4

Fast 3D Perception for Collision Avoidance and SLAM in Domestic Environments

Dirk Holz, David Droeschel and Sven Behnke
Department of Computer Science VI, University of Bonn
Germany

Stefan May
Institut National de Recherche en Informatique et en Automatique, Sophia-Antipolis
France

Hartmut Surmann
Fraunhofer Institute Intelligent Analysis and Information Systems, Sankt Augustin
Germany

1. Introduction

Autonomous service robots that assist in housekeeping, serve as butlers, guide visitors through exhibitions in museums and trade fairs, or provide care to elderly and disabled people could substantially ease everyday life for many people and present an enormous economic potential (Haegele et al., 2001; Pollack et al., 2002; Siegwart et al., 2003). Moreover, regarding the aging society in most industrialized countries the application of service robots in (elderly) health care might not only be helpful but necessary in the future. However, these service robots face the challenging task of operating in real-world indoor and domestic environments. Domestic environments tend to be cluttered, dynamic and populated by humans and domestic animals. In order to adequately react to sudden dynamic changes and avoid collisions, these robots need to be able to constantly acquire and process, in real-time, information about their environment. Furthermore, in order to act in a goal-directed manner, plan actions and navigate effectively, autonomous mobile robots need an internal representation or *map* of their environment. Nature and complexity of these representations highly depend on the robot's task and workspace.

When operating in preliminary unknown environments, e.g., when it is unfeasible (or simply uncomfortable) to manually model the environment beforehand, the robot needs to construct an internal environment model on its own. Moreover, in dynamic environments the robot further needs to be able to continuously acquire and integrate new sensory information to update the internal environment model in regions where changes have taken place. As integrating new information into the model (*mapping*) requires knowledge about the robot's pose (position and orientation in the environment) and determining the robot's pose requires a map of the environment, these two problems need to be considered jointly and the problem

of constructing or updating an internal environment model is commonly referred to as Simultaneous Localization and Mapping (SLAM). In fact, SLAM is regarded as one of the major prerequisites for truly autonomous robots (Wang, 2004).
Both, collision avoidance and SLAM are well understood in the two-dimensional case, e.g., when acquiring geometric information about surrounding environmental structures with 2D laser range finders. However, as will be shown in the following section, this information is not sufficient in order to adequately navigate in cluttered and dynamic, domestic environments. Presented in this work are methods and means for a fast 3D perception of the robot's surrounding environment as well as for extracting and processing relevant information in the context of robust collision avoidance and SLAM.
Section 3 will provide an overview on methods and means for acquiring three-dimensional information using laser range finders as well as related work. Extracting relevant information from the continuous 3D data stream for the purpose of collision avoidance and SLAM is described in Section 4. Efficient algorithms for collision avoidance based on this information as well as SLAM for constructing two-dimensional and three-dimensional environment models are described in Sections 5 and 6. Extensions for using recent Time-of-Flight cameras are presented in Section 7. Section 8 will contain some concluding remarks and an outlook on future work.

2. 2D-Perception in Domestic Environments

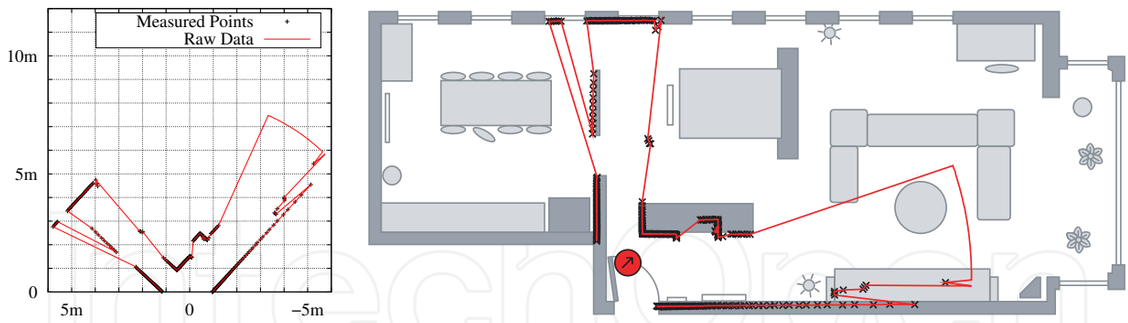
2D laser range-finders became the de facto standard sensor to tackle the problems of SLAM and collision avoidance. These sensors measure, with high frequency and accuracy, the distances to environmental structures surrounding the robot. They emit a laser range beam and measure the time until the emitted beam is received after being reflected on the surface of an object in the robot's vicinity. By means of a rotating mirror these beams are emitted over a two-dimensional plane differing, depending on the used sensor, in apex angle and angular resolution. Typically, 2D laser range finders are mounted horizontally in order to measure distances to surrounding objects in a plane being parallel to the floor. A laser scan S is a set of 2-tuples (d, θ) where d is a distance measurement and θ the angle under which the measurement has been taken, i.e.,

$$S = \{(d_i, \theta_i) \mid i \in [1, N_s]\}. \tag{1}$$

Each tuple (d_i, θ_i) in S forms the polar coordinates of a point $\mathbf{p}_i$ measured on the surface of an object in the surrounding environment, i.e.,

$$\forall i \in [1, N_s] : \quad \mathbf{p}_i = \begin{pmatrix} d_i \cos\theta_i \\ d_i \sin\theta_i \\ 0 \end{pmatrix} = \begin{pmatrix} x_i. \\ y_i \\ z_i \end{pmatrix} \tag{2}$$

using a right-handed coordinate frame. That is, the robot is looking along the x-axis with the y-axis extending to the left. The z-axis points upwards and represents the height of objects. Depending on the measurement principle, S can be a totally ordered set with respect to, respectively, i and θ_i in either clockwise or anti-clockwise direction, i.e., for $i < j : \theta_i < \theta_j$ or $\theta_i > \theta_j$. A 2D laser range scan is exemplarily depicted in Figure 1(a).
The inherent drawback of 2D laser range finders, in the context of simultaneous localization and mapping (SLAM) as well as collision avoidance, is that objects not intersecting the scanner's measurement plane are not perceived. Consider for example the couch table in Figure



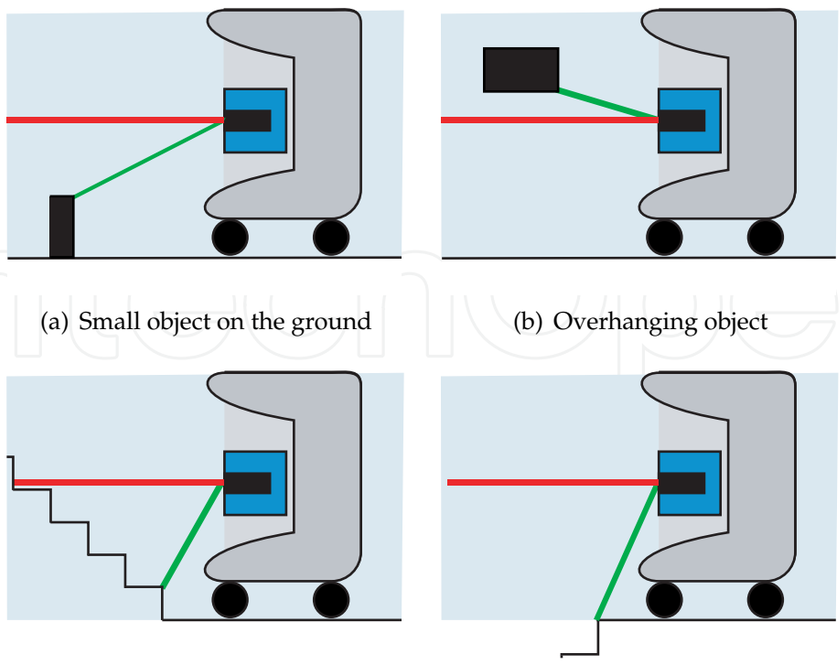
(a) Acquired 2D scan (b) Scan superimposed on a sketch of the environment

Fig. 1. 2D laser range scan in an example scenario. Depicted is a range scan (a) from a data set recorded by Zivkovic et al. (2007) together with an approximate floor plan of the scenario (b). Note, that the couch table has not been sensed at all since it did not intersect the scanner's measurement plane.

1(b). The 2D laser range scan taken in this example scenario does adequately model surrounding environmental structures whereas not a single measurement has been taken on the surface of the couch table. That is, the couch table is not at all perceived. Hence, it cannot be modeled in the robot's internal environment representation. As there is no obstacle in the corresponding model region, the robot might plan a path that directly leads through the couch table. Furthermore, a collision with the couch table cannot be avoided as it does not intersect the scanner's measurement plane. Even when standing directly in front of the table not a single measurement would be reflected. In this example, the scanner is mounted too high so that even the legs of the table do not intersect the measurement plane. However, even when intersecting the scanner's measurement plane, especially table and chair legs are not always perceivable. Depending on material and shape of table legs, e.g., round metal rods, only a portion of emitted laser beams are reflected in a way so that they are received by the scanner. The same holds true for objects whose surface is less reflective. That is, primary reasons for not perceiving an object with a 2D laser ranger finder are:

1. **The object does not intersect the 2D scan plane**. That is, objects below or above the two-dimensional measurement plane cannot be perceived.
2. **The surface of the object is less reflective** or reflects incoming range beams in directions other than the emitting range scanner. Black surfaces, for example, absorb a larger portion of the incoming light. On the other hand, metallic objects with round shape, like for instance table or chair legs, might cause diffuse reflections or completely diffract the emitted beam.

Tables and chairs are not the only objects in domestic environments that are hard to perceive with 2D laser range finders. Objects like for instance table tops, open drawers, small objects lying on the ground or stairs might not be appropriately perceivable by the robot and modeled in its internal environment representation (see Figure 2). When mounting the scanner in another height to perceive a specific class of obstacles, other types of obstacles are still not perceivable. Even the usage of several 2D range scanners in different heights does not appropriately solve this problem. For adequately handling all kinds of obstacles in a cluttered and dynamic environment 3D information becomes crucial.



(a) Small object on the ground (b) Overhanging object

(c) Ascending Stairs (d) Descending Stairs

Fig. 2. Different types of obstacles. Shown are four different examples of obstacles in a robot's workspace. They have in common that the robot is not able to reliably avoid them by means of simple 2D perception. The measurement plane of a standard 2D laser range finder is depicted in red only intersecting the ascending stair. By means of 3D perception, a single distance measurement (green line) would allow the robot for detecting the obstacles.

3. Using Laser Range Finders for 3D Perception

Important criteria for the choice of a particular sensor in the design of mobile robots are size, weight, power consumption and prize. Several approaches for acquiring 3D information have been proposed that differ, amongst others, in the type, number and setup of sensors. Here, we will focus on approaches that are based on range sensors.

3.1 Related Work

Thrun et al. (2000) use two 2D laser range finders. One scanner is mounted horizontally and used for localizing the robot with three degrees-of-freedom. The other scanner is mounted vertically. The data of the latter is used to compute a volumetric model of the scene based on the poses determined by using the horizontal scanner. However, whereas this approach allows for constructing three-dimensional environment models, it only provides 3D information about objects that are currently passed by the robot and intersect the measurement plane of the vertically mounted scanner. That is, objects in front of the robot are only, if at all, perceivable by the horizontally mounted 2D scanner.

Zhao & Shibasaki (2001) follow a similar approach but use multiple vertically mounted scanners in order to reduce the size of these occlusions. In fact, the usage of multiple 2D range scanners became a commonly found sensor setup in the context of the DARPA Grand and Urban Challenges. Thrun et al. (2006) mounted five 2D laser range scanners on the top of Stanley,

the robot that won the DARPA Grand Challenge. The scanners were mounted in driving direction (just like horizontally mounted 2D scanners), but under different pitch/tilt angles to perceive the surface of the ground in different distances. Similar setups can be found in many other participating teams of the Grand and the Urban Challenge. However, such a battery of 2D range scanners cannot be mounted on smaller household service robots.

Another possibility for acquiring three-dimensional information is the application of commercially available 3D laser scanners as used, e.g., in land surveying. Sequeira et al. (1998) use a RIEGL[1] laser scanner on an autonomous robot to construct 3D models of indoor environments. Allen et al. (2001) use a Leica CYRAX[2] laser scanner on a car to construct three-dimensional models or urban environments. A Zoller+Fröhlich[3] scanner is used by Huber et al. (2000) for reconstructing volumetric models of indoor environments. Urmson et al. (2008) use a Velodyne[4] laser scanner in addition to a battery of 2D laser range finders for detecting and avoiding obstacles in the immediate vicinity of an autonomous car in the DARPA Urban Challenge. Commercially available 3D laser scanners directly provide highly accurate three-dimensional point clouds. However, compared to 2D laser range finders, they are quite expensive and unwieldy for the application on mobile household service robots.

Jet another possibility to acquire three-dimensional data is to mount a single 2D laser range finder on a mechanical actuator to gain an additional degree of freedom. That is, in addition to the rotating mirror for scanning two-dimensional planes, the actuator rotates the complete scanner. Taking multiple 2D scans at different rotation angles allows for constructing locally consistent 3D point clouds. Different setups have been proposed that differ primarily in field-of-view (FOV) and spatial measurement density. The highest point density lies around the rotation axis. Amongst others, Surmann et al. (2003) and Hähnel et al. (2002) started using a horizontally mounted scanner where a rotation angle of 0° corresponds to acquiring a regular 2D laser range scan, i.e., with the measurement plane being parallel to the floor. Standard servo motors or pan-tilt units are used to rotate, respectively, the scanner and the measurement plane upwards and downwards in a nodding-like fashion. Wulf & Wagner (2003) evaluate this and other sensor setups and refer it to as a *pitching scanner* due to the rotation about the y-axis in a right-handed coordinate frame. For acquiring a locally consistent 3D laser scan, Surmann et al. stop the robot and rotate the scanner over the complete vertical aperture angle of 120°. Multiple 3D scans are then matched and registered into a global coordinate frame to construct a three-dimensional point model (Surmann et al., 2003). Strand & Dillmann (2008) let the scanner continuously rotate about the x-axis (*rolling scanner*) and use the acquired data for exploring and mapping indoor environments.

3.2 Continuously Rotating Laser Range Finders

The aforementioned approaches have in common that the robot is stopped in order to acquire a locally consistent 3D point cloud by rotating the scanner. The resulting behavior of the robot is thus composed of *stop-scan-move* cycles. Wulf et al. (2006) started using a continuously yawing scanner for acquiring 3D data while moving. They segment the data stream into individual point clouds and use the acquired information to localize the robot based on ceiling structures that are normally not occluded by people or objects. Cole & Newman (2006) use a pitching scanner that is continuously rotated over the complete vertical aperture angle

[1] RIEGL Laser Measurement Systems: `http://www.riegl.com`
[2] Leica Geosystems: `http://www.leica-geosystems.com`
[3] Zoller+Fröhlich: `http://www.zf-laser.com`
[4] Velodyne: `http://www.velodyne.com/lidar`

in a nodding-like fashion. Their robot is, however, not autonomously controlled and the acquired 3D data is solely used to construct three-dimensional environment models. However, augmenting a commercial 2D laser range finder with an additional degree of freedom seems to be the most appropriate approach for acquiring three-dimensional information on smaller mobile robots.

For safe navigation in dynamic and cluttered environments it is crucial to detect, as fast as possible, all obstacles with which the robot could eventually collide. Hence, the area in the robot's movement direction and in the height spanned by the robot's three-dimensional bounding box is of special interest. Since the continuously yawing scanner of Wulf et al. (2006) scans only vertical planes by means of a single 2D range scanner, objects in the robot's movement direction get out of sight when the scanner is sensing environmental structures behind the robot. For the DARPA Urban Challenge we have extended this setup by using two antipodally mounted 2D laser range scanners (Maurelli et al., 2009; Rojo et al., 2007). The two scanners are, furthermore, not mounted vertically but can be adjusted in the rotation angle. The resulting scanner, the Fraunhofer IAIS 3DLS-K is shown in Figure 3.a and consists of two SICK LMS 291 2D laser range finders mounted on a rotatable carrier. This carrier is continuously rotated around the vertical axis. Depending on the current orientation, the 2D laser range scans of the scanners are transformed into a sensor-centric coordinate frame. The transformed scans are aggregated to form a local 3D point cloud (see Figure 3.b). However, even when not waiting for a complete point cloud, but processing every single 2D scan as it arrives, it remains the drawback of having larger areas in the robot's vicinity not in sight until the other scanner arrives at the corresponding rotation angle. Furthermore, larger portions of the acquired information is obtained in regions that do not pose a threat to the robot, e.g., ceiling structures. This is, in fact, the reason why the two scanners are not mounted vertically as for the normal acquisition of 3D laser scans, but almost diagonally. This decreases the size of the unseen region when processing the acquired information scan-wise.

However, for further reducing the sensed regions to those that are relevant for collision avoidance, a pitching 3D laser scanner seems to be more appropriate (Holz et al., 2008; Wulf & Wagner, 2003).

(a) (b) (c)

Fig. 3. Continuously rotating 3D laser scanner (a) and example data (b) taken in front of the Robotics Pavilion at Fraunhofer IAIS. A photo of the scene is shown in (c). The color of the 3D points in (b) corresponds to the received remission values.

3.3 A Continuously Pitching Laser Scanner for Collision Avoidance and SLAM

In a *pitching scanner* setup the 2D laser range finder is mounted horizontally and rotated around the y-axis. A pitching scanner, the IAIS 3DLS, is shown in Figure 4. For the RoboCup@Home world championship in Atlanta 2007, it was mounted on a three-wheeled VolksBot RT3 platform allowing to acquire 3D scans of the arena. With the additional rotation axis, driven by a standard servo motor, the scanner has a vertical aperture angle of up to $\Theta_{\text{pitch}} = 120°$ with a maximum angular resolution of $\Delta\theta_{\text{pitch}} = 0.25°$. Taking a 3D scan by rotating the scanner over the complete vertical range and using a horizontal angular resolution of $\Delta\theta_{\text{yaw}} = 0.25°$ results in 3D point clouds containing 346 080 points. However, as a relatively low angular resolution is sufficient for robust collision avoidance and has benefits in terms of speed concerns while still providing a sufficient detail for mapping purposes, an angular resolution of $\Delta\theta_{\text{yaw}} = 1°$ is preferable. For the used SICK LMS 2xx range scanners, a single 2D laser scan of 181 distance measurements is read in approximately 13.32 ms (≈ 75 Hz) in this operating mode. Reliable navigation in domestic environments requires for a fast and continuous 3D perception of surrounding environmental structures and obstacles. Therefore, the scanner is continuously pitched around its horizontal axis in a nodding-like fashion allowing the robot to perceive surrounding environmental structures in 3D while moving through its workspace. Since a rotation over the complete aperture angle Θ_{pitch} might yield a couple of single 2D laser scans primarily containing information being not relevant for collision avoidance, e.g., only floor points if the scanner is directed downwards or ceiling structures if the scanner is directed upwards, we defined an *area of interest (AOI)* (Holz et al., 2008). This area restricts the range of used rotation angles θ_{pitch} to the interval $[\theta_{\text{pitch, min}}, \theta_{\text{pitch, max}}]$ so that it contains primarily relevant information.

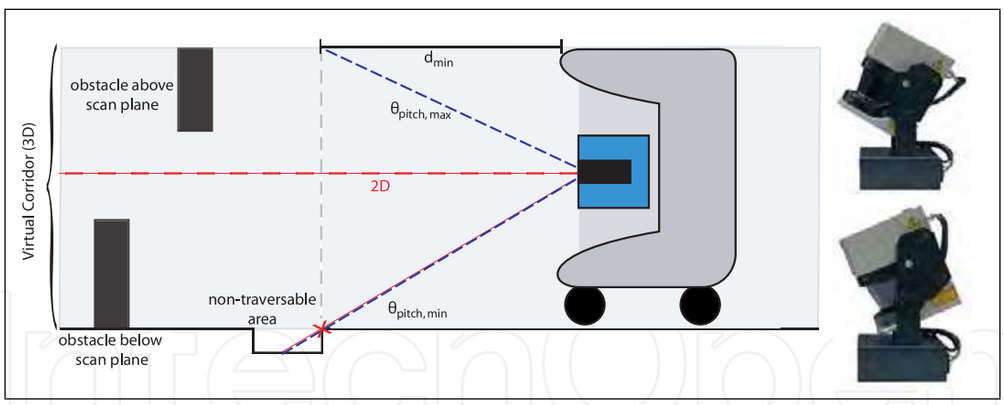


Fig. 4. Continuously pitching scanner and virtual corridor. By continuously rotating the scanner in a nodding-like fashion over the area of interest (AOI), all obstacles in the virtual corridor are perceived.

To be able to react to suddenly appearing obstacles in front of the robot even a restricted but fixed area might be too large to timely perceive especially small objects intersecting only a single measurement plane during one rotation. Therefore, the boundaries of the AOI ($\theta_{\text{pitch, min}}$ and $\theta_{\text{pitch, max}}$) as well as the scanner's pitch rate ($\Delta\theta_{\text{pitch}}$ / 13.32 ms corresponding to the number of consecutive 2D laser scans taken during one pitch movement), can be adjusted, e.g., to depend on the robot's current velocity or special characteristics of the environment. Increasing the upper bound $\theta_{\text{pitch, max}}$ when moving slow allows to perceive more information about the environmental boundaries such as walls due to the height of the measured points.

For moving faster it can be decreased so that only the volume corresponding to the robot's height is sensed. We refer to this minimal area as the *virtual corridor*. It extends the idea of *virtual roadways* (Lingemann et al., 2005a) to the third dimension, i.e., with respect to the robot's boundaries (in 3D) and thus possible areas of collision. The concept of the virtual corridor is depicted in Figure 4. Here lower bound $\theta_{\text{pitch, min}}$ and upper bound $\theta_{\text{pitch, max}}$ of the AOI correspond to the size of the virtual corridor and thus to the robot's boundaries (marked with the slightly colored background). The length $d_{\min}$ corresponds to the distance from which on the full virtual corridor can be perceived during one pitch movement. It has to be chosen appropriately, e.g., for a dense or narrow environment it has to be rather small whereas $d_{\min} = 1\,\text{m}$ is absolutely sufficient when driving fast along an uncluttered corridor. The minimum size of the AOI for driving fast covers exactly the virtual corridor while the maximum size corresponds to a complete 3D scan over the full 120° of Θ_{pitch}. This allows to construct *complete* 3D models of the environment containing all perceivable information as in (Surmann et al., 2003). For the pitching scanner, a scan point is represented by the tuple $(d_i, \theta_{\text{yaw},i}, \theta_{\text{pitch}})$ with d_i being the i-th distance measurement in the latest 2D laser scan while $\theta_{\text{yaw},i}$ and θ_{pitch} are the i-th measurement angle and the current pitch angle of the laser scanner respectively. The coordinates of the 3D points corresponding to acquired distance measurements result from rotating the 2D measurement plane by the current pitch angle θ_{pitch}. Here, the scanner's position on the robot (w.r.t. the robot's center of rotation) is taken into account with the translational part $\mathbf{t}_s = (\mathbf{t}_s^x, \mathbf{t}_s^y, \mathbf{t}_s^z)^T$. Note that the scanner's orientation with respect to the robot's movement direction, has to be taken into account using additional rotations (not necessary here).

$$\mathbf{p}_i = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\theta_{\text{pitch}} & 0 & \sin\theta_{\text{pitch}} \\ 0 & 1 & 0 \\ -\sin\theta_{\text{pitch}} & 0 & \cos\theta_{\text{pitch}} \end{pmatrix}}_{\mathbf{R}_y(\theta_{\text{pitch}})} \underbrace{\begin{pmatrix} d_i \cos\theta_{\text{yaw},i} \\ d_i \sin\theta_{\text{yaw},i} \\ 0 \end{pmatrix}}_{\mathbf{R}_z(\theta_{\text{yaw},i})} + \mathbf{t}_s \tag{3}$$

By continuously sensing and monitoring the virtual corridor the different types of obstacles (see Figure 2) can be perceived. Small obstacles lying on the ground and overhanging objects do not intersect the measurement plane when using simple 2D perception, i.e., when holding the scanner in a fixed horizontal position (red line). With the continuously pitching scanner they are perceived and can thus be avoided. The ascending stair is perceivable with 2D perception but depending on the scanner's height, the measured distance is larger than the distance to the first step. With 3D perception, the first step is perceived just like a small object lying on the ground. For descending stairs, however, a little trick needs to be applied that is presented in the following section.

4. Extracting Relevant Information from 3D Data

Real-time applicability does not only necessitate a fast acquisition of information but also to efficiently process the acquired information. Due to the larger amount of data and the higher dimensionality of information, directly processing raw 3D data is not feasible in many applications. Especially in the context of navigation, existing state-of-the-art approaches that show real-time applicability normally perform on less complex and less information bearing 2D laser data (see e.g. Lingemann et al., 2005a). In order to combine these well-studied and well-performing algorithms with the rich continuously gathered 3D data it is suggestive to break down the three-dimensionality of the data into a slim two-dimensional representation that still holds all necessary 3D information but is nevertheless efficient enough to apply these

efficient algorithms. For this purpose we are using *virtual maps* that have been initially presented in (Holz et al., 2008) and that are based on the idea of virtual 2D scans (Wulf et al., 2004). These egocentric maps, in which, respectively, the robot and the sensor form the origin of the coordinate frame, store only relevant information that has been extracted from 3D data. Two types of virtual 2D maps are distinguished: *2D obstacle maps* and *2D structure maps*. Both are generated from consecutive single 2D laser range scans acquired during the continuous pitching movement of the laser scanner presented above or extracted, for example, from one depth image of a Time-of-Flight camera (see Section 7). Note that the following descriptions will focus on continuously pitching lasers scanners. The actual implementation, however, is the same for all kinds of 3D sensors.

4.1 Structure of Virtual Maps

To be able to apply the same algorithms for collision avoidance, mapping and localization purposes to both standard 2D laser scans and virtual 2D maps constructed by means of 3D perception, the representation of the virtual maps is chosen to extend the representation of standard laser scans. That is, they are organized as a vector of distance measurements d_i ordered by the discretized measurement angle ($\theta_{\text{yaw},i}$). This extended representation has, compared to a 2D laser scanner, a variable aperture angle $\Theta \in [0°, \ldots, 360°]$ and a variable angular resolution $\Delta\theta_{\text{yaw}}$. It is implemented as a vector of $N = \Theta/\Delta\theta_{\text{yaw}}$ points indexed by the accordingly discretized angle in which the measured point is lying from the robot's perspective. Furthermore, each point is represented by means of Cartesian and polar coordinates to avoid algorithm-dependent transformations. An example of a virtual map modeling nearest obstacles in a cluttered environment is visualized in Figure 5.

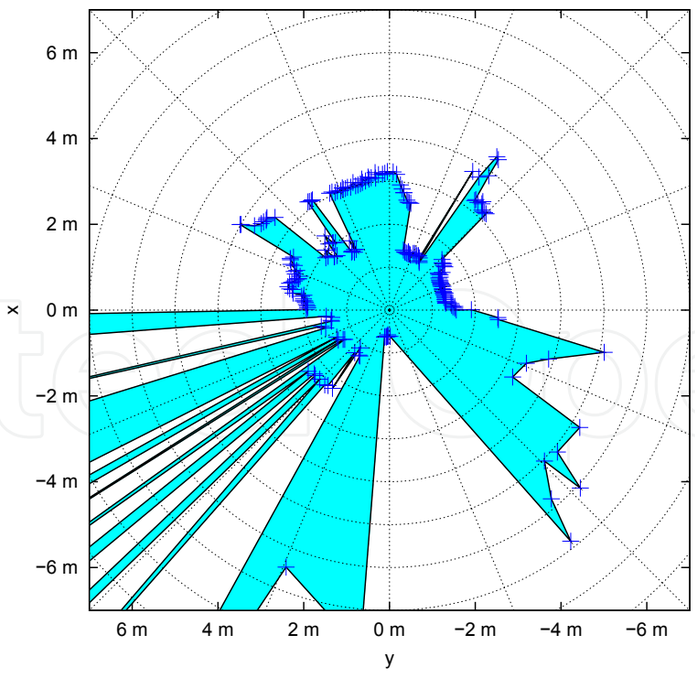


Fig. 5. Visualization of a 360° obstacle map containing information about nearest obstacles aggregated while moving through a cluttered indoor environment.

The virtual maps are ego-centric, i.e., the robot's center of rotation forms the origin of the map's coordinate frame and the coordinates of points stored in the maps are referenced to the base coordinate frame $\{B\}$ of the robot. In addition to being an efficient representation of relevant 2.5D information from 3D data, the maps can also be used to fuse information from several sensors, e.g., different 2D laser range finders or a combination of 2D and 3D laser scanners. In fact, the concept of virtual maps has recently been adopted by Stiene & Hertzberg (2009) to fuse information of different 2D range scans. Furthermore, as the representation (and the actually used implementation) does not differ from that of a standard 2D laser range scan, basically all algorithms working on 2D laser data can be applied to the aggregated or fused information in a virtual map. The remainder of this section will describe the two types of virtual maps as well as the procedures for updating them.

4.2 2D Obstacle Maps

In the case of the obstacle maps, the point corresponding to the *minimum* distance in each scan direction ($\theta_{\text{yaw},i}$), projected into the xy-plane in which the robot is moving, is extracted and inserted into the virtual map. These measurements correspond to the closest objects or obstacles in that particular direction regardless of the actual pitch angle θ_{pitch} of the scanner. Of course, only those points whose height above ground would intersect, respectively, the robot's bounds and the virtual corridor are inserted into the map. This explicitly includes obstacles like small objects lying on the ground or overhanging objects like open drawers as shown in Figure 4. Objects not intersecting the virtual corridor pose no threat to the robot and can thus be ignored. In the update procedure of *2D structure maps* they are, of course, used since a lot of information especially on higher environmental structures would be neglected otherwise.

In order to represent non-traversable areas and especially areas that correspond to holes in the ground, like for instance descending stairs in the examples of Figure 2, artificial obstacles are inserted into the map. Such non-traversable areas are characterized by those distance measurements that correspond to points in the real environment that are located below floor level. Note that the robot is assumed to be only able to traverse almost flat floor areas what is, after all, a feasible assumption for domestic indoor environments. Once a measurement below floor level occurs in an acquired laser scan its intersection with the floor plane is computed and an artificial measurement is added to the obstacle map at exactly this point. Thereby, the robot is able to perceive descending stairs and stop before the first step is reached. Such an intersection point and artificial distance measurement representing the stair as an obstacle is depicted with a red cross in Figure 4.

An important issue when constructing virtual obstacle maps is to not add points to the map that correspond to traversable surface as the robot would otherwise avoid to move through that particular region. Under the assumption of a perfect flat floor and minimum measurement inaccuracies, the most straightforward way for determining which points belong to the floor plane, is to apply a simple height thresholding. That is, all points whose height lies approximately at $z = 0$ are filtered out and ignored in the update procedure, i.e., $floor(\mathbf{p}_i) = \mathbf{p}_i^z \in [-\epsilon_z, \epsilon_z]$ where ϵ_z represents a tolerance according to the sensor's accuracy. Amongst others, Yuan et al. (2009) follow this approach. A value of $\epsilon_z = 2.5\,\text{cm}$, for example, is an appropriate tolerance for the aforementioned sensor setup leading to a robust removal of floor points but would also neglect measurements on the surface of small objects lying on the ground if their height does not exceed 2.5 cm.

A more reasonable, still simple and efficient, way for determining floor points is to evaluate the neighborhood of individual measurements. Nüchter et al. (2005), for example, apply a segmentation algorithm that exploits the order of points in a range scan, the order of range scans in the 3D point cloud and the 3D sensor setup to classify points regarding their correspondence to floor, wall, object or ceiling structures. It originates from the work in (Wulf et al., 2004) and will be used in the remainder of this section.
By the aforementioned means, i.e., filtering out minimum distances in each direction while ignoring points belonging to traversable surfaces, the robot obtains an egocentric map containing all obstacles and non-traversable areas close to the robot. An obstacle map that exemplarily shows how small objects are perceived is depicted in Figure 6.a.

(a) Obstacle map (b) Example scenario (c) Structure map

Fig. 6. Demonstration of the virtual 2D map types in an example scenario. The *obstacle map* (a) is generated by extracting minimum distances (projected into 2D) in the continuously acquired 3D data. Extracting maximum distances results in the *structure map* (c).

4.3 2D Structure Maps

In the case of virtual structure maps, the *maximum* distance in each scan direction ($\theta_{\text{yaw},i}$), projected into the xy-plane, is extracted and inserted into the map. Extracting maximum distances automatically filters out all objects that do not extend over the full height of the AOI since the scanner will eventually look above or beneath these objects. The robot thereby replaces a previously measured smaller distance value with the newly obtained larger distance reading in that direction. The resulting map will thus only contain those points that most probably correspond to the environmental bounds while all points that belong to smaller or overhanging obstacles are filtered out as are those that belong to dynamic obstacles. Such a structure map is exemplarily depicted in Figure 6.c. Whereas the obstacle map shows the red and blue cans representing the obstacle type of small objects lying on the ground, the structure map only contains the environmental boundaries. When updating structure maps points belonging to the floor or to traversable surfaces do not need to be ignored. They are inherently replaced by points being measured on environmental structures farther away from the robot. Instead, maximum range readings need to be sorted out as they would otherwise replace shorter but valid measurements on environmental structures. However, compared to determining floor points, an according procedure is straightforward.
While the obstacle maps are very valuable when it comes to local collision avoidance, the structure maps are, for instance, very suitable for robotic self-localization, i.e., for tasks that need large scale information about an environment. When sensing un-occluded parts of walls

and ceilings, using structure maps in a localization algorithm such as Monte-Carlo Localization will have similar effects as localization based on ceiling structures (Wulf et al., 2006). The obstacle maps will surely not be appropriate for such purposes as they would miss a lot of information about environmental structures.

4.4 Update Procedures of Obstacle and Structure Maps

The steps to keep both representations egocentric and to update them, according to newly acquired range scans and the definitions above, are:

1.) **Transformation** of the map to keep it sensor-centric (e.g., according to odometry or a known pose).

2.) **Removal** of obsolete points to handle dynamics and inaccurate pose shift estimates.

3.) **Replacement** of already saved points using more relevant points from the current laser scan.

If the robot stands still and no pose shift has been estimated respectively, steps 1.) and 2.) are skipped. The same holds true if the virtual maps are used as efficient representations of single 3D sensor readings, e.g., as obtained from 3D cameras (see Section 7). In its initial state, the map is filled with *dummy points* that are chosen in a way that they are immediately replaced during in the first update, i.e., points corresponding to the maximum measurable distance for obstacle maps and distances of 0 m for structure maps. As soon as a valid measurement has been taken in the direction of the dummy point, it replaces the dummy. As these measurements correspond to either 0 m or maximum range readings, they are naturally ignored in algorithms processing 2D laser range scans (and the virtual maps respectively).

4.4.1 Transformation of the map to keep it egocentric

According to the robot's movement the pose shift between the current and the last map update (e.g. current and last reception of a laser scan) consists of a rotation $\mathbf{R}_{\Delta\theta}$ around the z-axis by an angle $\Delta\theta$ and a translation $(\Delta x, \Delta y)^T$. The egocentric maps need to be transformed according to:

$$\begin{pmatrix} x_{i,t+1} \\ y_{i,t+1} \end{pmatrix} = \begin{pmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{pmatrix} \begin{pmatrix} x_{i,t} \\ y_{i,t} \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{4}$$

where t and $(t+1)$ represent discrete points in time.

As Eq. (4) transforms the map based on Cartesian coordinates, the values of the polar coordinates have to be adjusted accordingly. Note, that the polar coordinates are not only stored and kept up to date in order to avoid unnecessary transformations in individual algorithms but also to avoid repetitive re-calculations in the update procedure itself. That is, even if not a single algorithm operates on the polar coordinates of the points stored in the virtual maps, this step cannot be skipped without a degradation of performance.

Due to the discretization of the $N = \Theta/\Delta\theta_{\text{yaw}}$ valid angles two points can potentially fall into the same vector index. In this specific case the point being more relevant with respect to the map type has priority. That is, in an obstacle map, for example, a point with a smaller distance replaces a point in the same angular interval that is farther away from the robot. Furthermore, vector indices being unassigned after the transformation are filled with dummy points.

4.4.2 Removal of obsolete Points to handle Dynamics

If the maps are not intended to only represent the gathered 3D information of one rotation over the AOI or a single range image as acquired by a 3D camera, but to be used endlessly, i.e., updated with every sensor reading, it is suggestive to remove points after a certain while. Therefore, the number of transformations applied during step 1.) is stored for every single point. To deal with dynamic obstacles, a saved point is removed and replaced by a dummy point after its count of transformations exceeds a certain threshold (e.g., 500 transformations, approx. 5 s in the case of the continuously pitching IAIS 3DLS). This is because an obstacle passing by or crossing the robot's path leaves a trace of non-existent points in the obstacle map. This is not a drawback of the approach but a simple accounting for the uncertainty in the obstacle's movement. Points being removed by this means that correspond to static obstacles will immediately be measured again if the obstacle is still in the virtual corridor and thus still originates a possible source for a collision. The effect of applying this step can be seen in the region behind the robot in Figure 5. Here the point density is smaller than in the angular region in front of the robot, as some object points have already been removed and not sensed again since the removal.
Furthermore, if pure odometry is used to estimate the robot's pose shift for the transformation in step 1.) instead of pose tracking and scan matching to accurately determine the pose shift, errors and inconsistencies in both types of maps may arise from imprecise odometry. These are, in the same way, removed from the map. As a side note, it is to remark that even the rotated single 2D laser scans during the nodding-like movement of the sensor can be used for fast pose tracking algorithms like the one presented in (Lingemann et al., 2005b) if floor points are filtered out and the number of residual points is still sufficient for matching a newly acquired scan against the last one. Here, we apply the scan matching algorithm that is going to be described in Section 6.

4.4.3 Replacement of already saved Points

The final update procedure highly depends on the map type as described above. In a nutshell, a point $\mathbf{p}_i$ stored in an obstacle map is replaced by a point $\mathbf{s}_i$ in the current laser scan S if the angle of acquisition $\mathbf{s}_i^\theta$ falls into the discretized angular interval of $\mathbf{p}_i^\theta$ and the measured distance $\mathbf{s}_i^d$ is less than or equal to $\mathbf{p}_i^d$. In the same way a point $\mathbf{p}_i$ stored in a structure map is overwritten with $\mathbf{s}_i$ if $\mathbf{s}_i^\theta = \mathbf{p}_i^\theta$ and $\mathbf{s}_i^d \geq \mathbf{p}_i^d$. The height $\mathbf{s}_i^z$ of an acquired point in a perceived environmental structure is used as an additional information in both types of maps resulting in a 2.5D representation. That is, the virtual maps store for each discrete angle the polar coordinates $(d_i, \theta_{\text{yaw},i})$ as well as the Cartesian coordinates $(\mathbf{p}_i^x, \mathbf{p}_i^y, \mathbf{p}_i^z)$. This will be extended in future work to not only store the particular height of the most recent points but to store minimum and maximum height of all points measured within a range of approximately 10 cm around that most recent point or in a way comparable to *Multi-Level Surface Maps* (Triebel et al., 2006). By this simple extension a complete egocentric 3D model of the surrounding environmental structures can be reconstructed on the basis of the virtual 2D maps. In the case of obstacle maps the height information $\mathbf{p}_i^z$ of an acquired point $\mathbf{p}_i$ is also used to neglect those points that do not lie within the virtual corridor and are hence not relevant for representing nearby obstacles. This has the effect, that the robot can, for example, underpass a table as long as the table top is higher than the highest point on the robot and the passage between the table legs is not too narrow.

5. Simple Reactive Collision Avoidance using Virtual Obstacle Maps

In addition to the goal-directed motion control of a mobile robot, e.g., to reach a certain position, reactive collision avoidance is important in dynamic and human-populated environments. That is, the motion of the robot needs to be adapted in the presence of obstacles suddenly appearing in the robot's vicinity. The virtual obstacle maps can be used with any collision avoidance procedure known from processing standard 2D range scans. As a simple example, we use a set of three simple reactive behaviors controlling and adapting the robot's translational and rotational velocities based on the robot's current movement direction and speed as well as surrounding objects modeled in the obstacle map. The obstacle map is constructed and updated during navigation. These behaviors and the corresponding algorithms have been initially introduced in (Lingemann et al., 2005a).

The first behavior slows down the robot if obstacles appear, respectively, in the virtual corridor and in front of the robot in the virtual obstacle map. If the distance to the nearest obstacle in the virtual corridor falls below a certain threshold, the robot is completely stopped. Another behavior turns the robot on the spot once it has been completely stopped. This avoids that the robot gets caught in dead ends or corners. Alternatively, the robot can be moved backwards so that it is positioned in free space again. Then an alternative path can be planned to reach the position that is to be approached.

The third behavior is more complex compared to the aforementioned ones. It slightly adapts the rotational velocity of the robot so that it prefers moving along free space. Consider for example, the robot has planned a path along a longer corridor. As this path results from searching for the shortest path between two positions, it can run directly along one of the walls. When exactly following such a path, this third behavior causes that the robot is not directly moving along the wall, but instead along the center of the corridor. The concept of the behavior is to steer the robot towards a *freespace* orientation.

5.1 Determining the Freespace Orientation

The origin of determining the freespace orientation lies in the early work of Surmann & Peters (2001) for fuzzy-based control of autonomous mobile robots. A comparable behavior was obtained by applying a fuzzy controller with fuzzy rules like the following:

```
IF COMMAND is straight-ahead
  AND IF FRONT-SENSOR is very-near
     AND FRONT-LEFT-SENSOR is very-near
     AND FRONT-RIGHT-SENSOR is near
THEN SPEED is positive-small, ANGLE is negative-small
```

An adaption to 2D laser range scans has been presented in (Lingemann et al., 2005a). Here the following fuzzy rule is applied to every single distance measurement:

```
IF (angle_i is in driving direction) AND (distance_i is large)
THEN drive in this direction.
```

The actual driving direction of the robot, the wanted freespace orientation α_{free}, further adapted to meet our requirements, results as follows and is based on the above rule.

$$\alpha_{\text{free}} = \text{atan2}\left(\sum_{i=1}^{N} \sin \mathbf{s}_i^{\theta} \cdot f_{\theta}(\mathbf{s}_i^{\theta}) \cdot f_d(\mathbf{s}_i^{d}),\ \sum_{i=1}^{N} \cos \mathbf{s}_i^{\theta} \cdot f_{\theta}(\mathbf{s}_i^{\theta}) \cdot f_d(\mathbf{s}_i^{d})\right) \tag{5}$$

The functions $f_\theta(\mathbf{s}_i^\theta)$ and $f_d(\mathbf{s}_i^d)$ relate the i-th range reading in the form of the polar coordinates $(\mathbf{s}_i^\theta, \mathbf{s}_i^d)_{i=1\ldots N}$ as obtained from a 2D laser scanner or an obstacle map to the fuzzy sets *"angle is in driving direction"* and *"distance is large"*. N is the number of points in the map and the laser range scan respectively.

$$f_\theta(\theta) = \cos\left(\frac{\theta}{1.2}\right) \tag{6}$$

$$f_d(d) = \frac{1}{1+\exp\left(-\left(\frac{d-dto_{\max}}{dto_{\min}}\right)\right)} \tag{7}$$

Here $dto_{\min}$ and $dto_{\max}$ determining the slope and the inflection point of the exponential, correspond to the thresholds of the behavior that slows down the robot. That is, if an object appears in front of the robot within a range $dto_{\max}$, the behavior starts slowing down the robot according to the distance to that object. If the distance to the object falls below $dto_{\min}$, the robot is completely stopped or moved backwards. Plots of the weighting functions $f_\theta(\theta)$ and $f_d(d)$ as well as the resulting application of the fuzzy AND by means of a multiplication are shown in Figure 7.

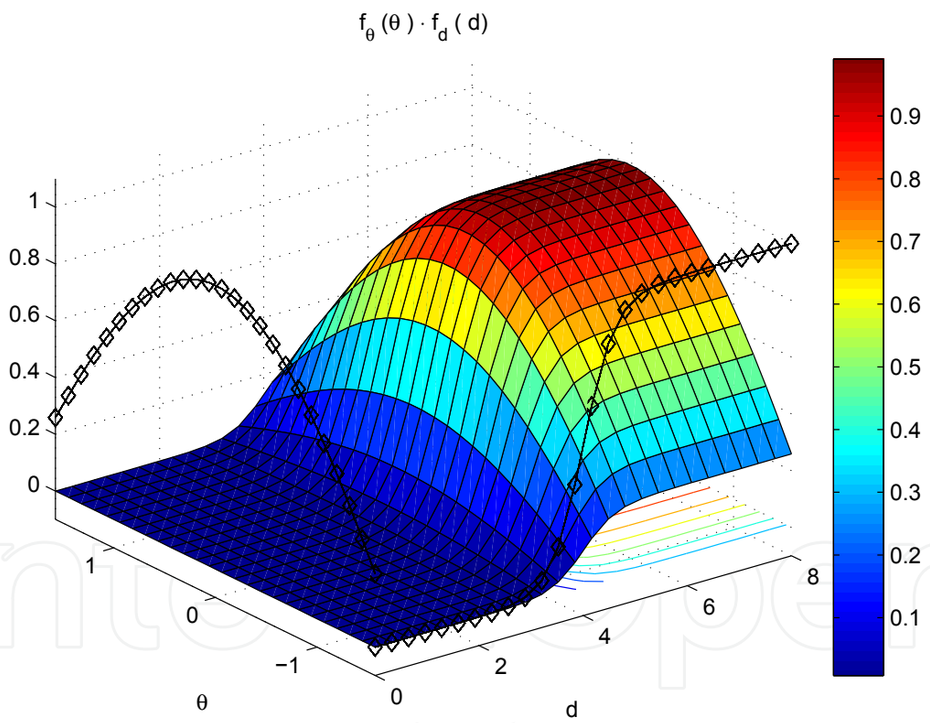


Fig. 7. Composed weighting function $f_\theta(\mathbf{s}_i^\theta) \cdot f_d(\mathbf{s}_i^d)$ to determine the freespace orientation in a laser scan or obstacle map S.

The behavior simply adapts the rotational velocity, e.g., as set by a motion controller for following a planned path, so that the robot slightly moves towards free space thereby swerving to avoid collisions. The influence of the behavior can be adapted and is kept rather small so that the robot can enter narrow passages and follow paths that lead away from the maximally free space in the robot's workspace. Referring to the resulting weighting function $f_\theta(\theta) \cdot f_d(d)$, the robot prefers moving straight and not adjusting its translational velocity.

5.2 Typical Results

A typical result of applying the three behaviors during navigation is shown in Figure 8. The robot was put into an example scenario bounded by movable walls with an exit in the opposite corner. The experiment was repeated two times. In the first run the scanner was held in a horizontal position (2D perception) comparable to a standard 2D laser range finder. In the second run, the scanner was continuously rotated over the aforementioned area of interest in a nodding-like fashion (3D perception). In both experiments, a constant translational velocity of $0.3\,\mathrm{m\,s^{-1}}$ was set with no rotational velocity. That is, the robot was commanded to move forward. The application of the behaviors successfully moved the robot through the exit and outside the scenario. With 2D perception, small test objects lying on the ground were not perceived. The robot crushed into these objects and pushed them through the exit. With 3D perception, the objects have been perceived and the robot successfully avoided them. Although it is not explicitly covered in this example, it is to note that the robot robustly perceives sudden dynamic changes in the environment due to the fast pitch rate of the scanner while driving. It therefore detects and avoids dynamic obstacles like, for instance, suddenly opened drawers or people passing by.

What can also be seen in the figure is that simply commanding the robot to move forward while enabling the three collision avoidance behaviors leads to an emergent behavior of wandering around. This strategy can, amongst other applications, be used to perform a random exploration.

(a) 2D Perception (b) Example scenario (c) 3D Perception

Fig. 8. Behavior-based collision avoidance in an example scenario by means of 2D perception (2D laser range finder in a fixed horizontal position) and continuous 3D perception together with the concept of obstacles maps. The small test objects are successfully avoided by the robot when using 3D perception. A video showing the robot perform in a similar experiment is available under `http://www.b-it-bots.de/media`.

6. Simultaneous Localization and Mapping

In the previous sections we showed that continuous 3D environment sensing together with the concepts of an area of interest and the virtual corridor enables an autonomous mobile robot to perceive and react to various obstacles being typical for domestic environments, like for instance open drawers, small obstacles lying on the floor or descending stairs. In this section we will show how the same continuous 3D data flow of the pitching laser scanner can be used for 2D and 3D *Simultaneous Localization and Mapping (SLAM)*, i.e., constructing two-dimensional and three-dimensional environment representations and localizing the robot

with three and six degrees-of-freedom (DOF) respectively. Central questions in this context are:

1. **Which information** can be used for 3DOF- and 6DOF SLAM respectively?
2. **Which SLAM algorithms** can be used to process this information?

6.1 Extracting Information and Forming Local Point Clouds for SLAM

The pitching laser scanner delivers a continuous stream of 3D data acquired during the robot's movement through the environment. The first question addresses the segmentation of this data stream and the extraction of locally consistent information about environmental structures usable in a SLAM approach. In the two-dimensional case (3DOF-SLAM) this is quite straightforward. By storing maximum distances in each measurement direction the virtual structure maps store aggregated information about boundaries of the environment, like for instance walls, as well as larger static obstacles. A these maps are already represented in the form of a 2D laser range scan, the contained information can be used with any known SLAM algorithm working on range scans. Constructing individual structure maps for every full rotation over the area of interest (AOI) results in locally consistent virtual range scans. Due to the fact that the rotational velocity of the scanner as well as the size of the AOI are adjusted according to the robot's current velocities, these virtual range scans have a sufficient overlap, e.g., for applying scan matching algorithms. However, the range scans are only locally consistent under the assumption that the pose shift estimates, used for keeping the structure maps egocentric during construction, are not too inaccurate. Solely transforming the maps based on inaccurate or erroneous odometric pose shift estimates can lead to inconsistent information, distorted environmental structures when turning fast for example. To account for that, we interrupt the construction of a structure map when the pose shift estimates suggest that the robot is turning fast and start building a new structure map after that turn. A similar procedure is followed by Cole & Newman (2006). However, matching consecutive scans, as described later in this section, can correct false or inaccurate pose shift estimates and the aforementioned interruption is only applied when solely using odometry information.

Another straightforward way of extracting information for 3DOF-SLAM is to take those 2D range scan that have a measurement plane being parallel to the ground, i.e., scans taken at $\theta_{\text{pitch}} = 0$ for a flat floor. Such a scan can be taken as is and is comparable to a range scan acquired by a fixed horizontally mounted 2D laser range finder. Both the horizontal laser scans as well as the constructed structure maps can be used with any SLAM algorithm processing range scans, e.g., Rao-Blackwellized Particle Filters (Grisetti et al., 2007).

For being able to construct three-dimensional environment models and to localize the robot with six degrees-of-freedom (6DOF-SLAM) we need to extract locally consistent 3D point clouds out of the continuous data stream delivered by the pitching scanner while moving. That is, we want to construct individual 3D point clouds that are each referenced to a distinct vehicle pose, the base pose $\mathcal{P}_b$, just like 3D range scans acquired while standing (as in Surmann et al., 2003). We therefore strip the robot's movement in space during one complete pitch movement by transforming successively the thereby gathered 2D scans according to the estimated relative pose shift between the current robot pose and $\mathcal{P}_b$. The scans are then combined to form one 3D point cloud that, depending on the currently used pitch rate and size of the AOI, consists of 15 to 500 single 2D laser scans. A point cloud being generated by this means is shown in Figure 9.

The figure also shows the results of different processing steps that are performed for every 2D laser range scan acquired during the nodding-like movement of the scanner: 1.) reducing

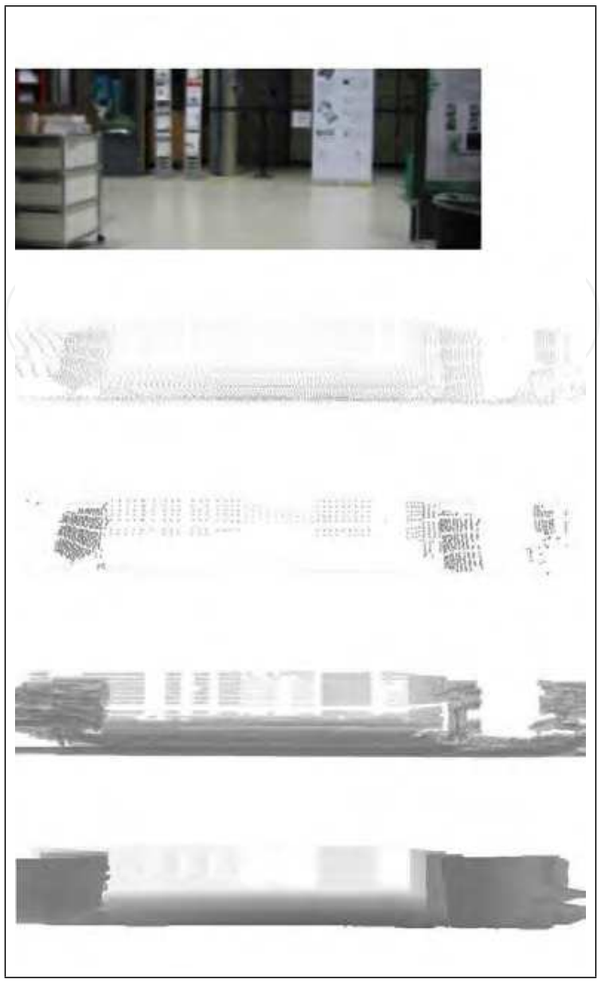


Fig. 9. Example of a generated point cloud. The figure shows (from top to bottom): a photo of the captured scene, the range measurements acquired during movement, the reduced set of points (without floor points), detected lines and a rendered depth image.

the point density in the scan by replacing clusters of points, whose distance to each other falls below some threshold, by their centroid, 2.) detecting line segments in the residual points, 3.) merging line segments to planes and 4.) classifying points whether or not they belong to the floor. All these algorithm are described, in detail, in (Surmann et al., 2003).

6.2 A Brief Overview on SLAM Algorithms

Performing SLAM to build maps and localizing in preliminary built maps are major preconditions for the autonomous operation of mobile robots in changing or preliminary unknown environments. Approaches addressing mapping and localization differ, amongst others, in formulating the problem, the means to cope with the addressed problem and in representing the environment. A majority of SLAM algorithms is probabilistic and based on formulations using *Extended Kalman Filters* (EKFs, Leonard & Feder, 1999), *Unscented Kalman Filters* (UKFs, Chekhlov et al., 2006), Sparse Extended Information Filters (SEIFs, Thrun et al., 2004) or *Rao-Blackwellized Particle Filters* (RBPFs, Grisetti et al., 2007). Graph-based SLAM algorithms

(Grisetti et al., 2008; Olson et al., 2006) address SLAM in terms of nonlinear optimization and are often used as generic back-ends to distribute accumulated errors after having detected loops in the robot's trajectory or to better align graphs of robot poses.
Another way of formulating SLAM is that by interpreting it as a problem of *multi-view range image registration*, i.e., integrating and aggregating range measurements taken at different positions and orientations in the environment into a common coordinate frame forming the environment model. This problem can be formulated as follows: Given two sets of geometrical features, a data set D or *scene* and a data set M or *model*, find a transformation $\mathbf{T}$ that minimizes the alignment error between the two sets and correctly maps D onto M. The goal is to transform D in a way that it "fits best" with M. Normally, range images are taken of non-deformable, static objects with the same sensor but from diverse viewpoints, i.e., from different positions and under different orientations. In this case, $\mathbf{T}$ is a rigid transformation that includes only translation and rotation.

6.3 Incremental Registration using the ICP Algorithm and Sparse Point Maps

A widely used solution to the registration problem is the Iterative Closest Point (ICP) algorithm by Besl & McKay (1992), which determines $\mathbf{T}$ in an iterative way. In each iteration step, the ICP algorithm determines pairs of corresponding points from D and M using a nearest-neighbor search. These correspondences are used to quantify and minimize the alignment error E:

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{|M|}\sum_{j=1}^{|D|} w_{i,j}\|\mathbf{m}_i - \left(\mathbf{R}\mathbf{d}_j + \mathbf{t}\right)\|^2, \quad w_{i,j} = \begin{cases} 1, & \mathbf{m}_i \text{ corresponds to } \mathbf{d}_j \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

$$\mathbf{T} = \begin{pmatrix} \mathbf{R}_{ICP} & \mathbf{t}_{ICP} \\ 0\,0\,0 & 1 \end{pmatrix} \quad \text{with } (\mathbf{R}_{ICP},\mathbf{t}_{ICP}) = \arg\min_{\mathbf{R},\mathbf{t}} E(\mathbf{R},\mathbf{t}) \tag{9}$$

Finding the nearest neighbors and determining the correspondences is the computationally most expensive step in the ICP algorithm ($O(|D|\,|M|)$ for a brute-force implementation), since for every point $\mathbf{d}_j \in D$ the closest point $\mathbf{m}_i \in M$ needs to be determined. Here, we use an approximate *kd*-tree search (Arya et al., 1998), which reduces the complexity of the algorithm to $O(|D|\log|M|)$.
To estimate the rigid transformation $\mathbf{T}$, consisting of a rotation $\mathbf{R}$ and a translation $\mathbf{t}$, that minimizes Eq. (8) there are closed form solutions in both the two- and three-dimensional case (Lorusso et al., 1995). In order to cope with only partially overlapping sets, we reject correspondence pairs for which the point-to-point distance exceeds a certain threshold. This threshold exponentially decays during the registration process. While initially permitting larger distances between corresponding points guarantees fast convergence of $E(\mathbf{R},\mathbf{t})$, smaller distances in later iteration steps allow fine-tuning the registration result. Furthermore, we reject pairs that contain the same model point and only keep the pair with the closest point-to-point distance (Zinßer et al., 2003).
For registering multiple range scans and constructing a consistent map that models environmental surfaces, an incremental registration procedure is used. The first laser scan D_0 is used as the initial environment model M_0. Thus, the local coordinate frame of D_0 forms the coordinate frame for the overall map. All subsequent scans $D_i, i > 0$ are matched against M_{i-1}. The resulting transformation $\mathbf{T}_i$ is used to correct the position of all points contained in D_i, yielding the transformed point set $\check{D}_i = \{\check{\mathbf{d}}_{i,j} \mid \check{\mathbf{d}}_{i,j} = \mathbf{R}\mathbf{d}_{i,j} + \mathbf{t}\}$. As an initial estimate $\hat{\mathbf{T}}_i$ for $\mathbf{T}_i$ in this incremental registration we use the transformation from the last registration, i.e.,

$\hat{\mathbf{T}}_i = \mathbf{T}_{i-1}$. This speeds up the convergence in the ICP algorithm and drastically reduces the probability of converging to a local minimum possibly resulting in an incorrect registration result. If odometry information is available, the estimate $\hat{\mathbf{T}}_i$ is further corrected taking into account the estimated pose shift between the acquisition of D_{i-1} and D_i. Furthermore, we only register a new range scan D_i if the robot traversed more than a certain distance, for example 50 cm, or turned more than a certain angle, for example 25°. Such a practice is quite common in a variety of recent SLAM algorithms.
To account for possibly new information in D_i, the transformed points are than added to M_{i-1}. That is, after matching range image D_i, the model set M_{i-1} computed so far is updated in step i to:

$$M_i = M_{i-1} \cup \{\check{\mathbf{d}}_{i,j} \mid \check{\mathbf{d}}_{i,j} \in \check{D}_i\}. \tag{10}$$

Thus, a model M_N, constructed by incrementally registering N range images, contains all points measured in the environment, i.e.

$$M_N = \bigcup_{i=[0,N]} \{\check{\mathbf{d}}_{i,j} \mid \check{\mathbf{d}}_{i,j} \in \check{D}_i\}. \tag{11}$$

The main problem of this incremental registration approach is its scalability with respect to the size of the environment and the number of range images taken. To fully cover a large environment, a lot of range images might be needed. When registering and adding all acquired range images, the model set M can get quite large, e.g. several million points for 3D scans taken in a large outdoor environment (Nüchter et al., 2007). However, when acquiring range images in parts of the environment which are already mapped, lots of points would be added to M without providing new information about the environment. This is exploited by the following improvement to our SLAM approach, which makes the point clouds sparse.
The key idea of *sparse point maps* is to avoid duplicate storage of points, and thereby minimize the amount of memory used by the map, by conducting an additional correspondence search. Correspondence is, thereby, defined just like in the ICP algorithm. That is, a point $\check{\mathbf{d}}_{i,j} \in \check{D}_i$ is not added to M_{i-1}, if the point-to-point distance to its closest point $\mathbf{m}_{i-1,k} \in M_{i-1}$ is smaller than a minimum allowable distance ϵ_{D}.

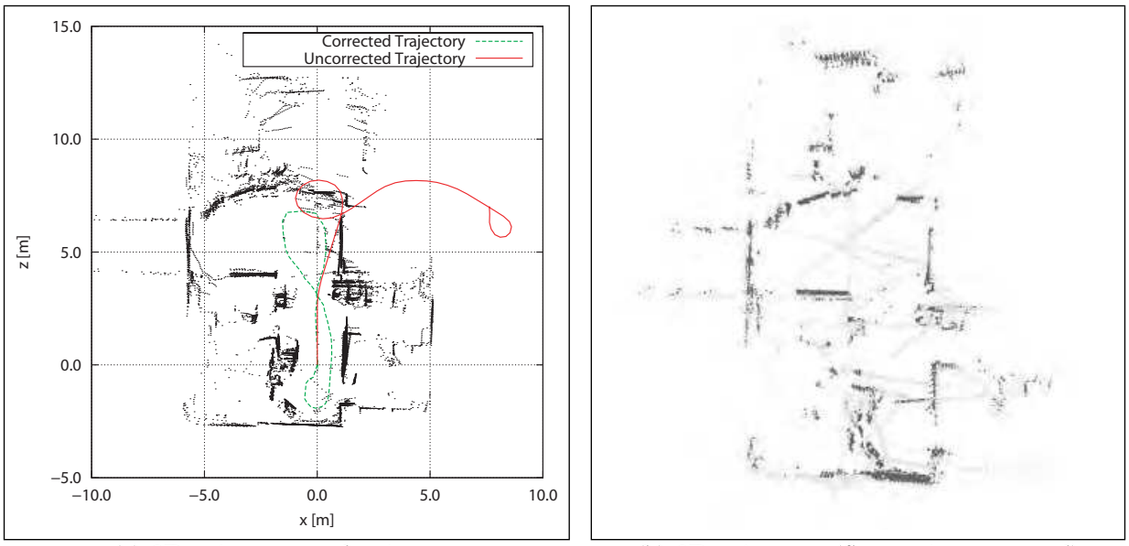
$$M_i = M_{i-1} \cup \{\check{\mathbf{d}}_{i,j} \mid \check{\mathbf{d}}_{i,j} \in D_i, \nexists \mathbf{m}_{i-1,k} \in M_{i-1} : \|\check{\mathbf{d}}_{i,j} - \mathbf{m}_{i-1,k}\| < \epsilon_{\mathrm{D}}\} \tag{12}$$

The threshold ϵ_{D} spans regions in the model in which the number of points is limited to 1, thereby providing an upper bound on the point density in a sparse point map M. Choosing a value of ϵ_{D} according to the accuracy of the range sensor used will exactly neglect duplicate storage of one and the same point assuming correct alignment of range images. Choosing, however, a larger value allows to reduce the number of points stored in the map. Although some details of the environment might not get modeled, a map constructed in this manner still provides a coarse-grained model of the environment. In the actual implementation, the additional correspondence search is carried out on the *kd*-tree built for the ICP algorithm but using ϵ_{D} as the distance threshold in the pair rejection step. However, here the rejected pairs are used to determine the points in $\check{D}_i$ that need to be added to M_{i-1}.

6.4 3DOF-SLAM and 6DOF-SLAM in an Example Scenario

The following figures show typical results from applying the above described algorithms for extracting 2D and 3D laser range scans out of the continuous 3D data stream and the incremental registration using the ICP algorithm. In this example the robot was manually driven

through a laboratory environment. The earlier described methods and means for avoiding collisions were applied in order to change the manually set velocities in order to automatically swerve around obstacles. Out of the stream of rotated 2D range scans delivered by the continuously pitching 3D scanner both virtual structure maps as well as locally consistent 3D point clouds have been extracted. These point sets were then incrementally registered using the ICP algorithm and the aforementioned extensions. The two-dimensional sparse point map obtained from matching the structure maps is shown in Figure 10(a) as well as the estimated trajectory of the robot. Incrementally registering the generated 3D point clouds results in the 3D point model shown in Figure 10(b) in a top-down perspective with points classified as corresponding to floor not being visualized. Views on the complete 3D model including floor points are shown in Figure 11.

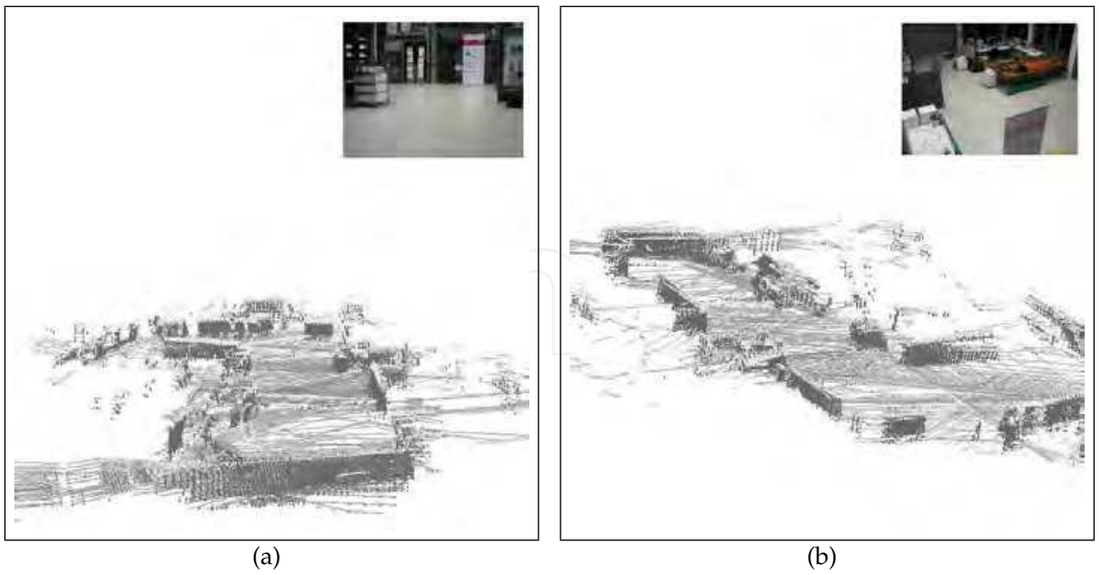


(a) 2D point map and trajectory (b) 3D point map (floor points removed)

Fig. 10. Shown here are a two-dimensional sparse point map (a) as well as a three-dimensional sparse point map (b) of an example scenario. Measurements corresponding to flat floor have been removed.

7. Extensions for using recent Time-of-Flight Cameras

An inherent drawback of the nodding-like movement of the continuously pitching laser scanner is the high mechanical load of the actuator especially when accelerating the scanner at the lower and upper bound of the AOI, i.e., when changing the rotation direction. Longer operations can cause an increase in gear backlash. Estimating the pitch angle solely based on the current position of the servo motor can result in inaccuracies, i.e., the estimated angle may deviate from the actual rotation angle of the pitching scanner. This effect can be seen in Figure 12(a). In this example a 3D scanner, that had already been used for several years without changing motor or gear, was rotated from $-30°$ to $15°$ and back to $-30°$. A vertical slice of the generated point cloud is extracted that contains measurements on the planar floor as well as on a vertical plane. Due to the inaccuracies in the estimation of the pitch angle, the resulting geometric structure is distorted. When explicitly measuring the pitch angle, like for



(a) (b)

Fig. 11. Different views on the complete 3D model. It can be seen that the map does only model environmental structures in a height of the robot. That is, only those regions are modeled with which the robot can interact.

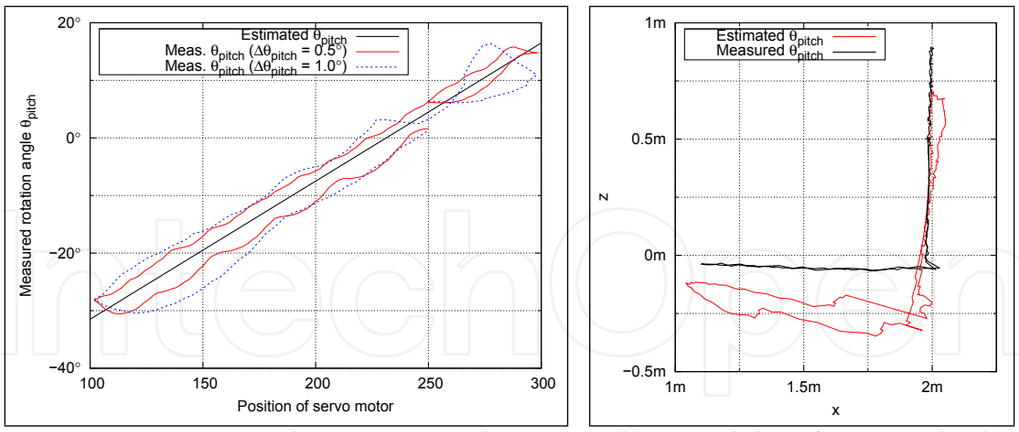
instance using gyroscopes or an inertial measurement unit, allows for correctly reconstructing the geometric structure as shown in Figure 12(b).

In contrast to custom built 3D scanners as the continuously pitching scanner used here, Time-of-Flight (ToF) cameras directly deliver 3D point clouds without a mechanical actuator. These solid-state sensors emit an amplitude-modulated signal using an array of near-infrared light emitting diodes (LEDs). The on-chip calculation of depth information is based on the phase shift between the emitted and the received signal. ToF cameras provide depth images at high frame rates while preserving a compact size, a feature that has to be balanced with measurement accuracy and precision. Besides the depth measurements, they also provide the amplitude of the reflected signal, which correlates to the reflectivity of the measured object.

Depending on external interfering factors (sunlight for example) and scene configurations, i.e., distances, surface orientations, and reflectivity, distance measurements from different perspectives of the same scene can entail large fluctuations. Furthermore, these sensors have, compared to 3D laser scanners, a restricted field of view, e.g., only 43° (H) $\times$ 34° (V) for a SwissRanger SR4000 from Mesa Imaging[5].

One of the first applications in robotics considering ToF cameras as an alternative to laser scanning, was presented by Weingarten et al. (2004). They evaluated a SwissRanger SR-2 camera in terms of basic obstacle avoidance and local path planning capabilities. In order to cope with the poor data quality, Weingarten et al. determined the parameters for the perspective projection into the image plane in a photogrammetrical calibration as well as scaling and offset values for correcting systematic errors in individual measurements. In order to detect obstacles they apply a simple thresholding to remove floor points. As a result they presented simple examples where an autonomous mobile robot using the SwissRanger camera was able

[5] Mesa Imaging SwissRanger cameras: `http://www.mesa-imaging.ch`



(a) θ_{pitch} in one up-down rotation cycle (b) Vertical slice of 3D point cloud

Fig. 12. Estimated and measured pitch angles. Depending on the gear backlash, pitch angles estimated based on the position of the servo motor may deviate from the actual pitch angle (a). Measuring the pitch angle, e.g., using gyroscopes still yields accurate and correct geometric information (b).

to stop in front of a table that would have not been perceivable using a 2D laser range scanner. However, they also mentioned that some objects and environmental structures have not been perceivable with the SwissRanger due to the smaller field of view (Weingarten et al., 2004). Sheh et al. (2006) also perform a per-pixel calibration similar to Weingarten et al. (2004) and determine a lookup table storing an offset and a multiplying factor for every distance measurement. In addition, Sheh et al. explicitly handle defect pixels. In order to handle the smaller field of view in the context of 3D mapping, they stop the robot, rotate the camera and acquire multiple range images under different rotation angles. The individual range images are then merged into a single point cloud and registered into a global point map. The actual registration is thereby assisted by a human operator. Swadzba et al. (2007) address the poor data quality of the sensor by using a sequence of filtering operations and register successive range images using a combination of feature tracking and registration with the ICP algorithm. Recently, Yuan et al. (2009) adopted the ideas of virtual scans (Wulf et al., 2004) and virtual obstacle maps (Holz et al., 2008) to fuse the sensory information of a ToF camera with that of a 2D laser range scanner. They apply the same pre-processing steps as Swadzba et al. to filter out and correct inaccurate or erroneous measurements. However, due to that fact the camera is fixed on the robot, they suffer from the same problem with the narrow field of view as Weingarten et al., namely that not all obstacles in the robot's vicinity can be perceived. Furthermore, just like Weingarten et al., they only perform a simple thresholding to classify floor points and detect obstacles. However, this procedure can cause different problems as already mentioned in Section 4.

The narrow field of view and the poor data quality necessitate several extensions when using a ToF camera in the context of SLAM and collision avoidance.

1. Due to inaccuracies and erroneous measurements, the acquired 3D point clouds need to be filtered and corrected in order to obtain accurate and correct geometric information.
2. Furthermore, the poor data quality necessitates a more robust classification of measurements for reliably detecting obstacles.

3. The narrow field of view needs to be explicitly taken into account in the registration with the ICP algorithm in order to avoid false correspondence when single range images contain less geometric features. When all measurements in the point cloud have been acquired on a single planar surface for example, the registration problem is underconstrained.
4. Again caused by the narrow field of view, the robot does not have all objects in sight when the camera is mounted at a fixed position. In order to adequately perceive obstacles in the robot's vicinity and its movement direction respectively, the camera needs to be rotated.

7.1 Filtering and Correction of Depth Measurements

Measurements from Time-of-Flight cameras are subject to different error sources (Lange, 2000). They can be divided into systematic and random errors. Systematic errors can be corrected by calibration (Fuchs & Hirzinger, 2008), whereas random errors can be coped with by means of filtering. A common way in filtering random errors is to neglect measurements based on their amplitude. Thresholding the amplitude neglects measurements from poorly reflecting objects, objects being farther away from the robot and objects in the peripheral area of the measurement volume.

Another source of errors in distance measurements are so called *jump-edges*. They occur at transitions from one shape to another. The shapes seem to be connected due to spurious measurements in between. These spurious measurements result from multiple-ways reflections which also cause that hollows and corners appear rounded off (Lange, 2000; May et al., 2009). Jump edges can be filtered by means of local neighborhood relations (May et al., 2009). From a set of 3D points $P = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, \cdots, N_p\}$, jump edges J can be selected by comparing the opposing angles $\theta_{i,n}$ of the triangle spanned by the focal point $\mathbf{f} = \mathbf{0}$, point $\mathbf{p}_i$ and its eight neighbors $P_n = \{\mathbf{p}_{i,n} \mid i = 1, \ldots, N_p : n = 1, \cdots, 8\}$ with a threshold ϵ_θ:

$$\theta_i = \max \arcsin \left(\frac{|| \mathbf{p}_{i,n} ||}{||\mathbf{p}_{i,n} - \mathbf{p}_i||} \sin \varphi \right), \tag{13}$$

$$J = \{\mathbf{p}_i \mid \theta_i > \epsilon_\theta\}, \tag{14}$$

where φ is the angle between two neighboring distance measurements. Since this filter is sensitive to noise, we apply a median filter to the depth image beforehand.

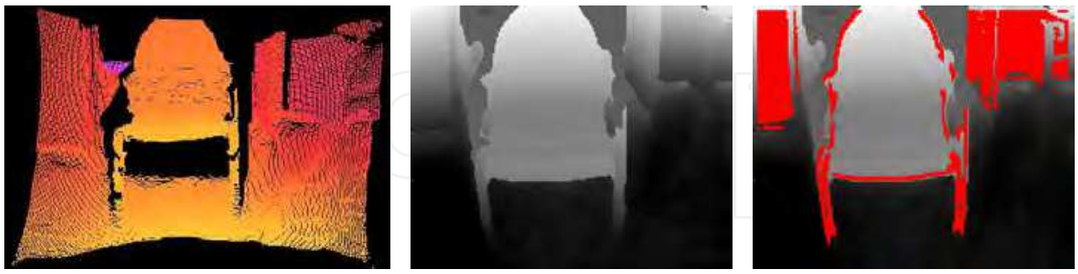
7.2 Detecting Obstacles and Information Fusion in Obstacle Maps

The filtered depth measurements are transformed to the robot's Cartesian coordinate frame by the extrinsic camera parameters, taking into account the sensor's height and orientation. A typical example of a resulting point cloud taken in an indoor environment is shown in Figure 13(a). The colors of the points correspond to the distance of a point from the sensor, brighter colors relate to shorter distances. This point cloud can be used to build a so-called *height image* as shown in Figure 13(b). The gray scale value of every pixel in the height value corresponds to the height, i.e., the z-coordinate of the respective point in the point cloud. A point $\mathbf{p}_{i,j}$ is classified as belonging to an obstacle if

$$(W_{\max} - W_{\min}) > \epsilon_H \tag{15}$$

where $W_{\max}$ and $W_{\min}$ are the maximum and minimum height values from a local window W, spanned by the Moore neighborhood around $\mathbf{p}_{i,j}$. The threshold ϵ_H thereby corresponds

to the minimum tolerable height of an obstacle. It needs to be chosen appropriately since it should not be smaller than the sensor's measurement accuracy. Due to evaluating a point's local neighborhood, floor points are inherently not considered as obstacles. The result of this filter is shown in Figure 13.



(a) Raw point cloud (b) Rendered height image (c) Obstacles

Fig. 13. Detecting obstacles in range images. Based on the acquired range image (a), a height image (b) is constructed. Filtering the height image results in the set of obstacle points (c).

The resulting obstacle points are used to extract a two-dimensional *virtual scan* similar to an obstacle map by 1.) projecting the 3D data into the *xy*-plane and 2.) extracting relevant information. The number of range readings in the virtual scan as well as its apex angle and resolution correspond to the acquired 3D data. For the SR4000, the number of range readings is 176, which is the number of columns in the image array. The apex angle and the angular resolution are 43° and 0.23°, which corresponds to the camera's horizontal apex angle and resolution. For every column of the ToF camera's distance image, the obstacle point with the shortest Euclidean distance to the robot is chosen similar to the update procedure of obstacle maps in Section 4. This distance constitutes the range reading in the scan. If no obstacle point is detected in a column, the scan point is marked invalid, by setting it to the maximum measurable distance of the sensor.
Figure 14(a) shows an example scene of an indoor environment. The point cloud which results from the ToF camera's depth image is shown in Figure 13. The result of the filtering and obstacle detection step is depicted in Figure 14(b). Points with a low amplitude are removed from the cloud. Points classified as belonging to obstacles and the extracted virtual scan are shown in Figure 14(c). Obstacle points are marked white and the obstacle points that contribute to the *virtual scan* are marked red. The remaining points are marked green.
The resulting *virtual scan* is fused with a 2D laser range scan yielding a common *obstacle map* modeling the closest objects in both sensors. The obstacle map for the aforementioned example scenario is visualized in Figure 15. Measurements from the laser range scan are illustrated by the blue points. The red points illustrate the *virtual scan*. The chair shows only a few points in the 2D laser range scan since only the legs of the chair are in the scan plane, whereas the *virtual scan* outlines the contour of the chair. By fusing the information of both sensors, the robot possesses correct information about traversable free space (light blue region) in its immediate vicinity. The obstacle maps can be used for collision avoidance just like the obstacle maps obtained from extracting relevant information from the continuous data stream of the pitching 3D laser scanner.

(a) Photo of the scene (b) Obstacles in height image (c) Point cloud and extracted scan

Fig. 14. (a) An example scene of an indoor environment. (b) The height image with detected obstacle points (red) . (c) The result of the filtering and obstacle detection step. Points with a low amplitude are removed from the point cloud. Obstacle points are marked white and the obstacle points that contribute to the virtual scan are marked red. The remaining points are marked green.

7.3 Frustum Culling as an Extension to the ICP Algorithm

The SLAM approach based on the ICP algorithm and described in Section 6 is quite sensitive to false correspondences, especially when the point sets only partially overlap. Points in a correspondence pair that do not model the same point in the physical environment can negatively affect the registration result possibly leading to an incorrect local minimum. Due to the narrow field of view of the SwissRanger camera, the overlap of the latest range image and the so far built model has to be handled more explicit than solely using the pair rejection extensions mentioned earlier. Here we apply a technique called *frustum culling* which known from 3D computer graphics. The *frustum* defines the volume that has been in the range of vision while acquiring the model point set M. Points in the data set D that do not lie within the model frustum, are neglected in the correspondence search as they cannot form a valid correspondence pair (May et al., 2009). Luck et al. (2000) use frustum culling in a preprocessing step to sort out points by means of an initial pose esimate. The registration is then conducted on the residual points. In contrast to that, we do not rely on an initial pose estimate and apply frustum culling in every iteration step of ICP algorithm. This takes into account that points in the data set are moved inside or outside the frustum during registration. Applying frustum culling in every iteration step effectively reduces the number of false correspondences in the registration and the total number of misregistrations caused by false correspondences. A detailed description of this approach as well as the aforementioned calibration and filtering procedures can be found in (May et al., 2009).

The procedure of neglecting points from the data set D that do not lie within the model frustum is visualized in Figure 16(a). Points of the model set M are shown in green. Those points of D that lie within the frustum and that are considered in, respectively, the nearest neighbor search and the registration of D are shown in red. Points from D that are neglected because of not lying in the model frustum are shown in blue. A point map constructed by incrementally registering range images using the aforementioned extensions and the frustum culling is visualized in Figures 16(b) and 16(c).

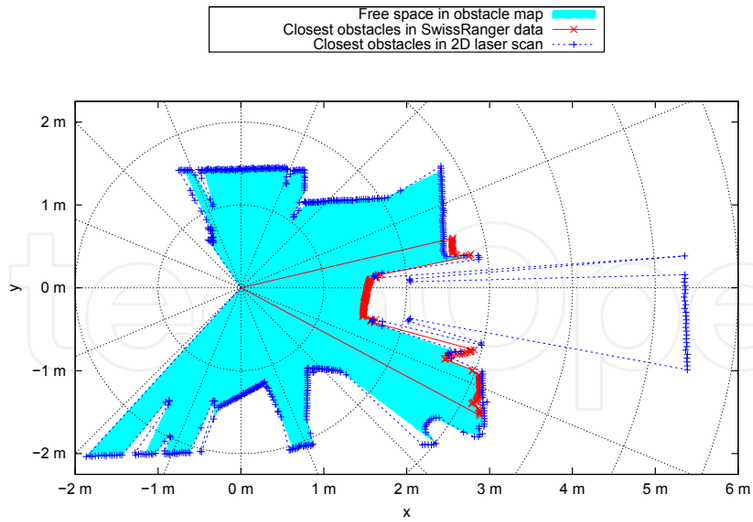
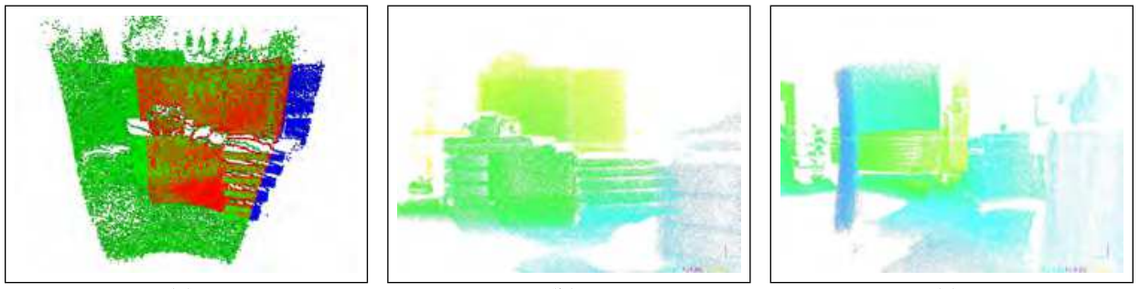


Fig. 15. The resulting *virtual scan* of the scene is compared with the base laser scan. The base laser scan is illustrated by the dashed blue line. The red line illustrates the virtual laser scan. The chair shows only a few points in the base laser scan since only the legs of the chair are in the scan plane, whereas the *virtual scan* outlines the contour of the chair.



(a) (b) (c)

Fig. 16. Visualization of the frustum culling in the ICP algorithm (a) and two views on a constructed 3D model (b+c).

7.4 Gaze Control

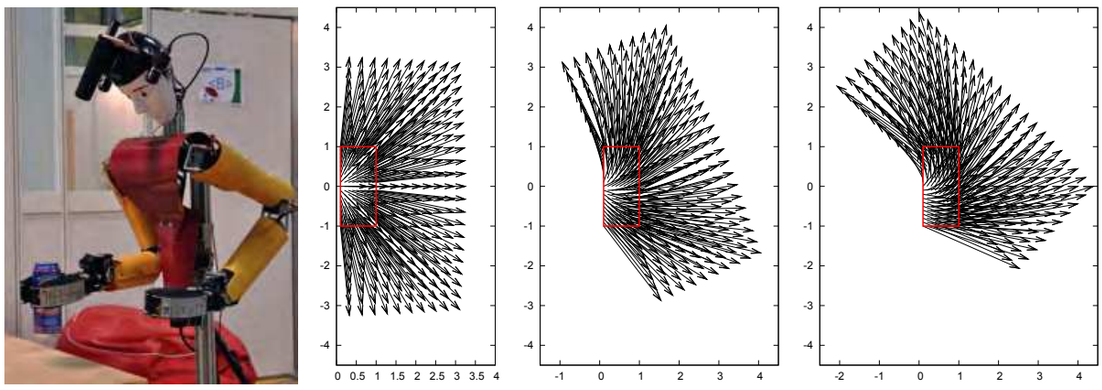
As already mentioned, the narrow field of view is a major issue in the context of collision avoidance as not all obstacles in the robot's vicinity can adequately be perceived. Here, we propose to mount the SwissRanger camera on some mechanical actuator, e.g., a pan-tilt unit, and to rotate the camera in a way that the robot perceives all obstacles in its movement direction. Rotating the camera is especially relevant when the robot is able to move sidewards or to turn fast.

In order to rotate the camera and control the viewing direction or *gaze*, we have mounted the SwissRanger on the head pan-tilt of an antropomorphic mobile service robot (see Figure 17(a)). As described in Section 7.2, the range measurements of the SwissRanger camera are transformed into the robot's coordinate frame and fused with the sensory information of a 2D laser scanner. In order to perceive all obstacles in the robot's vicinity and its movement direction, we determine an appropriate camera orientation for looking into and perceiving

all types of obstacles in the robot's movement direction which is given by the translational velocities $(\mathbf{v}^x \ \mathbf{v}^y)^T$ and the rotational velocity ω. We compute the camera's look at point $\mathbf{g} = (\mathbf{g}^x \ \mathbf{g}^y \ \mathbf{g}^z)^T$ using a simple controller.

$$\begin{pmatrix} \mathbf{g}^x \\ \mathbf{g}^y \\ \mathbf{g}^z \end{pmatrix} = \alpha \begin{pmatrix} \cos\beta\omega & -\sin\beta\omega & 0 \\ \sin\beta\omega & \cos\beta\omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(\frac{d_{\min}}{\|\mathbf{v}\|} + \gamma \right) \begin{pmatrix} \mathbf{v}^x \\ \mathbf{v}^y \\ 0 \end{pmatrix} \tag{16}$$

with $d_{\min}$ being the minimum distance (projected into the xy-plane) that can be perceived. It is typically limited by the minimum pitch/tilt angle of the actuator. The scaling factors α and β as well as the offset γ can be used to adjust the gaze vector according to a specific robot platform. We do not scale the function but use an offset of $\gamma \geq 1$ thereby preferring the perception of obstacles being farther away from the robot. That is, we assume that there is no single obstacle directly in front (touching) the robot.



(a) (b) $\omega = 0$ (c) $\omega = 0.125\pi\text{s}^{-1}$ (d) $\omega = 0.25\pi\text{s}^{-1}$

Fig. 17. The anthropomorphic service robot Dynamaid (a) and look at points $\mathbf{g}$ for different configurations of $\mathbf{v}^x$, $\mathbf{v}^y$ and ω.

The result of applying this simple controller yields a behavior of smoothly adjusting the orientation of the robot's head towards the current movement direction. Figure 17 shows the camera on top of the head of the antropomorphic robot as well as possible outcomes of the gaze controller described above. Although this behavior appears quite reasonable for a human spectator and allows for perceiving obstacles that would have not been perceivable with a fixed camera, it should be noted that elaborating a more sophisticated strategy for controlling the viewing direction is a matter of future work.

8. Conclusion and Future Work

With a focus on navigation in dynamic and cluttered domestic environments, we have presented a sensor setup for a 3D scanner that is especially appropriate for a fast 3D perception of those regions in the robot's vicinity that are relevant for collision avoidance. The 3D scanner is continuously pitched in a nodding-like fashion where the range of rotation angles (area of interest) is adapted in size just as the angular velocity with which the scanner is rotated. The acquired 3D data is projected into the xy-plane in which the robot is moving and used to construct and update egocentric 2.5D maps storing either the coordinates of closest obstacles or environmental structures. The representation of these maps is defined in a way that

they can be used with any algorithm for SLAM or collision avoidance that is operating on 2D laser range scans. The obstacle maps have been used in a set of simple behaviors for reactive collision avoidance. By means of the continuously pitching laser scanner and the concept of the obstacle maps, an autonomous mobile robot is able to successfully avoid different obstacles that can typically be found in domestic environments, like for instance table tops, open drawers or stairs. The structure maps have been used in an ICP-based SLAM approach to incrementally build two-dimensional point maps modeling the environmental structures of the robot's workspace and to localize the robot with three degrees of freedom. Furthermore, we segmented the continuously acquired 3D data stream into locally consistent 3D point clouds and used this information in the same SLAM approach to build three-dimensional point maps and to localize the robot with six degrees of freedom.
Current research and future work will focus on the application of recent Time-of-Flight cameras. Several extensions have been described that allow the application of these cameras in mapping tasks as well as for reactive collision avoidance by explicitly handling their narrow field-of-view as well as inaccuracies and erroneous measurements.

Acknowledgements

This work has been partially funded by the German Research Foundation (DFG) under contract number BE 2556/2-2 and the European Commission under contract numbers FP6-004381-MACS and FP7-247870-NIFTI. Furthermore, we would like to thank our current and former colleagues Christopher Lörken, Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, Kai Pervölz, Erich Rome, Jörg Stückler, Michael Schreiber, Matthias Nieuwenhuisen and Stefach Fuchs.

9. References

Allen, P., Stamos, I., Gueorguiev, A., Gold, E. & Blaer, P. (2001). AVENUE: Automated Site Modeling in Urban Environments, *Proceedings of the International Conference on 3D Digital Imaging and Modeling*, Quebec, Canada, pp. 357–364.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R. & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions, *Journal of the ACM (JACM)* **45**(6): 891–923.

Besl, P. J. & McKay, N. D. (1992). A Method for Registration of 3-D Shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(2): 239–256.

Chekhlov, D., Pupilli, M., Mayol-Cuevas, W. & Calway, A. (2006). Real-Time and Robust Monocular SLAM Using Predictive Multi-resolution Descriptors, *Proceedings of the 2nd International Symposium on Visual Computing*, Lake Tahoe, Nevada, USA, pp. 276–285.

Cole, D. & Newman, P. (2006). Using Laser Range Data for 3D SLAM in Outdoor Environments, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, USA, pp. 1556–1563.

Fuchs, S. & Hirzinger, G. (2008). Extrinsic and Depth Calibration of ToF-Cameras, *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska.

Grisetti, G., Rizzini, D. L., Stachniss, C., Olson, E. & Burgard, W. (2008). Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning, *Proceedings*

of the IEEE International Conference on Robotics and Automation (ICRA), Pasadena, California, USA, pp. 1880–1885.

Grisetti, G., Stachniss, C. & Burgard, W. (2007). Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters, *IEEE Transactions on Robotics* **23**(1): 34–46.

Haegele, M., Neugebauer, J. & Schraft, R. (2001). From Robots to Robot Assistants, *Proceedings of the 32nd International Symposium on Robotics (ISR)*, Seoul, South Korea, pp. 404–409.

Hähnel, D., Schulz, D. & Burgard, W. (2002). Map building with mobile robots in populated environments, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, Lausanne, Switzerland, pp. 496–501.

Holz, D., Lörken, C. & Surmann, H. (2008). Continuous 3D Sensing for Navigation and SLAM in Cluttered and Dynamic Environments, *Proceedings of the International Conference on Information Fusion (FUSION)*, Cologne, Germany, pp. 1469–1475.

Huber, D., Carmichael, O. & Hebert, M. (2000). 3-D Map Reconstruction from Range Data, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, California, USA, pp. 891–897.

Lange, R. (2000). *3D time-of-flight distance measurement with custom solid-state image sensors in CMOS/CCD-technology*, Phd thesis, University Siegen.

Leonard, J. & Feder, H. (1999). A computationally efficient method for large-scale concurrent mapping and localization, *in* D. K. J. Hollerbach (ed.), *International Symposium on Robotics Research*, Snowbird, Utah, USA.

Lingemann, K., Nüchter, A., Hertzberg, J. & Surmann, H. (2005a). About the Control of High Speed Mobile Indoor Robots, *Proceedings of the Second European Conference on Mobile Robots (ECMR)*, Ancona, Italy, pp. 218–223.

Lingemann, K., Nüchter, A., Hertzberg, J. & Surmann, H. (2005b). High-Speed Laser Localization for Mobile Robots, *Robotics and Autonomous Systems* **51**(4): 275–296.

Lorusso, A., Eggert, D. W. & Fisher, R. B. (1995). A Comparison of Four Algorithms for estimating 3-D Rigid Transformations, *Proceedings of the British conference on Machine vision BMVC*, pp. 237–246.

Luck, J., Little, C. & Hoff, W. (2000). Registration of Range Data Using a Hybrid Simulated Annealing and Iterative Clostest Point Algorithm, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, California, USA, pp. 3739–3744.

Maurelli, F., Droeschel, D., Wisspeintner, T., May, S. & Surmann, H. (2009). A 3d laser scanner system for autonomous vehicle navigation, *Proceedings of the 14th International Conference on Advanced Robotics (ICAR)*, Munich, Germany.

May, S., Droeschel, D., Holz, D., Fuchs, S. & Nüchter, A. (2009). Robust 3D-Mapping with Time-of-Flight Cameras, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, Missouri, USA. To appear.

Nüchter, A., Lingemann, K., Hertzberg, J. & Surmann, H. (2007). 6D SLAM – 3D Mapping Outdoor Environments, *Journal of Field Robotics (JFR), Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems* **24**(8–9): 699–722.

Nüchter, A., Lingemann, K., Hertzberg, J., Wulf, O., Wagner, B. & Surmann, H. (2005). 3D Mapping with Semantic Knowledge, *Proceedings of the RoboCup International Symposium 2005: Robot Soccer World Cup IX*, Osaka, Japan, pp. 335–346.

Olson, E., Leonard, J. & Teller, S. (2006). Fast Iterative Alignment of Pose Graphs with Poor Estimates, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, USA, pp. 2262–2269.

Pollack, M. E., Engberg, S., Matthews, J. T., Thrun, S., Brown, L., Colbry, D., Orosz, C., Peintner, B., Ramakrishnan, S., Dunbar-Jacob, J., McCarthy, C., M. Montemerlo, J. P. & Roy, N. (2002). Pearl: A Mobile Robotic Assistant for the Elderly, *Proceedings of the AAAI Workshop "Automation as Caregiver: the Role of Intelligent Technology in Elder Care"*, Edmonton, Canada, pp. 85–92.

Rojo, J., Rojas, R., Gunnarsson, K., Simon, M., Wiesel, F., Ruff, F., Wolter, L., Zilly, F., Santrač, N., Ganjineh, T., Sarkohi, A., Ulbrich, F., Latotzky, D., Jankovic, B., Hohl, G., Wisspeintner, T., May, S., Pervölz, K., Nowak, W., Maurelli, F. & Dröschel, D. (2007). A 3d laser scanner system for autonomous vehicle navigation, *Team Descriptions Papers of the DARTPA Urban Challenge 2007*. Available online: `http://www.darpa.mil/GRANDCHALLENGE/TechPapers/Team_Berlin.pdf`.

Sequeira, V., Ng, K. C., Wolfart, E., Goncalves, J. G. & Hogg, D. C. (1998). Automated 3D reconstruction of interiors with multiple scan views, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Vol. 3641, pp. 106–117.

Sheh, R., Kadous, M. W. & Sammut, C. (2006). On building 3D maps using a Range Camera: Applications to Rescue Robotics, *Techreport*, ARC Centre of Excellence for Autonomous Systems, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia.

Siegwart, R., Arras, K. O., Bouabdallah, S., Burnier, D., Froidevaux, G., Greppin, X., Jensen, B., Lorotte, A., Mayor, L., Meisser, M., Philippsen, R., Piguet, R., Ramel, G., Terrien, G. & Tomatis, N. (2003). Robox at Expo.02: A large-scale installation of personal robots, *Robotics and Autonomous Systems* **42**(3-4): 203–222.

Stiene, S. & Hertzberg, J. (2009). Virtual Range Scan for Avoiding 3D Obstacles Using 2D Tools, *Proceedings of the International Conference on Advanced Robotics (ICAR)*, Munich, Germany.

Strand, M. & Dillmann, R. (2008). Using an attributed 2D-grid for next-best-view planning on 3D environment data for an autonomous robot, *Proceedings of the IEEE International Conference on Information and Automation (ICIA)*, Zhangjiajie, Hunan, China, pp. 314–319.

Surmann, H., Nüchter, A. & Hertzberg, J. (2003). An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments, *Journal Robotics and Autonomous Systems (JRAS)* **45**(3-4): 181–198.

Surmann, H. & Peters, L. (2001). *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, chapter MORIA - A Robot with Fuzzy Controlled Behaviour, pp. 343–365.

Swadzba, A., Liu, B., Penne, J., Jesorsky, O. & Kompe, R. (2007). A Comprehensive System for 3D Modeling from Range Images Acquired from a 3D ToF Sensor, *Proceedings of the International Conference on Computer Vision Systems (ICVS)*, Bielefeld, Germany.

Thrun, S., Burgard, W. & Fox, D. (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, California, USA, pp. 321–328.

Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z. & Durrant-Whyte, H. (2004). Simultaneous Localization and Mapping with Sparse Extended Information Filters, *International Journal of Robotics Research* **23**(7-8): 693–716.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P.,

Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A. & Mahoney, P. (2006). Stanley: The Robot that Won the DARPA Grand Challenge, *Journal of Field Robotics* **23**(9): 661–692.

Triebel, R., Pfaff, P. & Burgard, W. (2006). Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, pp. 2276–2282.

Urmson, C., Anhalt, J., Bae, H., Bagnell, J. A., Baker, C., Bittner, R. E., Brown, T., Clark, M. N., Darms, M., Demitrish, D., Dolan, J., Duggins, D., Ferguson, D., Galatali, T., Geyer, C. M., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T., Kolski, S., Likhachev, M., Litkouhi, B., Kelly, A., McNaughton, M., Miller, N., Nickolaou, J., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Sadekar, V., Salesky, B., Seo, Y.-W., Singh, S., Snider, J. M., Struble, J. C., Stentz, A., Taylor, M., Whittaker, W. L., Wolkowicki, Z., Zhang, W., & Ziglar, J. (2008). Autonomous driving in urban environments: Boss and the urban challenge, *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I* **25**(1): 425–466.

Wang, C.-C. (2004). Simultaneous localization, mapping and moving object tracking, *PhD Thesis CMU-RI-TR-04-23*, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Weingarten, J., Gruener, G. & Siegwart, R. (2004). A state-of-the-art 3D sensor for robot navigation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, pp. 2155–2160.

Wulf, O., Arras, K. O., Christensen, H. I. & Wagner, B. (2004). 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception, *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA)*, New Orleans, Louisiana, USA, pp. 4204– 4209.

Wulf, O., Lecking, D. & Wagner, B. (2006). Robust self-localization in industrial environments based on 3d ceiling structures, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, pp. 1530–1534.

Wulf, O. & Wagner, B. (2003). Fast 3D-Scanning Methods for Laser Measurement Systems, *Proceedings of the International Conference on Control Systems and Computer Science (CSCS14)*, Bucharest, Romania.

Yuan, F., Swadzba, A., Philippsen, R., Engin, O., Hanheide, M. & Wachsmuth, S. (2009). Laser-based Navigation Enhanced with 3D Time-of-Flight Data, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, pp. 2844–2850.

Zhao, H. & Shibasaki, R. (2001). Reconstructing textured cad model of urban environment using vehicle-borne laser range scanners and line cameras, *Proceedings of the Second International Workshop on Computer Vision Systems (ICVS)*, London, UK, pp. 284–297.

Zinßer, T., Schmidt, J. & Niemann, H. (2003). A Refined ICP Algorithm for Robust 3-D Correspondence Estimation, *Proceedings of the International Conference on Image Processing (ICIP)*, Barcelona, Spain, pp. 695–698.

Zivkovic, Z., Booij, O. & Kröse, B. (2007). From images to rooms, *Robotics and Autonomous Systems* **55**(5): 411–418.

Mobile Robots Navigation

Edited by Alejandra Barrera

ISBN 978-953-307-076-6
Hard cover, 666 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

Mobile robots navigation includes different interrelated activities: (i) perception, as obtaining and interpreting sensory information; (ii) exploration, as the strategy that guides the robot to select the next direction to go; (iii) mapping, involving the construction of a spatial representation by using the sensory information perceived; (iv) localization, as the strategy to estimate the robot position within the spatial map; (v) path planning, as the strategy to find a path towards a goal location being optimal or not; and (vi) path execution, where motor actions are determined and adapted to environmental changes. The book addresses those activities by integrating results from the research work of several authors all over the world. Research cases are documented in 32 chapters organized within 7 categories next described.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen