In Proceedings of RoboCup International Symposium, Singapore, June 2010.

Towards Semantic Scene Analysis with Time-of-Flight Cameras

Dirk Holz[1], Ruwen Schnabel[2], David Droeschel[1],
Jörg Stückler[1], and Sven Behnke[1]

[1] University of Bonn, Institute of Computer Science VI,
[2] University of Bonn, Institute of Computer Science II
{holz, droeschel, stueckler}@ais.uni-bonn.de
{schnabel, behnke}@cs.uni-bonn.de

Abstract. For planning grasps and other object manipulation actions in complex environments, 3D semantic information becomes crucial. This paper focuses on the application of recent 3D Time-of-Flight (ToF) cameras in the context of semantic scene analysis. For being able to acquire semantic information from ToF camera data, we a) pre-process the data including outlier removal, filtering and phase unwrapping for correcting erroneous distance measurements, and b) apply a randomized algorithm for detecting shapes such as planes, spheres, and cylinders. We present experimental results that show that the robustness against noise and outliers of the underlying RANSAC paradigm allows for segmenting and classifying objects in 3D ToF camera data captured in natural mobile manipulation setups.

1 Introduction

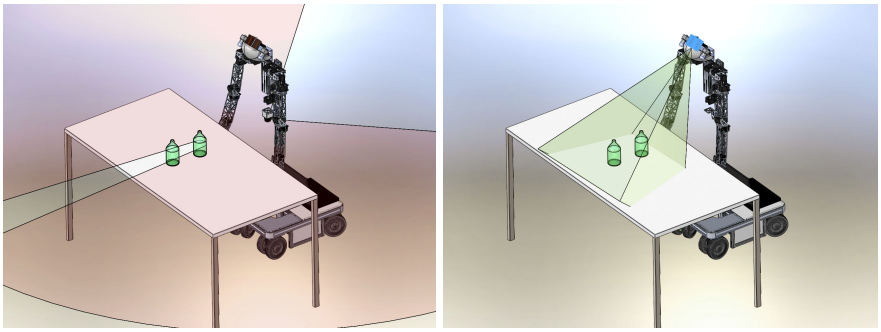
Autonomous mobile robots need environment models in order to plan actions and navigate effectively. Two-dimensional metric maps, built from 2D laser range scans, became the de-facto standard to tackle navigation problems such as path planning and localization of the robot platform. For planning arm motions and grasps, however, 3D semantic information becomes crucial since:

1. Objects need to be detected in the presence of other objects (e.g., on a cluttered table).
2. The robot needs to determine whether or not an object is graspable (e.g., with respect to its size).
3. The robot needs to determine the 3D pose of the object as a goal for its end-effector.
4. The robot needs to determine the 3D pose (and boundaries) of neighboring objects in order to plan an obstacle-free path.

The context of the work presented here is the RoboCup@Home league. This league addresses service robot applications and focuses on navigation (and SLAM) in dynamic environments, mobile manipulation and human-robot-interaction. A

typical task for a mobile service robot is to fetch and deliver objects such as beverages. This involves detecting and recognizing objects as well as planning arm motions and grasps.

In previous work, we used a 2D laser scanner for detecting possible object locations on tables. The 2D positions of the object candidates were then projected into the image plane of a color camera for feature-based object recognition. In case of a successful recognition (i.e., the object to deliver was among the candidates), grasping of the object was initiated. The drawback of this approach (as illustrated in Figure 1) is that objects can occlude each other in the two-dimensional measurement plane of the laser range scanner.



(a) Laser Perception (b) ToF Perception

Fig. 1. Perceiving objects on the table. With the trunk laser scanner (a), the second object is occluded and not perceived. Using the ToF camera (b), mounted in the robot's head, would allow for perceiving both objects.

In this work, we present our efforts in using Time-of-Flight (ToF) cameras for perceiving semantic information in the robot's workspace and, in particular, detect tables and objects as well as their shapes.

