The Attempto Tübingen Robot Soccer Team

Patrick Heinemann, André Treptow, and Andreas Zell

Wilhelm-Schickard-Institute, Department of Computer Architecture,
University of Tübingen, Sand 1, 72076 Tübingen, Germany
{heineman, treptow, zell}@informatik.uni-tuebingen.de

Abstract. This paper describes the Attempto Tübingen Robot Soccer Team 2004. The robot platform, its sensors and actuators, and the software system running on the onboard computer are presented. The main part of the paper concentrates on our current scientific work on modelling and tracking a dynamic environment. Information about dynamic objects moving around in the environment can be useful especially in RoboCup to predict the motion of the ball, to avoid collisions, or to consider objects which cannot be detected over a short period of time. In our robot soccer team we recently implemented an efficient object and landmark detection algorithm based on images of our omnidirectional vision system. To track the detected objects, we use a tracking approach which on the one hand combines the specific advantages of Kalman- and particle filters and on the other hand uses an interacting multiple model filtering approach to model object dynamics as accurately as possible. In addition to the general tracking techniques we present our real-time approach to detect and track uncoloured objects, such as a standard soccer ball.

1 Introduction

Teams of cooperative robots for solving a given task are often based on the idea of a highly precise sensor system giving the robot a complete and accurate view of its environment. In our RoboCup Middle Size League team we followed this idea by building our robots around a very precise laser scanner, at the cost of loosening other constraints like small size and light weight. Although we still played with the laser scanner at RoboCup 2003 at Padova, several aspects including recent rule changes force us to remove that sensor from our robots. In this paper we will present our current team of robots which fully relies on vision sensors for environment modelling. We believe that our scientific research on tracking dynamic objects will help us to cope with imperfect and incomplete sensor data. In any case we are prepared for future rule changes concerning the orange ball through our real-time approach to track uncoloured objects, such as a standard FIFA soccer ball. The remainder of this paper is structured as follows: Section 2 briefly describes the robot, sensor and computer hardware of the Attempto robots, whereas Section 3 focuses on the software controlling the robots. The main part of the paper, however, deals with our current scientific research on object detection and tracking in Section 4. Section 5 concludes the paper with a short summary.

2 Hardware

2.1 Robot Platform

We are currently using the Pioneer2 DX platform from ActivMedia Inc. as the basic robot platform for our field players. This platform is driven by a differential drive system which can achieve translational speeds up to 1.5 m/s and rotational speeds up to 2π/s. The Pioneer2 DX can carry weights up to 20 kg and can be equipped with a maximum of three 7,2 Ah batteries, which allow an operating time of nearly three hours including all additional hardware like onboard PC and additional sensors and actuators.

The two driving motors are equipped with 500 tick position encoders. With the data from these encoders the speed and the relative position and orientation of the robot can be calculated by the onboard Siemens C166 microcontroller board which is also responsible for controlling the actuators of the robot. The communication between the controller board and a remote computer is done via a RS232 serial connection at a maximum speed of 38400 baud.

Our new goalkeeper is based on an omnidirectional platform developed by the University of Dortmund for their own robot team ([1]). We equipped this platform with a set of motors enabling movements at a speed of up to 2.5 m/s.

2.2 Sensors and Actuators

In the past we were employing a diversity of sensors, being convinced that the use of several sensors can result in a highly redundant system and, by the use of suitable data fusion techniques, a better assessment of the situation around the robot. A maximum of six different sensor types including sonars, a 2D laser range finder, two different types of vision systems, infrared proximity sensors, and a digital compass was used on our robots in several configurations. However, the constantly changing environment in RoboCup reduced the applicability of several sensors while others, such as the sonars, and infrared proximity sensors where simply outperformed by better sensors like the high accuracy laser scanner. The trend towards a small number of sensors is further pushed by the need of fast and reactive robots that are able to handle a ball shot by the new kicking devices that are able to accelerate the ball to several metres per second. This forced us to even remove the accurate but heavy laser scanner and led to a new philosophy in the design of our robots, with vision sensors as the only type of external sensors.

Apart from the motors we have our robots equipped with only one more actuator. This pneumatic kicking device will be described in this section, too.

Cameras: The two vision systems we have installed on the robot (perspective camera and omnidirectional vision system) both use a Siemens SICOLOR C810 CCD-DSP color camera, comprising a 1/3" CCD chip with a resolution of 752x582 pixels. The output of the camera is a regular CCIR-PAL signal with 50 half frames per second.

