The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Domestic Environments

Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{rusu,blodow,marton,beetz}@cs.tum.edu

Abstract— **In this paper we present a framework for 3D geometric shape segmentation for close-range scenes used in mobile manipulation and grasping, out of sensed point cloud data. Our proposed approach proposes a robust geometric mapping pipeline for large input datasets that extracts relevant objects useful for a personal robotic assistant to perform manipulation tasks. The objects are segmented out from partial views and a reconstructed model is computed by fitting geometric primitive classes such as planes, spheres, cylinders, and cones. The geometric shape coefficients are then used to reconstruct missing data. Residual points are resampled and triangulated, to create smooth decoupled surfaces that can be manipulated. The resulted map is represented as a hybrid concept and is comprised of 3D shape coefficients and triangular meshes used for collision avoidance in manipulation routines.**

I. Introduction

Small, fast tilting and rotating laser scanners are becoming an interesting alternative to provide robots with dense 3D range images that can achieve an accuracy needed for many object manipulation tasks. To be of effective use for manipulation control we must equip the robots with scene interpretation mechanisms operating on 3D range images. In this context, a key computational problem is the fast and reliable segmentation of the point cloud maps and an accurate geometric reconstruction of selected object hypotheses of those objects that the robot intends to manipulate.

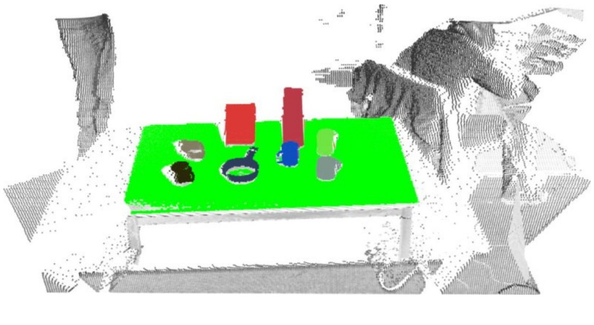


Fig. 1. A segmentation of a table scene and the object clusters found to lie on it from an unorganized point cloud dataset.

In this paper we tackle the problem of scene interpretation, including object reconstruction for everyday manipulation activities in domestic human living environments, in particular kitchens, out of point cloud data extracted from the range images acquired using laser scanners. Our proposed framework takes raw point cloud datasets as input, and segments horizontal planar areas that support objects used in manipulation scenarios (see Figure 1). Our mobile manipulation platform is described in detail in [1].

Our method estimates sets of hybrid geometric object models that consist of shape as well as surface models. The use of these hybrid models allows us to synergetically combine the strengths of shape representations and those of surface models (in particular, meshes). As Figure 2 illustrates shape models enable the system to add surface information for parts of the object that were not covered by sensor data. Also, shape models are parameterized representations for which grasps can be determined in advance and be reparameterized on demand. In addition, exploiting the knowledge about the shape enables the robot to eliminate inaccuracies caused by sensor noise as it determines the best shape fit for the sensor data. However, the use of shape models assumes that objects have strong regularities, which is often but certainly not always satisfied by the objects of daily use in domestic human environments. Thus, the remaining parts of the object cluster are reconstructed through meshes that can approximate arbitrary shapes but have the disadvantages that they are not compact, not generalizable and yield higher inaccuracies.

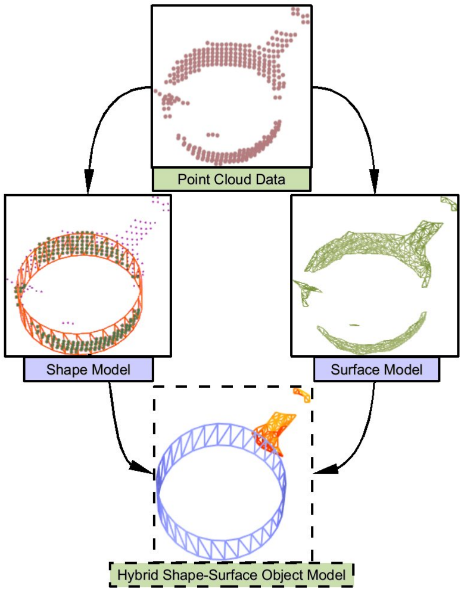


Fig. 2. The basic shape and surface model estimation step and the reconstruction of the hybrid shape-surface object model.