One of the first applications in robotics considering ToF cameras as an alternative to laser scanning has been presented in 2004 by Weingarten, Grüner, and Siegwart who evaluated a SwissRanger SR-2 camera in terms of basic obstacle avoidance and local path-planning capabilities [14]. In 2005, Sheh et al. used a ToF camera for human-assisted 3D mapping in the context of the RoboCup Rescue league [12]. Ohno et al. used a SwissRanger SR-2 camera for estimating a robot's trajectory and reconstructing the surface of the environment in 2006 [9]. Recently, May et al. presented and evaluated different approaches for registering multiple range images of a SwissRanger SR-3000 camera in the context of fully autonomous 3D mapping [6]. All the aforementioned approaches have shown that ToF cameras require to take care of their complex error model (see [6] for an overview).

The extraction of semantic information from 3D laser scan data has seen a lot of progress in the last decade, of which we want to mention two approaches.

Nüchter et al. extract environmental structures such as walls, ceilings and drivable surfaces from 3D laser range scans and use trained classifiers to detect objects, like for instance humans and other robots [8]. Rusu et al. extract hybrid representations of objects consisting of detected shapes, as will be done here, as well as surface reconstructions where no shapes have been detected [10]. Both approaches show good results when processing accurate 3D laser range data.

The remainder of this paper is organized as follows (referring to Figure 2): Section 2 covers pre-processing steps being necessary to cope with the complex error model of ToF cameras. Section 3 deals with the detection of table tops, the estimation of the corresponding planar models and the segmentation of individual objects. Section 4 finally covers the detection of primitive shapes in 3D point clouds and the classification of the segmented objects.

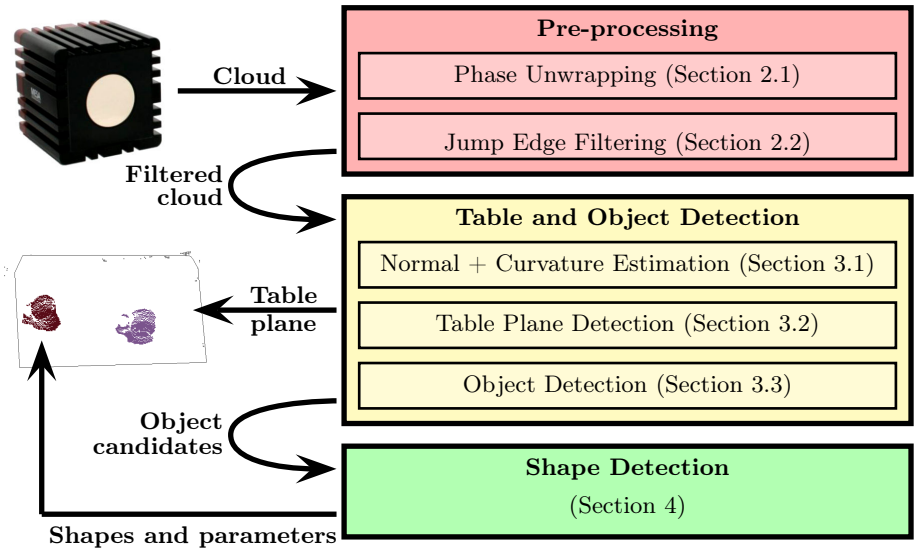


Fig. 2. System overview.

2 Pre-processing 3D Point Clouds

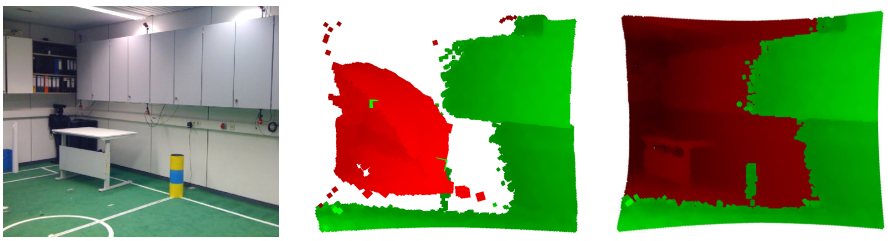
Besides a large variety of systematic and non-systematic errors (see [6]), ToF cameras show two problems that are characteristic for their measurement principle. The first problem is the ambiguity of distance measurements. The second problem is that the acquired point clouds contain phantom measurements occurring at distance discontinuities, i.e., at the boundaries of surfaces partially occluding each other. Both problems cause spurious measurements that do not correspond to any object in the real physical environment. By means of phase unwrapping and jump edge filtering both problems are addressed when preprocessing the acquired point clouds.

