An Omnidirectional Vision System that finds and tracks color edges and blobs

Felix v. Hundelshausen, Sven Behnke, and Raul Rojas

Freie Universität Berlin, Institut für Informatik
Takustr. 9, 14195 Berlin, {hundelsh|behnke|rojas}@inf.fu-berlin.de

Abstract. We describe the omnidirectional local vision system developed for the RoboCup F180 league soccer team FU-Fighters. Our system consists of a small video camera mounted vertically on top the robots. A concave parabolic mirror placed above the camera reflects the field around the robot. The image is sent by a radio link to an external PC for processing.
Our omnidirectional vision system can find the ball and detects the presence of other robots on the field. The walls of the field are also located and used to determine the position of the robot. In order to be able to process the video stream at the full frame rate the movement of all objects is tracked, including the ball, the obstacles, and the walls of the field. A global image analysis is used to initialize the tracking.
The key of our approach is to predict the location of color edges in the next frame and to search for such color transitions along lines that are perpendicular to the edge.

Introduction

We developed a robotic soccer team for the F180 RoboCup league, the FU-Fighters, that took part in the competitions held at Stockholm and Melbourne in the past two years. For RoboCup 2001 we decided to develop a new generation of soccer robots based on local vision where each robot carries its own camera.

Three tasks have to be accomplished by the computer vision software that analyzes the captured video stream: detecting the ball, localizing the robot, and detecting obstacles. These tasks are non-trivial, since sensor noise, and variances, such as inhomogeneous lighting are present in the images. The image analysis needs to be done in real time, which is not easy, due to the enormous data rate of video streams. Some teams need to reduce frame rate and resolution to match the available computing power, however, such an approach leads to less precise and less timely estimates of the game status, and ultimately to slow play. To be useful for behavior control, the system also needs to be robust. Unexpected situations should not lead to failure, but to graceful degradation of the system's performance.

Local vision is the method used by most teams in the F2000 league as the main sensor. Some of the successful teams adapted the omnidirectional vision approach. The Golem team [4] impressively demonstrated in Melbourne that,

using an omnidirectional camera, sufficient information for controlled play can be collected. Another example for the use of omnidirectional cameras is the goalie of the ART team [3, 6, 7]. Little is known about the implementation of the computer vision software that analyzes the omnidirectional videos.

In our league, F180, only three teams tried to play in Melbourne with local vision only. The limited game performance of these teams shows clearly, that in the smaller form factor the implementation of a local vision system is more challenging than in the F2000 league. The main reasons are that due to space and energy constraints smaller cameras of lower quality must be used and that less computing power is available on the robot. Recently, the OMNI team [5] demonstrated controlled play with omnidirectional local vision at the Japan Open competition. This team sends the video stream to an off-the-field computer that contains special purpose hardware for image processing.

The main idea of the paper is to implement a tracking system for the analysis of the video stream produced by an omnidirectional camera that needs to inspect only a small fraction of the incoming data. This allows to run the system at full frame rate and full resolution on a standard PC.

The remainder of the paper is organized as follows: The next section describes the omnidirectional camera we designed for local vision. Then, a color segmentation algorithm is presented that finds transitions between given colors along a line. Section 3 covers a radial search method that can be used for initial localization. In Sections 4 and 5 the initial search for the ball and the robot is detailed, respectively. Finally, the tracking system is described in Section 6 and some experimental results are reported.

