

Robust Color Segmentation for the RoboCup Domain*

Zbigniew Wasik and Alessandro Saffiotti
Center for Applied Autonomous Sensor Systems
Dept. of Technology, Örebro University, S-70182 Örebro, Sweden
zbych@aass.oru.se, asaffio@aass.oru.se

Abstract

Color segmentation is crucial in robotic applications, such as RoboCup, where the relevant objects can be distinguished by their color. In these applications, real-time performance and robustness are primary concerns. We present a hybrid method for color segmentation based on seeded region growing (SRG) in which the initial seeds are provided by a conservative threshold color segmentation. The key to the robustness of our approach is to use multiple seeds to perform local blob growing, and then merge blobs that belong to the same region. We have implemented our technique on a team of Sony AIBO 4-legged robots, and have successfully tested it in the RoboCup 2001 competition.

1. Introduction

RoboCup [6] is an international research and education initiative where teams of autonomous robots compete in a soccer game in an engineered field. In order to achieve a good performance, robots need sophisticated capabilities, including motion, object recognition, mapping, localization, planning, and cooperation [10]. In this paper we focus on object recognition. In RoboCup, all the relevant objects, e.g., the ball and the two nets, are identified by unique colors. This makes color segmentation a fundamental part of object recognition in this domain. The segmentation algorithm has to be robust, which means that it has to give the right results under nominal conditions and, without retuning, give acceptable results under slightly different conditions, such as blurred images or different lighting.

The main methods for color segmentation can be classified into the following approaches, sometimes used in combination [9].

Threshold techniques rely on the assumption that pixels whose color value lies within a certain range belong to the same class or object [8]. Many teams in RoboCup use

threshold techniques [4, 2] with slight modifications to increase its performance, e.g., for images that blur at object boundaries. Threshold techniques need a careful tuning of thresholds and are extremely sensitive to light conditions.

Edge-based methods rely on the assumption that pixel values change rapidly at the edge between two regions [7]. Edge detectors such as Sobel, Canny and Susan are typically used in conjunction with some post-procedure in order to obtain the closed region boundaries. This tends to make these methods very computationally expensive.

Region-based methods rely on the assumption that adjacent pixels in the same region have similar color value; the similarity depends on the selected homogeneity criterion, usually based on some threshold value. *Seeded region growing* (SRG) is a region-based technique that takes inspiration from watershed segmentation [11] and is controlled by choosing a (usually small) number of pixels, known as seeds [1]. Unfortunately, the automatic selection of the initial seeds is a rather difficult task [5]. For instance, Fan and Yau [3] use color-edge extraction to obtain these seeds. By doing so, however, they inherit the problems of edge-based methods, including sensitivity to noise and to the blurring of edges, and high computational complexity.

In this paper, we propose a hybrid method for color segmentation that integrates the thresholding and the SRG methods. We use a threshold technique to generate an initial set of seeds for each color of interest, and then use SRG to grow color regions from these seeds. Special provisions are included in our SRG to account for the fact that we can have many seeds inside the same region. The integration of the two methods allows us to use a conservative tuning for both of them, thus improving robustness: robustness with respect to changing lighting conditions is improved because we use a conservative thresholding, and robustness with respect to blurred edges is improved because we use a conservative homogeneity criterion in SRG. Our technique has been implemented on a team of 4-legged robots participating in RoboCup 2001 [10]. The implementation is efficient, since seed generation is done in hardware and since our SRG only needs two passes through the image.

*This work was partially supported by the Swedish KK foundation.

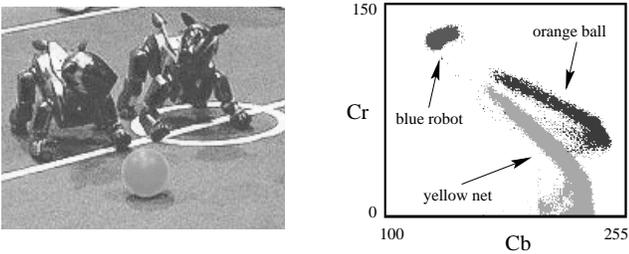


Figure 1. Left: robots used in our experiments. Right: points corresponding to three objects in the $Cb \times Cr$ color space.