2.1 Probabilistic Phase Unwrapping

ToF cameras illuminate the environment by means of an array of LEDs that emit amplitude-modulated near-infrared light. The reflected light is received by a CCD/CMOS chip. Depth information is gained for all pixels in parallel by measuring the phase shift between the emitted and the reflected light. This phase shift is proportional to the object's distance to the sensor modulo the wavelength of the modulation frequency. This characteristic results in a distance ambiguity. That is, objects farther away from the sensor than the maximum measurable distance $d_{\max}$ are, respectively, *wrapped* and projected into the interval $[0, d_{\max}]$.

A common way to handle these distance ambiguities is to neglect measurements based on the ratio of measured distance and intensity. The amplitude of the reflected signal decreases quadratically with the measured distance. Sorting out points not following this scheme, e.g., points with a low intensity at a short distance, removes the majority of wrapped measurements but also valid measurements on less reflective surfaces.

In contrast to these approaches, we correct the wrapped measurements instead of neglecting them. We apply phase unwrapping techniques to reconstruct depth measurements behind the sensor's non-ambiguity range. The goal of phase unwrapping is to infer a number of phase jumps from the wrapped signal. Under the assumption that neighboring measurements are more likely close to each other than farther apart, relative phase jumps between neighboring pixels can be extracted. The signal can be unwrapped by integrating these phase jumps into the wrapped signal. We use a probabilistic approach based on [3] that relies on discontinuities in the image to infer these phase jumps. In addition to depth discontinuities, we incorporate the intensity of the reflected signal, since it depends on the object's distance and can indicate inconsistencies between a measured and the corresponding real distance.



(a) Scene (b) Wrapped cloud (c) Unwrapped cloud

Fig. 3. Phase unwrapping. By correcting the depth image at detected phase jumps, we can obtain valid measurements from objects being farther away from the sensor than the maximum measurable distance $d_{\max}$. Here the red measurements need to be corrected, the green points naturally lie in the interval $[0, d_{\max}]$.

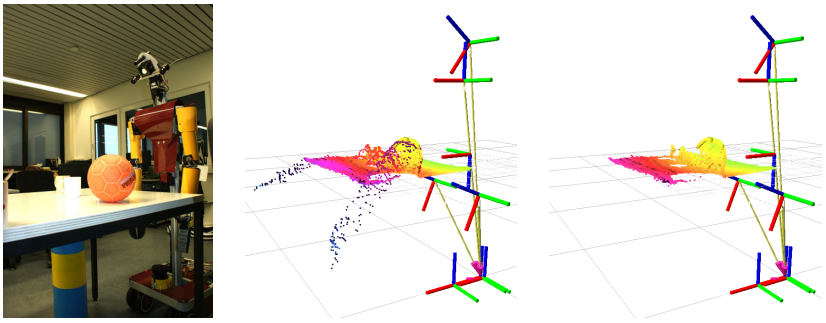
2.2 Jump Edge Filtering

Jump edges are known to cause spurious measurements that should either be corrected or neglected when processing ToF depth information. For simply neglecting these measurements, sufficient results are achieved by examining local neighborhood relations. From a set of 3D points $P = \{\boldsymbol{p}_i \in \mathbb{R}^3 | i = 1, \cdots, N_p\}$, jump edges J can be determined by comparing the opposing angles $\theta_{i,n}$ of the triangle spanned by the focal point $\boldsymbol{f} = \boldsymbol{0}$, point $\boldsymbol{p}_i$ and its eight neighbors $P_n = \{\boldsymbol{p}_{i,n} | i = 1, \cdots, N_p : n = 1, \cdots, 8\}$ with a threshold θ_{th}:

$$\theta_i = \max \arcsin \left(\frac{||\boldsymbol{p}_{i,n}||}{||\boldsymbol{p}_{i,n} - \boldsymbol{p}_i||} \sin \varphi \right), \tag{1}$$

$$J = \{\boldsymbol{p}_i | \theta_i > \theta_{th}\}, \tag{2}$$

where φ is the apex angle between two neighboring pixels. That is, neighboring points that lie on a common line-of-sight to the focal point $\boldsymbol{f}$ are, respectively, removed from the point cloud and marked as being invalid. A typical result of applying this filter is shown in Figure 4.