1 Omnidirectional Camera

For our omni-vision system we decided to use the chassis of the previous generation of robots and retrofit it with a small video camera and a mirror, as shown in Fig. 1. We mounted a small PAL camera on top of the robot, directed upwards and looking directly into a parabolic mirror. The mirror collects light rays from

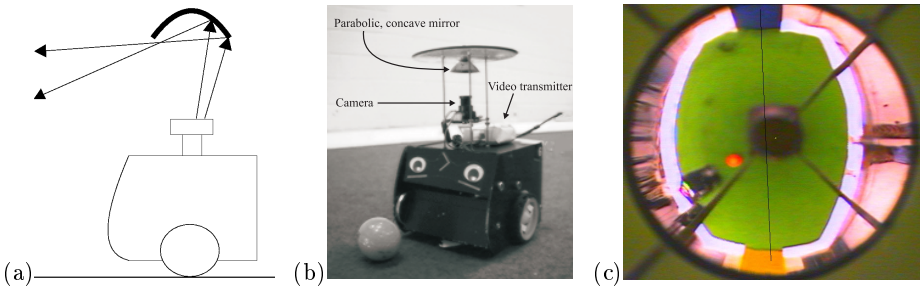


Fig. 1. Omnidirectional camera: (a) principle; (b) physical construction; (c) captured image (see fig 2 for the color classification of the pixels on the line)

all directions, reflecting them directly into the pinhole of the camera. Before settling for this mirror, we tried to use convex spherical and conical mirrors. The concave mirror yields better results because the parabolic shape closely approximates an ellipse. The ellipse has the property that light rays going through one focus and reflecting on the mirror, go through the other focus. If one places the pinhole of the camera at the lower focus, then a good focused image is obtained regardless of the distance of the objects. The mirror must be cut at the level of the other focus, such that an image of the plane from the current position to infinity can be obtained. The parabolic shape of the mirror produces less distortions, as compared to a spherical mirror. Far-away objects appear larger and are hence easier to detect.

In order to avoid carrying a large computer on the robots, the video stream is transmitted to an external computer via an analog radio link.

2 Color Classification and Segmentation

Image analysis in the RoboCup domain is simplified, since objects are color coded. Black robots play with an orange ball on a green field that has yellow and blue goals and white walls. Thus, a pixel's color is a strong hint for object segmentation. We utilize this hint by classifying the colors, that are captured with 15 bit resolution, using a look-up-table (LUT). The classes are not exclusive, since the LUT stores for each color and each class a bit that indicates that the color can be found in images of that object class. Fig. 2 displays the classification of the pixels that belong to a line trough the field as floor, wall, ball, obstacle, yellow and blue goal.

Since color classification is noisy and not exclusive, it can not be used directly for object segmentation. We aggregate the classification by locating color transitions along line that are perpendicular to color edges as shown in Fig. 3.

Assume we have a location $t \in \{0, ..., n\}$ of transition. We define two areas $I_a(t)$ and $I_b(t)$ on both sides of t with predefined lengths n_a and n_b, respectively. Let A be the set of all color classes and let $m_i \subseteq A$ be the color mask of pixel

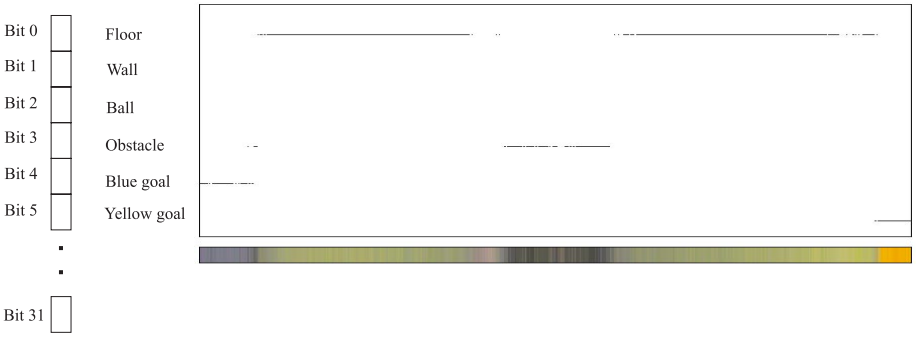


Fig. 2. Color classification of the pixels belonging to the line in Figure 1(c).

