Fast Image Segmentation, Object Recognition and Localization in a RoboCup Scenario

Thorsten Bandlow, Michael Klupsch, Robert Hanek, Thorsten Schmitt

Forschungsgruppe Bildverstehen (FG BV)
Technische Universität München, Germany
{bandlow,klupsch,hanek,schmittt}@in.tum.de
http://www9.in.tum.de/research/mobile_robots/robocup/

Abstract. This paper presents the vision system of the robot soccer team *Agilo RoboCuppers*[1] – the RoboCup team of the image understanding group (FG BV) at the Technische Universität München.
We present a fast and robust color classification method yielding significant regions in the image. The boundaries between adjacent regions are used to localize objects like the ball or other robots on the field. Furthermore for each player the free motion space is determined and its position and orientation on the field is estimated. All this is done completely vision based, without any additional sensors.

1 Introduction

The vision module is a key part of our robot soccer system described elaborately in [1, 5]. Given a video stream, the vision module has to recognize relevant objects in the surrounding world and provide their positions on the field to other modules. Each robot is only equipped with a standard PC based on a single Pentium 200 MHz processor. Consequently, we have to focus on efficient and computationally inexpensive algorithms to serve the real time constraints. This is done with the help of the image processing tool HALCON (formerly known as HORUS [3]). This tool provides efficient functions for accessing, processing and analyzing image data, including framegrabber access and data management.

In general, the task of scene interpretation is a very difficult one. However, its complexity strongly depends on the context of a scene which has to be interpreted. In RoboCup, as it is currently defined, the appearance of relevant objects is well known. For their recognition, the strictly defined constraints of color and shape are saved in the model database and can be used. These constraints are matched with the extracted image features such as color regions and line segments. Figure 1 shows a data flow diagram of our vision module.

Besides recognizing relevant objects, further tasks of the image interpretation module are to localize the recognized objects and to perform the self-localization

[1] The name is derived from the Agilolfinger, which were the first Bavarian ruling dynasty in the 8th century, with Tassilo as its most famous representative.

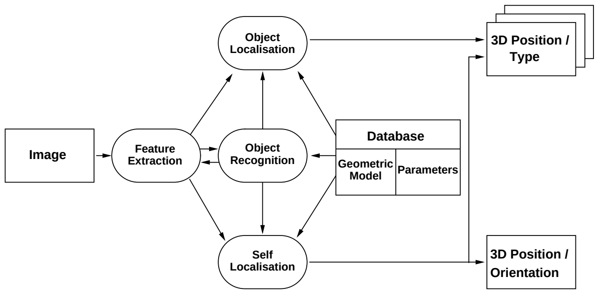


Fig. 1. Data flow diagram of the vision module.

of the robot on the field. For this the intrinsic camera parameters as well as the external ones relative to the robot were determined by calibration.

This paper is organized as follows: Section 2 describes the algorithms applied for the color based image segmentation and object extraction. In Section 3 the estimation of the position of relevant objects in the 3-D coordinate system of the observing robot is discussed. A video based self-localization algorithm using the field boundary lines is presented in Section 4. Section 5 shows the achieved results and, finally, a conclusion is given.

2 Color based Image Segmentation and Object Detection

This section describes how an YUV-image, as captured from the camera, is segmented into regions representing one color class each. The overall aim of this segmentation process is to extract one segment for each visible object of the current scene. Since all important objects of the RoboCup scenario have distinct and unique colors, color is the key feature used for object recognition.

