A Vision Based System for Goal-Directed Obstacle Avoidance used in the RC'03 Obstacle Avoidance Challenge

Jan Hoffmann, Matthias Jüngel, and Martin Lötzsch

Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, http://www.aiboteamhumboldt.com

Abstract. We present a complete system for obstacle avoidance for a mobile robot. It was used in the RoboCup 2003 obstacle avoidance challenge in the Sony Four Legged League. The system enables the robot to detect unknown obstacles and reliably avoid them while advancing toward a target. It uses monocular vision data with a limited field of view. Obstacles are detected on a level surface of known color(s). A radial model is constructed from the detected obstacles giving the robot a representation of its surroundings that integrates both current and recent vision information. Sectors of the model currently outside the current field of view of the robot are updated using odometry. Ways of using this model to achieve accurate and fast obstacle avoidance in a dynamic environment are presented and evaluated. The system proved highly successful by winning the obstacle avoidance challenge and was also used in the RoboCup championship games.

1 Introduction

1.1 Related Work

Obstacle avoidance is often achieved by direct sensing of the environment. Panoramic sensors such as omni-vision cameras and laser range finders are commonly used in the RoboCup domain [1, 10]. Using these sensors, a full panoramic view is always available thus greatly simplifying the task of obstacle avoidance. Free space is usually associated with green (i.e. floor color), whereas non-green colored pixels are associated with obstacles (see introduction of [5] for an overview of panoramic vision systems).

In the case of the Sony League, a camera with a limited field of view is used. As a basis for obstacle avoidance, a radial model of the robot's environment needs to be maintained. In such a model, current vision data is integrated with recent vision data. Recently presented work by Scott Lensor and Manuela Veloso show how such a model is constructed ("visual sonar", [5]). The approach to obstacle detection and obstacle modeling used by the GermanTeam in the RoboCup challenge turned out to be similar to this concept. The contribution of this papers is on how such a model can be used to achieve goal-directed obstacle avoidance. Potential field approaches [4] were not considered because, on the one

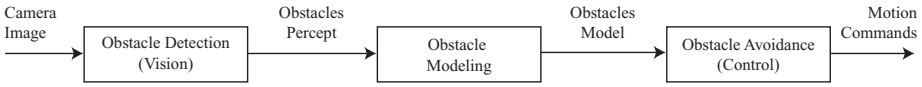


Fig. 1. Information processing in the obstacle avoidance system.

hand, the robot's environment changes rapidly which makes it hard to manage a global world model used for path planning. On the other hand, the static part of the environment is simple and elaborate path planning is not necessary. A comparative study of path planning and obstacle avoidance algorithms is given in [7]. Many of these algorithms work well in simulation but are difficult to implement in situations where information about the robot's surroundings are highly uncertain. We therefore decided on using a rather simplistic algorithm which can be summarized as follows: walk towards the goal - if there is an obstacle, cling to it and walk around it until it is circumvented - continue towards the goal.

1.2 Preliminary Experiments

Two preliminary experiments are described here to outline what can and what cannot be achieved using even simple approaches and to motivate the use of the later presented system.

Simple obstacle avoidance using minimal sensor data. For initial testing and benchmarking, a behavior similar to that of a 2nd order Braitenberg vehicle was implemented (light seeker/-avoider, [2]). It makes use of the color coding of the robot's environment: unoccupied floor (i.e. free space) on a soccer field is green while obstacles are colors other than green. In analogy to the light sensors of the Braitenberg vehicle, the camera image was divided in two halves. For each half, the number of green pixels (i.e. floor pixels) is calculated and normalized to the total number of green pixels in the camera image. The camera itself points slightly below the horizon to make sure that the camera is pointed (mostly) at the floor. The robot itself moves forward at a constant speed. Depending on the amount of floor colored pixels encountered in the left or right half, the robot turns towards the direction where there is more free space while still moving forward. Such a behavior lets the robot wander about the field aimlessly while avoiding obstacles (such as field borders, other robots, people's feet). At times, however, it cannot turn quick enough to avoid running into an obstacle.