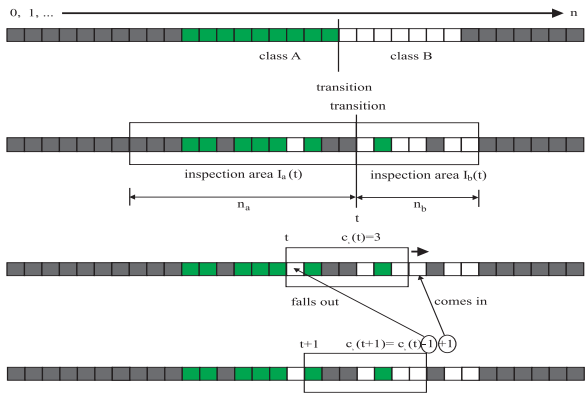


Fig. 3. Search for color transitions: (a) ideal transition; (b) noisy transition with inspection areas; (c) update of $c_b(t)$. It is only necessary to look at the pixel that comes in and the one that falls out of inspection area I_b. Both pixels have the color class b, thus c_b remains unchanged.

i, where $i = 0, 1, ..., n-1$. Let $a, b \in A$ be the color classes of the searched transition. Now the numbers of pixels $c_a(t), c_b(t)$ in each inspection area that have the appropriate color mask can be specified by:

$$c_a(t) = |\{i \in I_a(t) | a \in m_i\}|, \quad c_b(t) = |\{i \in I_b(t) | b \in m_i\}|.$$

Next, we define, that a transition is present at position t, if and only if $c_a(t)$ and $c_b(t)$ are greater or equal to some predefined minimum numbers min_a, min_b.

The task now is to calculate t. As depicted in Fig. 3(c) it is sufficient to look at the two outer pixels of an area for it's color count. The left-to-right search for a transition takes $O(n)$ steps, where n is the number of pixels.

Combined with an assembly coded Bresenham algorithm [1], that computes the pixels that belong to a line, over 45,000 lines at a length of about 300 pixels can be analyzed per second, using a 900 MHz Pentium-III processor.

3 Radial Search and Hough Transformation

The radial search sends lines radially from the robot's perception origin and searches for transitions from the floor to the specified color class of an object. Fig. 4(a) shows the result of searching the transitions from the floor to the walls. This method not only can be used to find the walls, but also to find the goals and the obstacles. The advantage is that it finds parts of the objects that are near the floor, which is important to calculate the correct distance to the objects.

The Hough Transform [2] is a standard tool in image analysis that was developed to detect straight lines but has been extended to detect any geometric shape that can be specified by a fixed small number of parameters. It uses a discretization of the parameter space to accumulate indications into bins. The task of recognizing a shape in an image is transformed to the task of recognizing a point with maximal accumulation in parameter space. Since it is a global transformation, it can detect objects that are incomplete and noisy.

To give an example that will be used for initial robot localization later, suppose that we want to find the walls of the playing field. We first determine several

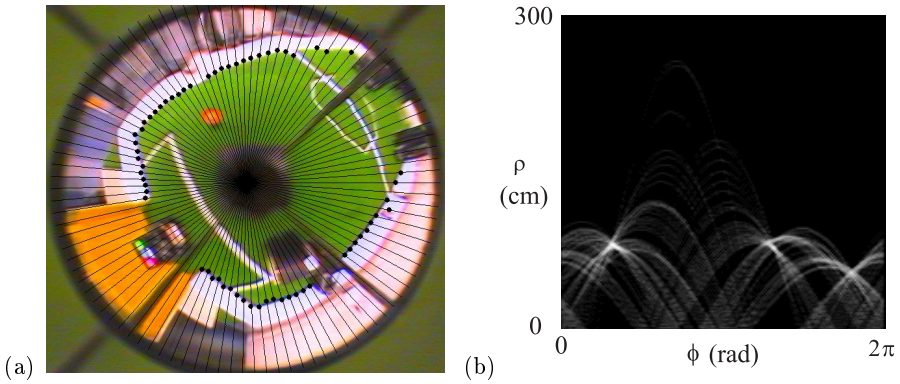


Fig. 4. Wall detection with the radial method: (a) seeking transitions to the wall; (b) parameter space.

points of the walls by applying the radial method, searching for transitions from the green floor to the white wall, as shown in Fig. 4(a).