(a) Scene (b) Unfiltered cloud (c) Filtered cloud

Fig. 4. Sorting out jump edges. Shown are a photo of an example scene (a), the captured unfiltered point cloud (b) and the filtered cloud (c). It can be seen that the majority of erroneous measurements caused by jump edges, e.g., between table and floor in (b), are sorted out in the filtered cloud (c).

3 Table and Object Detection

The detection of tables and objects in the filtered point clouds is conducted in three steps: We first compute local surface normals and variations for all points. This information is then used to detect larger horizontal planes and fitting corresponding planar models into the data. Points above these planes but inside their boundaries are then clustered in order to form the object candidates for later shape detection and feature-based recognition.

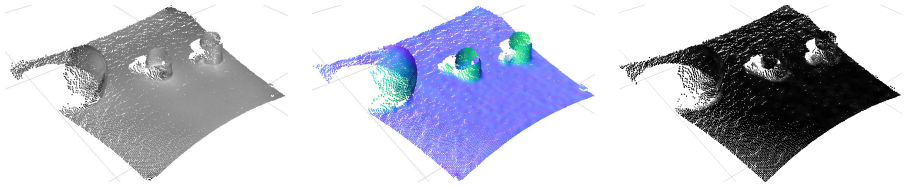
3.1 Computing Local Surface Normals and Curvature Changes

A common way for determining the normal to a point $\boldsymbol{p}_i$ on a surface is to approximate the problem by fitting a plane to the point's local neighborhood $\mathcal{P}_i$ in a least squares error sense. This neighborhood is formed either by the k nearest neighbors or by all points within a radius r from $\boldsymbol{p}_i$.

Searching for nearest neighbors is computationally expensive. Even specialized algorithms like approximate search in kd-trees [7] can cause longer runtimes when building the search structure for a larger point set. Instead of really searching for nearest neighbors, we approximate the problem and exploit the order of the measurements in the point cloud (176×144 distance measurements). We build a lookup table storing, for every point index, the ring-neighborhood being formed by the k closest indices in index space. That is, starting from $\boldsymbol{p}_i$ we circle around the image index ($x = i/176, y = i \mod 176$) in anti-clockwise order and store the point index for the traversed pixels in the lookup table. When processing a new point cloud we only update the squared distances from every point $\boldsymbol{p}_i$ to its k neighbors as provided by the lookup table.

The approximated nearest neighbors do not resemble the true nearest neighbors in the vicinity of transitions between different surfaces or partial occlusions, and if the point cloud is highly affected by noise and erroneous measurements. To take this into account, we check the computed squared distances and mark those being larger than some threshold r^2 as being invalid. That is, our local neighborhood $\mathcal{P}_i$ is bounded by both a maximum number of neighbors k and a maximum distance r.

Given the local neighborhood $\mathcal{P}_i$, the local surface normal $\boldsymbol{n}_i$ can be estimated by analyzing the eigenvectors of the covariance matrix $C_i \in \mathbb{R}^{3\times3}$ of $\mathcal{P}_i$. An estimate of $\boldsymbol{n}_i$ can be obtained from the eigenvector $\boldsymbol{v}_{i,0}$ corresponding to the smallest eigenvalue $\lambda_{i,0}$. The ratio between the smallest eigenvalue and the sum of eigenvalues provides an estimate of the local curvature Δc_i.



(a) Intensities (b) Normals (c) Curvature

Fig. 5. Computing local surface normals and curvature changes. Shown are the input point cloud with intensity information (a) as well as the computed surface normals (b) and local curvature changes (c). The used parameters are $k = 40$ and $r = 20\,\text{cm}$.

The aforementioned neighborhood approximation drastically decreases the computational complexity of the surface normal and curvature estimation. However, more important is that the involved inaccuracies did not considerably de-

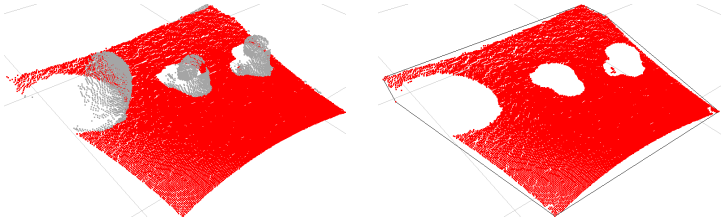
graded the results in our experiments. Figure 5 shows a typical example of local surface normals and curvature changes as computed for a pre-processed point cloud.

3.2 Detecting the Table Plane

For detecting tables in the vicinity of the robot, we extract those points $\boldsymbol{p}_i$ from the point cloud that satisfy the following constraints: 1.) the surface normal $\boldsymbol{n}_i$ is nearly parallel to the z-axis (i.e., $\boldsymbol{n}_i \parallel \hat{\boldsymbol{Z}}$) and 2.) the surface around $\boldsymbol{p}_i$ is smooth (i.e., $\Delta c \approx 0$). Points satisfying these constraints have likely been measured on the surface of a table and form an initial set of table points T. In order to distinguish multiple tables, we examine the distribution in the measured heights $\boldsymbol{p}_i^z, \boldsymbol{p}_i \in T$ and split T into multiple sets $T_1, \ldots, T_n$ in case of larger fluctuations. The same is done for larger variations in the position $(\boldsymbol{p}_i^x \ \boldsymbol{p}_i^y)^T$ of the points.

In order to obtain an efficient representation of tables and to segment individual objects, we fit a planar model into each table point set T_i using the *M-Estimator Sample Consensus* (MSAC) framework – an extension to the well-known RANSAC paradigm where inliers receive a certain score depending on how well they fit the data [13]. This M-Estimator is particularly robust against noise and outliers. For the point cloud from Figure 5, all points in T belong to the same table. The result of fitting a planar model to T is shown in Figure 6.a. The planar model that best fits the data is almost parallel to the xy-plane and is supported by 20 731 inliers. It can already be seen that, despite some points on the table's boundaries, the outliers correspond to the objects on the table.

Once the planar model has been found, we project all inliers onto the detected plane and compute the 2D convex hull by means of Graham's Scan Algorithm [4]. The convex hull for the 20 731 points from Figure 6.a is shown in Figure 6.b. It consists of 9 points and accurately represents the table top.



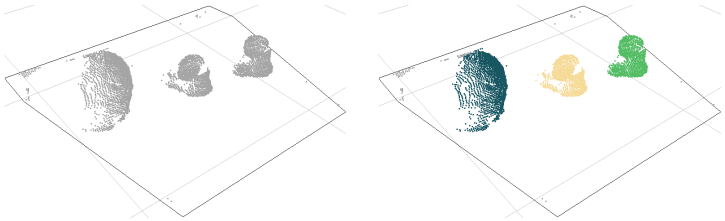
(a) Inliers supporting the model (b) Polygonal approximation

Fig. 6. Detecting the table plane. Shown are the inliers (red) supporting the planar model (a) as well as the 3D polygonal approximation (b) formed by the two-dimensional convex hull of the inliers (projected onto the table plane).

3.3 Clustering Outliers and Detecting Objects

All outliers from fitting the planar model as well as the points that have not been considered for the table point set T are potential object points. That is, they could have been measured on the surface of an object. Since we are only interested in objects on top of the table, we first sort out all points lying below the table plane as well as those points that do not lie within the bounding polygon. In order to obtain point sets that represent a common object, we apply a simple clustering based on the Euclidean distance between neighboring points. Neighboring points whose point-to-point distance is below a threshold $d_{\max}$ are recursively merged into clusters. Clusters whose cardinality exceed a minimum number $n_{\min}$ of support points are considered as object candidates.

The resulting segmentation of the ongoing examples from Figure 5 and Figure 6 is shown in Figure 7. In order to use the segmented object clusters for motion planning, we compute the centroid as well as the oriented bounding box for all points in each cluster. For planning grasps, however, we need to determine the shape of the objects.



(a) Outliers from planar fit (b) Object candidates

Fig. 7. Detecting object candidates. Shown are the unclustered outliers (a) and the object candidates (b) obtained from Euclidean clustering. Object candidates are colored. The remaining points are gray. Here, the parameters are $d_{\max} = 2.5\,\text{cm}$ and $n_{\min} = 250$.

4 Randomized Shape Detection

In order to robustly detect different kinds of geometric primitives, we employ an efficient RANSAC algorithm that directly operates on the point clouds and the associated surface normal information. Indeed, in our setting we can closely follow a simplified version of the approach proposed by Schnabel et al. [11]. While the original method focuses on achieving efficiency even on huge point clouds, the point clouds in the considered application are comparatively small and thus not all optimizations worthwhile.

Given a point cloud $P = \{\boldsymbol{p}_i \in \mathbb{R}^3 | i = 1, \ldots, N_p\}$ with associated normals $\{\boldsymbol{n}_i, \ldots, \boldsymbol{n}_{N_p}\}$, the output of the algorithm is a set of primitive shapes $\Psi = \{\psi_1, \ldots, \psi_n\}$ with corresponding disjoint sets of points $P_\Psi = \{P_{\psi_1} \subset P, \ldots, P_{\psi_n} \subset P\}$ and a set of remaining points $R = P \setminus \bigcup_\psi P_\psi$.

The shape extraction problem is framed as an optimization problem defined by a score function σ_P. In each iteration of the algorithm, the primitive with maximal score is searched using the RANSAC paradigm. New shape candidates are generated by randomly sampling minimal subsets of P. Candidates of *all* considered shape types are generated for *every* minimal set and all candidates are collected in the set C. Thus, no special ordering has to be imposed on the detection of different types of shapes. After new candidates have been generated, the candidate m with the highest score is computed employing the efficient lazy score evaluation scheme presented in Sec. 4.3. The best candidate is only accepted if, given the number of inliers $|m|$ of the candidate and the number of drawn candidates $|C|$, the probability that no better candidate was overlooked during sampling is high enough (see [2]). If a candidate is accepted, the corresponding points P_m are removed from P and the candidates C_m generated with points in P_m are deleted from C. The algorithm terminates as soon as the probability of detection for a shape with a user defined minimal size τ is large enough.

4.1 Shape Estimation

The shapes we consider in this work are planes, spheres, cylinders, cones and tori which have between three and seven parameters. Every 3D-point p_i fixes only one parameter of the shape. In order to reduce the number of points in a minimal set, we also use the unoriented approximate surface normal n_i for each point, so that the direction gives us two more parameters per sample. That way it is possible to estimate each of the considered basic shapes from at most three point samples. However, always using one additional sample is advantageous because the surplus parameters can be used to immediately verify a candidate and thus eliminate the need of evaluating many relatively low scored shapes [5].

4.2 Score

The score function σ_P is responsible for measuring the quality of a given shape candidate. We use the following aspects in our scoring function: 1.) To measure the support of a candidate, we use the number of points that fall within an ϵ-band around the shape. 2.) To ensure that the points inside the band roughly follow the curvature pattern of the given primitive, we only count those points inside the band whose normals do not deviate from the normal of the shape more than a given angle α. 3.) Additionally we incorporate a connectivity measure: Among the points that fulfill the previous two conditions, only those are considered that constitute the largest connected component on the shape.

4.3 Score Evaluation

Obviously the cost of evaluation would be prohibitive without any optimizations because in a naïve implementation, the distance to all points in P would have to

be computed together with a normal at a corresponding position on the shape for each candidate. But since in each run we are only interested in the candidate that achieves the highest score, using the entire point cloud P when computing $\sigma_P(\psi)$ is not necessary for every shape candidate.

We significantly reduce the number of points that have to be considered in the evaluation of $\sigma_P(\psi)$ by splitting the point cloud P into a set of disjoint random subsets: $P = S_1 \bigcup \ldots \bigcup S_r$.

After a shape candidate was generated and successfully verified, the candidate is only scored against the first subset S_1 and no connected component is extracted yet. From the score $\sigma_S(\psi)$ on a subset $S \subset P$ an estimate $\hat{\sigma}_P(\psi)$ for the score $\sigma_P(\psi)$ on all points can be extrapolated using the well known induction from inferential statistics:

$$\hat{\sigma}_P(\psi, S) = -1 - f(-2 - |S|, -2 - |P|, -1 - |S_\psi|) \ , \tag{3}$$

$$\text{where } f(N, x, n) = \frac{xn \pm \sqrt{\frac{xn(N-x)(N-n)}{N-1}}}{N} \tag{4}$$

is the mean plus/minus the standard deviation of the hypergeometric distribution. $\hat{\sigma}_P(\psi)$ is a confidence interval $[l_\psi, u_\psi]$ that describes a range of likely values for the true score $\sigma_P(\psi)$. The expected value $E(\sigma_P(\psi))$ is given by $\frac{l_\psi + u_\psi}{2}$. With this extrapolation the potentially best candidate ψ_m can be quickly identified by choosing the one with the highest expected value. Since the uncertainty of the estimation is captured in the confidence intervals, the truly maximal candidate can be found by comparing the confidence intervals of the candidates.

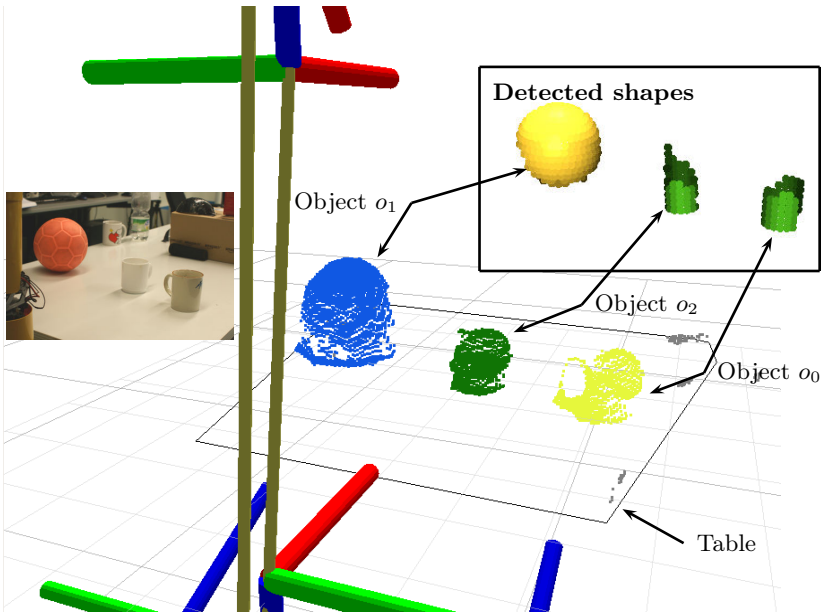
If the confidence intervals of ψ_m and another candidate ψ_i overlap, the score on an additional subset is evaluated for both candidates and new extrapolations are computed, now taking into account the scores on all subsets that have already been computed. The more subsets have been considered, the smaller becomes the range of the confidence intervals, since the uncertainty in the estimation decreases. Further subsets are included until the confidence intervals of ψ_i and ψ_m no longer overlap and it can be decided if either ψ_i or ψ_m is better.

The advantage of this priority-based candidate evaluation is that it is less dependent on the random order in which candidates are generated. Compared to the related but order-dependent approach of David Capel [1], we achieve a 20% speedup on average on a single core machine.

Results of applying the shape detection algorithm to the object clusters from Section 3 are shown in Figure 8.

5 Conclusion

We have presented a simple, yet efficient and robust, tool chain for extracting semantic information from ToF camera data. It comprises techniques for correcting erroneous measurements caused by the ambiguity in distance measurements as well as for filtering points on jump edges. By means of a MSAC-based approach and the information about the surface in a point's local neighborhood,



Detected Objects and Shapes (all units in m)	
Table	20 753 points supporting the planar model Normal vector: [−0.0232 − 0.0156 0.9996] Distance to **0**: 0.742 795
Object o_0 (cup)	758 points in the cluster, detected primitive: Cylinder Centroid: [0.6847 − 0.0166 0.7885] Radius: 0.0314 (ground truth: 0.034) Axis direction: [0.0869 − 0.0120 0.9961]
Object o_1 (ball)	1738 points in the cluster, detected primitive: Sphere Centroid: [0.6871 0.2058 0.8598] Radius: 0.0778 (ground truth: 0.082)
Object o_2 (cup)	815 points in the cluster, detected primitive: Cylinder Centroid: [0.6847 − 0.0166 0.7885] Radius: 0.0284 (ground truth: 0.032) Axis direction: [0.1028 0.0080 0.9915]
Residual points	1185 points on jump edges 95 points not assigned to table or object clusters

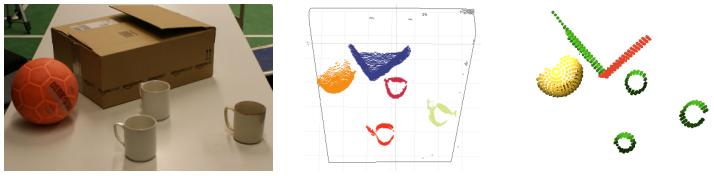


Fig. 8. Typical result of table, object, and shape detection. TOP: After fitting the planar model, the 20 753 inliers are removed and the table is represented solely by the model parameters and the convex hull of the inliers. The remaining 4591 points are segmented into 3 clusters (random colors) and successfully classified as being a sphere and two cylinders (see table below). BOTTOM: Example of a more complex scene with one false detection (the left side of the box is detected as a cylinder due to the highly inaccurate data in this region). Colors are red (plane), green (cylinder) and yellow (sphere).

we are able to detect table tops and objects thereon. Furthermore, we presented an approach for detecting primitive shapes in the detected objects that allow, e.g., for planning grasping motions.

In its current state, the presented approach can process complete point clouds of 25 344 points (i.e., without working on a sub-sampled copy of the cloud) with 2.5 Hz to 5 Hz. In future, we plan to further speed up the processing tool chain for being able to track objects with much higher frame rates. Furthermore, it is planned to use the extracted semantic information for registration and to construct 3D semantic maps of the robot's workspace.

References

1. D. P. Capel. An effective bail-out test for RANSAC consensus scoring. In *Proc. British Machine Vision Conf.*, pages 629–638, 2005.
2. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.
3. B. J. Frey, R. Koetter, and N. Petrovic. Very Loopy Belief Propagation for Unwrapping Phase Images. In *Neural Information Processing Systems Conference (NIPS), Algorithms & Architectures*, 2001.
4. R. L. Graham. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters*, 1(4):132–133, 1972.
5. J. Matas and O. Chum. Randomized RANSAC with t(d, d) test. In *Proc. of the British Machine Vision Conference 2002 (BMVC)*, 2002.
6. S. May, D. Droeschel, D. Holz, S. Fuchs, E. Malis, A. Nüchter, and J. Hertzberg. Three-dimensional mapping with time-of-flight cameras. *Journal of Field Robotics, Special Issue on Three-Dimensional Mapping, Part 2*, 26(11-12):934–965, 2009.
7. D. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. In *Proc. of the 2nd Annual Fall Workshop on Computational Geometry*, 1997.
8. A. Nüchter and J. Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.
9. K. Ohno, T. Nomura, and S. Tadokoro. Real-time robot trajectory estimation and 3d map construction using 3d camera. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.
10. R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.
11. R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, 2007.
12. R. Sheh, M. W. Kadous, and C. Sammut. On building 3d maps using a range camera: Applications to rescue robotics. Technical report, UNSW, Sydney, Australia, 2006.
13. P. H. Torr and A. Zisserman. MLESAC: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.
14. J. W. Weingarten, G. Grüner, and R. Siegwart. A State-of-the-Art 3D Sensor for Robot Navigation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.