Adding the distance sensor. The one-dimensional distance sensor (infra red) mounted in the robot's head was used to control the speed of the robot. The robot moves forward at full speed if the distance measured equals the distance to the ground (keeping in mind that the robot's head is pointed at the ground). For distances smaller or greater the robot slows down and even backs off. By also slowing down when the measured distance becomes greater than the assumed

distance to the ground, falling off ledges can be avoided. We were able to have the robot wander about on a table top, avoid obstacles on the table and to stay clear of the table's edge. Adding this control enables the robot to stop in its track when it happens to come too close to an object (which is often the case when turning). It did, however, have problems with small objects because the 1D distance sensor would miss them.

The previous examples show the limit of what can be achieved by simple sensor data based, reactive behaviors. Creating a model of the robot's surroundings is useful as it gives the robot information of what it has recently seen and it can help to avoid small obstacles (e.g. the leg of a chair) that the robot's distance sensor is currently missing but has detected recently.

The following paragraphs describe the obstacle avoidance system used by the GermanTeam in the RoboCup obstacle avoidance challenge and the actual games (see also fig. 1). The goal was to find an obstacle avoidance solution best suited for the demands of the dynamic RoboCup domain. This implies that obstacle avoidance is one important task among others the robot has to perform during the game.

2 Obstacle Avoidance System

The following sections will describe obstacle detection, obstacle modeling, and obstacle avoidance behavior. A Sony Aibo ERS210(A) robot was used in the experiments. The robot has a 400 MHz MIPS processor and a camera delivering YUV image with a resolution of 172x144 (8 bits per channel). A Monte Carlo localization was used [9]; other modules not covered here such as walking engine, etc. are described in more detail in the GermanTeam 2003 team description and team report [8].

2.1 Obstacle Detection

Image processing yields what we call a *percept*. A percept contains information retrieved from the camera image about detected objects or features later used in the modeling modules. A percept only represents the information that was extracted from the current image. No long-term knowledge is stored in a percept.

The *obstacles percept* is a set of lines on the ground that represents the free space in front of the robot in the direction the robot is currently pointing its camera. Each line is described by a *near point* and a *far point* on the ground, relative to the robot. The lines in the percept describe segments of ground colored lines in the image projected to the ground. For each far point, information about whether or not the point was on the image border is also stored.

To generate this percept, the image is being scanned along a grid of lines arranged perpendicular to the horizon. The grid lines have a spacing of 4°. They are subdivided into segments using a simple threshold edge detection algorithm. The average color of each segment is assigned to a color class based on a color look-up table. This color table is usually created manually (algorithms that

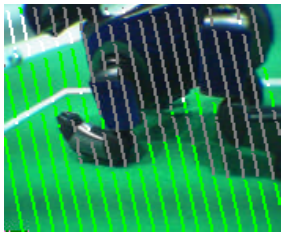
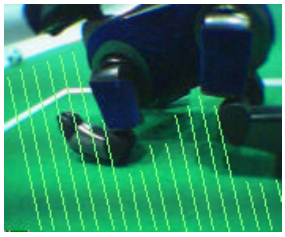
a)  b) 

Fig. 2. Obstacle detection. a) White, green and unclassified segments of the scan lines. b) Green lines: The obstacles percept is the conjunction of close green segments.

automate this process and allow for real-time adaptation exist [3, 6]). For each scan line the bottom most ground colored segment is determined. If this ground colored segment meets the bottom of the image, the starting point and the end point of the segment are transformed from the image coordinate system into the robot coordinate system and stored in the obstacles percept; if no pixel of the ground color was detected in a scan line, the point at the bottom of the line is transformed and the near point and the far point of the percept become identical.

Small gaps between two ground colored segments of a scan line are ignored to assure robustness against sensor noise and to assure that field lines are not interpreted as obstacles. In such a case two neighboring segments are concatenated. The size limit for such gaps is 4 times the width of a field line in the image. This width is a function of the position of the field line in the camera image and the current direction of view of the camera. Figure 2 shows how different parts of scan lines are used to generate obstacles percepts.

The different kinds of information about obstacles in the robot's field of view that can be deduced from the obstacle percept are illustrated in fig. 3.

2.2 Obstacle Model

The obstacle model described here is tailored to the task of local obstacle avoidance in a dynamic environment. Local obstacle avoidance is achieved using the obstacle model's analysis functions described below. The assumption was made that some high level controller performs path planning to guide the robot globally. Certain global set ups will cause the described algorithm to fail. This, however, is tolerable and that this is a different type of problem that needs to be dealt with by higher levels of action planning. This work therefore concentrates on a method to reliably steer the robot clear of obstacles while changing its course as little as possible.

