Autonomous 3D model generation of unknown objects for dual-manipulator humanoid robots

Adrian Llopart[1,2], Ole Ravn[1], and Nils A. Andersen[1] and Jong-Hwan Kim[2]

[1] AUT Group, Department of Electrical Engineering, DTU
Anker Engelunds Vej 1, 2800 Kgs. Lyngby, Denmark
{adllo,or,naa}@elektro.dtu.dk
[2] RIT Lab, Department of Electrical Engineering, KAIST
291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea
johkim@rit.kaist.ac.kr

Abstract. This paper proposes a novel approach for the autonomous 3D model generation of unknown objects. A humanoid robot (or any setup with two manipulators) holds the object to model in one hand, views it from different perspectives and registers the depth information using a RGB-D sensor. The occlusions due to limited movement of the manipulator and the gripper itself covering the object are avoided by switching the object from one hand to the other. This allows for additional viewpoints leading to the registration of more depth information of the object. The contributions of this paper are as follows: **1.** A humanoid robot that manipulates objects and obtains depth information **2.** Tracing the hand movements with the robots head to be able to see the object at every moment **3.** Filtering the point clouds to remove parts of the robot from them **4.** Utilizing the Normal Iterative Closest Point algorithm (depth points, surface normals and curvature information) to register point clouds over time. This method will be applied to those pointclouds that include the robots gripper for optimal convergence; the resultant transform is then applied to those point clouds that describe only the segmented object **5.** Changing the object from one hand to another **6.** Merging the resulting object's partial point clouds from both the left and right hands **7.** Generating a mesh of the object based on the triangulation of final points of the object's surface.
No prior knowledge of the objects is necessary. No human intervention nor external help (i.e visual markers, turntables ...) is required either.

Keywords: humanoid robot, 3D model creation, point cloud processing

1 Introduction

With the increase of social and service robotics, the demand for Human-Robot-Object collaboration has risen considerably. Therefore, robots necessarily have to understand their surroundings to be able to interact with them. Over the past years, special emphasis has been put on robots capable of adjusting to dynamically changing environments, especially when dealing with object recognition

and manipulation. Novel research has been proposed for a more rapid and precise detection of known objects. Despite this, robots must also be able to cope with unknown objects: being able to model them becomes a key feature for faster detection, recognition and manipulation in the future.

1.1 Related work

Many approaches to object recognition deal with identifying the 6 DoF pose estimation of the object based on the correspondence grouping of a set of points with a previously generated model [1], [2]. Other approaches use the synthetic data of 3D models to train Convolutional Neural Networks (CNN) for object detection [3], [4], [5]. Finally, some research has been done lately on the generation of grasping poses based solely on the object's point cloud or model [6]. Consequently, a previous 3D model of the model must be known.
Generating 3D models from unknown objects can be accomplished in many ways, each of which has their own advantages and inconveniences. The objects placement, when generating the model, can be divided into three main groups: **a.** static objects with the camera moving around it, **b.** static camera with object on top of a turntable which is rotated, **c.** non-static object and camera.
Concerning the first type, the major issue arises when transforming the camera poses as it revolves around the object. The usage of markers is a widespread solution but requires human intervention, not only for positioning the markers but also to move the camera. The generation of the model can then be achieved by ray-casting the objects silhouette from every view onto a 3D regular grid (volumetric image) as proposed by Denkowski [7]. A more common approach when using markers is to apply an Iterative Closest Point (ICP) algorithm to the point clouds extracted from the depth images of every viewpoint [8].
When dealing with the second type of object placement, namely doing so on a turntable with a fixed camera, the problem with a moving frame disappears. However, the necessity for a human or any other external agent to spin the turntable limits the autonomy of the model generation. Once again, to keep track of the objects viewpoint (in other words, how much the table has rotated), some approaches continue to use markers [9], whilst others rely on SIFT features [10] or the ICP algorithm with loop closure [11].
The last objects placement, whilst being the most difficult to track, allows for the maximum autonomy. Particularly, assuming a robot that wants to autonomously model unknown objects it has recently grasped, to be later used for a faster object recognition and detection. In this case, the objects frame will be inconstant (because the manipulator and gripper holding the object will be moved to try and view it from different perspectives), and so will be the camera frame (presuming the camera is mounted on the head of the robot, it needs to move to be able to track the manipulator's end effector and the object). Krainin *et. al.* [12] propose the usage of a Kalman filter for the camera to keep track of the gripper plus utilizing a modified ICP algorithm (that takes into account sparse feature matching, dense color matching and prior state information) and loop

closure to generate smooth object models. Even though this methodology could clearly be considered *state-of-the-art* in terms of autonomously generating 3D models, it still requires the help of external agents: the problem of grasping an object with a manipulator is that parts of the object will always be occluded. For this reason, the paper suggests leaving the object on a table or any other surface and regrasp it from another position to finish registering the point clouds.

1.2 Basic methodology

The approach proposed in this paper will try to make the registration process of unknown objects as autonomous as possible. The only assumption required is that the robot has an unknown object grasped in its gripper. Plenty of research has been conducted in this field; specifically, a method to achieve this has been previously proposed by Llopart *et. al.* [13].
The pipeline starts with a dual-manipulator humanoid robot holding an object in one hand. The arm and gripper will be moved in a way that **a.** the end effector will always be inside the field of view of the robot whilst the head is also tracking the object (Section 2), **b.** the difference between end poses is very small to allow better NICP convergence (Section 3.2), **c.** the total amount of viewpoints will try to cover the entire 360° around the object.
The RGB-D sensor located in the robot's head will provide depth images for each step, which are then converted to point clouds. The point clouds are then filtered with respect to the distance to the RGBD sensor and to the robot's surface (Section 3.3). A radius sparse outlier process will also take place. The resulting point clouds will then be transformed into the grippers frame to become invariant to movement (Section 3.4). The point cloud's normals and curvatures will be estimated and used during the NICP registration (Section 3.5). One of the major problems previous approaches had was that convergence failed when registering symmetrical or low-detailed objects. To solve this, the NICP is done to the point clouds that include those parts of the robot that are invariant to the object, i.e. the grippers. This allows for additional information and evident better results. The obtained transformations are then applied to the point clouds that represent only the object. The occlusions and lack of viewpoints that occur with one manipulator can be solved by changing the object from one hand to another, and carrying out the same procedure again (Section 3.6). This results in two partial models of the object. These two point clouds must be merged to obtain one complete model (Section 3.7). Finally, a Moving Least Squares (MLS) algorithm is carried out to smoothen the surfaces of the model and remove artifacts. A mesh of it is then built based on a greedy surface triangulation algorithm (Section 3.8).
The experimental setup, results and discussion is found in Section 4. Conclusions and future work are dealt with in Sections 5 and 6, respectively.

2 Tracking the gripper's movements

The key concept proposed in this paper is registering depth information of the object from different perspectives. To achieve this, the robot rotates and translates the object continuously. Thus, it becomes essential that the robot can follow the movements of the end effector instantly with its head, where the depth sensor is located.
Krainin *et. al.* [12] propose the usage of a Kalman filter, taking as input the previous time step mean and covariance, the clouds representing the manipulator and object, and joint angles reported by the encoders of the manipulator.
A different and much simpler approach to achieve gripper tracking is to constantly monitor the position (x, y, z) of both end effectors (left and right grippers). This information must be then transformed into two angles: pan and tilt, corresponding to the 2 DoF the robot's head has. Two simple equations solve this problem:

$$pan\_angle = atan2(y, x) \tag{1}$$

$$tilt\_angle = atan2(z, \sqrt{x^2 + y^2}) \tag{2}$$

This allows the robot to rapidly switch from following the left end-effector to the right one by simply changing the input coordinates to those of the selected gripper.

3 Model creation

This section describes the core concepts of the proposed methodology: the disabling of collision checking between manipulator and object, the multiple filtering steps of the depth data from the RGB-D sensor, the transformation to the grippers frame, the registration of the multi viewpoint data to generate the objects model, changing the object from one hand to another to create two semi-full models of the object, their merging procedure and, finally, the creation of a mesh for the model.

3.1 Disabling collision checking

Before the proposed pipeline can be executed, the collision checking between the objects point cloud and the robots links must be removed. The *MoveIt* libraries for ROS will be used to control the manipulators and to set their poses. An octomap of the environment will be continuously generated to evaluate whether an action can be executed or if the movement will end up colliding with the surroundings. The only issue with this approach is that the object's point cloud (which will be held by the robot) will also be included in the octomap, thus denying any possible robots movement because the gripper will already be in a collision state with the object. Hence, by removing the collision check between the robots links and the part of the octomap that represents the object, the manipulators will be free to move.

3.2 Selecting poses

Due to the robot's configuration and limitations, setting up multiple poses for all viewpoints becomes a tedious task with poor results. For that reason, instead, one manipulator pose is set and two specific revolute joints in the wrist are rotated, as seen in Fig. 1. By rotating in a step-wise manner both wrist joints, the robot is able to take 360° depth images of the objects. This allows for better control of the difference between poses, smoother transitions and, most importantly, leaves the object in a quasi-static state, where the center of the object barely moves, thus removing the requirement of a pre-alignment step prior to the NICP procedure. For every rotational step, the filtering and registration procedures take place. Some end effector poses are shown in Fig. 2.

Fig. 1: The red arrows represent the rotational wrist joints.

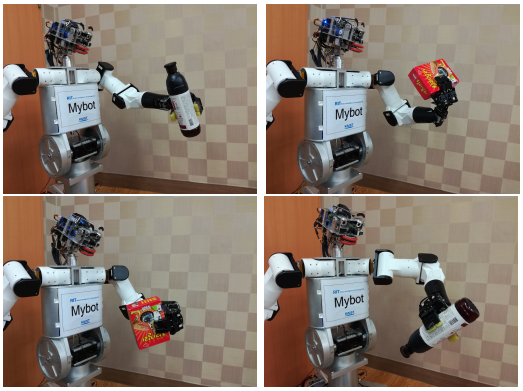

Fig. 2: Different manipulator poses to see the objects from all possible viewpoints.

3.3 Filtering of sensor data and robot links

Depth images are obtained from a RGB-D sensor and transformed into 3D point clouds. These represent the environment that surrounds the object. Yet, some filtering and segmentation must be done to be able to extract only the important information (the objects shape) from the large quantity of points the sensor outputs.

The first step is to limit the range in which useful information is found: considering that the maximum reach of the manipulators (when they are fully stretched out) is of around 1 meter, it makes no sense to keep point cloud data which is further away because the object is sure to not be outside that region. The depth sensor does not output points closer to 5 cm, thus, if a point is in that range, it is surely an error and must be discarded too. Removing so many points reduces drastically the processing time and improves performance in future steps.

Then, those points that represent some parts of the robot will be filtered out too. This means that two point clouds will result from this process. The former will have all points belonging to the robots links removed, thus, only the segmented object data will be visible. The latter, will have most of the robot's link's points also removed, except those that represent the gripper itself. As mentioned in Section 1, modeling symmetrical objects usually ends in bad results during the registration process due to poor convergence when applying the NICP algorithm. For this reason, the entire pipeline proposed in this paper will use point clouds that include the gripper since they add information that removes the symmetry problems when registering clouds.
All small noise and artifacts that are still present in the point clouds are minimized using a radius sparse outlier filter [14]. Those points that do not have a certain minimum number of neighbors inside a radial threshold will be eliminated from the cloud.

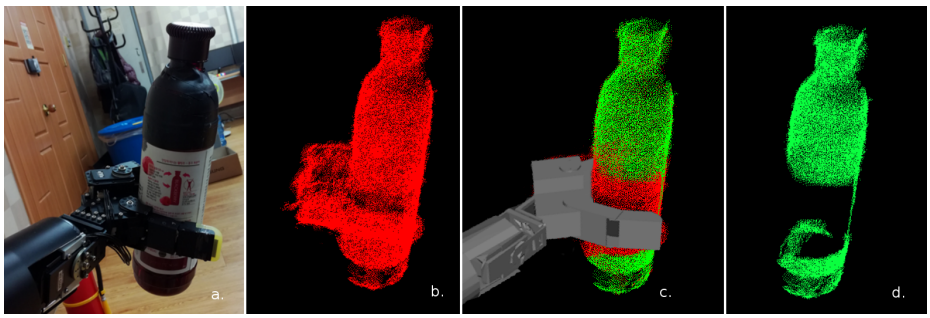

Fig. 3: From left to right. **a.** RGB image of robot grasping bottle **b.** Full model after registering point clouds from different viewpoints using only one hand. Both object and robot links are included as red points **c.** Simulated gripper on top of point cloud. The green points correspond only to the object's geometry **d.** Final partial model of the object.

3.4 Transforming points to the gripper frame

To be able to register point clouds, it is necessary that they have the same frame and have been slightly pre-aligned before the NICP algorithm is applied. Doing this increases greatly the chances of convergence.
The problem with multi viewpoint registration is that even if the base frame of the data is the same (the camera frame), due to the movement of the end-effector, the point clouds will never be the pre-aligned. To achieve this, an initial alignment based on local feature descriptors (e.g. FPFH) could be applied, but this process might produce bad results in itself, specially when dealing, once again, with symmetrical objects, where key feature density is low.

A simpler solution is to transform all received point clouds to the grippers frame. The advantages of this approach are that the objects points will be invariant to the hands movement (because once the object is grasped, it does not move in relation to the hand grasping it) and so no pre-alignment will be necessary. In fact, in the optimal scenario, this solution will allow to build the 3D model without the need for registration. Nonetheless, the reality is that small drift errors will accumulate over time between point clouds rendering the generated model useless (Fig. 4). Hence, the registration process becomes a high necessity to be able to correct for those small translational and rotational errors.

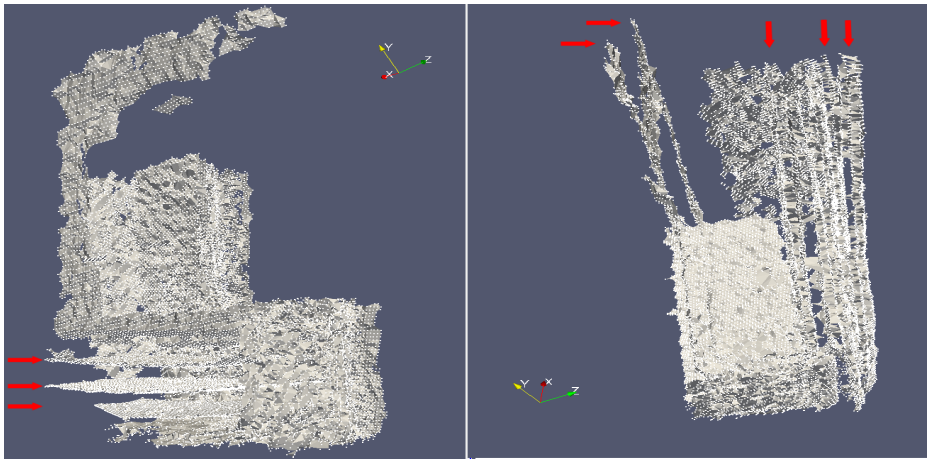

Fig. 4: Results of building a model without ICP registration: the accumulated drift renders the model unusable. The red arrows show those planes that have translational drift errors. A correct model achieved using registration based on the same data is shown in Fig. 5 and 6.

3.5 Normal Iterative Closest point (NICP) registration

The backbone of the proposed pipeline is the constant registration of point clouds over time from different viewpoints. These point clouds are the result of the objects segmentation, have the same reference frame (the gripper) and are roughly pre-aligned. To fuse these point clouds in the correct manner, the Normal Iterative Closest Point (NICP) algorithm will be used. NICP minimizes an augmented error metric (based on point coordinates and surface normals and curvature) during the least squares formulation of the alignment problem to find the best data association (transform) between two sets of point clouds. After that, a smoothing process (MLS) is performed to obtain better results.
Generally, the second point cloud will include most of the information from the first, but with additional details due to modifying slightly the perspective. It

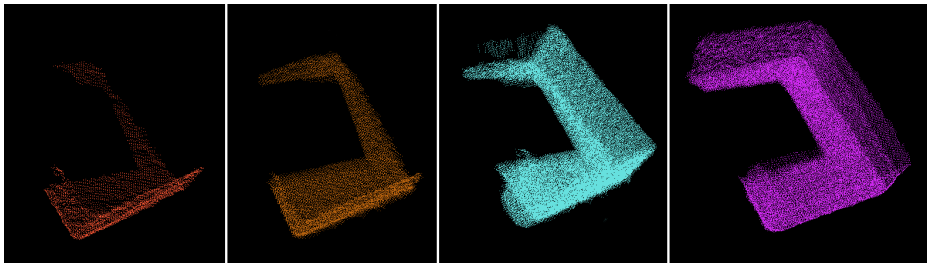

Fig. 5: NICP registration procedure of a box over several end effector poses.

is important that the changes from one point cloud to the next are very small so that the NICP algorithm can converge. If the differences are too great (for instance, by having rotated the hand a full 180° so that the opposite side of the object is being viewed), the NICP will fail to converge or give poor transforms which will accumulate over time, leading to suboptimal results. This is the main reason for using small rotational steps in the joints when changing the end effector's poses.

For even better results, a surface smoothing algorithm is applied. By estimating the normals of the point cloud's surface, and using the Moving Least Squares (MLS) algorithm as proposed in [11], [12] and [14], the resulting normals will be aligned producing a more precise model of the object, with less noise, occlusions and "double walls" artifacts. The general leaf size during the smoothing process is set to 2 cm, however depending on the geometry of the object, this value must be changed. For objects with sharper sides (e.g. boxes) this values should be lowered (1 cm) so as not to round off the edges too much. For initially already curved surfaces, like bottles or balls, this value can be slightly increased.

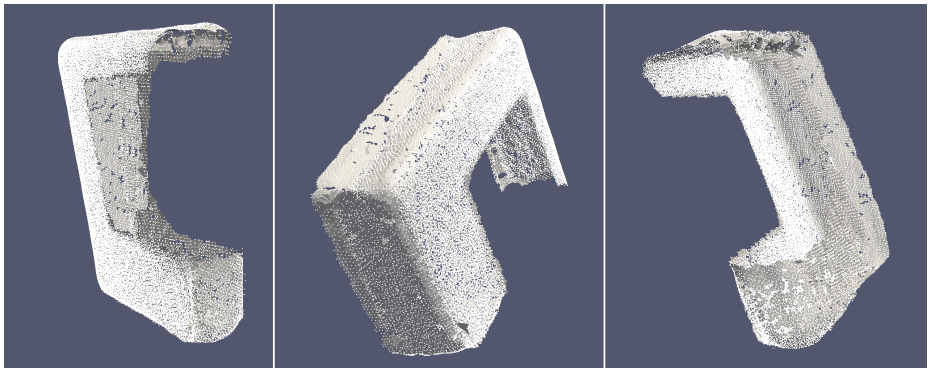

Fig. 6: Three views of the partial model generated by constructing a mesh from the resulting point cloud in Fig. 5

3.6 Changing hands

Occlusions are a big problem when modeling objects. These are due to a limited movement of the end effector (which does not allow the object to be seen all around) or links of the robot always blocking the view of part of the object (specifically the fingers and palm of the gripper).
A way to solve this is by grabbing the object from a different position and starting the registration process all over again. The results from the last registration will be merged with those of the previous registration to achieve a complete model. The fact that the system must not depend on external help, makes the grasp change difficult. If, for instance, a table could be used, the robot would have to simply place the object on it and re-grab it from a different position [12]. In spite of this, a second manipulator can be used: it is important that the second grasp allows for additional viewpoints and for the registration of the object's opposite side. For that, the second grasp must be rotated 180° with respect to the initial one. To do so, and due to the movement limitations of both manipulators, during the exchange one hand will be facing the opposite direction as the other, as seen in Fig. 7.b. In the current approach, the poses of the end effectors have been predefined. For an additional degree of autonomy, new grasping poses could be found based on the geometry of the partial model already created using the *agile_grasp* package [6], as seen in Llopart *et. al.* [13].
Finally, for the two sides of the object to be merged together, it is necessary that they are related to the same frame. For this reason, when changing hands, the registered point cloud resultant from the movements of the first gripper must be transformed to the frame of the second gripper.

***Mybot* characteristics:** The proposed pipeline has been tested out on the MyBot humanoid robot, developed in the Robot Intelligence Technology (RIT) Laboratory at KAIST (Fig. 2 and 7). It includes an Odroid XU board, Ubuntu 16.04 and ROS Kinetic. A Xtion ASUS RGB-D sensor is mounted on the 2 DoF head of the robot. It also has two 7 DoF arms with 3 finger grippers each. Finally, the torso of the robot includes another 2 DoF for panning and tilting.

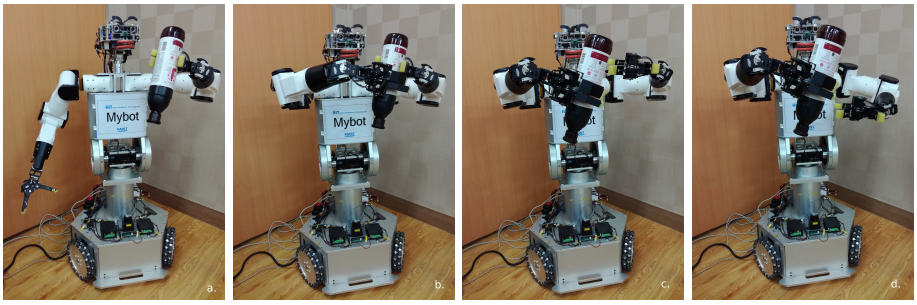

Fig. 7: *Mybot* humanoid robot changing object (bottle) from left to right hand.

3.7 Merging both partial models

Having two, almost complete, models of the object (one for each hand used) is not enough. For the model to be useful in future real life scenarios, it is necessary that it represents the object correctly from every angle. For this reason, both point clouds must be merged.
As aforementioned, both point clouds are related to the same frame. Generally, these point clouds will have an offset due to precision errors during the changing hands process. Hence, it is necessary to pre-align them. The Fast Point Feature Histogram (FPFH) descriptors can be calculated for both clouds. In this case, the FPFH based on *OpenMP* is used since its multi-threaded implementation produces results at a faster rate (6-8 times) than the normal FPFH descriptors estimation. When the features from both clouds are matched, a transformation between both models is found that allows them to be roughly aligned.
Then, by using the NICP algorithm, and setting the parameters so that the final convergence is achieved by small rotations and translations, a full and complete model of the object is achieved (Fig. 9.c). Once again, the Moving Least Squares algorithm is applied to reduce noise and errors during the merging process.

3.8 Building the mesh of model

For better visualization purposes, a greedy surface triangulation algorithm is run on the resulting point cloud with normals which results in a triangle mesh of the object. The maximum search radius and nearest neighbors values are set to 0.025 and 500, respectively, but may be altered depending on the desired number of resulting triangles. The file will be stored with a *.vtk* format [15]. Some results are seen in Fig. 6, 8 and 9.

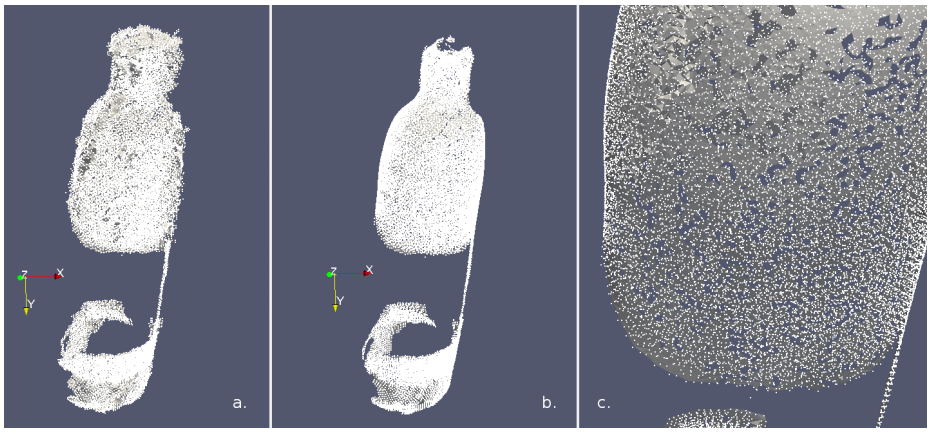

Fig. 8: Mesh result from Fig. 3, from left to right. **a.** Without prior smoothing **b.** With prior smoothing **c.** Detail of mesh.

4 Experimental results

The experiments were carried out on the *MyBot* humanoid robot. Four different unknown objects (two boxes and two bottles) were placed in one of the grippers and the process was started. The rest of the pipeline was completely autonomous. Some partial models of objects are seen in Figures 8, 6, 9.a and 9.b. Full models are seen in Fig. 9.c.

As seen in these figures, the proposed pipeline correctly outputs 3D models of unknown objects. Their overall shapes and sizes are very similar to the real ones, which is a great result considering the geometrical symmetries in all test cases. A way of assessing the overall success of the results is difficult to find. In spite of this, the measurement discrepancies between real object and generated model will be evaluated (Table 1).

It is seen that the size difference between real objects and models generally stays below the 6 mm mark: likewise, the percentage error never surpasses 9%. Concerning the overall shape of the models, the meshed result clearly matches the real surface of the objects, except for the last model of Fig. 9 where the chosen value for the smoothing process was set a bit to high, thus flattening parts of the geometry. Despite obtaining positive results, in some cases occlusions do still occur, hence the lack of details.

A way to achieve better results is by correctly differentiating between points that represent the object or the robot. When filtering the robots surface from the initial point cloud, an overall minimum distance threshold was selected. By fine-tuning the distance between robot and points, more precise results would be achieved. For instance, by reducing the threshold for the finger links, less object points would be filtered out, and, consequently, more details would appear in the final model (the red points in Fig. 3.c. would be green).

The major issues when building full 3D models occur when merging the partial results. As aforementioned (Section 3.7), the pre-alignment is done using FPFH descriptors followed by an NICP algorithm. However, if the partial models do not represent almost the entirety of the object, the alignment process will fail, rendering the final model unusable. Additionally, even if the merging process finishes correctly, usually, some parts of the model will still have not been modeled. These normally describe those surfaces of the object that were in contact with the grippers during the registration process, thus, being occluded. Another cause is the mechanical limitations of the manipulators to see the object from certain viewpoints. A good example of occlusion errors (holes in the model) in the final result are the second and fourth models in Fig. 9.c.

Object	Red box			Blue box			Black bottle		Blue bottle	
Dimension	w	h	d	w	h	d	ø	h	ø	h
Object (cm)	12.8	17.5	5.6	8.1	16.9	6.9	7.0	28.7	5.7	18.4
Model (cm)	12.7	17.9	6.1	8.5	17.3	6.6	7.6	28.8	6.1	18.2
Error (cm)	0.1	0.4	0.5	0.4	0.4	0.3	0.6	0.1	0.4	0.2
Error (%)	0.78	2.29	8.93	4.94	2.37	4.35	8.57	0.35	7.02	1.09

Table 1: Geometric measurement comparison between real object and model.

Fig. 9: From left to right. **a.** Partial model created using left gripper **b.** Partial model created using right gripper **c.** Full model after merge **d.** Real RGB image of model (for viewing purposes only).

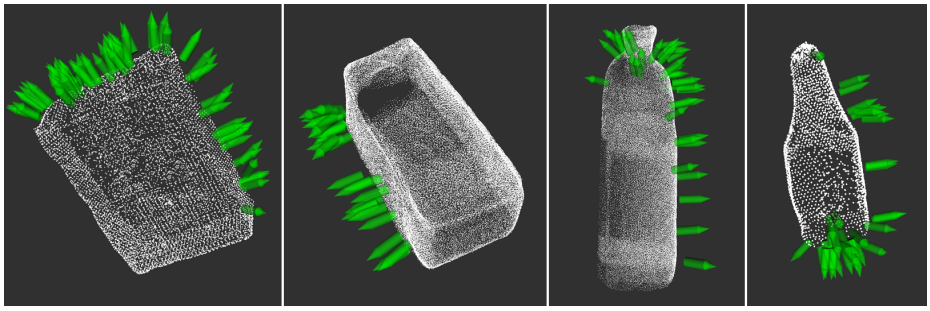

Fig. 10: Grasping poses for the generated 3D models based off solely the geometry of the point cloud and the grippers dimensions, as proposed by Pas *et. al.* [6]

5 Conclusions

Full 3D models of unknown objects are generated through the proposed pipeline. Registering the point clouds of different views whilst holding the object with one manipulator, enables the robot to create a partial (due to occlusions) model of it. By switching the object from one hand to another and repeating the process, a new partial model is obtained which will not have the occlusion errors of the first one (and vice versa). Merging both partial models accomplishes the removal of errors and the generation of a full 3D model of the object that can be later used for detection, recognition or manipulation purposes. The results show that correct models are being generated for diverse objects in spite of the geometrical symmetries. Fig. 10 shows possible grasping locations based off the geometry of the 3D model, which can be stored and applied whenever the object has to be manipulated.

6 Future work

For better results, a loop closure detection system, as proposed in [11], [12] and [16], could be implemented. Additionally, the poses from which the object is seen have been pre-selected by the user. These do not take into account occlusions nor geometry of the object, sometimes leading to redundant point clouds that add no information or, on the contrary, lack of viewpoints to generate a full model. Consequently, it would be interesting to add the *Next best view* concept presented in [12] and [17]. The ultimate goal of the proposed approach is to be combined with the pipeline introduced by Llopart *et. al* [13] to autonomously detect, segment and manipulate unknown objects using a humanoid robot.

References

1. A. Aldoma, F. Tombari, L. D. Stefano, and M. Vincze, "A Global Hypotheses Verification Method for 3D Object Recognition," 2012, european Conference on Computer Vision (ECCV) . Lecture Notes in Computer Science, vol 7574. Springer.

2. M. Zhu, K. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single Image 3D Object Detection and Pose Estimation for Grasping," in *Proc. International Conference on Robotics and Automation(ICRA)*, Hong Kong, China, May 31 - June 5 2014.
3. K. Sarkar, K. Varanasi1, and D. Stricker, "Trained 3D models for CNN based object recognition," in *Proc. International Conference on Computer Vision (ICCV)*, Santiago, Chile, December 13-16 2015.
4. X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning Deep Object Detectors from 3D Models," in *Proc. International Conference on Computer Vision (ICCV)*, Santiago, Chile, December 13-16 2015.
5. S. Gupta, P. Arbelaez, R. Girshick, and J. Malik, "Aligning 3D Models to RGB-D Images of Cluttered Scenes," in *Proc. International Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 7-12 2015.
6. A. Pas and R. Platt, "Using Geometry to Detect Grasp Poses in 3D Point Clouds," in *Proc. International Symposium on Robotics Research (ISRR)*, Genova, Italy, September 2015.
7. M. Denkowski, "GPU Accelerated 3D Object Reconstruction," 2013, procedia Computer Science 18, 290298. Web.
8. R.-G. Mihalyi, K. Pathak, N. Vaskevicius, T. Fromm, and A. Birk, "Robust 3d object modeling with a low-cost rgbd-sensor and ar-markers for applications with untrained end-users," in *Robotics and Autonomous Systems*, 2015, ch. 66, pp. 1–17.
9. J. Xie, Y.-F. Hsu, R. Feris, and M.-T. Sun, "Fine registration of 3d point clouds fusing structural and photometric information using an rgb-d camera," in *Journal of Visual Communication and Image Representation*, 2015, ch. 32, pp. 194–204.
10. T.Foissotte, O. Stasse, A. Escande, P.-B. Wieber, and A. Kheddar, "A Two-Steps Next-Best-View Algorithm for Autonomous 3D Object Modeling by a Humanoid Robot," in *Proc. nternational Conference on Robotics and Automation(ICRA)*, Kobe, Japan, May 12-17 2009.
11. M. Jaiswal, J. Xie, and M.-T. Sun, "3D Object Modeling with a Kinect Camera," in *Proc. Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Chiang Mai, Thailand, June 6-9 2014.
12. M. Krainin, P. Henry, and X. Ren, "Manipulator and object tracking for in-hand 3d object modeling," in *International Journal of Robotics Research*, September.
13. A. Llopart, O. Ravn, N. Andersen, and J.-H. Kim, "Generalized Framework for the Parallel Semantic Segmentation of Multiple Objects and Posterior Manipulation," in *Proc. International Conference on Robotics and Biomimetics (ROBIO)*, Macau, China, December 5-8 2017.
14. R. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," in *Robotics and Autonomous Systems*, 2008, ch. 56.11, p. 927941.
15. C. Marton, R. Radu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *Proc. International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.
16. T. Weise, T. Wismer, B. Leibe, and L. V. Gool, "In-hand Scanning with Online Loop Closure," in *Proc. International Conference on Computer Vision Workshops (ICCV Workshops)*, Kyoto, Japan, September 27 - October 4 2009.
17. M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3D object models using next best view manipulation planning," in *Proc. International Conference on Robotics and Automation(ICRA)*, Shanghai, China, May 9-13 2011.