Trained Pixels shows number of pixels that is trained and evaluated as the same object, it means that color of these pixels happened in Object(x) more than in other objects; Incorrectly Trained Pixels is number of pixels that their color is evaluated for Object(x), but they happened in other objects.

We used a Computer with AMD Athlon XP 2000 + Processor and 512 Mb Ram, MC 310 USB Webcam as capturing device, for our tests. In Table 2 time used for each process is shown:

Images from capturing to detecting objects are shown in Figure 4.

In Table 3 information of detected objects of Image in Figure 5 is shown.

The mobile robot use angle and distance to pick objects. Color Adjacency parameters for these tests shown in Table 4. Values of these parameters are found by experience, since it depends on the color of objects and the illumination of environment.

Figure 6 through Figure 10 show another example of object detection. These figures are gained by using different adjacent type searches.

Statistical Information of detected objects of sample image shown in Figure 6 is shown in Table 5 for detecting objects using different adjacent.

## 7. PREPARING TRAINING DATA

Since a robust, fast, and accurate vision system is needed to be installed on the sweeper mobile robot, a set of training data was created which contained R,G,B related to each object and the number of object. The data is needed for training of the neural network which will be discussed in the following sections. The objects are defined by a two digit binary number in the training data set. Also, the data are set in a random order which makes training the network easier.

## 8. SYSTEM ARCHITECTURE

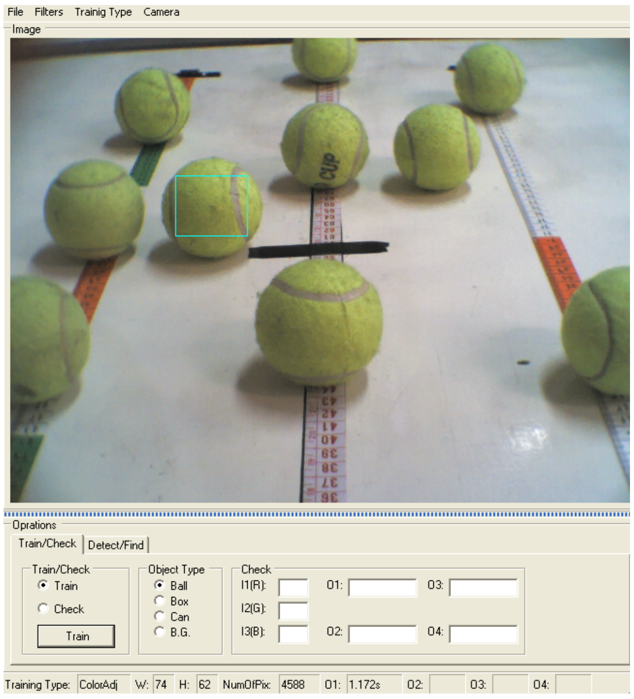
In order to recognize objects, the following system architecture was designed (Figure 11). At first the camera was precisely calibrated and some filters

**TABLE 1. Train Data Statistical Information.**

| Object | Trained Colors | Trained Pixels | Correctly Trained Pixels | Incorrectly Trained Pixels |
|---|---|---|---|---|
| 1 | 32211 | 134031 | 130631 | 3400 |
| 2 | 17081 | 85396 | 82317 | 3079 |
| B.G. | 143109 | 1313976 | 1307698 | 6278 |

**TABLE 2. Time Used for Each Process of Object Detection.**

| Process | Time Used |
|---|---|
| Capturing Image | 820 ms |
| Brightness and Contrast Increase and Noise Reduction | 1,490 ms |
| Color Adjacency Object Detection | 1,171 ms |
| Median Filter | 1,004 ms |



Train/Check: [✓ Train] Check

Object Type: [✓ Ball] Box Can B.G.

Training Type: ColorAdj W: 74 H: 62 NumOfPix: 4588 O1: 1.172s O2: O3: O4:

**Figure 4**. Training/testing application

**TABLE 3. Detected Objects Information.**

| Object Type | X of Cent. of Object | Y of Cent. of Object | Number of Pixels of Object | Angle with Camera Vertical Center Line | Distance to camera on X-Y Plane |
|---|---|---|---|---|---|
| Ball | 39 | 305 | 9573 | -22 | 41 |
| Ball | 320 | 301 | 12575 | 0 | 39 |
| Ball | 600 | 294 | 9218 | 22 | 41 |
| Ball | 204 | 180 | 7389 | -9 | 49 |
| Ball | 88 | 177 | 7429 | -19 | 51 |
| Ball | 436 | 116 | 5670 | 8 | 64 |
| Ball | 321 | 113 | 5316 | 0 | 64 |
| Ball | 147 | 72 | 4302 | -13 | 75 |
| Ball | 492 | 43 | 3722 | 12 | 83 |
| Ball | 322 | 20 | 2225 | 0 | 85 |

