Tabletop Object Scanning with an RGB-D Sensor*

Maria Dimashova[1] and Ilya Lysenkov[1]

***Abstract*— The paper presents a complete pipeline for tabletop object scanning which gives accurate 3D models from RGB-D sensor data and can be carried out both by a person and a robot. We consider the capturing scenario when a camera makes one turn around a table with an object. The online part of our approach consists of initial camera motion estimation using RGB-D visual odometry and detection of loop closure in the trajectory. The offline part is graph-based global refinement of camera poses and model points. We propose to perform global graph optimization in a coarse-to-fine scheme to increase its basin of convergence what is often used in odometry algorithms. In addition to usual ICP constraints we also enhanced a global cost function by constraints from RGB data to make the scanning pipeline more stable in case of planar and symmetric surfaces.**

The presented object reconstruction pipeline was tested using both a hand-held RGB-D sensor and a PR2 robot. As a result the algorithm successfully produced accurate models on a dataset of 42 household objects. The code of our system is available as open-source.

I. INTRODUCTION AND RELATED WORK

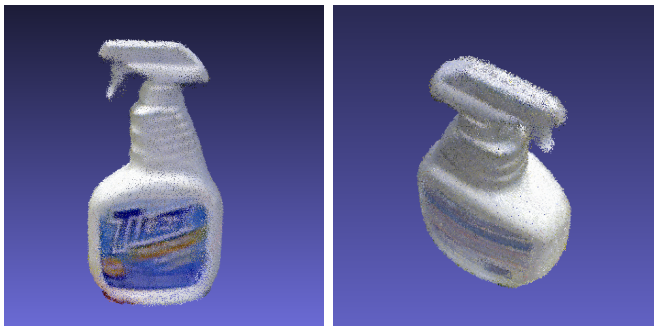
Object scanning is a common problem which has several important applications in robotics. 3D models of objects are commonly required for reliable grasping [1], [2], manipulation and motion planning and also can be used in perception tasks like recognition, pose estimation and tracking [3], [4]. Such applications will benefit from a convenient procedure to produce accurate models of new objects. In this paper we aim for creating an easy-to-use object scanning pipeline which can be used both by people and robots to create 3D models of objects. We consider rigid objects that can be encountered by a robot in a domestic environment and that can require robotic manipulation: for example, kitchenware, children's toys and small pieces of furniture to name a few.

There exist two possible ways to scan the considered kind of objects: in-hand or tabletop scanning. In-hand scanning faces the problems of automatic object segmentation and possible misregistration of symmetric object scans. Black gloves on hands and black curtain as background are used in capturing setup of [5] to solve the problem of object segmentation. Such restriction on a background is not always applicable, moreover the problem with symmetric objects isn't solved in this work, as well as in an earlier similar approach [6]. [7] overcomes both problems successfully but they use knowledge about an object pose from a robot manipulator that holds it. Such information is not available in case of a person hand and even a vision-based hand tracking will not help with the problem of symmetric objects in all cases at least due to occlusions of a hand by scanned objects. So in-hand scanning doesn't suit for our goal, creating a general scanning pipeline which can be used both by people and robots, and we use a table for scanning.

(a) PR2 moving around a fixed table

(b) Hand-held sensor moving around a fixed table

(c) Turning table in front of a fixed sensor



(d) A model of "Tilex" bottle captured by PR2

Fig. 1: Examples of the algorithm working: (a)-(c) are the supported capturing setups, (d) results of working

Probably the most common tabletop object scanning setup in computer vision or robotics laboratories is a table or a turntable with visual markers on it (e.g. chessboards) which are used for the camera pose estimation and can be applied for an object segmentation too. [8] also provides an option

*This work was supported by Willow Garage, Inc.

[1]All authors are with Itseez, Nizhny Novgorod, Russia {maria.dimashova, ilya.lysenkov}@itseez.com

to replace special visual markers by arbitrary texture. But in this case there is a training stage for textured visual features: the user has to capture a canonical image of the empty table from a planar frontal view before scanning. Preparing the markers and a training stage for texture are inconvenient for a person and hard or impossible for a robot.

Our approach also requires arbitrary texture on a table at our scanning setup. For the camera motion estimation we use state-of-the-art dense RGB-D odometry algorithm [9] instead of feature-based tracking and it allows us to eliminate a training stage. We can not abandon the requirement of texture because the depth-based techniques for the camera motion estimation like ICP [10] are not reliable in scenes with the dominant plane [11] which we have.

For the same reason the impressive technique KinectFusion [12] which uses ICP for camera poses estimation can not be applied in our capturing setup. Usage of KinectFusion without a table plane or intentionally cluttered table returns to the problem of automatic object segmentation. Also KinectFusion smooths captured surfaces too much, [13] measured that the system can resolve object details with a minimum size of approximately 10mm.

The other recent approach [14] for 3D model learning exploits both RGB and depth information of an input frame to estimate a camera pose in relation to an aggregated scene model represented by the multi-resolution surfels map. Using RGB data can allow this method to get stable pose estimation for scenes with a dominant plane. Also camera pose estimation in "frame-to-model" mode in this work as well as in KinectFusion leads to less accumulation of the pose estimation error. But it is still possible that there are small camera pose errors which result in small misalignment of RGB-D scans. Our approach applies global alignment of scans to increase their consistency to the input RGB and depth data.

Model reconstruction can be referred to the problem of simultaneous localization and mapping (SLAM). There are a lot of SLAM approaches including those ones working with RGB-D cameras, e.g. [15], [16], [17], [18]. But in contrast to our goal, a scanning relatively small objects, such approaches are more oriented for a more long-lasting navigation and large space mapping. Many of them aim at real-time working on a robot, so they can afford alignment of scans as rigid-bodies only. The recent work [19] introduced a post-processing procedure for the output of SLAM systems which considers the registered scans as non-rigid bodies and produces highly accurate surface of a model by simultaneous refinement of model points positions and camera poses. Pairwise and global alignment of camera poses and depth maps which includes non-rigid deformations is also applied in [20]. Their results include tests on the Kinect sensor data but their work is mainly oriented for Time-of-Flight (ToF) cameras. Considering non-rigid deformations of depth maps allowed them to get high quality 3D models even for very noisy ToF depth measurements. So we also follow the idea of simultaneous optimization of camera poses and model points for the final non-rigid refinement. We exploit the knowledge about the supported scanning setup and both RGB and depth data to produce initial alignment of scans as rigid bodies and then refine it by such optimization non-rigidly.

In this paper we provide a complete pipeline for object scanning in the tabletop capturing setup. This setup has natural requirements about a table and texture for the stable camera pose estimation, coping with symmetric objects and automatic object segmentation, and it is general enough to use it by a person or a robot. A base of 42 objects was scanned successfully by a PR2 robot, a person with a hand-held camera and using a turntable (see examples of scanning setups in Fig. 1).

II. OVERVIEW OF THE SCANNING PIPELINE

The goal of our scanning pipeline is to produce an accurate 3D model of an object captured by a person or a robot. In order to achieve this goal, we impose restrictions on a scanning setup and process: it should be one turn of an RGB-D camera around a textured table with an object.

A textured table allows solving the following problems:

- *Automatic object segmentation*. It is easy to find the table plane in a scene and segment an object placed on the table using RGB-D data.
- *Stable camera pose estimation*. Due to presence of texture a state-of-the-art RGB-D visual odometry algorithm can be applied. Alternative dense visual odometry methods based only on depth measurements like ICP are not reliable in a scene with a prevalent plane [6] because such algorithms need rich 3D structures for registration but a plane is too homogeneous (it is like trying to match textureless objects by looking for textured 2D features). Both dense approaches suppose a static scene and they can tolerate only slight movements in the scene. So to obtain a stable camera motion estimation we run the odometry using only points of a table and an object because they can be considered static. It allows applying the algorithm in dynamic scenes where there are moving objects e.g. people, pets or robots.
- *Scanning symmetric textureless objects*. Registration of scans of such an object is most likely to fail if it uses object points only. So an object pose should be known beforehand like in [7] or some part of a scene with texture or 3D features should also be used to add sufficient constraints. Textured table is exploited for this purpose in our pipeline and so the algorithm is able to create models of symmetric textureless objects.

One turn around the table simplifies loop closure detection and poses graph construction in contrast to an arbitrary camera trajectory and so makes the algorithm more robust because setting incorrect associations in the graph can lead to catastrophic implications in the offline graph optimization [21]. However, a complex object may require a more complicated trajectory or merging several scans together to capture all parts of the object and this is a direction for future work.

Our pipeline consists of online and offline stages. The online part is the capture of RGB-D data, initial estimation of camera poses using frame-to-frame odometry, selection of

keyframes set, detection of loop closure. The offline part is aimed for the global refinement of camera poses and model points. There are three stages of the offline refinement:

1) At first we seek the configuration of camera poses that is maximally consistent to the pure odometry estimations without considering the RGB-D frames data. The input camera poses of this stage are estimated by frame-to-frame odometry and this produces drift: the farther a camera pose is from the starting position of the trajectory, the larger an error of its estimation becomes. The drift is compensated by applying the constraint from the detected loop closure that redistributes the accumulated error among the poses evenly.
2) Then we refine the camera poses to achieve consistency of corresponding RGB-D scans as rigid bodies.
3) The last stage is mainly for the refinement of model points. It uses the idea similar to [19] that considers individual scans as non-rigid bodies and optimizes model points and camera poses simultaneously to achieve highly accurate alignment.

III. ONLINE STAGE

This stage is performed during the capture of RGB-D data. It includes frame-to-frame estimation of camera poses, selection of keyframes and detection of loop closure.

We use the dense RGB-D visual odometry method [9] for the camera motion estimation. It is accurate and fast enough to apply it online (execution time of 12.5Hz is reported in [9] on a single Intel Xeon E5520 CPU). Our open-source implementation[1] of this algorithm introduces optimization which allows it to work at 23Hz on one core of Intel Core i5 without using SSE. It is achieved by using only points with high gradient in the target intensity image to compute odometry (the same idea was proposed in [22] for another odometry algorithm). Other points provide less information for alignment and so they can be ignored without loss in accuracy of the RGB-D odometry algorithm as we checked using the TUM RGB-D benchmark [23].

The RGB-D odometry method assumes absence of motion in a scene and it is not robust to large movement in the scene. So to satisfy this requirement and keep stability of the odometry we run it only on the points of a table and an object. The mask of table points is found in real-time using an approach from [24]. An object mask is obtained by extracting points in a virtual 3D prism above the table [25]. Using only the points of a table and an object in the camera pose estimation brings one more advantage: moving a camera around a fixed table becomes equivalent to rotating a table in front of a fixed camera. So our system covers a scanning setup with a turntable too and we provide experiments both with a hand-held sensor and a turntable.

It is still possible to get an incorrect pose from RGB-D odometry because of light changes and motion blur from hand jitter. We filter out such outliers:

1) If a returned transformation between two frames is beyond the basin of convergence of the RGB-D odometry (about 15cm and 15 degrees in our experiments) most likely it is an outlier.
2) We also wrap a source frame using an estimated transformation, compute the difference between target and warped frames and filter the transformation out if depth and intensity differences are large for many pixels.

A pose of each new frame is estimated relatively to the last frame with a valid transformation from the odometry.

A keyframes set is selected during the capture. The used criterion is translation and rotation thresholds between a current frame and the last keyframe.

The frame-to-frame estimation of camera poses from the RGB-D odometry results in drift. It is usually not too large in our case due to high accuracy of the used odometry method [9] and only one turn around a table. But it still should be compensated and the loop closure event is detected to solve this problem. When length of the trajectory is large enough, we begin to estimate the odometry transformation between each current frame and the starting one. If the returned transformation passes through the described filters, this means the current pose is close to the first one and the trajectory has been closed. When the loop closure is detected the capture is stopped.

Thus each frame is passed through the steps:

1) plane and object segmentation,
2) odometry estimation and outliers filtering,
3) the check: is it a keyframe?
4) the check: is it loop closure?

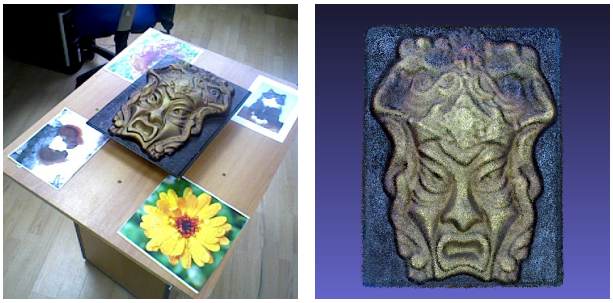
We also should note that loop closure detection using the RGB-D odometry requires the last pose of a camera to be in a basin of the algorithm convergence (about 15cm and 15 degrees) from the first pose. To allow a larger difference between the first and the last frames another approach can be used in detection of loop closure, e.g. an often used method of matching visual features, like SIFT [26], SURF [27], ORB [28]. However, these feature detectors and descriptors failed to match even relatively close frames in our capturing setup of turning around a textured table. The reason is large affine distortions and these standard features are not fully affine invariant. ASIFT [29] was able to overcome this problem and perfectly coped with the task in our experiments, however, it has to be strongly optimized before its usage in an online application.

The output of the online part is a set of consecutive keyframes with estimated poses and the transformation between the first and the last keyframes from loop closure.

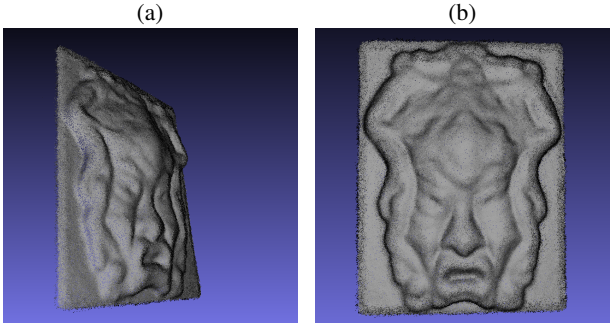
IV. OFFLINE STAGE

The purpose of the offline processing is refinement of the online-estimated camera poses as well as points of scans to get a model maximally consistent with the captured RGB-D data. We achieve this in three steps by solving least squares optimization problems. The cost function is extended at each

[1] http://bit.ly/rgbd_module



(a) (b)



(c) (d)

Fig. 2: Scanning the relief picture of Gorgon: (a) train image example, (b) colored model (point cloud), (c,d) uncolored point cloud

step by introducing additional terms which can be reliably computed thanks to refinement from a previous stage.

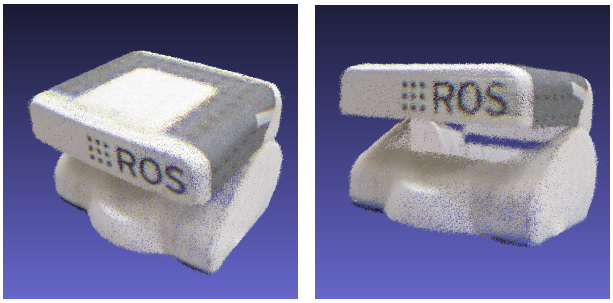
The input of the offline stage is a set of initial camera poses $\{x_n^{(0)},\ n = 1, \dots, N\}$ estimated by accumulating consecutive frame-to-frame odometry transformations $\{z_n,\ n = 1, \dots, N-1\}$ (where z_n is the odometry transformation from the frame $n+1$ to the frame n) and loop closure constraint z_N, i.e. odometry transformation from the last frame to the first one.

At the first step we compensate the drift in frame-to-frame estimated camera poses and make them consistent to the odometry constrains (including the transformation from the last pose to the first one) by minimizing the following cost function with regard to camera poses $x_{1:N}$:

$$F_{SE3}(x_{1:N}) = \sum_n (z_n \ominus y_n)^T \Omega_{SE3} (z_n \ominus y_n), \qquad (1)$$

where $\ominus$ is the inverse compounding operator [30], $y_n = x_{n+1} \ominus x_n$ $(n = 1, \dots, N-1)$, $y_N = x_N \ominus x_1$, the matrix Ω_{SE3} is an information matrix that weighs an odometry uncertainty [31].

The first step does not use directly RGB-D information of the frames and they are still slightly misaligned. So an RGB-D term is added to refine the camera poses at the second step to achieve better consistency among all RGB-D scans. The first step improved accuracy of poses by exploiting the loop closure event. So corresponding points between two frames can be found now by using the fast projective data association algorithm [11]. Two kinds of terms are added for each pair of corresponding 3D points of each frames pair to minimize their 3D distances and intensity difference.



(a) (b)

Fig. 3: Example of the problem with holes in a model point cloud due to occlusions of object parts: (a) view of PR2 head model close to one a camera had while capturing, (b) view that is lower a camera trajectory: there is a hole under the top part of the head

We use a common notation: p_{nk} denotes k^{th} 3D point of the n^{th} frame. So the cost function responsible for the shape consistency is a sum of the ICP odometry terms:

$$F_{ICP}(x_{1:N}) = \sum_{n,m,i,j} \Delta p_{nmij}^T \Omega_{mj} \Delta p_{nmij}, \qquad (2)$$

$$\Delta p_{nmij} = (p_{ni} \oplus x_n) \ominus x_m - p_{mj}, \qquad (3)$$

where summation is done over all corresponding points and Ω_{mj} is an information matrix that results in the point-to-plane distances in (2).

The cost function responsible for the intensity consistency is a sum of the RGB-D odometry terms:

$$F_{RGBD}(x_{1:N}) = \\ = \omega_{rgbd} \sum_{n,m,i} \Big[I_n(\pi p_{ni}) - I_m\Big(\pi\big((p_{ni} \oplus x_n) \ominus x_m\big)\Big)\Big]^2, \qquad (4)$$

where I_n is an intensity image, π is an image projection function, ω_{rgbd} is a weight of the RGB-D term.

So the following function is minimized at the second step:

$$F_{SE3}(x_{1:N}) + F_{ICP}(x_{1:N}) + F_{RGBD}(x_{1:N}). \qquad (5)$$

In order to increase basin of convergence we minimize (5) using a coarse-to-fine scheme what visual odometry algorithms often do. We recompute correspondences between frames after each iteration of each pyramid level.

The optimization of the function (5) is run twice for different points. The first time the points of a table and an object are used because the textured table points give necessary constrains in the case of symmetric and textureless object. The second time (5) is minimized using only object points to refine camera poses relatively to interesting parts of scans only.

We get accurate camera poses and scans aligned properly as rigid bodies after the second step. But there is camera noise in depth measurements and small misalignment of scans due to it. The third step produces a final accurate model

by simultaneous optimization of object points coordinates and camera poses.

This step exploits the idea of [19] but our version is simplified due to previous steps:

- As we already have well-aligned scans at this stage, we can apply the fast projective algorithm [11] (with filtering by differences in depth values, normals and intensity) to find correspondences instead of slower normal-shooting which [19] uses.
- We consider that each object 3D point of each frame is a surfel centered in this point i.e. for each frame there are as many surfels as object pixels it has. We don't aim for large space model refinement as [19] so it is not necessary for us to use a more compact scan surface representation by covering several pixels by one surfel.
- After the surfel 3D position was refined we update only depth value in the original pixel of a frame depth map. This conforms with the purpose to refine noisy depth measurements using information from other scans.

With these remarks the error function for the surfel positions refinement $F_{model}(x_{1:N}, M)$ (where M is a set of 3D points from all frames) in our approach is the same as formula (11) in [19].

So the following function is minimized at the third step:

$$F_{SE3}(x_{1:N}) + F_{ICP}(x_{1:N}) + F_{RGBD}(x_{1:N}) + \\ + F_{model}(x_{1:N}, M). \quad (6)$$

After (6) was minimized we filter out the points which don't have correspondences or have a small number of correspondences in all other frames because they were not refined or the refinement result is doubtful due to small number of used correspondences.

In order to solve the least squares optimization problems of the three steps (1), (5) and (6) they were formulated as the graph-based optimization tasks. These tasks were solved by applying the efficient g2o framework [31] which exploits sparseness of the graph structure.

The output of the offline stage is the final model representing a colored point cloud where the color of each 3D point is taken by its pixel coordinates from the frame where the point was originaly captured (see an example in Fig. 2).

V. RESULTS

The presented system was tested in three capturing scenarios:

- A person with a hand-held camera captured an object placed on a fixed table.
- A PR2 robot with its RGB-D sensor detoured around a fixed table with an object.
- A table was rotated in front of a camera fixed on a tripod.

As a result the algorithm successfully produced accurate models on a dataset of 42 household objects (see several examples in Fig. 4).

Fig. 4: Examples of reconstructed models

The offline refinement of camera poses and model points takes about 5-10 minutes for one model on Intel Core i5 with RAM of 12Gb depending on object size.

We also experimented to built a mesh for a model point cloud using the ball pivoting algorithm [32] implemented in MeshLab [33]. The accuracy of the point cloud allows building high quality meshes but still there are some problems. One of the important issues is holes in the mesh due to absent measurements of an RGB-D camera in occluded parts of object surface (see the Fig. 3 for an illustration of the problem). Allowing an arbitrary camera trajectory will reduce unobserved object area and give more complete point clouds and meshes. Projecting texture on a mesh is a next step. All of this is possible directions for future work.

VI. CONCLUSION

This paper presented the tabletop object scanning pipeline which gives accurate 3D models from RGB-D sensor data. It is convenient enough for usage by a person and also makes a robot closer to autonomous acquisition of a 3D model. The pipeline was successfully applied by a person and PR2 robot to scan a large dataset of objects and produced accurate models.

A code of the system is available as open-source at http://bit.ly/reconst3d.

ACKNOWLEDGMENT

We thank Vincent Rabaud for initiating the work as a Willow Garage project, active and useful discussions, help with testing on a PR2 robot as well as for the fast implementations of low-level steps of the algorithms like normal computing, plane finding, etc. We also thank Victor Eruhimov for the global supervision.

REFERENCES

[1] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110–122, 2004.

[2] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.

[3] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz, "Real-time cad model matching for mobile manipulation and grasping," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pp. 290–296, IEEE, 2009.

[4] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," 2012.

[5] T. Weise, T. Wismer, B. Leibe, and L. V. Gool, "Online loop closure for real-time interactive 3D scanning," *Computer Vision and Image Understanding*, vol. 115, no. 5, pp. 635–648, 2011.

[6] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," in *ACM Transactions on Graphics (TOG)*, vol. 21, pp. 438–446, ACM, 2002.

[7] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3D object modeling," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, 2011.

[8] Willow Garage, "object_recognition_capture: Data capture," `http://wg-perception.github.com/capture/`, 2013.

[9] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 719–722, IEEE, 2011.

[10] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Robotics-DL tentative*, pp. 586–606, International Society for Optics and Photonics, 1992.

[11] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152, IEEE, 2001.

[12] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pp. 127–136, IEEE, 2011.

[13] S. Meister, S. Izadi, P. Kohli, M. Hämmerle, C. Rother, and D. Kondermann, "When can we use KinectFusion for ground truth acquisition?," in *Workshop on color-depth fusion in robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[14] J. Stückler and S. Behnke, "Model learning and real-time tracking using multi-resolution surfel maps," in *Proc. of the AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012.

[15] J. Stuckler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pp. 162–167, IEEE, 2012.

[16] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1691–1696, IEEE, 2012.

[17] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[18] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2352–2359, IEEE, 2011.

[19] M. Ruhnke, R. Kummerle, G. Grisetti, and W. Burgard, "Highly accurate 3D surface models by sparse surface adjustment," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 751–757, IEEE, 2012.

[20] Y. Cui, S. Schuon, S. Thrun, D. Stricker, and C. Theobalt, "Algorithms for 3d shape scanning with a depth camera," 2012.

[21] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1879–1884, IEEE, 2012.

[22] T. Tykkala, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 2050–2056, IEEE, 2011.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[24] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3378–3383, IEEE, 2008.

[25] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and segmenting objects for mobile manipulation," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 47–54, IEEE, 2009.

[26] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[27] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011.

[29] J.-M. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 438–469, 2009.

[30] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[31] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[32] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 5, no. 4, pp. 349–359, 1999.

[33] Meshlab `http://meshlab.sourceforge.net`, 2013.