All the points found are transformed to world coordinates, using the inverse distance function. The corresponding sinusoidal curves are accumulated in parameter space. Fig. 4(b) shows the result of accumulation. The three local maxima correspond to the three visible walls.

4 Initial Ball Search

Finding the ball is of essential importance for successful play. Although it seems to be an easy task, detecting the ball clearly demonstrates the difficulties of computer vision. Figure 5 shows some images of the ball. Its appearance varies greatly in size, shape, and color. Furthermore, it can be easily confused with robot markers. For reliable ball detection it is necessary to look for clusters of pixels of appropriate color. To find these clusters a fast region growing technique can be applied. The algorithm starts with a region consisting of a single seed pixel, having the color class of the ball, and investigates all its neighbors. If they have the appropriate color and have not yet been assigned to any region, they are added to the region and again their neighbors are investigated.

Fig. 5. The ball's appearance varies in color and shape and is similar to robot markers.

Additional information is needed to detect the ball. One observation helps to discriminate it from spurious clusters: Consider the line that reaches from a pixel of the spurious cluster to the perception origin. Since the color markers are on the top of the robots and the sides of the robots are black, the line passes first through the black sides before it reaches the green floor. For the ball there is a similar effect, caused by shadows, but here the dark parts on the line are much smaller. So the idea is to look for a transition from ball color to the floor color, allowing small parts of other color classes (shadow) between them.

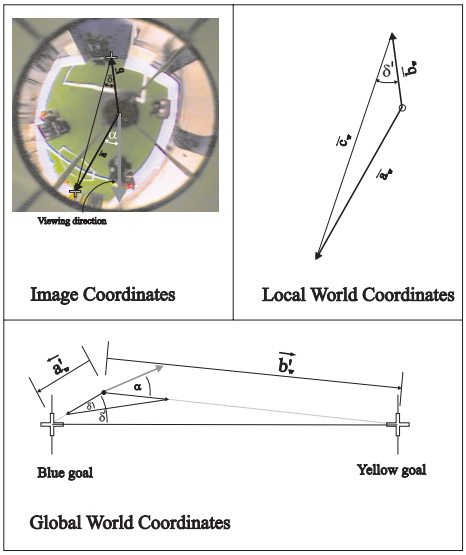
The detection of the ball can be summarized in three steps: (a) determine all clusters of the ball's color class and their sizes, (b) discard all clusters for which no transition to the floor can be found, and (c) choose the biggest cluster. Goals can be detected in a similar way.

5 Initial Robot Localization

The task of initial robot localization is to determine the robot's position and orientation on the playing field, given a captured omnidirectional image. Although there might be approaches for behavior control that need to know only relative positions of objects like the ball and the goals, we think that a global localization is important for intelligent play.

Two different methods have been developed for localizing the robot. The first is fast, but is only applicable when both goals can be found and yields wrong results when a goal has not been detected correctly. The second method is more flexible and robust, not relying on the correctness of single extracted features, but also is slower. In practice, one tries to apply the first method and if it fails the second method is used.

5.1 Direct Localization



Direct Localization is only applicable when both goals have been detected (see figure) The idea is to determine the angle δ', because knowing the distance of the blue goal, the robot's position and viewing direction can be calculated immediately. It is important to understand that δ' is different from δ, since the distance function is not linear. To obtain δ' the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ that reach from the perception origin to the two goals first have to be mapped to local world coordinates. If $\boldsymbol{a_w}$ and $\boldsymbol{b_w}$ denote these mapped vectors, we define $\boldsymbol{c_w} := \boldsymbol{a_w} - \boldsymbol{b_w}$. Now δ' is the angle between $\boldsymbol{c_w}$ and $\boldsymbol{b_w}$.

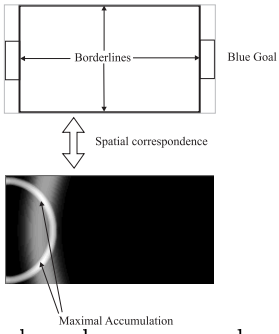
Hence, the position p of the robot lays on a line at an angle of δ' to the line

connecting the two goals. Knowing the distance to the blue goal determines p. The viewing direction also is known, because the angle α at which the yellow goal appears in the image in respect to the robot's viewing direction is preserved by the optical mapping. Of course, for constructing the position and viewing direction either of both goals can be used. In practice one will choose the closer goal, since its position is more likely to be precise.

5.2 Localization using Evidence Aggregation

The second localization algorithm consists of two steps. The first step computes a plausibility value for several positions where the robot could be. In a second step these positions are investigated in the order of their plausibility, until the correct position has been found. For determining the plausibility of each position the evidence accumulation technique is applied. If we recognize e.g. the yellow goal in the image and can estimate its distance from the robot, we have an indication that the robot must be on a circle around the yellow goal. To accumulate indications we divide the playing field into $m \times n$ cells that represent plausibilities, which are initialized to zero.

If the distance to goal x is denoted by r_x, then a semi-circle with radius r_x around goal x is added to the grid. The circles will be drawn more fuzzy for great distances, as the estimation of r_x becomes worse. Suppose, we can estimate the distance to the blue goal as well as to the yellow goal. Than we can draw two circles that may accumulate as shown in the figure. The intersections of the circles can be found by seeking for the cells with maximal accumulation.



Unfortunately, cases may occur, in which only one goal can be seen, e.g. due to occlusions. Thus, another feature needs to be used to indicate the robot's position: the best visible wall. To detect this wall, we use the radial search followed by a Hough Transform, as described above. Choosing the maximum

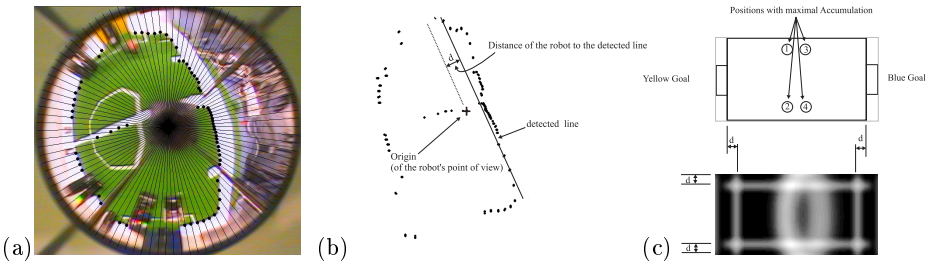


Fig. 6. Using the closest wall for localization: (a) robot next to a wall; (b) detected wall points transformed to world coordinates; (c) grid with goal-circles and wall-lines.

activation in the parameter space, yields the desired result. Since it is not known, which wall has been detected, for all walls parallel lines are drawn on the grid at the perceived distance (see Fig. 6).

After all entries have been made to the grid, local maxima are candidates for robot locations. These candidates are evaluated by a registration procedure (see next section) that computes a quality measure for a model fit. Evaluation starts with the highest maximum. By zeroing cells in the neighborhood of evaluated candidates, it is made sure that candidates are significantly different. The candidate with the best evaluation is used to initialize the tracking.

6 Tracking Objects

Tracking of color edges and blobs is key to the low computational load of our vision system. The idea is to utilize the fact that the world changes slowly, as compared to the frame rate. This makes it possible to predict the location where certain features will appear in the next frame. If most of these features are found close to their predicted positions, only small parts of the image need to be touched. The differences between the measured locations and the predictions can be used to update estimates of the parameters of a world model.

Our system employs a 2D-model of the playing field, with the the robots and the ball on it, as shown in Fig. 7. On the playing field we can find the "seeing" robot (gray), three other robots, and the ball.