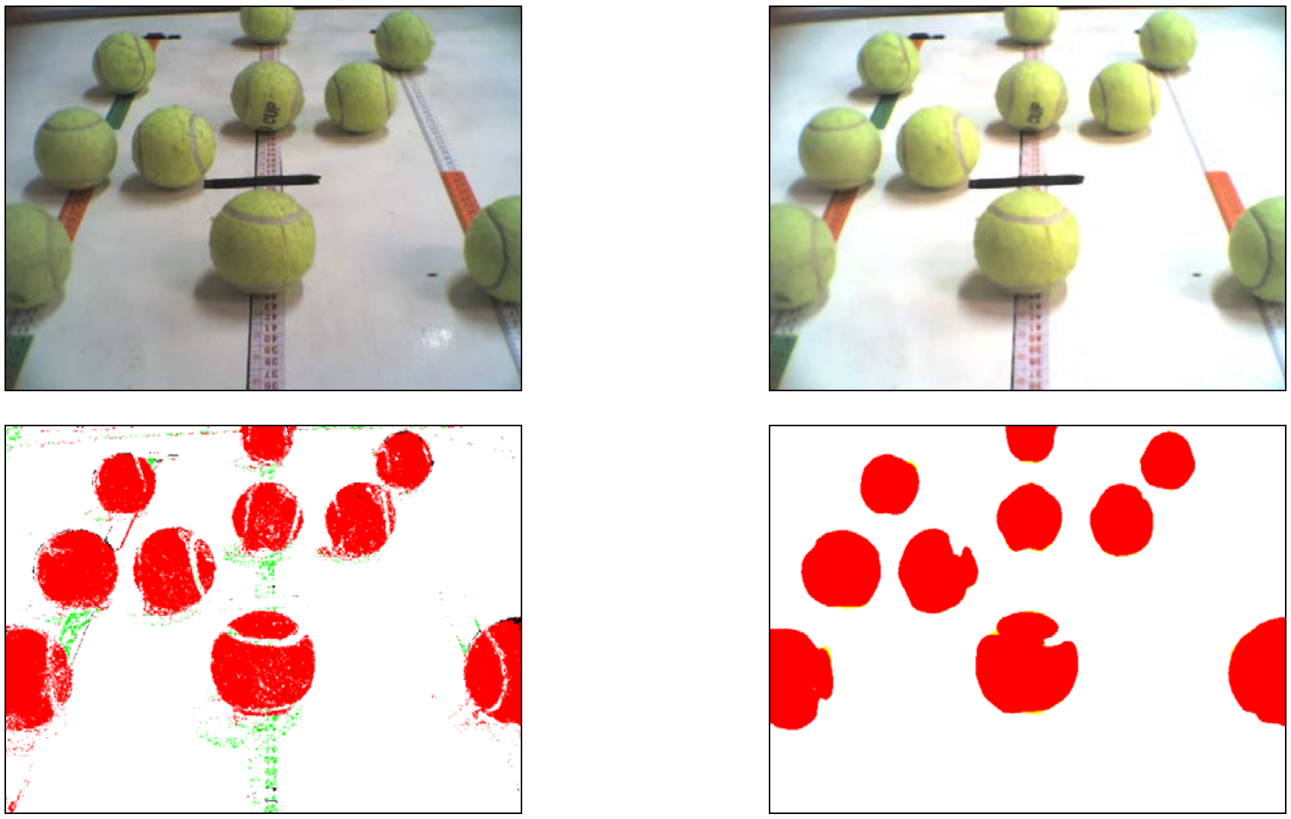


**Figure 5**. Color adjacency object detection steps.

**TABLE 4. Parameters of Color Adjacency Object Detection.**

| | |
|---|---|
| Linear Adjacent Range(Lr) | 10 |
| Luminance Adjacent Range Lower Band (Lrlow) | -20 |
| Luminance Adjacent Range Upper Band (LrHigh) | 20 |
| Cubic Adjacent Range (Cr) | 3 |

were applied to improve lighting conditions. Afterward a still image was acquired and the object detection neural network segmented the parts according to their colors. On the basis of color, a binary image for each object was obtained and the noises were removed by median filter. By exploiting statistical information on solidity and convex area of each object, different objects were detected and their centriod were extracted. Finally, by a mapping neural network the objects coordinates in the real world were extracted.

