VISION-BASED HUMANOID NAVIGATION USING SELF-SUPERVISED OBSTACLE DETECTION

DANIEL MAIER

CYRILL STACHNISS

MAREN BENNEWITZ

Department of Computer Science, University of Freiburg,
Georges-Köhler-Allee 079, 79110 Freiburg, Germany

{*maierd,stachnis,maren*}*@informatik.uni-freiburg.de*

Received 27 Decemeber 2011
Accepted 29 January 2013
Published 28 June 2013

Int. Journal of Humanoid Robotics (IJHR), Vol. 10 (2), 2013
Preprint, final version available at DOI 10.1142/S0219843613500163

In this article, we present an efficient approach to obstacle detection for humanoid robots based on monocular images and sparse laser data. We particularly consider collision-free navigation with the Nao humanoid, which is the most popular small-size robot nowadays. Our approach first analyzes the scene around the robot by acquiring data from a laser range finder installed in the head. Then, it uses the knowledge about obstacles identified in the laser data to train visual classifiers based on color and texture information in a self-supervised way. While the robot is walking, it applies the learned classifiers to the camera images to decide which areas are traversable. As we show in the experiments, our technique allows for safe and efficient humanoid navigation in real-world environments, even in the case of robots equipped with low-end hardware such as the Nao, which has not been achieved before. Furthermore, we illustrate that our system is generally applicable and can also support the traversability estimation using other combinations of camera and depth data, e.g., from a Kinect-like sensor.

1. Introduction

Collision-free navigation is an essential capability for mobile robots since most tasks depend on robust navigation behavior. Reliable navigation with humanoid robots is still challenging for several reasons. First, most humanoids have significant payload limitations and thus need to rely on compact and light-weight sensors. This size and weight constraint typically affects the precision and update rates of the sensors that can be employed. Second, while walking, the robot's observations are affected by noise due to the shaking motion of humanoids. Third, depending on the placement of the individual sensors on the robot, the area in front of the robot's feet may not be observable while walking. That raises the question of whether the robot can safely continue walking without colliding with unanticipated objects. This is crucial as collisions easily lead to a fall. In this paper, we address these three challenges by developing an effective approach to obstacle detection that combines monocular images and sparse laser data. As we show in the experiments, this enables the robot to navigate more efficiently through the environment.

Since its release, Aldebaran's Nao robot quickly became the most common humanoid robot platform. However, this robot is particularly affected by the aforementioned limitations and problems, due to its small-size and the installed low-end hardware. This might be a reason why, up to today, there is no general obstacle detection system available that allows reliable, collision-free motion for that type of robot outside the restricted domain of robot soccer. Accordingly, useful applications that can be realized with the Nao system are limited. In this work, we tackled this problem and developed an obstacle detection system that relies solely on the robot's onboard sensors. Our approach is designed to work on a standard Nao robot with the optional laser head (see left image of Fig. 1), without need for further modifications. Our system is, however, not limited to the Nao platform but can be used on every robot platform that provides camera and range data.

To detect obstacles, our approach interprets sparse 3D laser data obtained from the Hokuyo laser range finder installed in the robot's head. Given this installation of the laser device, obstacles close to the robot's feet cannot be observed since they lie out of the field of view while walking. Hence, the robot needs to stop occasionally and adjust its body pose before performing a 3D laser sweep by tilting its head for obtaining distance information to nearby objects. This procedure robustly detects obstacles from the proximity data but is time-consuming and thus leads to inefficient navigation.

To overcome this problem, we present a technique to train visual obstacle detectors from sparse laser data in order to interpret images from the monocular camera installed in the robot's head. Our approach projects detected objects in the range scans into the camera image and learns classifiers that consider color and texture

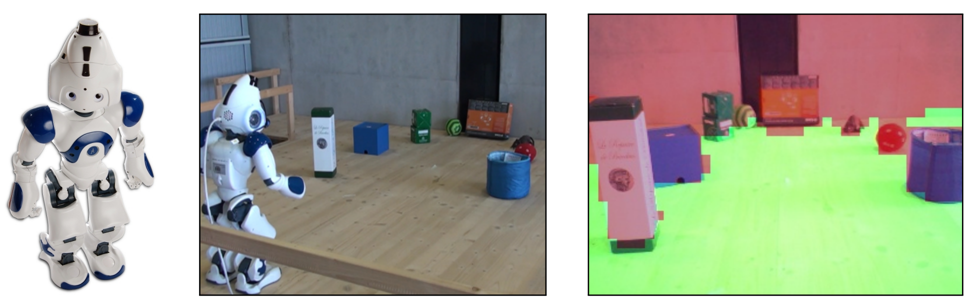


Fig. 1. Left: Humanoid Nao equipped with a laser range finder and a camera in its head. Middle: The robot is navigating in a cluttered scene. Right: Corresponding image taken by the robot's onboard camera together with traversability labels estimated by our approach (bright/green refers to traversable, dark/red to non-traversable areas).

information in a self-supervised fashion. While the robot is walking, it then applies the learned classifiers to the current camera image to decide which areas are traversable. Using this classification, the robot updates a local 2D occupancy grid map of the environment which it uses for path planning.

The main contribution of this paper is a full obstacle avoidance system that uses camera and sparse laser data for obstacle detection allowing for safe and efficient humanoid navigation, even in the case of a robot equipped with low-end hardware. The experiments carried out with a Nao humanoid illustrate that our approach enables the robot to reliably avoid static and moving obstacles while walking. We furthermore present results demonstrating that, using our technique, the robot reaches its goals significantly faster than with an approach that extracts obstacles solely based on 3D laser data acquired while stopping at certain intervals. Note that our technique is not limited to a specific setup but can be used with any combination of camera and depth data and improve the results. Often, not all of the environment in the robot's vicinity might be observable with a depth sensor. Reasons might be that the sensor data is sparse, e.g., from a sweeping laser, or because the sensor cannot operate under the prevailing conditions, e.g., because the obstacles are too close or illuminated by direct sunlight in case of the Kinect. Here, our approach can be used to identify obstacles in camera images and support the depth data, as we show in an experiment.

The remainder of this paper is organized as follows. In the next section, we describe our approach in detail. We first show how to acquire training data from a laser range finder, and then describe the visual classifiers that we developed, as well as our technique to use the classifiers' output for robot navigation. In Section 3 we present the experiments carried out to evaluate the approach. In Section 4 we discusses related work, before we conclude in Section 5.

2. Obstacle Detection Using Vision and Sparse Laser Data

In this section, we describe our approach to obstacle detection that is based on monocular images and sparse 3D laser data. Our robot continuously receives 2D range data from the laser sensor in its head with a frequency of approx. 10 Hz. To obtain 3D data of obstacles in the surroundings, the robot can stop walking, adopt a scanning pose, and tilt its head. In this way, a sweeping laser line is obtained that can be used to detect obstacles. Since the process of acquiring 3D laser data is time-consuming, we propose to additionally use the continuous stream of image data to identify the traversable areas in front of the robot. In particular, we present an approach to learn vision-based classifiers using the 3D laser data for training. Fig. 2 sketches the overall approach.

2.1. *Traversability Classification Based on Laser Data*

As a first step during the self-supervised training, our approach determines the traversable area in its surroundings by classifying the 3D laser range data. This

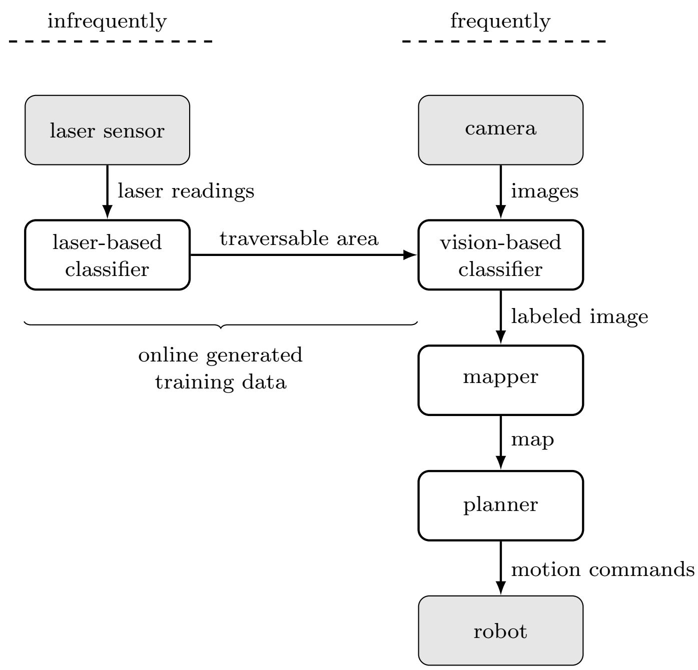


Fig. 2. Overview of the proposed system. During self-supervised training, sparse 3D laser data are interpreted to identify traversable and non-traversable areas and train the visual classifiers. These classifiers are then applied to detect obstacles in the images during navigation.

is achieved by analyzing the scan in a two-step procedure: (i) identify the ground plane and (ii) label areas as non-traversable which show a significant difference in height to the ground plane. Here, we insert the 3D end points measured by the laser scanner into a 2D grid structure (xy plane) and compute the mean elevation for each cell. All points in cells that show a deviation from the ground plane that is not compatible with the walking capabilities of the robot, are labeled as non-traversable and the remainder as traversable.

2.2. *Traversability Classification Based on Image Data*

The idea of our approach is to use the laser data whenever it is available to train image classifiers that are consequently used for traversability estimation during navigation. Our system automatically generates training data by considering a camera image and labels from the classified 3D scan. Each pixel in the training image is assigned the traversability label of the corresponding laser data, based on its projection into the camera frame. This data is then used to train the classifiers as described in the following.

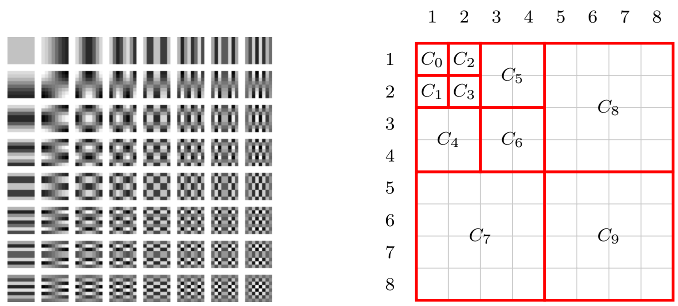


Fig. 3. Left: Basis functions of the 2D DCT for an 8×8 image. Right: Scheme for texture feature extraction using DCT coefficients. The illustration shows the matrix D composed of DCT coefficients for an 8×8 image. The C_i mark a subset of the DCT coefficients in D.

2.2.1. *Traversability Estimation Based on Color Information*

In most indoor environments, color information provides a good estimate about traversability. To be less sensitive to illumination changes, we consider only the hue and saturation values of image pixels and drop the value component. From the training data, our system first learns a distribution of color values for each traversability class, i.e., we compute normalized color histograms for the two classes. Once the histograms are learned, we can easily determine the likelihood that a certain color value occurs in a pixel given the individual classes by considering the respective bin.

Let t be the variable indicating traversability of the area represented by a pixel, hence $t \in \{\text{traversable}, \text{non-traversable}\}$, and let i_h and i_s be the pixel's intensity values of the hue- and saturation-channels, respectively. If we assume an uniform distribution of $P(t)$, $P(i_h)$, and $P(i_s)$ and independence of i_h and i_s, we can apply Bayes' theorem to evaluate the likelihood of traversability for each pixel as

$$\begin{aligned} P(t \mid i_h, i_s) &= P(i_h \mid t, i_s) P(t \mid i_s) P(i_h \mid i_s)^{-1} && (1) \\ &= P(i_h \mid t, i_s) P(i_s \mid t) P(t) P(i_h)^{-1} P(i_s)^{-1} && (2) \\ &\propto P(i_h \mid t) P(i_s \mid t). && (3) \end{aligned}$$

Hence, the likelihood that a pixel with intensities i_h and i_s belongs to class t is proportional to the product of the two histogram values for class t, evaluated at the corresponding bins for i_h and i_s.

2.2.2. *Texture-Based Classification*

Additionally, we use texture information to identify obstacles in the images. To do so, we classify rectangular patches in the image, based on their representation in the frequency-domain. In particular, we employ the discrete cosine transformation (DCT) to extract texture information[1]. For an input image, the DCT computes a set of coefficients which can be regarded as weights of a set of two-dimensional basis functions. Each basis function is an image constructed from a two-dimensional cosine function with a different frequency. As an illustration, the basis functions for an 8×8 image are shown in the left image of Fig. 3. The DCT transforms an input image into a matrix of DCT coefficient representing the amount of presence of a certain frequency in the original image. The frequencies increase horizontally to the right and vertically to the bottom. Accordingly, the lower-right part of the matrix contains information about the high frequency content of the image and is often dominated by noise. By considering only a small subset of the coefficients, mainly the low to mid frequency parts, an input image can already be reconstructed quite accurately.

For texture-based classification, we divide the input image's hue-channel into overlapping patches of size 16×16 computed at the fixed distance of 8 pixels in vertical and horizontal direction[a]. Each patch is assigned a traversability label t, based on the percentage of labeled pixels inside the patch using the classified laser data. If more than a certain percentage θ_P (in our experiments 90%) of the pixels in that image patch are labeled as traversable, we label it as an example for traversable texture. Analogously, if more than θ_P percent of the pixels in the patch are labeled non-traversable, we assign the label non-traversable to the patch. If neither condition holds, for example at the boundaries of obstacles, the patch is not used for self-supervised training. From the labeled image patch, we then compute a feature vector f_{DCT} based on the DCT transform.

The feature vector f_{DCT} is based on the multi-resolution decomposition of the DCT transform similar to Huang and Chang [2] and Nezamabadi-Pour and Saryazdi [3]. Let P be a patch of size 16×16 and D be the DCT of P. Further, let C_i represent the set of all the DCT coefficients in the correspondingly marked region of D, as per Fig. 3[b]. For example, C_0 is the DCT coefficient located at $D_{1,1}$ and C_5 is the set of the DCT coefficients located at $D_{1,3}, D_{1,4}, D_{2,3}$, and $D_{2,4}$, etc. Let M_i and V_i be the mean and the variance over all coefficients in C_i, respectively. Based on M_i and V_i we define the 13-dimensional feature vector f_{DCT} as

$$f_{\mathrm{DCT}} = (M_0, M_1, M_2, M_3, V_4, V_5, \ldots, V_{12}). \tag{4}$$

Accordingly, we represent the visually significant low frequency coefficients directly and accumulate the less significant high frequency components by their variance.

Our approach then trains a support vector machine (SVM) that best separates the feature space given the labeling from the training data. The SVM learns a function $p_T : \mathbb{R}^{13} \mapsto [0, 1]$, where $p_T(f_{\mathrm{DCT}})$ is the likelihood that the feature vector f_{DCT} represents traversable area.

[a] We also tested the saturation-channel and combinations of hue- and saturation channel. However we found the hue-channel to provide the most reliable results.
[b] Fig. 3 sketches the DCT basis functions and the feature extraction scheme for patches of size 8×8. The generation of an analogue illustration for patches of size 16×16 is straightforward.

2.3. *Incorporation of Neighborhood Information*

Obviously, the resulting classification of an image based on either of the classifiers presented above is not perfect. It might happen that spurious classification errors exist which can actually prevent the robot from safe navigation towards its goal. To account for such cases, we consider dependencies between nearby areas. We investigated two different labeling methods to take neighborhood information into account and to combine the result of both classifiers introduced above. In the remainder of this section, we introduce these techniques and show how they are used in our approach to model dependencies between neighboring areas.

Both algorithms operate on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$ and edges $\mathcal{E}$ between pairs of nodes. In our setting, the nodes correspond to small rectangular image patches of size 5×5 in the image and the edges describe their neighborhood relations[c]. Let $\mathcal{T}$ be the set of possible labels, in our case traversable and non-traversable. Both algorithms iteratively compute new estimates for the nodes v_i based on local neighborhood information and an initial estimate. For each node v_i, the neighborhood $\mathcal{N}(v_i) \subset \mathcal{V}$ refers to the nodes v_j that are connected to v_i via an edge. Here, we assume an eight-connected graph of neighborhood relations. That means that each node only influences its eight neighbors. Furthermore, we assume that each of the eight neighbors of v_i are equally important for estimating the label of v_i. We obtain the initial estimate for a node by computing the weighted average over the estimates from the color classifier and the texture classifier.

2.3.1. *Probabilistic Relaxation Labeling*

Probabilistic relaxation labeling[4] assumes that each node v_i stores a probability distribution about its label, represented by a histogram P_i. Each bin $p_i(t)$ of that histogram stores the probability that the node v_i has the label t. For two classes, P_i can efficiently be represented by a binary random variable.

Each neighborhood relation is represented by two values: r_{ij} describes the compatibility between the labels of nodes v_i and v_j and c_{ij} represents the influence between the nodes. $\mathcal{C} = \{c_{ij} \mid v_j \in \mathcal{N}(v_i)\}$ is the set of weights indicating the influence of node v_j on node v_i. In our case, the weights c_{ij} are set to $\frac{1}{8}$. The compatibility coefficients $\mathcal{R} = \{r_{ij}(t, t') \mid v_j \in \mathcal{N}(v_i)\}$ are used to describe neighborhood relations. Here, $r_{ij}(t, t')$ with $t, t' \in \mathcal{T}$ defines the compatibility between the label t of node v_i and the label t' of v_j by a value between -1 and 1. A value $r_{ij}(t, t')$ close to -1 indicates that the label t' is unlikely at the node v_j given that the node v_i has label t. Values close to 1 indicate the opposite. For computing the

[c] In theory, a node could be created for each pixel. This, however, is computationally too demanding for our online application so that small images patches are used.

compatibility coefficients, we follow the approach suggested by Yamamoto [5]

$$r_{ij}(t,t') = \begin{cases} \frac{1}{1-p_i(t)}\left(1 - \frac{p_i(t)}{p_{ij}(t|t')}\right) & \text{if } p_i(t) < p_{ij}(t \mid t') \\ \frac{p_{ij}(t|t')}{p_i(t)} - 1 & \text{otherwise,} \end{cases} \tag{5}$$

where $p_{ij}(t \mid t')$ is the conditional probability that node v_i has label t given that node $v_j \in \mathcal{N}(v_i)$ has label t', and $p_i(t)$ is the prior for label t. We determined these probabilities by counting given the training data.

Given an initial estimation for the probability distribution over traversability labels $p_i^{(0)}(t)$ for the node v_i, the probabilistic relaxation method iteratively computes estimates $p_i^{(k)}(t),\ k = 1, 2, \ldots$, based on the initial $p_i^{(0)}(t)$ in the form

$$p_i^{(k+1)}(t) = \frac{p_i^{(k)}(t)\left[1 + q_i^{(k)}(t)\right]}{\sum_{t' \in \mathcal{T}} p_i^{(k)}(t')\left[1 + q_i^{(k)}(t')\right]}, \tag{6}$$

where

$$q_i^{(k)}(t) = \sum_{v_j \in \mathcal{N}(v_i)} c_{ij}\left[\sum_{t' \in \mathcal{T}} r_{ij}(t,t') p_j^{(k)}(t')\right]. \tag{7}$$

On the equations above, the term $q_i^{(k)}$ represents the change of the probability distribution $p_i^{(k)}$ for node v_i in iteration k based on the current distribution for its neighbors, the compatibility coefficients and the weights c_{ij}.

After convergence, we obtain for each pixel in the image the probability that it is traversable from the probabilities of the corresponding nodes.

2.3.2. *Label Propagation*

In the label propagation algorithm [6], neighborhood relations in $\mathcal{G}$ are encoded in the affinity matrix $W \in \mathbb{R}^{N \times N}$. Here, $W_{ij} \geq 0$ represents the strength of the relation between v_i and v_j, where $W_{ij} = 0$, if v_i and v_j are not related. We set $W_{ij} = 1$ if v_j is in the eight-neighborhood of v_i, and 0 otherwise.

The algorithm iteratively computes new estimates for the labels of the nodes in $\mathcal{V}$ based on the previous estimate and the initial estimate. Let $\hat{y}_i^{(k)} \in [-1, 1]$ be the estimate for node v_i after iteration k of the algorithm and $\hat{y}_i^{(0)}$ denote the initial estimate for v_i, which we obtain by averaging the outputs from the color and texture classifiers, as explained before. To meet the condition that the values are from the interval $[-1, 1]$, we multiply the averaged probability by two and subtract 1. The classification for the node v_i after iteration k is then given by the sign of $\hat{y}_i^{(k)}$, where a positive value corresponds to classification as traversable and a negative value the opposite. We iteratively compute $\hat{y}_i^{(k+1)}$ from $\hat{y}_i^{(k)}$, starting with $k = 0$, by

$$\hat{y}_i^{(k+1)} = \frac{\sum_{j=1}^{N} W_{ij}\hat{y}_j^{(k)} + \frac{1-\alpha}{\alpha}\hat{y}_i^{(0)}}{\sum_{j=1}^{N} W_{ij} + \frac{1-\alpha}{\alpha}}, \tag{8}$$

where $\alpha \in (0, 1)$ is a weight, describing the influence of the initial estimate on the classification in each iteration. For our setup, we determined $\alpha = 0.05$ yielding the best results in terms of accuracy during a series of preliminary experiments with hand-labeled image data. After termination, we transform the $\hat{y}_i$ back to probabilities and proceed as with the probabilistic relaxation labeling algorithm.

2.4. *Map Update and Motion Planning*

To integrate the traversability information from the camera images, we maintain a 2D occupancy grid map where every cell is associated with the probability that it is occupied, i.e., not traversable. All cells are initialized with 0.5. The map is updated whenever a camera image has been classified. For the update step, we use the recursive Bayesian update scheme for grid maps to compute the probability $p(c \mid z_{1:t})$ that a cell c is occupied at time step t [7]:

$$p(c \mid z_{1:t}) = 1 - \left[1 + \frac{1 - p(c \mid z_t)}{p(c \mid z_t)} \, \frac{1 + p(c \mid z_{1:t-1})}{p(c \mid z_{1:t-1})} \, \frac{p(c)}{1 - p(c)} \right]^{-1} . \qquad (9)$$

Here, z_t is the observation (i.e., the labeled camera image) at time t and $z_{1:t}$ is the sequence of observations up to time t. For the inverse sensor model $p(c \mid z_t)$, we employ a projection of the labeled camera image onto the ground plane. The camera pose is hereby computed from the robot's pose estimation system [8]. For planning the robot's collision-free motion towards a goal location, we apply the A^*-algorithm on the 2D grid map.

2.5. *Retraining the Classifiers*

The approach for estimating the traversable area in the surroundings of the robot assumes that the scene does not change substantially – otherwise the floor or the obstacles may not be reliably identified anymore. Thus, whenever the robot notices that the appearance of the scene has changed, the classifiers are retrained from the current observations. Our system discards all previous training data to adapt to the new environment.

To detect changes, we monitor heuristics that indicate the need for retraining the image classifiers. We currently consider three heuristics: First, the correlation between the color histogram computed over the current image and the one obtained during the last learning step. Second, the consistency between the color classifier and the texture classifier. Finally, we use the certainties of the individual classifiers. To estimate that, we consider the number of pixels that have uncertain label assignments in both classifiers, i.e., their membership probabilities are close to 0.5. We trigger retraining in case any of these heuristics indicate that the current classification might contain failures due to substantial changes in the scene.

For our experimental setup, we observed that training the visual classifiers takes only about 0.16 s in average on a standard desktop computer, with the majority of the time consumed by the texture feature extraction. The SVM applied as a

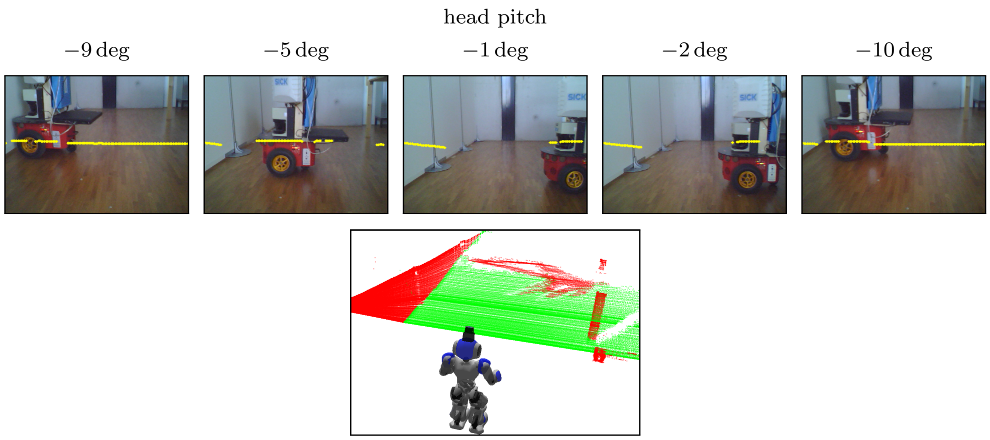


Fig. 4. Example for phantom obstacles resulting from moving objects. Top: Sequence of onboard camera images at different head pitch angles while taking a 3D scan. The red robot moves in the scene. The (yellow) points illustrate laser measurements projected into the image. Bottom: 3D scan from integrating the laser measurements classified with the approach from Sec. 2.1 into traversable (green) and non-traversable (red). Laser measurements of the red robot's trajectory form a phantom obstacle that would lead to incorrect training data for the visual classifiers.

texture-based classifier uses 300 features (for texture patches of size 16×16 and a camera resolution of $320\times\ 240$) of dimensionality 13, resulting in training data of size 3900. LIBSVM[9] requires only about 0.03 s for training the SVM. Therefore, we did not investigate incremental learning and instead apply simple re-training using the new training data. In this way, the robot can adapt to the appearance of the environment.

2.6. *Moving Obstacles during Training*

The approach presented so far cannot deal with moving obstacles during training, i.e., while the robot is acquiring the 3D laser scan. If an object moves in front of the robot, the 3D points belonging to its surface are spread along its trajectory (phantom obstacle). If we classify such 3D scans using the method described in Sec. 2.1, we will end up labeling the whole trajectory of the object as non-traversable. An example for such a situation is shown in Fig. 4. If we used this information for training the visual classifiers as described in Sec. 2.2 we would learn from incorrectly labeled data. This would lead to wrong estimates for free and occupied space.

To deal with such situations, we therefore use dense optical flow to identify moving obstacles in the images while taking the 3D range scan. In particular, we apply large displacement optical flow to the camera images[10]. This algorithm computes the movement of individual pixels between two consecutive images and thus can identify moving objects.

While tilting the head for obtaining 3D range data, the camera moves in vertical direction and so our approach identifies all pixels whose movement deviates from

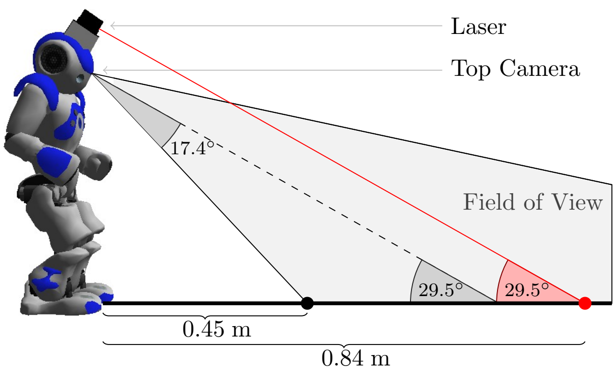


Fig. 5. Illustration of the Nao robot indicating the top camera's field of view and the laser's scan plane. The head of the robot is pitched to a maximum extend of 29.5°. With this setting, the closest region of the floor still observable with the laser is 0.84 m away from the robot's feet. By using the camera, this distance can be reduced to 0.45 m.

this direction. We identify the respective points in the laser scan that should be ignored for labeling the static scene because they hit moving objects. The remaining laser points are classified into traversable and non-traversable as before and used for learning the visual classifiers. Additionally, the pixels identified as belonging to moving obstacles are used as non-traversable examples.

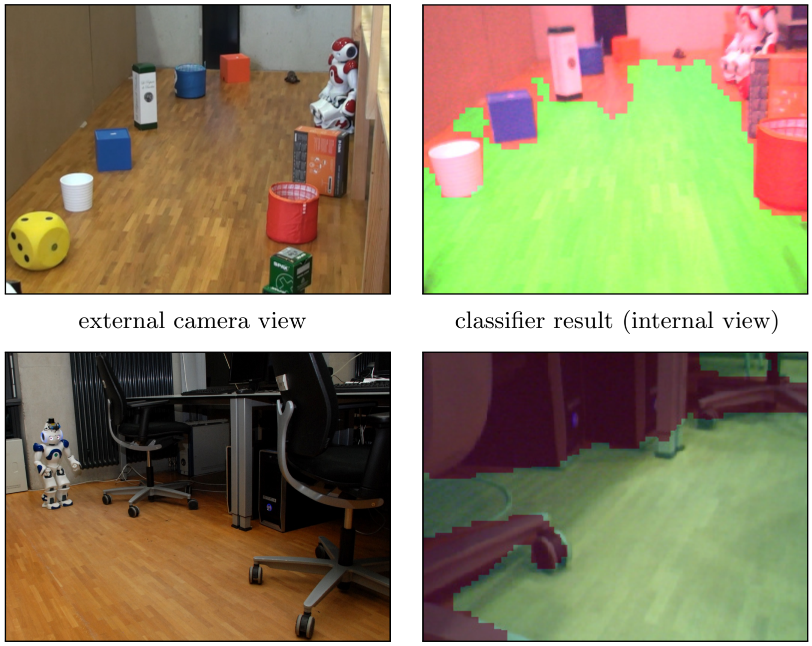
3. Experiments

The experimental evaluation has been designed to show that our robot reliably detects and avoids obstacles during walking in real-world environments using the learned image classifiers and that our method enables the robot to efficiently navigate in the environment. For all experiments, we use a Nao humanoid equipped with a Hokuyo URG-04LX laser range finder and a camera as can be seen from Fig. 5. When maximally tilting the head while walking, the closest region that is observable with the robot's camera is at a distance of 45 cm to the feet, whereas the laser plane is 84 cm away.

The computation is done off-board on a standard dual core desktop PC, because the Nao's onboard CPU is almost at full load when executing motion commands and reading the sensor data. Our system runs in real-time.

3.1. *Classification Accuracy*

To evaluate the accuracy of the vision-based classifiers, we performed extensive experiments in three environments with different floor surfaces and various obstacles on the ground. The scenarios can be seen in Fig. 6 top row (experiment 1), Fig. 1 (experiment 2), and Fig. 6 bottom row (experiment 3). In each experiment, the robot's task was to navigate without collision through the scene towards a given



external camera view | classifier result (internal view)

external camera view | classifier result (internal view)

Fig. 6. Two examples of obtained classification. Left: external camera view for reference, right: classified onboard camera image (bright/green refers to traversable, dark/red to non-traversable areas). This image is best viewed in color.

goal location. First, the robot took a 3D range scan, trained its visual classifiers, and then used only the classified camera images for navigation and mapping of traversable and non-traversable areas.

For the evaluation, we recorded an image every 10 s while the robot was navigating along with the traversability probabilities computed by the visual classifiers and the combined approach which also considers neighboring information. Whenever the probability for a pixel corresponding to an obstacle was bigger than 0.5, we counted it as non-traversable and traversable otherwise. Fig. 1 and Fig. 6 show qualitative classification results achieved in different environments using our system.

We then compared the results of our visual classification system to a manual labeling of the pixels. The classification rates in terms of confusion matrices and accuracy are shown in Table 1. As the table illustrates, the achieved accuracy of the combined approach was above 91 % in all three environments. The combined approaches especially improved the results in experiment 1. Here, the accuracy went up from 89% to 95% for label propagation and up to 96% for probabilistic relaxation labeling. This is mainly because the color classifier failed to identify the floor reliably in this experiment, due to shadows and the unsaturated color of the parquet floor under the prevailing lighting conditions. In the HSV color space, hue is unstable for

Table 1. Evaluation of the image classifiers. Confusion matrices for all classifiers during three experiments along with accuracy values.

	Experiment 1		Experiment 2		Experiment 3	
Number of images		102		38		31
	TEXTURE CLASSIFIER					
	Estimated as		Estimated as		Estimated as	
True class	Obstacle	Floor	Obstacle	Floor	Obstacle	Floor
Obstacle	0.87	0.13	0.99	0.01	0.83	0.17
Floor	0.10	0.90	0.09	0.91	0.03	0.97
Accuracy	0.89		0.97		0.90	
	COLOR CLASSIFIER					
	Estimated as		Estimated as		Estimated as	
True class	Obstacle	Floor	Obstacle	Floor	Obstacle	Floor
Obstacle	0.97	0.03	0.99	0.01	0.91	0.09
Floor	0.19	0.81	0.04	0.96	0.09	0.91
Accuracy	0.89		0.98		0.91	
	COMBINED APPROACH USING LABEL PROPAGATION					
	Estimated as		Estimated as		Estimated as	
True class	Obstacle	Floor	Obstacle	Floor	Obstacle	Floor
Obstacle	0.99	0.01	1.00	0.00	0.92	0.08
Floor	0.10	0.90	0.07	0.93	0.09	0.91
Accuracy	0.95		0.98		0.91	
	COMBINED APPROACH USING PROB. RELAX. LABELING					
	Estimated as		Estimated as		Estimated as	
True class	Obstacle	Floor	Obstacle	Floor	Obstacle	Floor
Obstacle	0.98	0.02	0.99	0.01	0.88	0.12
Floor	0.06	0.94	0.05	0.95	0.07	0.93
Accuracy	0.96		0.98		0.91	

unsaturated colors. Regarding all the experiments, we observed that probabilistic relaxation labeling and label propagation gave approximatively the same error rates. We chose probabilistic relaxation labeling as the method for classifying the image and used it in all the experiments presented in the following.

3.2. *Qualitative Results on Traversability Estimation*

The second experiment illustrates the functionality of our visual obstacle avoidance system. A video of this experiment is available online[d]. We placed several obstacles on the floor and changed the obstacle positions while the robot was walking through the scene. The robot first took a 3D range scan to train its classifiers, and then

[d]hrl.informatik.uni-freiburg.de/maierd11ijhr.mp4

Fig. 7. Left: scene from the robot's view, 2nd left: top view, 3rd left: scene changed while navigating, right: labeled image from the robot's camera.

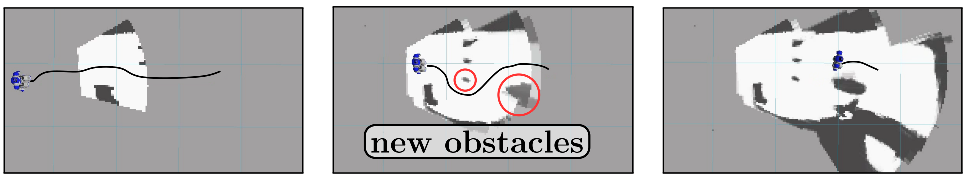


Fig. 8. Maps and planned trajectories while navigating. Left: initially built map (corresponds to the 1st and 2nd image in Fig. 7), middle: map after new obstacles have been detected (corresponds to the 3rd and 4th image in Fig. 7), right: updated map while approaching the goal.

started navigating to the goal location at the end of the corridor while updating its map based on the visual input. Fig. 7 and Fig. 8, illustrate snapshots taken during this experiment. The left image of Fig. 7 shows the initial scene during training. The second image shows a top view of the partial scene. The third image shows the same scene after placing the ball in the way of the robot and also changing the position of the blue basket. In the meanwhile, the robot approached the left corner of the top view image (only the feet are visible). The last image of Fig. 7 shows the correctly labeled image.

In addition to that, Fig. 8 illustrates the updated grid map at the different time steps and the recomputed collision-free trajectories. The first image shows the map right after training along with the initial trajectory. The second image shows the map after the red ball and the blue basket have been placed. This blocked the initial trajectory and forced the robot to detour. The last image shows the map while the robot is approaching the goal location. The dark area at the right corresponds to the wall visible in the first and last image of Fig. 7.

3.3. *Advantage of Considering Texture for the Classification*

In this section, we demonstrate that in some situations it is not sufficient to solely rely on color information while estimating traversability in an image, even if the color classifier often provides good classification results as can be seen from Table 1.

Fig. 9 shows a scenario where obstacles and floor have similar color distributions but differ in texture. In this experiment, the robot trained its classifiers as before and started to navigate to a predefined goal location behind the small table. While walking, a second obstacle appeared and blocked the robot's path, thus forcing the

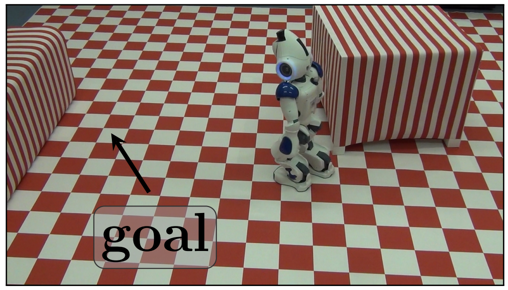


Fig. 9. Example scenario in which a classification based on color is not sufficient to distinguish between obstacles and the floor. Here, obstacles and floor have a similar color distribution but differ in texture. The robot's task was to navigate to the marked goal location.

robot to take a detour.

We recorded the classification based on color information with relaxation labeling and the classification with our combined approach, considering also texture information. We then learned occupancy grid maps from both classifications. Examples of the obtained classification results and the corresponding maps are shown in Fig. 10. As can be seen, the map created from the classification based on color is not usable for navigation due to the many phantom obstacles.

Fig. 11 illustrates why the color-based classifier failed. As one can see in the right image, neither obstacles nor floor are clearly identified as either traversable or non-traversable. In fact, most of the pixels are undecided with a probability of approximately 0.5. This is due to the similar color distributions for obstacles and floor in the training image. The fact that the classified image is not homogeneous but contains some brighter and darker spot comes from the fact that the color distribution for obstacles and floor are not completely identical as a result of reflections from the lighting, different viewing angles, etc. Texture on the other hand, allows to differentiate obstacles from traversable floor area. In conclusion, we achieve a more robust classification considering the two visual clues.

3.4. *Detecting and Avoiding People*

In the following experiment, we show that our approach is able to identify people as obstacles and to plan safe paths around them. As can be seen in Fig. 12, people were traversing the area in front of the robot and were correctly classified as obstacles in the camera images. Accordingly, the robot updated its map and planned a detour to its target location, thereby avoiding collisions with people blocking its way.

3.5. *Comparison to Laser-Only Observations*

In this section, we compare the overall travel time for the humanoid using only laser data for obstacle detection with the results obtained when applying the proposed method. When relying on laser data only, the robot has to record a 3D laser scan

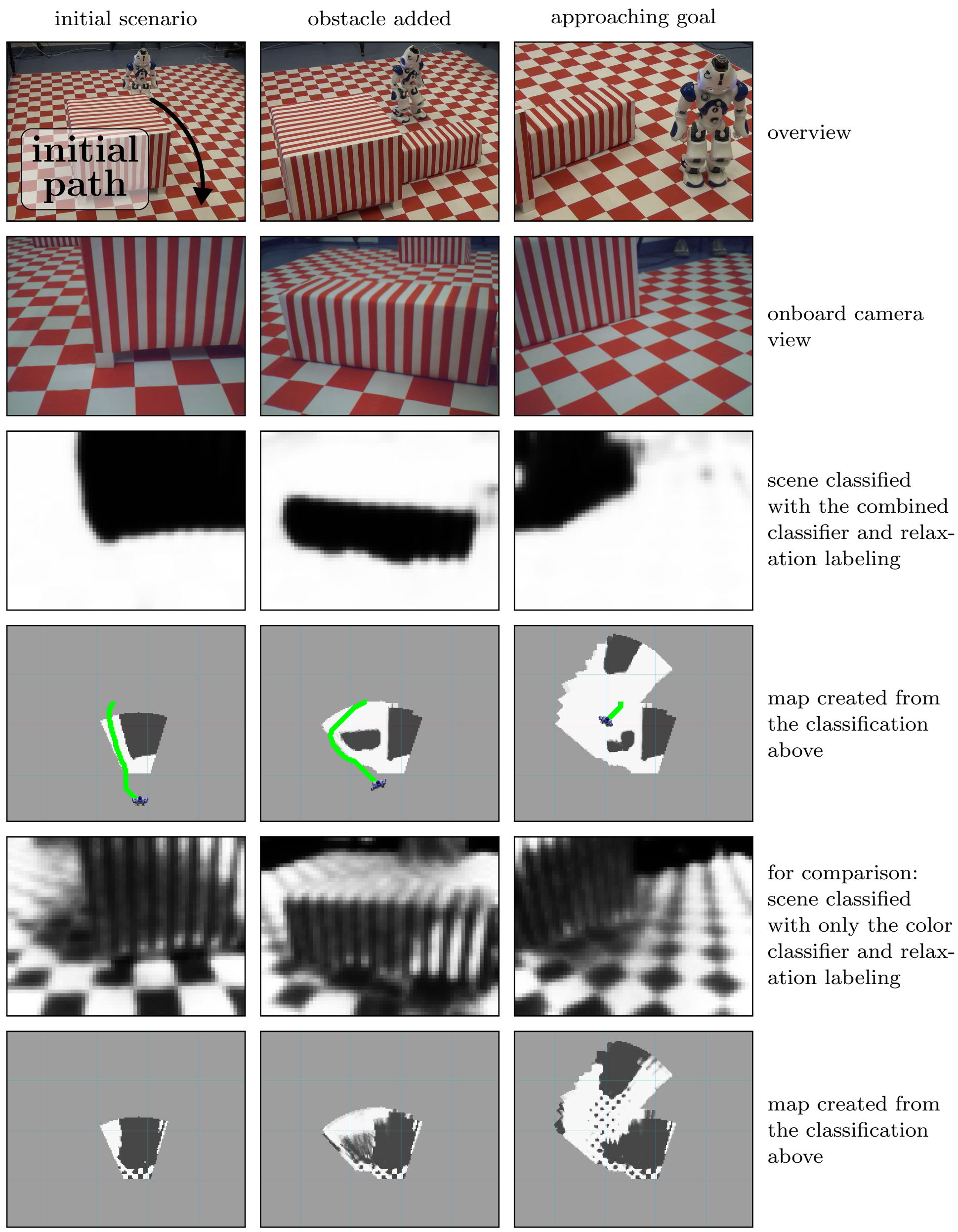


Fig. 10. Demonstration that relying solely on color information can be insufficient. In this experiment, the robot was navigating on a checkerboard floor. The columns show the scenario at different points in time. The green paths in row four is the planned path from the robot to the goal location, generated from the current map. As can be seen, the map generated by the color classifier is unusable for robot navigation due to the many phantom obstacles.

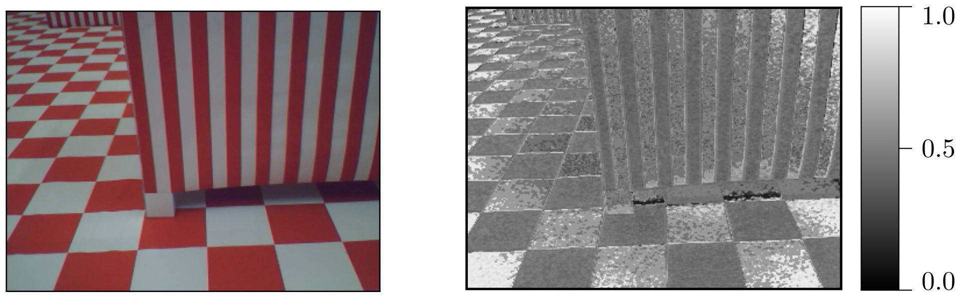


Fig. 11. Example for the classification results using only color in the environment shown in Fig. 9. The image corresponds to the scene shown in the first column of Fig. 10. The gray value in the right image corresponds to the probabilities for traversable area, where brighter values indicate higher probabilities.

Table 2. Travel time with the laser-based system and with our combined approach.

Technique	Travel time (5 runs)					Avg.
3D laser only	219s	136s	208s	135s	135s	167s
Vision & Laser	136s	94s	120s	96s	87s	107s
Scenario	1	2	3	4	5	

at least every 1.3 m to 1.4 m. For larger distances, typical floors (wood or PVC) provide poor measurements with the Hokuyo laser due to the inclination angle.

In the following set of experiments, the task of the robot was to travel through an initially empty corridor. We first uniformly sampled the robot's goal location, the number of obstacles (from 1 to 3) and their positions. After placing the obstacles in the scene, we measured the time it took the robot to navigate through the corridor with and without our vision-based system. The experiment was repeated five times. The travel times for each experiment are depicted in Table 2 and, as can be seen, our vision-based approach required on average 107 s compared to 167 s with the laser-based system. We carried out a paired two sample t-test with a 0.995 confidence level. The test shows that this result is statistically significant ($t_{value} = 5.692 >$ t-table(conf=0.995; DoF=4) $= 4.604$). Hence, our vision-based approach is significantly faster and hence more efficient than the laser-based approach with fixed 3D scanning intervals.

Furthermore, note the laser-based approach is not applicable in changing environments, e.g., scenarios with moving objects. Since taking a 3D scan during walking is not possible, the robot gathers only laser 2D data in-between two 3D scans and is thus not able to sense all objects blocking its way. Our vision-based approach on the other hand, updates the environment representation at a frequency of 3.5 Hz and can thus quickly react to changes in the environment.

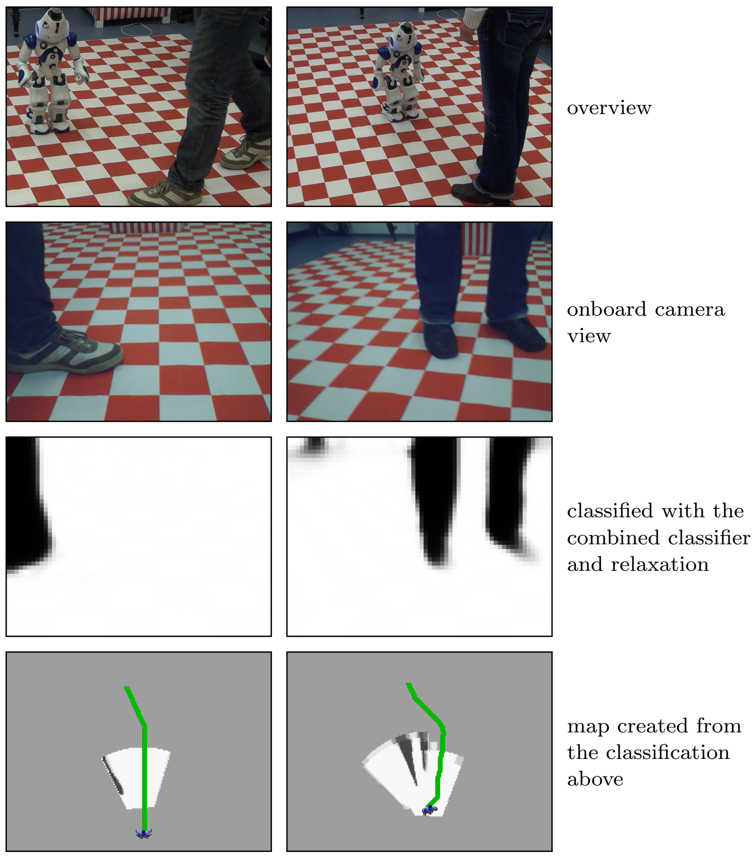


Fig. 12. Experiment with moving people. People traverse the area in front of the robot and are classified as obstacles in the camera images. In the left column, one can observe one error in the classification. The white shoe of the person is more similar to the previously seen background than to the obstacles and thus is classified wrongly. Nevertheless, given the detected leg of the person, the robot updates its map correctly and replans the trajectory (green) to avoid collisions.

3.6. *Comparison to Ground Truth*

Table 3. Path execution times when planning the path in a perfect map versus in a map created by our vision-based system.

Technique	Travel time (10 runs total)					Avg. and std.
Motion capture	92.5s	93.0s	90.0s	89.0s	93.5s	91.6s ± 1.8s
Vision & laser	91.5s	105.0s	91.0s	95.0s	99.0s	96.3s ± 5.2s

The next experiment was designed to evaluate the efficiency of the trajectories through a cluttered scene by comparing the ones generated by our system with

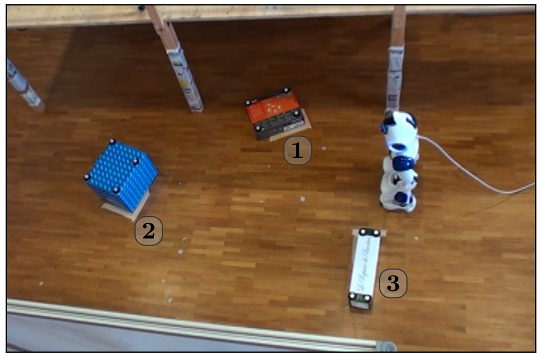


Fig. 13. Nao navigating in the course for the comparison with perfect observations. The boxes have been equipped with white ball-shaped markers so that they can be tracked with our real-time motion capture system.

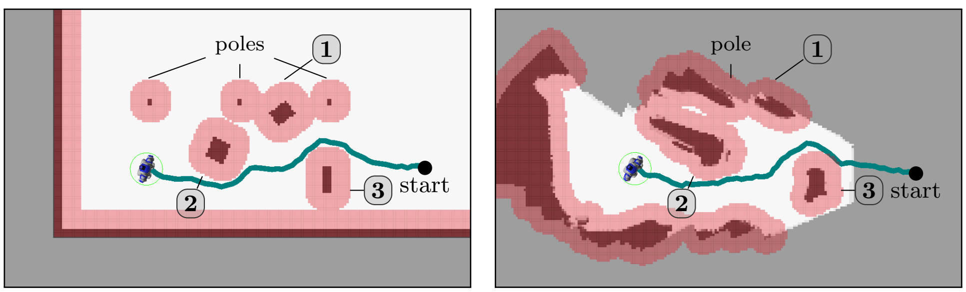


Fig. 14. Left: Map created from perfect observations along with the path taken by the robot (teal). The map shows the obstacles (dark red) along with the safety margin respecting the robot's radius used for navigation (light red). Right: Map created for a similar scene using our vision-based approach, along with the robot's trajectory. The left-most and right-most poles are not mapped because they were out of the field of view of the camera. Tall obstacles occupy more space in this map due to the inverse perspective mapping. This effect resolves if the perspective changes, e.g., if the robot gets closer to the object or observes it from another side.

the output of a system that has perfect knowledge about the robot's environment from an external motion capture system. In this experiment, the robot first navigated through a corridor using our vision-based approach. The robot initially took a 3D range scan to train its classifiers, and then started navigating to a fixed goal location at the end of the corridor, while updating its map based on the visual input. While the robot was moving, we placed three box shaped obstacles in the robot's path, thereby blocking the direct path to the goal and forcing the robot to take a detour twice. We measured the time it took the robot to walk to the goal from its starting location. We carried out 5 runs of such navigation tasks. Then, we repeated the experiment with a map constructed by tracking the obstacles with an motion capturing system. With this setup, we repeated the navigation task described above another 5 runs, but using the ground truth map for path planning. An image of the scenario is shown in Fig. 13, while Fig. 14 show an example map for each of

Fig. 15. Left: Scene during training (robot outside the camera's field of view). Middle: New but similar-looking objects do not trigger retraining. Right: The newly appeared carpet is correctly interpreted as a substantial change and retraining is triggered.

the two approaches. Note that the obstacles appear larger on the map created by our vision-based approach due to the re-projection of the classification from the image plane to the ground plane. However, once the robot observes the floor that is erroneously occupied by the obstacle, the free space is updated properly.

The average travel time using the motion capture system was 91.6 s. Our approach took 131.3 s for the same setup including 35 s for obtaining the 3D scans to train the classifiers. The training phase is obviously not needed if ground truth information is available. The pure travel time by our approach was 96.3 s, i.e. less than 5 s more than the setup with perfect knowledge on the environment. The path execution times for the experiments are shown in Table 3. The slightly longer travel times are explained by the fact that minor pose inaccuracies in the mapping phase lead to obstacles that are slightly larger than in reality and thus produce marginally longer trajectories. In one experiment using our proposed approach, the planned path "oscillated" for a short while (once planned a left and a right detour around an obstacle) which causes the single outlier in the data reported in Table 3.

3.7. *Retraining the Classifiers to Cope with Appearance Changes*

This following experiment shows that the robot can adapt to changes in the environment using the retraining heuristics presented in Sec. 2.5. Initially, two obstacles were placed in the scene. After the robot took a 3D scan to train its visual classifiers and started walking towards the goal location while avoiding newly detected obstacles. Afterwards, we placed a carpet in front of the robot.

Without the retraining heuristic, the robot correctly detected and avoided all obstacles. However, it also detected the carpet as an obstacle and hence, did not cross it. As it was blocking the passage, the robot could not reach the goal. We repeated the experiment with the retraining heuristics, Fig. 15 illustrates this scenario. Again, the robot detected and avoided the obstacles blocking its path. When we placed the carpet in the scene, retraining was triggered due to a different color distribution in the current image. Accordingly, the robot stopped walking and took a 3D scan to retrain its visual classifiers. Because the carpet was classified as traversable in

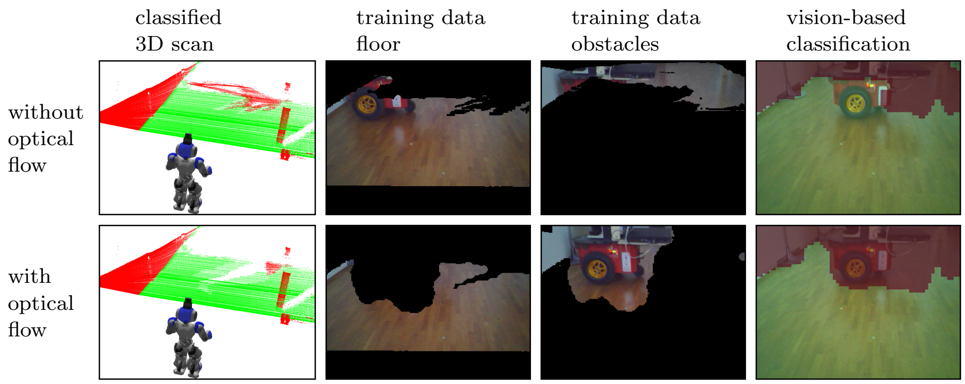


Fig. 16. Identification of moving obstacles during acquisition of training data. The scene corresponds to Fig. 4. The top row illustrate our approach without exploiting optical flow information. In the bottom row we consider optical flow for identifying moving obstacles during training. The first three columns depict the training data and the last column show the classification results obtained with our approach. Considering optical flow leads to better classification results.

the 3D scan, the visual classifiers also learned that the carpet is traversable. So the robot continued walking and crossed the carpet to reach its goal location.

3.8. *Learning Traversability Classification in Dynamic Scenes*

This experiment demonstrates that using our approach, the robot is able to acquire valid training data even in the presence of moving obstacles. While our robot was taking a 3D scan for training, a wheeled robot navigated in front of the humanoid, as shown in the top row of Fig. 4. Once the humanoid completed the 3D scan, our algorithm identified the traversable and non-traversable areas in the scan, and projected them into the camera image to train the visual classifiers.

The top row in Fig. 16 depicts the results we obtained without detection of moving obstacles. The first image shows the classification of the 3D scan with the original method. The whole trajectory of the wheeled robot is visible in the scan and identified as obstacle. The next two images show the parts of the training image labeled as traversable and non-traversable, respectively, as obtained by projecting the classified 3D scan into the training image. These parts are used as training samples for the two different classes. One can see that the training data is defective because parts of the wheeled robot are visible in the training data for traversable areas, while the wheeled robot itself is not contained as a whole in the training data for non-traversable areas. The last image shows an example classification with the vision-based approach, after training from the defective data. Here, parts of the wheeled robot are wrongly classified as traversable.

The bottom row in Fig. 16 shows the same experiment as described above, but this time using the extension to identify moving obstacles using optical flow, as proposed in Sec. 2.6. Here, the wheeled robot's trajectory is not visible in the 3D scan

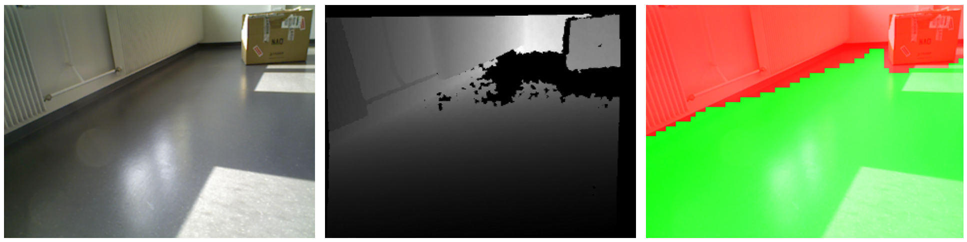


Fig. 17. Our approach can also be used to support data from Kinect-like sensors. The left image shows the RGB image from a Kinect that we used as training image. The middle image shows the corresponding depth data from the Kinect. The image contains some blind spots in the background. The right image shows the subsequent RGB image with our classifier's output.

(first image) and also not in the parts of the training data used to learn traversable area (second image). Furthermore, the whole robot contained in the training data for the non-traversable class (third image). Consequently, the classification obtained from our vision-based approach yields substantially better results (last image). As can be seen, the wheeled robot is completely identified as non-traversable and the floor area is correctly marked as traversable.

3.9. *Supporting RGB-D data*

Finally, we illustrate the versatility of our approach and apply our classification approach to support navigation based on Kinect-like RGB-D sensors. These devices return RGB images and the corresponding depth data and can, in principle, directly be used for obstacle detection and mapping. However, in some scenarios, the depth data may be absent, for instance in presence of sunlight through windows, a low inclination angle of the sensor, or objects close to the sensor. Fig. 17 gives an example of such a scenario. The first image shows a RGB image from a camera sequence and the second one the corresponding depth image. As can be seen, for some parts in the background, no depth data was returned (marked black), due to the bright sunlight and the low inclination angle. Navigation based on such data is still possible, as the effect reduces once the camera gets closer, but the missing depth data prevents foresighted planning.

We applied our approach to RGB-D data to illustrate how this problem can be avoided. To provide this proof of concept, we trained our classifiers from the left image in Fig. 17 and from the parts in the middle image where depth data was available. We then applied the classifiers to the consecutive RGB image. The obtained classification is shown in the right most image. As can be seen, our approach correctly classifies the traversable and non-traversable areas and has no blind spots, unlike the Kinect's depth data. Thus, combining our classifiers with the depth data from a Kinect, yields increased information about the environments which allows for more foresighted navigation.

3.10. *Summary of Results*

In the following, we discuss the properties of our approach. First, we demonstrated high classification accuracy and reliable obstacle avoidance in multiple experiments. Second, we increased the field of view for obstacle detection and the update rate compared to pure laser-based navigation of the Nao. Accordingly, our system leads to safer navigation and reduced travel time. Note that even on large-sized robots with fast sensors, the process of obtaining 3D laser data requires about 1 s [11]. In contrast to that, our vision-based approach currently observes the environment at a rate of 3.5 Hz, i.e. every 285 ms. Third, by exploiting optical flow information, our approach can react to dynamics in the scene while acquiring training data. Without optical flow information, phantom obstacles would appear in the 3D point cloud when dynamic objects pass in front of the robot while collecting data. Finally, we showed that our approach can also be combined with RGB-D data from Kinect-like sensors showing its general applicability.

Obviously, obstacles looking identical or very similar (regarding color and texture) to the ground will prevent the system from learning robust classifiers to distinguish ground from obstacles. Furthermore, the output of the visual classifiers becomes less significant when the obstacles vary strongly in appearance in the training data. In these cases, the distributions representing the obstacle appearance flatten and therefore distinguishing between obstacles and the floor becomes less reliable. One way to detect such situations is to classify the labeled training images directly after learning the classifiers. In case of large classification errors or uncertainties on the training image, the classifiers are not suited to detect the obstacles in the scene, and we can fallback to using only the laser.

4. Related Work

In the following, we discuss publications related to our paper. We first present obstacle detection techniques that were particularly developed for humanoid robots and, second, more general vision-based obstacle detection methods.

4.1. *Obstacle Detection Techniques for Humanoid Robots*

There exist several approaches that use external tracking systems to compute the position of obstacles. For example, Baudouin *et al.* employ such a system to locate obstacles in 3D and plan footsteps over non-static obstacles blocking the robot's path [12]. Stilman *et al.* also apply an external tracking system in order to plan actions for a humanoid navigating amongst movable objects [13] and Michel *et al.* to generate sequences of footsteps for avoiding planar obstacles [14].

There also exist techniques that use onboard sensing, like our approach. For instance, Stachniss *et al.* introduced a simultaneous localization and mapping system to learn accurate 2D grid maps of large environments with a humanoid equipped with a laser scanner located in the neck [15]. Such a map is subsequently used by

Faber *et al.* for humanoid localization and path planning [16]. During navigation, the robot carries out a potential field approach to locally avoid obstacles sensed with a 2D laser scanner and ultrasound sensors located at the hip. Also, Tellez *et al.* use 2D laser data to construct a 2D occupancy grid map which they use for path planning [17]. The laser scanners are mounted on the robot's feet and the robot has to walk in such a way that the feet are always parallel to the ground to obtain stable 2D measurements. Chestnutt *et al.* use 3D laser data acquired with a constantly sweeping scanner mounted at a pan-tilt unit at the humanoid's hip [11]. The authors fit planes through 3D point clouds and construct a height map of the environment. Afterwards, they distinguish between accessible areas and obstacles based on the height difference. Such sensor setups can only be used on robots with a significantly larger payload than the Nao. We augment navigation based on laser data with visual observations.

Several solutions to visual obstacle detection for humanoid robots have previously been presented. For instance, Gutmann *et al.* build a 2.5D height map given accurate stereo camera data and additionally update a 3D occupancy grid map to plan actions for the robot leading towards the goal [18,19]. Ozawa *et al.* also developed a system that relies on stereo data to construct a dense local feature map [20]. This system performs real-time localization and mapping with a humanoid robot based on 3D visual odometry in a local environment. Cupec *et al.* detect objects in monocular images and determine the robot's pose relative to these objects to adapt the trajectory accordingly. [21] This technique relies on the assumption that obstacles and floor are clearly distinguishable. Li *et al.* proposed a vision-based obstacle avoidance approach for the RoboCup domain [22]. The authors assume known shapes of the obstacles, i.e., the other field players and use gradient features learned from training images, a color-based classifier, and ultrasound data to determine obstacle positions. Their approach relies on a specific color coding of the scene which simplifies the problem. Our approach is also based on visual classification, but automatically generates training data from a laser range finder. Thereby it is possible to adapt to new environments without manual intervention. Furthermore, we obtain a dense classification of the images.

The approach presented in this article is an extension of our previous conference paper [23]. Compared to the conference paper, we present a significantly extended experimental evaluation of the approach and evaluate two different strategies to exploit neighborhood information. In addition to that, we extended our previous work so that it can now deal with dynamic objects in the scene during the training phase, which was not possible before.

4.2. *General Obstacle Detection Techniques Based on Visual Information*

In general, there exists a large variety of approaches for obstacle detection. For instance, for monocular cameras, some approaches obtain obstacle information by inferring geometric information from camera images[24 25 26]. Other techniques, similarly to our work, learn the appearance of obstacles from training data[27,28,29].

Further approaches combine visual information with other sensor data to overcome the limitations of a single sensor. In this context, Ohya *et al.* presented a self-localization and obstacle detection system that matches image edges to a map of vertical edges in a building[30]. Edges that can not be matched are treated as obstacles and are consequently avoided. To detect moving obstacles, the authors further employ ultrasonic sensors. Labayrade *et al.* proposed to combine laser data with stereo vision for obstacle detection in the context of autonomous automotives[31]. Obstacles are identified in the stereo vision data as well as in the laser range data. The proposed algorithm then checks for consistency between the obstacle locations as identified by both sensors to avoid false positives. An early approach in this field was proposed by Wang *et al.*[32]. Their approach uses color segmentation with adaptive thresholds to identify obstacles in a camera image. To avoid false positives, they check consistency with the image from a second camera. Both cameras are mounted rigidly on a vehicle.

Our approach also combines two sensors, namely a monocular camera and a laser range finder, but in our case, to enable self-supervised training. In the remainder of this section, we discuss publications that also apply automatic training. Fazl-Ersi and Tsotsos use stereo data to extract dense information about floor and obstacles[33]. The authors proposed to classify regions of neighboring pixels with similar color information and also consider the distances from the ground plane to distinguish between floor and obstacles. Subsequently, they learn color-based models for the two classes. Other authors proposed to divide the camera image into small rectangular patches and to compute feature vectors for each patch[34,35]. Consequently, they learn models for the traversability of a patch by clustering in feature-space. Both approaches obtain the training data from the robot's interaction with the environment, e.g., driving over the corresponding area. Such strategies are not applicable for humanoids as they can easily lose their balance.

Dahlkamp *et al.* use vision for extending the perception range of the autonomous car Stanley to allow for faster and more forward-looking driving[36]. The authors apply laser-based data for learning of a vision-based obstacle classifier. In contrast to their work, we combine different visual cues, i.e., color and texture, to distinguish between obstacles and the floor. We furthermore consider dependencies in neighboring regions in the image and can deal with dynamic elements in the scene during the self-supervised learning phase.

5. Conclusions

In this paper, we presented an efficient approach to vision-based obstacle detection for collision-free humanoid navigation. Our system uses sparse laser data to train image-based classifiers online in a self-supervised way. By analyzing image sequences using optical flow to identify moving objects, our technique is able to perform the training also in dynamic scenes. After learning the visual obstacle classifiers, they can be used for collision-free real-time navigation.

We thoroughly evaluated our system in real-world experiments with a Nao humanoid. The experiments illustrate that the robot avoids obstacles reliably, navigates efficiently, and can deal with dynamic obstacles in the scene. By applying our learning approach to RGB-D images, we illustrated that our approach is generally applicable when combining vision and depth data and that it can be used to support navigation systems relying on Kinect-like sensors.

Acknowledgments

The authors would like to acknowledge Armin Hornung and Christoph Sprunk for their help in the context of mobile robot navigation, Jörg Müller for his help with the motion capture system, and Andreas Ess for fruitful discussions on the topic. We would further like to thank Thomas Brox for his advice on optical flow algorithms.

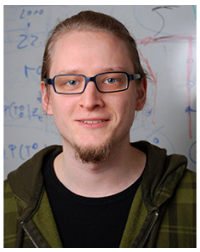
This work has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 and within the Research Training Group 1103 as well as by Microsoft Research, Redmond. Their support is gratefully acknowledged.

References

1. N. Ahmed, T. Natarajan and K. R. Rao, Discrete cosine transfom, *IEEE Transactions on Computers* **23**(1) (1974) pp. 90–93.
2. Y.-L. Huang and R.-F. Chang, Texture features for dct-coded image retrieval and classification, in *Acoustics, Speech, and Signal Processing*, vol. 6 (1999).
3. H. Nezamabadi-Pour and S. Saryazdi, Object-based image indexing and retrieval in dct domain using clustering techniques, in *Proceedings of World Academy of Science Engineering and Technology*, vol. 3 (2005).
4. A. Rosenfeld, R. Hummel and S. Zucker, Scene labeling by relaxation operations, *IEEE Trans. Systems. Man. Cybernet* **6**(6) (1976) pp. 420–433.
5. H. Yamamoto, A method of deriving compatibility coefficents for relaxation operators, *Compt. Graph. Image Processing* **10** (1979) pp. 256–271.
6. Y. Bengio, O. Delalleau and N. Le Roux, Label propagation and quadratic criterion, in O. Chapelle, B. Schölkopf and A. Zien (eds.) *Semi-Supervised Learning* (MIT Press, 2006), pp. 193–216.
7. H. Moravec and A. Elfes, High resolution maps from wide angle sonar, in *IEEE Int. Conf. on Robotics and Automation (ICRA)* (1985).
8. A. Hornung and M. Bennewitz, Humanoid robot localization in complex indoor environments, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2010).
9. C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines* (2001), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
10. N. Sundaram, T. Brox and K. Keutzer, Dense point trajectories by GPU-

accelerated large displacement optical flow, in *European Conference on Computer Vision (ECCV)* (Crete, Greece, 2010).
11. J. Chestnutt, Y. Takaoka, K. Suga, K. Nishiwaki, J. Kuffner and S. Kagami, Biped navigation in rough environments using on-board sensing, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2009).
12. L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse and E. Yoshida, Real-time replanning using 3d environment for humanoid robot, in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)* (2011).
13. M. Stilman, K. Nishiwaki, S. Kagami and J. Kuffner, Planning and executing navigation among movable obstacles, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2006).
14. P. Michel, J. Chestnutt, J. Kuffner and T. Kanade, Vision-guided humanoid footstep planning for dynamic environments, in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)* (2005).
15. C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke and W. Burgard, How to learn accurate grid maps with a humanoid, in *IEEE Int. Conf. on Robotics and Automation (ICRA)* (2008).
16. F. Faber, M. Bennewitz, C. Eppner, A. Goeroeg, A. Gonsior, D. Joho, M. Schreiber and S. Behnke, The humanoid museum tour guide Robotinho, in *18th IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN)* (2009).
17. R. Tellez, F. Ferro, D. Mora, D. Pinyol and D. Faconti, Autonomous humanoid navigation using laser and odometry data, in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)* (2008).
18. J.-S. Gutmann, M. Fukuchi and M. Fujita, A floor and obstacle height map for 3D navigation of a humanoid robot, in *IEEE Int. Conf. on Robotics and Automation (ICRA)* (2005).
19. ——, 3D perception and environment map generation for humanoid robot navigation, *Int. Journal of Robotics Research (IJRR)* **27**(10) (2008) pp. 1117–1134.
20. R. Ozawa, Y. Takaoka, Y. Kida, K. Nishiwaki, J. Chestnutt, J. Kuffner, S. Kagami, H. Mizoguchi and H. Inoue, Using visual odometry to create 3d maps for online footstep planning, in *IEEE Intl. Conf. on Systems, Man, and Cybernetics* (2005).
21. R. Cupec, G. Schmidt and O. Lorch, Experiments in vision-guided robot walking in a structured scenario, in *IEEE Int. Symp. on Industrial Electronics* (2005).
22. X. Li, S. Zhang and M. Sridharan, Vision-based safe local motion on a humanoid robot, in *Workshop on Humanoid Soccer Robots* (2009).
23. D. Maier, M. Bennewitz and C. Stachniss, Self-supervised Obstacle Detection for Humanoid Navigation Using Monocular Vision and Sparse Laser Data, in *IEEE Int. Conf. on Robotics and Automation (ICRA)* (Shanghai, China, 2011).
24. H. Wang, K. Yuan, W. Zou and Y. Peng, Real-time obstacle detection with a single camera, in *IEEE Int. Conf. on Industrial Technology (ICIT)* (2005).

25. Y.-G. Kim and H. Kim, Layered ground floor detection for vision-based mobile robot navigation, in *IEEE Int. Conf. on Robotics and Automation (ICRA)* (2004).
26. E. Einhorn, C. Schrter and H. Gross, Monocular scene reconstruction for reliable obstacle detection and robot navigation, in *European Conf. on Mobile Robots (ECMR)* (2009).
27. I. Ulrich and I. Nourbakhsh, Appearance-based obstacle detection with monocular color vision, in *National Conf. on Artificial Intelligence (AAAI)* (2000).
28. J. Michels, A. Saxena and A. Ng, High speed obstacle avoidance using monocular vision and reinforcement learning, in *Int. Conf. on Machine Learning (ICML)* (2005).
29. C. Plagemann, C. Stachniss, J. Hess, F. Endres and N. Franklin, A nonparametric learning approach to range sensing from omnidirectional vision, *Robotics and Autonomous Systems* **58** (2010) pp. 762–772.
30. I. Ohya, A. Kosaka and A. Kak, Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing, *IEEE Trans. on Robotics and Automation* **14** (1998) pp. 969 – 978.
31. R. Labayrade, C. Royere, D. Gruyer and D. Aubert, Cooperative fusion for multi-obstacles detection with use of stereovision and laser scanner, *Autonomous Robots* **19** (2005) pp. 117–140.
32. J. Wang, F. Zhu, J. Wu and X. Xu, Road following and obstacle detection for the mobile robot visual navigation, in *Proceedings of the third International Symposium on Robotics and Manufacturing* (1990), pp. 887–892.
33. E. Fazl-Ersi and J. Tsotsos, Region classification for robust floor detection in indoor environments, in *Int. Conf. on Image Analysis and Recognition (ICIAR)* (2009).
34. D. Kim, J. Sun, S. M. Oh, J. M. Rehg and A. F. Bobick, Traversability classification using unsupervised on-line visual learning for outdoor robot navigation, in *IEEE Int. Conf. on Robotics and Automation (ICRA)* (2006).
35. L. Ott and F. Ramos, Unsupervised incremental learning for long-term autonomy, in *IEEE Int. Conf. on Robotics and Automation (ICRA)* (2012).
36. H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun and G. Bradski, Self-supervised monocular road detection in desert terrain, in *Robotics: Science and Systems (RSS)* (2006).



Daniel Maier studied at the University of Freiburg and the University of Washington. He received his Diplom (M.S.) in Computer Science from the University of Freiburg in 2010. Currently, he is a PhD student in the humanoid robots laboratory at the University of Freiburg. His research focuses on autonomous navigation and perception.

Cyrill Stachniss is a lecturer at the University of Freiburg in Germany. Before being a lecturer, he was a postdoc at Freiburg University, served as a guest lecturer at the University of Zaragoza, and worked as senior researcher at the Swiss Federal Institute of Technology. He received his PhD degree in 2006. Since 2008, he is an associate editor of the IEEE Transactions on Robotics and since 2010 a Microsoft Research Faculty Fellow. In his research, he focuses on probabilistic techniques in the context of mobile robotics, perception, learning, and navigation problems.

Maren Bennewitz is an assistant professor for Computer Science at the University of Freiburg in Germany. She got her Ph.D. in Computer Science from the University of Freiburg in 2004. From 2004 to 2007, she was a Postdoc in the humanoid robots laboratory at the University of Freiburg which she heads since 2008. The focus of her research lies on robots acting in human environments. In the last few years, she has been developing novel solutions for intuitive human-robot interaction and navigation of robots in complex indoor environments.