Robotics and Autonomous Systems 59 (2011) 296–309

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Attention-driven monocular scene reconstruction for obstacle detection, robot navigation and map building☆

E. Einhorn [a,b,*], Ch. Schröter [a], H.M. Gross [a]

[a] Neuroinformatics and Cognitive Robotics Lab, Ilmenau University of Technology, Germany

[b] MetraLabs GmbH, Germany

ARTICLE INFO

Article history:
Available online 5 March 2011

Keywords:
Shape-from-motion
Visual obstacle detection
Monocular vision
Attention
EKF

ABSTRACT

In this paper, we present a feature-based approach for monocular scene reconstruction based on Extended Kalman Filters (EKF). Our method processes a sequence of images taken by a single camera mounted frontally on a mobile robot. Using a combination of various techniques, we are able to produce a precise reconstruction that is free from outliers and can therefore be used for reliable obstacle detection and 3D map building. Furthermore, we present an attention-driven method that focuses the feature selection to image areas where the obstacle situation is unclear and where a more detailed scene reconstruction is necessary. In extensive real-world field tests we show that the presented approach is able to detect obstacles that are not seen by other sensors, such as laser range finders. Furthermore, we show that visual obstacle detection combined with a laser range finder can increase the detection rate of obstacles considerably, allowing the autonomous use of mobile robots in complex public and home environments.

1. Introduction and related work

For nearly ten years we have been involved in the development of an interactive mobile shopping assistant for everyday use in public environments, such as shopping centers or home improvement stores. Such a shopping companion autonomously contacts potential customers, intuitively interacts with them, and adequately offers its services, including autonomously guiding customers to the locations of desired goods [1]. As part of long-term field trials, 9 shopping robots have been in daily use in three home improvement stores in Germany since March 2008. Currently, we are developing an interactive assistant that operates in home environments as companion for elderly people with mild cognitive impairments (MCI) living at home alone. Both, public environments like home improvement stores, as well as home environments like kitchens or living rooms, contain a large variety of different obstacles that must be detected by an autonomous robot.

For obstacle detection the robots are equipped with an array of 24 sonar sensors at the bottom and a laser range finder SICK S300 mounted in front direction at a height of 0.35 m as shown in Fig. 1. Using these sensors, most of the obstacles can be reliably detected. However, during the field trials it became apparent that there remain certain obstacles which are very difficult to recognize.

The main extent of some obstacles like shopping carts or tables are mainly located above the plane that is covered by the laser range finder. Also, small obstacles like flat pallets are difficult to perceive since they lie below the laser range finder and can hardly be seen by the sonar sensors due to the diffuse characteristics and low precision of the latter. Therefore, it turned out to be necessary to use additional methods for robust and reliable obstacle detection. Vision-based approaches are suitable for this purpose since they provide a large field of view and supply a large amount of information about the structure of the local surroundings.

Recently, time-of-flight cameras have been used successfully for obstacle detection [2]. Similar to laser range finders, these cameras emit short light pulses and measure the time taken until the reflected light reaches the camera again. Due to their high costs these cameras may be suitable for robot prototypes but at present are no option for a series product that we are planning to develop. Another alternative is to use stereo vision as described in [3]. However, a stereo camera for detecting distant obstacles requires a large base distance. Compared to a single camera it is therefore less compact and difficult to protect from damage and vandalism.

In [4] a monocular approach for depth estimation and obstacle detection is presented. Information about the scene's depth is drawn from the scaling factor of image regions, which is

☆ The research leading to these results has received funding from the European Community's Seventh Framework Program [FP7/2007-2011] (CompanionAble: http://www.companionable.net/, Grant #216487).

* Corresponding author at: Ilmenau University of Technology, Neuroinformatics and Cognitive Robotics Lab, 98693 Ilmenau, Germany. Tel.: +49 3677 69 1305; fax: +49 3677 69 1665.

E-mail address: Erik.Einhorn@tu-ilmenau.de (E. Einhorn).

0921-8890/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2011.02.008

determined using region tracking. While this approach may work well in outdoor scenes where the objects near the focus of expansion are separated from the background by large depth discontinuities, it will fail in cluttered indoor environments like home improvement stores. In [5] we propose an early version of a feature-based approach for monocular scene reconstruction. This shape-from-motion approach uses Extended Kalman Filters (EKF) to reconstruct the 3D position of the image features in real-time in order to identify potential obstacles in the reconstructed scene. Davison et al. [6,7] use a similar approach and have done a lot of research in this area. They propose a full covariance SLAM algorithm for recovering the 3D trajectory of a monocular camera. Both the camera position and the 3D positions of tracked image features or landmarks are estimated by a single EKF. Another visual SLAM approach was developed by Eade and Drummond [8]. Their graph-based algorithm partitions the landmark observations into nodes of a graph to minimize statistical inconsistency in the filter estimates [9].

However, Eade and Drummond's "Visual SLAM" as well as Davison's "MonoSLAM" are both mainly focusing on the estimation of the camera's motion, while a precise reconstruction of the scenery is less important. As we want to use the reconstructed scene for obstacle detection and local map building, our priorities are vice versa. We are primarily interested in a precise and dense reconstruction of the scene and do not focus on the correct camera movement, since the distance of the objects relative to the camera and the robot respectively is sufficient for obstacle avoidance. Actually, we are using the robot's odometry to obtain information on the camera's movement. In contrast to Eade et al. and Davison et al. who use a hand-held camera which generally is moved sidewards in their examples, our camera is mounted in front of the mobile robot and, therefore, moves along its optical axis (see Fig. 1). Compared to lateral motion, this forward motion leads to higher uncertainties in the depth estimates due to a smaller parallax as proven in a sensitivity analysis by Matthies et al. [10]. This makes the monocular reconstruction more difficult and the estimation must be applied over a longer base distance.

The approach presented in [11,12] is most related to our work. The authors also use a single camera that is oriented towards the robot's movement direction in order to obtain information used for obstacle detection. For that purpose, features are tracked in two consecutive frames. The optical flow vectors formed by the corresponding features are used to compute time-to-contact values that resemble the estimated time until a collision occurs and are an implicit measure for the obstacle distances. However, the approach does not use a single closed-form solution to handle the problem. Instead, three different types of movement like constant motion, accelerated lateral motion and rotational motion are distinguished using the robot's odometry and are handled separately. Therefore, the algorithm may fail if the robot's movement is not well represented by one of these three motion models. Additionally, the time-to-contact measurements are evaluated using two frames of the image sequence only, without taking into account information that was inferred in previous iterations. Although the resulting noisy and oscillating measurements have been reduced in [12], accumulating all information in a probabilistic approach could lead to more stable results.

Most of the aforementioned feature-based approaches apply interest operators like the Shi–Tomasi corner detector or the Harris corner detector for feature selection. These detectors provide a bottom-up feature selection scheme where the position and number of the chosen features depend on the content of the input images. This results in a risk of missing obstacles. Taking top-down knowledge into account could lead to better results by choosing features in image regions that result in the largest information gain

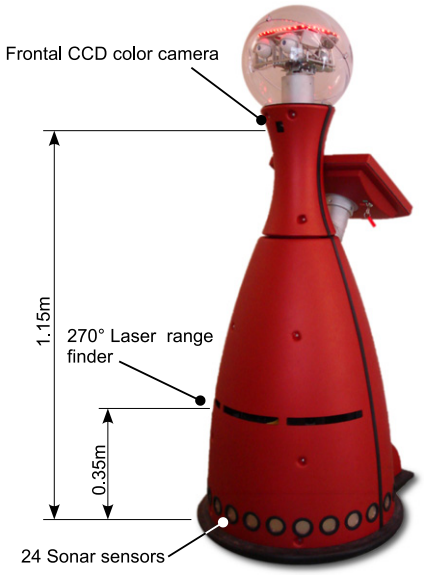


Fig. 1. The robot platform SCITOS A5, a joint development of MetraLabs GmbH and the Neuroinformatics and Cognitive Robotics Lab, that is used for our experiments and field trials is equipped with sonar sensors, a laser range finder and a frontal CCD camera that is tilted towards the ground.

for the environment knowledge instead of choosing the features based on the information content of the images only.

In [13,14] such a top-down approach is presented, not for feature selection, but for feature tracking using an improved active search strategy. In [15] the authors present a visual SLAM approach for hand-held cameras that instructs the user to perform position and orientation changes of the camera to optimize the localization. The actions and movements are chosen so as to maximize the mutual information gain between posterior states and measurements.

Another active vision approach is presented by Frintrop and Jensfelt [16], where the camera is controlled by an active gaze control module according to three behaviors for redetection of known features, tracking of features and detection of new features in unknown areas. Using a predesigned decision tree the system switches between these behaviors depending on the number and expected location of known features.

In summary, the above visual SLAM algorithms use the active vision approach basically for controlling the camera's viewing direction in a way to improve the camera's position estimates and to enhance loop closings.

One main contribution of this paper is a monocular feature-based approach for scene reconstruction that combines a number of different techniques that are known from research areas like Visual SLAM or stereo vision. A second important contribution is an attention-driven approach for feature selection. In contrast to some publications mentioned above, we are using a fixed camera with a wide-angle lens whose viewing direction cannot be altered dynamically. However, instead of moving the whole camera we can choose particular image regions that our algorithm pays more attention to. By combining bottom-up and top-down information we select features in those image regions that provide the highest information gain for our obstacle detection algorithm. By choosing new features at the right places we can detect more obstacles, allowing us to reduce the total number of reconstructed features without increasing the risk of missing obstacles. This results in an improved performance of the whole obstacle detection algorithm.

In the next section, we describe our approach for monocular scene reconstruction in detail. In Section 3 we present methods that transform the reconstructed information into different three-dimensional representations of the robot's environment like

occupancy voxel maps and textured triangle meshes. These representations are finally used for navigational tasks like obstacle avoidance. Additionally, the created voxel maps provide top-down information for our novel attention-driven feature selection scheme that is presented in Section 4. Finally, we present experimental results and conclude with an outlook for future work.

2. Monocular scene reconstruction

As stated before, we use a single calibrated camera that is mounted in front of the robot (see Fig. 1). During the robot's locomotion, the camera captures a sequence of images $(\ldots, I_{t-1}, I_t, I_{t+1}, \ldots)$ that are rectified immediately according to the intrinsic camera parameters in order to correct the lens distortions. Without loss of generality we notate the latest image that was captured at the current moment with I_0, while I_i with $i > 0$ denotes images, that were previously recorded. Using the robot's odometry data, the corresponding camera position, expressed by its projection matrix $\mathbf{P}_i = \mathbf{K}\mathbf{R}_i[\mathbf{I} \mid -\mathbf{c}_i]$, can be computed for each image I_i, containing the orientation $\mathbf{R}_i$, the position $\mathbf{c}_i$, and the intrinsic calibration matrix $\mathbf{K}$ of the camera (see [17] for details). Both the camera's position and its orientation are expressed with respect to a global reference coordinate frame.