2. The Hardware

All teams in the Robocup legged robot league use the same robot (ERS_210) made by Sony Corporation. The ERS_210 is equipped with a color CMOS camera which gives three 88×72 gray-scale images corresponding to the Y, Cb, and Cr components. The ERS_210 is also equipped with a hardware device, called CDT, that realizes color segmentation using a threshold technique. The CDT can distinguish at most eight colors and produces a binary (thresholded) image for each one of them.

Because it is implemented in hardware, the CDT provides a convenient and fast way to do color segmentation in the ERS_210, and many teams have chosen to use it. However, the threshold technique used in the CDT relies on rectangular areas of $Cb \times Cr$ color space, while the colors that are needed in RoboCup are distributed over non-rectangular areas. Tuning the CDT is therefore very critical, and it leads to difficult choices between accepting many false positives or many false negatives. Figure 1 shows a number of points that belong to three objects in the $Cb \times Cr$ space. Any rectangular region for, say, the orange ball must either include many yellow points, or sacrifice many orange points.

Figure 2 illustrates this tradeoff. If the CDT is tuned so that the net is segmented correctly (top center), then many of the ball pixels are missed. If it is tuned for optimal ball segmentation (top right), then many of the network pixels are missed. In some cases, like when the ball is inside the yellow net (bottom row), there is no optimal tuning for the orange threshold: CDT segmentation of the ball will either miss many of the ball pixels (center), or include many of the network pixels (right). These segmentation errors imply that the region parameters will be incorrectly estimated, leading to large errors in the estimates of the object position and hence to incorrect robot behavior.

3. The algorithm

In a nutshell, our color segmentation algorithm works as follows. We fix a set of colors to be recognized. We use the

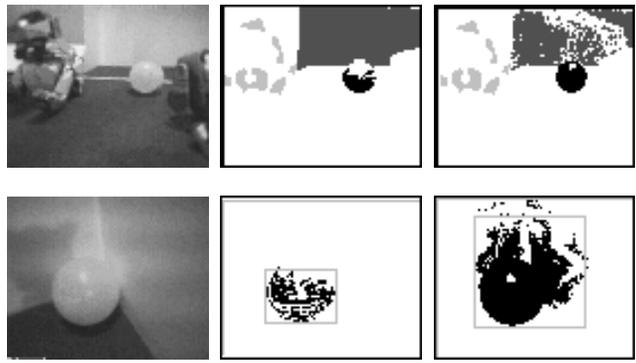


Figure 2. Problems of threshold segmentation. Top: tuning is either optimal for the net (center) or for the ball (right). Bottom: a case where no optimal tuning exists for the ball.

CDT to produce a labeled image where only pixels that belong to one of these colors with high certainty are marked as such. This is obtained by setting very strict thresholds in the CDT. (Color thresholds for the CDT are computed off-line from color samples.) Then, we use each pixel labeled by the CDT as an initial seed for a region of that color. From each one of these seeds, we grow a “blob” by including adjacent pixels until we find a significant discontinuity. The so called *homogeneity criterion* embodies the notion of a significant discontinuity: in our case, this is a change in one of the Y, Cr or Cb values beyond a given threshold.

The choice of the homogeneity criterion is critical in SRG techniques. If this criterion is too strict, the blob growing can stop prematurely because of local variations, e.g., as produced by shadows or noise. If it is too liberal, the blob can flood to an adjacent region though a weak edge, e.g., as caused by blurring. These two cases are illustrated in Fig. 3, where the single initial seed is indicated by a black dot (the ball outline is drawn for clarity). The thresholds used in the homogeneity criterion are 2 and 3, respectively.

In our technique, we address this problem by using many seeds for each color region—all those generated by the CDT. When growing a blob from a given seed, we use a strict homogeneity criterion, thus reducing the risk of flooding. We then reconstruct the full region by merging all the

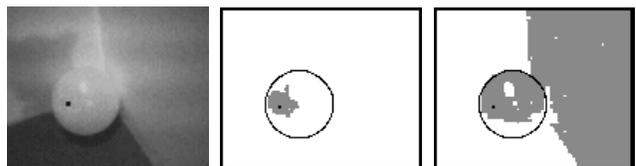


Figure 3. Problems of SRG: homogeneity criterion too strict (center) or too liberal (right).

