Learning to Recognize Objects - Toward Automatic Calibration of Color Vision for Sony Robots

Tatjana Zrimec TATJANA@CSE.UNSW.EDU.AU
Centre for Health Informatics & School of Computer Science and Engineering, University of New South Wales, Sydney Australia

Andy Wyatt ANDYWYATT@OPTUSHOME.COM.AU
The University of Sydney, Sydney, Australia

Abstract

Color detection can be seriously affected by lighting conditions and other variations in the environment. The robot vision systems need to be recalibrated as lighting conditions change, otherwise they fail to recognize objects or classify them incorrectly. This paper describes experiments toward object recognition under different lightning conditions. We propose to train the vision system to recognize objects under different lightning conditions using machine learning. Learned knowledge is then used for object recognition. Having attached leaning module to the vision system facilitates the object recognition and provides conditions for automatic adaptation of the vision system to new environment.

1. Introduction

Every year, since 1997, there is a RoboCup competition where different robot leagues compete in soccer games. The aim of the competition is not to create robots to play soccer so much as to encourage research institutions such as universities to do research in cooperation between autonomous agents in dynamic multi agents environment. Playing soccer requires concentrating on a small number of well-defined problems, which results in developing competitive innovations in the areas of computer vision, locomotion, strategy and teamwork.

The work presented in this paper is about Sony legged robot league. Each team consists of four quadruped robots, similar to the entertainment robot AIBO ERS-210. (see Figure 1a). The robots are equipped with different sensors, among which is the on-board CCD camera. The vision system is responsible for analyzing the scene correctly and it is quite important in the success of one of these robots in the competition. The robots use visual information alone to determine information about the surrounding robots, the ball as well as their own position and direction.

The current systems in use have one major problem, which this project attempts to address. The problem is that the robot vision systems need to be recalibrated as lighting conditions change, otherwise they fail to recognize objects or classify them incorrectly. Changes of the lighting conditions during the training period are tolerable since there are no time constraints. The more crucial are the changes of the lighting conditions during the actual competition. Usually there are two play fields and each has different lighting. The robot teams have to be ready to play in a short period of time. So far, the recalibration is performed manually prior the game or even the previous day (evening). That is very risky since the light can change between the period of calibration and the actual game. The aim of this project is to develop an automatic calibration procedure in order to increase the robustness of the robot vision system.

The approach we propose here is to train the vision system to recognize objects under different lightning conditions using machine learning. Learned knowledge is then incorporated in the vision system and is used for object recognition. The learned classifier should be able to recognize objects although their colors due to the lightning condition have changed. In order to generate such a classifier, we have to choose "good" vision parameters (attributes) to describe the objects in the scene.

2. Sony Robot Vision

The robot camera delivers color images with low image resolution. Color images are usually represented in Red Green Blue (RGB) color scheme. An alternative representation is the YUV color scheme. In the Sony Legged Robot vision the YUV color scheme is used as the native image format. The U and V components represent the `chromacity' of the pixel that relates to the actual color. The Y component represents the `luminance' of the pixel which describes how bright, or luminent the pixel is (Hengst et al 2001). The YUV format can be easily converted into RGB format by a matrix multiplication (Hengst et al 2001, Chen et al, 2002).

To make the soccer play easier, the objects on the play field are color coded. The objects of interest are the ball which is orange, the goals that are blue and yellow and the filed carpet which is light green. The filed is equipped with six landmarks in a form of beacons with pink on green, pink on blue and pink on yellow.

The task of the vision system is to enable the robot to find the ball, to detect the beacons in order to localize itself, to detect other robots and avoid the collision, or to find the "correct" goal in order to score. Object recognition is mainly based on finding regions of specific colors. Consequently accurate color classification is critical (Chen et al, 2002).

Color detection can be seriously affected by lighting conditions and other variations in the environment. The lights at the competition venue are usually very bright and often have caused significant mis-classifications. There is a need for a vision system that can be easily adapted to new surroundings. Since colors may appear differently under different lighting conditions, it is necessary to perform some kind of training prior color classification.

The Sony legged robot has hardware on-board to perform the color classification. It is capable of classifying 8 colors simultaneously (Hengst et al 2001). 256 levels of luminance are divided into 32 equal size levels. At each level, user has to provide rectangles coordinates for each color s/he wants to classify. The rectangle that a pixel lies in determines its color. After providing all the color rectangles, upon receiving an image in YUV format, the hardware returns an image in YUVC format where YUV values are unchanged and C is the classified color of the pixels. Because C value is stored in one byte and each bit encodes one color, the system is capable of classifying only 8 colors at a time. Most of the competing teams are using Sony hardware to classify images. The variation is how they determine the rectangles for colors in each of 32 levels (Hengst et al 2001, Sammon & O'Gorma, 2000).

In practice, the provided hardware was not very convent for use and there were problems of ill-fitting rectangles (Hengst et al 2001, Chen et al, 2002). Instead the UNSW team has moved toward a software solution to the color classification problem. They have used polygons instead of color rectangles and each color classification was described by a polygon in the UV plane. They perform color calibration by collecting 25 snapshots of different objects at different locations on the field. By using a simple painting program, every pixel is manually classified. All the pixels are used as training data for learning. From the training data, for every color, one polygon that best fits the training data of that color is automatically generated. The color of an unknown pixel is then determined by checking what polygons its UV values lie in. The learnt polygons are encoded in such a way that the on board classification can be done fast (Hengst et al 2001, Chen et al, 2002).

2.1 Object recognition - current and new system

Using the offline training module a look up color table is generated. The on board software uses the color table to classify each pixel in the input image. The object recognition module is then used to identify the objects in the image. The object recognition is based on the "Four-connected color blobs" and the unique color of the object. Bounding box around each color blob determines the size of each blob found. The Bounding box is used in object identification and in calculation of different parameters. Figure 1a shows the image and 1b shows the identified objects (from Chen et al, 2002).

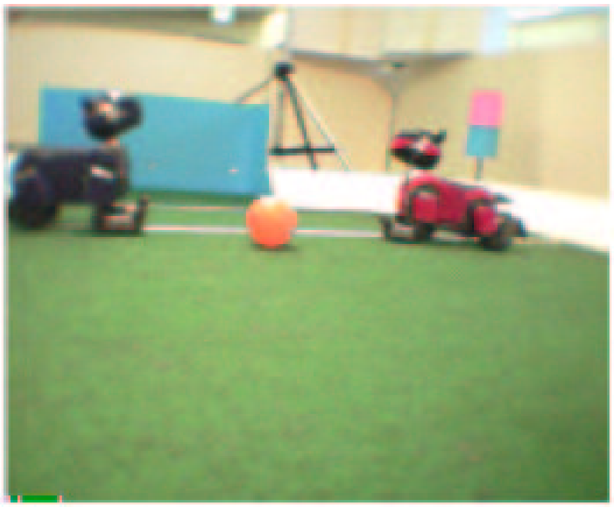

Figure 1a: What the robot sees

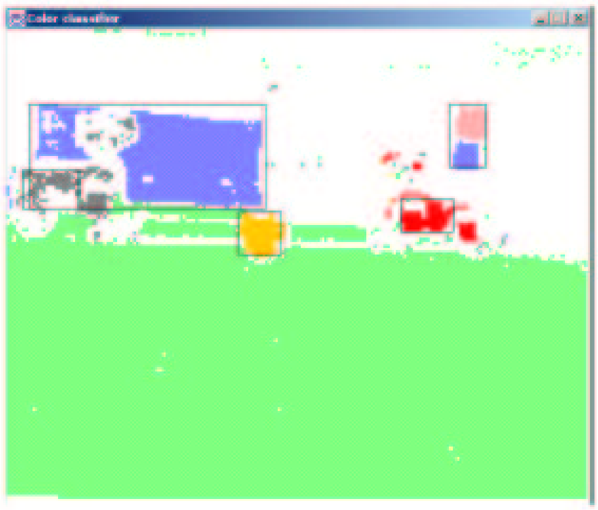

Figure 1b: Identified objects within bounding boxes

We have taken different approach to object recognition. We firstly identify the regions using an edge detection algorithm, and then categorize the regions using a decision tree process. The proposed approach should have an advantage over the existing one, since the edge detection is used to find the regions in the image, the processing will be more robust than a process which operates on a few pixels at a time. It also means that other attributes specific to the region are available, such as moment of inertia and wiggliness, which may help identify an object.

