A Fast Vision System for Middle Size Robots in RoboCup

M. Jamzad, B.S.Sadjad, V.S.Mirrokni, M.Kazemi, H.Chitsaz, A. Heydarnoori, M.T.Hajiaghai, and E.Chiniforooshan

Computer Engineering Department
Sharif University of Technology, Tehran, Iran
jamzad@sina.sharif.ac.ir
{sadjad,mirrokni,kazemi,chitsaz,noori,hajiagha,chinif}@ce.sharif.ac.ir
http://sina.sharif.ac.ir/~ceinfo

Abstract. A mobile robot should be able to analyze what it is seeing in real time rate and decide accordingly. Fast and reliable analysis of image data is one of the key points in soccer robot performance. In this paper we suggest a very fast method for object finding which uses the concept of perspective view. In our method, we introduce a set of jump points in perspective on which we search for objects. An object is estimated by a rectangle surrounding it. A vector based calculation is introduced to find the distance and angle of a robot from objects in the field. In addition we present a new color model which takes its components from different color models. The proposed method can detect all objects in each frame and their distance and angle in one scan on the jump points in that frame. This process takes about $\frac{1}{50}$ of a second. Our vision system uses a commercially available frame grabber and is implemented only in software. It has shown a very good performance in RoboCup competitions.

1 Introduction

In a soccer robot, there are several key components such as good mechanical design and its stability, reliable hardware and fast vision system. In this paper we focus on our experiments in developing a fast image processing and vision system for our robots. The basic mechanics and hardware of our robots are fully described in [1]. The hardware of our vision system consists of a Genius grabber which uses BT878 IC. This grabber is mounted on a main board with Pentium, 233 MHz processor. The camera used is a Sony handycam. The software is written in C++ under DJGPP compiler (DJ Gnu Plus Plus) in MS-DOS.

The fast changing environment in RoboCup prevents us from using standard image processing methods for segmentation and edge detection. Although there exist many reliable such methods, but with our processor power, they can not be implemented in real time (25 frames per second). Therefore, we developed a very fast algorithm for region estimation and object detection which can be used for finding mobile targets as well.

2 Proposing a new color model

We used to use the HSI color model for segmentation, but in practice we found some limitation especially in I and H parameters. As a result of a search for a more suitable color model, we concluded that we should combine the parameters of different color models and introduce a new color model. In this way we propose a color model named $\hat{H}SY$. Where the $\hat{H}$ is taken from CIELab, S from HSI and Y from YIQ color models [2]. We have obtained satisfactory results from this new color model. The reason for this selection is that, the component Y in YIQ represents the conversion of a color image into a monochrome image. Therefore, comparing with I in HSI which is simply the average of R,G and B, the component Y is a better mean for measuring the pixel intensity in an image. The component S in HSI is a good measure for color saturation. And finally the parameter $\hat{H}$ in CIELab is defined as follows:

$$\hat{H} = tan^{-1}\frac{b^*}{a^*}$$

Where a^* denotes relative redness-greenness and b^* shows yellowness-blueness [2]. The reason for selecting $\hat{H}$ is that, it has been reported that $\hat{H}$ is a good measure for detecting regions matching a given color [3]. This is exactly the case in RoboCup where we have large regions covered with a single color.

3 Segmentation

Any class of objects in RoboCup environment is supposed to have a predefined color. We have eight such colors according to the rules in practice in year 2001. These colors are green, white, red, yellow, blue, black, light blue, and purple. Therefore, we have to segment the image frames taken by the robot into one of the above colors. In practice we accept one don't care color for objects which can not be assigned to any of the eight standard colors.

In RoboCup, we think it is worth getting fast and almost correct results than slow but exact. Therefore, unlike the traditional image processing routines which define a segment to have a well defined border with its neighboring segments [4], the segments that we define always have rectangular shapes. This shape is a rough estimation for a bounding box around the object that should be segmented. An object is defined to be a connected component with a unique color.