In the model, a radial representation of the robot's surroundings is stored in a "visual sonar" [5]. The model is inspired by the sensor data produced by panoramic sensors such as 360° laser range finders and omni-vision cameras. In this model, free space in a certain direction θ is stored. θ is divided into n discrete sectors ("micro sectors").

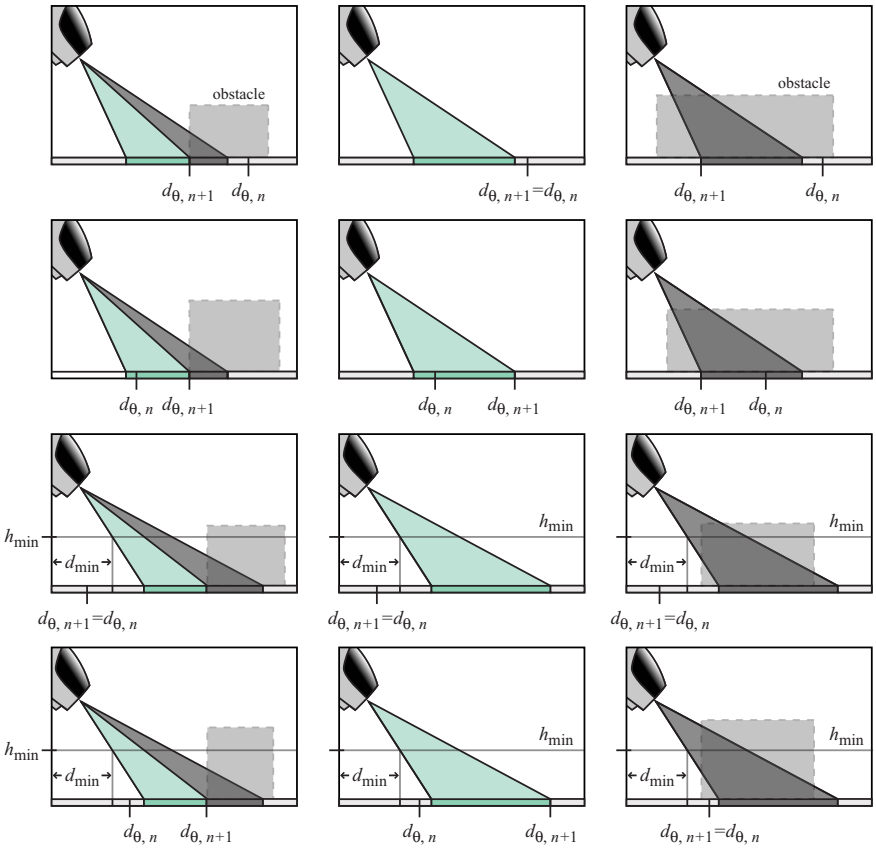


Fig. 3. Updating the obstacle model. The above diagrams illustrate the way that the entry in the obstacle model for a given sector θ is updated using vision information. d_n denotes the information stored in the model and d_{n+1} the updated data. Three cases of vision data are shown (columns): 1) some free space in front of the robot but also an obstacle at a certain distance (left column), 2) no obstacle within the field of view (middle column), and 3) totally obstructed view (right column). Cases in which an obstacle is detected and also some free space behind the obstacle are treated as 3).
The lower two rows illustrate the information that can be derived if the distance stored in the model lies between the robot and the bottom edge of the viewing cone. By assuming that obstacles are of a minimum height $h_{\min}$, free space can be inferred even for regions that are not visible.