For tracking the field, the 2D-model is matched sequentially to the images, seeking the transitions from the floor to the walls and from the floor to the blue and yellow goal. Therefore for each line of the border of the playing field, orthogonal equidistant lines have been added to the 2D-model, referred to as tracking lines in the following. The group of all lines that belong to the same line of the border will be referred as tracking grids.

The endings of each tracking line specify two color classes, according to the expected colors at the position of the line. For instance, the color classes of a tracking line belonging to the transition from the green floor to the yellow goal has the color classes green and yellow.

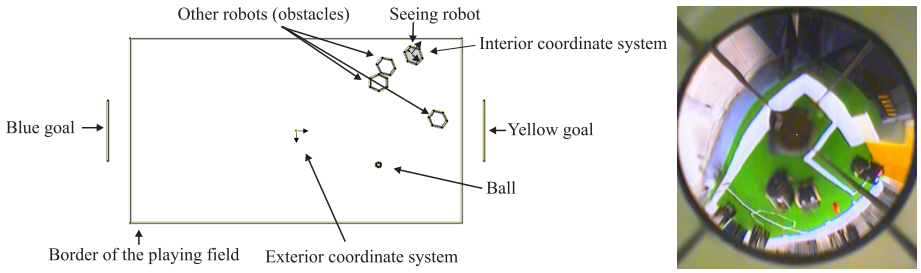


Fig. 7. Model seen from an exterior point of view.

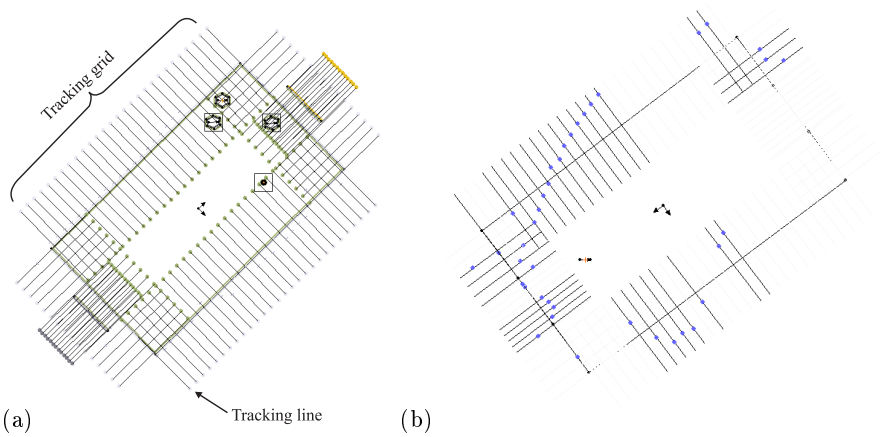


Fig. 8. Tracking of the field: (a) enhanced CAD model; (b) transitions found.

Each tracking line is mapped into the image, using the inverse camera function. This is done by mapping the endpoints of each transition line into the image and then reconnecting them with a line. Of course this is not correct, because the mapped lines would be curves, but if the lines are not too long, the error will be small. Next, the pixels along the lines are searched for a color transition. In figure 8 lines for which a transition has been found are marked with a black dot at the position of the transition. It can be seen that the model does not fit precisely to the playing field in the image, due to a rotation of the robot. Sometimes false transitions may been found, e.g. at field lines or at the secondary field wall. They need to be detected as outliers that must be ignored.

The next step is to calculate a 2-D rigid body transformation that brings the model is correspondence with the found transitions. In the case here, the model should be rotated slightly to the left. To determine the model's transformation, first a rotation and translation is calculated for each track grid independently. Then the results are combined to obtain a transformation for the whole model.

Next, the CAD-model, including the track grids, is rotated around its center and translated according to the estimated parameters. Repeating the seek for transitions and updating the positions of the model while perceiving a sequence of images yields the desired result: the pose of the playing field seen from the robots point of view is tracked and so the position and orientation of the robot on the playing field is known by a simple coordinate transformation.

