FIRST STEPS TOWARDS A ROBOTIC SYSTEM FOR FLEXIBLE VOLUMETRIC MAPPING OF INDOOR ENVIRONMENTS

Rudolph Triebel * **Barbara Frank** * **Jörg Meyer** *
Wolfram Burgard *

* *Department of Computer Science, University of Freiburg*

Abstract: The problem of building maps of the environment is one of the fundamental problems in mobile robotics. So far, the majority of research has focused on the problem of how to learn two-dimensional maps such as occupancy grids. Robots, however, operate in a three-dimensional world. Accordingly, robots that use tree-dimensional maps can be expected to be more reliable and robust than those relying on 2d maps. In this paper we describe a robotic system that is able to learn volumetric maps of the environment. The robot is equipped with a laser range scanner attached to a manipulator with four degrees of freedom. This allows the robot to scan into arbitrary directions and accurately explore its environment. We also describe the techniques used for 3d collision avoidance and path planning.

Keywords: 3d mapping, OBB-trees, 3d collision detection, exploration

1. INTRODUCTION

Whereas mobile robots act in the three-dimensional world, most of the research regarding spatial representations of the environment of mobile robots has focused on two-dimensional maps. The restriction to two-dimensional representations, however, is error-prone and has serious limitations. For example, the planning of paths can be incomplete if the three-dimensional world is mapped into two dimensions or even incorrect if not all obstacles are contained in the two-dimensional description. Additionally, two-dimensional representations do not support typical tasks like searching for objects. For example, without knowledge about the three-dimensional structure of a shelf, a robot cannot plan appropriate viewpoints to find an object in the shelf. Thus, two-dimensional maps are not sufficient in situations in which robots are deployed in real-world scenarios. On the other hand, 3d models of buildings (exterior and interior) and man-made objects are envisioned to be useful in a wide area of applications, which goes far beyond robotics, like architecture, emergency planning, visualization etc. In all of these application domains, there is a need for methods that can automatically construct 3d models.

The problem of constructing real-world 3d models has received considerable attention over the past few years. (Bajcsy *et al.*, 2000), (Hakim *et al.*, 1997), and (Rous *et al.*, 2000) reconstruct three-dimensional structures from camera images. Recently, several authors used 3d range scanners for the acquisition of volumetric models. For example, (Stamos and Leordeanu, 2003) construct 3d models by combining multiple views obtained with a 3d range scanner. (Sequeira *et al.*, 1999) present a system that automatically reconstructs textured 3d indoor environments with a laser range finder. (Thrun *et al.*, 2000) uses two 2d range scanners. The first is oriented horizontally whereas the second points towards the ceiling. By registering the horizontal scans the system generates accurate three-dimensional models. (Früh and Zakhor, 2001) generate photo-realistic 3d reconstructions from urban scenes by combining aerial images with textured 3d data acquired with a range scanner

Fig. 1. Robotic platform (left) and simulated robot including the configuration space of the manipulator (right).

and a camera mounted on a vehicle. Again, the alignment of scans is achieved by an accurate 2d registration. (Thrun *et al.*, 2003) used several range scanners to learn models of underground mines. Whereas the range scanners are fixed on these systems, recently several authors, e.g., (Hähnel *et al.*, 2003), (Nüchter *et al.*, 2003), used pan-tilt devices to allow additional scanning directions. In all the systems described above, the major focus lies on the acquisition of the volumetric data, their automatic registration as well as on their integration into a potentially simplified model. Some approaches also considered the problem of planning the next vantage point. In particular, (Pito, 1996) or (Roberts and Marshall, 1998) addressed this issue and applied it to the acquisition of objects on a turntable. None of the mentioned systems, however, includes techniques for 3d collision avoidance and path planning. Furthermore, the devices used to orient the scanners had at most two degrees of freedom. Accordingly, the systems described above provided only a small number of possible viewpoints.

In this paper we present a robotic system for flexible volumetric mapping of indoor environments (see Figure 1). The robot consists of an iRobot B21r platform equipped with a manipulator with four degrees of freedom that carries a SICK laser range finder. In contrast to using a pan-tilt-unit, this setup allows the robot to flexibly orient the scanner to different view points for generating 3d range scans. To allow the safe and reliable operation of the robot we developed a navigation system that can efficiently detect and avoid collisions in the three-dimensional data acquired by the robot and to explore given regions of interest. We describe the major components of the navigation system and present results regarding collision avoidance, path planning and exploration.

This paper is organized as follows: After describing the robotic platform in the following section we present an approach for efficient 3d collision detection and avoidance in Section 3. Then Section 4 is concerned with the path planning component of our navigation system. Finally, we present an algorithm for effective exploration of volumetric scenes in Section 5.

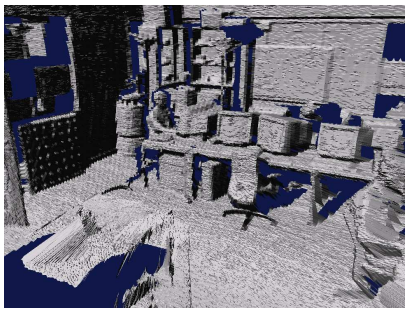

Fig. 2. Indoor environment scanned by the system.

2. THE ROBOT SYSTEM FOR VOLUMETRIC MAPPING

Our robot system for 3d mapping consists of a four DOF manipulator mounted on a B21R robot (see left image of Figure 1). The manipulator carries a SICK laser range finder (and a Sony FireWire camera which will be used for acquiring textured models in later phases of the project). The four joints are rotational and will be denoted j_0 through j_3 (from bottom to top) throughout this paper. Whereas the joints j_0 and j_2 rotate about the vertical Z-axis from $-180°$ to $180°$, j_1 and j_3 rotate about the horizontal X-axes from $-90°$ to $90°$. Thus, the configuration space $\mathcal{C}$ of the manipulator is given as:

$$\mathcal{C} = \left\{ \mathbf{j} \in \mathbb{R}^4 \;\middle|\; \begin{array}{l} -180 \leq j_0, j_2 \leq 180, \\ -90 \leq j_1, j_3 \leq 90 \end{array} \right\}$$

Due to the length of the connections between the joints, the individual configurations in $\mathcal{C}$ geometrically correspond to points on small half spheres S_s whose centers lie on the surface of a bigger half sphere S_b (see right image of Figure 1).

By moving the manipulator along paths in $\mathcal{C}$ and by subsequently mapping of the range measurements from $\mathcal{C}$ to $\mathbb{R}^3$ we obtain 3d range scans such as the one depicted in Figure 2.

3. 3D COLLISION AVOIDANCE

3.1 3D Collision Detection

Whenever a robot has to navigate in its environment, it must be able to generate paths that are collision-free. This, however, requires the capability to quickly check whether the robotic platform in its current configuration intersects with the obstacles given in the map. In our scenario, in which we use a mobile robot to automatically acquire three-dimensional maps from laser-range data, the models used for collision detection contain hundreds of thousands of polygons or even more. Therefore, it is of utmost importance to have an efficient approach for computing possible intersections of the robot's shape with objects in the map.

The key operation that has to be carried out to avoid collisions of the robot with obstacles is the test

whether there is an intersection of the robot's shape with polygons stored in the map. A fast and robust method has recently been proposed by (Gottschalk, 2000). In this approach the collision check is performed using *Oriented Bounded Boxes* and by organizing these in a tree-structure (*OBB-tree*). The tree is built from top to bottom for a given set of 3d polygons. Each inner node of the tree consists of a 3d oriented bounding box for a subset of the polygons. The bounding boxes are oriented along the principal directions of the polygon vertices. This way we obtain a tight fit of the bounding boxes to the polygons.

The main idea is that the overlap test for two oriented bounding boxes can be performed efficiently by projecting both boxes onto a line and checking the resulting line segments for overlap. As (Gottschalk, 2000) shows, only 15 different line directions need to be tested, namely the 6 principal directions of both boxes and the 9 mutual cross products of these.

For the collision check between the robot and a set of obstacle polygons, we build an OBB-tree both for the robot and for the obstacles. We begin by testing the root node boxes for overlap and then proceed in both trees until we reach a level where the boxes do not overlap or until we end up in a leaf node. In the latter case we need to test the 3d polygons for intersection.

3.2 3D Collision Avoidance

The 3d collision detection system is designed to efficiently check whether or not the robot has collided with an obstacle represented in the three-dimensional map of the environment. In practice, however, we want to *avoid* that the robot collides and stop all its actions before it comes into contact with an object. Accordingly, the robot must be able to predict a collision with an object in order to stop early enough or to choose alternative actions that prevent it from colliding with an object.

One way to achieve this is to run a forward simulation at every step in time: Given the current speed vectors of the robot we calculate all possible collisions for the next time-step(s). This, however, can be time consuming, especially if potential motion changes must be taken into account.

The OBB-tree approach fortunately provides us with a convenient way of avoiding collisions. First we apply an approach that is also known as obstacle growing. In our system this is achieved by enlarging the bounding boxes in the tree by a constant factor. Additionally we can avoid collisions by pruning the OBB tree at a certain level. For example, if we skip the polygon intersection test at the leaf nodes and just rely on the test on the level above, we become more conservative with respect to collision checks, since the overlap of the upper-level bounding boxes is a sufficient condition for an intersection at a lower level.

The left image of Figure 3.2 shows a situation in which our robot is close to an obstacle. Although none of the polygons representing the robot intersects with an object in the scene, the collision avoidance strategy based on obstacle growing and OBB tree pruning already reports a potential collision. The right image of Figure 3.2 shows in red/dark grey the enlarged OBB intersecting with polygons of the map. Here, the laser attached to the manipulator gets too close to the wall.

3.3 Implementation Details and Efficiency

The collision avoidance system based on the OBB-tree approach has been implemented and tested in simulation using real-world data. The typical example is shown in Figure 3. This particular scene consists of over $750,000$ scan points and about $750,000$ triangles. The triangles were obtained by connecting neighboring points in the point set. The resulting OBB-tree for this scene had a depth of 41, and it took about 12 minutes to build the tree on a 2.8 GHz Pentium 4 machine. For this scene we are able to perform 10 collision tests per second. For a typical motion speed of the manipulator of 5 degrees per second this amounts to two tests per degree of motion, which is sufficiently fast for typical manipulation tasks and corresponding slow movements of the platform.

4. PATH PLANNING

To scan its environment the robot must be able to efficiently plan collision-free paths. Whenever the platform moves, we currently reset the manipulator to the upright position so that the robot's movements can be planned in its three-dimensional configuration space. In our current system we assume that the platform stops whenever the robot scans its environment by moving the scanner with its manipulator. The fact that the configuration space of the manipulator has four dimensions makes the application of standard path planning strategies such as A^* infeasible for planning trajectories. Assuming a grid representation of the 4d configuration space, each grid cell would have $3^4 - 1 = 80$ neighbors, resulting in a branching factor of 80 in each planning step. Even for short plans with about 10 plan steps this would result in an untreatable number of nodes in the search tree.

To reduce the size of the search space we exploit the particular geometry of the manipulator. Considering that small variations in the lower joints j_0 and j_1 result in larger volumes claimed by the manipulator than small variations in the upper joints j_2 and j_3, we use a projection $\mathcal{C}^\perp$ of $\mathcal{C}$ onto the first two dimensions. [1] All obstacles are then mapped from 3d Cartesian space into $\mathcal{C}^\perp$. This mapping results in a partition of $\mathcal{C}^\perp$ into

[1] This can be interpreted as a *p-level tree* (Latombe, 1991), a data structure to efficiently store the configuration space (here, $p = 2$).