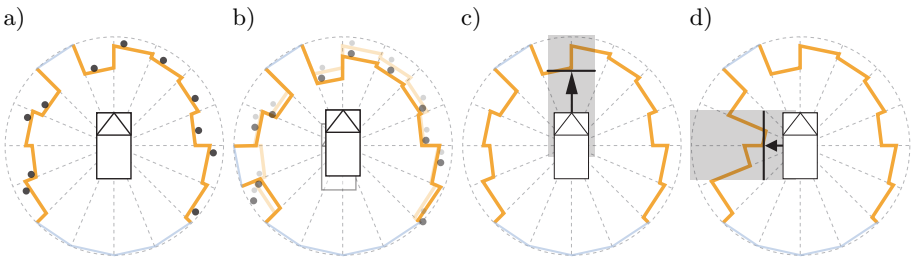


Fig. 4. Illustration of the obstacle model. The actual number of sectors is greater than shown here, it was reduced for illustration purposes. Fig. 5 shows the actual obstacle model used.
a) The robot is at the center; dashed lines show sectors; solid orange lines (dark) show the free space around the robot; light grey lines are used if there is no information about free space in a sector (in this examples, the robot has little knowledge about the sectors behind it); small circles denote *representatives*. b) illustrates how the model is updated using odometry when the robot is moving. Updated representatives are shown as dark circles dots. Note that one of the right sectors does not contain a representative after updating, hence the free distance is reset to infinity. c) and d) illustration of analysis function used when determining the free space in front of the robot and to its side (important when the robot is turning).

If new vision information is received, the corresponding sectors are updated. Sectors that are not in the visual field are updated using odometry, enabling the robot to "remember" what it has recently seen. If a sector has not been updated by vision for a time period greater than t_{reset}, the range stored in the sector is reset to "unknown".

Micro sectors are 5° wide. Due to imperfect image processing the model is often patchy, e.g. an obstacle is detected partially and some sectors can be updated while others may not receive new information. Instead of using the model as such, analysis functions that compute information from the model are used. These functions produce high level output such as "how much free space is (in the corridor) in front of the robot" which is then used by the robot's behavior layers. These functions usually analyze a number of micro sectors. The sector with the smallest free space associated to it corresponds to the greatest danger for the robot (i.e. the closest object). In most analysis functions this sector is the most important overruling all other sectors analyzed. In the above example, the sector in the corridor containing the smallest free space is used to calculate the free space in front of the robot. Using analysis functions makes using the model robust against errors introduced by imperfect sensor information. It also offers intuitive ways to access the data stored in the model from the control levels of the robot.

In addition to the free space, for each sector a vector pointing to where the obstacle was last detected (in that sector) is stored. This is called a *representative* for that sector. Storing it is necessary for updating the model using odometry.

Fig. 4 illustrates the obstacle model. The following paragraphs will explain in more detail how the model is updated and what analysis function are.

Update Using Vision Data The image is analyzed as described in 2.1. Obstacle percepts are used to update the obstacle model. The detected free space for each of the vertical scan lines is first associated to the sectors of the obstacle model. Then the percept is compared to the free range stored for a sector; fig. 3 illustrates the possible cases for updating the information stored in a sector θ.

If the distance in a sector was updated using vision information, the obstacle percept is also stored in the representative of that sector. The necessity to store this information is explained in the following paragraphs.

Update Using Odometry. Sectors that are not in the visual field of the robot (or where image processing did not yield usable information) are updated using odometry. The representative of a sector is moved (translated and rotated) according to the robot's movement. The updated representative is then remapped to the - possibly new - sector. It is then treated like an obstacle detected by vision and the free space is re-calculated. In case more than one representatives are moved into one sector, the representative closest is used for calculating the free space (see Fig.4 b. for an example). If a representative is removed from a sector and no other representative ends up in that sector, the free space of that sector is reset to infinity). The model quality deteriorates when representatives are mapped to the same sector and other sectors are left empty. While this did not lead to any problems in our experiments and the way we were able to use the model, we would like to point out two ways to compensate for the "loss" of representatives:

1. Instead of storing just one representative per sector, any number of representatives could be stored per sector. This only lessens the effect and shifts it to a smaller scale.

2. Another approach was presented by [5] which is based on a similar radial obstacle model. Whenever a gap between two formerly adjacent sectors of the obstacle model occurs, one or more new points are created in-between the two by linear interpolation. This completely eliminates the deterioration effect but also makes the assumption that two adjacent obstacle representatives belong to the same obstacle which is not always the case.

The described effect was observable in some experiments and could easily be reproduced in simulation, but since no significant effect on real world performance was observed it was disregarded.