To reduce the variance of the estimated pose, a Kalmann filter can be used to combine the measurements from different frames. Figure 9(a) shows how the playing field is tracked while the robot rotates.

During initial search candidate positions have to be evaluated using the tracking mechanism. Given a position of the robot on the playing field and an image where we can detect one of the two goals, we can easily calculate the orientation

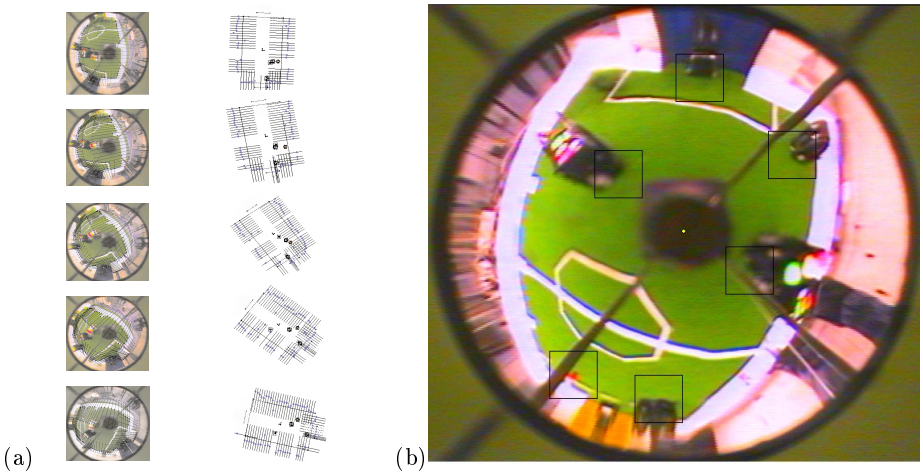


Fig. 9. Tracking of (a) the field while rotating; (b) ball and obstacles.

of the robot. As described above, the field can now be projected into the image and the ratio of found transitions can be used as quality measure for model fit. This quality indicator is also used during tracking to detect situations when the tracking fails and the initial search is needed to localize the robot again.

The system does not only track color edges, but also color blobs, as the ball or obstacles. The blobs are searched for only in small square windows around their predicted positions, as shown in Fig. 9. Again, to compute the predictions, Kalmann filters can be used. If an object cannot be found within its rectangle, initial search is started to find it again.

Conclusions

We implemented a local vision system for the F180 league that uses an omni-directional camera. The system fits a world model to the input by finding and tracking color edges and blobs. It is able to process a full resolution, full frame rate video stream on a standard PC. We plan to use the extracted information about the status of the game as input for a behavior control system. Further, we want to fuse multiple local views to a single global view. Currently, the image analysis is done on an external PC. As semiconductor technology advances, it will be possible to integrate a small computer on-board the robots.

References

1. Bresenham, J. E., *Algorithm for Computer Control of a Digital Plotter*, IBM Systems Journal, 4(1), 1965, 25-30. Hough59 P.V.C.

2. Hough, *Machine Analysis of Bubble Chamber Pictures*, International Conference on High Energy Accelerators and Instrumentation, CERN, 1959.
3. Azzurra Robot Team - ART, *F2000 team homepage*, http://www.dis.uniroma1.it/∼ART/.
4. Golem, *F2000 team homepage*, http://www.golemrobotics.com/gteam/gteam.html.
5. OMNI, *F180 team homepage*, http://robotics.me.es.osaka-u.ac.jp/OMNI.
6. Bonarini, A., (2000) The Body, the Mind or the Eye, first? In M. Veloso, E. Pagello, A. Asada (Eds), Robocup99 - Robot Soccer World Cup III, Springer Verlag, Berlin.
7. Bonarini A., Aliverti, P., Lucioni, M. (2000). An omnidirectional sensor for fast tracking for mobile robots. IEEE Transactions on Instrumentation and Measurement. 49(3). 509-512.