The object detection process involves a few steps: preprocessing, edged detection and region detection.

2.2 Preprocessing

Once the picture has been loaded or capture, some preprocessing must occur on the image before its edges can be detected. Image pre-processing by using a bandpass filter is performed in order to reduce noise in the image. The filtering process is performed by means of a two dimensional convolution in the spatial domain (Sammon & O'Gorma, 2000, Henrich, 1999).

2.3 Edge Detection

Edge detection was performed using the Sobel edge detection method, which resulted in a measure of "edginess" of each pixel in both horizontal and vertical components. Once we have a map of the "edginess" of each pixel, the next requirement is to determine which pixels are actually part of a genuine edge. The next function applied returns a binary "edge" image. Unfortunately the resulting edges are quite thick, often several pixels. This could be reduced by applying thinning operations with edge linking, however this was not attempted in the first place because the images were relatively simple and the thickness of the edges was deemed not to be a severe problem. However, if we deal with more complex images and if the edges are so thick the objects delimited by them can end up being quite small, which makes identifying them more difficult.

2.4 Region Detection

Region detection is done by performing a “flood fill” of the image for each region. The function starts at the top of the edge map, and finds the first non-edge pixel. Then a flood fill is performed, setting each pixel to an edge pixel as it finds more adjacent non-edge pixels, and also writing this pixel into another map (which will give the shape of the region). As more pixels are discovered, a few of the region characteristics are updated. These include the area, bounding rectangle, average red, green and blue values and average x and y co-ordinates (that is, position of the centroid, sometimes called first moment, or first moment of area). Once the fill has finished, other attributes of the region are calculated: such as moment of inertia (also called second moment of area), perimeter (measured by counting the number of pixels on the edge) and derived attributes, such as average hue, saturation and value, wiggliness (square of perimeter divided by area) and scaled moment of inertia (moment of inertia divided by square of area).

The next step after generating edge map is learning to recognize objects.

3. Machine learning experiments

The aim of the project is to train the vision system to recognize objects under different lightning conditions. The ideas is to describe each segmented object with a set of attributes that are calculated from the image and then to apply machine learning to build a classifier that can recognize objects in the image. In order to develop such a classifier, we have to choose good vision parameters - attributes to describe the objects

3.1 Region characterization

Each region in the edge map was described by a set of the following attributes: the area, bounding rectangle, average red, average green and average blue values and average X and Y co-ordinates (that is, position of the centroid, sometimes called first moment, or first moment of area), moment of inertia (also called second moment of area), perimeter (measured by counting the number of pixels on the edge), and derived attributes, such as: average hue, saturation and value, wiggliness (square of perimeter divided by area) and scaled moment of inertia (moment of inertia divided by square of area) (Mojsilovic et al 1991).

3.2 Problems with the parameters

Note that area, perimeter and moment of inertia are all dimensional properties. We can't use dimensional quantities to help recognize a region in this case. In this domain, the camera may be positioned anywhere in the arena, and perspective projection will ensure that objects further away will appear smaller in our image. As a result, we can't say for example that a particular region is a goal because it appears to have the right sort of area, since the area will change by orders of magnitude with camera position. As a result, we need to find some dimensionless quantities.

We have used two such quantities. The first is **"wiggliness"**, defined by the square of the perimeter divided by its area. Note that ((L) ^ 2) / (L^2) = 1, showing that this quantity is indeed dimensionless. If you make the region twice as "big", it will have double the perimeter and quadruple the area, and as a result will have the same wiggliness.

The second is **scaled moment of inertia**, which we call SMOI. This is defined as moment of inertia divided by the square of the area. Note that MOI has dimensions

ML^2. We will assume that each pixel represents a fixed mass, or conversely that the object has uniform area density. We can then define MOI as being the area density times the area (to get mass) times the square of the RMS distance from the centroid. If we ignore the area density, since we will not be measuring this or changing it, we see that MOI has the dimensions L^4. Hence SMOI has dimensions (L^4) / (L^2)^2 = 1.

Other attributes are calculated as well, such as the average value, which is the average of the average red, green and blue components, the average saturation, which is 1.0 - the minimum of the average red, green and blue components divided by the average value and the hue, which is the angle on the color wheel, where red is 0, green is 120 and blue is 240, and is calculated using inverse tangents.

