STEREO IMAGE BASED COLLISION PREVENTION USING THE CENSUS TRANSFORM AND THE SNOW CLASSIFIER

Christian Küblbeck, Roland Ach, Andreas Ernst
Fraunhofer Institute for Integrated Circuits
Am Wolfsmantel 33

Keywords: stereo, obstacle detection, census transformation, SNoW classifier.

Abstract: In this paper we present an approach for a mobile robot to avoid obstacles by using a stereo-camera system mounted on it. We use the “census transformation” to generate the features for the correspondence search. We train two SNoW (Spare Network of Winnovs)-classifiers, one for the decision wether to move straight forward or to evade and a second one for deciding whether to turn left or right when evading. For training we use a sample set collected by manually moving around with the robot platform. We evaluate the performance of the whole recognition chain (feature generation and classification) using ROC-curves. Real world experiments show the mobile robot to safely avoid obstacles. Problems still arise when approaching steps or low obstacles due to limitations in the camera setup. We propose to solve this problem using a stereo camera system capable of pan and tilt movements.

1 INTRODUCTION

The detection of obstacles is an important research topic in the field of autonomous robots. Collisions have to be avoided, since robots must neither endanger human beings through their movement nor damage themselves or objects in the environment. The aim of the present work is to develop a system for obstacle detection and collision avoidance for an autonomous robot with the aid of a stereoscopic camera setup. Further on aiming at a later 3D-map generation of the environment an unsupervised exploration trip through a building should be permitted. For that purpose appearing obstacles like walls, objects or persons must be detected.

1.1 Overview

The robot is based on a mobile platform named *VolksBot* [1] that consists of a main frame with the dimensions 40 x 40 x 80 cm. The drive mechanism is made up of two DC-motors with 22 W power and 24 V nominal voltage. The two drive wheels forms a triangle together with a third rear wheel. On the lower platform a standard notebook with an 1.6 MHz mobile processor is placed. The communication with the motor controller is established by the serial RS232-Interface.

[1] The *VolksBot* has been developed by the Fraunhofer Institute for Autonomous Intelligent Systems AIS in Sankt Augustin, Germany.

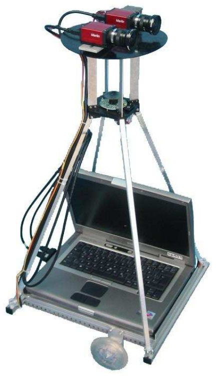
Figure 1: The robot based on the mobile platform *VolksBot*.

For stereo image capturing, we use two firewire cameras. They are mounted parallel to each other on a small upper platform at a height of 80 cm. The optical axes are arranged in a distance of approximately

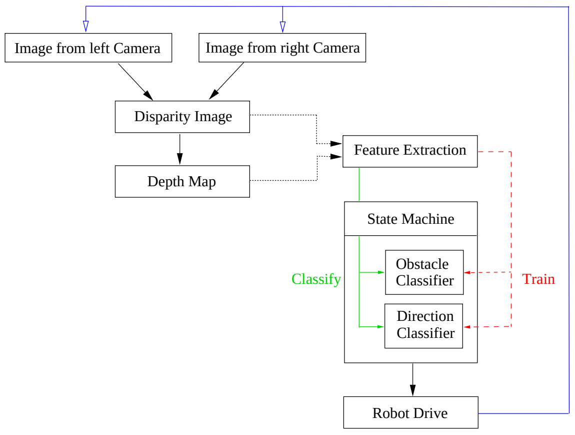

Figure 2: System structure.

10 cm. Both cameras are equipped with a wide angle lens with 4.2 mm of focal length.

Figure 2 shows the structure of the system setup. Two cameras are used for image capturing. We are scaling down the images to a size of 320 x 240 pixels and only use one single color channel. A pair of images is grabbed approximately every 240 ms. Thus about four frames per second are processed. In order to extract depth-information, a disparity image is determined by means of block matching. To reduce noise artifacts smoothing filters like mean or median are applied to the original images. Additionally we use the census transform to get suitable images for the subsequent block matching. Using a lookup table derived from the camera geometry, we can calculate a depth map from the disparity image. These results serve as the basis for the feature extraction. Hence the complete disparity image or depth map can be used. It is also possible only to use the distance of the nearest obstacle for each picture column of the depth map. Alternatively height information of the objects situated in the area can be included.

Based on the classification results the robot is controlled in its movements. We are using two *state machines*. Therefore, two classifiers are integrated. The first one recognizes obstacles that have to be avoided. If an obstacle is detected, the second classifier is asked for a preferred evasive direction. This direction is retained until a movement straight forward is possible again.

1.2 State of the Art

Most solutions of obstacle avoiding robots use laser range sensors, ultrasonic or infrared sensors, but there are also realizations using cameras. Especially 2-camera-systems often serve as a basis for the obstacle detection.

In this sense, Kumano, Ohya and Yuta (Masako Kumano and Yuta, 2000) follow a very simple approach. Using two cameras directed in a fixed angle towards the ground, images of the environment are obtained. Starting from the geometrical setup between the cameras and the floor, corresponding points in both images are determined. Intensity differences between corresponding points in both images are assumed to be an obstacle in the way. The scan lines correspond to different distances according to the camera arrangement. Evaluating merely three scan lines that represent the distances 40 cm, 65 cm and 100 cm, a fast computing time (35 ms) can be guaranteed.

Sabe, Fukuchi, Gutmann, Ohashi, Kawamoto and Yoshigahara (Kohtaro Sabe et al., 2004) also use two CCD-cameras. These are integrated in the humanoid *Sony QRIO Robot*. Firstly landmark points of the environment are searched in both images. These points appear in different picture coordinates according to the different camera views and the distance. Using these disparities, 3D coordinates can be computed. On this basis the robot detects possible obstacles on the floor level, that are used for path planning.

The electrical wheelchair *Victoria* (Libuda and Kraiss, 2004) computes a depth map from the included scene with the aid of a camera pair. Additional corners and edges are determined in different detection stages and grouped to regions which correspond to certain objects in the room. Using fixed relations between these landmarks a model of the surroundings can be produced with the corresponding distance information. One can search the walls for regions like doors and the floor for obstacles.

There is still a number of further approaches with two camera systems. For example the *NEUROS* project at the Ruhr University of Bochum with its visually controlled service robot Arnold uses such a construction (Ruhr-Universität, 1997). Arnold extracts edges from both images and calculates their distances by the shift of these landmarks. Hence apart from an obstacle detection, card generation of the environment is possible. In unknown areas the robot turns by 360 degrees to get the necessary data. To evade people optical flow is included.

Another example is the *Ratler* (robotic all terrain lunar exploration rover) (Reid Simmons and Whelan, 1996). It moves in an unknown area with a height-map of the environment. This map is generated by the extraction of 3D-points from a stereo-camera-system.

Daimler Chrysler develops a system for the detection of pedestrians (D.M. Gavrila and Munder, 2004). At first a disparity image indicates potential areas. By an edge extraction of these regions a comparison with sample forms of persons is possible.

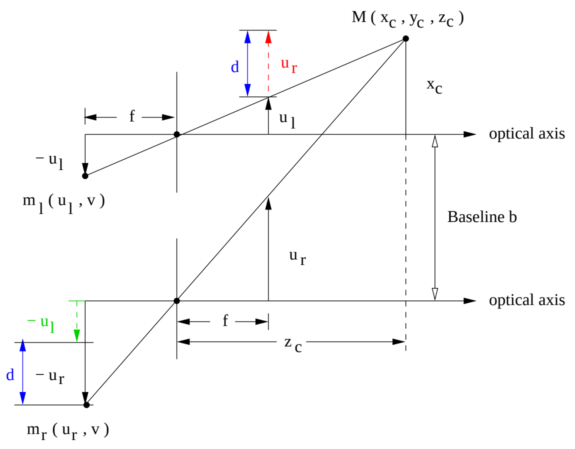

Figure 3: Simple model of the stereo camera system.

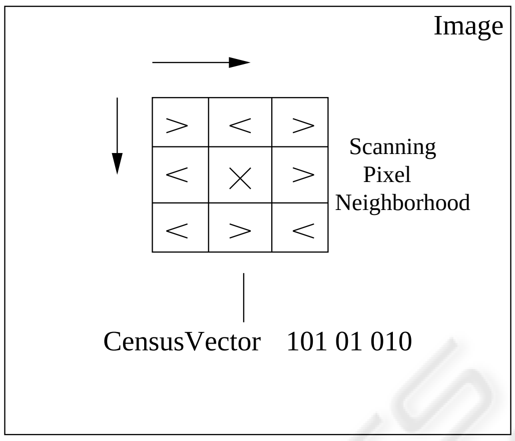

Figure 4: The census transform: Each pixel is compared with its eight neighbours. From the binary results of the comparisons a feature of one byte is constructed.

2 FEATURE GENERATION

2.1 Camera Setup and Stereo Geometry

A standard camera projects three-dimensional objects of the real world onto the 2D image plane. A very simple model of this process is given by the pinhole camera. In this case the optic consists of one infinitely small hole.

A simple possibility to recover depth information of the captured scene is an extension of a single camera by a second identical camera with same focal length. The visual axes of both cameras are arranged in parallel and on the same level. In figure 3 a point M of the real world is projected to different coordinates $m_r(u_r, v)$ and $m_l(u_l, v)$ on the image planes because of the varying camera viewpoints. The difference is called disparity d. Using the knowledge about the geometrical camera setup and the focal length f the distance z_c of the point M from the camera plane can be computed by means of the disparity. It is inversely proportional to the distance.

Depending on the distance b of the two camera axis the following relations can be derived from the stereo-geometry:

$$d = u_r - u_l = \frac{f * (x_c + b)}{z_c} - \frac{f * x_c}{z_c} = \frac{f * b}{z_c}. \quad (1)$$

Using the disparity d the distance z_c can therefore be computed by

$$z_c = \frac{f * b}{d} = \frac{f * b}{u_r - u_l}. \quad (2)$$

The standard stereo geometry assures that arbitrary points of the 3D scene are always projected into the same line in both images. For this reason only horizontal disparities appear. This restriction is called *epipolar constraint*. Hence the search area finding corresponding points in both image planes can be reduced to one dimension.

It is important to mention that disparities can be determined only for points from objects that can be observed in both images. With increasing camera distance b this overlap becomes smaller.

2.2 The Census Transform

In order to compute disparity informations it is necessary to find corresponding points in both images. It has to be considered, that the internal attitudes of the two cameras can differ, regarding gain and biases. Therefore corresponding regions may have different absolute intensities. To solve this problem we use the *census transform* proposed by Zabih and Woodfill. It offers a brightness invariant comparison possibility. Additionally it provides an efficient and fast methods to extract features for structured image regions that can serve as feature points.

The census transform compares each pixel with its eight neighboring pixels. If the intensity of the center is greater, the appropriate bit of the census feature is set, otherwise cleared. Therefore the census feature has eight bit and can be used to represent a grey tone in a new “census-image”. The shift of the window over the entire image results in a complete census image. This procedure is applied to both camera images.

The distance between two census features can be computed using the hamming distance. It is calculated by counting the number of bits that differ within both census features.

To find corresponding points for each area of the

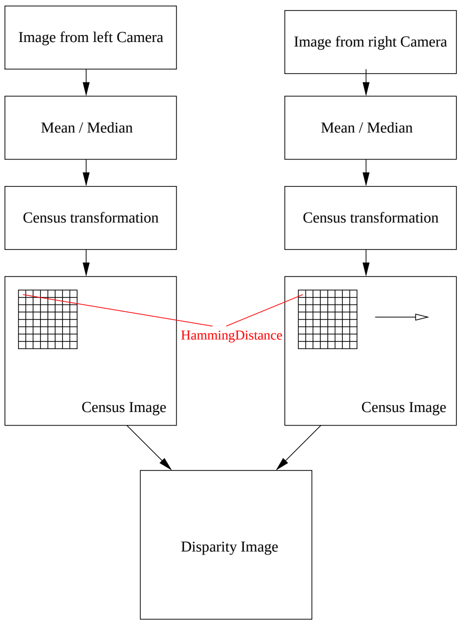

Figure 5: Generation of the disparity image.

first census image every point is compared to each region in the second census image within the same image line. The hamming distance between all pairs of points is calculated and summed up for the complete area. The minimum distance for each pixel corresponds to the most similar picture region.

2.3 Generation of Disparity Maps and Features

As figure 5 shows, the original camera images are filtered at the beginning of the process, alternatively with a mean or median filter of size 3×3. Then the census transform is used to find corresponding pixels in both images.

To generate the disparity image a block matching procedure is used. This method takes a pixel neighborhood to find similar areas. The size of this region is parameterizable.

The best disparity is estimated at the points where the hamming distance reaches its minimum. Additionally the applied algorithm uses a so-called *confidence procedure* which describes the reliability of the estimation. This is done by computing firstly the hamming distance at the estimated disparity and the average of discrete similarity values obtained shifting the window along the image line. If the distance between both is smaller than a fixed confidence threshold the result is discarded and the appropriate pixel in the disparity picture is set to zero. This method affects particularly weakly textured, homogeneous areas where

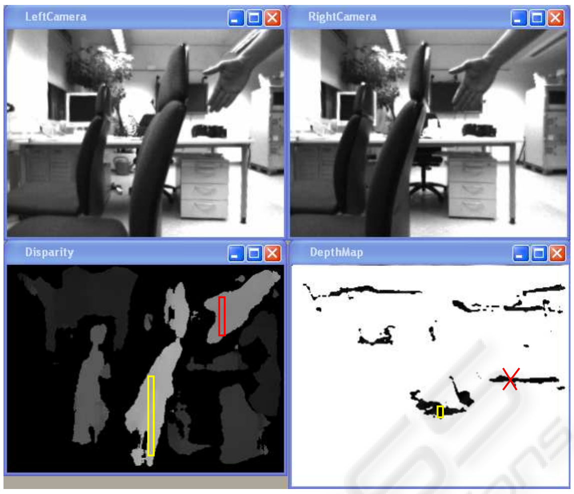

Figure 6: Disparity image and depth map.

wrong assignments occur rather often. Figure 6 shows an example of a disparity image. Brighter parts of the image correspond to closer ranges, dark regions belong to more distant objects in the scene.

From the disparity image a kind of bird's eye view of the captured scene can be generated by summing up the existing disparity values for each column according to their occurrence by registering them in a depth map. The y-axis represents the disparities, the x-axis the picture column. The value contains information about the vertical expansion of the objects. The objects can be found according to their order in the depth map. With help of the stereo geometry and equation 2 the distance can be determined in real metric numbers. So the two office chairs are about 54 cm and 104 cm away from the cameras. The hand is found approximately 69 cm in front of the camera. As we will see later this knowledge is not necessary for our approach. The critical distance for objects in front of the robot is implicit given by the disparities of the training data containing an obstacle. Of course the reliability of the depth map strongly depends on the quality of the disparity image.

Human beings are able to recognize directly from the depth map whether an obstacle is in a critical range. In opposite a computer has to perform an additional preprocessing step for feature extraction prior to the classification. For this purpose a very simple approach is coding the respective pixels of the disparity image or depth map with coordinates and grey tones in a characteristic feature vector. Of course the feature vector and each feature respectively, dependent on the dimensions of the image, can become very large. In order to handle these data the images can be scaled down. On basis of a size of 320×240 a reduction to 80×60 makes sense. Besides the above

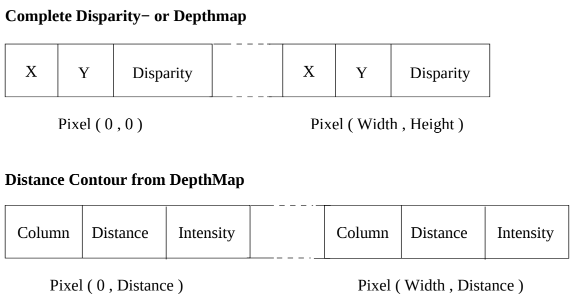

Figure 7: Possible encodings of the feature vectors.

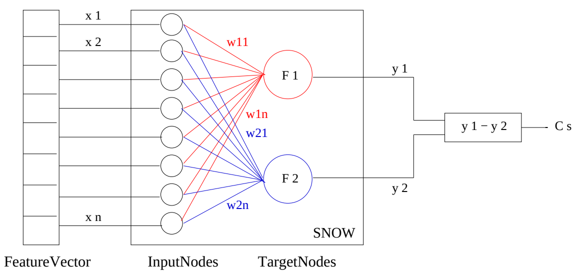

Figure 8: Architecture of the SNoW-classifier

rather simple approach it is also possible to use only one value representing the nearest obstacle for each column of the depth map. With respect to the whole image pair thus a distance contour of the objects arises from the field of vision of the robot. Figure 7 illustrates both methods of feature encoding.

3 CLASSIFICATION WITH SNOW

The classification has to categorize a characteristic feature vector by comparison with reference samples. Due to the very complex features a neural network which can build high-dimensional separation functions between the individual classes is used. In this work we use a special network named *SNoW* (Sparse Network of Winnows)(Yang et al., 2000). The SNoW is characterised by high detection achievement and speed. Therefore it is used in speech processing or in the field of face detection. In this case the classifier has to decide between the classes ’forward’, ’turn left’ and ’turn right’. Hence, two classifiers are used. The first one decides between free and obstacle, the second one suggests a suitable evasive direction if it is necessary.

The architecture of the network consists of a layer of input nodes and a layer of target nodes. One target node represents a special class. Both levels are connected by weighted edges. The number of input nodes

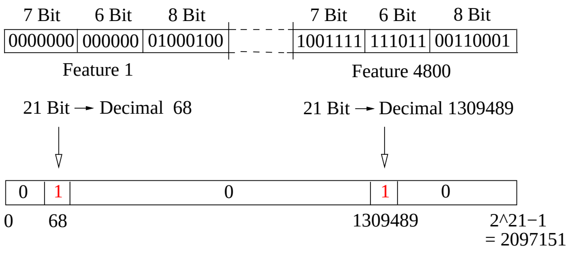

Figure 9: SNoW-feature vector

depends on the binary representation of the characteristic feature vectors. Figure 9 shows such a structure for a pixel-coded depth map with a dimension of 80×60 pixels. The vector length depends on the number of bits per feature: $length = 2^{\text{bits per feature}}$. In this case each feature represents a position within the vector according to its binary value. In these places the vector takes the value ’1’. The remaining positions are set to ’0’.

Vector entries with the value 1 are called active characteristics. Only these components are attached over weights to the target nodes t. The activation F_t of these elements is determined over the sum of the connected weights according to equation 3. However, these nodes are active only if their activations exceed a certain class threshold Θ_t:

$$\Theta_t < F_t = \sum_{i \in A_t} w_i^t \qquad (3)$$

A_t here is the amount of all active features i connected to the node t: $A_t = \{i_1, ..., i_m\}$.

The weights are at first uniformly determined. During the training phase an adjustment takes place only if a false classification happens. The update strategy is based thereby on a promotion parameter $\alpha > 1$ for increasing and a demotion parameter $0 < \beta < 1$ for reducing the values.

With well-known input vectors $\widehat{t}$ for the class t and the quantity of the active features m_j the following relationships are established:

$$\text{if } \widehat{t} = t \text{ and } F_t(m_j) \le \Theta_t : \quad \forall_i \in A_t : w_t^i \leftarrow \alpha * w_t^i \qquad (4)$$

$$\text{if } \widehat{t} \ne t \text{ and } F_t(m_j) \ge \Theta_t : \quad \forall_i \in A_t : w_t^i \leftarrow \beta * w_t^i \qquad (5)$$

The values supplied by each target node, also called scores are charged with each other. Given that we only have two-category problems the affiliation is carried out using the score difference $c = F_1 - F_2$. So a special decision instance is used that can be parametrized with a threshold. Thus a sensitivity regulation for a certain class becomes feasible.

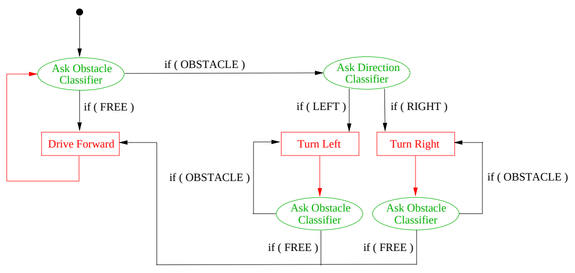

Figure 10: State machine.

The decision for class 1 is made if $c + Sensitivity > 0$ and similarly the decision for class 2 is made if : $c + Sensitivity \leq 0$

In order to guide the robot, control commands must be sent to the motor controller permanently. The coordination of these instructions needs a *state machine* presented in figure 10. This machine is in one of the three possible states 'forward', 'turn left' or 'turn right' at each time. The transitions are fixed exactly and are followed according to the appearing events. Each instance contains one SNoW-classifier for the obstacle detection and one for the direction decision, respectively. At the beginning an image pair is examined and it is decided, whether an obstacle is in front of the robot. If this is true the second classifier is asked about a suitable evasive direction. The unity therefore represents the condition 'turn left' or 'turn right'. This decision has to be maintained until the obstacle detector indicates a free way again. Then the machine again represents the condition 'forward'.

4 EXPERIMENTS AND RESULTS

The training data was collected by controlling the robot manually. Normally the robot moves straight on. If there is an obstacle in front of it a change of direction can be enforced by pressing the left or right mouse button depending on the best way to elude. This movement is maintained till the button is released. In this case the vehicle goes straight on again.

During the complete journey the system saves the two current camera images on hard disk. This happens approximately at a rate of five frames per second. The size of the images is 320×240 pixel using only grey level format. Hence the file size is reduced to about 40 kB. In order to mark obstacles the filename consists of a time stamp and the current control information like forward, left or right.

This way approximately 100,000 training images were recorded. 39552 pairs correspond to the class 'forward', 9342 couples containing obstacles. To find alternate directions 4788 pairs represent turns to the left, 4554 turns to the right. This data is grouped in two different lists. On the one hand all files are taken up to an obstacle list, whereby the classes 'turn left' and 'turn right' correspond to an obstacle. On the other hand a direction list covers all image pairs which are not part of the class 'forward'. These two lists are used for the training of the obstacle classifier and direction classifier respectively.

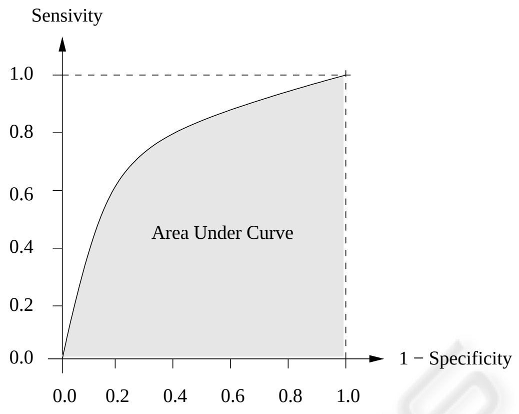

Figure 11: ROC-curve.

Depending on preprocessing and feature extraction the training takes up to eight hours on standard hardware (with a 1.5 GHz processor). Both classifiers are trained separately. For evaluation purposes further test data has been collected. It consists of 14447 image pairs divided in 13119 with the label 'forward', 790 of the class 'turn left' and 538 of the class 'turn right'. By comparison of the well known class memberships with the classification result a statement about the quality of the trained classifiers can be met.

With the search for a proved decision criteria for the optimization of the preprocessing and feature extraction parameters like block size, search width, confidence threshold, mean or median filtering of the depth map, etc. the choice felt on the so-called ROC curve. The ROC method applies the fraction of correct classified obstacles to the y-axis of a diagram. On the x-axis the ratio of wrong classified free areas is registered. These informations are based on sample data. Figure 11 shows a sample ROC curve. The appearance of the curve is based on different decision borders. On the left side the curve starts with a very high separative work. One surveys everything, however no images of the class 'forward' were classified falsely as an obstacle. The lowest decision limit lies on the right side. One does not survey anything, since one categorizes all inputs as obstacle. In view on the choice of these separative borders it is necessary to look on the scores determined by the obstacle classifier for both classes.

The area below the curve is a measure for the qual-

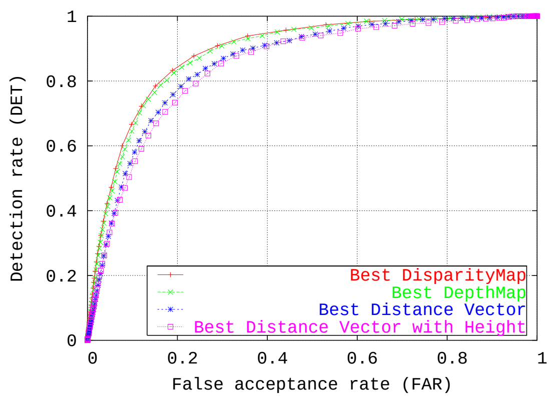

Figure 12: ROC-curves for obstacle classifier.

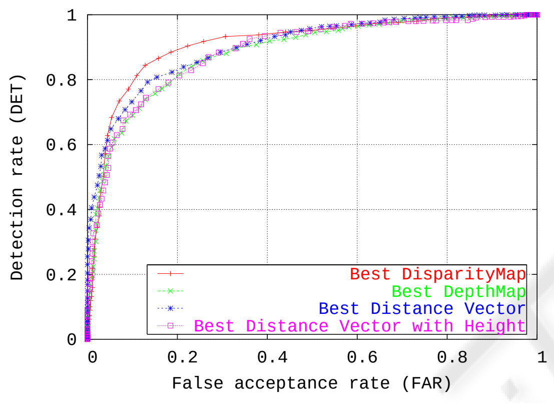

Figure 13: ROC-curves for direction classifier.

ity of the classifier. The curve ideally arises vertically up on the left. So the curve forms a square of area 1 with the coordinate axes. The area under the ROC curve serves as a criterion for the improvement of the adjusted parameters. The more this value reaches 1, the better a separation into the two classes is possible.

Like the obstacle classifier, the direction classifier uses the ROC method to optimize the parameters. Based on sample images correct classified pictures of the class 'turn left' lay over wrongly assigned pictures of the class 'turn right'.

Despite all improvement some pairs of images from the sample are still categorized falsely. Examining more closely these images makes clear, that the majority of wrong classified samples are quite problematic. Some pictures are almost without any lighting. Furthermore reflecting surfaces or images with sparse texture (for example white walls) cause certain difficulties. The main problem however lies in the cameras mounted parallel to the ground. Since they are fixed at the height of 80 cm smaller objects on the floor are not in the field of view and therefore

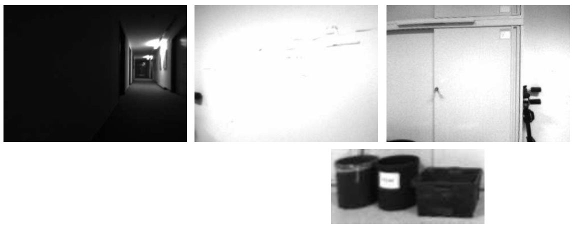

Figure 14: Examples of wrong classified images.

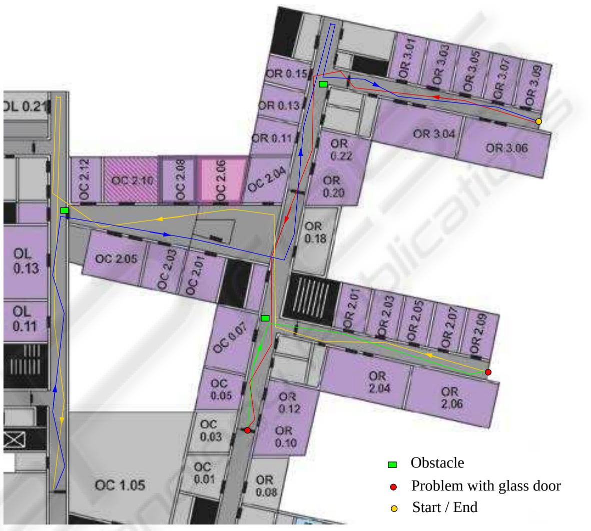

Figure 15: Travel through a building.

cannot be detected. During the generation of test data by means of a mouse control the evade of these obstacles is guaranteed by the supervisor. But these objects are not seen within the image. So the classifier is not able to make the right decision. For this reason the cameras should be operated looking to the ground. Figure 14 shows some examples of wrong classified images.

5 CONCLUSION AND OUTLOOK

Figure 15 represents as an example a travel of the robot through our institute building. The best classifiers are used. Problems occured only in front of glass doors.

In order to make the system more robust, the cameras should be installed with view to the ground. The optimum would be a mobile camera head. Its adjustment could be included in the feature vectors too. Thus also smaller objects could be detected such as legs of a chair or boxes standing on the floor. At the current setup smaller objects disappear when moving towards them in critical distances.

Furthermore sequences with different lighting conditions or shades could be generated to consider these influences particularly. Also a speed control could be installed by mapping the score differences of the obstacle classifier. The robot could accelerate if no obstacles are seen and slow down if objects are near. At collisions an image sequence could be saved to retrain and improve the classifier after the event.

REFERENCES

D.M. Gavrila, J. G. and Munder, S. (2004). Vision-based pedestrian detection: The protector system. Daimler Chrysler Research , Ulm.

Kohtaro Sabe, Masaki Fukuchi, J.-S. G., Ohashi, T., Kawamoto, K., and Yoshigahara, T. (2004). Obstacle avoidance and path planning for humanoid robots using stereo vision. Sony Corporation, Tokyo.

Libuda, L. and Kraiss, K.-F. (2004). Identification of natural landmarks for vision based navigation. Technische Universität Aachen.

Masako Kumano, A. O. and Yuta, S. (2000). Obstacle avoidance of autonomous mobile robot using stereo vision sensor. University of Tsukuba.

Reid Simmons, Lars Henriksen, L. C. and Whelan, G. (1996). Obstacle avoidance and safeguarding for a lunar rover. Carnegie Mellon University.

Ruhr-Universität, B. (1997). Arnold, ein autonomer service-roboter. http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PROJECTS/NEUROS/ff97-katalog/arnold_d.html.

Yang, M.-H., Roth, D., and Ahuja, N. (2000). A snow-based face detector. In *Advances in Neural Information Processing Systems 12 (NIPS 12)*, pages 855–861. MIT Press.