3.1 Segmentation training stage

The output of most commercially available CCD cameras are in RGB color model. In training stage, we put a robot in the field and run programs which convert RGB values of each pixel into $\hat{H}SY$ components as described above. However, to obtain a relatively good segmentation, by try and error, we find the minimum and maximum range values for each parameter $\hat{H}$, S and Y such that the interval in between covers for example almost all tones of red colors seen in the ball. Doing this, we put the ball and robot in different locations on the field and update the interval ranges for red color. The same procedure is done for the remaining 7 colors. At the end of this training stage, 8 (colors) $\times$ $3(\hat{H}, S, Y) =$ 24 intervals are determined. All RGB values in one interval will be assigned to one of the 8 standard colors. However, there might be a few pixels which can not be assigned to any of the 8 colors. These pixels will be given a don't care color code and are considered as noise.

At the end of training stage an LUT (look up table) is constructed which can assign any RGB to a color code 1 to 9 representing one of the eight standard colors and the don't care color.

4 Search in a perspective view

In real world we see everything in perspective, in which objects far away from us are seen small and those closer are seen larger. But, to recognize the objects in real world, we do not examine the scene pixel by pixel and do not check for certain conditions, but, somehow we just recognize the object. It will be a great job if we could define an algorithm for the term "just recognize the object".

The following method is an effort to make the robot detect objects in a quick way which is different from traditional image processing methods.

We use the fact that the robot camera sees everything in perspective. Figure 1 shows an image of RoboCup middle size soccer field with certain points on it. These points that are displayed in perspective with respect to robot front camera are called "Jump points". Actually the jump points are set at equal physical distance. They have equal spacing on each horizontal perspective line. The vertical spacing is relative to the RoboCup soccer field prespective view. To find the actual spacing between jump points on an image, we consider the facts that, the smallest object in RoboCup soccer field is the ball, and the longest distance between a robot and the ball is when they are located in two opposite corners diagonally. If the ball is located in its farthest position and we imagine a square bounding box around it, we want to make sure that robot can detect the ball. Therefore, this square should be large enough to house approporate number of jump points in itself.

The size of this square is very critical, because it determines the maximum distance between adjacent jump points in the perspective view of figure 1. In practice we found it safe to take the distance of each two horizontal jump points on the farthest distance to be about $\frac{1}{3}$ of the side of a bounding box drawn around the ball at that position. In this way, at least 5 jump points (3 horizontal and 2 vertical), will be located on the ball. Therefore, there is a high probability to find the ball, even in the farthest area, by checking only the jump points. In this method, all objects can be found by examining the jump points only, and there is no need to process the image pixel by pixel. In our system we obtained satisfactory results with 1200 jump points.

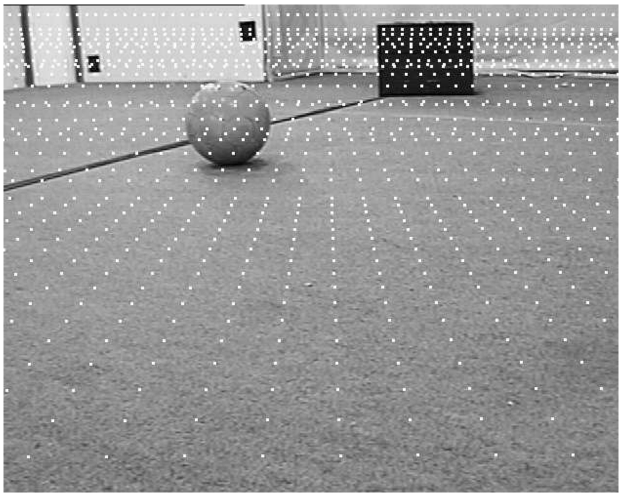

Fig. 1. Position of jump points in a perspective view of robot.

4.1 Search for the ball

We scan the jump points from lower right point towards upper left corner. The reason for this order of search is that we assume a higher priority for finding objects closer to robot.

