Low Latency Color Segmentation on Embedded Real-Time Systems

Dirk Stichling, Bernd Kleinjohann
C-LAB, Fürstenallee 11, D-33102 Paderborn, Germany

Abstract: This paper presents a color segmentation algorithm for embedded real-time systems with a special focus on latencies. The algorithm is part of a Hardware-Software-System that realizes fast reactions on visual stimuli in highly dynamic environments. There is furthermore the constraint to use low-cost hardware to build the system. Our system is implemented on a RoboCup middle size league prototype robot.

Key words: computer vision, color segmentation, embedded systems, low latency, RoboCup

1. INTRODUCTION

Today most low-cost robotic systems navigate very slowly, e.g if you watch a RoboCup game you will see a lot of slowly moving robots. There are two reasons for this: At first, mostly commercially available platforms are used. Because of their widespread usage they are built to be safe in a lot of different areas, thus they are comparatively slow. The second reason is the system architecture. Often PCs are used to control the robot. There are some major drawbacks on PCs because they are not designed to be used as an embedded real-time system. They lack special hardware to control all sensors and actors and the operating system often has no real-time capabilities. There often arise high latencies between an event and the reaction of the system.

We are building a robot which is capable of navigating safely through a RoboCup field with a speed of up to 20 km/h. But there is one major constraint: usage of low-cost hardware for building such the robot.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35599-3_29

B. Kleinjohann et al. (eds.), *Design and Analysis of Distributed Embedded Systems*

2. APPROACH

Every robotic system consists of three major parts: The *perception*, the *processing* and the *manipulation*. For each of these parts we use microcontrollers which are specialized on their tasks. The only sensors we use are consumer electronic analog video cameras because of their low price. We developed a color image segmentation algorithm especially for embedded real-time systems. During the last years a lot of color image segmentation algorithms have been proposed [1], but only little attention has been paid to minimizing latency.

Thus we built a system with only vision-based perception and a behavior-based control system which is directly coupled with the perception and the actors. A good overview of behavior-based robotics is given in [4].

This paper presents a color segmentation algorithm as part of the vision-based perception of our system. The focus is on minimizing the latency of the vision part. The algorithm meets the following requirements which suffice for the RoboCup scenario:

- It has to be able to analyze 25 frames per second
- There are no predefined colors for the segmentation process.
- The shape of the objects is not relevant. The algorithm should only yield the position and size of the objects.
- It has to cope with the YUV color-space because most image hardware provide the data using this color space. Detailed information about this color space is given on [5].

3. RELATED WORK

Somehow similar to our work is the Polly System built in 1993 at the MIT [6]. One design goal was to use off-the-shelf low-cost hardware. So the only sensor input was one analog camera. They realized a task-based vision system capable of analyzing a 64x48 grey-scale image at 15 hertz. The control system had different layers and the low-level navigation also used behavior-based programming techniques. Newer robots have been built at the MIT based on the Polly System using the Cheap Vision Hardware [7]. Our system uses 352x288 pixel color images as input which permits higher flexibility and also analyzes the images with 25 Hz.

At CMU in Pittsburgh a fast color segmentation algorithm and implementation for robots called CMVision has been developed in 2000 [8]. It can discriminate 32 different colors organized in color cubes which has to be defined statically. So the algorithm is very sensible to changing lightning

conditions. Compared to CMVision and other fast color segmentation algorithms our color segmentation algorithm does not use any predefined colors.

An algorithm for fast and robust segmentation of natural color scenes has been proposed in [9]. The algorithm called CSC (Color Structure Code) is based on hierarchical region-growing on a special hexagonal topology. The system is applied in two reactive applications from the field of autonomous vehicle guidance.

Another very low-cost active vision system for embedded microcontrollers has been proposed in 1997 [10]. It uses a special frame grabber chip in conjuction with a 8-bit microcontroller.

4. METHODOLOGY

Most computer vision algorithms are divided into different steps which are evaluated sequentially. This has two major drawbacks:

1. *High latency:* Because the second step of an algorithm does not start before completion of the first step (which implies that the full input image has already been transmitted to the main memory) the latency is at least the sum of one image's transmission time and the run-times of all steps (except the first one because the first step may run in parallel to the image transmission).
2. *High memory consumption:* Because each step is evaluated individually the complete data that each step produces has to be stored in memory.

Thus we propose a methodology that overcomes these drawbacks. The main idea is to perform the image analysis during the image transmission instead of doing it afterwards. To realize this we introduce two techniques:

1. *Linear Processing:* The camera's input image is transmitted to the main memory top-down line-by-line. Therefore each step of the computer vision algorithm has to work line-based only depending on data of lines already processed.
2. *Incremental Concurrency:* Instead of evaluating the steps of the algorithm sequentially for every *input image* the steps are evaluated for every *line* of the input image. That is possible because the Linear-Processing technique takes care of the right data dependencies.

Using these techniques the latency of the implementation may be lowered dramatically. If the processor is fast enough the latency is as low as one image's transmission time plus the run-times of all steps (except the first one) only for the last image line.

Because the individual steps are processed for every image line the data for only one line has to be turned over to the next step. In some cases a step only needs the data of the actual line, thus there is no need to store the data of other lines which minimizes the overall memory consumption.

Section 5 depicts a color segmentation algorithm which was designed using these techniques *Linear Processing* and *Incremental Concurrency*.

5. COLOR SEGMENTATION ALGORITHM

Using the techniques described in section 4 we designed a color segmentation algorithm for embedded systems. The algorithm is divided into two individual steps:

1. *Line-based Region-Growing:* In every scan line of the image regions with similar color are grown from left to right.
2. *Region-Merging:* The regions created in the first step are merged if they are similar in color and spatial near to each other.

Region-growing and region-merging techniques are not new to computer based image segmentation [11,12]. What's new here is the usage of these techniques in combination with the techniques proposed in section 4.

5.1 Data Structures

On embedded devices you often don't have much computation power. Thus you need to use data structures which are easy to compute. We use moments up to second order as region descriptors because the operations for adding pixels to a region or merging two regions are computationally fast and they are sufficient to describe position and size of the regions. Moments are often used in computer vision algorithms to describe image objects [2,3].

The definition of the moment $M(p,q)$ of a region is as follows:

$$M(p,q) = \sum_{(x,y) \in region} x^p y^q$$

We use the moments $M(0,0)$, $M(1,0)$ and $M(1,1)$. They describe the number of pixels and the center of the region and are computational simple.

Additionally we need the central moments $C(p,q)$ of a region with (c_x,c_y) being the center of the region:

$$C(p,q) = \sum_{(x,y)\in region} (x-c_x)^p (y-c_y)^q$$

We use the central moments $C(1,1)$, $C(2,0)$ and $C(0,2)$. For easier reading we use the following notation:

$$a = (M0,0), c_x = \frac{M(1,0)}{M(0,0)}, c_y = \frac{M(0,1)}{M(0,0)}, v_x = C(2,0), v_y = C(0,2), v_{xy} = C(1,1)$$

Using these six moments every region is represented as an ellipse. The representation as an ellipse is only for visualization and is not used for the algorithm because ellipses are computationally difficult to handle.

The following computations show that two regions are merged very fastly. The data of the two regions to be merged are notated with ^ and ~.

$$a = \hat{a} + \tilde{a}, c_x = \frac{\hat{a}\hat{c}_x + \tilde{a}\tilde{c}_x}{a}, c_y = \frac{\hat{a}\hat{c}_y + \tilde{a}\tilde{c}_y}{a},$$

$$v_x = \hat{v}_x + \hat{a}(c_x - \hat{c}_x)^2 + \tilde{v}_x + \tilde{a}(c_x - \tilde{c}_x)^2,$$

$$v_y = \hat{v}_y + \hat{a}(c_y - \hat{c}_y)^2 + \tilde{v}_y + \tilde{a}(c_y - \tilde{c}_y)^2,$$

$$v_{xy} = \hat{v}_{xy} + \hat{a}(c_x - \hat{c}_x)(c_y - \hat{c}_y) + \tilde{v}_{xy} + \tilde{a}(c_x - \tilde{c}_x)(c_y - \tilde{c}_y),$$

The computations for the merging process take constant time and do not depend on the number of pixels of the two regions. The same holds for adding one pixel to a region ($\hat{a}=1$). This is important for the WCET of this algorithm as described in section 5.4.

Beside these six moments the average color value using the YUV color space of all pixels of the region completes the data structure of a region. The regions are managed using a connected list. Inside the list the regions are sorted by the y-coordinate of the uppermost pixel of the region. The major advantage of this kind of sorting is that during the region-growing step the regions are automatically created in that order und during the region-merging step this ordering speeds up the search process dramatically as explained in section 5.3.

5.2 Region-Growing

As describe before the region growing step is executed individually for every line and from left to right. Vision hardware often provides pixels using the YUV color space. Therefore the region-growing step only uses this color space to avoid time consuming color space conversion of all pixels.

All achromatic pixels are skipped because only colored regions are relevant for our scenarios (It is simple to extend the algorithm to detect non-colored regions as well). We discriminate chromatic and achromatic pixels in a way similar as proposed in [13]. Achromatic pixels are those which are too dark, too light or are nearly grey.

When finding the first colored pixel a new region is created. Then the next pixel to the right is visited and a decision is made whether the color is similar to the color of the actual region. If it is similar the pixel will be added to the region otherwise the region is finished and the algorithm skips to the next pixel.