This preprocessing step yields different two-dimensional views of a scene including the projection matrix of the corresponding ideal pinhole camera. For scene reconstruction we use a feature based approach. Distinctive image points (image features) are detected in the preprocessed input images and tracked over the acquired image sequence while the 3D positions of these features are estimated using EKFs. Similar to the camera's pose, these positions are computed with respect to the same global reference coordinate frame.

The camera pose obtained using the odometry is used for feature tracking only. For most of the other processing steps, we use a corrected camera pose in order to achieve a better accuracy. Therefore, we perform an odometry correction step that uses the tracked features to correct systematic and non-systematic odometry errors. We will get back to this with some more information in Section 2.6. The complete architecture of our approach is depicted in Fig. 2.

2.1. Feature selection

For feature detection, we are currently using the Shi–Tomasi corner detector [18]. Previously, we applied the "FAST" high-speed corner detector [19], which in general achieves similar results while requiring less computation time. However, it occasionally produced inferior detections during our field tests and selected image points that were difficult to track and hence resulted in wrong estimations.

In contrast to other authors and our previous publications, our current implementation does not select the features uniformly over the image. Instead, we use an attention-driven selection scheme that focuses the feature selection to image areas where the highest information gain for obstacle detection and mapping can be achieved by selecting new features. More details on this attention-driven feature selection follow in Section 4.

We also experimented with SIFT features [20] for feature selection and tracking. However, SIFT still requires too much computation time and does not allow one to process as many frames per second as we require. Additionally, using the SIFT descriptor for tracking did not improve the results of the scene reconstruction compared to our tracking approach.

2.2. State representation

Davison et al. [6,7] use a single EKF for full covariance SLAM, i.e. for recovering the camera's pose as well as the 3D positions of the tracked image features simultaneously. In this algorithm the inversion of the innovation covariance matrix as part of the EKF update will dominate the overall runtime, resulting in a complexity of $O(n^3)$, where n denotes the number of features. Currently, such an approach is able to handle only up to 100 features in real-time.

In [21] the computation of pose and structure is split into two steps. In the first step, a single EKF is applied to recover the camera's position using a fixed number of reconstructed features. During the second step, n EKFs are used to recover the 3D positions of n features, where one EKF is used per feature. Both steps are repeated in an interleaved way. Obviously, this is a coarse approximation of the full covariance SLAM since correlations between the different features are not taken into account. However, this approximation results in a heavy reduction of the computational complexity to $O(m^3) + O(n)$. Here $O(m^3)$ – the complexity of the pose estimation during the first step – is constant, since the number m of features that are used in the first step remains constant, too. Thus, the overall complexity for large feature counts n is $O(n)$. Since we require a dense reconstruction of the scene for obstacle detection, we have to cope with a large number of features, which cannot be handled by a full covariance SLAM approach in real-time. Therefore, we also use one EKF per feature to recover the structure of the scene similar to [21]. Each feature j is associated with a state vector $\mathbf{y}^j$ that represents the 3D position of the feature and a corresponding covariance matrix $\boldsymbol{\Sigma}^j$. For better readability we drop the superscripts for different features in the following.

Different parameterizations for the 3D positions of the features have been proposed in the literature. The most compact representation is the XYZ-representation where the position of each feature is parameterized by its Euclidean coordinates in 3D space. Davison et al. [7] have shown that this representation has several disadvantages since the position uncertainties for distant features are not well represented by a Gaussian distribution. Instead, they propose an inverse depth representation, where the 3D position of each feature j can be described by the following vector:

$$\mathbf{y} = (\mathbf{c}, \theta, \varphi, \lambda)^\top, \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^3$ is the optical center of the camera from which the feature was first observed, and θ, φ is the azimuth and elevation of the unit ray that points from $\mathbf{c}$ to the 3D point of the feature. This ray is given by its direction vector:

$$\mathbf{m}(\theta, \varphi) = (\cos\theta\cos\varphi, \cos\theta\sin\varphi, -\sin\theta)^\top. \tag{2}$$

The last element λ of the state vector in equation Eq. (1) denotes the inverse of the features depth $d = \lambda^{-1}$ along the ray.

Parsley and Julier [22] have shown that this inverse depth λ might become negative during the EKF update and proposed an alternative negative logarithmic parameterization where the inverse depth λ is replaced by $l = -\log(d)$. In our experiments, this parameterization resulted in an inferior convergence of the EKFs. Therefore, we keep using the inverse depth parameterization here.

2.3. Feature state initialization

After new features were selected their 3D positions, i.e. their corresponding EKF states, must be initialized using a suitable a priori estimate. For this purpose various methods have been proposed in the related literature. A common way is to assume a constant depth of the features [21] and to compute the features' 3D positions along the ray that is defined by the camera's position and the image position of the feature (see Eq. (3)). This may be valid if the extent of the object is limited and its approximate distance to the camera is known. However, if the camera is moving through a large indoor or outdoor environment, this kind of initialization

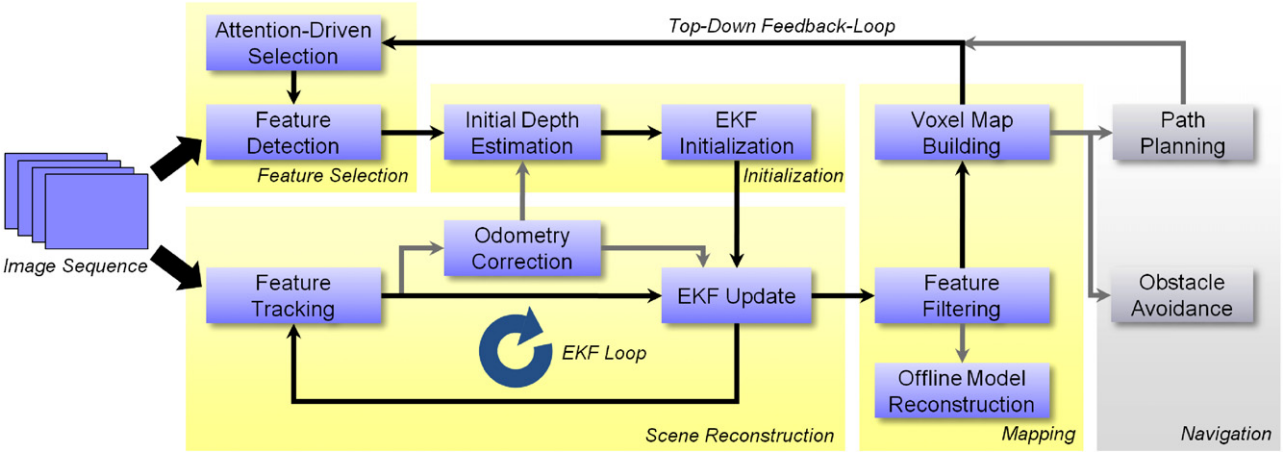


Fig. 2. The architecture of our approach. The black arrows show the main information flow between the components that are described in this paper. Features are selected in the input images using an attention-driven approach. After computing a reliable initial estimate using a classic depth estimation approach the features join the main scene reconstruction loop, where they are tracked in the captured image sequence while reconstructing their 3D positions using EKFs. The resulting point cloud is then filtered before building a voxel map that is used for navigation tasks and that provides information for the attention-driven feature selection.

yields large initial errors, slow convergence and inferior position estimates.

Another method for initializing the features is to choose their depths in a way that the height of the initial feature position is zero, i.e. the features are initialized on the ground plane. This kind of initialization can be used for tilted cameras. It has certain advantages when used for obstacle detection since false positive detections are reduced. Using this method, we achieved good results for obstacle detection, although it leads to high initial model errors, since many points are initialized at too large depths.

A different method is to choose the depth of new features according to the reconstructed 3D position of older features in the image neighborhood, whose positions have already been estimated during previous iterations.

A more sophisticated approach was presented in [23] where new features are first estimated using a particle filter (PF) until they join the EKF estimation framework. Although the initialization is less important for the convergence of the EKFs if an inverse depth parameterization is used [7], it is essential for an approach used for obstacle detection to obtain a reliable estimate as early as possible in the estimation process. Therefore, a delayed feature initialization using particle filters is not applicable for our purposes.

Instead, we have proposed another method in [5] which uses a classic multi-baseline stereo approach for initializing new features. The approach uses the images that were captured *before* the features were first detected and therefore can immediately obtain valid depth estimates for newly detected features. Hence, such a hybrid approach can react quickly on obstacles that suddenly appear in front of the robot.

The approach is inspired by the work of Bunschoten and Kröse [24], where a multi-baseline depth estimation algorithm for panoramic image data is presented. Based on their work we are using a similar correlation-based algorithm for pinhole cameras.

Let $\mathbf{x}_0$ be the image position of a newly detected feature in the image I_0 (see Fig. 3), then its corresponding 3D scene point $\mathbf{y}$ must be located on the back-projected ray in homogeneous coordinates[1]:

$$\tilde{\mathbf{y}}(\lambda) = \mathbf{P}_0^+ \tilde{\mathbf{x}}_0 + \lambda \tilde{\mathbf{c}}_0, \tag{3}$$

[1] For better differentiation we notate homogeneous vectors as $\tilde{\mathbf{x}}$ and Euclidean vectors as $\mathbf{x}$, where $\tilde{\mathbf{x}} = (\mathbf{x}, 1)^\top \cdot s, s \in \mathbb{R}$.

