IMAGE SEGMENTATION FOR OBJECT DETECTION ON A DEEPLY EMBEDDED MINIATURE ROBOT

Alexander Jungmann¹, Thomas Schierbaum² and Bernd Kleinjohann¹

¹Cooperative Computing & Communication Laboratory, University of Paderborn, Paderborn, Germany
²Product Engineering, Heinz Nixdorf Institute, University of Paderborn, Paderborn, Germany

Keywords: Image Segmentation, Run-length Encoding, Moments, Robotics.

Abstract: In this paper, an image segmentation approach for object detection on the miniature robot BeBot - a deeply embedded system - is presented. In order to enable the robot to detect and identify objects in its environment by means of its camera, an efficient image segmentation approach was developed. The fundamental algorithm bases on the region growing and region merging concept and identifies homogeneous regions consisting of adjacent pixels with similar color. By internally representing a contiguous block of pixels in terms of run-lengths, the computational effort of both the region growing and the region merging operation is minimized. Finally, for subsequent object detection processes, a region is efficiently translated into a statistically feature representation based on discretized moments.

1 INTRODUCTION

Embedded systems are usually very restricted with respect to their memory and computational power. The miniature robot BeBot (see Figure 1), which combines an ARM Cortex-A8 600MHz processor, 256MB main memory and a small camera in a 9x9cm chassis (Herbrechtsmeier et al., 2009), is a mobile representative of an embedded system with such restrictions. In addition, it is able to explore its surroundings by means of its differential chain drive. However, for being able to act autonomously, the robot has to perceive its environment by means of its camera. For this purpose, an efficient image segmentation approach that takes the mentioned restrictions into account was developed and is presented within the scope of this paper. Object detection mechanisms are not content of this particular work though.

![Figure 1: Miniature robot BeBot, (a) with and (b) without light guide, enabling the robot to express its internal state.](image-url)

This paper is organized as follows. The fundamental principles of the realized segmentation process are described in Section 2. Section 3 deals with the external feature representation for subsequent object detection processes. Results of the segmentation algorithm running on a BeBot are content of Section 4. The paper finally concludes with Section 5.

2 SEGMENTATION APPROACH

The basic idea of image segmentation is the identification of contiguous blocks of pixels, that are homogeneous with respect to a pre-defined criterion. By doing so, the pixel-based visual information is abstracted in order to get a reduced data representation, which is more convenient on the one hand and less computational expensive on the other hand. Regarding the computational power of the BeBots, object detection based on raw pixel data is not feasible at all.

2.1 Color as Criterion of Homogeneity

In the context of this paper, the criterion for constructing homogeneous regions is based on color information. Since the camera of the BeBot delivers YUV images, a simple but very efficient heuristic $H_{yuv}$, which is based on the Manhattan distance, is applied in or-
order to decide, whether two color tuples \((y_1, u_1, v_1)\) and \((y_2, u_2, v_2)\) reside in a pre-defined neighborhood within the three dimensional YUV color space:

\[
H_{yuv} = H((y_1, u_1, v_1), (y_2, u_2, v_2)) =
\begin{cases}
1 & \text{if } |y_1 - y_2| \leq c_1 \\
|u_1 - u_2| + |v_1 - v_2| \leq c_2, \\
0 & \text{else.}
\end{cases}
\]

By separating the luma value \(Y\) from the chrominance values \(U\) and \(V\), the different components can be independently weighted by means of the two parameters \(c_1\) and \(c_2\).

### 2.2 Internal Data Representation

For the internal representation of the regions during the segmentation process, the Run-Length Encoding concept is incorporated: sequences of adjacent pixels are compactly encoded as so called run-lengths (or runs), whereas a single run is defined in terms of three integer values:

\[
run_i = \langle(x_i, y_i), l_i \rangle,
\]

with \((x_i, y_i)\) being the coordinates of the starting pixel, whereas \(l_i\) represents the number of adjacent pixels within the same row and therefore denotes the length of a single run. Furthermore, an entire region \(R\) may consist of a sequence of \(n\) adjacent runs:

\[
R = \{(x_1, y_1), l_1\}, \{(x_2, y_2), l_2\}, \ldots, \{(x_n, y_n), l_n\}.
\]

The computational effort for both the region growing as well as the region merging operation (cf. Section 2.3) is minimal. While adding a pixel to a region is nothing but incrementing the length \(l_i\) of the associated run \(run_i\), merging of two regions is realized by simply appending the sequence of runs of the first region to the sequence of runs of the second region.