One of the cameras is equipped with a 2,8f wide angle lens and is mounted at the front of the robot. It is used for a precise detection of the ball.

The second camera is equipped with a 4,2f lens pointing up towards a hyperbolic mirror. Although this mirror was designed to maximize the surrounding mapped into the vision system of the FhG Volksbot, a complete mapping of the surrounding is achieved with this vision system, when mounted on top of our Pioneer 2-DX Robots, too.

Kicker: Our kicking device is a self-made construction actuated by compressed air. The air is compressed into a 2 litre tank before the games at a pressure of 10 bar. The air reservoir is connected via an electric valve to two pneumatic actuators that can accelerate a metallic bar which shoots the ball. The special feature of this device compared to others is that the bar is mounted and connected to the pneumatic cylinders in a way that accelerate the bar in a circular motion forwards and also upwards. This reduces the overall speed of the ball but leaves the possibility to lift the ball over a goalkeeper as we could show in a game against ISePorto at RoboCup 2003. Currently we are thinking about an electronic kicking device for our new goalkeeper.

2.3 Onboard Computer

Our onboard computer is a custom designed system based on a PISA-3P4I Backplane by JUMPtec which provides 4 ISA/PISA slots and 3 PCI slots. One of these slots is used to plug a CoolMonster/P3 PISA board by Jumptec which integrates the complete functionality of a motherboard, like a network controller, IDE controller, and USB controller. This board is equipped with a low power Pentium-3 running at 850 MHz, 128 MB of RAM and a 20 GB harddisk. Additionally two PCI framegrabber boards based on the Booktree BT848 chipset are added to the computer to simultaneously grab the images of the two vision systems at 25 fps. The laser scanner is connected via a high speed RS422 serial device card which was modified to achieve the 500 kbps data rate. The computers of different robots can communicate via IEEE 802.11b wireless LAN by ARtem Datentechnik over an external access point. Therefore each robot is equipped with a WLAN client which is connected to the onboard computer via RJ45 network cable. The communication to the controller board of the robot is done over the RS232 serial device and a crosslink cable. We are running RedHat 7.3 Linux on the computer.

3 Software

The software system of the Attempto Tübingen Robot Soccer Team is based on a Client/Server architecture and can be divided into three layers: the data server layer, an intermediate layer and the high level robot control layer.

In the data server layer several server programs perform the communication with the sensor and robot hardware. They provide the data from the sensors and the robot to the preprocessing clients in the intermediate layer via shared

memory segments. These segments are organised in a ring buffer structure to provide a free buffer for the next data packet even if one or more clients are processing data from other segments and thus blocking the use of these segments. The robot server that supplies odometry data is actually a client, too, as it reads command data from a shared memory segment and makes the robot fulfill these commands. All the servers in this layer can be replaced by simulation servers which provide previously recorded or artificial data for simulation purposes.

The intermediate layer acts as a data compression layer. Several preprocessing stages extract the relevant information out of the raw sensor data and provide it to the high level layer, again being both client and server. The image preprocessing stages (one for each camera) compute the position of objects (own robots, opponent robots, and the ball) relative to the robot and extract points on the white field markings. In an object tracking stage the objects generated from the image preprocessing stages are fused to further reduce the amount of data and to remove inconsistencies and the remaining objects are tracked over time by our tracking system presented in [4]. A localisation stage processes the field markings from the images, and the odometry data from the robot to generate new position estimations. These estimations are used to update a filter that keeps track of the robot's position. The output of the stages in the intermediate layer provide a consistent world model to the high level layer.

The high level robot control layer realises the hybrid robot control architecture [3]. It consists of a reactive component where a set of independent behaviours like obstacle avoidance, ball search, or ball following try to fulfill their tasks. The behaviours can react quite fast to changes in the environment because they can work on the compact world model data from the immediate layer. The behavioural system is easy to expand beacuse it is possible to start and stop behaviours at runtime. Control commands are passed to the robot via the robot server. A more detailed description of the software system is given in [9].

4 Research Topics

4.1 Object Detection

This section briefly describes the algorithm to detect objects and landmarks with our vision systems. A detailed description can be found in [5].

In order to create an environment model from the images obtained with our vision systems, two basic steps have to be performed. First, the objects and landmarks in an image have to be identified, according to the distinct colour coding in the RoboCup environment. Second, the real world position of objects and landmarks has to be calculated from their pixel coordinates. For these steps of environment modelling, mappings from colours to different classes of objects and landmarks (e.g. black for robots and orange for the ball), and from pixels to real world coordinates are needed. These mappings are identified during colour calibration and a distance calibration, once for each of the two vision systems, each robot, and for each different environment. To reduce the time and effort

needed, we implemented a graphical tool helping us to do the calibration, but nevertheless these steps are still done manually. Currently we are working on algorithms to automatically calibrate the vision systems.

Once the vision systems are calibrated, the algorithm cycles through five major steps for every image:

1. Transformation of the image into colour classes
2. Segmentation of the image by colour class
3. Clustering of segments that are believed to belong to the same object
4. Mapping of the cluster to world coordinates using the point with minimum distance to the image centre
5. Output of this position as new object, if size constraints are fulfilled.

These steps are described in the following subsections.

Colour Transformation In the colour transformation step the pixels of the image are transformed into the different colour classes using the trained colour LUT. Instead of transforming the whole image, we use a regular grid in world coordinates and map this grid back to image coordinates to reveal the position of interesting pixels. With a resolution of the grid that is high enough to cover even the smallest robot with at least one gridline, we process less than 5% of the image, which makes the colour transformation very efficient. The pixel coordinates of the gridpoints can be calculated during the calibration steps and are stored into a look up table ($gridLUT$).

Colour Segmentation To find the longest sequences of a given colour class in the colour transformed grid the fast and fault-tolerant Smith-Waterman algorithm for local alignment of common molecular subsequences ([11]) was adapted. This algorithm finds best matching substrings by comparing the amino acids (aa) of two strings one by one, assigning high scores for exact matches and lower or negative scores for unequal pairs. In the end of this process, the substring with the best score is considered as the best matching substring. The algorithm is fault-tolerant, as it allows a certain number of unequal aa in the two strings. This number is based on the scores given to the compared aa, which in turn are based on statistical analysis of how likely two aa may be exchanged without affecting the characteristics of the whole sequence.

Regarding the colour transformed gridlines as an aa string, with the colour class of the pixels being an identifier similar to the character of an aa, substrings (segments) of a given colour class can be found when comparing the gridline against a string completely filled with this colour class. The algorithm assigns positive scores to pixels belonging to the same colour class and a negative score to pixels of a different colour class. Pixels of a different colour class would usually end a segment, without consideration of errors in the colour transformation step due to noise in the image. Using the Smith-Waterman algorithm, however, segments are extended over gaps of pixels of a different colour class.

The algorithm processes each gridline, one at a time. For each gridpoint in the current line a new segment of the pixel's colour class is started, if there was currently no unfinished segment of this colour class. For all unfinished segments, regardless of their colour class, the score for this pixel based on the pixel's colour class and the colour class of the segment is added to a grand total for this segment.

A segment is finished, when its score drops below zero, which is the case after a certain number of non-matching pixels occured in a row, or when the current gridline is finished. However, the end of the segment is determined as the position which reached the highest score in the segment, as this is the point of the highest number of matching pixels and only some small errors in a row.

Clustering of Segments The clustering step clusters the segments based on their centre points in pixel coordinates. This is only done for segments of black colour (robots) and orange colour (ball) as these are the only objects that appear on a RoboCup field. For the white line markings and blue and yellow goal colour no further processing is needed after the segmentation step.

Calculation of Real World Position From all segment points of a cluster the point nearest to the image centre is mapped to its world coordinates using the distance mapping learned in the calibration step.

Apply Size Constraints Finally some size constraints are verified to ensure that small objects resulting from pixel noise are dropped and objects that are underestimated in their size are enlarged at least to their minimum size to prevent the robot from collision with such objects.

In [5] we were able to show that this algorithm is very efficient as the computation time for the whole algorithm was 7.2ms on a Pentium III 850MHz that is used on our robots, leaving enough time for other tasks while processing the full 25 frames per second of the camera. In addition to the efficiency we could show that the accuracy of the object detection is sufficiently high, when compared to laser scanner measurements.

4.2 Self-Localisation

In the RoboCup 2003 competition at Padova we already used a new approach to the problem of self-localisation based on the white field markings obtained from the omnidirectional vision system. From a given starting position the robots were able to constantly correct the odometry errors and track the current position. This approach, however, had some drawbacks concerning accuracy and efficiency. As it was designed to correct small estimation errors from the odometry, longer periods without a correction often caused the localisation to end in a local minimum, e.g. an estimation error of 90° compared to the correct orientation. Futhermore the algorithm was unable to detect this estimation error,

as there was no quality measure for the estimation. Although it was possible to do a global positioning from scratch this last one or two seconds which was too much to use during a match.

Inspired by the ideas given in [6] we implemented a new version of the field markings based self-localisation. As a quality measure this approach uses the *matrix of qualities* explained in [6]. This is a precalculated look up table containing the distance to the nearest element of the field markings model for every position on the field.

If the set of white line points observed by the omnidirectional vision system is transformed according to the current position estimation of the robot, we can efficiently add up the distance to the field markings for each point, given in the matrix, and devide by the number of points. The result is the total distance of all observed line points to the field markings model, which serves as a quality measure of our position estimation. With this approach we are able to control the accuracy of our localisation and reinitialise the tracking in case of a possibly wrong position estimation. In this case, we calculate the quality of a set of position estimations lying on a coarse grid with 16 possible orientations at each position. The position with the highest quality is considered as the correct position estimation. Due to the precalculated distances, this reinitialisation is very efficient.

Once we have an initial position estimation, the tracking is done as before. The position estimation is updated with the odometry and corrected with the data of the vision system. To correct the position, however, we need to know in which direction we have to correct the estimation. Therefore we calculate a *matrix of forces* representing the direction in which the nearest element of the field markings model lies and in which a point should be moved to match the model. If we add up all forces that attract the observed line points, we get an idea in which direction the position estimation has to be corrected to improve the quality. The correction in orientation is calculated as a torque, the forces exert on the centre of gravity of all line points. Iteratively we can now correct the position estimation until a desired quality is achieved, or until a given maximum number of iterations was exceeded. If the quality is still lower than a specified value, we start again with the reinitialisation step.

First experiments show that this method is very efficient and accurate.

4.3 Tracking Uncoloured Objects

In the RoboCup environment every object is marked with a special colour so that fast and robust colour segmentation algorithms can be used for object detection [7][12]. In the future these colour markers will be removed in order to come to a more realistic setup. Therefore, the aim of our research is to introduce algorithms to detect and track objects that do not have colour information. In a first step we want to be able to replace the orange soccer ball and play with a standard FIFA ball. In this section we will give a short overview of this work. A more detailed description can be found in [2].

To build a colourless model of the ball we use an algorithm proposed by Viola and

Jones [8] that has been used to detect faces in real-time based on simple gray-level features. We used their approach to come to a feature based description of the ball. As proposed in [8] we use four different types of features (see figure 1).

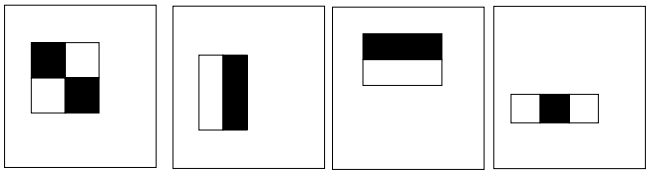

Fig. 1. Four different types of rectangle features within their bounding box. The sum of pixels in the white boxes are subtracted from the sum of pixels in the black areas.

The advantage of using these features is that they can be calculated very fast on top of a so called integral image (see [8] for details). To build a ball classifier based on these gray level features one has to select the most relevant features. Remember that within a box sized 24x24 there are more than 160000 features, which is far more than the number of pixels. As proposed by Viola *et. al.*, we use the machine learning procedure called Adaboost [13] to select a small number of relevant features. For the offline training of the ball classifier we collected a set of 980 pictures (sized 19x19) showing the ball under different viewing conditions and 8290 pictures that do not contain the ball. These sets are randomly split into a training and a test set. To classify the training set correctly, Adaboost selects 132 features. On the test set we achieve a detection rate of 91.22% and a false positive rate of 0.018%.
To track the ball we use a particle filter: The ball is described by the state vector

$$x_t^{(i)} = [x_I, y_I, s_I, v_x, v_y, v_s]^T \tag{1}$$

where (x_I, y_I) is the position of the ball in the image, (v_x, v_y) is the velocity in x- an y-directions, s_I represents the size of the ball and v_s is the velocity in size. The dynamics of the ball are modelled as a movement with constant velocity and small random changes in velocity (random walk). Every particle is weighted with the classification result of the ball classifier that has been learned offline by the Adaboost mechanism. Instead of using the binary value we weight every particle with the result of the linear combination of the features. We use random initialization for the particle filter.
Using 400 samples and a ball classifier with 40 features, one timestep of the tracking algorithm requires about 9ms on a AthlonXP 1600MHz processor so that we are able to track the ball with more than 25 fps. In different experiments the tracker has shown to be robust against occlusion and distraction. Examples of our ball tracking system can be seen in figure 2 and 3.

We treat the weighted mean of the best 30% of the particles to be the final hypothesis of the ball position. Nevertheless, there are situations were we get

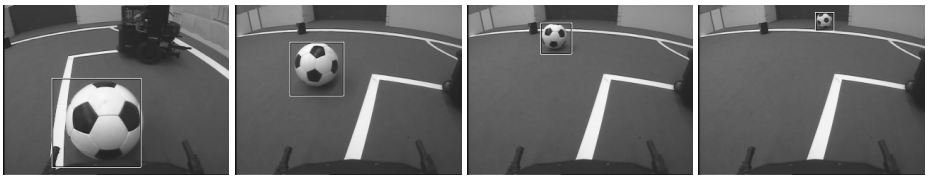

Fig. 2. Tracking the ball at different scales.

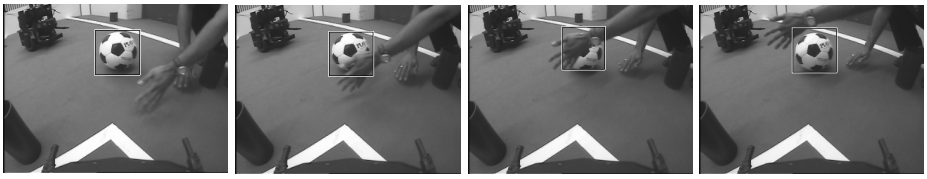

Fig. 3. Tracking the ball through occlusion.

false detections so that the tracker is not able to recover from distraction. To improve robustness further, we will implement methods to measure the shape. Besides the work of Hanek *et. al.* [10] who do not deal with the problem of global detection, our approach is one of the first to detect and track a "normal", non-coloured FIFA-ball in real-time. The presentation of this work was the main reason for winning the Technical Challenge Award of the Middle Size League at the RoboCup world championship in Padova 2003.

5 Summary and Diskussion

In this paper we introduced our current and new approaches to stay competitive in future RoboCup competitions. Besides the introduction of a new goal keeper, we mainly focus on research to improve the capabilities of our team. Our ability to do robust and reliable self-localisation, object detection, and object tracking, enables us to concentrate on higher level tasks, such as improved cooperative team play, in the future. Being able to detect and track a standard uncoloured FIFA ball in real-time as one of the first teams in the world we are also prepared for a further reduction of the colour markings on the field. In RoboCup 2003 at Padova we could successfully present this new ability.

References

1. http://ls1-www.cs.uni-dortmund.de/~pg425/.
2. A. Treptow, A. Masselli and A. Zell. Real-Time Object Tracking for Soccer Robots without Color Information. In *Proceedings of the European Conference on Mobile Robots (ECMR 03)*, September 2003.
3. Ronald C. Arkin. *Behavior-Based Robotics.* The MIT Press, 1998.

4. P. Heinemann, M. Plagge, A. Treptow, and A. Zell. Tracking Dynamic Objects in a RoboCup Environment - The Attempto Tübingen Robot Soccer Team. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup-2003: Robot Soccer World Cup VII*, Lecture Notes in Computer Science (CD-Supplement). Springer Verlag, 2003.
5. P. Heinemann, T. Rückstieß, and A. Zell. Fast Object Detection in RoboCup using Omnidirectional Vision (to appear).
6. F. von Hundelshausen, M. Schreiber, F. Wiesel, A. Liers, and R. Rojas. MATRIX: A force field pattern matching method for mobile robots. Technical Report B-09-03, FU-Berlin, 2003.
7. J. Bruce, T. Balch and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061–2066, 2000.
8. P. Viola and M.J. Jones. Robust real-time object detection. In *Proc. of IEEE Workshop on Statistical and Theories of Computer Vision*, 2001.
9. M. Plagge, R. Günther, J. Ihlenburg, D. Jung, and A. Zell. The Attempto RoboCup Robot Team. In H. Kitano M. Veloso, E. Pagello, editor, *RoboCup-99: Robot Soccer World Cup III*, volume 1856 of *Lecture Notes in Artificial Intelligence*, pages 424–433. Springer Verlag, 2000.
10. R. Hanek, T. Schmitt, S. Buck and M. Beetz. Towards RoboCup without Color Labeling. In *RoboCup International Symposium*, Fukuoka, Japan, 2002.
11. T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.
12. T. Bandlow, M. Klupsch, R. Haneka and T. Schmitt. Fast Image Segmentation, Object Recognition and Localization in a RoboCup Scenario. In *3. RoboCup Workshop, IJCAI'99*, 1999.
13. Y. Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999.