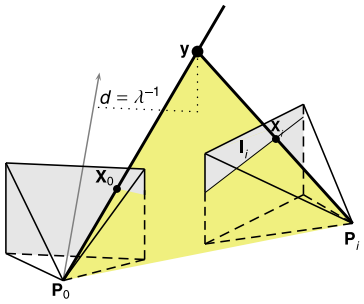


Fig. 3. Epipolar geometry of two cameras $\mathbf{P_0}$ and $\mathbf{P_i}$. The 3D point $\mathbf{y}$ is located on the ray defined by the camera's center $\mathbf{c}_0$ and the image point $\mathbf{x}_0$. The corresponding image point $\mathbf{x}_i$ can be found along the epipolar line $\mathbf{l}_i$.

where $\mathbf{P}_0^+$ is the pseudoinverse of the camera projection matrix and $\mathbf{c}_0$ is the camera's center in image I_0. If the 3D scene point $\mathbf{y}$ is visible in a previously recorded image I_i from a different position, it must be located on the projection of the ray on the image plane of image I_i:

$$\tilde{\mathbf{x}}_i(\lambda) = \mathbf{P}_i \mathbf{P}_0^+ \tilde{\mathbf{x}}_0 + \lambda \mathbf{P}_i \tilde{\mathbf{c}}_0. \tag{4}$$

This equation is the parametric description of the epipolar line that $\mathbf{x}_0$ induces in image I_i. In order to estimate the depth of the image point $\mathbf{x}_0$, we subsequently move along the epipolar line by varying the inverse depth parameter λ. For each candidate $\mathbf{x}_i$ the similarity with pixel $\mathbf{x}_0$ is evaluated. As measure of correlation we use the sum of absolute differences (SAD) between windows centered at $\mathbf{x}_i$ and $\mathbf{x}_0$. As in [24] and [25] we accumulate the SAD values obtained for different images I_i for the same $\mathbf{x}_0$ and λ. Eventually, the most likely depth value for the given pixel $\mathbf{x}_0$ can be determined immediately by means of the parameter λ^* which yields the minimal accumulated SAD.

This inverse depth value λ^* is used as reliable initial inverse depth estimate to initialize the EKF of the newly detected feature. Additionally, the variance of the SAD values near the minimum λ^* is computed and used for initializing the error covariance matrix $\boldsymbol{\Sigma}$ of the feature's EKF. For computing the variance, the SAD values for each computed λ around the minimum λ^* are normalized so that their sum equals 1. Hence each SAD value (respectively its inverse) can be treated as a probability $p(\lambda)$. The variance σ^2 is then defined by:

$$\sigma^2 = \sum_{\lambda} (\lambda - \lambda^*)^2 * p(\lambda). \tag{5}$$

For the described multi-baseline depth estimation the approach automatically chooses different previously recorded images where the corresponding camera positions have a base distance of at least 0.1 m to each other. Since the buffer for storing the recorded images has a fixed size, the number of used images depends on the frame rate and the robot's speed. It typically varies between 3–5 images.

2.4. Feature tracking

While the robot is moving, previously selected image features are tracked in subsequent frames. In [5] we used a feature matching approach that finds correspondences between homologous features in subsequent frames based on a bipartite graph matching. Such an approach is suitable if features are extracted in each frame independently as it is done with SIFT or SURF features. However, with less complex feature descriptors like image patches it has several shortcomings and tracking over long image sequences is less stable.

Therefore, we now use a guided active search for tracking the features through the image sequence. As descriptor we utilize a 16×16 pixel image patch around each feature. As measure of similarity we again use the sum of absolute differences (SAD). This simple tracking approach has a low computation time and achieves good tracking results for image sequences of a moving camera. Descriptors that are invariant to rotation and scale do not improve the tracking results significantly since scale changes between consecutive frames of the image sequence are small.

For tracking, the image position $\mathbf{x}^-$ of each feature is predicted by projecting the current estimate of its 3D position $\mathbf{y}$ back onto the image plane using $\tilde{\mathbf{x}}^- = h(\mathbf{y}, \mathbf{P})$ with the measurement function:

$$h(\mathbf{y}, \mathbf{P}) = \mathbf{P}\left(\lambda\tilde{\mathbf{c}} + \begin{pmatrix}\mathbf{m}(\theta, \varphi)\\ 0\end{pmatrix}\right). \tag{6}$$

As stated before, the camera's projection matrix in the above equation is computed using the robot's odometry that may contain errors. Hence, the predicted image position might be erroneous too and the corresponding image point must be searched in the current image in a search region around the predicted image position $\mathbf{x}^-$. This is done by computing the SAD with the image patch that is stored as a descriptor of the feature. The image point that yields the lowest SAD is chosen. To achieve sub-pixel precision, we fit a 2D parabola into the computed SAD error surface around the chosen image point and use the coordinates of the apex as position of the corresponding image point. The search is restricted to an elliptical region that is defined by projecting the error covariance $\boldsymbol{\Sigma}$ of the feature's 3D position estimate to the image plane. The covariance matrix of this elliptical region is computed within the EKF and is known as the innovation covariance:

$$\mathbf{S} = \mathbf{H}\boldsymbol{\Sigma}\mathbf{H}^\top + \mathbf{R}, \tag{7}$$

where $\mathbf{H}$ denotes the Jacobian of the measurement function in equation Eq. (6) and $\mathbf{R}$ is the 2×2 measurement covariance matrix that is set to $\mathbf{R} = 5\mathbf{I}$ in our implementation. The measurement covariance matrix also represents the uncertainty in the camera's position, since an error in the camera's pose results in an erroneous projection on the image plane and hence an erroneous measurement.

One major problem of patch-based approaches for feature matching are occlusions near object edges, where the patch covers two different objects with large depth discontinuities (see Fig. 4(a)). During the matching, this leads to a decision conflict since the part of the patch that belongs to the background object moves in a different way than the foreground object. As a result, the reconstructed 3D points along object borders are blurred in different depths. For stereo matching, various adaptive window approaches have been proposed to tackle this problem.

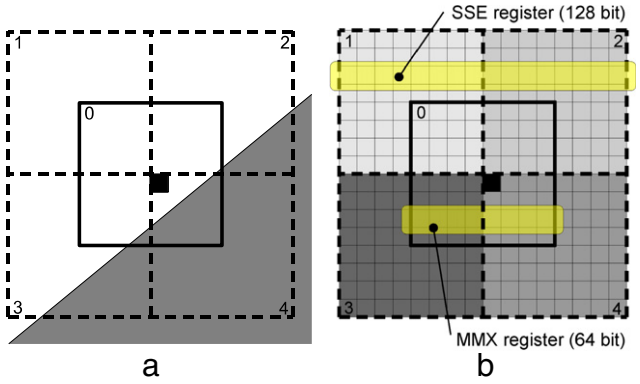


Fig. 4. (a) The correlation window is split into 5 sub-windows and allows better tracking along object boundaries. The black square indicates the center pixel. (b) For computing the SAD of each sub-window the data can be stored efficiently in SSE/MMX registers.

Here, we apply a variation of the multiple window approach presented in [26,27]. Instead of using a single 16×16 pixel correlation window, the window is split into five sub-windows as shown in Fig. 4. The SADs are computed for each sub-window C_i. The final correlation value C is formed by adding the correlation value C_o of the central sub-window and the values of the two best surrounding correlation windows C_b and C_s:

$$C = C_0 + C_b + C_s, \qquad b = \operatorname*{argmin}_{i>0} C_i, \qquad s = \operatorname*{argmin}_{i>0, i\neq b} C_i. \tag{8}$$

This measure of similarity performs better near object boundaries since at least two sub-windows are located on a single object in most cases. Depending on the dominant image structure, the correspondence is either attached to the foreground or the background object, reducing the blur along the reconstructed object borders. Using the SSE2 processor instruction `PSADBW`, the correlation values can be computed efficiently. This instruction simultaneously computes the SAD for 16 consecutive pixels within one 128 bit SSE register. In doing so, the SADs for the first 8 pixels that belong to the left sub-window are summed separately from the back most pixels that belong to the right sub-window. Therefore, splitting the window into 5 sub-windows results in very little computational overhead compared to using a single correlation window. This performance improvement is a major reason for choosing the SAD as similarity measure. Besides the correlation value, we compute an occlusion score by adding the correlation values of the two worst matching surrounding sub-windows:

$$C_{\text{occ}} = C_b + C_s, \qquad b = \operatorname*{argmax}_{i>0} C_i, \qquad s = \operatorname*{argmax}_{i>0, i\neq b} C_i. \tag{9}$$

Both the correlation value C and the occlusion score C_{occ} are normalized by the number of pixels in the used sub-windows.

2.5. Descriptor update

Davison et al. [7] also use the image patch around the feature as a descriptor. While they capture this descriptor only once when the feature is first observed, we used a contrary philosophy in [5], where we update the descriptor every time the feature is tracked in a new image. Both variants have pros and cons. If the descriptor is never adapted, the feature cannot be tracked over long distances since the appearance changes too much due to affine and perspective deformations, especially when using a forward moving camera or robot. If, on the other hand, the descriptor is updated with every frame, tracking errors might be accumulated

over several frames, and the descriptor could move along the edges of object boundaries and would not represent a single fixed feature. This usually occurs near occlusions and leads to incorrect estimates.

Therefore, we use the aforementioned occlusion score C_{occ} to determine whether updating the descriptor is reasonable or not. If the normalized occlusion score C_{occ} exceeds a certain threshold the descriptor remains unchanged, otherwise it is replaced by the corresponding patch in the current image. Using this technique, most features can be tracked over long distances while the projective deformations are compensated by permanent descriptor updates. Feature descriptors near occlusions are not updated to allow stable tracking along object boundaries.

2.6. Odometry correction

As stated before, we use the robot's odometry to retrieve the position of the camera for each image. However, due to latencies in transmitting the image data from the camera to the memory the time when the images were captured cannot be determined precisely. Depending upon the current CPU usage and load on the main bus, the delay may vary between 30 and 50 ms. Therefore, the odometry data cannot be assigned exactly to an image. These inaccuracies constitute a negative impact, especially if the angular velocity of the robot is changing rapidly. Additional errors are caused by the joggle of the camera when driving over a bumpy floor.

To correct these inaccuracies, we tried to estimate the camera's position using different methods: In addition to the EKFs of the features' positions, we used another EKF in an interleaved way similar to [21]. Alternatively, we applied a Gauss–Newton method to estimate the orientation of the camera by minimizing the back-projection error. Both, the EKF and the Gauss–Newton method were able to recover the camera's pose or orientation respectively but did not achieve a higher precision than SCITOS's odometry, which already has a good accuracy. As an alternative we tried to use a particle filter (PF) for estimating the camera pose. First the particles are updated using a motion model. Then we choose a constant number of $m = 15$ features that were tracked in the current image. Thereby, features whose 3D positions are estimated with sufficient precision are chosen in a way to cover the image uniformly. The importance weight of each particle k is then computed by adding the squared Mahalanobis distances between the projected 3D positions $\mathbf{x}_i^{-(k)} = h(\mathbf{y}_i, \mathbf{P}_k)$ of the selected features and their tracked image position $\mathbf{x}_i$ with respect to the innovation covariance $\mathbf{S}_i$ from Eq. (7):

$$w_k = -\log\left(\sum_{i=0}^{m}\left(\mathbf{x}_i^{-(k)} - \mathbf{x}_i\right)^{\top}\mathbf{S}_i^{-1}\left(\mathbf{x}_i^{-(k)} - \mathbf{x}_i\right)\right), \tag{10}$$

where $\mathbf{P}_k$ is the projection matrix computed from the camera's pose that is estimated by the k-th particle.

This PF achieves better results than the Gauss–Newton method and the EKF for pose estimation. We assume that this is a result of the shape of the error function that is minimized by both methods. Although the error function is smooth in large scale, it is bumpy near the minimum due to slightly erroneous image measurements making it difficult to find the proper minimum for an iterative method. However, this needs to be further investigated.

2.7. Measurement update

After the features are tracked and the camera's pose is refined, the 3D positions of the features are updated using the common EKF update equations leading to a more precise reconstruction of the scenery. This step is straightforward and is described in [7,5] in more detail.

3. Obstacle detection and 3D map-building

For obstacle detection, we perform the described monocular scene reconstruction for 200–300 salient features of the scene simultaneously. Before the reconstructed features are used to build a representation of the environment, they have to undergo some post-processing where unreliable estimates are removed. From all features that were tracked in the current frame, we only use those that meet the following two criteria: First of all, the variance of the estimated inverse depth must be smaller than a threshold of 0.005 (i.e. a standard derivation of ≈0.07 m) to ensure the usage of reliable estimates only. The variance is taken from the error covariance matrix $\boldsymbol{\Sigma}_i$ that is estimated by the EKFs. Besides, the estimated distance to the camera must be smaller than 3 m. This condition removes most virtual features that arise where the boundaries of foreground and background objects intersect in the image. These features do not correspond to a single 3D point in the scene and cannot be estimated properly.

The features that pass the above filters are used to build a three-dimensional model of the environment that can be used for subsequent navigational tasks. Depending on the task we use two different models. For obstacle avoidance and local path planing we use a *volumetric model* that is built in realtime. Since our main focus is on these two areas this volumetric model is most important for our purposes. However, additionally a *surface model* can be built offline after all the data has been processed and all 3D points have been reconstructed.

3.1. Volumetric voxel-based model

For building the volumetric model we partition the robot's surrounding three-dimensional volume $V = \{v_i\}$ into disjoint cube-shaped 3D cells (voxels) v_i. Similar to 2D occupancy grid maps each voxel v_i is associated with an occupancy value $p(v_i)$ which specifies the probability of the volume covered by the voxel being occupied by an obstacle. The voxel map is modeled as a Markov Random Field (MRF) of order 0, where the state of each individual voxel can be estimated as an independent random variable.

At the beginning all voxels are initialized with a probability of 0.5 to indicate that nothing is known about the robot's environment. After a new frame I_{t+1} has been processed, the voxel map is updated using the estimated features in a similar way as laser and sonar range scans are inserted into a 2D occupancy grid map. The estimated 3D position $\mathbf{x}_j$ of each feature, its error covariance matrix $\boldsymbol{\Sigma}_j$ and the current camera's projection matrix $\mathbf{P}_t$ are used to form a measurement

$$z_j^{[t+1]} = (\mathbf{x}_j, \boldsymbol{\Sigma}_j, \mathbf{P}_{t+1}).$$

For each measurement $z_j^{[t+1]}$ the new occupancy probability $p(v_i^{[t+1]})$ of each voxel v_i can be updated recursively from its previous value $p(v_i^{[t]})$ using Bayes rule [28] as follows (for better readability we drop the superscripts of $v_i^{[t]}$ and $z_j^{[t+1]}$ on the right side):

$$p(v_i^{[t+1]}|v_i^{[t]}, z_j^{[t+1]}) = 1 - \left[1 + \frac{p(v_i)}{1-p(v_i)}\frac{p(v_i|z_j)}{1-p(v_i|z_j)}\right]^{-1} \tag{11}$$

where $p(v_i|z_j)$ denotes the *inverse sensor model*, which we will describe in the following. Each measurement $z_j = (\mathbf{x}_j, \boldsymbol{\Sigma}_j, \mathbf{P})$ is considered to be a range measurement along the viewing ray $\mathrm{r}(\mathbf{P}, \mathbf{x}_j)$ that is defined by the position of the camera's center taken from the camera's projection matrix $\mathbf{P}$ and the estimated 3D position $\mathbf{x}_j$ of the feature. The features position is estimated during the scene reconstruction in terms of a trivariate normal

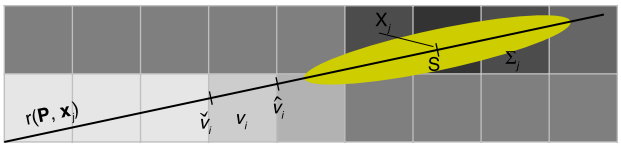


Fig. 5. Each feature x_j is inserted into the voxel map by updating the voxels along the ray r($\mathbf{P}$, $\mathbf{x}_j$) according to an inverse sensor model by taking the error covariance matrix $\mathbf{P}$ of the reconstructed feature into account.

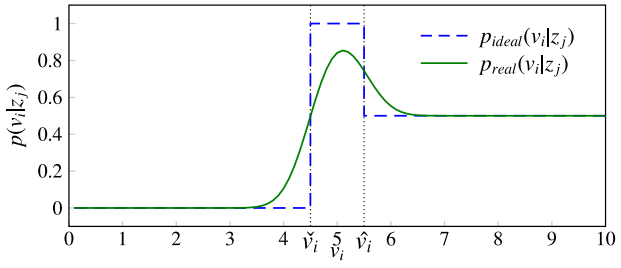


Fig. 6. Occupancy probabilities for an ideal sensor and our Gaussian sensor model.

probability distribution that is defined by the mean value $\mathbf{x}_j$ and the corresponding covariance matrix $\boldsymbol{\Sigma}_j$. As seen in Fig. 5 the position uncertainty of a reconstructed point is larger in the depth direction, i.e. along the ray, while the uncertainty orthogonal to the ray is minor and smaller than the width of a voxel. Therefore, it is a sufficiently good approximation to update only the voxels along the ray when inserting the measurement into the voxel map.

This allows us to apply a one-dimensional sensor model where we take the marginal probability of the trivariate normal distribution along the ray into account. On this ray s denotes the distance of the reconstructed point $\mathbf{x}_j$ to the camera which is located at its origin. Let us first assume that our measurement is free from errors. Then we can update the voxels along the ray according to the ideal sensor model, where the occupancy value for the voxel that contains the estimated point $\mathbf{x}_j$ is set to $p_{\text{occ}} = 1.0$ since it is occupied by the surface of an object where the feature is located on. The occupancy values of voxels that are located on the ray in the line-of-sight between the camera and the estimated 3D point are set to $p_{\text{free}} = 0.0$, since these voxels are free — otherwise the feature had not been visible to the camera. The state of voxels that lie 'behind' the estimated feature position is unknown since they cannot be observed. The characteristic of this ideal sensor model $p_{\text{ideal}}(v_i|z_j)$ is shown in Fig. 6 and can be described formally as follows:

$$p_{\text{ideal}}(v_i|z_j) = \begin{cases} p_{\text{free}} & \breve{v}_i < s \\ p_{\text{occ}} & \breve{v}_i \leq s \leq \hat{v}_i \\ 0.5 & \text{otherwise} \end{cases} \tag{12}$$

where $\breve{v}_i$ and $\hat{v}_i$ denote the starting and end position of the area that is covered by the voxel v_i along the ray $r(\mathbf{P}, \mathbf{x}_j)$, while s denotes the position of the reconstructed point $\mathbf{x}_j$ along the same ray.

However, in practice the positions of the features are error-prone and given as probability distributions as stated before. Therefore, we apply an inverse sensor model that takes the Gaussian error distribution into account. While most researchers use an approximated sensor model, we derived our inverse model analytically. Using Kolmogorov's theorem [28] one can verify that the real sensor model can be obtained by convoluting the above ideal sensor model with a Gaussian $\mathcal{N}(0, \sigma^2)$. Taking the discretization of the voxels into account, the real sensor model can be described by:

$$\begin{aligned} p_{\text{real}}(v_i|z_j) &= p_{\text{ideal}}(v_i|s) * \mathcal{N}(0, \sigma^2) \\ &= \mathrm{F}\left(\breve{v}_i, s, \sigma^2\right) - \frac{1}{2}\mathrm{F}\left(\hat{v}_i, s, \sigma^2\right) \end{aligned} \tag{13}$$

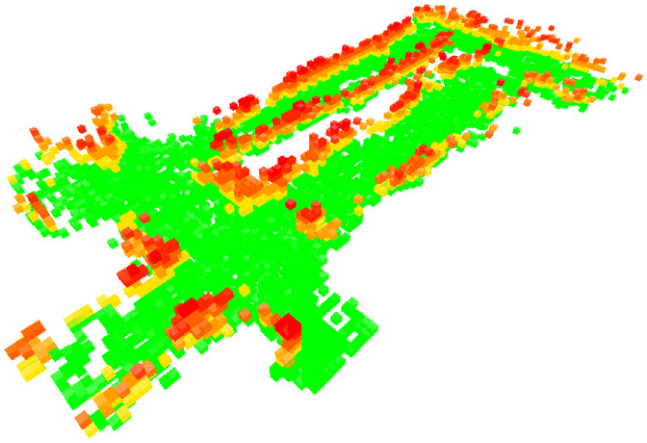


Fig. 7. A 3D voxel map that was created while driving through a test environment. Each place was only visited once. The colors of the voxels indicate their heights (green: <0.10 m, yellow-red: 0.10 m–1.15 m). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $\mathrm{F}\left(x, \mu, \sigma^2\right) = \int_{-\infty}^{x} \mathcal{N}(\mu, \sigma^2)$ is the cumulative normal distribution. The variance σ^2 is taken from the error covariance matrix $\boldsymbol{\Sigma}_i$ and expresses the position uncertainty of the reconstructed point in depth direction.

The resulting voxel representation (see Fig. 7) that is built using the above update rules can finally be used for path planning and obstacle avoidance. However, most of the existing navigation algorithms still operate on 2D occupancy grid maps. Therefore, the 3D voxel representation can be transformed into such a 2D occupancy grid map by choosing the maximal occupancy value of all voxels located in the column above each grid cell.

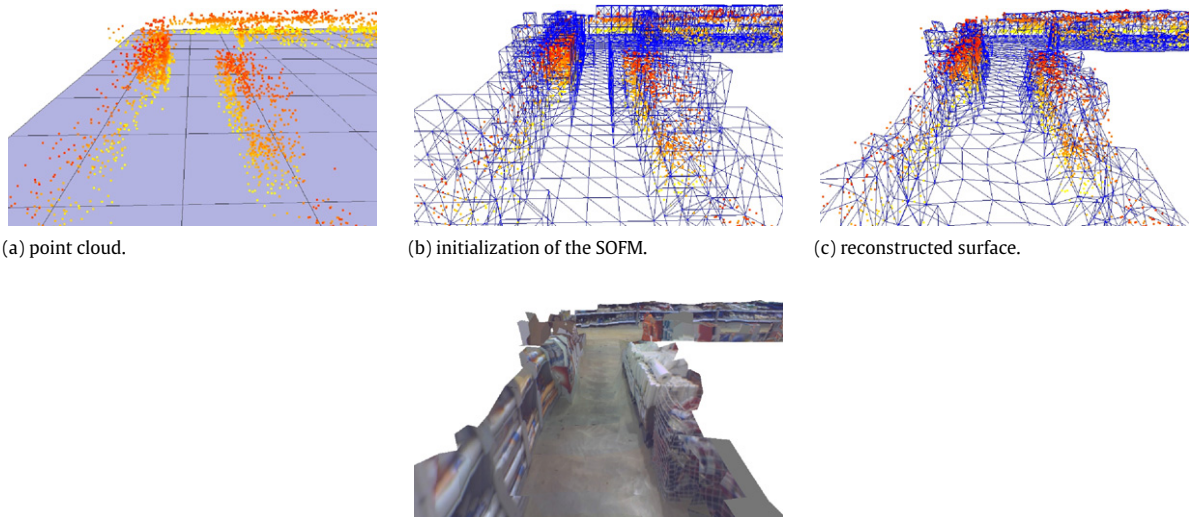
3.2. Textured surface model

Beside obstacle detection we also use our approach for scene reconstruction to build 3D maps consisting of boundary surfaces with associated textures. Such 3D maps can e.g. be used for visual robot localization using appearance based techniques. Using the textured map virtual views of the environment can be computed. By comparing these virtual views with its actual camera image the robot can estimate its location e.g. using particle filters.

For building a 3D surface model we use an approach similar to the algorithm presented in [29], where a modification of Kohonen's two dimensional self-organizing feature map (SOFM) is used to reconstruct the 3D surfaces of a noisy point cloud. The reconstructed 3D points are assumed to be uniformly distributed over the objects' surfaces and are used as training input for the SOFM. If each 3D point of the training input is chosen randomly, the SOFM will unfold along the surface of the scene.

In contrast to [29], we use a different method for initializing the SOFM to ensure a fast and correct convergence. We use the voxel representation that was described in the previous section to obtain the initial position of the SOFM's nodes and its structure. Each voxel is marked if it contains at least one reconstructed 3D point. Afterwards, the eight corners of each marked voxel are added as neurons to the SOFM. Additionally, the four neurons that belong to the same face of a marked voxel are connected by edges if the neighboring voxel, that is adjacent to that face, is not marked. This simple method produces a coarse surface that covers all reconstructed points as shown in Fig. 8(b).

After the SOFM is trained as described in [29], the triangulation of the reconstructed surface can be obtained immediately by means of the SOFM's topology. Finally, a texture is mapped to each triangle. For each triangle its texture is obtained from that image of the input sequence where the area of the back-projected



(a) point cloud. (b) initialization of the SOFM. (c) reconstructed surface.

(d) textured scene.

Fig. 8. Different steps for creating a textured 3D map. Based on the reconstructed point cloud (a) a SOFM is initialized (b). In (c) the reconstructed surface is shown after training the SOFM for 5000 epochs. The lower right image shows the final texture-mapped scene (d).

triangle onto the image plane is maximized. An example of the final texture-mapped scene is shown in Fig. 8(d).

4. Attention-driven feature selection

As stated before, most approaches select their features uniformly in the whole image using feature detectors. These detectors choose the strongest features in the input images, i.e. pixels that yield the strongest response of a certain interest operator. Therefore, the positions of the selected features depend on the image content only. In this section we use a more biologically inspired attention-driven approach that chooses new features in image areas that are most relevant for obstacle detection and for preventing a collision. This procedure is similar to humans and animals who usually turn their gaze to areas where the obstacle situations are unclear in order to use monocular and stereo cues that help them to ascertain if they can move on safely. Especially close obstacles are watched carefully to observe their precise position and to avoid a potential collision.

In order to achieve the desired behavior, our approach first computes an attention map that has the same dimensions as each image of the captured sequence. For all pixels $x' \in I_0$ of the input image I_0 the attention map $A = \{a(x')|x' \in I_0\}$ contains values $a(x') \in \mathbb{R}$ which indicate the importance of selecting new features in a certain region of the input image.

The actual feature selection is performed by a standard feature detector like the Shi–Tomasi detector in a region of interest $R \subset I_0$ which is a subset of the input image. Since most standard feature detectors rely on the rectangular shape of the input images we use a rectangular region of interest R with a fixed size. In our experiments the size of R is set 3–4 times smaller than the complete image. Applying the feature detector in the region of interest makes a big difference compared to detecting features in the whole image, as the detector will select the strongest features in a local image region only. These features usually do not belong to the globally strongest features and would not have been selected if the feature detector were applied on the whole image. Hence, by choosing the position of the region of interest R we can control the location of newly selected features. The position of R is chosen in a way to maximize the sum of the attention values that are covered by the region:

$$R = \underset{R' \subset I_0}{\operatorname{argmax}} \sum_{x' \in R'} a(x'). \tag{14}$$

Adding new features in this region therefore maximizes the gain for the whole approach.

For computing the attention map, different measures and objectives o_i can be taken into account. For computing the final attention values $a(x')$ the weighted sum of the attention values $o_i(x')$ of all objectives is used:

$$a(x') = \sum_i \left(w_i o_i\left(x'\right)\right). \tag{15}$$

Currently, we use two different objectives — an obstacle uncertainty objective and an inhibitory objective. In order to allow the objectives to compute their attention value based on the current scene reconstruction and using information from the navigator about the planned path we implemented a feedback-loop as seen in Fig. 2. Hence, the feedback-loop provides top-down information that is used to guide the feature detector.

4.1. Obstacle uncertainty objective

The most important objective is the *obstacle uncertainty objective* which is used to focus the feature selection to areas where the obstacle situation is unclear and where more observations are necessary. As a measure for the uncertainty we use the entropy of the voxel map that was described in the previous section. The entropy is known from information theory and defined as: $H(X) = -\sum p(x) \log_2 p(x)$. The entropy $H(v_i)$ of a single voxel v_i is given as the binary entropy function:

$$H(v_i) = -p(v_i) \log p(v_i) - [1 - p(v_i)] \log[1 - p(v_i)]. \tag{16}$$

It is maximal when the voxel is initialized with 0.5 and nothing is known about that part of the environment. With additional observations using the reconstructed features the entropy decreases and will finally converge near zero after the voxel has been explored and is classified as either free or occupied. Obviously, each measurement z_j decreases the expected entropy $H(v_i)$ of the voxel and leads to an expected information gain that can be expressed by the mutual information:

$$I(v_i; z_j) = H(v_i) - H(v_i|z_j) \tag{17}$$

where $H(v_i|z_j)$ is the entropy of the voxel v_i *after* inserting the measurement z_j according to Eq. (11) in Section 3.

If a pixel x'_j is selected as a new feature in the input image, the resulting measurement z_j will affect the occupancy probability and

hence the entropy of each voxel v_i along the ray $r(P, x'_j)$ in different ways, depending on whether the reconstructed point $\mathbf{x}_j$ is located inside, behind or in front of the voxel v_i. Unfortunately, the location $\mathbf{x}_j$ of the point is still unknown when the corresponding pixel x'_j is selected as a feature. However, using the occupancy probabilities we can compute the probability for the point $\mathbf{x}_j$ to be located in a certain voxel along the ray $r(P, x'_j)$. If e.g. the occupancy value of a voxel v_n on the ray is near 1.0 while the values of the previous voxels $v_0, \ldots, v_{n-1}$ on the ray are near 0.0, the point will most likely be located in v_n. In general, the probability for the point $\mathbf{x}_j$ to be located in v_n is:

$$p(\mathbf{x}_j \in v_n) = p(v_n) \prod_{i=0}^{n-1} 1 - p(v_i) \tag{18}$$

while the probability for the point to be located in some voxel behind v_n is:

$$p(\mathbf{x}_j \succ v_n) = \prod_{i=0}^{n} 1 - p(v_i). \tag{19}$$

In the first case the voxel is assumed to be occupied, its occupancy probability is increased and its entropy changes to $H(v_i|occ)$. In the latter case the voxel is assumed to be free, the occupancy probability is decreased and the entropy changes to $H(v_i|free)$. Taking these considerations into account we can predict the expected information gain $I(v_i; x'_j)$ for each voxel v_i along the ray $r(P, x'_j)$ after selecting the feature x'_j:

$$\begin{aligned} I(v_i; x'_j) &= H(v_i) - p(\mathbf{x}_j \in v_n) H(v_i|\text{occ}) \\ &\quad - p(\mathbf{x}_j \succ v_n) H(v_i|\text{free}). \end{aligned} \tag{20}$$

For computing the entropies $H(v_i|\text{occ})$ and $H(v_i|\text{free})$ we simulate updating the voxel as occupied and free using Eq. (11). In order to simplify the computation we use the ideal sensor model here which is sufficient for this purpose. To approximate the real sensor model coarsely the models parameters are chosen to $p_{\text{occ}} = 0.8$ and $p_{\text{free}} = 0.2$.

In Fig. 9 the information gain for a single voxel is plotted against the occupancy probability of the voxel according to Eq. (20) using the ideal sensor model with different values for p_{occ} and p_{free}. In all graphs the information gain drops at both ends. Hence, updating voxels that are already identified as free or occupied with a high certainty does not lead to a significant information gain. Surprisingly, the function of the information gain may have a local minimum for occupancy values near 0.5 depending on the chosen parameters p_{occ} and p_{free}. This is a result of the characteristics of the the binary entropy function with its steep slope for probabilities near 0 and 1. Small changes in the probability lead to large information gains for these values. The occupancy update function counteracts this tendency. Here, the change in the probability for cells with a occupancy probability near 0.5 is dominant compared to those with probabilities near 0 or 1. For diffuse sensor models the binary entropy function will dominate the characteristics of the information gain and updating voxels whose occupancy state is completely unknown is expected to yield a smaller information gain compared to voxels where at least some information is already available and where additional observations can ascertain their real states. However, when using values $p_{\text{occ}} \geq 0.8$ and $p_{\text{free}} \leq 0.2$ for the ideal sensor model this effect disappears.

As stated before, Eq. (20) yields the information gain of a single voxel along the ray after selecting a new feature x'_j. In order to obtain the total information gain for selecting the feature x'_j the gains of all voxels along the ray have to be accumulated. This can be implemented efficiently using ray casting. Putting all together

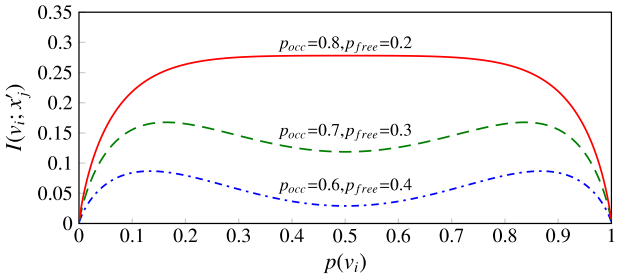


Fig. 9. Information gain $I(v_i; x'_j)$ of a single voxel plotted against the occupancy probability of the voxel using different values p_{occ} and p_{free} of the ideal sensor model.

we get the final attention function for the *obstacle uncertainty objective*:

$$o_1(x') = \sum_i u(v_i) I(v_i; x'). \tag{21}$$

In this equation we added an additional weight function $u(v_i)$ that can be used to control the importance of a each voxel. For obstacle avoidance, for example, the occupancy states of voxels along and near the path that was planned by the navigator are more important than voxels that are far away. Additionally, we use higher weights for voxels near the robot than for distant voxels.

4.2. Inhibition of Return — Objective

The second objective we apply is an inhibitory objective. It is required to avoid the attention getting stuck at the same image region while other parts of the image are never chosen for selecting new features. *Inhibition of Return — Objective* manages an activity map $M = \{m(x')|x' \in I_0\}$ that has the same size as the attention map and the input image. This activity map keeps track of the image regions $R_{t-1}, R_{t-2}, \ldots$ previously selected for feature selection according to Eq. (14). Therefore, each element $m(x')$ of the current activity map $M = M_t$ is updated as follows:

$$m(x') = \eta m_{t-1}(x') + \beta\delta\left(x'\right), \quad \text{where } \delta\left(x'\right) = \begin{cases} 1 & x' \in R_{t-1} \\ 0 & \text{otherwise.} \end{cases}$$

Here, η denotes a decay rate that decreases the previous activation $m_{t-1}(x')$. It is chosen as $\eta = 0.95$ in our experiments. The parameter β denotes some activation parameter that adds activation to all elements in the activity map that correspond to the image region R_{t-1} chosen for feature selection in the previous iteration. Reasonable values for this parameter are $\beta = 0.1, \ldots, 0.2$. Finally, the attention value of this objective can be easily defined as:

$$o_2(x') = -m(x'), \tag{22}$$

where the activation that accumulates the positions of the previously selected regions has an inhibitory influence on the overall attention $a(x')$ in Eq. (15).

Using the Inhibition of Return — Objective together with the Obstacle Uncertainty — Objective results in a movement of the region of interest used for feature detection similar to the saccade-like movement of the eyes of vertebrates, allowing one to cover the whole field of view while concentrating on the most interesting parts of the environment.

Although the above objectives were designed to handle our obstacle detection problem, the approach is very flexible and can be modified depending on the purposes the scene reconstruction is used for by applying different objectives. If a dense reconstruction for building precise 3D models is desired, one could use an objective that ensures a dense and uniform coverage of the reconstructed points over the whole 3D scene. If the framework is used for visual odometry on the other hand, an objective that leads to a uniform feature distribution in the images is more suitable.

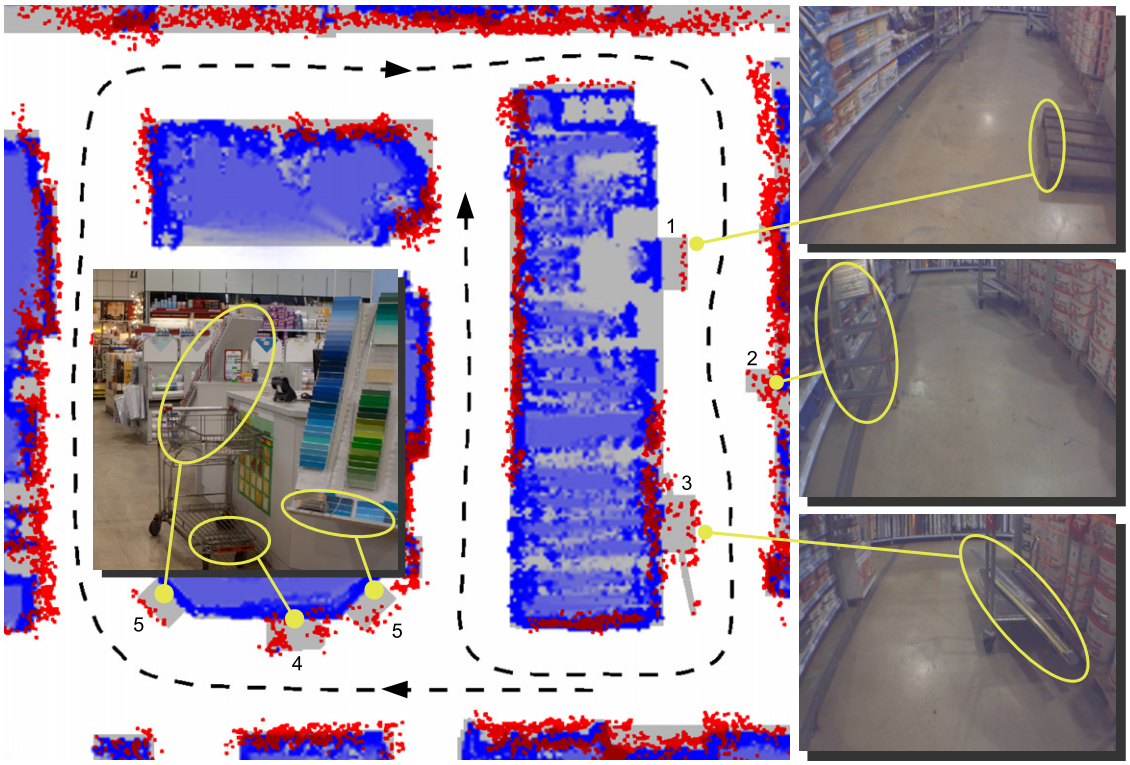


Fig. 10. Comparison of maps created from visual information (red dots) and laser range finder (blue). The robot's trajectory and moving direction is denoted by the dashed line. The ground truth is highlighted in gray. The visual map consists of about 8200 reconstructed points. Obstacles detected using vision only are labeled using numbers. The images on the right show the obstacles as seen by the front camera. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Results

In our experiments we used a 1/4″ CCD fire-wire camera for image acquisition. The camera is mounted in front of our robot at a height of 1.15 m and tilted by 35° towards the ground (see Fig. 1). For processing, our robot is equipped with an Intel Core 2 Duo, 2 GHz CPU. In spite of utilizing one core only we are able to process up to 30 frames per second while reconstructing 200–300 features simultaneously. Depending on the robot's driving speed (up to 1 m/s), we only need to process 10–15 frames per second leaving enough CPU resources for other applications like map building, navigational tasks, user tracking and human–machine interaction.

The following results were derived from four extensive tests, where two were carried out inside of a home improvement store, one took place outside in a garden center of a home improvement store and one test was conducted in a home environment.

For visualization purposes only, we additionally used a laser based SLAM approach for correcting the odometry before creating the maps printed on the following pages. In our final application the robot's corrected odometry is sufficient for mapping since we are only interested in the local environment in relation to the robot for obstacle detection, where slight position errors and a drift in the orientation can be neglected.

5.1. Obstacle detection

Fig. 10 shows a map that was created by merging the reconstructed obstacles that were detected by our approach with information provided by a laser range finder. The occupancy map that was created using the laser range finder is colored in blue where the different shades of blue correspond to the probability that a cell is occupied. The position of the features that were reconstructed using visual information with the approach presented in this paper are colored in red. In the map, a total number of about 8200 visual features is shown. While creating the map, a total number of ca. 15,400 points was reconstructed, where about 6000 features were filtered due to a bad variance, ca. 1000 features were classified as belonging to the ground and 100 were detected as outliers.

For better evaluation and for visualization purposes a ground truth map was created and is highlighted in gray in the background of Fig. 10. For building the ground truth, we took images of the scene using a hand held Canon EOS 350D 8.0 megapixel camera and used a bundle adjustment tool[2] for creating a precise reconstruction of the scene which finally was edited and labeled manually.

The map covers an area of 14 m × 12 m within a home improvement store. This test area contains typical obstacles that we identified as problematic during the field test since they cannot be detected by the laser range finder due to their reflection properties, their form or too low height. Some of these obstacles are numbered from 1 to 5 in Fig. 10. In detail these obstacles are:

1. an empty Euro-pallet with a height of 11 cm
2. a ladder
3. a low shopping cart with goods that jut out at both ends
4. a high shopping cart
5. shelves that extend into the scene.

All of these obstacles cannot be seen by the laser range finder and, therefore, might result in collisions. However, using our visual approach these obstacles can be detected robustly. In Fig. 11 we try to quantify this result. For each obstacle, we have manually labeled those parts of the outline that are relevant for navigation

[2] Bundler: http://phototour.cs.washington.edu/bundler/.

obstacle	visual	laser	visual+laser
1	63%	-	63%
2	71%	-	71%
3	71%	-	71%
4	68%	10%	68%
5	82%	-	82%
others	85%	78%	96%
total	**83%**	**72%**	**93%**

Fig. 11. Percentage of obstacle boundaries that can be detected using the presented visual approach, a laser ranger finder and a combination of both for the 5 labeled obstacles and the rest of the scene shown in Fig. 10.

and obstacle avoidance during the above test run using the ground truth map. The statistics in Fig. 11 show the percentage of the relevant obstacle boundaries that were detected by our visual approach, the laser range finder and a combination of vision and laser. Although some obstacles were not detected by our visual approach since these parts were not visible to the camera when the robot was driving around corners, these results show that major parts of the above-mentioned obstacles can be detected. Furthermore, it can be seen that the detection rate for all relevant objects in the scene can be increased significantly by 20% compared to obstacle detection using a laser range finder only.

The home improvement store where we carried out the above tests also contains a garden center where a bumpy stone floor leads to strong vibrations. However, the increased shaking of the camera did not result in degradation of the reconstruction (Fig. 12).

5.2. 3D mapping

In Fig. 13 a 3D map of the test environment shown in Fig. 10 is depicted. In the upper left image the reconstructed features are shown as colored dots, where the color indicates the estimated height of each feature. The reconstructed features were used to generate a surface model (Fig. 13(b)) as described in Section 3.2. Using the textured model different synthetic views of the scene can be rendered (Fig. 13(c–d)).

Additional tests were carried out in a special test area of our lab that contains typical elements of a living room: two arm chairs, a table and two shelves as well as a floor with a strong repetitive texture. Furthermore, the lighting conditions are inferior compared to the home improvement store. The top left image in Fig. 14 shows the scene as seen by the front camera of the robot. In the upper right image the reconstructed features are shown. All obstacles that were seen by the camera are detected robustly, while features on the floor are estimated correctly and classified as free and passable. The two images at the bottom of Fig. 14 again show two synthetic views of the environment.

5.3. Attention-driven feature selection

Fig. 15(a) shows an image of the scene taken in a home improvement store as seen by the front camera. In Fig. 15(b) the expected information gain is shown for each pixel of the image. High values are drawn using red colors and indicate high benefits for selecting new features in these regions. As seen in Fig. 15(b) the highest information gain can be achieved by selecting new features in the upper part of the image especially near the obstacle shown in Fig. 15(a). According to Eq. (14) our approach selects the features in this region as indicated by the blue rectangle in Fig. 15(a). After the robot has approached the obstacle, our algorithm for monocular scene reconstruction has estimated the 3D positions of the selected features as seen in Fig. 15(c). The reconstructed points are again shown as dots, where the color indicates their estimated height. Since the obstacle has now been discovered adding new features in this area results in minor information gain as denoted by the blue and yellow colors in Fig. 15(d). Instead, new features will now be selected in the image regions around the detected obstacle in order to discover these unknown parts of the environment.

Similar desired behaviors of our approach are shown in Fig. 16 where we tried to visualize the saccade-like movement of the region of interest. The regions that were used for feature detection during the last 10 frames are shown as transparent yellow rectangles. Image areas that were used more often are more opaque than areas where less attention was paid to. Fig. 16(a) was taken while the robot was turning around a right-hand bend. Here, our approach guides the feature selection to the upper right image areas that newly became visible to the camera in order to discover those parts of the environment not observed before. This is important for local path planning since the robot must react quickly to obstacles that suddenly appear behind corners. Fig. 16(b) shows an image where the robot is moving along a corridor. Here, most features are selected on distant objects in the upper parts of the images. This is reasonable since these features remain visible over more frames of the captured image sequence compared to foreground features that move out of the field of view very quickly due to the robot's forward motion. Additionally, the foreground objects have already been discovered by previous measurements.

In Fig. 17 two occupancy maps are shown that were created from a voxel map as shown in Fig. 7. While creating these maps we reduced the number of features that are tracked per frame to 50–100 in order to show the advantages of our proposed guided feature selection. The map in Fig. 17(a) was created by selecting the features according to the attention-driven approach presented in this paper while Fig. 17(b) shows a map that was creating using features that were detected uniformly in each image.

Although the same number of features was used for creating both maps, the map in Fig. 17(b) contains several cells whose occupancy probability is unknown though they have been visible to the camera. Additionally, some cells were erroneously estimated as occupied. However, the map 17(a) that was created using our guided feature selection approach contains significantly fewer errors and fewer cells whose obstacle situation is unclear. These results show that the guided feature selection can improve the map-building and the detection of obstacles by reducing the uncertainty in the created map and therefore by decreasing the risk of missing an obstacle.

Further results and videos of our approach and the presented algorithms can be found on the following website: http://www.youtube.com/user/neurobTV.

6. Conclusion and future work

In this paper, we have presented an algorithm for feature-based monocular scene reconstruction and shape from motion. We have described several improvements to [5] that make the reconstruction more reliable and help to reduce outliers. Furthermore, we have described methods for creating textured surface maps as well as volumetric voxel maps using the reconstructed 3D points. The volumetric voxel maps are used for obstacle avoidance. Additionally, they provide top-down information for an attention-driven feature selection scheme which we have proposed in this paper. Our approach selects new features in those image regions that maximize the information gain. As a result, the created maps contain fewer uncertainties and more obstacles can be detected without increasing the number of reconstructed features and therefore without increasing the runtime of the whole algorithm.

With realistic field tests in different environments, we have shown that the described techniques allow the approach to be used for robust real-time obstacle detection and 3D map building under

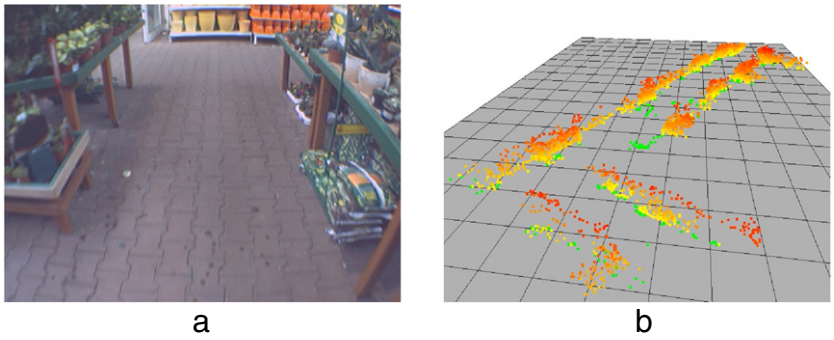


Fig. 12. (a) Front camera image. (b) The reconstructed scene where the height of the reconstructed features is coded by the color (green: <0.10 m, yellow-red: 0.10 m–1.15 m). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

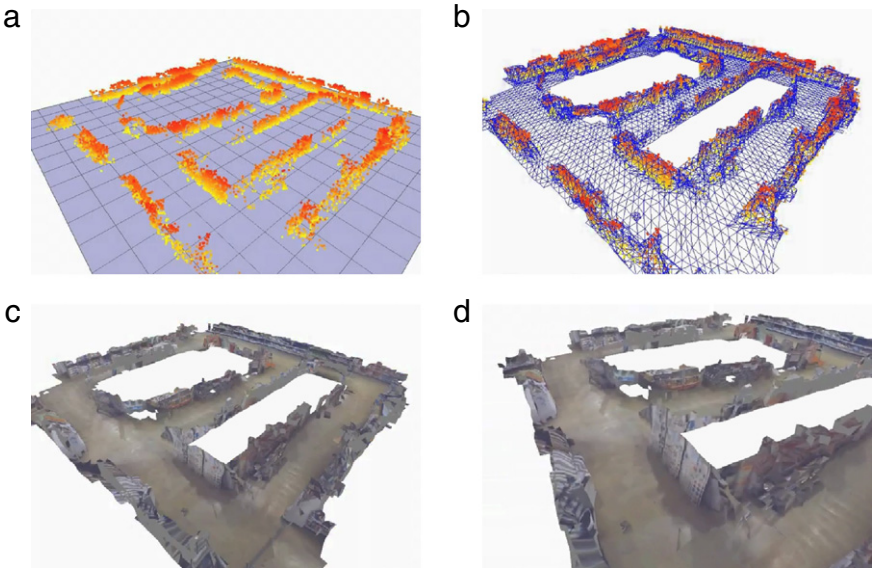


Fig. 13. (a) Reconstructed features that were classified as obstacles in the test environment shown in Fig. 10. (b) Mesh that was created using these features. (c–d) Synthetic views of the scene created using a textured 3D model estimated by our approach.

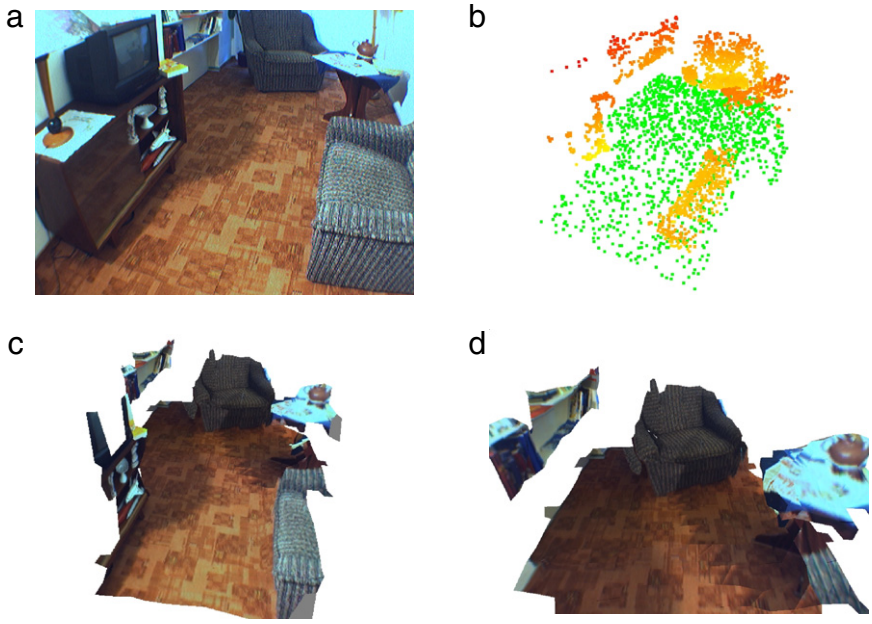


Fig. 14. (a) Image of home environment as seen by the front camera. (b) The reconstructed features shown as dots, where the height is coded by different colors (c–d) Synthetic views of the scene created using a textured 3D model estimated by our approach. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

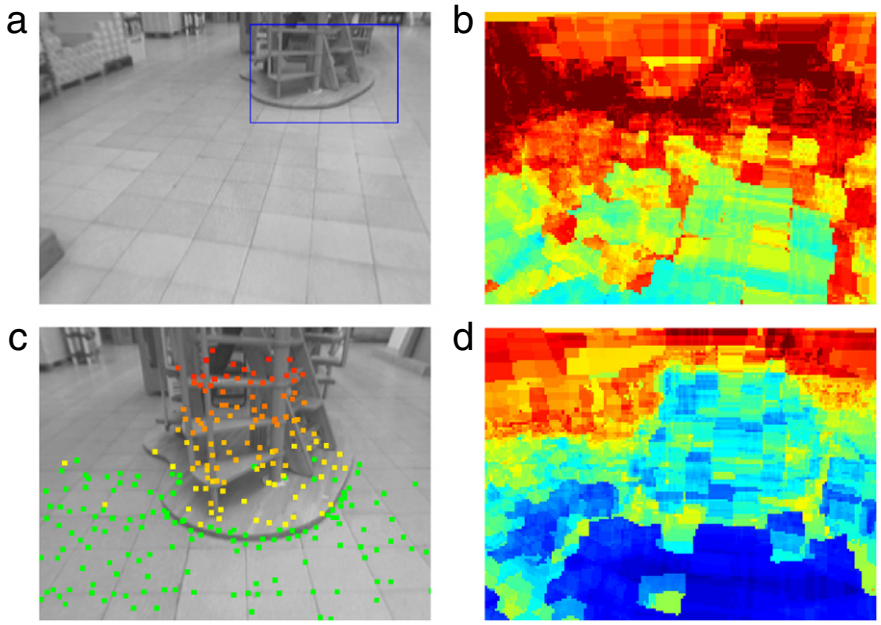


Fig. 15. (a) Input image as seen by the robot's front camera. The region that is used for feature selection is marked by the blue rectangle. (b) The information gain for each pixel of the upper left image, where red color indicates high values and blue corresponds to low values. (c) Input image taken some frames later. The reconstructed features are shown as dots, where the height is coded by different colors (green: <0.10 m, yellow-red: 0.10 m–1.15 m). (d) Information gain for each pixel of the lower left image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

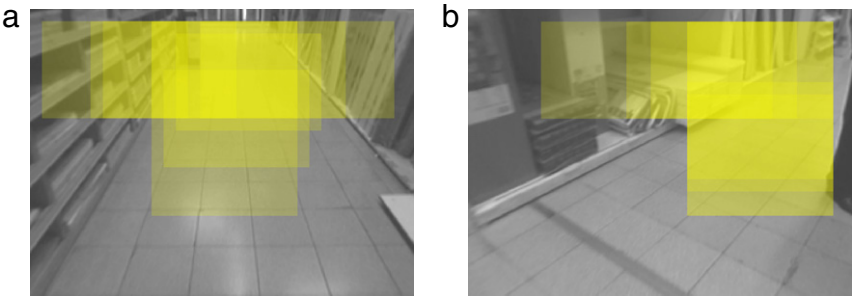


Fig. 16. Image regions that were used for feature selection during the last 10 frames are shown as transparent rectangles. Areas where more attention was paid to are more opaque. Images were taken while (a) driving around a right-hand bend and (b) driving along a narrow corridor.

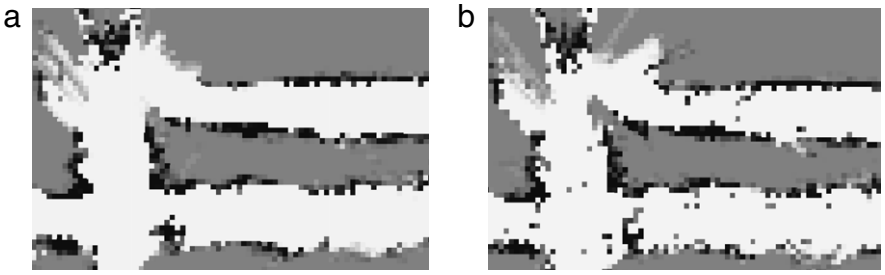


Fig. 17. Occupancy maps that were created from voxel maps. (a) Map created using the proposed attention-driven feature selection approach. (b) Map created by selecting the features uniformly in each image.

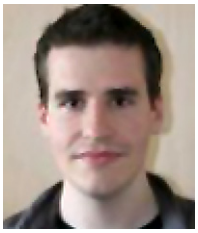
different conditions. We were able to show, that some obstacles which are not visible to active distance measuring sensors, like laser range finders, can be safely detected by our vision based approach and that a combination of visual obstacle detection with a laser range finder can increase the detection rate of obstacles considerably. Currently, we are carrying out long-term tests to evaluate how much the number of collisions can be decreased during the daily usage of the robots.

Moreover, we are developing a method to estimate the position of moving objects. Using additional constraints the position of moving objects that reach to the ground can be recovered. We hope to publish the first results of that algorithm soon. In the approach presented in this paper, features along moving objects are rejected during feature tracking and filtered after the reconstruction due to their high variance in the position estimates.

References

[1] H.-M. Gross, H.-J. Böhme, C. Schröter, S. Müller, A. König, C. Martin, M. Merten, A. Bley, ShopBot: Progress in developing an interactive mobile shopping assistant for everyday use, in: Proc. of IEEE Int. Conf. on SMC, 2008, pp. 3471–3478.

[2] T. Schamm, S. Vacek, J. Schröder, J. Zöllner, R. Dillmann, Obstacle detection with a Photonic Mixing Device-camera in autonomous vehicles, International Journal of Intelligent Systems Technologies and Applications 5 (2008) 315–324.
[3] P. Foggia, J. Jolion, A. Limongiello, M. Vento, Stereo Vision for Obstacle Detection: A Grap-Based Approach, in: LNCS Graph-Based Representations in Pattern Recognition, vol. 4538, 2007, pp. 37–48.
[4] A. Wedel, U. Franke, J. Klappstein, T. Brox, D. Cremers, Realtime depth estimation and obstacle detection from monocular video, DAGM Symposium on Pattern recognition 4174 (2006) 475–484.
[5] E. Einhorn, C. Schröter, H.-J. Böhme, H.-M. Gross, A hybrid kalman filter based algorithm for real-time visual obstacle detection, in: Proc. of the 3rd ECMR, Freiburg, Germany, 2007, pp. 156–161.
[6] A. Davison, I. Reid, N. Molton, O. Stasse, MonoSLAM: Real-time single camera SLAM, IEEE Trans. on PAMI 29 (6) (2007) 1052–1067.
[7] J. Civera, A. Davison, J. Montiel, Inverse depth parametrization for monocular SLAM, IEEE Transactions on Robotics (2008) 932–945.
[8] E. Eade, T. Drummond, Monocular SLAM as a graph of coalesced observations, in: IEEE Int. Conf. on Computer Vision, 2007, pp. 1–8.
[9] E. Eade, T. Drummond, Unified loop closing and recovery for real time monocular SLAM, in: Proc. of the BMVC, 2008.
[10] L. Matthies, T. Kanade, R. Szeliski, Kalman filter-based algorithms for estimating depth from image sequences, International Journal of Computer Vision 3 (1989) 209–238.
[11] M. Sagrebin, A. Noglik, J. Pauli, Robust time-to-contact calculation for real time applications, in: Proc. of 18th International Conference on Computer Graphics and Vision, 2008, pp. 128–133.
[12] M. Sagrebin, J. Pauli, Improved time-to-contact estimation by using information from image sequences, in: Proc. Autonomous Mobile Systems (AMS), 2009.
[13] A. Davison, Active search for real-time vision, in: ICCV, 2005.
[14] M. Chli, A. Davison, Active matching, in: Proc. ECCV, 2008.
[15] T. Vidal-Calleja, A. Davison, J. Andrade-Cetto, D. Murray, Active control for single camera SLAM, in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA, 2006, pp. 1930–1936.
[16] S. Frintrop, P. Jensfelt, Attentional landmarks and active gaze control for visual SLAM, in: Visual SLAM, IEEE Transactions on Robotics 24 (2008) (special issue).
[17] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed., Cambridge University Press, ISBN: 0-521-54051-8, 2006.
[18] J. Shi, C. Tomasi, Good features to track, in: 9th IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 593–600.
[19] E. Rosten, T.T. Drummond, Machine learning for high-speed corner detection, in: Proc. of the ECCV, Vol. 1, 2006, pp. 430–443.
[20] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[21] Y. Yu, K. Wong, M. Chang, A fast recursive 3D model reconstruction algorithm for multimedia applications, in: Proc. of the Int. Conf. on Pattern Recognition, ICPR, Vol. 2, 2004, pp. 241–244.
[22] M. Parsley, S. Julier, Avoiding negative depth in inverse depth bearing-only SLAM, in: Int. Conf. on Intelligent Robots and Systems, IROS, 2008.
[23] A. Davison, Y. Cid, N. Kita, Real-time 3d SLAM with WIDE-angle vision, in: Proc. IFAC Euron Symposium on International Autonomous Vehicle, 2004.
[24] R. Bunschoten, B. Kröse, Robust scene reconstruction from an omnidirectional vision system, IEEE Transactions on Robotics and Automation (2) (2003) 351–357.
[25] M. Okutomi, T. Kanade, A multiple-baseline stereo, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (4) (1993) 353–363.
[26] H. Hirschmüller, P. Innocent, J. Garibaldi, Real-time correlation-based stereo vision with reduced border errors, International Journal of Computer Vision 47 (2002) 229–246.
[27] W. van der Mark, D. Gavrila, Real-time dense stereo for intelligent vehicles, in: IEEE Trans. on Intelligent Transportation Systems, 2006.
[28] A. Elfes, Using occupancy grids for mobile robot perception and navigation, Computer 22 (6) (1989) 46–57.
[29] A. Baader, G. Hirzinger, A self-organizing algorithm for multisensory surface reconstruction, in: Proc. Intelligent Robots and Systems, IROS, Vol. 1, 1994, pp. 81–88.



E. Einhorn is a Ph.D. student at the Department of Neuroinformatics and Cognitive Robotics at Ilmenau Technical University since 2008. He received his Diploma degree in Computer Science from the Ilmenau Technical University in 2007. His research interests are in visual 3D perception and environment modelling for mobile robot navigation.

Ch. Schroeter is a Postdoc at Ilmenau Technical University, Faculty of Computer Science and Automation, Department of Neuroinformatics and Cognitive Robotics. He received his Diploma degree in Computer Science in 2001 and his Doctorate degree in Robotics in 2009. His main research interests are autonomous robot navigation, and visionbased humanrobot interaction.

H.M. Gross has been a full professor of Neuroinformatics at the Ilmenau Technical University, Faculty of Computer Science and Automation, and has headed the Department of Neuroinformatics (now Neuroinformatics and Cognitive Robotics) since 1993. He received his Diploma degree in Technical and Biomedical Cybernetics in 1985 and his Doctorate degree in Neuroinformatics in 1989. Among his main research interests are neural computing, autonomous robots, reinforcement learning, and visionbased humanrobot interaction. He is a member of INNS and ENNS.