To have a computational simple decision whether two colors are similar we simply use the euclidian distance of the two colors inside the YUV color space and define a threshold for similarity.

All lines are processed this way which results in a representation as shown in figure 1. All regions created during this step have height one because the y-coordinate of the pixels is always the same. This implies that v_y and v_{xy} is 0 and v_x only depends on a and is calculated using a lookup-table. Thus all moments are calculated very fastly. The number of pixels is needed to get a. The sum of all x-coordinates of the pixels and one division has to be done to calculate c_x. c_y is equal to the actual line number.

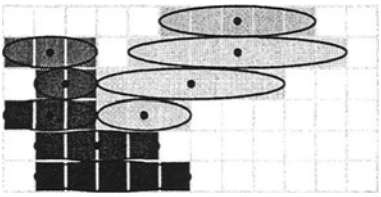

Figure 1. Representation after the region-growing step

This region-growing step handles each input line individually without any data dependability to other input lines. Therefore the requirements of the techniques introduces in section 4 are fulfilled. Because there is no data dependability to other input lines there is no need to store the whole input image in memory if the algorithm is in sync with the image input hardware.

5.3 Region-Merging

After the region growing step we need to merge the created regions. We have to compare each region with every other region whether they are spatial near to each other and similar in color.

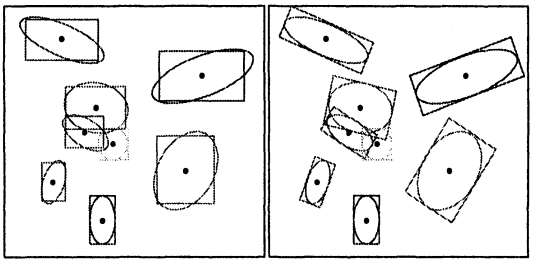

Figure 2. Possible approximations of an ellipse

How can we define *spatial near* using our data structure. The moments can be visualized using ellipses, thus *spatial near* could mean overlapping ellipses. Those calculations are computationally expensive because you need a lot of trigonometrical operations and thus you can't do it on low-cost embedded devices. Figure 2 shows two possible approximations of an ellipse. On the left figure you see an approximation using axis-parallel rectangles and on the right figure rectangles which are parallel to the main axis of the ellipses. The former is the easiest one in respect to computation time but also the most inaccurate one. The approximation can be provided using a lookup-table.

We also had to compare the colors of two regions. During the region growing step we use the YUV color space to decide whether pixels are similar in color or not. The YUV color space is not very suitable for comparing the colors of the regions because here higher euclidian distances in the color space are treated as similar. That's why a color space like HSI is more suitable. So when comparing two regions we calculate the euclidian distance of the colors and use a threshold to decide whether to merge or not.

A problem is the time of the search process because in the simplest case every newly created region has to be compared with every other region. So with n regions you'll get a time cost of $O(n^2)$. That's why we sort the regions spatially using a connected list.

For every region in the list the algorithm follows the list backwards until the first region which lowermost pixel is higher than the uppermost pixel of the actually processed region. Using this technique the processing time of the region-merging part is shorten dramatically.

This region-merging step only needs the data of the line-based region-growing step of the actual image line and the previously created regions. Therefore this step complies to the requirements introduced in 4. Figure 3 shows an example image with the segmentation result.

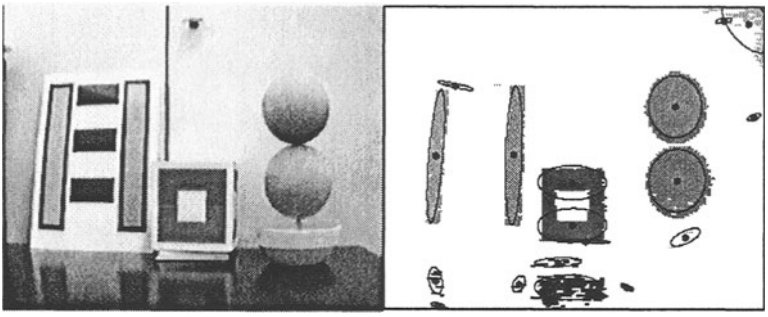

Figure 3. Example image and color segmentation

5.4 WCET-Analysis

For a lot of applications real-time capabilities are needed (e.g. to hold the image analysis in sync with the input image hardware). Therefore the Worst-Case-Execution-Times (WCET) must be known *a priori*.

A WCET-analysis is possible for the line-based region-growing step because all of the following computations take constant time and the number of pixels per line is constant and known a priori:

1. Decision whether a pixel's color is similar to the color of a region (Euclidian distance)
2. Adding a pixel to a region's data structure (shown in 5.1)
3. Creation of a new region (Initialization of data instance); the maximum number of regions in one line is known a priori and can be shorten introducing a minimum length of newly created regions.