## 9. OBJECT DETECTION NEURAL NETWORK

A truly autonomous robot must sense its environment

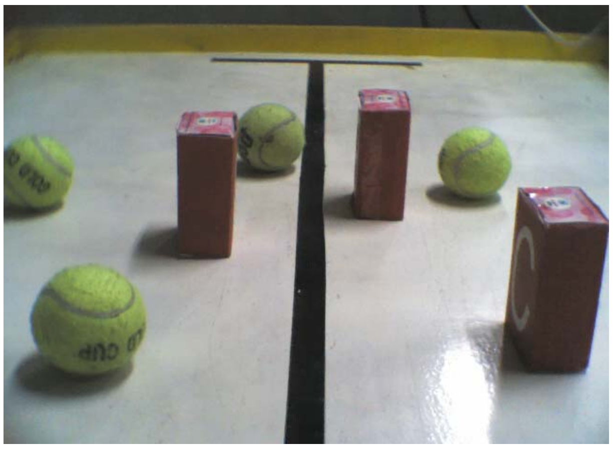


**Figure 6**. Original image.

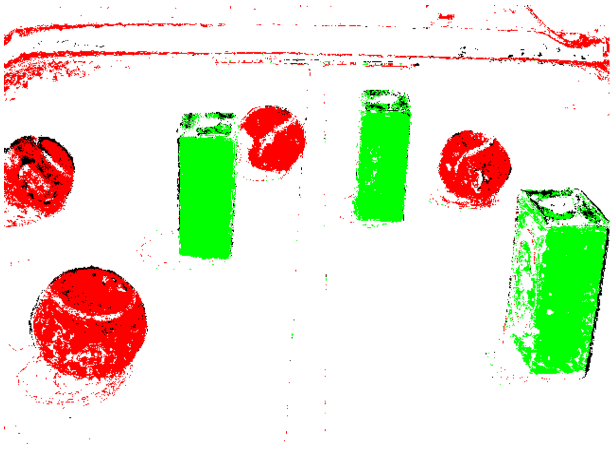


**Figure 8**. Detection result using linear adjacent search only.

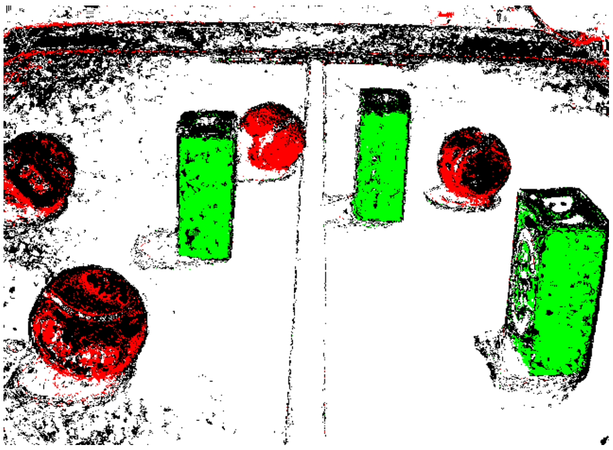


**Figure 7**. Object detection result using lookup table without using color adjacency search methods.

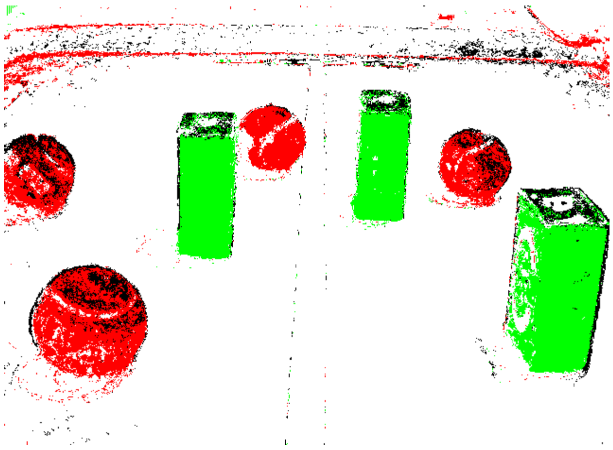


**Figure 9**. Detection result using luminance adjacent search only.

and react appropriately. Previous mobile robot perception systems have relied on hand-coded algorithms for processing sensor information [14]. A neural network-based mobile robot perception system is able to robustly handle a wider variety of situations than hand-programmed systems because of the ability of artificial neural networks to adapt the new situations [15].

