A Vision Based System for Goal-Directed Obstacle Avoidance

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

Obstacle avoidance is an important problem for any mobile robot. While being a well studied field, it remains a challenging task to build a robust obstacle avoidance system for a robot using vision data.

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, 9]. Using these sensors, a full panoramic view is always available which greatly simplifies the task. In detecting obstacles from vision data, heuristics can be employed such as the "background-texture constraint" and the "ground-plane constraint" used in the vision system of the robot Polly [3]. In the case of the RoboCup world, this means that free space is associated with green (i.e. floor color), whereas non-green colored pixels are associated with obstacles (see introduction of [6] for an overview of panoramic vision systems).

In the Sony League, the robot is equipped with a camera with a rather limited field of view. As a basis for obstacle avoidance, a radial model of the robot's environment is maintained. In this model, current vision data is integrated with recent vision data. The approach to obstacle detection and obstacle modeling used by the GermanTeam in the RoboCup challenge turned out to be similar the concept of "visual sonar" recently presented by [6]. Both bear a strong resemblance to the "polar histogram" used in [2]. Our work extends [6] and shows how such a model can be used to achieve goal-directed obstacle avoidance. It proved highly robust and performed extremely well in dynamic game situations and the obstacle avoidance challenge.

Other approaches such as using potential fields [5] were not considered because the robot's environment changes rapidly which makes it hard to maintain a more complex world model.

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 [8]; other modules not covered here such as walking engine, etc. are described in more detail in the GermanTeam 2003 team description and team report [7].

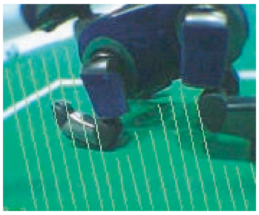
2.1 Obstacle Detection

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 automate this process and allow for real-time adaptation exist [4]).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 inter-



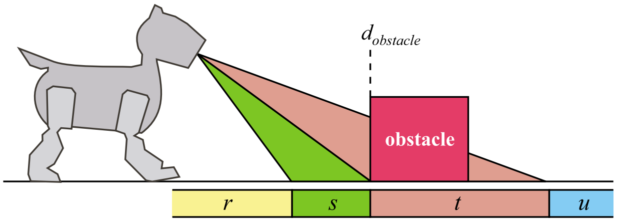


Fig. 1. Obstacle detection and diagram to illustrate what can be deduced from what is seen. Green lines in image: The obstacles percept as the conjunction of green segments close to the robot. Diagram: The robot detects some free space in front of it (s) and some space that is obscured by the obstacle (t). The obstacle model is updated according to the diagram (in this case the distance in the sector is set to $d_{obstacle}$ unless the distance value stored lies in r)

preted 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 1 shows how different parts of scan lines are used to generate obstacle percepts and also illustrates how information about obstacles in the robot's field of view can be deduced from the *obstacle percept*.

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 it is a different type of problem that needs to be dealt with by higher levels of action planning. We therefore concentrate 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" [6]. 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").

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

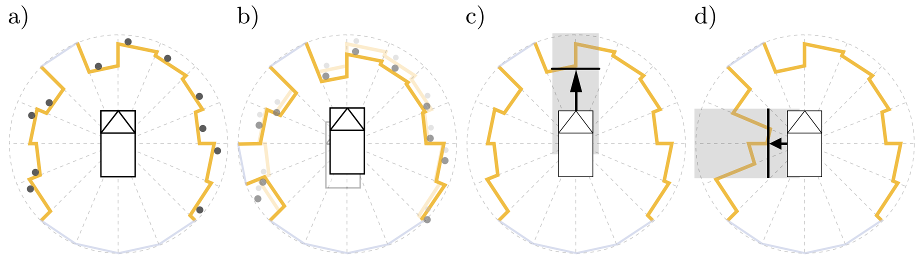


Fig. 2. Illustration of the obstacle model. The actual number of sectors is greater than shown here, it was reduced for illustration purposes. Fig. 3 shows the actual obstacle model used. a) The robot is at the center; dashed lines show sectors; solid orange lines (dark) show the free space around the robot; light gray lines are used if there is no information about free space in a sector; 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. c) and d) illustration of analysis function used when determining the free space in front of the robot and to its side

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. 2 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. 1 illustrates one of the many 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.

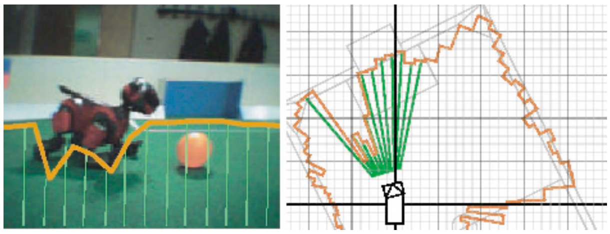


Fig. 3. *Left.* Camera image with superimposed obstacle percepts and obstacle model (projected onto the floor plane) *Right.* Actual obstacle model

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.2 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 performance problems in our experiments, [6] shows how these gaps can easily be closed using linear interpolation between formerly adjacent sectors.

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 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 Obstacle Avoidance

Goal-directed obstacle avoidance as used in the challenge. Obstacle avoidance is achieved by the following control mechanisms:

A. **Controlling the robot’s forward speed.** The robot’s forward speed is linearly proportional to the free space in the corridor in front of the robot.

B. **Turning towards where there is more free space.** If the free space in the corridor in front of the robot is less than a threshold value, the robot will turn towards where there is more free space (i.e. away from obstacles).

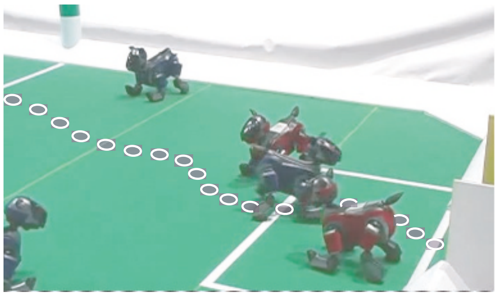
C. **Turning towards the goal.** The robot turns toward the goal only if the space in front of it is greater than a threshold value.

D. **Override turning toward goal.** If there is an obstacle next to it that it would run into while turning, turning is omitted and the robot will continue to walk straight.

When approaching an obstacle, *B.* causes the robot to turn away from it just enough to not run into the obstacle. *C.* and *D.* cause the robot to cling to a close obstacle, thereby allowing the robot to effectively circumvent it.

Obstacle Avoidance in RoboCup Games. In the championship games, a similar obstacle avoidance system was used. It worked in conjunction with a force field approach to allow for various control systems to run in parallel. The obstacle model itself was 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, it would try to shot the ball in a different direction.

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 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.



Rank	Team	# Collisions	Time [s]
1.	GermanTeam	0	35.7
2.	UT Austin	0	63.3
3.	ARAIBO	0	104.4
4.	UTS Unleashed	0	108.7
5.	ASURA	1	87.2
6.	rUNSWift	1	100.0
7.	Baby Tigers	2	141.5
8.	Team Sweden	2	179.9
9.	NUbots	1	(not reached)
10.	UW Huskies	1	(not reached)

Fig. 4. Image extracted from a video of the RoboCup World Cup 2003 obstacle avoidance challenge and table of results

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, 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 it was able to steer around it.

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 our search for the simplest, most robust solution to the problem, maintaining a model of the obstacles was a necessity to achieve high performance, i.e. to alter the path of the robot as little as possible while assuring the avoidance of obstacles currently visible *and* invisible to the robot. The control mechanisms make use of this model to achieve the desired robot behavior. In the RoboCup 2003 obstacle avoidance challenge, the robot reached the goal almost twice as fast as the runner up without hitting any obstacles. The system was not used to its full potential and a further increase in speed has since been achieved. An improvement in game play in the RoboCup championship games was observed although this is very hard to quantify as it depended largely on the opponent.

Acknowledgments

The project is funded by Deutsche Forschungsgemeinschaft, Schwerpunktprogramm 1125. Program code is part of the GermanTeam code release and 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. J. Borenstein and Y. Koren. The Vector Field Histogram Fast Obstacle Avoidance For Mobile Robots. In *IEEE Transactions on Robotics and Automation*, 1991.
3. I. Horswill. Polly: A Vision-Based Artificial Agent. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*, 1993.
4. 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.
5. O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1), 1986.
6. S. Lenser and M. Veloso. Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision. In *Proceedings of IROS'03*, 2003.
7. 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.
8. T. Röfer and M. Jüngel. Vision-Based Fast and Reactive Monte-Carlo Localization. *IEEE International Conference on Robotics and Automation*, 2003.
9. T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J.-S. Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski. CS Freiburg 2001. In *RoboCup 2001 International Symposium*, Lecture Notes in Artificial Intelligence. Springer, 2003.