Fast, Robust Techniques for Colored Object Detection in Variable Lighting Conditions

Brett Browning and Dinesh Govindaraju
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
brettb@cs.cmu.edu, dineshg@cmu.edu

***Abstract*— In this paper, we present two new algorithms, Histogram Threshold Vector Projection (HTVP) and Histogram Threshold Ellipsoid Distance (HTED), for fast color-based detection of objects under variable illumination. We have developed these algorithms for the purposes of robot perception in an adversarial multi-robot task, where the competing needs of fast processing and robustness must be balanced. The adversarial environment combined with the high speeds of the robots places constraints on the computational resources any algorithms can use. We examined the performance of these algorithms in a number of realistic settings and compared their performance against the benchmark of CMVision [3], a publicly available fast color segmentation library widely used in the task of robot soccer. Our results demonstrate comparable computational performance to CMVision, with the added robustness to changing illumination.**

Keywords--Robot vision, color object recognition, multi-robots, adversarial robots.

I. INTRODUCTION

In this paper we present two new techniques for the problem of colored object detection by autonomous robots operating in dynamic, adversarial environments under variable lighting conditions. Concretely, we examine the problem of colored object detection in an outdoor environment by an autonomous robot using a monocular color camera. We investigate this problem using the new Segway RMP platform for operating in the task of Segway Soccer, where teams of Segway riding humans and robots compete in a soccer-like game outdoors. Hence, the environment is adversarial, and highly dynamic, the combination of which presents many interesting and unique challenges to perception.

The combination of variable lighting conditions, dynamic environments, and autonomous robots makes object segmentation and recognition challenging due to the competing needs for fast, computationally tractable algorithms that are at the same time able to adapt or overcome changes in lighting. Here, fast means that the algorithm runs as close as possible to the camera's full frame rate of 30Hz, while still leaving processing resources available for robot autonomy algorithms. Such speed requirements are necessary when the domain is adversarial, and robots are able to achieve speeds of 3.5m/s (8mph) and operate in close proximity to one another.

This research was sponsored by the United States Army under Grant No. DABT63-99-1-0013. The content of the information in this publication does not necessarily reflect the position or the policy of the Defense Advanced Research Projects Agency (DARPA), the US Army or the US Government, and no official endorsement should be inferred.

There have been numerous investigations into vision based scene segmentation, object recognition, and/or tracking that provide a wealth of potentially useful techniques for this problem (e.g. [1, 5, 9,10]). However, the competing needs for fast vision and robust adaptation to lighting changes presents unique challenges that, to our knowledge, have not yet been addressed within the literature. On one hand, the stringent need for fast algorithms precludes many approaches that have been developed within the computer vision community that might otherwise be effective. Likewise, the need to handle variable lighting conditions precludes direct use of the fast color segmentation techniques that have been developed. Of the latter, the algorithms that have grown out of the RoboCup research including predominantly colored region based approaches (e.g.[2,3,6,7]) but also including edge based approaches [8], and arc fitting techniques [4]. These algorithms are well suited to adversarial tasks due to their impressively fast processing speeds. The common use of fixed color threshold maps means that these algorithms are not well suited to variable lighting conditions, however. We attempt to extend beyond these limitations whilst retaining the computational efficiency needed for an adversarial robot.

Figure 1. Segway RMP

In the following section we concretely describe the problem domain and the needs for the task. In section III, we describe the two new techniques we have developed to address this problem. Additionally, we provide a brief description of the CMVision algorithm used as a benchmark for the experimental results described in section IV. We then conclude with a discussion on the relative merits of the three approaches.

II. Problem Description

The problem involves autonomously running a team of Segway RMPs in a game of outdoor soccer. The RMPs have on board laptop computers to handle all vision and motion processing and obtain raw vision data from a Philips 690 USB webcam. The camera is a monocular color camera with a frame rate of 30Hz, a field of view of 110 degrees and will be used in the detection of various types of objects during the game, namely the ball, other robots and field markers. The camera is mounted at the front of the robot, roughly half a meter off the ground and is tilted at an angle of approximately 50 degrees above the vertical. Due to the mobile nature of the robot, a bulkier but more accurate camera which might have been suitable for static mounting (such as above the field if the game was played indoors), is not practical in this situation and as a result, variances in observed color values of the USB camera may be higher resulting in more challenging vision processing.