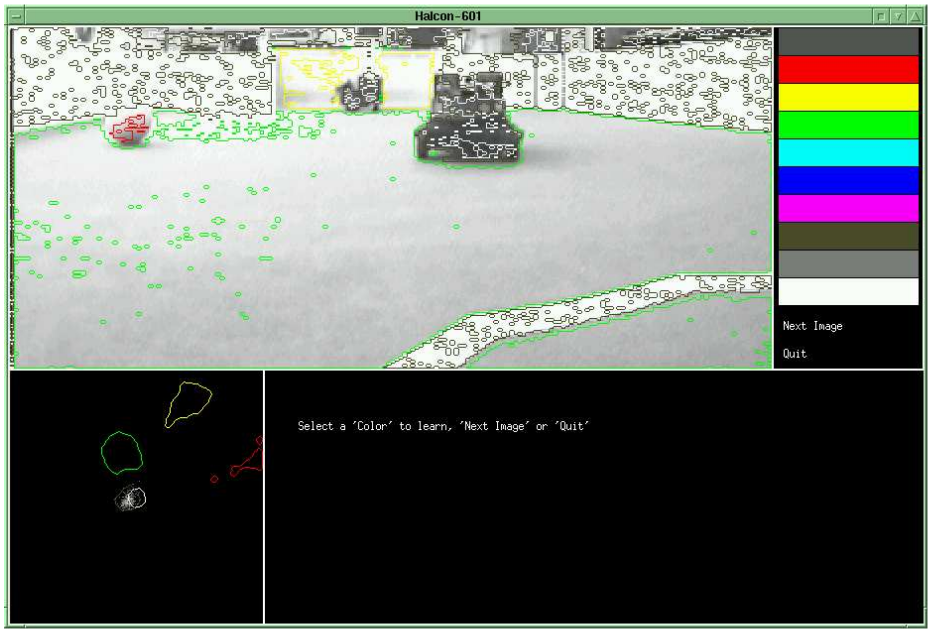
First, color regions are determined by the image processing module using a fast classification algorithm which assigns a color class label to each pixel (see Equation 1) according to its YUV-values.

$$\begin{aligned}(y,u,v) \longrightarrow \{&no\_color, black, white,\\ &green, blue, cyan,\\ &magenta, red, yellow\}\end{aligned} \tag{1}$$

Then various image processing operators are applied in order to determine the regions which contain the important objects of the RoboCup scenario, e.g. ball, lines, robots and goals.

2.1 Building a robust color Classifier

Previous RoboCup events have shown that the illumination conditions differ from field to field. Therefore, an adaption to the actual lighting conditions of the field is needed.



Select a 'Color' to learn, 'Next Image' or 'Quit'

Fig. 2. The color classifier training tool.

For building an adapted color classifier we have developed an interactive classification tool which supports supervised learning of the appropriate color classes (see Figure 2). The YUV-color space is used in order to distinguish different color classes. While the Y-channel heavily depends on the light intensity, the U- and V-values are relatively invariant regarding the brightness of the light. Since we use a camera with deactivated auto white balance the U- and V-values of a class have a quite small variance. The classification tool determines color clusters in the UV-plane according to test-samples and assigns color class labels to them. For achieving a robust classifier the training can be performed over several different images.

A color class label is assigned to a cluster in the YUV-color space as follows. An YUV-image is grabbed and the Y-channel is displayed. The user draws a region which contains only pixels of the same color class and assigns a label to it. First, the minimum and maximum brightness values according to the Y-channel are determined. Then a 2-D histogram for the U- and V-channels of the selected region is computed. This 2-D histogram is interpreted as a 256×256 image, and several threshold, closing and opening operators are applied to it. This eliminates faulty responses arising form color noise in the camera image, and provides a compact cluster representing one color class in the UV-plane.

This procedure is repeated for all color classes of interest incorporating different images. As a final result the color calibration tool saves a 256×256 color

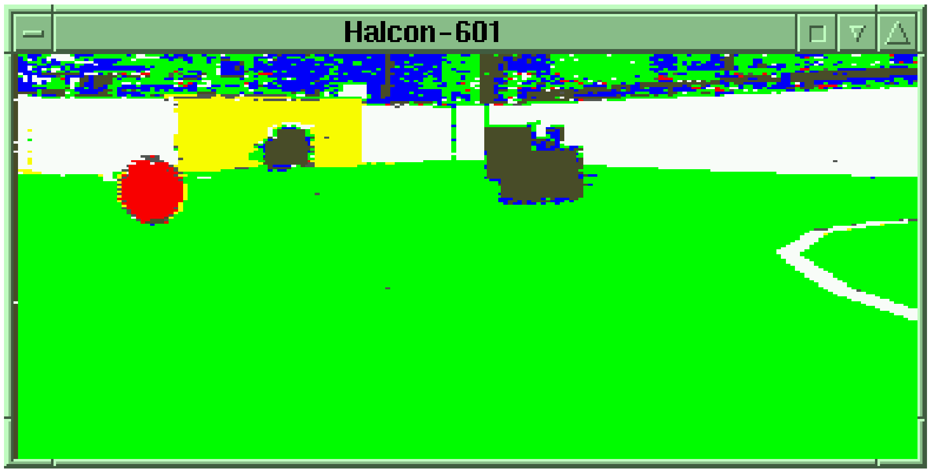


Fig. 3. The color segmented view of a robot.

class label image for the U- and V-channels and the minimum and maximum Y-values for each color class label.

2.2 Fast color based Segmentation

During the match the interface integrating the framegrabber device driver into the image processing library performs the color based segmentation. For each pixel the U- and V-values serve as indices into the previously determined color class label image. Each pixel is assigned to one region which corresponds to a specific color class, if its brightness is within the previously determined brightness interval for that color class. It is noteworthy that this procedure determines one region for each color class and that these regions need not to be interconnected and may be distributed all over the image.

After the whole image has been processed the framegrabber interface returns the determined color regions using a runlength encoding. This encoding is the default data structure used by the image processing system HALCON for storing regions. Consequently all HALCON operators can now easily and efficiently be applied to the pre-segmented image.

2.3 From Color Regions to Object Segments

Once the color regions are known, several different operators are applied in order to determine the independent segments which contain objects such as the ball, robots or goals.

First, different morphological operators to the color regions are applied, in order to remove small disturbances and make the regions more compact. Then we determine the single object segments performing a connected component analysis using the 8-neighborhood. Each of these segments is now regarded as an object hypothesis. Finally, objects can simply be recognized through the computation of certain shape features and verified by plausibility tests. For example, the ball segment can be detected using the shape feature anisometry, goals are discovered by their size, and lines by their length and their angle of inclination. Figure 3 shows a result of our segmentation and object recognition process. The yellow goal, the goal keeper, the ball and one opponent are clearly visible.

3 Object Localization

In this section we explain how the 2-D regions, introduced in the previous section, are used to estimate the position of relevant objects such as the ball, the goals or other robots in the 3-D coordinate system of the observing robot.

The cameras of the robots are calibrated using the approach presented in [6]. With the help of the intrinsic camera parameters pixel coordinates can be converted into image coordinates. Since we use a camera with a wide viewing angle, about 90 degree, for this conversion it is important to take the radial distortions of the lens into account even if not high precision is needed and high speed is desired.

3.1 Restriction on a 2-D Localization Problem

We assume that all relevant objects are located on the ground of the field, i.e. the distance between an object and the plane defined by the ground of the field is zero. Of course for a jumping ball this is not correct. However, such cases are quite rare in RoboCup games. The restriction onto the ground provides an one-to-one correspondence between a point on the ground plane $\mathbf{E}$ and its observation in the image plane.

In order to estimate the location of an opponent, for example, we determine in the corresponding segmentation result the lowest point $\mathbf{p}$. This point and the optical center of the camera define a viewing ray $\mathbf{r}$. Its intersection with the plane $\mathbf{E}$ of the field yields an estimate of the opponents maximum distance and its direction in camera coordinates, see Figure 4. The 3-D point $\mathbf{P}$ corresponding to the observed image point $\mathbf{p} = [\mathrm{p_x}, \mathrm{p_y}]^\mathrm{T}$ is given by

$$\mathbf{P} = \frac{\mathrm{h}}{-\mathrm{p_y}}[\mathrm{p_x}, \mathrm{p_y}, \mathrm{f}]^\mathrm{T} \tag{2}$$

Here the focal length of the camera is denoted by f and its height, the distance to the ground, by h. The point $\mathbf{P}$ given in camera coordinates can easily be expressed in the coordinate system of the robot, since the pose of the robot's camera is given in robot coordinates.

In general the 3-D point $\mathbf{P}$ corresponding to the observed image point $\mathbf{p}$ lies not exactly on $\mathbf{E}$. However for the robots taking part in the RoboCup competition the distance $d(\mathbf{P}, \mathbf{E})$ is in general small enough.