The WCET-analysis of the region-merging step is more difficult. As shown in section 5.1 the computation of merging two regions is independent of the regions' data and thus the computation time is constant. The time to compare two regions whether they are spatial near and similar in color is constant and known a priori as well. But the number of comparisons is not known a priori because it depends on the number of regions as already shown in section 5.3. A maximum number of regions can be given because a maximum number of regions for every line is known and the number of images lines is also known. But this estimation is a lot higher than the number of comparisons normally needed in average input images. Therefore

the region-merging step has to be optimized in the future (e.g. by using different data structures) to yield a better WCET.

6. IMPLEMENTATION

We implemented the algorithm using a Philips TriMedia 1100 based microcontroller board running at 100MHz. While the image hardware is sending the input image to the main memory the CPU already does the image analysis of the first lines. For the merging step we use the axis-parallel approximations of the ellipses because the operations for the comparisons are fast and tests have shown that the approximation is sufficient for most applications. The implementation needs about 500kByte RAM at run-time. During several tests we showed that in most cases the system was capable of processing 25 frames per second and the delay time between start of the analog transmission of an image and the segmentation results was less than 40ms. Table 1 shows a comparison of our algorithm with the CMVision [8] and the CSC [9]. Latencies were not given in those papers.

Table1: Comparison with other color segmentation implementations

	Image Size	Framerate	System
CMVision [8]	320x240	30 fps	30% of 350MHz Pentium II
CSC [9]	256x256	5 fps	100% of 167MHz SPARC CPU
Our Implementation	352x288	25 fps	100MHz TM1100

7. CONCLUSION

We have shown that it is possible to build a low-cost embedded vision system which is capable of doing low latency color segmentation. Low latency vision systems are important to build very fast acting and reacting robots e.g. in the RoboCup scenario.

Color segmentation is one important part of a vision system but normally that is not sufficient. So we plan to integrate edge-detection to our system in the near future.

The color segmentation algorithm will be connected to a special behavior-based system which is also designed be have low-latencies. Crucial is the overall response time of a system therefore the perception, the processing

and the manipulation have to have low latencies. Only pure computing power is not enough to build fast reacting systems.

8. REFERENCES

[1] W. Skarbek and A. Koschan, "Colour image segmentation - a survey" Tech. Rep. 94-32, Technical University Berlin, 1994.

[2] C.-H. Teh and R. T. Chin, "On image analysis by the methods of moments" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-10, pp. 496-513, July 1988.

[3] R. J. Prokop and A. P. Reeves, "A survey of moment-based techniques for unoccluded object representation and recognition" *Computer Vision, Graphics, and Image Processing. Graphical Models and Image Processing*, vol. 54, pp. 438-460, Sept. 1992.

[4] R. C. Arkin, *Behavior-Based Robotics*. Cambridge: MIT Press, 1998.

[5] C. Poyton, "Frequently asked questions about color." Website: http://www.inforamp.net/~poynton.

[6] I. Horswill, "Polly: A vision-based artificial agent" in *Proceedings of the 11th National Conference on Artificial Intelligence*, (Menlo Park, CA, USA), pp. 824-829, AAAI Press, 1993.

[7] L. Lorigo, R. Brooks, and W. Grimson, "Visually guided obstacle avoidance in unstructured environments" in *Proc. IROS97*, pp. 373-379, 1997.

[8] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots" in *IEEE/RSJ International Conf. on Intelligent Robots and Systems*, IEEE, 2000.

[9] V. Rehrmann and L. Priese, "Fast and robust segmentation of natural color scenes" in *Computer Vision - ACCV 98* (R. Chin and T.-C. Pong, eds.), vol. I, (Hong Kong, China), pp. 598-606, jan 1998.

[10] G. Wyeth, "Implementing active vision in embedded systems" in *Proc. of Mechatronics and Machine Vision in Practice 4*, pp. 240-245, IEEE Computer Society Press, 1997.

[11] A. Tremeau and N. Borel, "A region growing and merging algorithm to color segmentation", *Pattern Recognition*, vol. 30, no. 7, pp. 1191-1203, 1987.

[12] F. Moscheni and F. Dufaux, "Region merging based on robust statistical testing" in *SPIE Proc. VCIP'96*, (Orlando, Florida), March 1996.

[13] N. Ikonomakis, K. Plataniotis, and A. Venetsanopoulos, "A region-based color image segmentation scheme" in *Proceedings of Electrical Imaging '99, vol. 3653 of SPIE*, (San Jose, California), pp. 1202-1209, Jan 1999.