The camera also views objects on the playing field in real time and so it is imperative that vision processing is optimized to run in a dynamically changing environment. Higher-level algorithms that rely on vision processing to generate strategies and tactics also demand that raw vision data is interpreted quickly so that they can issue effective commands to the robot. With this limitation in mind, any highly accurate but computationally intensive algorithms and operations prove to be unfeasible when implementing the vision system.

Figure 2. View from Camera Mounted on Segway RMP in the afternoon sunlight during the early autumn. A size 4 orange soccer ball and a human ruding a segway are visible in the image.

Dynamic and uncontrollable field lighting conditions are a natural consequence of operating the robots outdoors and this causes certain factors to be taken into account when designing the vision processing system. Initial calibration to determine color threshold values cannot simply be adopted from previous such systems without some amount of adaptation as the amount of light on the field of play is not adjustable and may not be consistent among different games resulting in the need for robust calibration to be implemented on a per game basis. Color threshold values may also not be consistent throughout a particular game as lighting conditions constantly change such as when the ball rolls into shadowy areas. Due to this, some manner of dynamic calibration, utilizing constantly changing threshold values, might also be beneficial to the implementation of an efficient algorithm.

III. Three Fast Techniques

We now describe the three approaches compared in this paper for the vision problem outlined above. We begin with the fixed-threshold CMVision [2,3] based approach, followed by the new techniques proposed here; Histogram Threshold Vector Projection (HTVP), and finally the Histogram Threshold Ellipsoid Distance (HTED). We finish this section by presenting the high-level filters used to identify the ball object from labeled blobs. We use this technique for all three algorithms compared in this paper.

A. CMVision with Fixed Color Thresholds

Bruce *et al*. developed CMVision [2,3] for the task of performing fast color blob tracking for multi-robot teams in adversarial environments. Since its inception, it has gained widespread use and has been used as a benchmark for comparing new techniques (e.g. [6]). It is freely available at *http://www.cs.cmu.edu/~jbruce/cmvision*. Much of CMVision's capability resides in its very efficient, cross platform implementation. In terms of operation, the algorithm first segments an image using a pre-fixed color threshold map and then collates connected regions through an efficient connected component analysis. Users then build high-level object detection algorithms using the resulting labeled 'blobs'.

More formally, the algorithm takes an image consisting of N pixels drawn from a color space C. Here we consider only YUV pixels, such that each pixel p_j is a vector $(y_j, u_j, v_j,)^T$. In other words, $C=\{(y, u, v) \mid y, u, v \in [0,1]\}$. During the initial segmentation process, CMVision maps each pixel in the image to a symbolic label drawn from the pre-defined set of symbolic labels $L=\{l_1,\ldots, l_M\}$. The threshold map, $T:C \rightarrow S$, is implemented as a look up table for speed purposes. Thus, we derive a set of symbolic pixel labels as:

$$s_j = T(p_j), \quad s_j \in L, p_j \in C, j = 1 \ldots N \tag{1}$$

Upon labeling each pixel, blobs are formed in a two-step process. First, contiguous horizontal runs are compressed via Run-Length-Encoding. Vertically contiguous runs are then connected to form regions, or blobs. Once a list of regions, or blobs, is extracted high-level filtering techniques can be used to detect the desired object. We describe these techniques further in Section D.

To calibrate CMVision one must generate the threshold map *T(.)*. [2] developed GUI tools to generate *T(.)* by hand, while [3] use a supervised learning technique to generate *T(.)* from hand labeled images. In either case, operator expertise is required and there are no algorithms for *adapting T(.)* as the lighting conditions change. For outdoor environments adaptation is critical due to the high variability in lighting conditions. We now describe the second technique, which incorporates adaptation into the initial segmentation process.