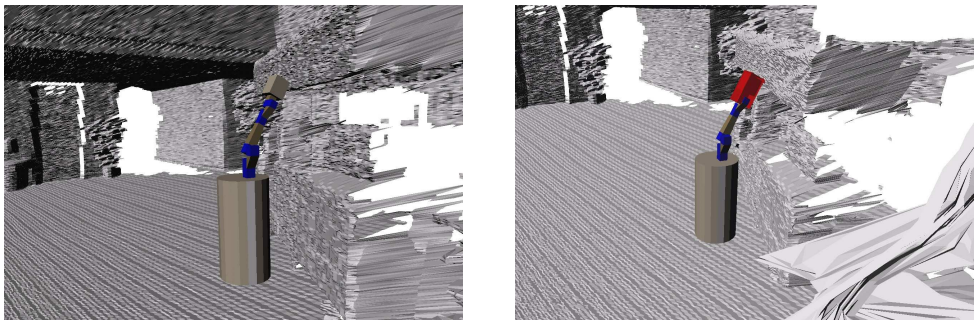

Fig. 3. Collision avoidance using a conservative extension of OBB trees as well as obstacle growing: Situation in which the laser attached to the manipulator gets very close to an obstacle (left) and detection of a potential collison of the range scanner (right image).

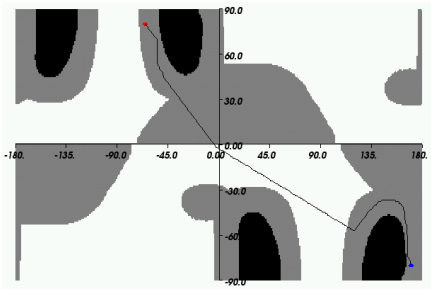

Fig. 4. Projection of nearby obstacles onto $\mathcal{C}^{\perp}$; axes represent the position of j_0 and j_1 in degrees.

three different regions: Areas where there is no collision, independent of the configuration of j_2 and j_3, are considered as free and therefore can safely be attained. Areas where a collision is unavoidable for any setting of j_2 and j_3 can obviously never be attained and therefore are considered as occupied. In the remaining areas a collision depends on the position of j_2 and j_3. Such areas in $\mathcal{C}^{\perp}$ are considered as "dangerous." Figure 4 shows for a simulated scene the projection of the obstacles in the vicinity of the robot into $\mathcal{C}^{\perp}$. The white region corresponds to safe configurations, the black areas are the occupied configurations, and the dangerous regions are depicted in grey.

The path planning now works as follows: Whenever the start and goal configurations both are safe and are in the same connected component, we apply 2d A^* planning. In any other case 2d A^* is not a complete planning strategy. Therefore, we apply another technique described in (Latombe, 1991) called *slicing*: We consider the configuration space $\bar{\mathcal{C}}^{\perp}$ where j_3 is held fixed at position 0 and j_2 is arbitrary. This way the volume claimed by the manipulator is smaller and, thus, the free space is larger than in $\mathcal{C}^{\perp}$. That means that there is a higher chance that 2d A^* planning in $\bar{\mathcal{C}}^{\perp}$ finds a free path. However, we need to check if there is a free path from the start position (s_0, s_1, s_2, s_3) to the position $(s_0, s_1, s_2, 0)$ – and analogously for the goal position. Furthermore, this technique is still not complete. Thus, if this strategy also fails, we have to plan in the whole 4d space. We do this using a probabilistic roadmap approach (PRM) as described in (Kavraki *et al.*, 1996).

The above planning algorithm has been implemented and tested extensively. A statistical analysis has shown that the PRM-technique needs to be applied only in 6% of all cases. Figure 5 depicts an example path generated by our planner. The starting position is $(-65, 80, -40, 20)$ and the goal position $(170, -80, 40, -20)$, both measured in degrees for all four joints. The projection of the resulting path into $\mathcal{C}^{\perp}$ is shown in Figure 4.

5. EXPLORATION

One of the tasks of our robot is the autonomous exploration of its environment. In the case of our robotic system, we can split up the exploration problem into two distinct subtasks, namely the *local* exploration, where only the manipulator is moved and the robot has a fixed position, and the *global* exploration, where good vantage points for the whole system are demanded as e.g. in (Klein and Sequeira, 2000). The distinction here makes sense, because in the local exploration there is no need for localisation and registration algorithms, due to the high accuracy of the manipulator joints.

In this paper we will focus on the local exploration. This means that the environment which is explored, is restricted to a 3d area B in the vicinity of the manipulator, because far away regions can not be scanned accurately enough and only nearby occlusions can be resolved. In theory, B can be defined arbitrarily, but we will consider it as an axis-aligned box in front of the manipulator. The information that is acquired during the exploration process is represented in a 3d occupancy grid inside B. In the remainder, B is also denoted the *grid box*. Now, we define the local exploration task as follows: For our given 3d rectangular region B we search for a set of sensor paths along which the acquired sensor information is maximized while the overall path cost is minimized.

A frequently used measure for the information provided by a measurement is the *information gain* I. The information gain of a single measurement for a particular cell c_l in B is the difference of the entropies

Fig. 5. Path of the manipulator generated by the path planner in a simulated environment.

of that cell before and after incorporating the new sensor information z:

$$I(c_l \mid z) = H(c_l) - H(c_l \mid z) \tag{1}$$

Based on the information gain and an appropriate path cost function f, we can evaluate possible paths P by calculating the weighted difference between the information gain and $f(P)$ (see also (Stachniss and Burgard, 2003)). A typical problem in this context is that the information gain cannot be calculated in advance as one does not know which measurement will be received along P. The usual solution is to compute the *expected information gain* by integrating over all possible measurements. In our case, however, this is infeasible, since the number of possible measurements grows exponentially with the number of time steps or with the length of P.

In our current system we approximate the expected information gain by considering the most likely measurement $\bar{z}$ for each beam. This measurement is determined by traversing the grid in B along each beam until a cell with probability higher than a given constant $\bar{c}$, which is set to .5 in our current system, is reached. The expected information gain of a particular path is then computed as:

$$I(P) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \sum_{c \in R(\bar{z}_{m,n})} I(c \mid \bar{z}_{m,n}), \tag{2}$$

where N is the number of discrete points along P at which a range scan is obtained, M is the number of beams of each scan, and $R(\bar{z}_{m,n})$ is the set of all cells intercepted by the beam with length $\bar{z}_{m,n}$.

5.1 The utility of a path

For a good path evaluation we need a path cost function f that penalizes dangerous paths. In our implementation we define f as the inverse distance to the next object. We approximate this value by creating a set of k sample points $\{\mathbf{s}_0, \ldots, \mathbf{s}_k\}$ inside the volume of the manipulator and determining the sample with minimum distance to the set $\mathcal{B}$ of bounding boxes representing the environment. In the simplest case, where there is no obstacle other than the ones inside $\hat{B}$, $\mathcal{B}$ only consists of $\hat{B}$. We will consider this case, but we note that the definition of f can easily be extended to the general case if we take into account that the environment is given as an OBB-tree. Thus, we have:

$$f(P) = \text{argmin}_{i=1,\ldots,k}\{d(\mathbf{s}_i, \hat{B})^{-1}\}. \tag{3}$$

The distance $d(\mathbf{s}_i, \hat{B})$ from a sample $\mathbf{s}_i$ to the box $\hat{B}$ is efficiently calculated using the generalized voronoi diagram (GVD) of $\hat{B}$ (Lin, 1993): After creating the GVD, we only need to check into which voronoi region $\mathbf{s}_i$ falls. Then, $d(\mathbf{s}_i, \hat{B})$ is given as the distance to the box feature (face, edge or vertex) that corresponds to the found voronoi region.

Using equations (2) and (3), we define the best exploration path P^* as the one that maximizes the utility:

$$P^* = \text{argmax}_{P \in \mathcal{P}}\{I(P) - \lambda f(P)\} \tag{4}$$

where λ is a fixed weighing factor and $\mathcal{P}$ is the set of all possible paths.

5.2 The exploration algorithm

The major problem in evaluating equation (4) is that $\mathcal{P}$ cannot be determined efficiently. In our current system, we consider only a small subset $\mathcal{S}$ of $\mathcal{P}$, which contains all paths that are reachable by the manipulator and at the same time include good view points. To determine a set of good view points, we consider the bounding box $\hat{B}$ of all scan points inside B after the first scan. $\hat{B}$ defines the region of interest for the further exploration process. In order to get a good coverage of the object(s) inside $\hat{B}$, we look for paths so that the laser sweeps across the edges $\{e_1, \ldots, e_{12}\}$ of $\hat{B}$. The start and end points of such paths can be determined by finding joint positions at which one of the e_i is inside the laser plane. Geometrically, this corresponds to points at which a plane passing through a given e_i is tangent to a given small sphere S_b. Thus, the set $\mathcal{S}$ of good view paths is constructed as follows:

First, we create a set of points on the surface of S_b so that no point is nearer to $\hat{B}$ than the maximum distance of the laser to the rotation axis of joint j_3. This way we ensure that the laser does not collide with $\hat{B}$. For each of these points we determine all points on the corresponding upper sphere S_s that lie on a plane tangent to S_s and passing through an e_i as described.

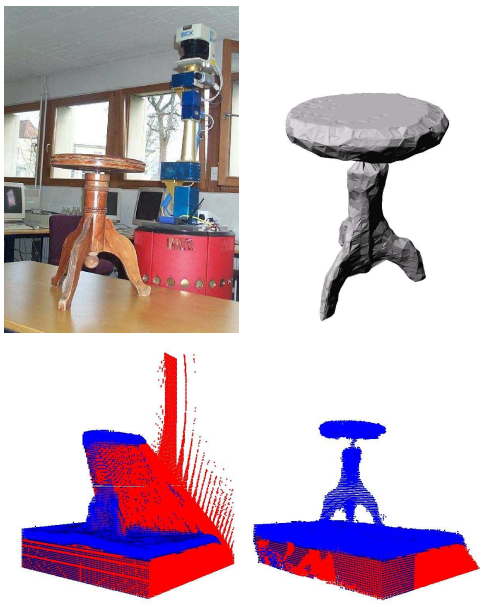

Fig. 6. upper-left: experimental setup; lower line: grid box after the first scan (left) and at the end(right), red cells are unknown, blue cells have high occupancy probability; upper-right: triangulation of the final 3d model.

We obtain a set of *vantage points* $\mathbf{v}_j$ together with the corresponding observation edges $e_{i(j)}$. Those vantage points, for which the observation edges are parallel, are then connected to *sub-paths*. These sub-paths correspond to different sweeping motions of the scanner along the faces of $\hat{B}$.

Given this set $\mathcal{S}$ of sub-paths we proceed as follows: Out of a set of h sub-paths that are nearest to the current arm position we select the one with maximum utility. The value h will be denoted as the *exploration horizon*. If the obtained utility is lower than a given *minimum utility* u_{min}, the sub-path is omitted and the next h sub-paths are considered. If a sub-path is found, it is executed and the newly gathered information is incorporated into the occupancy grid. The algorithm terminates if there is no sub-path left.

5.3 Implementation

We have tested this exploration strategy with a real 3d object – a piano stool (see fig. 6). We chose an exploration horizon of 10 paths. Initially, there were 424 sub-paths, out of which only 16 were executed (the maximum would have been 42). The results are shown in fig. 6. For the triangulation, we created an α-shape (Bernardini and Bajaj, 1997) from all scan points that fell into occupied regions in the final grid.

6. CONCLUSIONS

In this paper we presented a robotic system for acquiring three-dimensional maps of indoor environments. The robot is a B21r platform equipped with a manipulator that carries a SICK laser range scanner. To control this robot we developed a software system that includes techniques for 3d collision avoidance, path planning and exploration. The techniques have been implemented and evaluated using real-world data and in simulation.

There are several directions for future research. First, the generation of the OBB-tree is computationally demanding so that techniques to efficiently update such a tree based on sensory input are demanded. Furthermore, the path planning and exploration system should also include movements of the platform itself and not only movements of the robotic arm. This will additionally increase the complexity of the search and further techniques for increasing the efficiency will have to be developed.

REFERENCES

Bajcsy, R., G. Kamberova and L. Nocera (2000). 3D reconstruction of environments for virtual reconstruction. In: *Proc. of the 4th IEEE Workshop on Applications of Computer Vision*.

Bernardini, F. and C. Bajaj (1997). Sampling and reconstructing manifolds using alpha-shapes. Technical report. Departement of Computer Sciences, Purdue University.

Fr¨uh, C. and A. Zakhor (2001). 3D model generation for cities using aerial photographs and ground level laser scans. In: *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gottschalk, Stefan (2000). Collision Queries using Oriented Bounding Boxes. PhD thesis. University of North Carolina, Chapel Hill.

H¨ahnel, D., W. Burgard and S. Thrun (2003). Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems* **44**, 15–27.

Hakim, S.E., P. Boulanger and F. Blais (1997). A mobile system for indoors 3-d mapping and positioning. In: *Proc. of the 4th Conference on Optical 3-D Measurement Techniques*.

Kavraki, L., P. Svestka, J. C. Latombe and M. Overmars (1996). Probabilistic road maps for path planning in high-dimensional confi guration spaces. *IEEE Transactions on Robotics and Automation* pp. 566–580.

Klein, K. and V. Sequeira (2000). The view-cube: An effi cient method of view planning for 3d modelling from range data. In: *Proceedings Fifth IEEE Workshop on Applications of Computer Vision (WACV 2000)*. Palm Springs, CA.

Latombe, J.C. (1991). *Robot motion planning*. Kluwer.

Lin, Ming C. (1993). Effi cient Collision Detection for Animation and Robotics. PhD thesis. Department of Electrical Engineering and Computer Science, University of California, Berkely.

N¨uchter, A., H. Surmann and J. Hertzberg (2003). Planning robot motion for 3d digitalization of indoor environments. In: *Proc. of the 11th International Conference on Advanced Robotics (ICAR)*.

Pito, R. (1996). A sensor based solution to the next best view problem. In: *IAPR Int. Conf. Pattern Recognition, ICPR'96*. Vienna, Austria.

Roberts, D. and A. Marshall (1998). Viewpoint selection for complete surface coverage of three dimensional objects. In: *Proc. of the Britsh Machine Vision Conference*.

Rous, M., A. Matsikis, F. Broicher and K.-F. Kraiss (2000). Erzeugung eines planaren 3D-Modells aus Kameraaufnahmen zur Anwendung in der mobilen Robotik. In: *Proc. Fachgespräche Autonome Mobile Systeme*. In German.

Sequeira, V, K. C. Ng, E. Wolfart, J.G.M Gonalves and D.C. Hogg (1999). Automated 3d reconstruction of interiors with multiple scan-views. In: *Proceedings of SPIE, Electronic Imaging*. San Jose Convention Center, San Jose, California, USA.

Stachniss, C. and W. Burgard (2003). Exploring unknown environments with mobile robots using coverage maps. In: *Proc. of the International Conference on Artificial Intelligence (IJCAI)*.

Stamos, I. and M. Leordeanu (2003). Automated feature-based range registration of urban scenes of large scale. In: *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thrun, S., D. H¨ahnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer and W. Whittaker (2003). A system for volumetric robotic mapping of abandoned mines. In: *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*.

Thrun, S., W. Burgard and D. Fox (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In: *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*.