Analysis Functions. As explained above, the model is accessed by means of analysis functions. The micro sectors used to construct the model are of such small dimensions that they are not of any use for the robot's behavior control module. The way we model robot behavior, more abstract information is needed, such as "There is an obstacle in the direction I'm moving in at distance x" or "In the front left hemisphere there is more free space than in the front right." Of interest is usually the obstacle closest to a the robot in a given area relative to

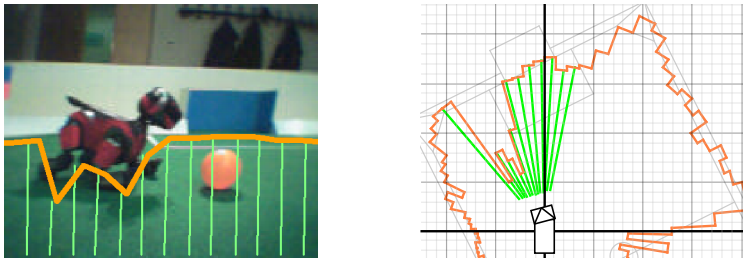


Fig. 5. *Left.* Camera image with superimposed obstacle percepts and obstacle model (projected onto the floor plane) *Right.* Actual obstacle model.

the robot. In the following paragraphs, some analysis functions that were used for obstacle avoidance and in RoboCup games are described. Other possible function exist for different kind of applications which are not covered here.

Macro sector $sect(\theta, \Delta\theta)$ This function is used to find out how much free space there is in a (macro) sector in direction θ an of width $\Delta\theta$. Each sector within the macro sector is analyzed and the function returns the smallest distance in that macro sector. This can be used to construct a simple obstacle avoidance behavior. The free space in two segments ("front-left", $-22,5° \pm 22,5°$ and "front-right", $+22,5° \pm 22,5°$) is compared and the behavior lets the robot turn in the direction where there is more free space.

Corridor $corr(\theta, \Delta d)$ If the robot is to pass through a narrow opening, e.g. between two opponent robots, the free space not in a (macro) sector but in a *corridor* of a certain width is of interest. Usually, a corridor of about twice the width of the robot is considered safe for passing.

Free Space for Turning $corr(\theta = \pm 90°, \Delta d =$**length of robot**$)$ When turning, the robot is in danger of running into obstacles that are to its left or right and thereby currently invisible. These areas can be checked for obstacles using this function. If obstacles are found in the model, the turning motion is canceled. (Note that this is a special case of the corridor function described above.)

Next Free Angle $f(\theta)$ This functions was used in RoboCup games to determine which direction the robot should shoot the ball. The robot would only shoot the ball in the direction of the goal if no obstacles were in the way. Otherwise the robot would turn towards the "next free angle" and perform the shot.

2.3 Ostacle Avoidance

The obstacle avoidance behavior used in the challenge is described here. The way the obstacle model was used in the championship games is also touched briefly.

Goal-directed obstacle avoidance. The following three states were used to achieve goal-directed obstacle avoidance (see also fig. 6):

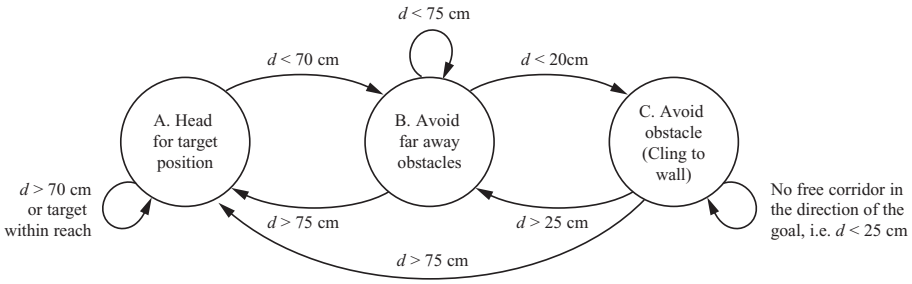


Fig. 6. This illustrates the states used in the obstacle avoidance. d denotes the free space in the corridor in the direction of the target position. Hystereses are used to avoid oscilations between states.

A. Head for target position. If there is more than 70 cm of free space d in the corridor in the direction of the target (which is determined using the "corridor"-analysis function), the robot turns in that direction. It avoids running into obstacles while turning using the "free space for turning"-analysis function. The forward speed of the robot is determined by the amount of turning necessary: if little turning is necessary, the forward speed is maximal, if it has to turn a lot, the speed is throttled.