### 2.3 Basic Algorithm

The basic segmentation process is depicted in Algorithm 1. The main loop (lines 3-25) iterates row by row over the whole image, starting at the topmost row. The inner iteration loop (lines 4-24) processes each pixel within a row once, starting at the leftmost pixel position. After identifying the left adjacent run \(run_{left}\) as well as its associated region \(R_{left}\), heuristic \(H_{yuv}\) (1) is applied in order to check if the colors of the current pixel and \(R_{left}\) are similar (line 7). If so, the region growing step takes place by adding the current pixel

---

**Algorithm 1: Image Segmentation Algorithm.**

1. \(image\) = latest camera image
2. \(regions = \{\emptyset\}\) // set of identified regions
3. for all \(row \in image\) do
   4. for all \(pixel \in row\) do
      5. \(run_{left} = \text{left adjacent run}\)
      6. \(R_{left} = \text{region}(run_{left})\)
      7. if \(H(yuv(\text{pixel}), yuv(R_{left}))\) then
         8. // region growing
         9. \(\text{add}(run_{left}, \text{pixel})\)
         10. continue with next pixel
      else
         11. \(regions_{top} = \text{top adjacent regions}\)
         12. for all \(R_{top} \in regions_{top}\) do
            13. if \(H(yuv(R_{left}), yuv(R_{top}))\) then
               14. // region merging
               15. \(\text{merge}(R_{left}, R_{top})\)
               16. \(\text{remove}(\text{regions}, R_{top})\)
            end if
         end for
         17. \(\text{new}_{run} = \text{new run}(\text{pixel})\)
         18. \(R_{new} = \text{new region}(\text{run}_{new})\)
         19. \(\text{add}(\text{regions}, R_{new})\)
      end if
   20. end for
21. end for
22. return \(\text{regions}\)

---

pixel to the left adjacent run \(run_{left}\) (length of \(run_{left}\) is incremented by value 1) and updating the associated region \(R_{left}\) with respect to its average color. Afterwards, the algorithm continues with the next pixel of the row (line 10).

If heuristic \(H_{yuv}\) fails, the left adjacent run \(run_{left}\) is considered to be completed. The algorithm proceeds with its region merging mechanism. For this purpose, all regions bordering run \(run_{left}\) at the top are identified. Subsequently, another iteration loop tries to identify every region \(R_{top}\) within \(Regions_{top}\) that can be merged with the \(run_{left}\) associated region \(R_{left}\) (lines 13-19) by again applying heuristic \(H_{yuv}\). If \(H_{yuv}\) succeeds for two regions \(R_{top}\) and \(R_{left}\), the regions are merged by appending all runs of \(R_{top}\) to \(R_{left}\). Furthermore, region \(R_{left}\) is updated with respect to its average color, whereas region \(R_{top}\) is completely discarded by removing it from the set of heretofore identified regions.

Independent of the region merging step, a new run \(run_{new}\) with length 1 and with the current pixel as its starting position as well as a new region \(R_{new}\) with \(run_{new}\) as its first run are allocated. Finally, \(R_{new}\) is added to the set of heretofore identified regions.
3 FEATURE DESCRIPTION

By interpreting a region’s associated pixels as a two-dimensional Gaussian distribution in the image plane, a region can be implicitly described by means of statistical parameters, namely the two mean values $m_x$ and $m_y$, the two variances $\sigma_x^2$ and $\sigma_y^2$, and the covariance $\sigma_{xy}$. A generalization of these specific parameters are the statistical moments, which can be directly applied in our context in discretized form (Hu, 1962): the two mean values correspond to the two moments of first order ($m_{10}$ and $m_{01}$), whereas the two variances and the covariance correspond to the centralized (or central) moments of second order ($\mu_{20}$, $\mu_{02}$ and $\mu_{11}$). In addition, the mass of a region is equivalent to the moment of zeroth order ($m_{00}$).