Since it is extremely challenging for a mobile robot to be fast and precise and be adaptable to different lighting conditions, it was considered that neural network [16] be applied on the mobile sweeper robot. Training of the neural network was done in MATLAB and operations which give us

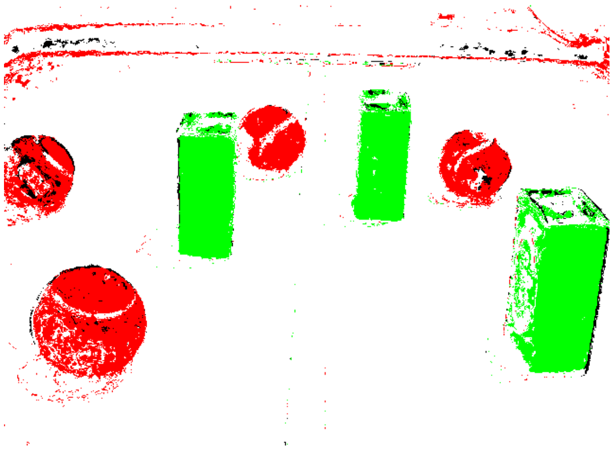


**Figure 10**. Detection result using cubic adjacent search only.

**TABLE 5. Statistical Information of Object Detection Results Using Different Color Adjacent Type Searches.**

| Adjacent type used | Number of Not Detected Pixels | Number of Pixels in Object1 | Number of Pixels in Object2 | Number of Pixels in Background |
|---|---|---|---|---|
| No Adjacent method Used | 59863 | 11201 | 22020 | 213116 |
| Linear Adjacent | 3536 | 25859 | 26172 | 251633 |
| Illumination Adjacent | 12124 | 22643 | 27583 | 244850 |
| Cubic Adjacent | 3211 | 24151 | 27929 | 251909 |

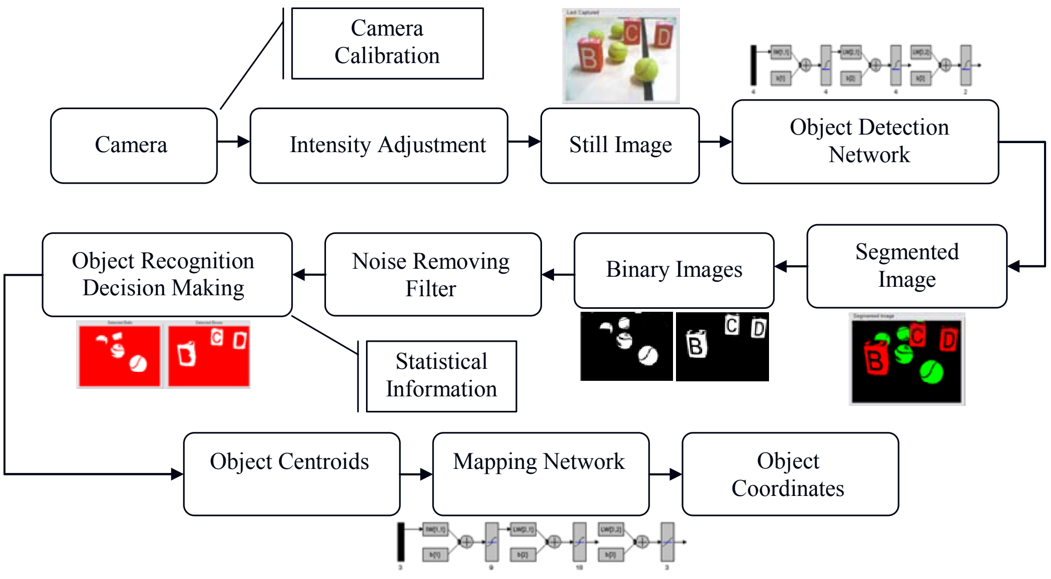


**Figure 11**. System architecture.

the color of objects were done in Delphi. The color features, shape features, and combination of color and shape features can be used for classification [17,18]. However, overlapped objects in 2D image, lighting condition that cause deformations on objects' boundary view, similarity in shapes and expensive computational cost [19], forced us to employ a neural network on the basis of color classification.

Time plays an important role in robocontests, so it is essential to have a neural network which has the minimum number of layers and neurons as it dramatically affects the calculation time. Networks such as perceptron and feed-forward back propagation that are suitable for classification were tested with outputs in the form of two binary digits and four binary digits in order to detect four types of objects presented in the match field [11]. The networks were provided with different image formats with different lighting conditions in order to make the robot vision more stable. The best input format for our network is rgbI. Several reasons support this choice; we will find out that RGB in much more sensitive to lighting condition than HIS.

Besides, adding the intensity feature to RGB, will give a more robust response to lighting conditions. Also, in different lighting conditions RGB ratios will remain constant but in HIS model only hue and saturation remains constant and intensity differs [8].