B. Vector Projection with Histogram Based Threshold

Our second technique builds from the work of CMVision, but incorporates adaptation as a key part of the segmentation process. The key to the approach is to first transform the raw

image from the input color space to an intensity space where 'color' of the object in question dominates the local surrounding. A histogram-based technique can then be used to derive a suitable threshold for segmenting the image. After segmentation, the same connected component analysis and high level filtering used for CMVision can be applied to detect the desired obstacles.

The first part of the process, mapping the input image to a single dimensional 'color' space follows standard tracking techniques (e.g. [11]) with the use of vector projection. Thus, we have the input pixel $p_j=(y_j, u_j, v_j)^T$, which is transformed by:

$$S_j = \frac{(a \quad b \quad c)^T \cdot (y_j \quad u_j \quad v_j)}{|a|+|b|+|c|} \tag{2}$$

Here S_j is the resulting intensity for pixel j, and $(a, b, c)^T$ is the prototype vector used for projection. The projection is partially normalized and in practice bounded to the limits of [0, 255]. With good choices for the projection vector, the resulting intensity image will be bright for pixels on the object and dark for non-object pixels. If so, one needs to choose a suitable threshold θ to determine if for pixel j, S_j is sufficiently bright to be an object pixel or not.

The color space mapping takes no account of variations in luminance. If the sun should go behind a cloud the choice of θ, which may otherwise have been a good value, will suddenly become a poor value. Our approach to addressing this problem is to use a histogram-based technique to determine an adaptive threshold. Histogram-based approaches have long been used to determine dynamic thresholds for gray scale images. Our initial color space mapping is chosen to ensure that the object in question becomes the brightest part of the local image. Thus, the problem resolves to determining the threshold for segment the brightest peak in the image. In other words, detecting the peak in the histogram and selecting the threshold from this. Figure 3 shows an example processed image and the resulting histogram.

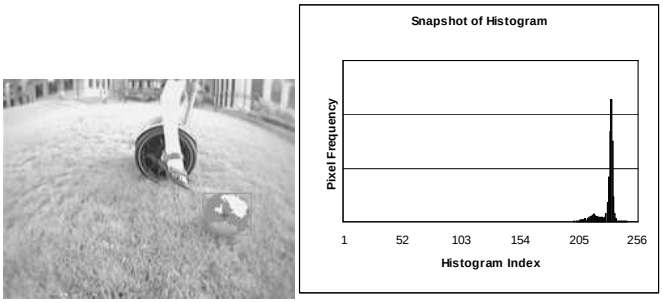

Figure 3. A sample image and the resulting peak detected from the histogram of the vector projection image.

We developed a fast, constraint based algorithm for finding the peak and determining the desired threshold. The algorithm works by processing the resulting histogram in two steps; one to find the peak and one to find the trough. Constraints based on the expected size of the ball are enforced during each step. Thus, the algorithm is:

1. Starting from $p = N_H$ find p such that

$$H(p) > p_{\min}, \quad H(p) \geq \max_{i=p\ldots N_H} H(i) \tag{3}$$

2. Starting from $m = p$, find m such that

$$H(m) \leq \min_{i=m\ldots p} H(i), \quad m < p \tag{4}$$

Where m and p are intensity values constrained to lie in $[p_{min}, N_H]$, where N_H is the maximum possible intensity value (255 in practice), and $H(x)$ is the value of the histogram at x. In practice, the histogram value $H(x)$ is a local average over a window of size w, thus:

$$H(x) = \frac{1}{w} \sum_{x-w/2 \leq i < x+w/2} h(i) \tag{5}$$

where $h(.)$ is the raw histogram function. The peak found by the algorithm is only accepted if the area under the peak, A_{peak}, is within the range $[A_{min}, A_{max}]$. The area is defined as:

$$A_{peak} = \sum_{j=i}^{N_H} H(j) \tag{6}$$

The trough value, m, is chosen as the threshold value for the image. Each pixel is then labeled using this threshold. The resulting binary image is passed through CMVision to extract the 'blobs' from the image. The blobs are then filtered using the high-level filters described in Section D.

C. Ellipsoid Distance with Histogram Based Threshold

The approach just described works well for a range of single color classes. The color class must be one for which a suitable vector can be chosen to produce a histogram with a recognizable peak consisting of the color class in question. For multiple, more widely useful colors, a better approach is desired. Our last approach attempts to address this issue. We reuse the technique of histogram-based adaptive threshold and CMVision for blob extraction, but modify the initial color mapping process to first populate the histogram.

Using each pixel's color values as a three-dimensional color vector, we now generate scalar 'intensity' values by first defining an ellipsoidal region in the color space that encapsulates most or all of the color values found in the object we wish to detect. This can be achieved by taking a sample screenshot of the object and finding the mean color values of the pixels in the object, as well as maximum difference for each color dimension, D_y, D_u, D_v, from the mean for that color as:

$$D_\mu = \max_i \left| \mu_i - \bar{\mu} \right| \quad \forall i \in \{1\ldots N\}, \quad \mu = y,u,v \tag{7}$$

Here the mean has its usual meaning as $\bar{a}=n^{-1}\Sigma a_j$ for $a=y,u,v$. This would result in an ellipsoid characterized by:

$$\frac{|\bar{y}|}{D_y^2} + \frac{|\bar{u}|}{D_u^2} + \frac{|\bar{v}|}{D_v^2} = k \quad (8)$$

We then find the scalar value S_i, of each pixel, by determining the distance of the pixel from the center of our ellipsoid while accommodating for the different axial deviations. These deviations are taken into account due to the different color variation in each dimension of an object. For example, the range of values of the Y component are usually much wider that those of the U and V components. This makes object detection more accurate. These constants can be calibrated off-line using hand-labeled sample images to estimate the values of D_y, D_u, D_v, respectively. The resulting color space mapping is given as:

$$S_i = \sqrt{\left(\frac{|y_i - \bar{y}|}{D_y}\right)^2 + \left(\frac{|u_i - \bar{u}|}{D_u}\right)^2 + \left(\frac{|v_i - \bar{v}|}{D_v}\right)^2} \quad (9)$$

Performing this computation for each pixel can be a time consuming task. However, we can once again vastly improve the speed of the technique through a pre-calculated lookup table as per CMVision. The resulting image can then be processed using the histogram threshold technique described above, region extraction with CMVision, and high level filtering. We now describe the high-level filtering techniques used before proceeding on to the results of the approach.

D. High-level Object Detection

In this framework, high-level filters are used to filter the regions segmented by whichever color segmentation process is used. The filters are based upon known geometric constraints derived from the object to be found. For the ball, we essentially utilize the same techniques developed by [12]. In particular, we run a number of parallel confidence estimators that report a value in the range [0, 1], based on the match quality for that particular constraint. The product of confidences produces the overall confidence estimate. For the ball, these estimates are the expected bounding box size, given the location of the blob in the image and knowledge of the camera's projective geometry and a ground plane constraint. A second filter matches the expected pixel count within the boundary box given its size, a last filter analyzes the bounding box shape.

IV. Results and Discussion

To gauge the feasibility and efficiency of the three methods of object detection in variable lighting conditions, we performed a number of tests examining the recognition performance as well as the examining the processor usage of the algorithm. The latter is critical to the task in question, where an algorithm that runs too slowly, even if it works perfectly, is of little value. Figure 4 shows the typical segmentation result for finding the orange ball in an indoor environment. The image on the right shows pixels labeled in a uniform color and the bounding box and '+' around the found object.

Figure 4. The left image shows the raw output from the camera in a lab setting. The right image shows the results after processing. 'Ball' pixels are shown as orange (shown in gray here), a bounding box is drawn aroun the resulting region, and its centroid is marked with a '+'.

A. Performance

Object detection accuracy was measured by running the three vision algorithms on three image sequences focused on an orange, size 4, soccer ball. All sequences were recorded from the Segway RMP platform and run through the different vision systems separately. The high-level filters were identical for each approach. The color table for the complete CMVision approach was created by hand using sample images from each domain. The histogram parameters for the remaining two techniques were identical and were found by hand calibration. The mean and distance estimates for the ellipsoid approach HDET, were found from sample images in each setting. Figure 5 shows a sample snap-shot from each testing domain. The three test sequences were:

1. Indoors, stationary robot, stationary ball
2. Indoors, stationary robot, moving ball, lights changed during sequence
3. Outdoors, moving robot, moving ball

Figure 5. A snap-shot from each image sequence.

The first evaluation consists of constructing a confusion matrix displaying the correctness of the detection algorithms. For ground truth, a human operator for each frame of the sequence recorded the presence or absence of the ball. Thus, the confusing matrix represents the detection results of each algorithm when compared to this hand labeled sequence. The following table shows the resulting confusion matrix for each technique for the outdoor image sequence.

TABLE I. Confusion Matrix using Fixed Color Thresholds (A), HTVP (B), HTED (C) Using Outdoor Image Sequence

	% Frames in which Object is Detected			% Frames in which Object is Undetected		
	A	**B**	**C**	**A**	**B**	**C**
Object Present in Image	63.5	47.0	57.7	11.2	27.7	16.9
Object Absent in Image	19.2	0.0	0.0	6.2	25.4	25.4

TABLE II. CONFUSION MATRIX USING FIXED COLOR THRESHOLDS (A), HTVP (B), HTED (C) USING INDOOR IMAGE SEQUENCE WITH VARIABLE LIGHTING

	% Frames in which Object is Detected			% Frames in which Object is Undetected		
	A	**B**	**C**	**A**	**B**	**C**
Object Present in Image	53.4	100	100	46.6	0	0
Object Absent in Image	0	0	0	0	0	0

From these results, we can see that ellipsoid, HDET technique is clearly superior. We attribute the lack of type II errors to the histogram-based threshold technique correctly identifying a useful threshold if the ball is truly visible. Although we report no explicit results, we note that the type I errors occur mostly due to shadows, for which none of the approaches is particularly good at handling.

The second evaluation comprises of gauging how object detection varies with changing luminance values – one of the major goals of this work. Using the second test sequence, the ball was placed in front of the stationary robot in an indoor environment (a cluttered laboratory). We varied the lights by enabling/disabling some overhead lights independently. All three detection algorithms were calibrated using the initial luminance values. Figure 6 shows the results of the ball identification for each algorithm along with a plot of the average luminance value of each frame.

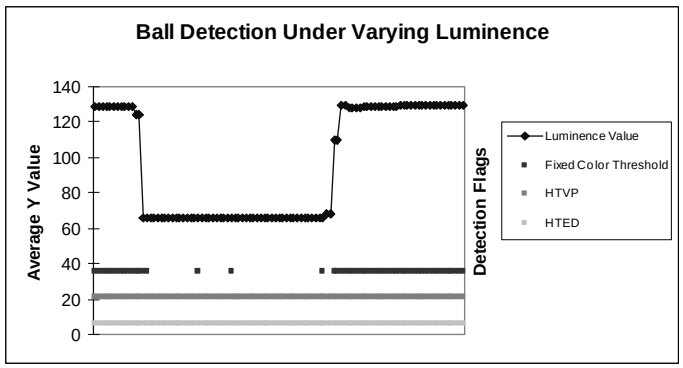

Figure 6. Identification performance under variable illumination.

From these results it is clear that employing a dynamic threshold mechanism improves the detection of the object when the luminance of the scene varies.

The final set of accuracy evaluations we report here consist of monitoring the reported position of the ball with respect to its actual position. Using the same calibration procedure as above, for the first test image sequence where no motion was involved, the reported locations for hand labeling had a mean distance error from the hand chosen location of 6.2 pixels (σ^2 =22.3) for CMVision, 3.8 pixels (σ^2 =4.0) HTVP, and 4.9 pixels (σ^2 =14.6) for HTED. Clearly, each technique is reasonably accurate at reporting the correct region centroid.

For the outdoor sequence, where both robot and ball were in motion, we show the reported *x* image coordinate as a function of time. Also shown in each graph is the ground truth as labeled by a human operator.

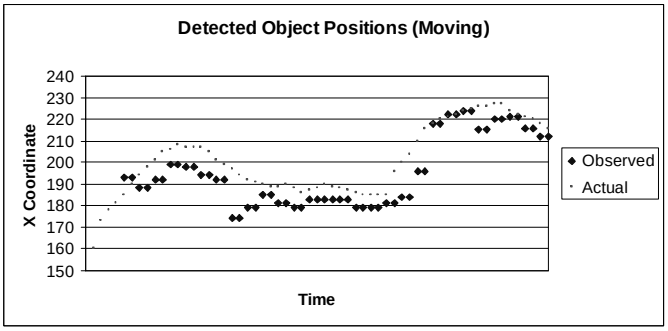

Figure 7. Detected X Coordinate using Hand Labeling of Color Thresholds

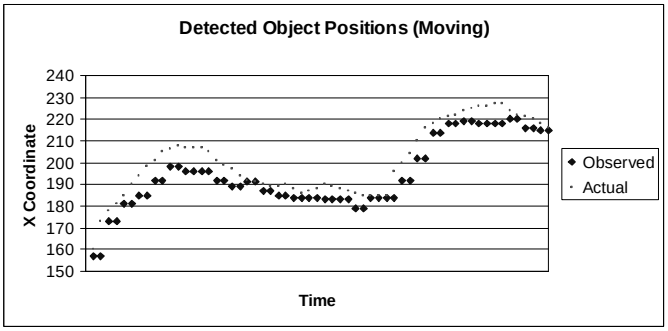

Figure 8. Detected X Coordinate using Vector Projection

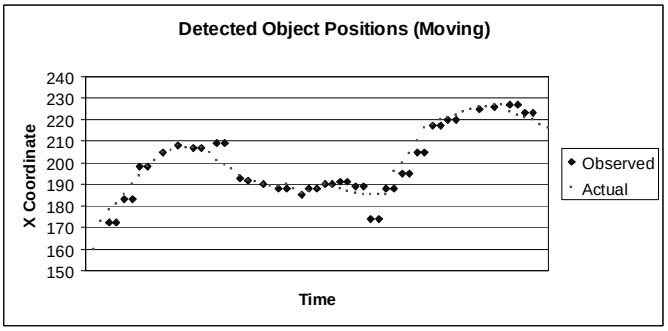

Figure 9. Detected X Coordinate using Ellipsoid Distance

Examination of these graphs, in which the X coordinates are in pixels reveals that all three object detection methods perform fairly well in tracking an object in outdoor conditions but also that the ellipsoid distance algorithm performs slightly better in this task.

B. Processor Test

We now examine the computational requirements for each technique. Processor performance comparisons among the three methods of object detection were carried out by sampling

the register clock during the execution of the object detection portion of the algorithms and the results of this evaluation can be seen in Figure 11. These results are for the Pentium IV, 17GHz processor. For gauging each algorithm, three sequences were used, namely one with a stationary ball and robot, moving ball and stationary robot, and finally one with a moving ball and robot. From the results of the evaluation, it can be seen that processing done by the vector projection and ellipsoid distance algorithms requires roughly the same amount of time and that they are slower than the hand labeling algorithm by approximately six times. This can be attributed to the fact that for the hand labeling algorithm, marking object pixels is done using an immediate lookup without the need for calculations on a per pixel basis as carried out by the vector projection and ellipsoid distance methods.

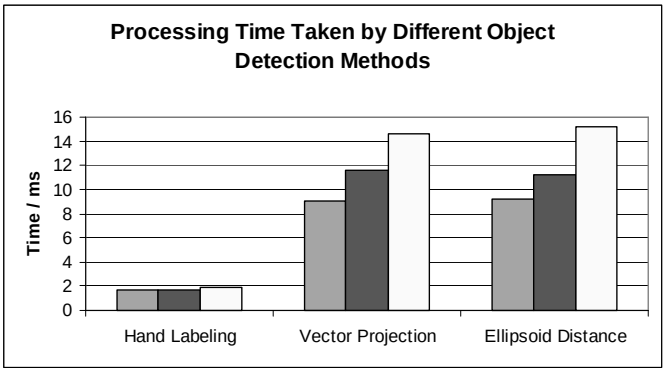

Figure 10. Processor Usage Statistics

Clearly, the CMVision based approach has the best computational usage, but clearly lacks the ability to adapt to changes in lighting. Of the two adaptive techniques presented, the ellipsoid technique represents the best combination of processor usage and adaptability. Moreover, it can be adapted to a wider range of color classes than the HTVP approach. Thus, we conclude that it is the algorithm of choice for this particular task.

V. Conclusion

In this paper we have presented two new approaches for robustly, and efficiently, detecting colored objects in variable lighting conditions such as those expected in Segway Soccer. We examined the performance of each of these algorithms in comparison to the performance of CMVision, a popular technique within the domain of RoboCup robot soccer for fast color segmentation. Based on the resulting performance, it appears that the ellipsoid HTED technique provides the best performance in domains with variable lighting. Moroever, although additional computation resources are required, the needs of this technique compare favorably with CMVision using current processing technology.

Acknowledgment

The authors would like to thank Manuela Veloso, Paul Rybski, Jeremy Searock, Michael Sokolsky, and David Rozner for their support in this project. Additionally, the authors would like to thank Dr. Douglas Gage and Segway for making this project possible.

References

[1] T. Bräunl and B. Graf. “Autonomous Mobile Robots with On-Board Vision and Local Intelligence”. In Proceedings of Second IEEE Workshop on Perception for Mobile Agents, 1999.

[2] J. Bruce, & M. Veloso, “Fast and accurate vision-based pattern detection and identification”. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA'03), 2003.

[3] J. Bruce, T. Balch, and M. Veloso. “Fast and Inexpensive Color Image Segmentation for Interactive Robots”. In Proceedings of IROS-2000, Japan, October 2000.

[4] G. Coath, P. Musumeci, Adaptive Arc Fitting for Ball Detection in RoboCup, APRS Workshop on Digital Image Computing 2003.

[5] D. Gavrila and V. Philomin. “Real-time object detection for smart vehicles”. Proceedings of the International Conference on Computer Vision (ICCV), pages 87-93, 1999.

[6] F. Hundelshausen and R. Rojas. “Tracking regions”. D. Polani, B. Browning, A. Bonarini, K. Yoshida (eds.). RoboCup 2003: Robot Soccer World Cup VII. Lecture Notes in Artificial Intelligence. Springer, to appear.

[7] M. Jamzad, *et. al.* “A Fast Vision System for Middle Size Robots in RoboCup”. A. Birk, S. Coradeschi, S. Tadokoro (eds). RoboCup 2001: Robot Soccer World Cup V. Lecture Notes in Computer Science 2377, Springer, 2001.

[8] T. Rofer, M. Jungel. “Fast and Robust Edge-Based Localization in the Sony Four-Legged Robot League”. D. Polani, B. Browning, A. Bonarini, K. Yoshida (eds.). RoboCup 2003: Robot Soccer World Cup VII. Lecture Notes in Artificial Intelligence. Springer, to appear.

[9] L. A. Vese and T. F. Chan. “A multiphase level set framework for image segmentation using the Mumford and Shah model”. UCLA Department of Mathematics CAM Report 01-25, to appear in International Journal of Computer Vision, 2001.

[10] P. Viola and M. Jones. “Robust real-time object detection”. Technical Report 2001/01, Compaq CRL, February 2001.

[11] J. R. Parker, “Algorithms for Image Processing and Computer Vision”, John Wiley & Sons, 1996.

[12] S. Lenser, J. Bruce, and M. Veloso. CMPack: “A Complete Software System for Autonomous Legged Soccer Robots”. In Proceedings of the Fifth International Conference on Autonomous Agents, May 2001.