Automatic Object Modeling Through Integrating Perception and Robotic Manipulation

Zhou Teng[1,2], Huitan Mao[1], and Jing Xiao[1(✉)]

[1] Department of Computer Science, UNC Charlotte, Charlotte, NC, USA
xiao@uncc.edu

[2] ABB US Corporate Research Center, Bloomfield, CT, USA

Abstract. In this paper, we introduce a method to build 3D object models from RGB-D images automatically by interleaving model building with robotic manipulation. Using a fixed RGB-D camera and starting from the first view of the object, our approach gradually builds and extends a partial model (based on what has been visible) into a complete object model. In the process, the partial model is also used to guide a robot manipulator to change the pose of the object to make more surfaces visible for continued model building. The alternation of perception-based model building and pose changing continues until a complete object model is built with all object surfaces covered. The method is implemented, and experimental results show the effectiveness and robustness of this approach.

Keywords: Automatic object modeling · RGB-D imaging · Robotic manipulation

1 Introduction

Many robotic manipulation tasks require the knowledge of 3D object models of the target objects. How to build a 3D object model based on geometric and visual appearance of an object is thus an important problem. 3D object modeling has been studied for a long time with many methods introduced by researchers. Existing literature can be classified in two categories: *passive* approaches [1,2], where cameras and viewpoints are fixed, and *active* approaches, where cameras are moved by either people [3,4] or robots [5]. Robot-guided object modeling is also referred to as a view planning problem [6], which is focused on finding a set of views to enable reliable and accurate object reconstruction. Model-based algorithms aim to find certain optimal set of views [7,8], while non-model based algorithms obtain more object information and reduce uncertainty view by view [9–13]. However, with the target object static, some part of the object cannot be viewed and modeled, such as the bottom surface of the object on a table.

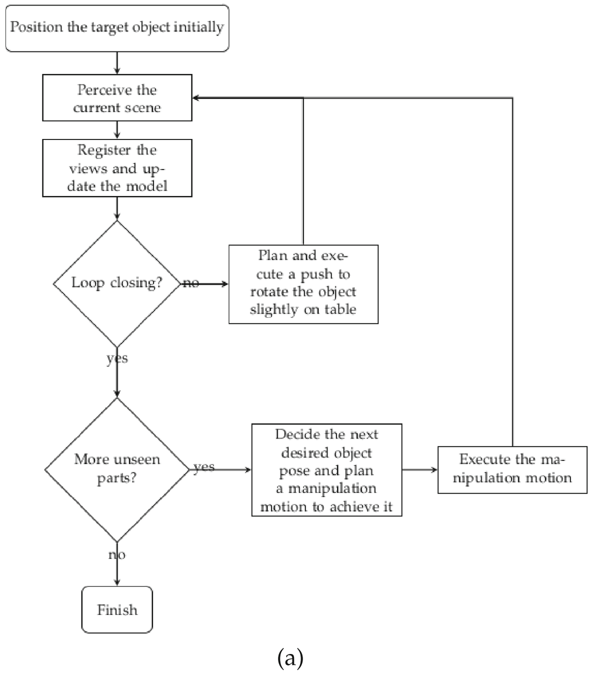
This work was supported by the NSF grants IIP-1432983, IIP-1439695, EPRI, and ABB.

Some recent work also involves changing object poses through robotic manipulation for object modeling [14–16]. However, with the end-effector holding the object, it always occludes the object, especially if it is of similar size to the object. How to separate the end-effector from the object in images is an issue. Moreover, if some parts of the object are always occluded in all feasible grasps, a complete object model cannot be achieved. Also, assuming the object is held by the manipulator before model building implies human intervention.

In general, object model-building in exiting work often requires manual operations or interventions in order to build a complete model that covers all surfaces of an object. Automatic 3D model building for objects of arbitrary shapes remains a challenging problem not yet solved.

2 Overview of Our Approach

In this paper, we address automatic building of a complete 3D model of a rigid object by covering all its surfaces. Our approach is characterized by progressive and interleaving RGB-D perception and manipulation to gradually build the entire object model. In our modeling environment, an RGB-D camera is placed in a fixed position with a fixed direction to view the target object, which is placed on a table.


Position the target object initially
Perceive the current scene
Register the views and update the model
Loop closing?
no
Plan and execute a push to rotate the object slightly on table
yes
More unseen parts?
yes
Decide the next desired object pose and plan a manipulation motion to achieve it
Execute the manipulation motion
no
Finish
(a)

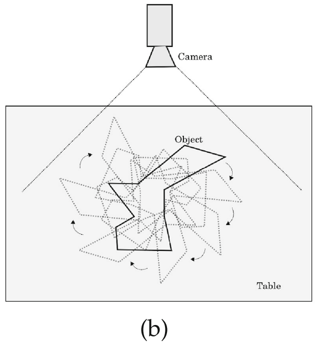


Fig. 1. Overview of automatic object modeling: (a) high-level flowchart; (b) illustration of object rotation on table (top view)

Figure 1(a) provides a high-level flowchart of our approach. In the inner loop, the manipulator rotates the target object on table (in one direction) step by step through pushing until about 360° to cover side surfaces of the object. After each push step, an image is taken and used to build the object partial model. After the object is rotated 360°, all aspects/views [17] of the side surfaces of the object are captured, as shown in Fig. 1(b). In the outer loop, the manipulator changes the support surface of the object, i.e., rotate the object to expose its previous bottom surface, which can be achieved by a greater push or by picking up the object, rotating it in the air, and placing it down (on a different surface), in order to cover unseen surfaces. In the following, the related algorithms are introduced in detail.

3 Image Registration in Perception

Image registration serves two main purposes in our object modeling process: one is to merge the currently sensed object point cloud (through the RGB-D camera) into the gradually built point cloud of the (partial) object model; the other is to obtain the accurate pose (i.e., position and orientation) of the target object in the scene to enable object manipulation.

Image registration is conducted in the following three ways in our approach.

Pairwise Registration of Neighboring Object Point Clouds: We use the ICP algorithm [18] to register neighbouring object point clouds obtained from RGB-D images taken at two adjacent steps from the same 360° rotation loop. If sufficient ASIFT [19] keypoints can be matched between the two RGB-D images first, compute a transformation matrix [20] based on their 3D coordinates as the initial estimate; otherwise, use an identity matrix as the initial estimate. With an initial transformation estimate from the keypoints matching, the ICP algorithm converges much faster and also achieves registration results in better quality. To make the registration more robust, we also consider additional factors, such as the color similarity and the point-to-plane distance, in evaluating the quality of matched point pairs in the ICP algorithm.

Global Optimization of Pairwise Registration Results: In order to reduce the accumulated registration error from pairwise registration, our approach further registers all object point clouds obtained in all steps of the rotation from the same 360° rotation loop using a global optimization algorithm, developed based on the virtual mate approach [21].

Registration of Partial Object Models from Different Rotation Loops: Once an optimized partial object model is obtained from each 360° object rotation loop, the next task is to register such partial models from different rotation loops. We still apply the ICP algorithm. The transformation matrix for changing the object pose from the robotic manipulation is used as the initial estimate of the transformation for the ICP algorithm.

4 Manipulation for Perception

In the following sections, we describe the details of the algorithms used for the two kinds of manipulation outlined in Fig. 1(a).

4.1 Pushing

Given that the target object sits stably on a table (or some other support structure), after the first RGB-D image of the object is taken, our strategy is to detect the leftmost edge from the object point cloud C_1 and the corresponding leftmost surface segment and choose a point with position $\mathbf{u_1}$ on the surface segment that is close to the middle of the leftmost edge as the point on the object for pushing. The direction for pushing $\mathbf{v_1}$ is oppose to the normal of the surface segment. Both $\mathbf{u}_1$ and $\mathbf{v}_1$ are with respect to the frame O_1.

In general, given a point cloud C_{i+1}, i=1, ...,m, the pushing position and direction $\mathbf{u_{i+1}}$ and $\mathbf{v_{i+1}}$ with respect to frame O_{i+1} can be computed by applying the transformation ${}^{O_{i+1}}T_{O_i}$ to $\mathbf{u}_i$ and $\mathbf{v}_i$ respectively.

4.2 Change of Object Support Surfaces

After each 360° rotation loop, our approach simultaneously determines:

1. if new object surfaces can be observed by changing the support surface of the object, and how to change the support surface;
2. if no new object surface can be found so that the model building process can be terminated.

The basic idea of our approach is to form a finite set of candidate tasks for changing object support surfaces based on considering all possible sides of the oriented bounding box (OBB) [22] of the object and evaluate each candidate task based on a set of constraints/criteria to either (1) find the best task to execute or (2) to discover that the termination condition for model building is satisfied.

Candidate Tasks for Changing Object Support Surface. There are five available OBB sides above the table that can be considered for the selection of a new support surface. Without losing generality, as shown in Fig. 2, we denote all OBB sides based on their positions in the OBB frame as the $+X$, $-X$, $+Y$, $-Y$, $+Z$, and $-Z$ OBB sides.

A robotic task to change the current object support surface to a new support surface can be described at a high level in terms of two available sides of the object OBB involved: one side S_1 defines the approach vector of the robot hand/gripper by its normal pointing into the object, and the other side S_2 is the side for the new support surface. Thus, we denote a high-level candidate task simply as S_1S_2, and there are a total of 17 high-level candidate tasks: $+X-X$, $+X+Z$, $-X+X$, $-X+Z$, $+Y-Y$, $+Y+Z$, $-Y+Y$, $-Y+Z$, $+Z+X$, $+Z-X$, $+Z+Y$, $+Z-Y$, $+X+X$, $-X-X$, $+Y+Y$, $-Y-Y$, and $+Z+Z$.

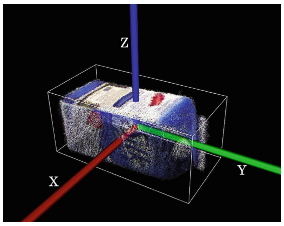


Fig. 2. Oriented bounding box based the partial model of the object *milk box blue*

Evaluation of Candidate Tasks. In order to choose the best candidate task for execution, our approach evaluates the result of each candidate task through simulation: for the simulated object on the new support OBB side (specified by the candidate task), our algorithm virtually moves the camera around the object (instead of rotating the object) to capture the simulated view of the object after each rotation step via the ray tracing algorithm [23].

We further use a voxel map [13,23,24], to organize the existing observed information of the object within its current OBB and to facilitate evaluation of the simulated observation of the object for each candidate task. The voxel map discretizes the OBB into a grid of voxels, where each voxel can be in one of five states depending on whether and how it is observed: *unlabeled*, *empty*, *occupied*, *occluded* and *occplane* [13].

Our approach then evaluates each candidate task based on the usefulness of the observed information after the change of the object support surface through checking if there is enough overlap between new and existing object point clouds and also if there are new surfaces in the new object point cloud [13]: the occupied voxels observed in simulated camera views (from the candidate task) indicate overlap surfaces, while the occplane voxels observed indicate new surfaces. If no occplane voxels exist or can be further observed any more, our algorithm terminates the model building process. We also consider the model quality in the evaluation [25].

Our approach also evaluates the following factors to assess how feasible a candidate task can be executed:

- if the target object can be grasped from the given approached OBB side;
- if an approximately flat surface s exists for the given OBB side for support, and if the target object can stand stably on the surface s;
- if the target object can be pushed effectively with the new support surface;
- if the manipulation utility is sufficient.

If all the above conditions can be satisfied, the candidate task is considered executable.

Theoretically, to make an object stable, two conditions must be satisfied: the support surface should define a surface region that has a sufficiently large area, and the projection of the center of gravity of the object on the surface of the support region falls inside the support region. In our approach, we use the center of the OBB as the estimated gravity center and look for a nearly flat surface based on the support surface side of the OBB, in which the projection of the center of the OBB falls.

We also introduce a threshold as the minimum object width, beyond which the object can be rotated by pushing effectively.

To evaluate if the target object can be successfully grasped from the considered approach vector, we use a simple gripper model with two fingers, and check if two approximately parallel surfaces (based on their surface normals) on the two contacted OBB sides can be found for grip; we also check if the approached OBB side is too wide to fit into the gripper.

The manipulation utility is used to evaluate the path cost of the robotic motion for changing the object support surface, given a candidate task. We use a mathematical function [13] that is monotonically decreasing with respect to the path cost for the evaluation. The greater the path cost, the smaller the numerical value of the manipulation utility.

To evaluate all the candidate tasks for the change of the current object support surface, we compute a single value Θ that combines all the factors described above with a combination strategy similar to [13,24]. If Θ of a candidate task equals zero, then it is an invalid task. Based on all the valid candidate tasks, we choose the one with the greatest Θ for execution.

Note that when Θ becomes zero for all candidate tasks, our modeling procedure is terminated.

5 Experiments and Analyses

Figure 3 shows the experimental set up for testing our approach for automated 3D object modeling. The target object is positioned on a table in the center of view of the Microsoft Kinect camera. Some colorful pages on the table are used as landmarks for the calibration between the robot and camera coordinate systems. A Barrett WAM of 7 degrees of freedom (DOFs) with a hand of 4 DOFs are used for object manipulation.

5.1 Experiments

The two important procedures in our automatic model-building process, the 360° rotation loop by pushing and the change of object support surfaces, are illustrated as follows, using the object *milk box blue* as an example.

Figures 3 and 4 show snapshots of the procedures for pushing and support surface change of the object *milk box blue* respectively during model building. Note that the chosen task in this example used the top side of the object *milk box blue* in Fig. 4(a) as both the approached side for grasping and the new support surface side, and thus, it was completed in two consecutive steps.

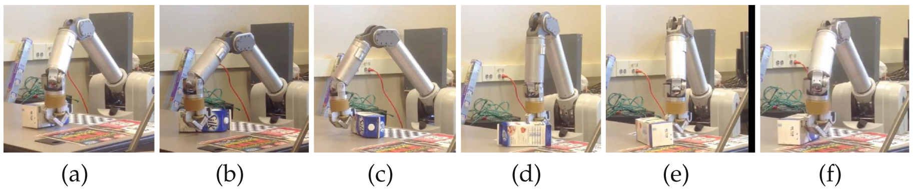


Fig. 3. Snapshots of the pushing procedure of the object *milk box blue*, step by step in the 360° rotation loop

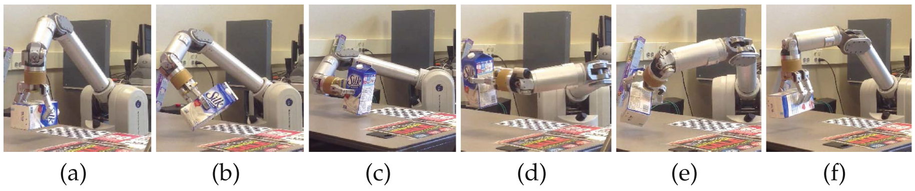


Fig. 4. Snapshots of the procedure of changing the support surface of the object *milk box blue*

5.2 Comparing Registration Results Before and After Optimization

Figure 5 compares the registration results before and after global optimization, using all the point clouds of the object *milk box blue*, captured from a 360° rotation loop. It shows that all the distorted parts of the object surfaces in red circles are corrected after global optimization.

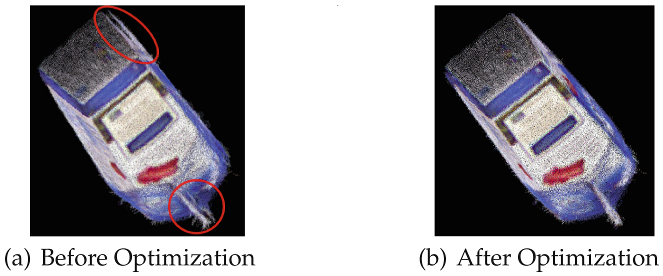
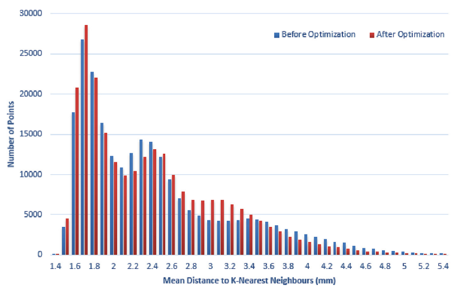
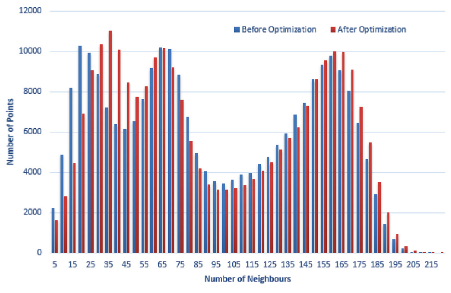


Fig. 5. The registration results before and after global optimization of the object *milk box blue* from a 360° rotation loop

Figure 6(a) compares two histograms of the mean K-nearest ($K = 100$) neighbor distance of the object point cloud before and after global optimization: many points have their mean K-nearest neighbor distances reduced from the range 1.8 mm∼2.4 mm to the range 1.4 mm∼1.7 mm and from the range 3.5 mm∼5.4 mm to the range 2.5 mm∼3.4 mm.



(a)



(b)

Fig. 6. Comparison of histograms of the partial models of the object *milk box blue* before and after global optimization, as shown in Fig. 5: (a) comparison of histograms of the mean K-nearest neighbor distances ($K = 100$); (b) comparison of the histograms of the number of the neighbors within the radius of 3 mm

Figure 6(b) compares two histograms of the number of neighbors within the radius of 3 mm of the object point cloud before and after global optimization: many points have their numbers of neighbors within the radius of 3 mm increased from the range 5∼25 to the range 30∼60 and from the range 65∼150 to the range 155∼215.

Reduced mean K-nearest neighbor distance and increased number of neighbours in the respective histograms indicate that the points in the object point cloud are distributed more closely together after global optimization, which means that the registered point clouds are better aligned after global optimization.

5.3 Analyses of Voxel Maps

The complete model of the object *milk box blue*, as shown in Fig. 7, is built using two 360° rotation loops. All the information of the voxel map after each 360° rotation loop is given in Table 1. In our experiments, the voxel size is set as 3 mm × 3 mm × 3 mm.

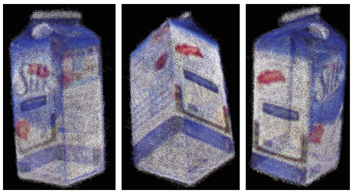


Fig. 7. The built complete model of *milk box blue*, visualized from different viewpoints

As we can see from Table 1, after two 360° rotation loops, there are only a few occplane voxels left in the voxel map, and the model of the object *milk box* *blue* is almost complete. From the 1st loop to the 2nd loop, occplane voxels are greatly reduced, indicating the great reduction of the unobserved region after the 2nd loop. This is also reflected in the significantly increased number of occupied voxels. However, the number of increase in occupied voxels is much greater than the number of reduction in occplane voxels; this is because the occupied voxels might come from both the previously occluded and occplane voxels. Besides, due to the limited depth accuracy of the Microsoft Kinect camera and the sensing noise, after image registration from multiple views, the object surface is usually thicker than one voxel, meaning that more occupied voxels are generated. The few remain occplane voxels might be caused by the sensing inaccuracy and noise, or tiny self-occluded surfaces which cannot be observed by more 360° rotation loops.

Table 1. The voxel map after each 360° Rotation Loop for *milk box blue*

Loop no.	Size of OBB (mm^3)	Size of voxel map	#Empty	#Occupied	#Occluded	#Occplane
1	128 × 257 × 119	45 × 88 × 42	72299	21275	70237	2509
2	128 × 261 × 123	45 × 90 × 44	86826	30902	60448	24

Figure 8 shows built models of different daily objects, with their corresponding OBBs. Note that even for objects with irregular shapes, such as the object *spray*, there are still smooth surfaces corresponding to some OBB sides that can be used as support surfaces in our approach.

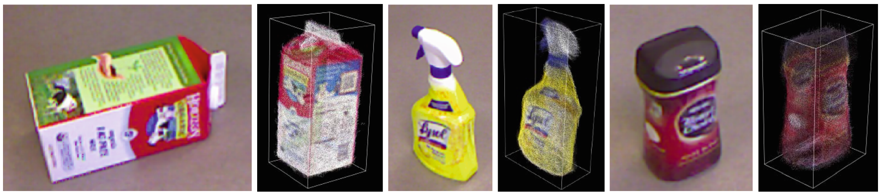

(a) *milk box red* (b) (c) *spray* (d) (e) *coffee box* (f)

Fig. 8. Built models of daily objects and their corresponding OBBs

6 Conclusions

We have introduced a method to build 3D object models from RGB-D images automatically with robotic manipulation. With an automatic push procedure, our approach does not assume that the robot can achieve a first grasp of the object somehow [14] without the knowledge of the object model. After observing many side surfaces of the object through the push loop, the partial object model built can guide automatic grasping for changing the object support surface. The experimental results also show that our approach is robust to motion uncertainty in pushing. It provides a promising solution for automatic 3D rigid object modeling in situations where a camera cannot be moved and re-positioned freely, for example, in cluttered environments, and where objects cannot be pre-positioned on some special hardware (such as a turntable) or pre-grasped.

Our approach of interleaving perception with manipulation provides more flexibility to enable observing all object surfaces and building a complete object model. Such strategy works well for many daily objects. However, for objects with many concavities and extremely complicated self-occlusion, such as a sculpture with many small details, further integrating the approaches [14–16] can help complete the coverage for object modeling.

References

1. Choi, C., Christensen, H.I.: 3D pose estimation of daily objects using an RGB-D camera. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3342–3349, October 2012
2. Singh, A., Sha, J., Narayan, K.S., Achim, T., Abbeel, P.: BigBIRD: A large-scale 3D database of object instances. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 509–516, May 2014
3. Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P., Scopigno, R.: Protected interactive 3D graphics via remote rendering. ACM Trans. Graphics (TOG) **23**(3), 695–703 (2004)
4. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 127–136 (2011)
5. Connolly, C.: The determination of next best views. In: ICRA, pp. 432–435 (1985)
6. Scott, W.R., Roth, G., Rivest, J.F.: View planning for automated 3D object reconstruction inspection. ACM Comput. Surv. (CSUR) **35**(1), 64–96 (2003)
7. Tarbox, G., Gottschlich, S.: Planning for complete sensor coverage in inspection. Comput. Vis. Image Underst. **61**(1), 84–111 (1995)
8. Bowyer, K., Dyer, C.: Aspect graphs: an introduction and survey of recent results. Int. J. Imaging Syst. Technol. **2**(4), 315–328 (1990)
9. Maver, J., Bajcsy, R.: Occlusions as a guide for planning the next view. IEEE Trans. Pattern Anal. Mach. Intell. **15**(5), 417–433 (1993)
10. Massios, N., Fisher, R.: A best next view selection algorithm incorporating a quality criterion. In: British Machine Vision Conference, pp. 780–789 (1998)
11. Foissotte, T., Stasse, O., Escande, A., Wieber, P., Kheddar, A.: A two-steps next-best-view algorithm for autonomous 3D object modeling by a humanoid robot. In: ICRA, pp. 1159–1164 (2009)
12. Kriegel, T., Bodenmuller, T., Suppa, M., Hirzinger, G.: A surface-based next-best-view approach for automated 3D model completion of unknown objects. In: ICRA, pp. 4869–4874 (2011)
13. Vasquez-Gomez, J., Sucar, L., Murrieta-Cid, R., Lopez-Damian, E.: Volumetric next-best-view planning for 3D object reconstruction with positioning error. Int. J. Adv. Rob. Syst. **11**(159), 1–13 (2014)

14. Krainin, M., Curless, B., Fox, D.: Autonomous generation of complete 3D object models using next best view manipulation planning. In: ICRA, pp. 5031–5037, May 2011
15. Ma, L., Ghafarianzadeh, M., Coleman, D., Correll, N., Sibley, G.: Simultaneous localization, mapping, and manipulation for unsupervised object discovery. In: ICRA, pp. 1344–1351 (2015)
16. Browatzki, B., Tikhanoff, V., Metta, G., Bulthoff, H., Wallraven, C.: Active in-hand object recognition on a humanoid robot. IEEE Trans. Robot. **30**(5), 1260–1269 (2014)
17. Plantinga, H., Dyer, C.R.: Visibility, occlusion, and the aspect graph. Intl. J. Comput. Vis. (IJCV) **5**(2), 137–160 (1990)
18. Besl, P., McKay, H.: A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)
19. Yu, G., Morel, J.M.: ASIFT: an algorithm for fully affine invariant comparison. In: Image Process. On Line **1**, 1–28 (2011)
20. Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-D rigid body transformations: a comparison of four major algorithms. Mach. Vis. Appl. **9**(5–6), 272–290 (1997)
21. Pulli, K.: Multiview registration for large data sets. In: 1999, Proceedings, Second International Conferences on 3-D Digital Imaging and Modeling, pp. 160–168 (1999)
22. Gottschalk, S.: Collision queries using oriented bounding boxes. Ph.D. thesis, The University of North Carolina at Chapel Hill (2000)
23. Klingensmith, M., Hermann, M., Srinivasa, S.: Object modeling and recognition from sparse, noisy data via voxel depth carving. In: International Symposium on Experimental Robotics (ISER) (2014)
24. Sanchiz, J., Fisher, R.: A next-best-view algorithm for 3D scene recovery with 5 degrees of freedom. In: British Machine Vision Conference, pp. 163–172 (1999)
25. Albalate, M.T.L., Devy, M., Miguel, J., Marti, S.: Perception planning for an exploration task of a 3D environment. In: Pattern Recognition, 2002. In: 16th International Conference on Proceedings, vol. 3, pp. 704–707 (2002)