At each jump point, the RGB values are passed to a routine which searches in a look up table as described in section 3.1, and returns the color code corresponding to this RGB. If this color is red, then we conclude that this jump point can be located on the ball. Since this jump point can be any point on the ball, therefore, from this jump point, we move toward right, left, up and down, checking each pixel for its color. As long as the color of pixel being checked is red, we are within the ball area. This search stops in each direction, when we reach a border or limit point which is a non red color pixel. However, in checking the pixels on the ball, we perform a simple noise reduction method as follows. A non red pixel is determined to be red if at least 6 of its 10 connected neighbor pixels on the same direction are red.

In figure 2-a, point A_1 is the initial jump point, and points T_1,T_2,T_3 and T_4 are the 4 border points on the ball detected by the above algorithm. Rectangle R_1 passes through these 4 border points and is the initial guess for the rectangle surrounding the ball. O_1 is the center of rectangle R_1.

However, rectangle R_1 is considered to be the final answer if the distance of A_1O_1 is less than a certain threshold. This threshold guarantees that point A_1 is close enough to the rectangle center. If the above condition does not satisfy for rectangle R_1, we repeat the procedure of finding another rectangle from point O_1 which is assumed to be a jump point. This procedure is stopped when the above mentioned threshold condition is satisfied. An example of a satisfactory result is shown in figure 2-b, where the distance A_2O_2 is less than the threshold. Rectangle R_2 is the acceptable fit rectangle which surrounds the ball. Figure 3. shows a real illustration of this situation.

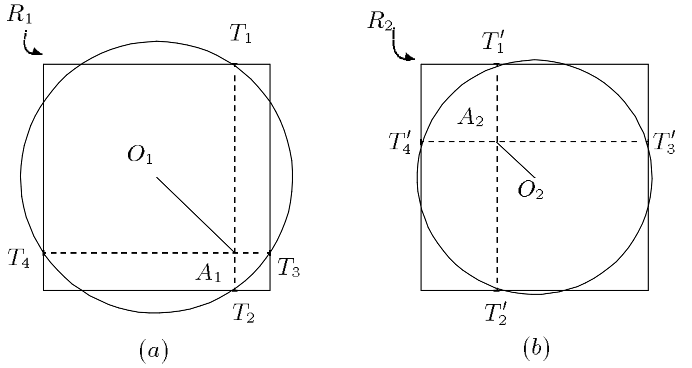

Fig. 2. (a) R_1 is the initial guess rectangle to surround the ball. (b) R_2 is the acceptable fit rectangle surrounding the ball.

Finally, we perform a size filter to eliminate the very small objects, which is, if the rectangle size is less than a certain threshold (this threshold is different for each object) it is considered to be a noise. Then the procedure of examining jump points is continued from the next jump point.

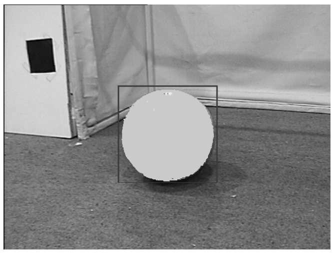

Fig. 3. An illustration of ball segmentation by a surrounding rectangle.

4.2 Search for other objects

In one scan of all jump points, we can find all objects such as robots (our team or opponent team robots), yellow goal, blue goal, etc., in the image.

For each object such as ball, robot, yellow goal, etc. we return its descriptive data such as, its color, size and the coordinate of its lower left and upper right corner of its surrounding rectangle and a point Q on the middle of its lower side.

Point Q is used to find the distance and angle of the robot from that object. However, there is an exception when the ball is too close to our robot. In this case point Q will be on the lower side of the image and is handled differently. The procedure for calculating the distance and angle from an object is given in section 5. Our software can provide this information for an image in about $\frac{1}{50}$ of a second.

5 Finding object distance and angle from robot

In RoboCup, we have fixed position objects such as walls, goals, lines on the field and dynamically changing position objects such as ball and robots. But, as a robot moves around, the distance of fixed and moving objects from that robot continuously changes. To calculate distance and angle of the robot from objects, we need to define a dynamic coordinate system of which origin is located on the middle front side of robot body. Figure 4-a shows the projection of robot body on the field and point O is the origin of the robot dynamic coordinate system (we selected the reversed direction for the x axis).

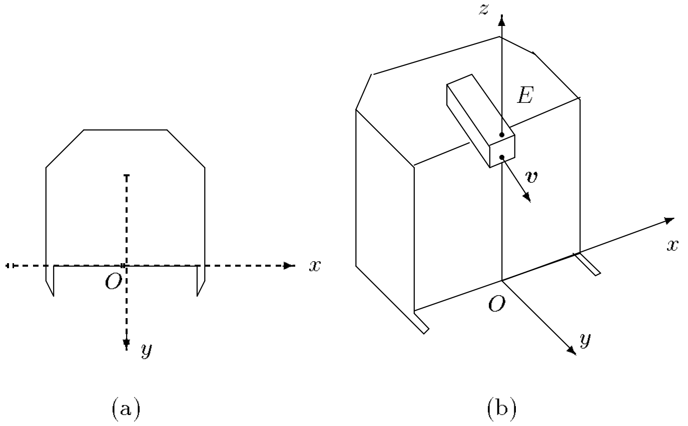

Fig. 4. (a) Projection of a robot body on the field. (b) A 3D view of robot, and the 3D coordinate system.

Figure 4-b shows a 3D view of the CCD camera in a 3D coordinate system on the robot body. For simplicity of calculations, we assume, point E which is the camera gimbal center to be located on the z axis which passes through origin O and is perpendicular to xy plane of figure 4-b.

The idea behind the following mathematical description is that we assume to calculate the distance of robot from an object as the distance from point O to a point Q. If we assume a rectangle surrounding an object such that its lower side touches the soccer field, then point Q is considered to be in the middle of the lower side of this rectangle. By this assumption we do not need to consider the z coordinate value of objects in the 3D world.

Let α be the angle of CCD camera with respect to the xy plane (pan angle: the angle between the line passing through image plane center and lens center with the xy plane or the field). And β be the angle of CCD camera with zy plane (the tilt angle, which is usually zero or has a small amount).

Figure 5 shows the geometry according to which the mathematical relations are constructed. The point Q on the field is projected on point T on the image plane in such a way that the line EQ passes through image plane at point T.

Now we can define the problem as "Finding the length of line OQ which is the distance from an object to our robot". In addition, the angle θ is the relative angle of robot front side with the object represented by point Q.

A unit vector $\hat{v}$ with origin of E and along the camera axis (along the line passing through image plane center and lens center) is calculated as $\hat{v}_x = \cos\alpha \sin\beta$, $\hat{v}_y = \cos\alpha\cos\beta$, and $\hat{v}_z = E_z - \sin\alpha$. Thus,

$$\hat{v} = (\cos\alpha\sin\beta, \cos\alpha\cos\beta, E_z - \sin a)$$

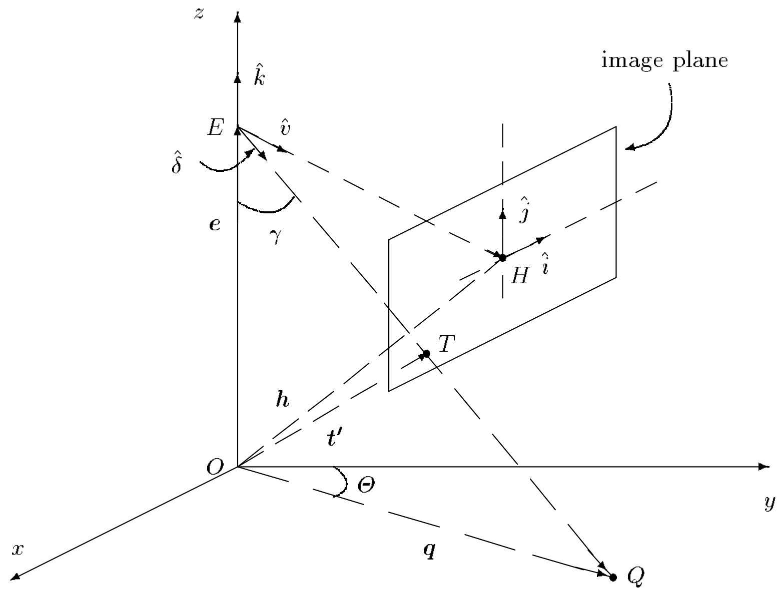

Fig. 5. Image plane in robot coordinate system and its vectors relationships.

We assume the image plane is perpendicular to the camera axis and is formed in a distance f from point E. Let H be the the center of the image plane. So the image plane is perpendicular to vector $\hat{v}$ and f is the distance between E and H. Therefore, we can define the following vector equation.

$$\boldsymbol{h} = \boldsymbol{e} + f\hat{v}$$

Let $\hat{k}$ be the unit vector along the z axis in figure 5. Unit vectors $\hat{\imath}$ and $\hat{\jmath}$ are along the x and y axis of the image plane. $\hat{\imath}$ is perpendicular to $\hat{k}$ and $\hat{v}$, thus unit vector $\hat{\imath}$ is the cross product of unit vector $\hat{k}$ by unit vector $\hat{v}$ and unit vector $\hat{\jmath}$ is the cross product of unit vectors $\hat{v}$ and $\hat{\imath}$.

$$\hat{\imath} = \hat{k} \times \hat{v}$$

$$\hat{\jmath} = \hat{v} \times \hat{\imath}$$

But the coordinate of point T on image plane is given by (T_x, T_y) and vector $\boldsymbol{t'}$ is calculated as follows:

$$\boldsymbol{t'} = \boldsymbol{h} + T_x \hat{\imath} + T_y \hat{\jmath}$$

and

$$\boldsymbol{\delta} = \boldsymbol{t'} - \boldsymbol{e}$$

Where $\boldsymbol{t'}$ is the vector between points O and T as seen in figure 5, and $\boldsymbol{\delta}$ is the ET vector and $\hat{\delta}$ is the unit vector on $\boldsymbol{\delta}$.

We know that

$$\boldsymbol{q} = \boldsymbol{e} + EQ \Rightarrow \boldsymbol{q} = \boldsymbol{e} + (\frac{E_z}{\cos\gamma})\hat{\delta}$$

Where E_z is the z component of vector OE (i.e. the height of camera).

Referring to figure 5, we see that $\cos\gamma$ is the dot product of unit vectors $\hat{k}$ and $\hat{\delta}$, because $\hat{k}$ and $\hat{\delta}$ are unit vectors and γ is the angle between these two lines.

Therefore, the x, y components of vector $\boldsymbol{q}$ represent the (x, y) coordinates of point Q on the field. The angle θ which represents the angle of object with respect to the robot front is defined as

$$\theta = \arctan(\frac{Q_y}{Q_x})$$

and

$$OQ = \sqrt{Q_x^2 + Q_y^2}$$

To find the parameters f, α and β we start from an initial estimates and update them by minimizing the error measured for a set of sample points on the field. This optimization process is done by fine tuning f, α and β around their estimated values.

However, after finding these parameters, we introduce a set of new sample points on field for which the distance OQ is calculated. By comparison of the real distance and the calculated one, we find an interpolation function which are used in practice.

6 Straight line detection

During the matches, there are many cases when the robot needs to find its distance from walls. In addition, the goal keeper in all times need to know its distance from walls and also from white straight lines in front of the goal. If we have one such line, and we know to what wall it belongs to, then we can find the angle of our robot with respect to that wall. However, if two such lines can be detected, the robot can detect its position on the field. This method was used to position the goal keeper in the middle of goal area.

Although there are many edge detection routines which use different filters [4], but all filtering methods are very time consuming and are not appropriate in a real time environment. To solve this speed problem, we propose a fast method for straight line detection. Since in RoboCup soccer field, the border line between walls and field, and also the 5cm wide white lines in front of goal keeper, all are straight lines, in our method, we detect a few points on each straight line and then by using Hough transform [4] we find the equation of the straight line passing through these points.

As it is seen in Figure 6, to locate points on the border of wall and field, we select a few points on top of the image (these points are on the wall), and assume a drop of water is released at each point. If no object is on its way, the water will drop on the field, right on the border with wall. To implement this idea, from a start point w_i we move downward, until reaching a green point f_i.

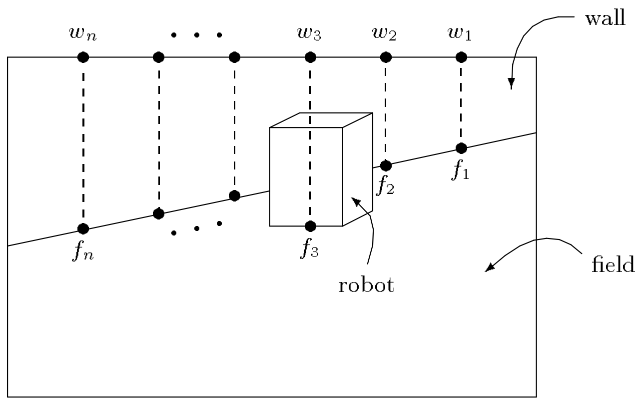

Fig. 6. An illustration of a robot view. Straight lines $w_i f_i$ shows the pass of water dropped from top of wall.

A green point will be accepted as a border point between the wall and field, if it is below a white point (wall), for example points such as f_1 and f_2. By this criteria, point f_3 is considered to be noise. One of the advantages of using Hough transform is that it can tolerate many noise points and also the selected non noise points need not be in close neighborhood of each other.

7 Discussion

There are many aspects in constructing a reliable mobile robot. But even if we have good mechanical and hardware control on our robots, still there are many situations we wonder why the robot can not do things it must do. Here, we would like to bring the issue of sensing devices and the software control capability. The more sophisticated the sensing system and the software, the more time consuming it will be. Therefore, to make a soccer robot act fast enough in a dynamically changing environment we shall reduce the number of sensors as much as possible and design fast software for object finding and good decision making. We believe in long term, when the soccer robot becomes like a humanoid robot to play in the real soccer field, most sensing devices such as infrared, laser, ultrasonic, etc. might not have the same good performance as at present they have in a relatively small field (5×9 meter with 50cm high wall around the field). Therefore, we think it is more appropriate to focus more on vision sensors, and develop methods using images taken from different camera systems. We believe the combination of a CCD camera in front and an omnidirectional viewing system on the top of robot will give a more reliable performance [5]. The omnidirectional camera system can be used for localization and also sensing objects close to or touching the robot.

Although, the fast object finding method proposed in this paper has shown a high performance in previous years in RoboCup competitions, we are currently working on improving the system by using several cameras which enable us to have more reliable localization and sensing methods. Because, when we use only one camera, and detect the straight border lines between walls and field, it is difficult to know to what wall does this line belong. In these situations, an omnidirectional view can provide additional data for correct detection of walls.

8 Acknowledgments

We would like to thank a lot, the mechanical engineering students of Sharif CE team, A.Forough Nassiraei and R.Ghornabi who played important role in design and construction of the robots. Also, we are very grateful to all our sponsors.

References

1. M. Jamzad, et al, *Middle sized Robots: ARVAND*, RoboCup-99: Robot Soccer world Cup II, Springer (2000), pp 74-84 .
2. S.J Sangwine and R.E.N Horne, *The colour image processing handbook*, Chapman and Hall (1998).
3. Y. Gong, and M. Sakauchi, *Detection of regions matching specified chromatic features* , Computer vision and Image Understanding, 61(2), pp163-269, 1995.
4. R.C. Gonzalez, and R.E. Woods, *Digital Image Processing*, Addison-Wesley, 1993.
5. A. Bonarini, P. Aliiverti, M. Lucioni, *An Omnidiretional vision sensor for fast tracking for mobile robot.*, IEEE Instrumentation and Measurement technology Conference, IMTC 99, Piscataway, NJ, 1999, IEEE Computer press.