However, RGBI is quite satisfactory too, and can be further applied to images under various lighting conditions. As a comparison, when we use HIS information a problem may occur to identify two colors with the same hue and different intensities, such as light and dark green samples. In order to compare RGB and rgbI processing, the original (bright) sample images are classified by these two procedures. Then the influence of lighting is studied by processing a (dark) image. Intensity is divided by two, but RGB ratios remain the same in the (bright) image [20].

Since there were nearly 47000 samples (pixels) to be trained to the network and we only had two options for our output (two or four binary digits), the optimum neural network is a feed forward back propagation network that has four layers. As properly trained back propagation networks tend to give reasonable answers when presented with inputs they have never seen [11], a feed forward network has been employed. For pattern recognition problems, if a comparison is made between different kinds of function approximation and pattern recognition problems in MATLAB [11], we will find out that for color segmentation which is a pattern recognition problem a four-layer network will be suitable. The input layer has four neurons and the output layer has two neurons. A sigmoid (logsig) function was applied for output layer in as much as our output is in binary form. The transfer function for all layers of feed forward network is logsig since it compresses input range into a finite output range.

Finally, appropriate training function for pattern recognition problem is trainrp (resilient back propagation) or conjugate gradient algorithms such as trainscg (scaled conjugate gradient) or traincgb. The network architecture is shown in Figure 12.

Training of the network was done with the following parameters (Figure 13).

Later on, the weight and bias matrices were extracted from MATLAB and the results were passed to Delphi. Figure 14 shows the output of the neural network after converting the binary output format to a color image.

## 10. REGIONS SEGMENTATION AND DECISION MAKING

Having determined pixels related to each object, we have three binary images. What needed are images that are segmented with regions according to each object (Figure 15). To this end, sharp noises should be removed; a pixel surrounded by pixels with a different color is not related to the desired object; consequently, these pixels can be assumed as noise. To remove these noises from our images, a median filter with $3 \times 3$ neighborhood was used.

Having extracted regions related to each object, Convex-Hall of each region is calculated. Those regions that have solidity and convex area obtained according to statistical information are accepted as objects. Solidity is the ratio of area to convex-area. Afterward, objects' center of area is extracted. The procedure is shown in Figure 16.

## 11. DETECTING POSITION OF OBJECTS

Through region growing procedure the centers of convex-hulled regions is also calculated, which these kinds of data for the detected objects will be sent to mapping procedure to produce commands for motion control module of the robot [21]. These commands contains the distances of objects from robot and also the angles that robot should turn to reach each object. For example for the image shown in Figure 14, which has the size of $640 \times 480$, the centers of objects are such described in Table 6.

## 12. NEURAL NETWORK FOR MAPPING

Having determined the objects centers in the image, now we have to convert these coordinates into distances and angles that the robot can put them into action. To this end, another neural network was employed. This network which is used for function approximation gets x, y coordinates and type of the object as input and will hand in $(r, \theta)$ which is distance and angle as output [22]. If we make a comparison between different

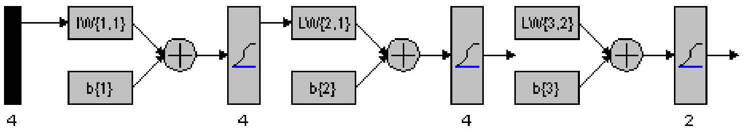
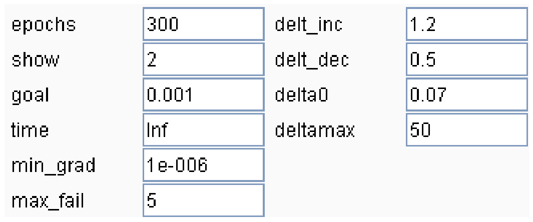


**Figure 12**. The network architecture.



|  |  |  |  |
|---|---|---|---|
| epochs | 300 | delt_inc | 1.2 |
| show | 2 | delt_dec | 0.5 |
| goal | 0.001 | delta0 | 0.07 |
| time | Inf | deltamax | 50 |
| min_grad | 1e-006 |  |  |
| max_fail | 5 |  |  |

**Figure 13**. Network training parameters.

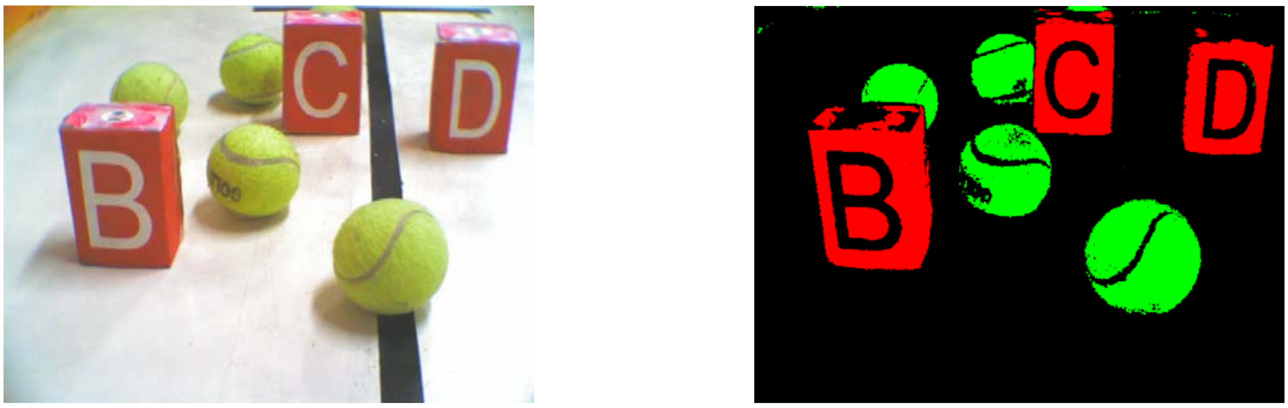


**Figure 14**. Input image/output image.

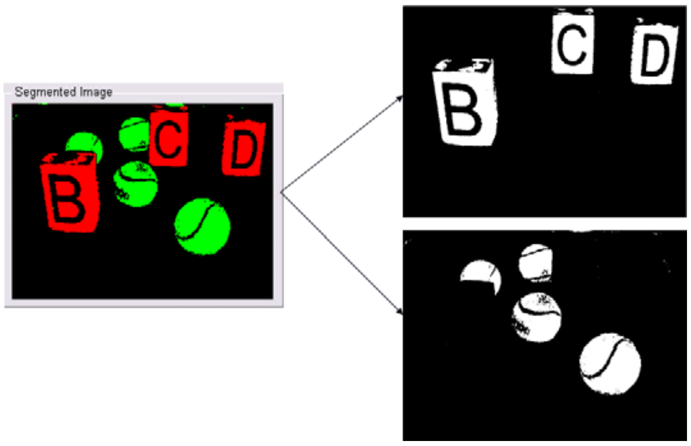


**Figure 15**. Input image/output binary images.

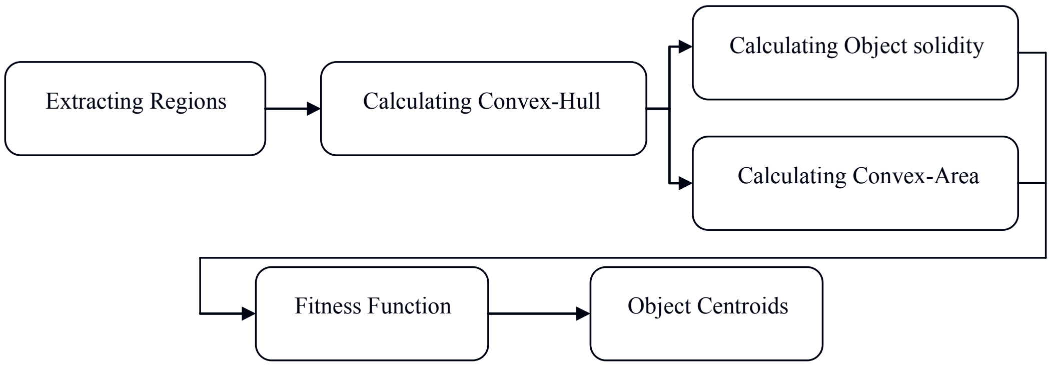


**Figure 16**. Procedure for detecting objects' center of area.

**TABLE 6. Center of Detected Balls and Boxes.**

| Detected Balls | | | | | |
|---|---|---|---|---|---|
| X | 180.8272 | 305.7861 | 301.9716 | 309.2569 | 468.0674 |
| Y | 99.9695 | 216.6263 | 79.5090 | 167.0538 | 303.4099 |

| Detected Boxes | | | |
|---|---|---|---|
| X | 155.4228 | 385.9351 | 565.2396 |
| Y | 220.5851 | 86.8265 | 109.1754 |

problems that have been solved by MATLAB neural network, we will find out that for function approximation, we can use a network with three layers [23]. Again we had to find optimum network. Likewise, networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Eventually, a network with the architecture is an appropriate network as shown in Figure 17.

