Novel Approaches to Vision and Motion Control for Robot Soccer

Manu Chhabra[*], Anusheel Nahar, Nishant Agrawal, Tamhant Jain[+], Amitabha Mukerjee, Apurva Mathad and Siddhartha Chaudhuri[++]

[*]Department of Computer Science
Rochester University
Rochester, NY 14627, USA
Email: manuc@cs.rochester.edu

[+](formerly of) Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Kanpur 208016, India

[++]Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Kanpur 208016, India
Email: {amit, apurvams, sidc}@cse.iitk.ac.in

Abstract
In this paper we present innovative solutions to two major challenges faced in the design and construction of a Micro-Robot Soccer team. First, we examine the vision system, which requires very fast processing of images (60 Hz) for recognizing, segmenting and tracking coloured blobs across the playing field. This work introduces a simple neural network-based colour recognition module, followed by robust segmentation based on MacQueen's method. This approach results in reliable detection and tracking even when the robots are fairly close to each other. Second, we look at the strategy and motion control system, in which we use potential fields to compute strategy, low-level decomposition to break up a complex trajectory into a sequence of primitive motions, and PID controllers to drive the robot along this path.

Key Words
Robot Soccer, MIROSOT, Neural Networks, Colour Segmentation, Tracking, Trajectory Decomposition

1. INTRODUCTION

Robot Soccer, since the idea was born in 1995 in the Korean Advanced Institute of Science and Technology, has been an intriguing field of research in Robotics and Artificial Intelligence. The game consists of two teams of robots playing a scaled-down version of soccer (association football). Robot Soccer is a multidisciplinary project, involving research in motor control using a microcontroller, radio communication, image processing and strategy programming. Various Robot Soccer championships are organised under the Federation of International Robot-Soccer Associations (FIRA) and Robocup. Robot Soccer serves as a rigorous testing ground for advanced technologies in autonomous agent control and collaboration between artificial agents. We maintain a team of robots which participates in the MiroSot (*Mi*cro-*Ro*bot *So*ccer *T*ournament) division of the FIRA games. In this paper we describe some of the approaches that we have developed/implemented in order to successfully play a game of Robot Soccer.

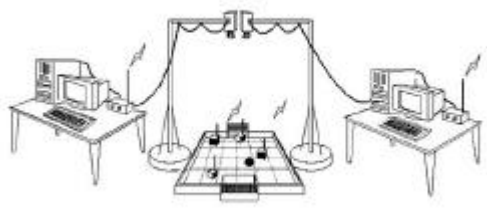

Fig. 1: The Micro-Robot Soccer Setup

2. THE ROBOT SOCCER GAME

A Micro-Robot Soccer match is played with two teams of three robots on a 1.5m x 1.5m field. Each robot (excluding the communication antenna) must fit into a 7.5cm cube. The ball is an orange golf ball. Each team is assigned either the colour blue or the colour yellow: the team colour should occupy atleast 60% of the top surface of the robot. Colours other than orange, yellow and blue may be used for marking orientation detection patterns on the top surface.

The playing field is black, with white markings to indicate the outer boundary, centre spot, half-line, penalty boxes, and certain reference positions. The playing field establishes the frame of reference for computation.

Our setup uses a Canon YC-100 camera mounted 1.5m above the playing field. The analog input of this camera goes to the frame grabber connected to a computer, which digitizes the input and dumps it into a buffer. Software routines then analyse the input data, detect the positions of various objects on the playing field, compute game strategies and send motion commands to the robots. A schematic diagram of the setup is shown in Figure 1. A view of the field as seen by the camera is shown in Figure 2.

3. VISION

To accurately determine the positions of the robots and the ball, we need to analyse the image captured by the overhead camera. Since each object in the image is colour-coded, the first step is to identify the regions of the image with a specific colour. This is followed by position identification and tracking. In this section we present our implementation of these steps, with some prefatory background material.