B. Avoid far away obstacles. If the free distance d is less than 70 cm, the robot turns away from the obstacle using the "Next free angle" analysis function.

C. Avoid close obstacles. If obstacles are close, the robot clings to the obstacle and performs a wall following behavior. This is achieved in the following way: the robot tries to turn toward the direction of the target. The "free space for turning"-analysis function is used to keep it from running into the obstacle, causing the robot to closely cling to the obstacle. As soon as the obstacle is circumvented and the corridor to the target is free, the robot turns toward the target. The forward speed of the robot is throttled according to how close obstacles are in the direction the robot is currently moving in. If obstacles in that direction become closer than a threshold of 5 cm, the robot backs off.

Obstacle avoidance in RoboCup games. In the championship games, obstacle avoidance was also used. It was used in conjunction with a force field approach to allow for various control systems to run in parallel. The obstacle model was also used for shot selection. When the robot was close to the ball, the model was used to check if there were obstacles in the intended direction of the shot. If there were obstacles in the way, countermeasures were taken (such as turning away from the obstacle).

Scanning motion of the head. In the challenge, the robot performed a scanning motion with its head. This gives the robot effective knowledge about its vicinity (as opposed to just its field of view), allowing it to better decide where it should

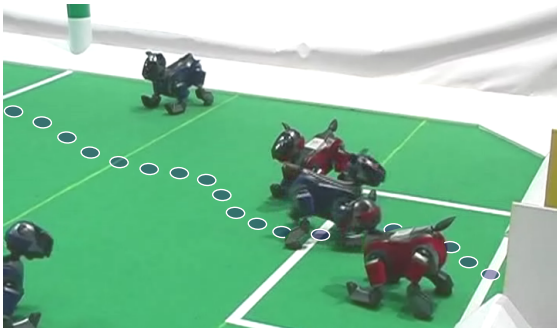


Fig. 7. The above image was extracted from a video of the obstacle avoidance challenge run of the GermanTeam. The robot's path is shown. It is noteworthy that the corridor between the two robots new the goal is only about 3 times wider than the robot's lateral dimensions and about 1.5 times wider as its length.

head. The scanning motion and the obstacle avoidance behavior were fine tuned to allow for a wide scan area while making sure that the area in front of the robot was scanned frequently enough for it to not run into obstacles.
In the actual RoboCup games, the camera of the robot is needed to look at the ball most of the time. Therefore, very little dedicated scanning motions were possible giving the robot a slightly worse model of its surroundings.

3 Application and Performance

RoboCup 2003 Technical Challenge. In the obstacle avoidance challenge, a robot had to walk as quickly as possible from one goal to the other without running into any of the other 7 robots placed on the field. The other robots did not move and were placed at the same position for all contestants. The algorithm used was only slightly altered from the one used in the actual, dynamic game situations. As can be seen from the results (table 1), the system used enabled the robot to move quickly and safely across the field. Avoidance is highly accurate: on its path, the robot came very close to obstacles (as close as 2 cm to touching the obstacles) but did not touch any of them. Very little time is lost for scanning the environment (as the obstacle model is updated continuously while the robot's head is scanning the surroundings) enabling the robot to move at a high speed without stopping. The system used in the challenge was not optimized for speed and only utilized about 70% of the robot's top speed. Furthermore, some minor glitches in the behavior code caused the robot to not move as fast as possible.

RoboCup 2003 championship games. The obstacle model was used for obstacle avoidance and for shot selection in the games. An improvement in game play was noticeable when obstacle avoidance was used. In several instances during the games, situations in which the robot would otherwise have run into an opponent