Since the central moments can be directly derived from the moments through second order (Hu, 1962), the basic feature descriptor for representing an extracted region is given by the following set $M$ of moments:

$$M = \{m_{pq} | p + q \leq 2\}$$

(4)

For efficiently computing the required five moments, the runs of a region can be directly used by applying the Delta “$\delta$” Method (Zakaria et al., 1987). In this context, $S_1$ and $S_2$ are defined as follows:

$$S_1_i = \sum_{k=0}^{\delta_i-1} k = \frac{\delta_i^2 - \delta_i}{2}$$

(5)

$$S_2_i = \sum_{k=0}^{\delta_i-1} k^2 = \frac{\delta_i^3}{3} - \frac{\delta_i^2}{2} + \frac{\delta_i}{6}$$

with $\delta_i$ corresponding to the length $l_i$ of a single run $r_i$. The required geometric moments $m_{00}$, $m_{10}$, $m_{01}$, $m_{11}$, $m_{20}$, and $m_{21}$ of run $r_i$ can then be computed in the following way:

$$m_{00} = \delta_i$$

$$m_{01} = \delta_i \cdot y_i$$

$$m_{02} = \delta_i \cdot y_i^2$$

$$m_{10} = \delta_i \cdot x_i + S_1_i$$

$$m_{11} = y_i \cdot [\delta_i \cdot x_i + S_1_i] = y_i \cdot m_{10}$$

$$m_{20} = \delta_i \cdot x_i^2 + 2 \cdot S_1_i \cdot x_i + S_2_i$$

(6)

Finally, the moments of an entire region $R$ correspond to the sums of the particular moments of all associated $n$ runs:

$$M = \{m_{pq} | m_{pq} = \sum_{i=1}^{n} m_{pq_i} \wedge p + q \leq 2\}$$

(7)

For representing a feature in a more explicit manner, additional geometric attributes can be derived from $M$ and the associated central moments (Teague, 1980; Prokop and Reeves, 1992). In this context, the coordinates $(\bar{x}, \bar{y})$ of the center of mass of an extracted feature in the image plane are defined by the moments of zeroth and first order:

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

(8)

Furthermore, due to the statistical interpretation in terms of a Gaussian distribution, a feature is equivalent to an elliptical disk with constant intensity, having definite size, orientation and eccentricity and being centered at the origin of the image plane. The lengths of its major and minor axes as well as its angle of inclination can be computed with the use of the associated central moments (Teague, 1980). By combining both the center of mass and the elliptical disk, a feature can be geometrically described in terms of an ellipse that is located at the center of mass of the respective feature (cf. Figure 3).

4 RESULTS

The image segmentation approach was implemented in C/C++ and successfully applied to the BeBot. While capturing a YUV image of size 320x240 pixels, the algorithm enables us to process at least 10 entire images per second. Regarding the load factor of the BeBot during our experiments, the overall performance of the segmentation algorithm heavily depends on the number of different regions that are constructed, which in turn depends on the values of the calibration parameters in combination with the homogeneity of the original image.

Figure 2 shows different segmentation results with respect to different values of the calibration parameters $c_1$ and $c_2$ of heuristic $H_{uv}$. Whereas the goal of the algorithm was to clearly extract the colored objects (four balls and one rectangular block), we constructed the background (the horizon) as inhomogeneous as possible (see Figure 2(a)). From Figure 2(c) to Figure 2(d), only parameter $c_1$ for modifying the influence of the luma component $Y$ was changed. By increasing $c_1$, the segmented image becomes more and more homogeneous in comparison to Figure 2(b). The same holds true when the second parameter $c_2$ of heuristic $H_{uv}$ is exclusively modified (Figure 2(e) to Figure 2(f)). However, when comparing Figure 2(c) and Figure 2(f), changing the allowed range of luma similarity ($c_1$) seems to have a greater influence than changing the allowed range of chrominance similarity ($c_2$). Keeping that in mind, Figure 2(g) shows a great overall result with respect to the homogeneity of the
segmented image. In this context, the value of parameter $c_1$ was set to 30, whereas the value of parameter $c_2$ was set to a slightly lower value of 20.

We tried to relax the restriction of color similarity even more by simultaneously increasing both parameters $c_1$ and $c_2$. At some point, the algorithm begins to merge color values, that obviously are not similar at all, whereas regions, which are of similar color but differ with respect to their intensity (brightness), are not merged (see Figure 2(h)). This issue can be traced back to the characteristics of the YUV color space.

Last but not least, Figure 3 exemplary depicts the external region representation in terms of equivalent ellipses, which are located at the centers of mass of the extracted features.

5 CONCLUSIONS

In this paper, an efficient color-based image segmen-