The result of above network for the picture of Figure 14 is given in Table 7.

## 13. DESIGNED WORKSPACE

In order to put theory into practice an integrated GUI was designed in MATLAB workspace (Figure 18). Capturing image, applying filters, calculating convex-hall and solidity, and sending data to robot are done in MATLAB; however, color segmentation is done with a Delphi program. This interface is modular and each module works independently.

## 14. DISCUSSION

As described above, two points are noticeable. The first thing we should consider is the algorithm we used, to distinguish regions belonged to specified class of object from other regions in image matched with the color pattern belonged to that class of object [24]. This problem may occure

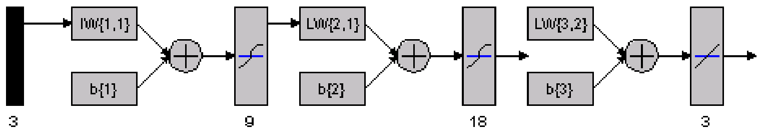


**Figure 17**. Mapping neural network.

**TABLE 7. Position of Objects Relative to Camera.**

| Detected Balls | | | | | |
|---|---|---|---|---|---|
| Distance from Camera | 34.5 | 52 | 55 | 63.5 | 71 |
| Angles from Camera | -1 | -23 | -20 | -40 | -13 |

| Detected Boxes | | | |
|---|---|---|---|
| Distance from Camera | 53 | 68 | 63 |
| Angles from Camera | -43 | -9 | 15 |

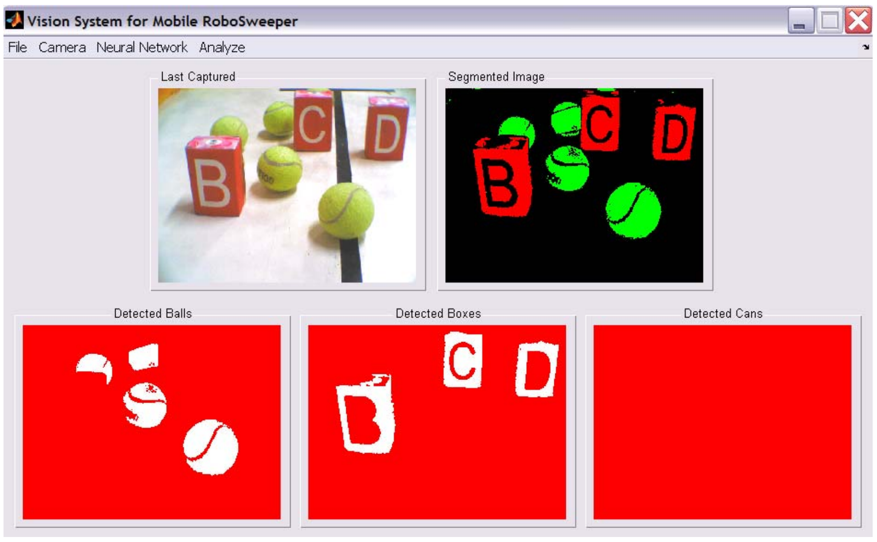

Vision System for Mobile RoboSweeper
File Camera Neural Network Analyze
Last Captured
Segmented Image
Detected Balls
Detected Boxes
Detected Cans

**Figure 18**. GUI designed in MATLAB workspace.

when we have spotted noise caused by shading effects; which results in some regions bigger than could be removed by normal noise reduction filter.

To avoid these problems we consider the convex area of segmented regions to find those big enough to construct our desired objects. Using convex area has another advantage to improve the accuracy of coordinates reported for detected objects. As it is obvious in simulation results, when lines on the surface of tennis balls make some null points on the surface of detected balls and deform the shape of the regions detected as balls, making decision on information from convex area could help us to omit these situations. As expected, when we use area of region instead of convex area the center of the object will not sit on the center of the shape.

Another point laid on providing the train data for network. The procedure starts with generating an array containing rgbI information with another 3 integer expressing the number of times which that specified color is reported as a member of our three different types of objects. In our training data the specified color will be reported as a member of color class which happened more. In other words, we selected colors used in each object as set of rgbI and type of object. In some cases one color is used in two different objects, in this case our decision is based on number of occurrence of that color on those objects. It means that if color c with [r1, g1, b1, I1] is occurred in object1 for 100 times and also it occurred in object2 for 150 times we assume that this color belongs to object 2. Because of these reasons, this method has advantages over other methods, for being precise and being easy to implement.