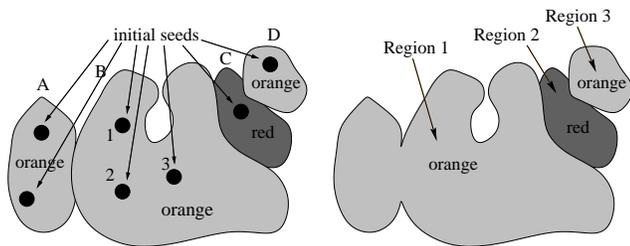


Figure 4. Merging blobs grown from multiple seeds into regions.

local blobs that belong to the same region, that is, all the adjacent blobs grown from seeds of the same color. This is illustrated in Fig. 4. Blob B is grown from seed 1. Seeds 2 and 3 are already included in B so they are not grown. Blobs A and B are generated from seeds of the same color and are adjacent, so they are merged into one region. Blobs B and C, C and D, and B and D are not merged.

The following procedure constitutes the core of the segmentation algorithm. (N is the size of the image.)

```

procedure FindBlobs (seeds)
  reset label[0:N], color[0:N], conn_table
  blob_id = 1
  for each color_id
    for each pixel p in seeds
      if (seeds[p] = color_id) and (label[p] = null)
        queue = p
        while queue is not empty
          q = pop(queue)
          label[q] = blob_id
          color[q] = color_id
          for s in 4-neighbors(q)
            if (label[s] = null)
              if CheckHomogeneity(s, q)
                push(s, queue)
            else if (label[s] ≠ blob_id)
              if color[s] = color_id
                add ⟨p,q⟩ to conn_table
          blob_id = blob_id + 1
  return (label, color, conn_table)

```

The FindBlob procedure operates on the following data structures: $label[N]$, an array that associates each pixel to a region ID, or *null*; $color[N]$, an array that associates each pixel to a color ID, or *null*; and $conn_table$, a list of connected regions of the same color. The procedure takes the $seeds$ labeled image produced by the CDT, and returns a set of blobs plus a connection table. For each desired color, the procedure scans the $seeds$ image to identify labeled pixels (seeds) of that color that have not yet been incorporated in a blob. For each such seed, it starts a standard growing loop based on 4-connectivity. If during the growing we hit

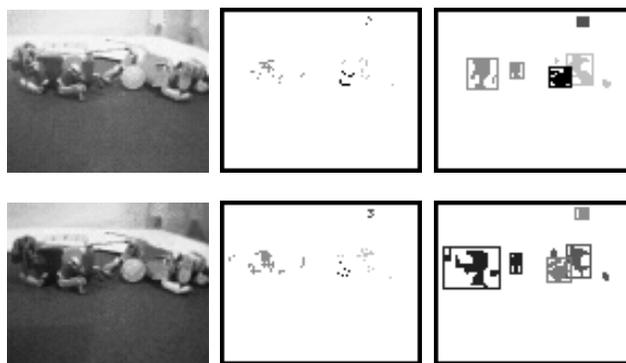


Figure 5. Camera images (left), seeds (center) and extracted regions (right). The bottom image is taken in with lower camera gain.

a pixel that has already been labeled as belonging to a different blob, then we check its color. If its color is the same as the one of the current blob, we mark the two blobs as connected in the connection table. When we add entries to the connection table, we also check it for cycles and remove them. The growing stops at the blob border, as identified by the homogeneity criterium discussed above.

After all the blobs in the original image have been individuated, a post-processing phase goes through the connection table and merges all the connected blobs of the same color as discussed above. In our experiments, 5 to 10 adjacent blobs were often generated for the same connected region. A larger number of blobs (up to 30) were occasionally generated for very large regions.

We have incorporated in the above algorithm the computation of blob parameters like the number of pixels, center, width and height. These are updated incrementally as the blobs are grown during the main phase and fused during the post-processing phase. These parameters are needed, e.g., to estimate the position of the objects.

4. Experiments

We have tested our technique extensively on a team of ERS_210 robots both in our laboratory and during the RoboCup competition. This technique provided good results, and it was fairly robust with respect to slight changes in the lighting conditions and blurring of the images.

Figure 5 shows the segmentation results on two images taken on the field. Each image contains a blue robot (left), a red robot (right), a ball (middle) and a visual landmark (top right).¹ The bottom image is taken with the camera gain reduced from 12dB to 6dB. The center image in each row shows the output of the robot's CDT for each color. Only

¹In the games, robots are "dressed" with patched colored uniforms.

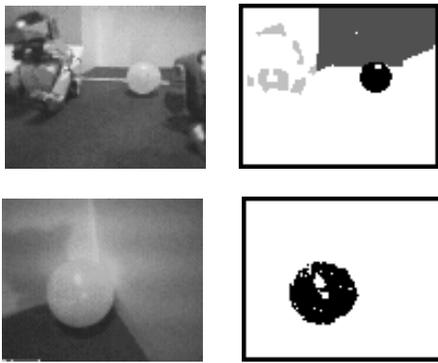


Figure 6. Segmentation results from the images in Fig. 2.

the “safest” pixels of each color have been labeled by the CDT, and are used as seeds. The right images show the regions found by our algorithm, together with their bounding boxes. These regions correctly correspond to the actual objects in the image. The only exception is that the blue patch on the ear of the left robot was not found from the top image since the CDT could not find any blue seed in that area.

Figure 6 shows the results obtained by applying our algorithm to the images in Fig. 2. In both cases, our algorithm correctly segments all the objects and correctly computes the region parameters. In the bottom image, the seeds for the ball region are the pixels shown in Fig. 2 (center) and the homogeneity criterion is the one used in Fig. 3 (center).

Our algorithm runs at frame rate using the limited on-board computational power of the ERS_210 (a MIPS 192MHz processor). Speed is achieved by the fact that our SRG only needs two passes through the image, and that threshold segmentation is done in hardware. The average time is 4 msec for full segmentation of eight colors in a 88×72 image. The maximum observed time has been 21 msec when a single object fills the image, since in this case many blobs have to be merged.

5. Conclusions

Threshold techniques for color segmentation are attractive because of their simplicity and the possibility to implement them in hardware, as it is the case in the robot platform considered here. For this reason, they are widely used in robotic applications, including the RoboCup domain. However, they suffer from two main problems: (i) tuning is difficult and makes the results sensitive to small changes in the lighting conditions: and (ii) they usually rely on the assumption that the colors of interest occupy rectangular areas in the color space, which is rarely the case. Our technique overcomes this problems by integrating (hardware) thresholding segmentation and seeded region growing (SRG).

Robustness is achieved by two means. First, thresholding is only used to determine a set of initial seeds. This means that we can use very strict thresholds corresponding to small and “safe” rectangular areas in the color space (cf. Figure 1). By doing so, we guarantee that the seeds are inside the intended regions even in the presence of noise. Second, SRG is only used to grow local blobs from several seeds inside the same region. The full regions are obtained by merging these blobs. Using SRG locally means that we can use a very strict homogeneity criterion, thus improving robustness with respect to blurred edges.

Although our technique has been developed having in mind the application needs of RoboCup, it can be used in other domains as well. For instance, we have successfully tested it in a pick-and-place manipulation task involving colored blocks laying on a green background under natural lighting conditions. In that application, threshold segmentation is performed in software.

References

- [1] R. Adams and L. Bischof. Seeded region growing. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 16, pages 641–647, 1994.
- [2] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS*, volume 3, pages 2061–2066, 2000.
- [3] J. Fan, D. Yau, A. Elmagarmid, and W. Aref. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *IEEE Trans. on Image Processing*, 10(10), 2001.
- [4] B. Henst, D. Ibbotson, S. Pham, and C. Sammut. The UNSW united 2000 sony legged robot software system. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 64–75. Springer-Verlag, 2001.
- [5] N. Ikonomakis, K. Plataniotis, and A. Venetsanopoulos. Un-supervised seed determination for a region-based color image segmentation scheme. In *IEEE Int. Conf. Image Processing*, pages 537–540, 2000.
- [6] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, 1997.
- [7] P. L. Palmer, H. Dabis, and J. Kittler. Performance measure for boundary detection algorithms. *Computer Vision and Image Understanding*, 63(3):476–494, 1996.
- [8] P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2):233–260, 1988.
- [9] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing Analysis and Machine Vision*. International Thomson Computer Press, 1996.
- [10] Team Sweden. Team homepage and online publications. <http://www.aass.oru.se/Agora/RoboCup/>.
- [11] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE T. Pattern Analysis and Machine Intell.*, 13(6):583–598, 1991.