Rank	Team	Collisions	Zone A	Zone B	Zone C	Zone D	Zone E	Goal (Total Time)
1	GermanTeam	0	6.31s	10.16s	16.52s	20.73s	29.83s	35.76s
2	UT Austin	0	10.74s	20.81s	32.82s	41.11s	59.31s	63.38s
3	ARAIBO	0	10.71s	32.20s	47.20s	72.22s	80.53s	104.45s
4	UTS Unleashed	0	15.99s	62.49s	73.13s	83.24s	94.46s	108.72s
5	ASURA	1	10.35	17.19s	24.81s	30.92s	40.51s	87.22s
6	rUNSWift	1	9.95	15.09s	27.83s	36.75s	46.35s	100.09s
7	Baby Tigers	2	15.01s	30.22s	50.41s	70.23s	94.32s	141.53s
8	Team Sweden	2	10.97s	19.82s	31.06s	41.32s	53.63s	179.97s
9	NUbots	1	5.89s	13.56	36.86s	45.69s	54.81s	(not reached)
10	UW Huskies	1	45.05s	55.66s	59.12s	62.53s	72.44s	(not reached)

Table 1. RoboCup World Cup 2003 obstacle avoidance challenge results. Only the results of the first of the 24 participating teams are shown. Note that some of the teams did not reach the goal at all. The ranking is based on speed and the number of collisions.

it was able to steer around it. In addition to using the ego-centric obstacle model, communicated global position information was used to make sure that no two robots of our team tried to reach the same target position. Shot selection did not yield a positive effect mostly because realigning the robot to shoot at a better angle took too long. By the time the robot had turned, opponent robots would have closed in on it rendering the desired kick useless.

Issues. Small errors in odometry and image processing have, of course, a negative effect on the model. Since no global model is produced and the model is updated frequently, no negative effects on performance were observed. Bad color calibration, however, can keep the robot from walking anywhere if it "sees" obstacles all around it. The models accuracy is diminished by the fact the robots camera shakes quite a bit when the robot is walking. Therefore, the absolute height of the camera and its orientation differ from the assumed parameters which are calculated using forward kinematics.

4 Conclusion

The presented system enables the robot to reliably circumvent obstacles and reach its goal quickly. The system was developed for use in highly dynamic environments and limits itself to local obstacle avoidance.

In the RoboCup 2003 obstacle avoidance challenge, the robot reached the goal almost twice as fast as the runner up without hitting any obstacles. In the challenge, the system was not used to its full potential and a further increase in speed is well within reach. An improvement in game play in the RoboCup championship games was observed although this is very hard to quantify and

depended largely on the opponent. The GermanTeam reached the quarter final of the competition.

Although the presented system is based on vision data and was optimized for the application in the RoboCup domain, integration of other sensor data is easily done (such as the Aibo's infra red distance sensor). Using the system in other environments can be achieved by adjusting the domain specific heuristics in the obstacle detection module.

5 Acknowledgments

The project is funded by Deutsche Forschungsgemeinschaft, Schwerpunktprogramm "Kooperierende Teams mobiler Roboter in dynamischen Umgebungen" (Cooperative Teams of Mobile Robots in Dynamic Environments).

Program code was developed by the GermanTeam, a collaboration of the Humboldt University Berlin, University of Bremen, University of Dortmund, and the Technical University of Darmstadt. Source code is available for download at http://www.robocup.de/germanteam.

References

1. R. Benosman and S. B. K. (editors). *Panoramic Vision: Sensors, Theory, and Applications*. Springer, 2001.
2. V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1984.
3. M. Jüngel, J. Hoffmann, and M. Lötzsch. A real-time auto-adjusting vision system for robotic soccer. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004. to appear.
4. O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1), 1986.
5. S. Lenser and M. Veloso. Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision. In *Proceedings of IROS'03*, 2003.
6. G. Mayer, H. Utz, and G. Kraetzschmar. Towards autonomous vision self-calibration for soccer robots. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2002.
7. H. Noborio, K. Fujimura, and Y. Horiuchi. A Comparative Study of Sensor-Based Path-Planning Algorithms in an Unknown Maze. pages 917–924, 2000.
8. T. Röfer, I. Dahm, U. Düffert, J. Hoffmann, M. Jüngel, M. Kallnik, M. Lötzsch, M. Risler, M. Stelzer, and J. Ziegler. GermanTeam 2003. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004. to appear. more detailed in http://www.robocup.de/germanteam/GT2003.pdf.
9. T. Röfer and M. Jüngel. Vision-Based Fast and Reactive Monte-Carlo Localization. *IEEE International Conference on Robotics and Automation*, 2003.
10. T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J.-S. Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski. CS Freiburg 2001. 2003.