## 15. CONCLUSION

The results of this method were compared with other methods in order to find the optimum algorithm that should be implemented on the robot for the RoboContest. This method has a great precision for different lighting conditions. In addition, it is an extremely easy way to implement a neural network in MATLAB. Since this method uses a training data based on statistical method, this method gives us robust answers. Also, neural network which was applied here gave us a suitable result in different lighting conditions because it uses r, g, b, I as inputs.

## 16. REFERENCES

1. Braunl, T., "Research Relevance of Mobile Robot competition", ***IEEE Robotic and Automation Magazine***, Vol. 6, (1999), 32-37.
2. Korayem, M., Bani Rostam, H. T. and Nakhai, A., "Design, Modeling and Experimental Analysis of Wheeled Mobile Robots", ***International Journal of AMT***, Vol. 28, No. 3-4, (2006), 403-416.
3. Crandall, D. and Luo, J., "Robust Color Object Detection using Spatial-Color Joint Probability Functions", ***Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)***, Vol. 1, (2004), 379-385.
4. Duffy, N., Crowley, J. and Lacey, G., "Object Detection Using Color", ***Proceedings of International Conference on Pattern Recognition (ICPR'00)***, Vol. 1, (2000), 700-703.
5. Browning, B. and Govindaraju, D., "Fast, Robust Techniques for Colored Object Detection in Variable Lighting Conditions", ***United State Army***, U.S.A., (2003).
6. Vinod, V., Murase, H. and Hashizume, C., "Focused Color Intersection with Efficient Searching for Object Detection and Image Retrieval", ***Proceedings of International Conference on Multimedia Computing and Systems (ICMCS' 96)***, Vol. 17, (1996), 229-233.
7. Stoksik, M., Nguyen, D. T. and Czernkowski, M., "A Neural Net Based Color Recognition System", ***2nd International Conference on Artificial Neural Networks***, (1991), 86-90.
8. Lalanne, T. and Lempereur, C., "Color Recognition with a Camera: A Supervised Algorithm for Classification", ***IEEE Southwest Symposium on Image Analysis and Interpretation***, Vol. 5, (1998), 198-204.
9. Schroter, S., "Automatic Calibration of Lookup-Tables for Color Image Segmentation", ***Proceeding of 3D Image Analysis and Synthesis***, (1997).
10. McNaughton, M. and Zhang, H., "Color Vision for RoboCup with Fast Lookup Tables", ***IEEE International Conference on Robotics, Intelligent Systems and Signal Processing***, Vol. 1, (2003), 399-404.
11. Demuth, H. and Beale, M., "Neural Network Toolbox For Use with MATLAB®", Version 4, The Mathworks Inc, U.S.A., (2000).
12. Pomerleau, D. A., "Neural Network Perception for Mobile Robot Guidance", Kluwer Academic Publishers, New York, U.S.A., (1993).
13. MathWorks Inc, "Image Processing Toolbox for Use with MATLAB®", Version 3, The Mathworks Inc, U.S.A., (2001).
14. Asfahl, C. R., "Robots and Manufacturing Automation", 2nd Edition, John Wiley and Sons, New York, U.S.A., (1992).
15. Bekey, G. A. and Goldberg, K. Y., "Neural Networks In Robotics", Kluwer Academic Publishers, New York, U.S.A., (1993).
16. Horn, B., "Robot Vision", The MIT Press, Massachusetts, U.S.A., (1986).
17. Jähne, B., "Digital Image Processing", 5th Edition, Springer-Verlag, Berlin Heidelberg, Germany, (2005).
18. Gonzalez, R. C. and Woods, R. E., "Digital Image Processing", 2nd Edition, Prentice Hall, Washington

DC, U.S.A., (2002).

19. Mann, S., “Intelligent Image Processing”, John Wiley and Sons Inc, New York, U.S.A., (2002).
20. Seul, M., O'Gorman, L. and Sammon, M. J., “Practical Algorithms for Image Analysis (Description, Examples, and Code)”, Cambridge University Press, Cambridge, U.K., (2000).
21. Amit, Y., “2D Object Detection and Recognition”, The MIT Press, Massachusetts, U.S.A., (2002).
22. Freeman., J. A. and Skapura, D. M., “Neural Networks, Algorithms, Applications, and Programming Techniques”, Addison-Wesley Publishing, New York, U.S.A., (1991).
23. Fausett, L., “Fundamentals of Neural Networks”, Prentice Hall, Washington DC, U.S.A., (1994).
24. Chenavier, F. and Crowley, J., “Position Estimation for a Mobile Robot Using Vision and Odometry”, ***IEEE International Conf. on Robotics and Automation***, Vol. 3, (1992), 2588-2593.