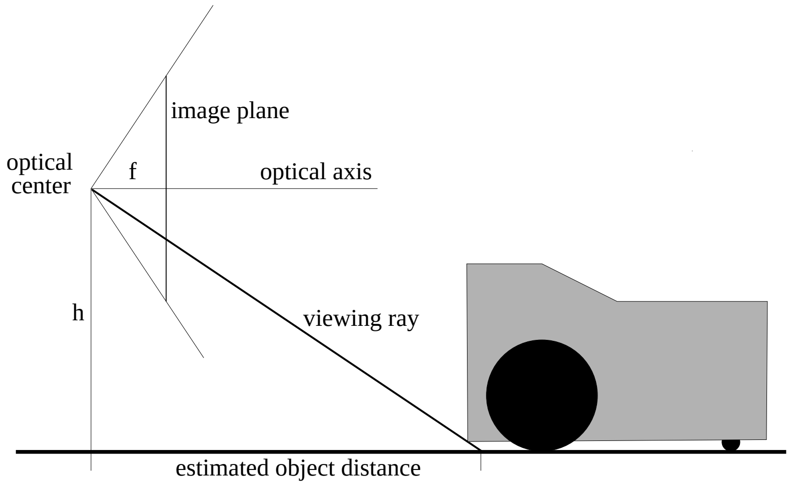


Fig. 4. An estimate of the object distance is given by the intersection of the viewing ray with the ground plane of the field.

The segmentation algorithm described in the previous section only classifies a pixel if its classification is quite sure. This means some pixels do not belong to any segment, especially at the border of two segments. Therefore, the distance computed as discussed above is an upper limit for the object distance. In order to obtain a lower limit we use the highest point $\mathbf{p}\prime$ under $\mathbf{p}$ classified as field. This point defines a second viewing ray $\mathbf{r}\prime$. Its intersection with the plane $\mathbf{E}$ yields a lower limit for the object's distance.

3.2 Using Shape Restrictions for the Localization of the Ball

Due to noise, reflections or other disturbances the segmentation process may provide more than one red segment as candidates for the ball. In order to decide if a segment actually corresponds to the ball we use several plausibility checks. For all hypothesis given by red segments we compute the distances. Using the ball's distance we determine the radius r of the ball's projection onto the image plane. A segment of size s_x in x-direction and size s_y in y-direction is rejected as ball hypothesis if the condition

$$(s_x < b_x r) \text{ or } (s_y < b_y r) \qquad (3)$$

holds. The parameter b_y is chosen to be smaller than b_x since due to reflections on the top of the ball this part quite often can not be classified as red. Furthermore we reject segments which cover a number of pixels which is too small in comparison with the calculated projection of the ball.

3.3 Free Motion Space

In order to navigate autonomously it is essential to know where the robot can move without collision. The localization approach described above yields the positions of objects localized with a relatively high probability. However, objects which could not get localized could still represent an obstacle. Therefore, we compute the free motion space independently of the object localization.

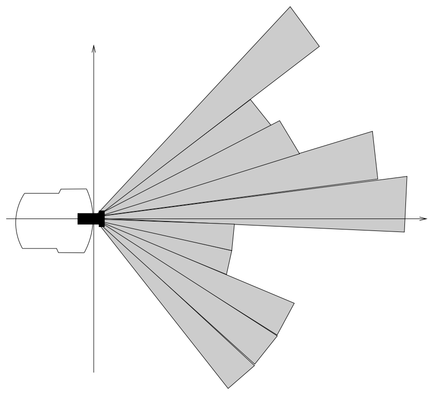


Fig. 5. Free motion space: The possible moving area of the robot is equally divided into sectors. To each of the sectors a maximal moving distance is associated, representing the space in which the robot can move without collision.

We divide the space of possible motion directions into sectors of equal angle as depicted in Figure 5. Each of these sectors on the field corresponds to a region in the image plane. Since the white lines of the field are no obstacles in contrary to the also white wall, we distinguish the white lines from the white wall by investigating the neighborhood of white regions. For determining the free motion space the green field regions and the white line regions are merged. Then morphological operations are applied in order to eliminate small artifacts. Since the ball can be moved by the robot the ball is also no real obstacle. Therefore, we compute for each sector the furthest point such that all closer points are either green or red, which means they are no obstacle. This point defines the length of a sector.

4 Self Localization

Obtaining the pose of the robots in the field coordinate system is a crucial term for the strategic planning of the robots, especially for actions, where several robots have to collaborate. Our self-localization algorithm is based on the boundaries of the field. They are easy to detect and allow a robust pose estimation. If a robot detects only one boundary line, the distance to this line and the robot's orientation can be adjusted. With two lines the robot can determine its absolute position and orientation. In the following sections we will discuss the feature extraction and pose estimation process.

4.1 Feature Extraction

The aim of the feature extraction process is to detect one or two lines of the field boundary, for which the relative 3-D coordinates are computed.

To detect the field boundaries only the border area between the field regions – green field and white lines – and the white wall regions is investigated. This area or region of interest can easily be achieved by a dilatation operation applied on both the wall and the field region followed by intersecting the two resulting regions.

Two different methods were implemented for ascertening the field boundaries. The first approach uses directly the field-wall-border region mentioned above. The skeleton is calculated and transformed into a set of contours $\mathcal{C}$. In the second method, a subpixel accurate edge filter is applied onto the Y-channel only within the previously determined region of interest, also calculating a set of contours. This method results in much more accurate contours but needs more computation time (approx. 30 ms vs. 5 ms). Both methods can be used alternatively.

The next steps remove camera distortions and approximate the contour segments with straight line segments. For this the following process is performed:

1. $\forall c_i \in \mathcal{C}$: Compute a regression line.
2. Filter lines by angle and length. Vertical lines and too short lines are discarded.
3. Join contours c_i and c_j which are collinear and closer than a maximum distance.

To achieve 3-D line segments we project the endpoints of the 2-D line segments onto the ground plane using the method described in Section 3.1. Collinear 3-D line segments are joint.

4.2 Obtaining Correspondences

For the self-localization we use a model of the field consisting of the four boundary lines. In order to estimate the pose we have to find correspondences between the 3-D model lines and the 3-D backprojection of their 2-D observations resulting from the method described in Section 4.1.

The two goals with their distinct colors are used to obtain the needed correspondences. If a 2-D line segment is adjacent to a goal segment then this line segment corresponds to the 3-D line next to the observed goal. This test is performed using a dilatation method described in [2]. If a second orthogonal 3-D line segment is given then this segment corresponds to a side line. The position of the goal segment and the 2-D line segment in the image defines whether the

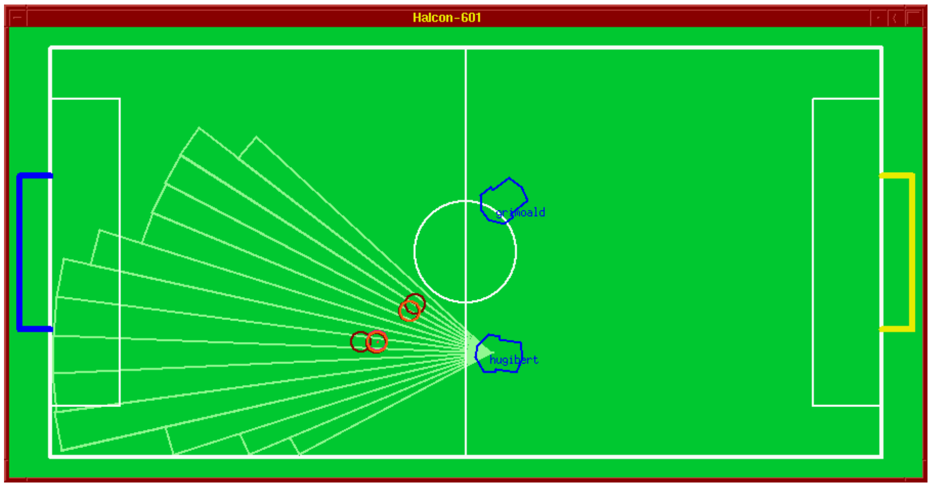


Fig. 6. The RoboCup monitor visualizes the actual positions of robots, ball hypotheses and free motion spaces.

line segment corresponds to the left or right side line. Due to noise the two back-projections are usually not exactly orthogonal. In this case we substitute the back-projections by two orthogonal lines having minimal distance to the original back-projected lines.

If no goal is extracted we can not establish correspondences as described above. In this case an absolute relocalization is not possible. Therefore, the pose of the robot at time t_i is predicted from the data at time t_{i-1} and the odometric data from the robot. We match the back-projected lines given in the robot coordinate system with the model line given in the world coordinate system such that the difference of the robot's new and predicted orientation is minimal. With this renewing method a correct match is performed, if the error of the orientation prediction is lower than 45 degree which holds most of the time.

4.3 Pose Estimation

Once the correspondences are given, we have to determine the robot's pose, such that the back-projected lines, given in robot coordinates, fit to the model lines, given in world coordinates.

The robot's pose has three degrees of freedom and is represented by a two-dimensional translation vector $\mathbf{T} := (\mathrm{T_x}, \mathrm{T_y})^{\mathrm{T}}$ and a rotation angle Φ. Since the feature extraction yields either zero, one or two line segments we have to distinguish three cases. In the quite rare case where no line segment is given no vision based update of the robot's pose is possible.

For a single line segment the problem is underdetermined. However, the rotation and one component of the translation can still be determined. The new orientation Φ_t and the previous orientation Φ_{t-1} are related by

$$\Phi_t = \Phi_{t-1} + \Delta\Phi_t \tag{4}$$

where $\Delta\Phi_t$ denotes the angle between the back-projected line and the corresponding model line. Furthermore the robot is able to update its distance to the observed line. Similar to the orientation, the relation between the new translation $\mathbf{T}_{\mathrm{t}}$ and the old translation $\mathbf{T}_{\mathrm{t}-1}$ is

$$\mathbf{T}_{\mathrm{t}} = \mathbf{T}_{\mathrm{t}-1} + \Delta\mathbf{T}_{\mathrm{t}}. \tag{5}$$

Here the vector $\Delta\mathbf{T}_{\mathrm{t}}$ is the distance vector between the model line and the back-projected line. Note, after the orientation is updated using Equation (4), these two lines are parallel and $\Delta\mathbf{T}_{\mathrm{t}}$ is usually not zero. The translation along the line can not be estimated from a single line. Therefore the component of $\mathbf{T}_{\mathrm{t}}$ parallel to the line is obtained from the previous translation vector $\mathbf{T}_{\mathrm{t}}$.

If two line correspondences are given, the orientation can be updated again using Equation (4). Since the two back-projected lines are forced to be orthogonal (see Section 4.2), the quantity $\Delta\Phi_t$ is the same for both lines. With two lines both components of the translation can be updated. Once again the new translation is given by $\mathbf{T}_{\mathrm{t}} = \mathbf{T}_{\mathrm{t}-1} + \Delta\mathbf{T}_{\mathrm{t}}$. However here $\Delta\mathbf{T}_{\mathrm{t}}$ denotes the distance vector between the intersection of the model lines and the intersection of the back-projected lines.

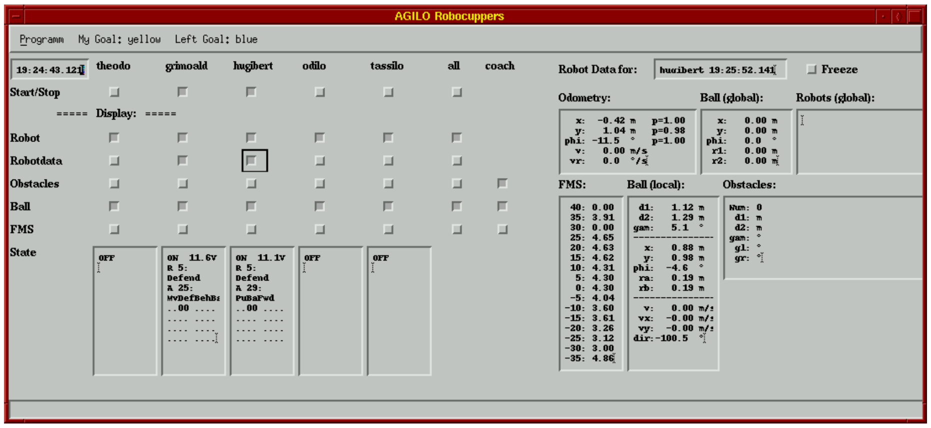
5 Results

We have implemented the above robot vision system in an object-oriented framework that allows the implicit modeling of time within image processing systems [4]. The whole system is based on a modular design and we are able to exchange all image processing algorithms at run-time. This offers great flexibility and enables us to build rapid prototypes of sophisticated vision algorithms. The algorithms presented in this paper are currently applied and represent up-to-now the best solution for our robot soccer team.

In order to verify and measure the accuracy of the (self) localization algorithms we have developed a monitoring program that visualizes the positions, orientations, free motion spaces and ball hypotheses of the robots (see Figure 6). In a second window the states of the robots are displayed, such as the current action and role as well as the planning state (see Figure 7).

The vision system has been tested on a Linux operating system using an low-cost Pentium 200 MHz processor. An inexpensive PAL color CCD camera (Siemens SICOLOR 810) is mounted on top of the robot console and linked to the S-VHS input of the video capture card (BT 848 based with PCI interface). Gain, shutter time, and white balance of the camera are adjusted manually.