3.1. Colour

The sensation of light arises when the human retina (or more precisely, the cone cells present in the retina) is stimulated by electromagnetic radiation. These cone cells are responsible for the sampling of different wavelengths present in the visual field. There are three different types of cone cells, each of them sensitive to a different wavelength. Typically these wavelengths are 445, 535 and 570 nm, perceived respectively as red, green and blue. Any colour in the visible range of light (380 - 780 nm) may be described as a weighted combination of the red, green and blue primaries. The sensation induced by light has not only a physiological but also a psychological (subjective) component [1]. After being sampled by the human optical system, the visual information is fed to structures of the nervous system: in essence, a cognitive function acts on the sampled electromagnetic waves. Hence, when a person observes some colour (with any saturation or brightness), this colour is immediately classified as belonging to one of a number of previously known classes, such as purple, red or yellow. However, the exact procedure for classification and the range of each class is not

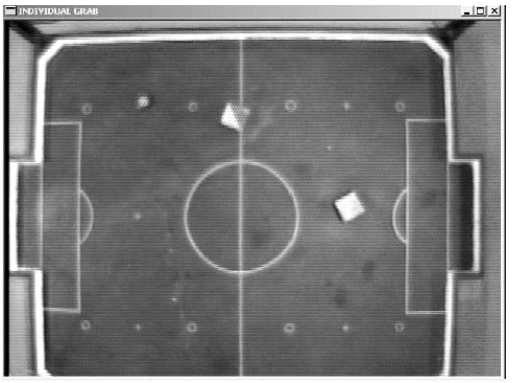

Fig. 2: View of the field captured by the overhead camera

well-established. It has been observed that the psychological characteristics of an individual influence the classification of colours.

3.2. Colour Representation

Colour can be described in two ways: physically or perceptually. Some perceptual measurements are:

1. *Hue*: Distinguishes the dominant wavelength. A person subjectively perceives dominant wavelengths, i.e. colours.
2. *Saturation*: Describes the purity of a colour, on a scale between a monochromatic wave (pure colour) and white light (a mixture of all colours).
3. *Brightness*: Evaluated by establishing a visual intensity equivalence with a shade of grey.
4. *Luminance*: Closely linked to brightness, and measured in units of luminous flux.

Physical measurements typically try to represent colours as a combination of certain primary colours. It has been shown [1] that the stimulus induced in cone cells by incident light of any colour may be reproduced by a combination of light of certain "pure" wavelengths. In order to uniquely specify a given colour, it is possible to add together three primary colors (trichromatic representation) in amounts such that their additive mixture matches the given colour. In choosing the primary colours, we must ensure that they are independent of each other, i.e. some mixture of two primaries should not produce the third. Once the primaries of a system are chosen, it is usual to represent them as an orthogonal basis of a space in which any colour is identified by the triple of values specifying the individual amounts of the three primaries that constitute it. Let *X*, *Y* and *Z* be three primaries of a colour system. The plane having equation

$$X + Y + Z = 1$$

is called the Maxwell plane or chromancy plane, as shown in Figure 3. Spectral colours (pure colours) are represented in the Maxwell plane by points on a curve called the spectrum locus. The shape of the spectrum locus depends on the choice of the primary colour vectors. The spectrum locus encloses all possible combinations of the chosen primaries, i.e. all possible colours without the variation due to luminosity. The complete colour space is 3-dimensional.

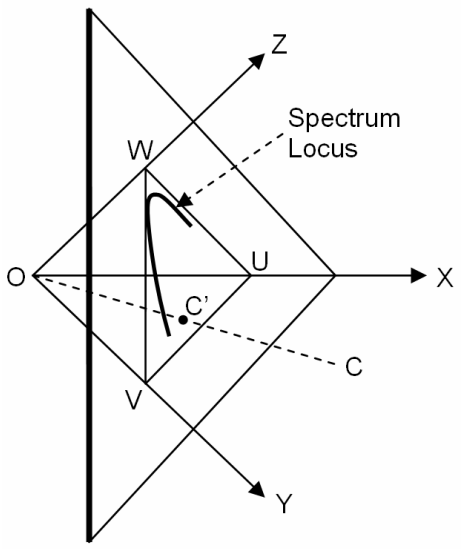

Fig. 3: The Maxwell Plane

3.2.1. The RGB Colour Space

It is natural to choose the retinal primaries as the basis of our colour space. In practice, the actual wavelengths used are 235.8, 546 and 700 nm, perceived as red, green and blue. The primaries are denoted by *R*, *G* and *B* respectively.

3.3. A Review of the Problem

Let us consider Figure 4, which is an image of some sheets of coloured paper captured by a digital camera under a fluorescent light. In this image we find five different classes of colours: blue, yellow, orange, pink and black. Figure 5 shows the location of each pixel in the RGB colour space. This representation does not contain information about the spatial arrangement of the pixels in the original image.

We observe that the different colour groups may be separated by edges in a suitable

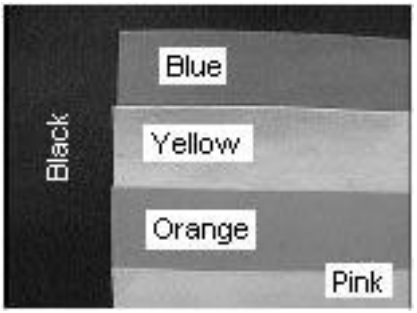

Fig. 4: An image with five distinct colours

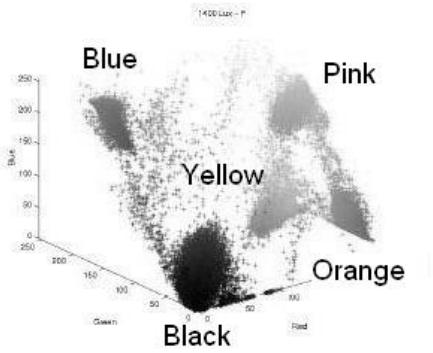

Fig. 5: The five colours of Fig. 4, plotted in RGB space

planar projection of the RGB space. The problem of colour classification can thus be reduced to the problem of finding the optimal edges for a suitable partioning of the projected RGB space. We note that these edges need not be straight, or even regular.

3.4. Towards a Solution

In the past, several techniques have been presented to identify different colours in a scene. A survey can be found in [2]. However, the emphasis has been on accuracy and shape matching rather than speed. In our case, speed is an important criterion since we require real time control. Hence a fast, simple yet robust technique to identify colour is essential. [3] presents an algorithm to index and group different colours available in an image, resulting in distinct clusters, but we cannot use it because it is too slow. A real time algorithm proposed in [4] tracks a hand in an image without supervision. In the field of robot soccer, work on colour identification has been mostly ad hoc. Teams have tried to use an instance-based algorithm in which they approximate a colour space with a cuboid. This leads to obvious problems as (a) colour spaces are not cuboidal and (b) because the intensity of a single colour may vary over several frames and at different positions in the field. The commercially available robot soccer system by Yujin Robotics frequently loses track of the robots and the ball over time when they move from one place to another. This system uses background subtraction and Gaussian filtering to remove noise [5], followed by an instance-based colour identification algorithm to identify patches.

3.5. An Instance-Based Algorithm to Identify Colours

We initially used a simple algorithm to classify colour. The training input was a set of positive examples, each example being a colour vector of the form $<R\ G\ B>$. To train the system to recognise, say, the yellow colour, about 100 pixels in different shades of yellow were provided as input to the program. The mean and the standard deviation were calculated for the set and stored. We classified a test pixel as yellow if its $<R\ G\ B>$ vector lay within the standard deviation distance from the mean in 3D RGB space. This scheme is very fast and fairly accurate if we give examples over multiple frames and create a linked list of means and standard deviation for each frame. However, it frequently produces false positives. Hence we had to abandon this approach.

3.6. A Neural Network-Based Algorithm to Identify Colours

After identifying the limitations of the instance-based algorithm, we decided to implement a back-propagation neural network to identify colour. This choice was influenced by the fact that artificial neural networks have been successfully used in many pattern classification applications where the class boundaries are not clearly specified. Their key features are:

- Ability to approximate almost any function.

- Ability to answer queries (by interpolation) for regions in which the network was not trained.
- Ability to learn from examples.
- Robustness over errors in training.