Thus our method first tries to explain parts of the point cloud through simple geometric shape models by fitting 3D geometric primitives such as spheres, cylinders, cones, and planes. It then reconstructs the remaining parts of the point cloud cluster using meshes using our surface reconstruction techniques [2] and combines the partial models of the point cloud cluster into a hierarchically structured hybrid model. An example of this reconstruction is presented in Figure 2.

The key contributions of this paper are: (1) a computational model for close scene segmentation and reconstruction; and (2) a novel approach that uses a combination of robust shape primitive models with triangular meshes to create a hybrid shape-surface representation useful for object grasping. The term *close-range* here refers to parts of the world reachable by the robot arms (i.e., working space) from the current position of the robot base.

The remainder of the paper is organized as follows. We address our system architecture in the next section, followed by the description of its two major components in Sections IV and V. We present experimental results for the segmentation of different table settings in Section VI and conclude in Section VII.

II. Related Work

Due to the unavailability of good sensing devices that could be installed on robotic platforms, the problem of perceiving and building complete 3D models for mobile manipulation and grasping has always been difficult. Some of the solutions adopted consist in creating complete 3D models offline and finding feature spaces to match the partial views online with models in the database [3], or using computer vision and machine learning models to train classifiers that can predict the grasping points in 2D and then triangulate their position with stereo cameras for example to find the 3D grasp points [4]. The latter can deal with novel objects (ones where a 3D model is not available) to some extent, but in principle both approaches are sensitive and work reliably only up to the training data used for learning the classifier. A different approach is presented in [5], where a highly accurate line laser works in combination with a camera to build models and create grasping points online, for novel objects. The results are very encouraging, however the authors test their system on just 2 different objects, so the problem of scalability is not addressed.

A vision-based grasping system which segments objects on a table and constructs triangular meshes for them is presented in [6]. While the presented method is general and works for many objects, it creates complicated models for certain objects, which could be simplified through the usage of geometric primitives. A simplification of the modeling problem is presented in [7], where the authors use geometric shape primitives and model each object as a sphere, cylinder, cone or box. A similar model decomposition approach is presented in [8], where the authors present a sample consensus approach to fit similar primitive models to noisy point clouds. In our case, we only model parts of the objects with geometric primitives, and reconstruct the geometry of the remaining parts using triangular meshes, thus obtaining a more precise object model. In [9], the authors use geons (geometric icons) to develop generic category descriptions using geometric primitives, but also additional category knowledge, for the purpose of building libraries of grasp models for a variety of objects. The recognition part is performed by looking at an object from different views, creating a volumetric approach, and then fitting a geon to the resultant volume. The most critical aspect of this approach is the performance of the object reconstruction algorithm (as the authors mention it themselves), which is highly sensitive to gripper positioning errors, and the evaluation takes a very long time to compute. The model fitting problem is extended to the use of superquadrics as geometric primitives in [10], [11], where complex objects decomposed a priori into superquadric parts can be fit to a point cloud dataset in a sample consensus approach. Though the results are interesting, this approach requires additional knowledge about the individual parts of an object, which is often hard to get.

III. System Architecture

The architecture of our mapping system is presented in Figure 3. Our methods take an unorganized 3D point cloud $\mathcal{P}$ as input and produce a set of shape coefficients $\mathcal{S}$ and a set of surface models $\mathcal{O}$ as output. We identify two main modules, namely the Scene Interpreter which extracts the supporting planes and segments the object clusters, and the Object Modeler, which creates the shape and surface models for each segmented object cluster. To process the data faster, a world coordinate frame with the $\mathcal{Z}$-axis pointing upwards is defined, and $\mathcal{P}$ is transformed into this frame.

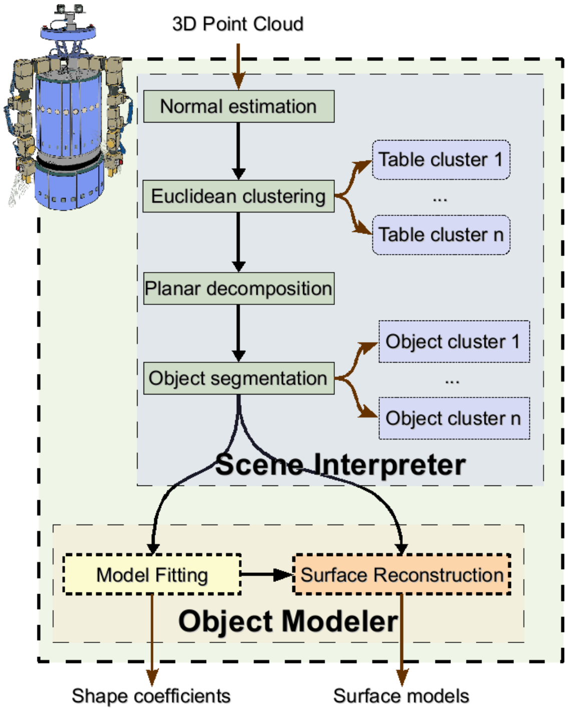


Fig. 3. The architecture of our mapping system. The input is comprised of unorganized 3D point cloud datasets, and the output is a set of reconstructed surface models $\mathcal{O}$ and a set of shape coefficients $\mathcal{S}$.

The mapping pipeline can be briefly described using the following computational steps:

1) for every 3D point $\boldsymbol{p}_i \in \mathcal{P}$, if no $\boldsymbol{n}_i$ surface normal information is already present, an estimate of the normal to the best least-squares plane fit locally to the k-nearest neighbors of $\boldsymbol{p}_i$ is taken;
2) point normals $\boldsymbol{n}_i$ that are approximatively parallel with the world $\mathcal{Z}$-axis, are grouped into a set $\mathcal{T} = \{\mathtt{t}_1 \cdots \mathtt{t}_n\}$ of Euclidean *table* candidate clusters;
3) for every cluster $\mathtt{t}_i \in \mathcal{T}$, a sample consensus robust estimator is used to find the best planar model;
4) for every set of point inliers $\mathcal{P}^i$ corresponding to the planar model for $\mathtt{t}_i$ found, a bounding polygon is computed as the table bounds;
5) all Euclidean point clusters $\mathcal{C}^i = \{\mathtt{c}^i_1 \cdots \mathtt{c}^i_n\}$ supported by the table model (i.e. sitting on the table, and within the polygonal bounds) are extracted;
6) for every cluster $\mathtt{c}^i_j \in \mathcal{C}^i$ a robust search for the best set of primitive shapes $\mathcal{S}^i$ is conducted;
7) a set of surface models $\mathcal{O}^i$ that takes $\mathcal{S}^i$ into account is built;
8) finally the resultant hybrid shape-model $\mathcal{M}$ is built by concatenating the shape model with the surface model.

In the following sections we will discuss each of the above mentioned steps separately and give insight on their implementation details.

IV. Scene Interpreter: Planar Decomposition and Object Segmentation

In general the segmentation of planar areas out of sensed point cloud data representing indoor environments can be done in a bruteforce manner (as presented in [12] for example) by simply picking 3 random points, estimating the parameters of a plane from them, and then scoring points to this model using a distance threshold, in a sample consensus framework. While this approach needs no a-priori data processing and works in a straightforward manner on simple datasets, it might fail on more complex environments, where a resultant model might contain points from various different parts of a room, for example, located on different objects. Furthermore, a model could count as inliers points which do not respect the plane equation, that is, the estimated surface normal $\boldsymbol{n}_i$ at an inlier candidate point $\boldsymbol{p}_i$ is not parallel to the plane's normal n.

To account for such complex environments, but also to considerably speed up the planar segmentation results, we proceed as follows. For a given point cloud dataset $\mathcal{P}$, we construct a downsampled version of it, $\mathcal{P}_d$, where $\boldsymbol{p}_j \in \mathcal{P}_d$ represents the centroid of a set of points $\mathcal{P}_j = \{\boldsymbol{p}_i \in \mathcal{P}\}$ obtained using a fixed spatial decomposition of the data (e.g. kD-tree). Furthermore, we estimate a normal $\boldsymbol{n}_j$ to the underlying surface represented by $\mathcal{P}_d$ by fitting a local plane to the set of points $\mathcal{P}_j$, and approximating $\boldsymbol{n}_j$ as the normal n of the plane.

Then, in general, we use similar sample consensus techniques as [12], but impose an additional constraint on the sample selection step, that is, for every two pair of points $\boldsymbol{p}_i, \boldsymbol{p}_j$ (with their estimated surface normals $\boldsymbol{n}_i, \boldsymbol{n}_j$) in the three required chosen samples: $\boldsymbol{n}_i \cdot \boldsymbol{n}_j \approx 0$.

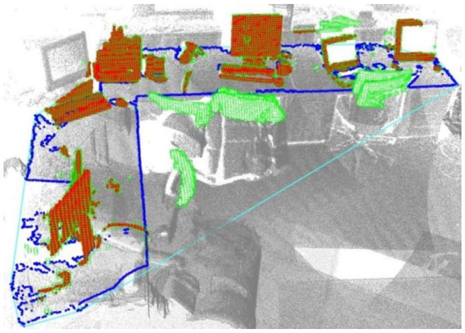


Fig. 4. Removing object candidates which fall within the convex polygon but do not intersect with the table. The table's boundary points are shown in blue, the convex hull lines with cyan, octree leaves with green, and the accepted object candidates in red.

In our research scenario however, we are mostly interested in the segmentation of tables as horizontal planes which can support objects on them. Therefore, we proceed at introducing another optimization in our planar decomposition approach, that is we look at the entire set of points $\boldsymbol{p}_i \in \mathcal{P}_d$, and only select those with their estimated surface normals $\boldsymbol{n}_i$ approximatively parallel with the world $\mathcal{Z}$-axis. After that, we perform a fast clustering of the selected points in an Euclidean sense and construct a set $\mathcal{T} = \{\mathtt{t}_1 \cdots \mathtt{t}_n\}$ of *table* candidate clusters. This has the advantage that since the clusters are independent with respect to each other, we can perform the subsequent planar segmentation step in parallel for more than one cluster, thus decreasing the computational requirements of this step considerably.

The next major step in our geometric processing pipeline is the object segmentation step. Given a set of validated $\mathtt{t}_i$ models, a search for sets $\mathcal{C}^i$ object candidates which are supported by these models is performed. These objects are said to be *movable*, in the sense that we are expecting the robot to be able to manipulate them. In a kitchen scenario they usually fall into the category of small kitchen utensils, dishware, food products, and so on.

To segment the set of objects, the $\mathtt{t}_i$ inliers are taken and a bounding 2D polygon is created for each candidate. Then, we look at the points whose projection on the $\mathtt{t}_i$ model falls inside the polygon. Because we treat the bounding as a convex problem, there are situations where the resulted shape includes additional points which do not lie on the table surface, as shown in Figure 4. To solve this, we break all the point candidates into regions in an Euclidean sense using an octree connectivity criterion (i.e. occupied neighboring leaves belong to one region), and then impose an additional constraint: each region's projection onto the $\mathtt{t}_i$ table has to have a minimum footprint. More specifically, we project the octree leaves of each point region onto the octree leaves of the table candidate, and then count the intersections. If the footprint area is too small, the point region will not be considered, as it might simply be sensor noise. The result of these processing steps is a set of Euclidean point clusters $\mathcal{C}^i = \{\mathtt{c}^i_1 \cdots \mathtt{c}^i_n\}$.

The right part of Figure 5 presents the remaining point clusters $\mathcal{C}^i$ after the previously mentioned segmentation steps

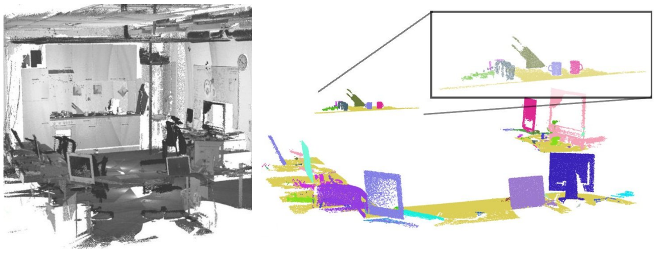


Fig. 5. Left: A snapshot of our kitchen lab dataset, comprising roughly 15 millions of points. Right: Point clusters supported by tables or shelves (shown in yellow) which constitute candidates for movable object clusters (shown in random colors).

for a given table from the kitchen dataset shown in the left part of the same figure.

V. Object Modeler: Model Fitting and Surface Reconstruction

In each segmented point cluster c_i we perform a RMSAC [13] (Randomized M-Estimator Sample Consensus) based search for 3D primitive geometric surfaces with a focus on 3D primitive geometric shapes such as planes, cylinders, spheres and cones.

We maintain a list of all models $\mathcal{S}^i = \{\mathsf{s}^i_1 \cdots \mathsf{s}^i_n\}$ that have been generated in the course of the search and score each of them based on their number of inliers. This list has the advantage that if a model is good enough to be accepted, the rest of the models get rescored quickly by subtracting the inliers of the extracted model from their inlier lists. This way, the remaining models will be found much faster, since we may have sampled them already. The implementations of these different models to be fitted share a common interface, so they can be easily enabled and disabled for different scenarios and new models can easily be added.

To achieve fast results, we use an octree for a localized sampling strategy [8] and optimized inlier calculations by hierarchically pruning octree leaves that cannot contain inlier points to the current model. This is done by traversing the octree, starting at the root node, and intersecting the bounding boxes of the current leaf's children with the shape. We don't descend into a child branch if its box is further away from the shape than the inlier threshold. For this reason, we implemented several box-shape intersection routines for all the primitives given above.

The box - *plane* intersection test can be performed by computing the distance of the box center to the plane and comparing that with the dimensions of the box in the direction towards the plane.

The key principle behind the box - *cylinder* intersection test is to reduce the problem to a two dimensional one by orthogonally projecting the box vertices along the cylinder axis, which forms an irregular hexagon. The cylinder will be projected onto a circle, and most cases can be decided if one of the hexagon edges forms a separating axis [14]. However, if no edge fulfills this property, there could still be the case that the separating line is present but not parallel to any hexagon edge. This requires more expensive checks, but in order to keep the intersection test simple and fast, this is not implemented in our framework. This can result in false positives, but that does not impact the correctness of the resulting inliers, since the boxes merely contain *potential* inliers.

Deciding whether a *sphere* intersects a box is an instance of the more general problem of intersecting a hypersphere with a hyperbox as described in [15].

The box - *cone* intersection test is performed in four steps, as follows:

- The simplest case is when the cone apex $\boldsymbol{a}$ is inside the box, which is trivial to check since the octree boxes are axis aligned.
- Next, if any of the box vertices is inside the cone, the box is regarded as possibly containing inliers. Note that a box that is completely inside the cone can result in a false positive.
- Even if no box vertex is inside the cone, an edge could penetrate the cone. This can be tested by perspectively projecting the cone and the box such that the cone becomes a circle. Then, one can look at every box edge in the projection and perform a two dimensional line - circle intersection test.
- At this point, the only possibility left that the box intersects the cone is if the apex is outside the box, all box vertices are outside the cone, and no box edge penetrates the cone surface, but the cone axis intersects with one of the box sides. The axis - box intersection is performed using Kay and Kayjia's ”slab” method [16].

The result of these intersection tests are the points contained in intersecting octree leaves, which can now be tested against the shapes for inlier selection. Since a large number of boxes will be checked against the same shape, it is helpful to precompute values that do not change for different boxes before traversing the octree.

During inlier selection, we discard points with normals contradicting those of the underlying model, and we enforce a connectivity criterion, i.e. we select the biggest connected component among the inlier candidates.

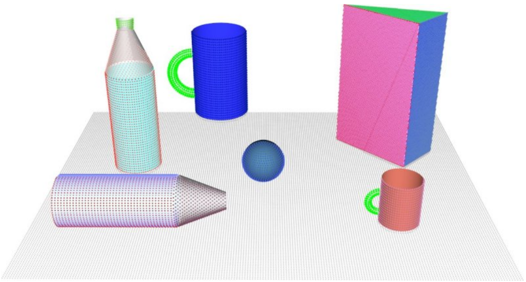


Fig. 6. A synthetic scene demonstrating the different estimated primitive shape models.

When a model s_i has been found, we perform a non-linear optimization of the shape parameters using the Levenberg-Marquard algorithm. For some shapes, like cones, it proved beneficial to use a more restrictive distance threshold in the first inlier selection stage, and adjust this threshold to the expected noise level during the final inlier search. While this reduces the number of false positives and increases the number of false negatives, the refitting step is optimizing the shape parameters based on better positive points and converges to a better solution.

After extracting a shape, we perform a search for possible object *handles* by inspecting the vicinity of the model for fairly large clusters of points that can be grown from the current shape surface. These points get marked as handles to the current shape and are disregarded in the remaining model candidate list. We found this model-less approach to work more reliable than trying to fit specific geometric shapes like torii, since handle geometry can vary greatly. An example is given in Figure 6 where we show results for a synthetic scene. Note that only the point positions are synthetically generated, as normals are estimated, and no information on the original shape parameters is available to the object detection.

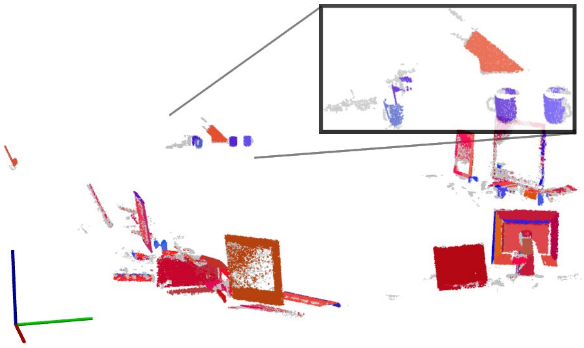


Fig. 7. Cylindrical (blue) and planar (red) shape candidates detected on tables. All remaining points are shown in gray.

Figure 7 presents the results of cylinder and plane fitting on the point clusters from Figure 5. We noticed that these two models appear the most often in our kitchen datasets. The different cylinders are marked with different shades of blue, and the planes with shades of red. Notice that we did not constrain the cylinder radius to a specific range, and therefore several small edges appear as cylinders due to the properties of our normal estimation algorithm. However, since the model parameters are known, these shapes can be grouped together and removed if necessary.

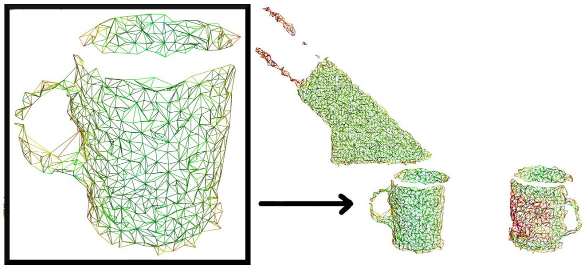


Fig. 8. Example of triangulated surface models for the kitchen counter point clusters (see Figures 5 and 7), shown in intensity (red shades).

Our triangulation method [2] propagates an advancing front of triangles, adding new triangles at each vertex by a local projection of the nearby points onto a supporting plane and creating a mesh out of them. Figure 8 presents an example of surface models created for the objects in Figure 7. The hybrid model is then created by triangulating the remaining outliers from the model fitting step, and adding it to the shape coefficients model: $\mathcal{M} = \mathcal{S} \cup \mathcal{O}$.

VI. Discussions and Experimental Results

We ran our modeling pipeline on multiple datasets representing table setting scenes, acquired in our kitchen laboratory. Each of the computational steps described in Section III was ran on every scene, and the results were inspected by comparing the estimated shape models with real world measurements. In Table I we present the results obtained for 4 different datasets. From top to bottom, the table presents: i) the table and object cluster segmentation from a raw point cloud dataset; ii) the shape model segmentation; iii) the surface models obtained by triangulating the points; and finally iv) the hybrid shape-surface models.

Even though the above results were obtained by applying our segmentation and model fitting methods on individual point clusters, the algorithms work directly for cluttered scenes where clustering is not possible due to all object surfaces being connected. Figure 9 shows the shape classification results for a cluttered table scene. Most shapes are reliably detected even if partially occluded.

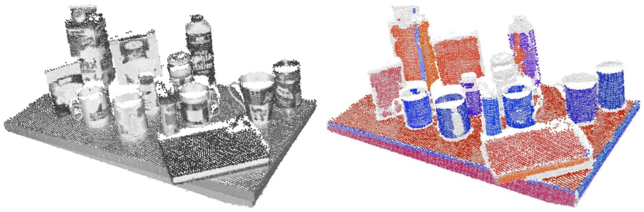


Fig. 9. Shape classification and fitting results for a cluttered scene without a priori object clustering: cylinders in blue, planes in red, and the remaining points in gray.

Figure 10 presents an important limitation of our mapping approach. Due to the fact that our sensing devices cannot return any measurements from shiny black surfaces such as the stove cooktop or the kitchen sink, it is impossible to determine the fact that there are any planar areas which could support objects there. Therefore, the horizontal table planar support is incomplete (as seen in Figure 10), and some points which could be part of some objects will be left out, as their footprint (support) on the table is very small or inexistent. We are currently investigating ways of fusing multiple sensor modalities together to improve the results.

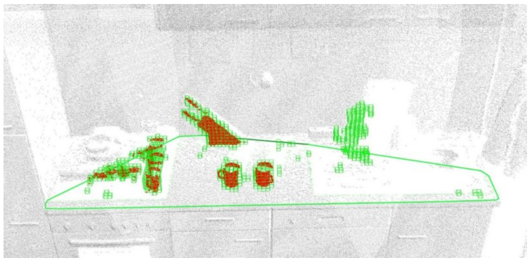
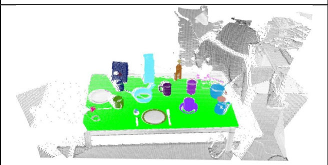
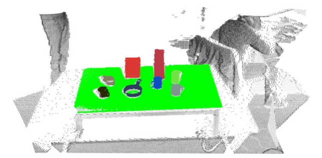
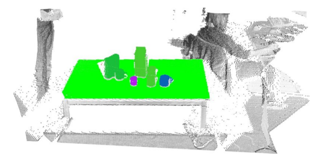
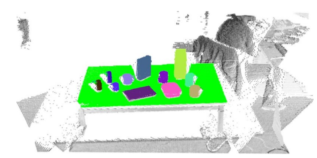
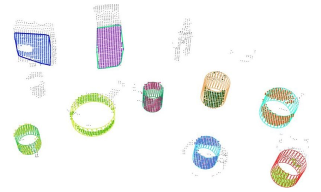
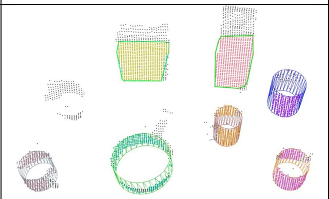
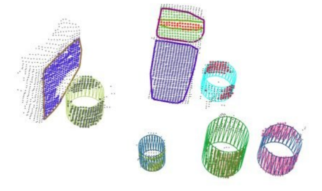
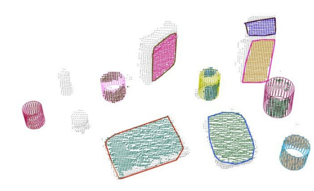
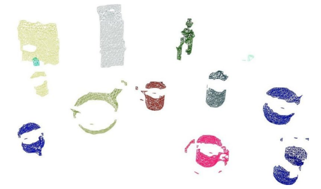
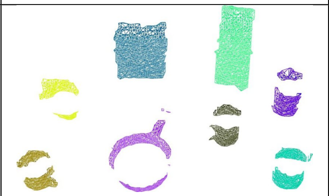
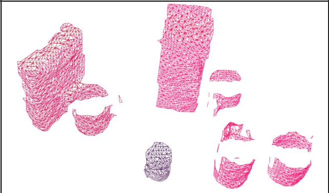
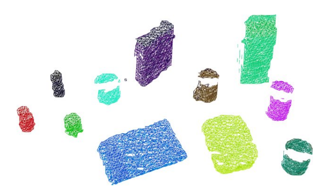
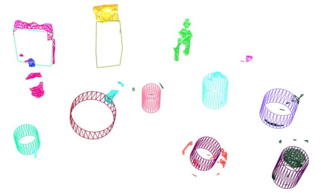
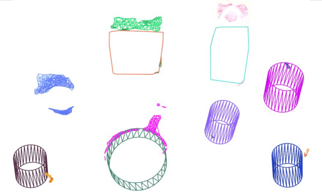
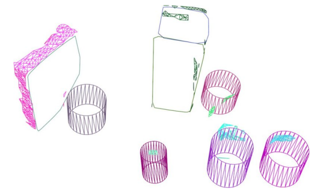
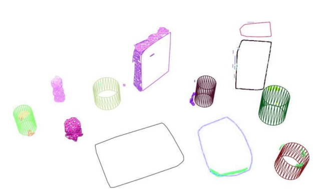


Fig. 10. Object clustering based on octree connectivity segmentation on a planar horizontal surface (i.e. table). Only points in red are selected as object candidates, as the rest does not fulfill the object footprint criterion.

TABLE I

MODEL FITTING RESULTS FOR 4 DIFFERENT TABLE SETTING SCENES. FROM TOP TO BOTTOM: TABLE AND OBJECT CLUSTER SEGMENTATION, SHAPE MODELS, SURFACE MODELS, AND FINALLY THE HYBRID SHAPE-SURFACE MODELS.

	Scene 1	Scene 2	Scene 3	Scene 4
Segmentation				
Shape coefficients				
Surface model				
Hybrid shape-surface				

VII. CONCLUSIONS

In this paper we presented a comprehensive system for the acquisition of 3D hybrid shape-surface geometric models in close-range scenes useful for mobile manipulation and grasping [1]. The models are acquired out of sensed point cloud data using robust shape segmentation and surface triangulation methods. By fitting primitive geometric shapes to the data, our framework is able to reconstruct and infer missing data, and thus improve the resultant models for grasping. Furthermore, by splitting the object data into clusters, we create a decoupled triangular mesh map, which needs fewer computational updates in the presence of environment dynamics.

Acknowledgements This work was supported by the CoTeSys (Cognition for Technical Systems) excellence cluster.

REFERENCES

[1] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, “Towards 3D Point Cloud Based Object Maps for Household Environments,” *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge)*, 2008.

[2] Z. C. Marton, R. B. Rusu, and M. Beetz, “On Fast Surface Reconstruction Methods for Large and Noisy Datasets,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, 2009.

[3] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, “Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation,” in *IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, 2009.

[4] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic Grasping of Novel Objects using Vision,” *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[5] G. Bone, A. Lambert, and M. Edwards, “Automated Modeling and Robotic Grasping of Unknown Three-Dimensional Objects,” in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, USA*, 2008.

[6] R. Mario and V. Markus, “Grasping of Unknown Objects from a Table Top,” in *Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments*, 2008.

[7] A. Miller and P. K. Allen, “Graspit!: A Versatile Simulator for Robotic Grasping,” *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[8] R. Schnabel, R. Wahl, and R. Klein, “Efficient RANSAC for Point-Cloud Shape Detection,” *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, June 2007.

[9] F. Bley, V. Schmirgel, and K.-F. Kraiss, “Mobile Manipulation Based on Generic Object Knowledge,” in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, 2006.

[10] G. Biegelbauer and M. Vincze, “Efficient 3D Object Detection by Fitting Superquadrics to Range Image Data for Robot’s Object Manipulation,” in *IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy*, 2007.

[11] Y. Zhang, A. Koschan, and M. Abidi, “Superquadric Representation of Automotive Parts Applying Part Decomposition,” *Journal of Electronic Imaging, Special Issue on Quality Control by Artificial Vision, Vol. 13, No. 3*, pp. 411–417, 2004.

[12] A. Nuechter and J. Hertzberg, “Towards Semantic Maps for Mobile Robots,” *Journal of Robotics and Autonomous Systems (JRAS), Special Issue on Semantic Knowledge in Robotics*, pp. 915 – 926, 2008.

[13] P. Torr and A. Zisserman, “MLESAC: A new robust estimator with application to estimating image geometry,” *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[14] C. Ericson, *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3D Technology)*, December 2004.

[15] J. Arvo, “A Simple Method for Box-Sphere Intersection Testing,” in *Graphics Gems*. Academic Press Professional, Inc., 1990.

[16] T. L. Kay and J. T. Kayjia, “Ray tracing complex scenes,” in *Computer Graphics (SIGGRAPH ’86 Proceedings)*, vol. 20, 1996.