With this configuration we are currently able to process 7 to 12 frames with a size of 384×172 pixels per second. The frame rate mostly depends on the method used for determining the field boundaries in order to perform self localization.



```
19:24:43.121

State:
theodo     grimoald       hugibert       odilo    tassilo
OFF        ON   11.6V     ON   11.1V     OFF      OFF
           R 5:           R 5:
           Defend         Defend
           A 25:          A 29:
           MvDefBehBa     PuBaFwd
           ..00 ....      ..00 ....
           .... ....      .... ....
           .... ....      .... ....
           .... ....      .... ....
```

```
Robot Data for: hugibert 19:25:52.141

Odometry:
  x:   -0.42 m    p=1.00
  y:    1.04 m    p=0.98
phi:  -11.5  °    p=1.00
  v:    0.00 m/s
 vr:    0.0  °/s

Ball (global):
  x:    0.00 m
  y:    0.00 m
phi:    0.0  °
 r1:    0.00 m
 r2:    0.00 m

FMS:
 40:  0.00
 35:  3.91
 30:  0.00
 25:  4.65
 20:  4.63
 15:  4.62
 10:  4.31
  5:  4.30
  0:  4.30
 -5:  4.04
-10:  3.60
-15:  3.61
-20:  3.26
-25:  3.12
-30:  3.00
-35:  4.86

Ball (local):
 d1:     1.12 m
 d2:     1.29 m
gam:     5.1  °
----------------
  x:     0.88 m
  y:     0.98 m
phi:    -4.6  °
 ra:     0.19 m
 rb:     0.19 m
----------------
  v:     0.00 m/s
 vx:    -0.00 m/s
 vy:    -0.00 m/s
dir:-100.5    °

Obstacles:
Num:  0
 d1:  m
 d2:  m
gam:  °
 gl:  °
 gr:  °
```

Fig. 7. A second window displays the states of the robots.

Overall the self-localization and object detection algorithms work quite robust. The robots are capable to detect the ball and score goals as well as to detect other robots and avoid collisions. However, there are still cases where a robot estimates its pose incorrectly. In general this occurs when most of the goal and border lines are hidden behind other robots. We are currently investigating further methods, incorporating lines on the field as well as global sensor fusion, to overcome this problem.

6 Conclusions

We have presented a system that segments objects by their color and generates a 3-D interpretation of the scene in real time. Our robot team has proven it's playing capabilities at several occasions (i.e. RoboCup'98 in Paris and the German Vision RoboCup'98 in Stuttgart). However a few problems remain.

The presented color classification algorithm is fast and robust, but we still have to adjust the classifier manually before each game. A more precise assignment of the color class labels to regions in the YUV-space might be one solution, but we are also considering time, position and orientation dependent solutions.

We hope to overcome the problem of incorrect pose estimations with a global sensor fusion system, that constructs a global view of the playing field from the local robot views. So far we had difficulties in exploiting this possibility, as we were relaying on a very unstable wireless radio ethernet. A further approach will also exploit the positions of field lines and the non-linear center circle in the 3-D CAD model.

References

1. Bandlow, T., Hanek, R., Klupsch, M., and Schmitt, T. Agilo RoboCuppers: RoboCup Team Description. In *RoboCup'99 (http://www9.in.tum.de/research/mobile_robots/rob ocup)* (1999), Lecture Notes in Computer Science, Springer-Verlag.
2. Eckstein, W. Unified Gray Morphology: The Dual Rank. *Pattern Recognition and Image Analysis 7*, 1 (1997).
3. Eckstein, W., and Steger, C. Architecture for Computer Vision Application Development within the HORUS System. *Journal of Electronic Imaging 6*, 2 (Apr. 1997), 244–261.
4. Klupsch, M. Object-Oriented Representation of Time-Varying Data Sequences in Multiagent Systems. In *International Conference on Information Systems Analysis and Synthesis (ISAS'98)* (Orlando, FL, USA, 1998), N. Callaos, Ed., International Institute of Informatics and Systemics (IIIS), pp. 833–839.
5. Klupsch, M., Bandlow, T., Grimme, M., Kellerer, I., Lückenhaus, M., Schwarzer, F., and Zierl, C. Agilo RoboCuppers: RoboCup Team Dscription. In *RoboCup'98* (1998), Lecture Notes in Computer Science, Springer-Verlag.
6. Lenz, R., and Tsai, R. Y. Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology. *IEEE Trans. on Pattern Analysis and Machine Intelligence 10*, 5 (Sept. 1988), 713–720.