A Kinect-based Front-end for Graph-SLAM Using Plane Matching in Planar Indoor Environments

Zehui Yuan, Stefano Rosa, Ludovico Russo, Basilio Bona

Dipartimento di Automatica e Informatica, Politecnico di Torino, Italy
{<name>.<surname>}@polito.it

Abstract. We present a pose graph optimization approach which enables a mobile robot to create 3D maps of planar indoor environments using a Microsoft Kinect camera. Rather than using point features, the approach relies on the registration of planar surfaces. Plane matching is used to implement a (*front-end*) for the construction of a pose graph, which is then optimized by a state-of-the-art (*back-end*). Vertical planes are extracted from acquired point clouds associated to the poses of the robot; then a plane matching algorithm is used to create constraints among successive robot poses. Place revisiting episodes are detected using 3D features in order to provide loop closing constraints. These constraints provide the input for a pose graph optimization algorithm, which computes an estimate of the robot trajectory. Finally, the 3D map is created by attaching to each pose of the trajectory the corresponding planes. Planar surfaces are more robust and descriptive with respect to point features and provide an accurate estimate of rotations. Moreover, the front-end combines geometric and appearance-based information to filter out outliers and perform robust plane association. Preliminary experimental results in real environments show that the approach is able to create 3D maps which are consistent and close to reality.

Keywords: SLAM, RGB-D, Plane matching

1 Introduction

The problem of 3D Simultaneous Localization and Mapping (SLAM) [1, 2, 3, 4] has attracted a conspicuous attention from the robotics community in the latest years. For mobile robots, the knowledge of a map containing a rich description of the environment is essential for navigation, surveying tasks, and manipulation, especially when the robot has to operate in 3D scenarios. In contrast to planar occupancy grid maps, 3D maps include more detailed information about the environment and are becoming more and more popular due to the availability of new kinds of 3D sensors. Typically, laser range finders and depth cameras were used for 3D SLAM approaches in order to acquire dense point clouds [5, 6]. Recently, Microsoft Kinect [7] has dominated the stage of 3D robotic sensing, as a low-cost, low-power sensor, that is able to acquire color and depth images with an high frame rate.

In principle, a robot equipped with a 3D camera and wheel encoders would be able to create a 3D dense map of the environment by attaching the point clouds to the corresponding poses estimated from wheel odometry, but this strategy suffers from error accumulation [8]. A well assessed strategy for avoiding unbounded error accumulation is the so called *pose graph optimization*, which is an optimization-based SLAM approach [9, 10]. In pose graph optimization, the poses assumed by a mobile robot at successive time steps are modeled as nodes in a graph (the *pose graph*), while edges between nodes represent inter-nodal measurements. For instance, odometric measurements are modeled as edges (or *constraints*) connecting consecutive nodes, while *loop closing edges* connect arbitrary nodes' pairs and model place revisiting episodes. Nonlinear optimization algorithms (e.g., [11]) can then be then used to find the nodes' poses that maximize the likelihood of the inter-nodal measurements, hence obtaining an accurate estimate of the poses assumed by the robot.

A standard approach for retrieving inter-nodal constraints is based on the Iterative Closest Point (ICP) algorithm [12], which is widely used for scan matching [13, 6, 5]. In visual SLAM, most approaches rely on the extraction and matching of sparse 2D visual features, like SIFT [14], SURF [15], and ORB [16]. The features are first extracted from the 2D image and then projected in 3D using the depth image. From these 3D features, the relative transformations between frames are computed using vector registration algorithms. However, due to the fact that the depth map is digitalized, the depth data is often imprecise or incorrect. Moreover, features often lie at the border of objects; this translates to depth estimation errors. Another crucial problem is that no visual feature provides perfect robustness. Usually, after feature matching, RANdom SAmple Consensus (RANSAC) [17] is used to find a subset of feature pairs and to estimate their correspondences. Recently 3D feature descriptors have been presented, like Normal Aligned Radial Feature (NARF) descriptor [18] and Fast Point Feature Histogram (FPFH) [19].

In indoor environments, several structures like doors, walls, tables, ground floor, etc., can be modeled as planar surface patches, which are parallel or perpendicular to each other. Therefore, planar patches have been found to be a good feature for 3D visual SLAM, while also being a good representation for the final 3D map. Approaches to 3D environment mapping using planar features have been proposed in [4, 20, 21, 22]. In those works planar surfaces are matched based on geometric constraints in order to estimate their relative roto-translation. It should be noted that in those works planes are obtained using laser range finders, which provide higher accuracy, larger field of view, and longer range compared to RGB-D (depth) cameras.

In this paper we propose an approach for 3D mapping in planar indoor environments using a Kinect sensor. The approach is based on the alignment of vertical planar surfaces and corners, which are dominant features in indoor areas. The idea is (i) to extract planes and corners from raw point clouds, (ii) to recognize corresponding planes in different point clouds, (iii) to retrieve the relative geometric constraints between frames by using a plane matching algorithm, and (iv) to construct a pose graph with these constraints and feed it to a pose graph optimizer algorithm to obtain a 3D map. The core of the paper is the creation of a plane-based front-end for a graph-SLAM algorithm. The front-end combines geometric constraints, appearance-based information and the Viewpoint Feature Histograms (VFH) [23] to compute robust and accurate inter-nodal constraints. Experimental results show that the algorithm is able to find a good estimate of the 2D trajectory of the robot, and the 3D map obtained from the estimated trajectory is a faithful representation of the real environment.

The plane-based 3D mapping approach is presented in Section 2. After briefly discussing the plane extraction phase in Section 2.1, a new algorithm for finding correct correspondences is presented in Section 2.2. Section 2.3 describes an approach for loop closure detection using VFH features, and Section 2.4 comments on the pose graph optimization phase. Experimental results are discussed in Section 3. Conclusions and future work are reported in Section 4.

2 Plane-based Trajectory Estimation

2.1 Plane Extraction

This section describes the extraction of planes from each point cloud $\mathcal{P}_t$ acquired at time t. In this work we only consider vertical or nearly vertical planes. The motivation for this choice is twofold: first, since we are addressing a planar navigation problem, horizontal planes do not provide meaningful constraints on robot pose; second, in an indoor environment large planar patches are most likely to be walls, pillars, or other static structures, then the approach is expected to be more resilient to the presence of dynamic objects.

At each time instant we sample a color image and a corresponding point cloud. Acquired point clouds suffer from different noise and error sources. So we apply a passthrough filter on distance is applied to discard points which are too far and have low accuracy. Downsampling is also applied in order to reduce processing time.

The plane extraction approach is based on the well known RANSAC [17] method for robust model fitting. RANSAC is iteratively executed to extract the largest plane from the point cloud until a pre-defined ending condition is met. For each iteration of the algorithm, the plane with the largest number of inliers is returned. We define the extracted plane as $P_{t,i}$, where t is the index of point cloud, and i is the index of extracted plane in point cloud $\mathcal{P}_t$. The overall set of vertical plane extracted at time t is called P_t^v. In order to consider only planes that are roughly vertical, we compute the angle between its normal vector $\boldsymbol{n}$ and the $\mathbf{z}_{\boldsymbol{r}}$ axis of the robot reference frame, which points upwards. A plane which meets the following equation

$$|\boldsymbol{n}^T \cdot \mathbf{z}_{\boldsymbol{r}}| < \cos\left(\frac{\pi}{2} - \phi\right) \tag{1}$$

is considered a vertical plane, where ϕ is the maximum acceptable deviation from the $\mathbf{z}_{\boldsymbol{r}}$ axis.

If the plane satisfies condition (1), we perform a distance-based clustering on its points to find large contiguous regions of points in the plane and each cluster with a sufficient number of points is saved as a new plane, while the remaining are discarded. With this process we remove noisy points or small clusters that fit to the extracted plane but are not part of a large contiguous surface (e.g., a door frame leaning out from the surrounding wall), and separate multiple surfaces that are co-planar but physically belong to different objects, such as two tables at same height. The points corresponding to the preprocessed planes are stored in the set P_t^v (together with the planes parameters) and removed from the point cloud.

Due to noise and occlusions, a plane could be split into several planar patches (over-segmentation). Therefore, we compare the vertical planes in P_t^v and we merge planes that are coplanar and whose support points are close enough. After all the extracted planes have been refined, small planar surfaces having area less than a pre-defined threshold are ignored.

After computing the set P_t^v, we are interested in understanding how the planes in the set relate with each other: for instance, a pair of orthogonal planes defines a corner, which is more robust and distinguishable in indoor environment. Moreover, a single corner is enough to lock all degrees of freedom in space, since we can associate to a corner both an orientation and a position, while a plane only constraints a distance and an orientation. Two vertical planes $P_{t,i}$ and $P_{t,j}$ define a corner if the two conditions are met:

1. they are perpendicular to each other, i.e., ${\boldsymbol{n}_i}^T \cdot \boldsymbol{n}_j \simeq 0$.
2. their intersection points $\boldsymbol{p}_C$ lies (up to some tolerance D_{CB}) inside the boundary of each of them. Mathematically, this can be expressed as

$$d(\boldsymbol{p}_C, \boldsymbol{P}_B) \leq D_{CB} \tag{2}$$

where $\boldsymbol{P}_B$ is the boundary of the vertical plane.

2.2 Plane Matching

We define the robot frame $\mathcal{F}_t$ as the pose of the robot at time t, identified by three axes: $\mathbf{z}_{\boldsymbol{r}}$ (already introduced in the previous section), which is perpendicular to the plane in which the robot moves, $\mathbf{y}_{\boldsymbol{r}}$ heading towards the direction of motion of the robot, and $\mathbf{x}_{\boldsymbol{r}}$, completing the tern.

We consider the set P_t^v of planes observed from frame $\mathcal{F}_t$ and containing N_t planes. We want to establish correspondences between planes in P_t^v and planes in P_{t-1}^v, i.e., which planes' pair $(P_{t,i}, P_{t-1,j})$ represent the same physical location.

On the assumption that the floor is flat, and considering vertical planar patches only, it is possible to project them onto the $\mathbf{x}_{\boldsymbol{r}}\mathbf{y}_{\boldsymbol{r}}$-plane and use their projections, i.e., 2D lines, to represent the planes. We parametrize the 2D lines in terms of distance d from the robot and the relative angle $\theta \in (-\pi, +\pi]$ (both expressed in the robot frame). The orientation θ of the projected plane is defined to be the angle between $\mathbf{y}_{\boldsymbol{r}}$ axis and the projection line. The projection line is oriented in clockwise direction such that the robot is always on the right-hand side of the plane. The advantage is that the robot can distinguish from which side a plane is observed.

In our work, the orientation angle θ, combined with distance d of the extracted planes are used together to estimate correspondences. Moreover their color histograms are compared to make correspondences more robust and to disambiguate planes which have similar values for θ and d. Odometry information is used as a constraint for plane matching. The use of different tests is important in order to discard most of the false correspondences, which are unavoidable in practice. For each pair $(P_{t,i}, P_{t-1,j})$, with $P_{t,i} \in P_t^v$ and $P_{t-1,j} \in P_{t-1}^v$ we apply the following tests.

Odometry rotation agreement test Since odometry is available, we can use it to choose, among the candidate matches, the ones that meet the rotation agreement with the odometry values. Given odometry information, we can easily compute the odometric rotation matrix $\boldsymbol{R}_{t-1}^t$ between frame $\mathcal{F}_{t-1}$ and $\mathcal{F}_t$. The plane in current frame $\mathcal{F}_t$ is transformed into its previous reference frame $\mathcal{F}_{t-1}$, then its parameters $(\hat{\theta}_i, \hat{d}_i)$ in reference frame $\mathcal{F}_{t-1}$ are computed. Then, we look for all planes in P_{t-1}^v satisfying

$$\|\hat{\theta}_i - \theta_j\| \leq \Delta\theta \tag{3}$$

where θ_j is the orientation of the j-th plane in P_{t-1}^v and $\Delta\theta$ is a fixed threshold. If the orientation θ_j of the plane is roughly equal to $\hat{\theta}_i$ (expressed in frame $\mathcal{F}_{t-1}$), i.e. Eq. (3) is satisfied, then the j-th plane of P_{t-1}^v is selected as a candidate match for the i-th plane in the set P_t^v.

Odometry translation agreement test After the first test, a subset of candidate planes corresponding to the query plane $P_{t,i}$ is obtained. Similarly to the previous test, we can then check if the candidate matches are in agreement with the translation information provided by odometry. Odometry provides the the translation $\boldsymbol{T}_{t-1}^t$. Then, we can use it to select, among the candidate plane matches, the ones that are in agreement with the odometry translation values. More formally, the candidate pairs pass the odometry translation agreement test if they satisfy the following inequality:

$$\|\hat{d}_i - d_j\| \leq \Delta d \tag{4}$$

where $\hat{d}_i$ and d_j are distances of plane $\hat{P}_{t,i}$ (expressed in frame $\mathcal{F}_{t-1}$) and $P_{t-1,j}$ to the origin respectively, and Δd is the pre-defined threshold describing how close two planes are required to be.

Appearance test In indoor environments, some planes can be very close to each other. For instance when a door is closed, it is parallel to the wall, and usually there is a small displacement between them along their normal direction. In such a case, color can be used to disambiguate plane correspondences. Color histograms in RGB space for the candidate matching planes are calculated and compared. Planes with the highest color similarity are chosen as corresponding planes. Given two color histograms H_i and H_j, the correlation measure shown in in the following equation is used to estimate their similarity:

$$d(H_i, H_j) = \frac{\sum_I (H_i(I) - \bar{H}_i)(H_j(I) - \bar{H}_j)}{\sqrt{\sum_I (H_i(I) - \bar{H}_i)^2 (H_j(I) - \bar{H}_j)^2}} \tag{5}$$

For each query plane $P_{t,i}$, the corresponding plane with the maximum correlation value is chosen as best match.

Plane matching consistency test In our work, most of the times only few corresponding planes are found in two successive frames; therefore RANSAC could not be used. In order to discard wrong matches, we use the following certainty factor instead:

$$u = k_a \times \frac{1}{\min(S_i, S_j)} + k_o \times |\hat{\theta}_i - \theta_j| + k_d \times |\hat{d}_i - d_j| \tag{6}$$

where k_a, k_o, k_d are three coefficients weighting the importance of corresponding plane areas, orientation and distance agreement with odometry constraint, which is similar to the measure metric proposed in [24]. S is the area of a plane. Eq. (6) implies that a matching between two large planes is more reliable than a matching between small ones. Since the odometric error is usually large, in our work we choose k_a to be bigger than k_o and k_d.

After plane matching we check the relative rotations obtained by the determined pairs of corresponding planes. For every two relative rotations, their absolute difference is computed, and the maximum one is obtained. If the maximum difference is less than a threshold, all the relative rotations are valid, i.e. all of the matches are correct. Otherwise wrong matches are assumed to exist. Based on the above discussion, the worst match indicated by the largest index is discarded. The same step is executed repeatedly until the maximum difference between relative rotations is less than the pre-defined threshold.

As for the problem of plane matching, corner matching consists in finding which corners represent the same physical corner and labelling them with the same index. Since all detected corners are made by intersecting planes, corner matching can be built on the basis of plane correspondences results. That means that corners which are constructed by the same planes are considered to be the same physical corner.

When all the detected planes and corners are labeled with indexes, we can estimate the transformations between pairs of robot poses, which include planes and corners with common indexes. The obtained edges are added to the pose graph. It is important to note that only the relative rotation angle can be obtained from a pair of corresponding planes, while a pair of corresponding corners can provide both relative rotation and translation information.

2.3 Loop Closure Detection

The main issue for vision-based loop-closure detection concerns the difficulty of recognizing already visited areas. In our work we use VFH descriptor [23]. VFH encodes both geometry and viewpoint information and stores the relative angular directions of surface normals in a histogram. The strong recognition ability of VFH is used here to find corresponding point clouds in order to recognize place revisiting events. Given a point cloud $\mathcal{P}_i$, its VFH descriptor D_i is computed from the current camera position and compared to the VFH estimated by the previous point clouds. Since comparing the current point cloud with all the previous ones would be computationally intensive, we only consider point clouds which contain at least one corner.

To make the loop closure detection more reliable, histogram color features are used. Given two point clouds $\mathcal{P}_i$ and $\mathcal{P}_j$, if the distance of their VFH descriptors is less than a maximum tolerance threshold D_{th}, and, the covariance of their color descriptors is larger than a pre-defined threshold H_{th}, $\mathcal{P}_i$ is considered to be similar to $\mathcal{P}_j$. It should be noted that there may be more than one point cloud satisfying this constraint; in this case, the point cloud with the largest color covariance is considered to match $\mathcal{P}_i$.

2.4 Pose Graph Optimization

After finding correspondences between planes and corners, the relative rotations and roto-translations which have been found are used to construct the edges of a pose graph. To create a globally consistent trajectory, we optimize the pose graph using the linear pose-graph optimizer presented in [10]. That approach has been proven to be accurate and fast in practice. After obtaining the corrected robot trajectory we attach the planar surfaces P_t^v or the point clouds to corresponding corrected poses, in order to create a 3D map of the environment.

3 Experimental Results and Discussion

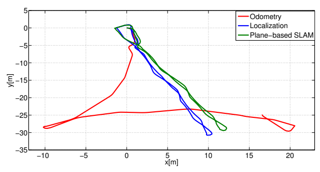
In order to evaluate the proposed plane-based 3D mapping algorithm, three experiments were carried out inside Dipartimento di Automatica e Informatica at Politecnico Di Torino. A Pioneer P3DX robot equipped with a laser range finder (SICK LMS-200) and a Microsoft Kinect sensor, is used in all the experiments. Laser is used only for obstacle avoidance and for benchmarking, while 3D point clouds are collected using the Kinect camera.

Point Cloud Library (PCL) [25] is used for point cloud processing, while OpenCV library [26] is used for processing the color histograms. In our experiments planes with an area smaller than 0.25 m^2 and with a number of supporting points smaller than 500 are filtered out. We use the following values for plane detection and matching: $\phi = 0.2\ rad$, $D_{CB} = 0.1\ m$, $\Delta\theta = 0.4\ rad$, $\Delta d = 0.2\ m$, $k_a = 0.05$, $k_o = k_d = 2$, $H_{th} = 0.45$.

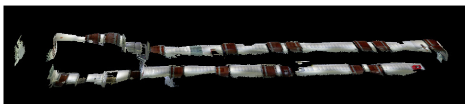
3.1 Long corridor

In this experiment, the robot travels in a corridor of dimensions 35 m $\times$ 4 m. In our experiment the robot starts from the left side of the corridor and goes along the corridor up to the right side, then it returns to the starting location, in order to form a loop closing. The robot travels autonomously using a simple obstacle avoidance algorithm. During the run the robot captures 258 point clouds. In order to evaluate the performances of the proposed approach, while gathering the data for the experiment we also perform localization on the robot using the laser scanner, in order to obtain an estimate of the trajectory which is close to the ground-truth. In order to localize the robot, we use a 2D occupancy grid map of the environment. For localization we use the sequential Montecarlo localization (MCL) algorithm described in [27]. The trajectory obtained from localization, while not precise, can be used as a coarse reference since the Cartesian error on the pose estimation was shown to be lower than 0.4 m with the same setup we used in this experiment [27], once the robot is correctly localized. At the end of the experiment we compare the trajectory estimated by our SLAM algorithm, as well as the one obtained by odometry only, with the trajectory estimated from robot localization.

Fig. 1(a) shows a comparison between the robot trajectory estimated by odometry only and by our SLAM algorithm with respect to localization estimates. It can be noted that the trajectory estimated by our approach is close to the reference trajectory estimated by localization, while the odometric trajecotry is quickly drifting from the ground-truth, with a difference at the end of approximately 35 m from the reference.



(a)



(b)

Fig. 1. Long corridor experiment. (a) Estimated trajectories. Red line is the odometric trajecotry only, the green line is the optimized trajectory, the blue line is the ground-truth estimated by MCL. (b) Top view of resulting map of the corridor.

Fig. 1(b) shows the resulting 3D map created using our algorithm. Walls and doors are correctly mapped. Notice that the floor does not appear in the final 3D map since the Microsoft Kinect was parallel to the ground plane (at an angle of 45° around the vertical) so few planes are detected on the floor due to the field of view of the sensor, and most of them are discarded for being too small or having few supporting points.

3.2 A Large Loop

We also tested the approach in a larger environment. The scenario consists of three corridors. The corridors are labeled as A, B, and C respectively, and their configuration is shown in Figure 2(b). The real appearance is also shown in Figure 2(c). In this experiment the robot moved according to a wall following strategy; the approximate trajectory is shown in Figure 2(c). The robot traveled a total distance of 180 m and collected 650 sensor samples. A smaller loop was closed, shown in red line in 2(c), followed by one global loop, shown in magenta color.

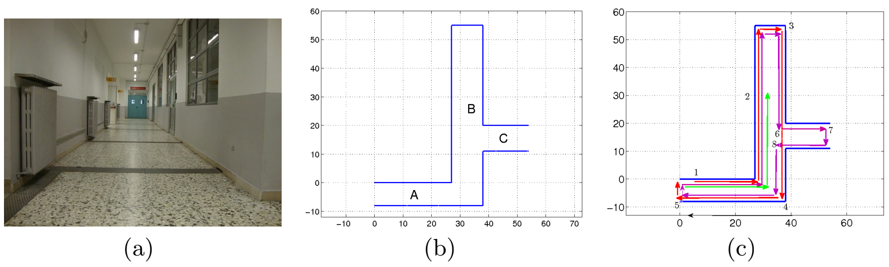


Fig. 2. (a) The corridor. (b) The path followed by the robot. (c) The three areas.

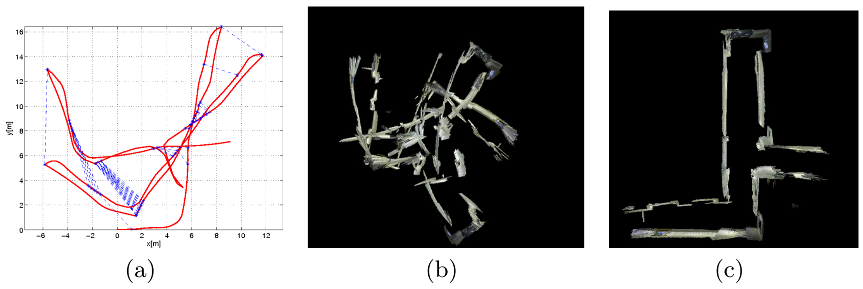


Fig. 3. (a) Graph for the large loop experiment. Red line represents odometric contraints, blue dashed lines show loop closure constraints. (b) The 3D map obtained using odometry only. (c) The 3D map estimated using plane-based SLAM.

In figure 3(a) we show the constrained graph, while the corresponding 3D maps are shown in Figure 3.

To show the effectiveness of the proposed loop closure detection using VFH, we shown in Figure 3(a) the estimated trajectory using plane-based SLAM but without loop closings.

As it can be seen in Figure 3, the obtained 3D map matches the real environment; in particular walls, doors and pillars are correctly represented in the map. However, it should be noted that in corridor A the walls are not parallel. The reason is that corridor A is not a closed, so when the robot turned no planar features could be detected in that part of the map, so only odometric constraints were used, leading to orientation errors.

Some misalignments are also present in corridor C. The reason for that is the presence of window glasses on one side of the corridor. Those surfaces are mostly invisible to the Kinect sensor.

4 Conclusion

In this work we presented a plane-based approach for the creation of 3D maps of planar indoor environments. We used a robot equipped with a Microsoft Kinect camera and wheel encoders. Vertical planar surfaces are extracted from the obtained point clouds. From the extracted planes correspondences are found between consecutive frames, as well as corners formed by two intersecting vertical planes, using geometric and color constraints in order to discard wrong matches. Loop closings are detected using a combination of 3D features, color features and geometric constraints. The obtained constraints are used to create a pose-graph which is then fed to a pose-graph optimization algorithm. Finally a 3D map of the environment is built by attaching the extracted planes to their relative optimized poses. Experimental results validates the proposed approach by showing that the reconstructed 3D maps are consistent and close to the real scenarios. Ongoing work is being devoted to enhancing the field of view by using two Kinect sensors and improving the back-end by introducing switchable constraints [28]. Future work will be devoted to the refinement of the pose and appearance of the planes over different observations, in order to improve the overall robustness and enhance the final map.

References

[1] P. Henry et al. “RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments”. In: *Proceedings of the International Symposium on Experimental Robotics (ISER).* 2010.

[2] M. Magnusson, A. Lilienthal, and T. Duckett. “Scan registration for autonomous mining vehicles using 3D-NDT”. In: *Journal of Field Robotics* 24.10 (2007), pp. 803–827.

[3] A. Nüchter, K. Lingemann, and J. Hertzberg. “6D SLAM-3D mapping outdoor environments”. In: *Journal of Field Robotics* 24.8-9 (2007), pp. 699–722.

[4] K. Pathak et al. “Online three-dimensional SLAM by registration of large planar surface segments and closed form pose-graph relaxation”. In: *Journal of Field Robotics, Special Issue on 3D Mapping* 27.1 (2010), pp. 52–84.

[5] S. May et al. “Three dimensional mapping with time-of-flight cameras”. In: *Journal of Field Robotics* 26.11-12 (2009), pp. 934–965.

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

[7] Microsoft. In: `http://www.xbox.com/en-US/kinect` (2010).

[8] J. Borenstein and L. Feng. “Measurement and correction of systematic odometry errors in mobile Robots”. In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 869–880.

[9] G. Grisetti, C. Stachniss, and W. Burgard. “Non-linear constraint network optimization for efficient map learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.3 (2009), pp. 428–439.

[10] L. Carlone et al. “A linear approximation for graph-based simultaneous localization and mapping”. In: *Proc. of Robotics: Science and Systems.* 2011.

[11] G. Grisetti et al. “A Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping”. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation.* 2010.

[12] P. J. Besl and N. D. Mckay. “A method for registration of 3-D shapes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256.

[13] Y. Furukawa et al. “Reconstructing building interiors from images”. In: *Proceedings of the International Conference on Computer Vision (ICCV).* 2009.

[14] D.G. Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[15] H. Bay and A. Ess. “SURF: Speeded up robust features”. In: *Computer Vision and Image Understanding (CVIU)* 110.3 (2008), pp. 346–359.

[16] E. Rublee et al. “ORB: an efficient alternative to SIFT or SURF”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 2011.

[17] M.A. Fischler and R.C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (1981), pp. 381–395.

[18] B. Steder et al. “Point feature extraction on 3D range scans taking into account object boundaries”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).* 2011.

[19] R.B.Rusu, N.Blodow, and M.Beetz. “Fast point feature histograms (FPFH) for 3D registration”. In: *Proceedings of the 2009 IEEE International Conference on Robotics and Automation(ICRA).* 2009.

[20] K. Pathak et al. “Fast 3D mapping by matching planes extracted from range sensor point-clouds”. In: *International Conference on Intelligent Robots and Systems (IROS).* St. Louis, 2009.

[21] K. Pathak et al. “Fast registration based on noisy planes with unknown correspondences for 3D mapping”. In: *IEEE Transactions on Robotics* 26.3 (2010), pp. 424–441.

[22] A. Birk et al. “Surface representations for 3D mapping: a case for a paradigm shift”. In: *Künstl Intell* 24.3 (2010), pp. 249–254.

[23] R.B. Rusu et al. “Fast 3D recognition and pose using the viewpoint feature histogram”. In: *Proceeding of the International Conference on Intelligent Robots and Systems (IROS)*. 2010.

[24] A.Harati and R.Siegwart. “Orthogonal 3D-SLAM for indoor environments using right angle corners”. In: *Proceedings of the 3rd European Conf. Mobile Robotics (ECMR’07)*. 2007.

[25] R. B. Rusu and S. Cousins. “3D is here: Point Cloud Library (PCL)”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011.

[26] G.Bradski. “The OpenCV Library”. In: *Dr.Dobb’s Journal of Software Tools* (2000).

[27] F.Abrate et al. “Three-state Multirobot collaborative localization in symmetrical environments”. In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*. 2009.

[28] N. Suenderhauf and P. Protzel. “Switchable Constraints for Robust Pose Graph SLAM”. In: *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*. 2012.