Specially developed program for classification enables the user to classify each area and to automatically generate a training data file with the attributes of each region along with its classification. The file with the training data and the file with the specifications of the attribute and classes were fed into See5 (C4.5 machine learning program) (Quinlan, 1993).

The program for classification has an auto-classify function, where the region is classified according to a decision tree. This allows the user to see how the program has identified each region, and also aids in further classification since most of the classification can be performed automatically and anomalies can be corrected, rather than having to classify each region separately.

By analyzing the results from the first series of experiments we have found out that a few of the region attributes were deemed to be suitable for region recognition. These were the average red, green and blue components, the SMOI, the wiggliness and the average hue, saturation and value components. This choice of attributes ensures that the ML algorithm does not draw conclusions based on unrelated information, such as position of centroid or the area of the region. The above mentioned attributes were used in the following experiments.

4. Experiments and results

The resolution of the images used in our experiments was 160 by 120 pixels and the information in each pixel was 24 bits per pixel in YUV format. The images were collected and saved in PPM binary format as images 320 by 240 pixels. We have written a program that converts those images into lower resolution for faster processing. The PPM images were subsampled by averaging every four pixel region into a single color, and written as a binary file (160 by 120 by 4 bytes long).

We have collected a large number of images under varying lighting conditions and of different scenery. We were quite methodical in the pictures we took in an effort to include all the important objects at various distances, and also changing lighting conditions.

Once we had classified 22 of the images, we fed the results of this classification through C4.5, a machine learning program, which produces pruned decision trees based on the input data. We then incorporate this decision tree into the automatic classification code in the program, so that it could automatically classify more accurately.

We classified additional 12 images, bringing the total to 34 images, and repeated the learning process. Figure 2 shows original image (2a), edge map image (2b) and classified image using learned classifier (2c).

We repeated the classification/learning steps by increasing the number of the training examples. Fig 3 shows the results from the learning with 117 images.

The total number of images collected was 3000, and the ones chosen for classification were a broad range, rather than all from the same set of lighting conditions. Figure 4 shows more complicated image with the average colors (4c) and true colors (4d).

5. Discussion

Color detection can be seriously affected by lighting conditions and other variations in the environment. One important problem for the Sony robots is that the robot vision system need to be recalibrated as lighting conditions change, otherwise they fail to recognize objects or classify them incorrectly. Manual calibration as is used so far is very time consuming and is limited.

The approach we are taking toward solving this problem is to use learning to train the vision system to automatically classify objects in the scene. The training is performed by using supervised machine learning.

A few image-processing steps were applied to the images in order to prepare them in a suitable form to be used for learning. The bandpass filter was chosen because it can remove specular highlights as well as reducing noise while maintaining edges. In addition, the Sobel filter was chosen based on its computational simplicity and its apparent effectiveness.

The set of fourteen image attributes was originally calculated and used for area (object) characterization. A sub set of eight image attributes was experimentally determined and used in the further experiments.

The experiments were performed in such a way that the learned knowledge was used to help the classification and only misclassified areas were classified manually. With each iteration the learned knowledge was augmented and a fewer mistakes were detected. By attaching a leaning module to the vision system we provide condition that can enable automatic adaptation of the vision system to new environment conditions.

Next, we plan to test the results and to evaluate our method on real Sony robots. We also plan to investigate other methods for automatic color and object classification.

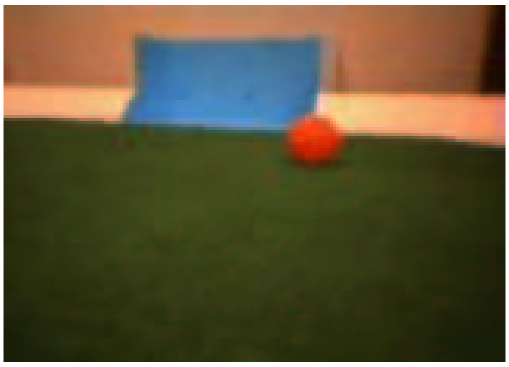
Figure 2a:An image from the robot camera

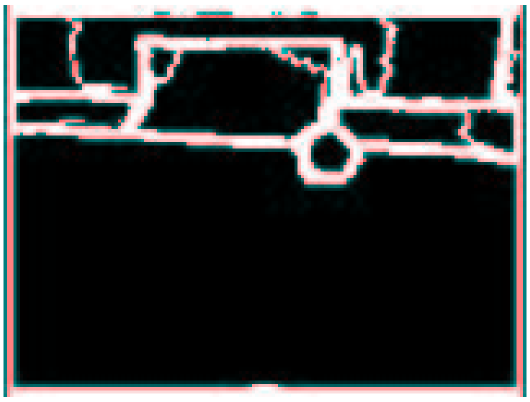
Figure 2b: The edge map derived from the image

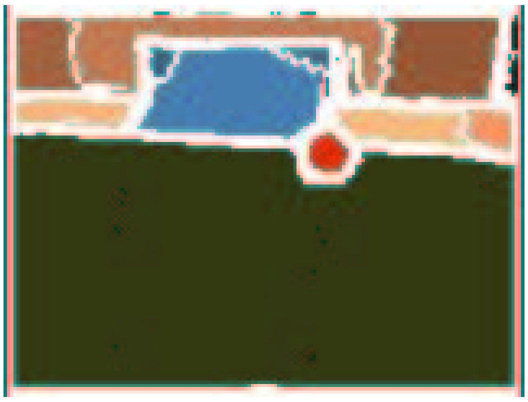
Figure 2c: The composite image showing the identified regions and their average colors.

```
C4.5 [release 8] decision tree generator
Fri Oct 19 12:56:50 2001
----------------------------------------
    Options:File stem <bigset>

Read 631 cases (8 attributes) from bigset.data

Simplified Decision Tree:

green <= 210 :
|   saturation > 29 : goal2 (38.0/2.6)
|   saturation <= 29 :
|   |   saturation <= 10 :
|   |   |   blue <= 181 : dk (121.0/1.4)
|   |   |   blue > 181 :
|   |   |   |   wiggliness > 18 : dk (17.0/1.3)
|   |   |   |   wiggliness <= 18 :
|   |   |   |   |   scaledMOI <= 77 : dk (7.0/1.3)
|   |   |   |   |   scaledMOI > 77 : wall (8.0/1.3)
|   |   saturation > 10 :
|   |   |   blue <= 131 :
|   |   |   |   wiggliness <= 5 : dk (8.0/1.3)
|   |   |   |   wiggliness > 5 :
|   |   |   |   |   scaledMOI <= 160 : ground (121.0/5.0)
|   |   |   |   |   scaledMOI > 160 :
|   |   |   |   |   |   green <= 135 : dk (9.0/1.3)
|   |   |   |   |   |   green > 135 : ground (6.0/2.3)
|   |   |   blue > 131 :
|   |   |   |   value <= 173 : dk (47.0/2.6)
|   |   |   |   value > 173 : wall (3.0/1.1)
green > 210 :
|   saturation <= 10 :
|   |   red <= 217 : dk (3.0/2.1)
|   |   red > 217 :
|   |   |   value <= 254 : wall (212.0/5.0)
|   |   |   value > 254 :
|   |   |   |   wiggliness <= 12 : dk (4.0/1.2)
|   |   |   |   wiggliness > 12 : wall (14.0/1.3)
|   saturation > 10 :
|   |   blue <= 251 : wall (2.0/1.0)
|   |   blue > 251 : dk (11.0/1.3)


Tree saved

Evaluation on training data (631 items):

          Before Pruning        After Pruning
         ----------------   ---------------------------
         Size    Errors      Size  Errors      Estimate
         39     8( 1.3%)     33   10( 1.6%)   ( 5.3%)
```

Figure 3: Induced decision tree derived from the 117 images. Only the pruned tree is shown.