In our implementation, three input nodes received the *R*, *G* and *B* values of the pixel. The output was a single node, for which a value of 0.9 denoted a positive response and 0.1 denoted a negative response. A friendly and flexible Graphical User Interface (GUI) was created as a front-end for the training module. The GUI can be used to load, save, reset and test a neural network. It also simplifies the process of grabbing multiple images. Training pixels are selected by clicking on the grabbed image. Positive and negative examples were fed to the network using this GUI and the examples were used to train the network over several thousand iterations. Experiment showed that 2000 iterations over positive and negative examples across several frames with 10 hidden nodes gave the best results. Adjusting network parameters and retesting undoubtedly formed the most time-consuming part of the project. A major advantage of using the neural network is that it does not identify too many false positives.

3.6. A Neural Network-Based Algorithm to Identify Colours

The iterative nature of a neural network computation makes it unsuitable for fast real-time processing. To speed up the computation drastically, we used the brute-force method of storing the neural network outputs for a discrete set of inputs (256 values of *R* x 256 values of *G* x 256 values of *B*) as a lookup table. Now we could replace the iterative computation by a simple table lookup. Of course, this approach needed a large amount of memory, and we were forced to invest in an extra memory card for our main computer. The results were worth the expense: the rate of computation, measured in a test run of the robots, increased from about 10 frames processed per second (fps) to 60 fps.

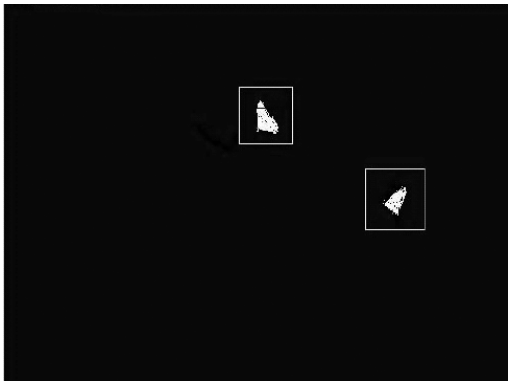

Fig. 6: Two robots recognized and segmented by blue triangular markings on their top surfaces

3.7. Initial Identification

Let us assume the team colour is blue. We scan the entire image to identify all blue pixels by consulting the neural network lookup table. There will be three patches of blue pixels in the scene, one corresponding to each robot. To identify these patches, we apply a clustering method.

3.7.1. MacQueen's K-Means Clustering

The k-means clustering algorithm uses an interchange (switching) method to partition a graph into clusters. An initial partition is given, and new partitions are obtained by switching an object from one cluster to another. MacQueen's method starts by randomly picking *k* points, each corresponding to a cluster to be made. A set of points is taken from the graph, and each point is added to the closest cluster (the "closeness" of a cluster is measured as the distance to its centroid – the mean position of its points). We then iterate over all points, checking if, for each point, the closest cluster (in the updated structure) is no longer the one to which it belongs. If this is the case, we move the point to the closest cluster. When such a switch occurs, the centroids of both modified clusters have to be recalculated. This procedure is repeated until no more switching takes place, after which we choose the next set of points to add and proceed as above.

Although the initial points may not generate an optimal solution, MacQueen's method reduces the sum of squared distances (population variance) within the clusters to a

local optimum. MacQueen's algorithm is guaranteed to converge, hence the algorithm is robust.

Once the blue pixels have been identified, MacQueen's algorithm is used to locate the 3 principal clusters (the 3 optimal means). This establishes the spatial extents of the three robots (Figure 6). The markings in the second colour (we use purple) determine the orientation of each robot, and identify whether it is one of the two strikers or the goalkeeper. A similar procedure locates the opponent's robots.

3.8. Object Tracking

Evidently, scanning the entire frame (640 x 480 pixels) for blue pixels is expensive. We need to exploit the fact that the physical motion of the robots is continuous and not very fast, hence the change in position between two successive frames is small. Given the position of a robot in one frame, its position in the next frame will lie in the immediate neighbourhood if its velocity is not too large. This reduces the size of the search space considerably.

To do this, we must keep track of the locations of the robots and the ball. For each frame, we use the position information from the last few frames and the frame rate to compute the expected location of the object. A window of fixed size (say thrice the size of the robot) centred at this point gives the bounds of our search. If we find that enough pixels of the appropriate colour do not lie in the window, we double the window size and repeat the search.

We obtained tracking speeds of up to 60 frames per second with this setup in an interlaced image.

4. STRATEGY

4.1 Potential Fields

The objective is to design a reactive strategy function which will indicate the appropriate actions of each robot in different configurations. Designing such behaviors involves planning for the individual robot as well the team dynamics, thus the reactive function is defined on a state vector which consists of the position and orientation relative to goal, position and headings of other robots (opponent/self team), ball ownership, etc. Now, based on the current task (defence, pass receiving, striking, etc.), a suitable reactive strategy has to be defined. In our approach, this is done by defining the strategy as a sum of several multi-dimensional fuzzy functions more popularly known as potential fields [6]. In this approach the robot is represented as an entity under the influence of several artificial potential fields whose local variations are expected to reflect the favourability of a particular action concerning that region. The potential functions are defined over entire field: lower potentials move the robot towards a more favourable location (e.g. closer to the ball or closer to the goal), higher potentials repel the robot from unfavourable locations (e.g. away from opponent robots).

4.2 Fields Used

Defensive

- *Goal Field*: favours own goal.
- *Obstruction Field*: favours positions that obstruct the opponents' paths.

Offensive

- *Opponents' Goal Field*: favours opponents' goal.
- *Ball Occlusion Field*: Avoids

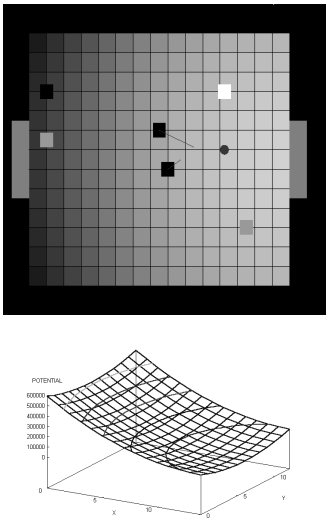

Fig. 7: The goal field. Darker regions are unfavourable, lighter regions favourable.

configurations that will result in the ball being struck close to an opponent robot.

Miscellaneous

- *Energy Conservation Field*: favours nearer positions, to save time taken to move there.
- *Ball Field*: favours positions near the ball.
- *Point Occlusion Field*: avoids positions blocked by opponent robots.
- *Other Player Field*: avoids collisions with other players.

These fields are modelled with exponential or inverse linear functions. For example, the goal field has the function

$$Field_g = k_g * (dist_g)^{n_g}$$

where $dist_g$ is the distance of the robot from its own goal, and k_g and n_g are parameters.

4.3 Superimposing Fields

Depending on the nature of play required (offensive/defensive) the fields are assigned appropriate weights and combined linearly. Figure 8 shows the result of this approach in defense mode.

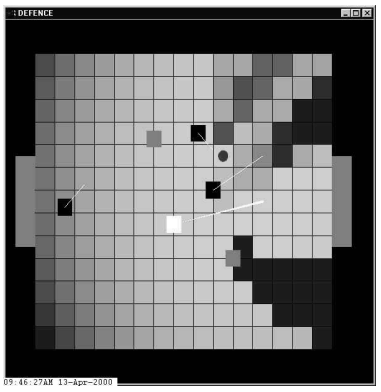

Fig. 8: Defensive field for a sample configuration. The white robot will preferably move back to defend its goal.

4.4 Iterative Improvement

We established strategy groundtruths for different situations by polling a large number of human experts (Delphi approach). The deviations of the potential field strategy from the groundtruths were recorded. These were then used as fitness measures in a genetic algorithm which iteratively improved the parameters used to define the potential fields.

5. MOTION CONTROL

The basic problem is control of a nonholonomic robot. Each of our robots has two traction wheels, which are completely independent of each other (they have individual motors and can be separately programmed). This enables very tight turns (including point rotation) and complicated manoeuvring, but accurate calibration is necessary for straight line motion and path following. PID control is useful in this regard.

5.1 Trajectory Decomposition and Minimal Systems

Due to the limited amount of processing time available to the robot for dynamic strategy planning (a large part of the timeslice is taken up by the vision system), the collision avoidance and motion code has to be kept as minimal and efficient as possible. Simple strategies have traditionally worked better than more complex and less robust ones. We were inspired by the approach of the *Spirit of Bolivia* robot soccer team to "fully exploit minimal systems" [7]. This approach decomposes complex trajectories into very simple primitives. We are in the process of implementing and testing a small library of routines that allow straight line motion, target approach and orbiting motions (clockwise and anticlockwise). These basic actions can be combined to generate more involved sequences.

5.2. Path Following

We use software PID controllers to keep the robot on the designated path. An error value is computed taking into account deviation in both orientation and location. A linear combination of this error value, its integral and its derivative (with respect to time) is used for generating the feedback (a correctional angular velocity). A

straightforward calculation determines the individual wheel velocities, assuming maximum possible linear speed is to be maintained.

5.3 The Goalkeeper

The goalkeeper is a special robot which moves parallel to the centre line in the penalty box region and tries to stop the ball with its side. It is the last line of defense and has to be fairly robust. This robot should position itself at an optimal point in the goalkeeping area so that it covers the position from which the ball is most likely to come. Our current code uses an experimentally determined formula to compensate for overshoots in the position of the robot. We have determined that the amount of overshoot is approximately linearly proportional to the velocity of robot motion, and are trying to refine the method using PID control.

5.4. Simulation.

It is difficult to directly test new methods for motion control, since the robot batteries get drained quickly and long experimental runs are not possible. Hence we designed a software simulator for the micro-robot environment (a screenshot is shown in Figure 9). The simulator implements linear and rotational collision dynamics [8]. It is parametrized and highly configurable – we have tested non-standard environments with more than one ball and arbitrary polygonal fields. We are working to adapt the interface of the simulator library to be exactly the same as that of the actual robot control module, and to generate images of the virtual environment which can be fed to the vision module for analysis. This would enable the physical equipment to be efficiently replaced by their virtual counterparts for testing purposes. Of course, since such simulations can never be entirely accurate, the final testing will have to be on the hardware.

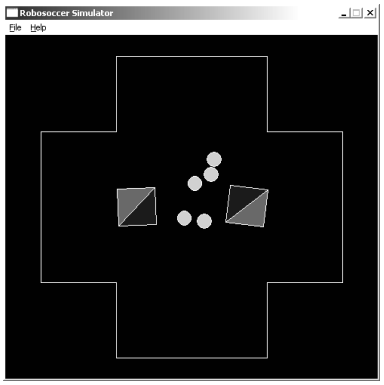

Fig. 9: Robot Soccer Simulator, in a symmetric field with 5 balls

5. REFERENCES

1. **S. V. Novakovsky,** "Colour television: A theory of colour reproduction", Mir Publishers, 1975.

2. **M. J Jones, J. M. Rehg,** "Statistical color models with application to skin detection", Technical report, Cambridge Res. Lab., Compaq Computer Corp, 1998.

3. **M. J. Swain, D. H. Ballard**, "Color Indexing", *Int. J. Computer Vision*, Vol. 7, No. 1, pp. 11-32, 1991.

4. **Ying Wu, Qiong Liu, Thomas S. Huang**, "An Adaptive Self-Organizing Color Segmentation Algorithm with Application to Robust Real-Time Human Hand Localization", Proceedings of Asian Conf. on Computer Vision, 2000.

5. **A. N. Venetsanopoulos and K. N. Plataniotis**, "Color Image Processing and Applications", Springer Verlag, 2000.

6. **Jean-Claude Latombe**, "Robot Motion Planning", Kluwer Academic Publishers.

7. **Birgit Graf**, "Robot Soccer", Diploma Thesis, Fakultät Informatik, Universität Stuttgart, 1999.

8. **David Baraff**, "An Introduction to Physically Based Modeling: Rigid Body Simulation", SIGGRAPH '95.