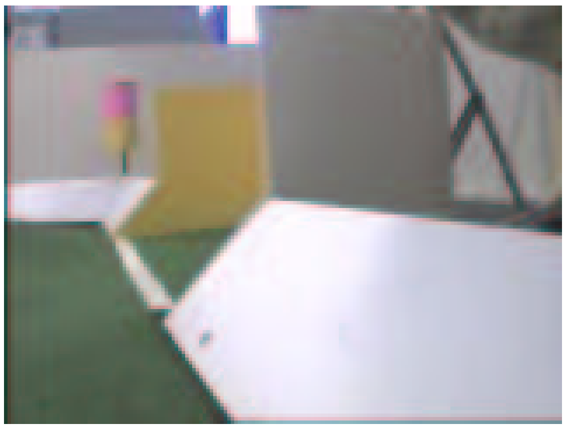

Figure 4a: More complex image

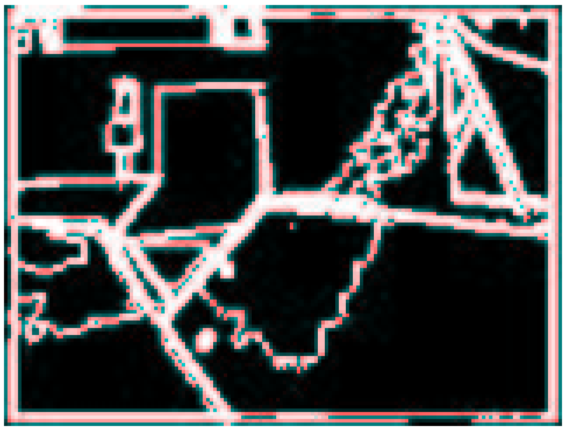

Figure 4b:The edge map

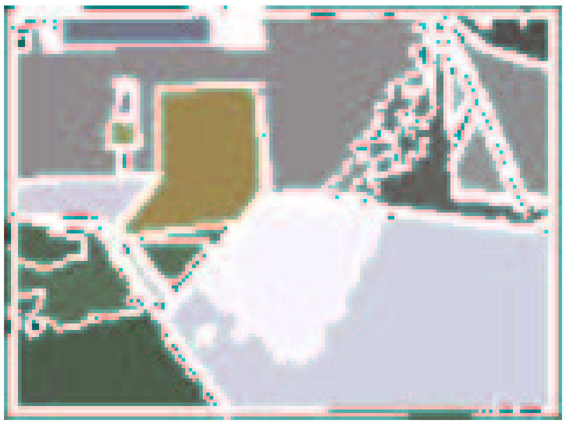

Figure 4c: The composite image showing regions map and the average color of each region

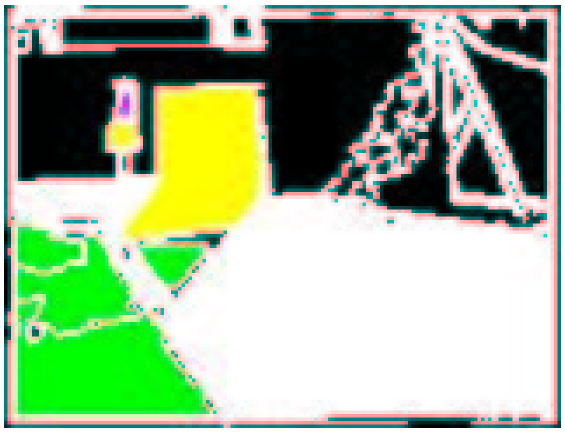

Figure 4c: The composite image showing regions map and the average color of each region

Acknowledgements

We would like to thank Phil Preston for his help during image collection and Claude Sammut for giving us access to the robotics laboratory.

References

Hengst, B., Ibbotson, D., Pham, S. B., Dalgliesh, J., Lawther, M., Preston, P., Sammut, C. (2001) The UNSW RoboCup 2000 Sonny Legged League Team, *RoboCup 2000: Robot Soccer World Cup IV*, Springer

Chen, S,. Siu M., Vogelgesang, T., Yik, T.F., Bernhard Hengst, B., Pham, S.B., Sammut, C.(2002) The UNSW RoboCup 2001 Sony Legged League Team, *RoboCup 2001: Robot Socer World Cup IV,* Springer,

Sammon, M.J., O'Gorman, L. (2000) *Practical Algorithms for Image Analysis*, Cambridge University Press

Henrich, D., (1999) Space-efficient Region Filling in Raster Graphics, *The Visual Computer: An International Journal of Computer Graphics*, **10**(4), pp205-215

Mojsilovic, A., Kovacevic, J., Hu, J., Safranek, R., Ganapathy, S.K. (1991) Matching and Retrieval Based on the Vocabulary and Grammar of Color Patterns, *International Journal of Computer Vision*, 7(1):11-32, 1991

Quinlan, J.R (1993*) C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann.