Multi-Contour Initial Pose Estimation for 3D Registration

Ernest C. H. Cheung and Chao Cao and Jia Pan

Abstract— **Reliable manipulation of everyday household objects is essential to the success of service robots. In order to accurately manipulate these objects, robots need to know objects' full 6-DOF pose, which is challenging due to sensor noise, clutters and occlusions. In this paper, we present a new approach for effectively guessing the object pose given an observation of just a small patch of the object, by leveraging the fact that many household objects can only keep stable on a planar surface under a small set of poses. In particular, for each stable pose of an object, we slice the object with horizontal planes and extract multiple cross-section contours. The pose estimation is then reduced to find a stable pose whose contour matches best with that of the sensor data, and this can be solved efficiently by convolution. Experiments on the manipulation tasks in the DARPA Robotics Challenge validate our approach. In addition, we also investigate our method's performance on object recognition tasks raising in the challenge.**

I. Introduction

A robot will need to accurately determine the 6-DOF pose of an object for achieving reliable grasp and manipulation of everyday household objects. For instance, in the scenario of the DARPA Robotics Challenge (DRC) [1], a humanoid robot needs to manipulate a set of objects designed for human beings, including valves, steering wheels, door handles, power tools, cordless drills, cutting tools, and fire hoses. Figure 1 shows some of these tools. Before being able to compute a stable grasp for these objects using techniques such as friction cones [2], or form- and force- closures [3], the robot must first provide an accurate estimation of the object pose. Given a 3D object point cloud and a point cloud observation of a scene containing the object, the pose estimation is usually formulated as an optimization problem, that is, by solving for the best rotation and translation between the two point clouds such that the distance between the overlapping areas of these point clouds is minimal, given an appropriate metric space. This procedure is also known as the 3D registration, which consistently aligns two overlapping point clouds. Without any prior pose knowledge, the pose estimation problem is difficult and most optimization techniques may fail to find the optimal solution. This is mainly because the function to be optimized is multi-dimensional and has multiple local optimum solutions possibly close to the global one. The noise in the sensor data (e.g., due to LIDAR's range uncertainty) and the occurrence of partial observation due to occlusion also add up to the difficulties of accurate pose estimation.

A lot of work have been done in 3D registration and the most popular registration method to date is the Iterative Closest Point (ICP) algorithm [4]. The ICP method has improved a lot from its original form, including using non-linear optimization methods, designing better point alignment features, and finding good initial guesses for the registration. Our contributions fall within the area of finding good initial guesses for pose estimation, which brings the search process into the convergence basin of non-linear optimization problems. In particular, we leverage the fact that in most manipulation scenarios, the target object is posed in a stable configuration sitting on flat support surfaces, e.g., tables, shelves and floors. This fact implies several simplifications to the 3D registration that could greatly improve the efficiency and accuracy of pose estimation. First, the points corresponding to the target objects can be extracted easily by recognizing the planar support plane and establishing a lower bound for the points of interest. Second and even more important, most man-made objects have only a small number of stable *pose classes*, where a 'class' of pose means all the poses that have the same support while sitting on a horizontal plane. For instance, the drill shown in Figure 2 has four stable pose classes. Within each pose class, the registration can be reduced to a 3-DOF problem, i.e., the pose is restricted to the x-y translation along the support plane and the yaw rotation about the support normal.

Fig. 1. Examples of man-made tools used in the DARPA Robotics Challenge (from left to right): a barrel-type drill, a fire hose standpipe, a cutting tool and a gun-type drill.

Fig. 2. All four stable pose classes for a gun-type drill.

In this paper, we present a novel approach for efficient and accurate 3D registration by using multiple contours extracted from objects. For each pose class of an object, we slice the object with a set of planes parallel with the support plane associated with this pose class, and extract a set of K slice-

Authors are with the Department of Computer Science, the University of Hong Kong, Hong Kong.

plane boundary contours. Given a partial observation of the object during the robot's execution, we first detect its support plane and then also extract K contours by slicing the point cloud using the support plane. By computing the similarity between contours with convolution, we can find the pose class that best explains the object's partial view, and further determine a good initial guess within this pose class. This initial guess is then used by the ICP algorithm to obtain a high quality pose estimation.

We investigate the performance of our method using a rotating Hokuyo sensor mounted on a Carnegie Robotics Sensor [5], which is used by the Atlas robot (Figure 3) while executing the DRC tasks. From the experiment, we show that our method can significantly improve the efficiency and quality of the pose estimation, while comparing with the state-of-the-art 3D registration approaches. In addition, it is able to provide better performance for man-made objects with symmetries, and is more robust to data uncertainty.

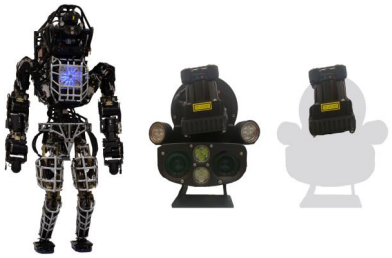

Fig. 3. The Hokuyo LIDAR sensor (right) mounted on the Carnegie Robotics Sensor (middle) used in ATLAS (left).

II. Related Work

Most techniques for pose estimation of a rigid 3D object are correspondence-based, relying on the presence of features on the object's surface so that a rigid body transform can be computed which minimizes the distance between corresponding points. According to the underlying optimization method used, these approaches can be categorized into two types: global and local. The global approaches use various global stochastic optimization techniques, including genetic algorithms [6] and various evolutionary techniques such as scatter search [7] or CMA-ES [8]. These methods are slow in computational time and lack a rigorous guarantee of convergence. Other work fall into the second category of local search, and the most popular method is the Iterative Closest Point (ICP) algorithm. As a non-linear local search algorithm, ICP suffers from many problems commonly associated with local search, including slow convergence and the tendency to fall into local optima. Thus, many improvements from its original form have been proposed, including using non-linear optimization methods [9], [10], finding good initial guesses [11], [12], and estimating better point correspondence by leveraging more advanced point features [11], [13], [14], [15], and more efficient ICP variants [16], [17]. However, many common household objects – such as tools used in DRC – do not have enough number of unique identifiable visual features to lock down a pose.

There are some recent alternatives to the correspondence-based pose estimation. Classification-based approaches such as [18] classify object observations by viewpoints, and work well when the object is perfectly segmented out of the scene cloud, but these methods are sensitive to clutter and occlusions. The generalized Hough transform [19] uses local geometric information to vote for the object pose, and is robust to occlusions, and it can be combined with Random-Sampling-Consensus (RANSAC) alignment framework for continuous search within the pose space [20].

III. Data and System

In 3D registration, we match up the scene data with a set of model data for different objects, where both the scene and model data are in form of point clouds.

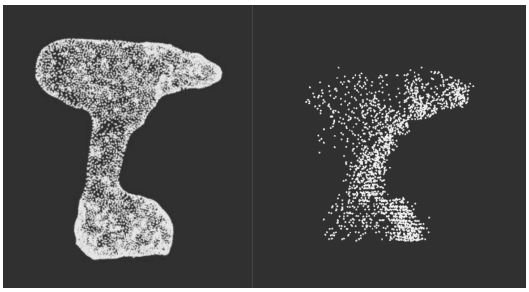

Fig. 4. Examples of model data (left) and scene data (right) used in our experiments.

The point cloud model data for each object is built using a Kinect sensor [21] and provides a complete geometry about the object. For the scene data collected during the online execution of the robot, we do not use Kinect sensor since its range is too limited and cannot operate properly outdoors. Instead, we use a rotating LIDAR mounted on the robot's head for online scene data collection, as shown in Figure 3. The rotating LIDAR provides data in polar coordinates (the range and the sweep angle), but the reference frame of the LIDAR is rotating. After accounting for this transform, 3D points can be obtained over a hemisphere. The range data is acquired at the rate of 40,000 samples per second, per 0.25 degrees within a slice plane. The resolution of the scene data depends on the LIDAR rotation rate, the duration of data collection, and the distance between the object and the sensor. Given that the accuracy of the LIDAR sensor is only up to ± 30 mm for one frame, points over different frames are accumulated over time to reduce the data uncertainty. In addition, the data obtained by the LIDAR is only a partial view of the entire scene, due to the LIDAR's limited range and occlusions. An example of the model data and scene data for a drill object is shown in Figure 4.

Given the scene data from the LIDAR, we identify the region of interest in the point clouds by first locating the plane frame and then filtering out the points above the plane. Next, we apply the Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [22] to separate the point clouds into individual objects.

Figure 1 illustrates all the target objects that will be used in our experiment. These tools are likely to be used in the DARPA Robotics Challenge, including a barrel-type drill, a gun-type drill, a cutting tool and a fire hose standpipe.

IV. Contour Initial Pose Estimation

For fast 3D registration, we consider the special case where the target object is stably posed on a flat surface, e.g., tables, shelves, floors, etc. This is a common scenario in most manipulation tasks, for instance, in DARPA Robotics Challenge, the robot needs to use a tool placed on a table or inside a shelf, or to retrieve a tool box placed on the floor, or to turn a valve fixed on a wall. For these cases, given the fact that the extraction of the flat surface can be implemented efficiently, we can leverage such flat surface to reduce the search space of pose estimation from a 6-dimensional space to a set of 3-dimensional spaces. In particular, we first compute a pose within each pose class that best matches the partial view of the object, by only considering the rotation angle about the normal of the support plane and two translations along the support plane. After finding the best poses for all the pose classes, we take the one with the highest match score as the result for the pose estimation.

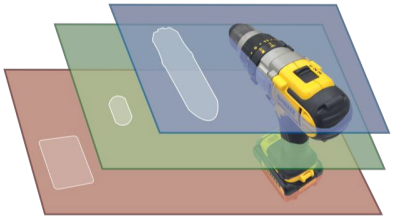

Fig. 5. Contours for three different cross-sections of a gun-type drill.

In this section, we provide a brief review of our previous work on Contour Initial Pose Estimation (CIPE) [23], that is, while knowing the object's pose class, how to find a pose within this class that can best match the sensor's observation, by exploiting the object's contour features. We leave the problem about how to determine the optimal pose class to Section VI.

Given an object model and its stable pose classes, we preprocess the model to extract a set of contour features. In particular, we slice the model with a set of planes parallel with the contact face associated with a pose class, and compute slice-plane boundary contours. We call these contours the *model contours*. Figure 5 shows the three model contours extracted for a gun-type drill. The same operation can be performed for the sensor data, by first extracting the support flat surface from the point cloud and then slicing the point cloud using the extracted plane. The obtained contours are called the *scene contours*.

Given the extracted contours, the 3D registration problem is reduced to determine the rigid body transform between the model contours and the scene contours. We first compute the curvature along these contours, and then perform convolution between them with respect to the arc length, as shown in Figure 6. The convolution result describes the similarity between the model contour and the scene contour at different arc length offsets, and thus we call it the *contour similarity*. For instance, when the arc length offset is zero, the two contours are aligned as shown in Figure 6. Due to the definition of convolution, the value of contour similarity is equivalent to the overlapping area (shown by the purple region in Figure 6 of two curvature series. As the offset increases, the overlapping area under the curves of the scene and model curvatures increases and thus the convolution results arrives at a local peak.

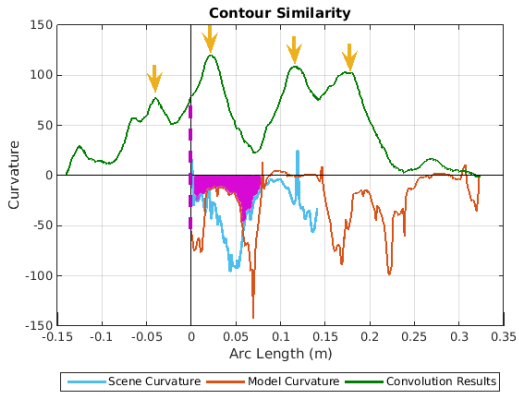

Fig. 6. Curvatures of the model contour (orange) and the scene contour (blue) and their convolution results (green). Since the point with maximum arc length (about 0.325 meter) and the point with minimum arc length (0 mm) are next to each other in the model contour, the model curvature has been repeated for the purpose of obtaining convolution results with negative offsets.

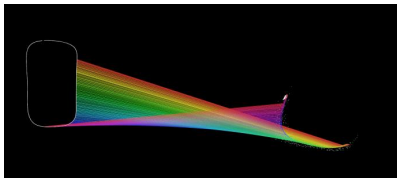

Fig. 7. Correspondence between the model contour (left) and the scene contour (right).

The convolution result will contain several local peaks (as the yellow arrows marked in Figure 6), which correspond to the potential matches between the model contour and the scene contour. Given the arc length offsets related with these peaks, we can establish the point correspondence between the model contour and the scene contour, as shown in Figure 7.

Once the correspondence between the contours is established, we can compute the transform between these two contours by solving a linear system:

$$\begin{pmatrix} x_1^m & y_1^m & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1^m & y_1^m & 1 \\ x_2^m & y_2^m & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2^m & y_2^m & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} r_{00} \\ r_{01} \\ r_{02} \\ r_{10} \\ r_{11} \\ r_{12} \end{pmatrix} = \begin{pmatrix} x_1^s \\ y_1^s \\ x_2^s \\ y_2^s \\ \vdots \end{pmatrix}, \tag{1}$$

where $\{(x_1^m, y_1^m), (x_2^m, y_2^m), ...\}$ are the 2D points in the model contour and $\{(x_1^s, y_1^s), (x_2^s, y_2^s), ...\}$ are the 2D points in the scene contour, and $\{r_{ij}\}$ describes the 2D affine transform between these two contours

$$\begin{pmatrix} x^s \\ y^s \end{pmatrix} = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \end{pmatrix} \begin{pmatrix} x^m \\ y^m \\ 1 \end{pmatrix}. \quad (2)$$

We then use QR transform to project the affine transform into the space of rigid body transforms and the result is a matrix $\mathbf{T}_{\text{init}} = \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{pmatrix}$. $\mathbf{T}_{\text{init}}$ is then passed to the ICP algorithm as the initial guess for refining the relative pose between the point clouds of the object model and the scene. We repeat the above process for each of the convolution's local optimum, and choose the pose with the lowest registration error as the result of the pose estimation.

V. Multi-contour Initial Pose Estimation

The performance of CIPE is promising as shown by our previous experiments in [23], and it can provide higher pose estimation accuracy than the state-of-the-art pose estimation approaches such as [15]. However, CIPE may not be able to provide enough accuracy while handling objects that are highly symmetric. For instance, the gun-type drill as shown in Figure 8 has two symmetric planes: one is the plane along the direction of the drill borer (as shown by the red plane in Figure 8) and the other is the plane perpendicular to it (as shown by the green plane in Figure 8). Due to the multiple symmetric planes, an incorrect registration may produce smaller error than a correct registration according to a partial observation of the object, as shown in Figure 8. This problem can be even more severe when the scene data is corrupted by noises.

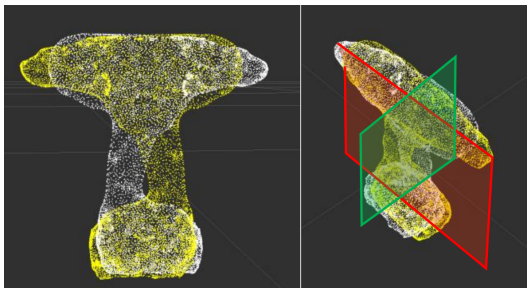

Fig. 8. An incorrect registration (in white) for a highly symmetric object from two different perspectives. The green and red planes are two symmetric planes of the object.

In order to overcome this difficulty, we develop a new approach which computes the contour similarity and performs convolution based on more than one layers of contours, and thus we call this method the Multi-Contour Initial Pose Estimation (MCIPE). MCIPE is not a trivial extension of CIPE, because we cannot simply add up the contour similarities of different layers of cross-sections since these contours are of different scales in length. Instead, we use the angle of each contour point relative to the contour centroid to establish the correspondence between different layers of cross-sections. In this way, MCIPE can compute a contour similarity which takes into account the information from multiple cross-sections of the object.

The detail of the MCIPE algorithm is as follows. Given the 3D point clouds M and S, for the model and scene respectively, we consider the 2D contours generated by slicing the point clouds with K parallel planes:

$$M_i = \{(x, y, z) : (x, y, z) \in M \wedge |z - h_i| \leq \epsilon\}$$

and

$$S_i = \{(x, y, z) : (x, y, z) \in S \wedge |z - h_i| \leq \epsilon\},$$

where $i = 1$ to K, and K is the number of contours to be considered. h_i is the height of the cross-sections pre-selected and ϵ is the threshold while determining whether a point belongs to a cross-section or not. The MCIPE algorithm includes three steps:

1) For each pair of model contour M_i and scene contour S_i, we perform the following computations:
   a) For each point (x, y) on a contour, we compute its arc length relative to an arbitrary but fixed reference point. This results in a mapping from any 2D point on the contour to the arc length. We compute such a mapping for both M_i and S_i, and the resulting arc length mappings are denoted as $P_{M_i}(x, y)$ and $P_{S_i}(x, y)$ respectively.
   b) We compute $M_i(s)$ and $S_i(s)$ as the inverse mapping of $P_{M_i}(x, y)$ and $P_{S_i}(x, y)$, where s is the arc length parameterization. In other words, these two functions provide the arc length parameterization for these two contours.
   c) We compute the signed curvature at each contour point and generate the curvature functions $\kappa_{M_i}(x, y)$ and $\kappa_{S_i}(x, y)$ for two contours. Based on the arc length parameterization, we can further obtain the curvature functions parameterized by the arc length: $\kappa_{M_i}(s)$ and $\kappa_{S_i}(s)$.
   d) For each point (x, y) on the model contour M_i, we use polar coordinate transform to compute the angle of the contour point relative to the model centroid, and the resulting angle function is denoted as $\alpha_i(x, y)$. Based on the arc length parameterization, we can further compute the angle function parameterized by the arc length: $\alpha_i(s)$.
   e) We then perform convolution between $\kappa_{M_i}(s)$ and $\kappa_{S_i}(s)$ to obtain the contour similarity function $c_i(s)$ of this cross section.
2) We select an arbitrary but fixed layer I ($1 \leq I \leq K$) as the reference layer. For each other layer j, we first align it with the I-th layer according to the angle function. That is, we reparameterize the j-th layer curvature functions as $s'_j = \alpha_j^{-1} \circ \alpha_I(s)$. After the reparameterization, we can add the curvature functions from all layers together as $c(s) = c_I(s) + \sum_{j \neq I} c_j(s'_j)$,

which is a contour similarity taking into account the information from all K layers.

3) We find all arc length offsets s_k such that $c(s)$ arrives at local optimum at s_k. For each s_k, we repeat the following computation:
   a) Use s_k to establish the correspondence between $M_i(s)$ and $S_i(s)$, by matching $M_i(s+s_k)$ with $S_i(s)$.
   b) Collect the registered points from all the layers and denote the collections as $M' = \bigcup_i M_i(s+s_k)$ and $S' = \bigcup_i S_i(s)$.
   c) Compute $\mathbf{T}_{\text{init}}$ by applying CIPE on M' and S'.
   d) Perform ICP on M' and S' using $\mathbf{T}_{\text{init}}$ as the initial guess and obtain $\mathbf{T}_k$.
4) After obtaining all $\mathbf{T}_k$, we choose $\mathbf{T}_{\text{MCIPE}}$ as the one with the minimal registration error. Finally, we perform ICP on M and S using $\mathbf{T}_{\text{MCIPE}}$ as the initial guess and generate the result for pose estimation.

Figure 9 shows how the contour similarities of two cross sections of a yellow drill are summed up to generate the multi-contour similarity as described in step 2).

The advantage of MCIPE is not merely about its ability of overcoming the limitation of CIPE on symmetric objects. In the context of obtaining $\mathbf{T}_k$ and estimating the error (i.e., step 3 and 4), instead of using all the points in the model and scene data M and S, MCIPE only considers the data from the sampled layer contours, i.e., M' and S'. In contrast, CIPE has to perform ICP for the entire point clouds because a single layer of contour points may not be enough for determining the pose for symmetric contours. As a result, MCIPE is more efficient than CIPE.

On the other hand, step 1 has introduced extra computational burden in MCIPE beside that of CIPE, because MCIPE needs to consider more than one layer of contours. However, this computation is highly parallelizable, and hence we use multi-core processors or multiple machines to accelerate the computation.

VI. Pose Class Determination

Both CIPE and MCIPE assume that the target object is stably placed on a plane. Most objects have a small number of stable pose classes. For instance, Figure 2 lists all the possible stable pose classes for a drill.

In our previous work [23], we assumed that the pose class of an object would be specified by the operator before the robot uses CIPE to estimate the object's pose for manipulation purposes. Such human assistance was considered as low-cost because human being can easily identify the pose class. However, such effort may become expensive when the robot has to be operated under environments with degraded network communication: for instance in DRC, the network packages would have a high latency and may be interrupted and lost during the communication. Such kind of poor network environment may also happen during the real life, e.g., in the disaster. As a result, it is preferable that such human assistances can be replaced by autonomous operations, especially when the robot needs to manipulate more than one objects.

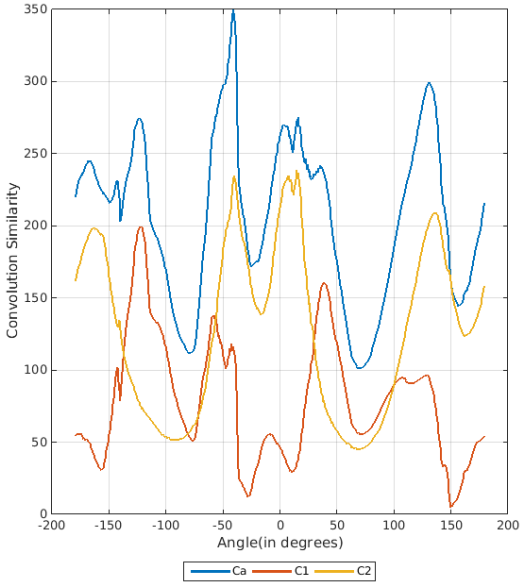

Fig. 9. The convolution similarities of two layers c_1 and c_2 are added together to compute the multi-contour convolution similarity c_a.

Given that most objects have limited number of pose classes, we repeat MCIPE for each pose class to obtain a set of candidate poses, and then determine the pose estimation result as the pose with the minimal registration error. In addition, by taking account of our assumption that the object lies on a plane, we can further reduce the number of candidate poses by using the height heuristic. For instance, let's consider the first and second pose class in Figure 2, whose overall heights are smaller than other pose classes. Thus, if the distance from a point in the scene data to the support plane exceeds a given threshold, we can skip the MCIPE computation for this pose class.

VII. Experiments

A. Correctness of MCIPE

We compare the performance of MCIPE and CIPE using the gun-type drill because it is a highly symmetric object. In addition, we also apply MCIPE on the cutting tool and the fire hose to validate its correctness. The pose estimation results are shown in Figure 10 and Figure 11. We can observe that for the MCIPE algorithm, the average error on the estimated yaw angle is 3.237 degrees, and the position error is always smaller than 10 mm. These results are smaller as compared to those of CIPE [23].

We also observe that in CIPE, 3 out of 16 tests make an estimation that is 180 degrees different from the actual angle, which shows CIPE's limitation while handling symmetric objects. In contrast, MCIPE returns with correct guesses in all the 16 tests for the gun-type drill as shown in Figure 10, and all the 32 attempts for the two other objects as shown in Figure 11. Since our previous work [23] has shown that the

accuracy of CIPE is significantly higher than the state-of-the-art methods such as [15], we can conclude that MCIPE is more effective than the state-of-the-art approaches.

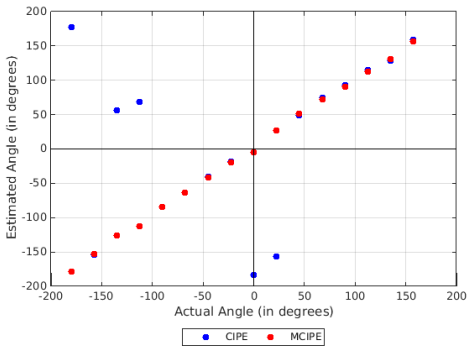

Fig. 10. Comparison between the accuracy of the rotation angles computed by MCIPE and CIPE on the gun-type drill. The x and y axes represent the estimated and the actual angle of the drill.

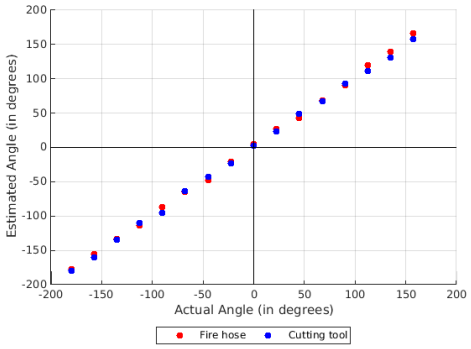

Fig. 11. The accuracy of the rotation angles computed by MCIPE on the fire hose standpipe and the cutting tool. The x and y axes represent the estimated and the actual angle of objects.

The efficiency of MCIPE is comparable to CIPE, and the total time taken for the entire algorithm going through all the layers is 200ms on average.

B. Pose Class Determination using MCIPE

For determining the pose class of an object, we perform experiments on the gun-type drill and the barrel-type drill. The results are shown in Figure 12 and 13.

For the gun-type drill, we denote class I to class IV corresponding to the poses illustrated in Figure 2, from left to right respectively. Of all the 32 test cases we have experimented (with 8 test cases for each pose class), the results show that all estimated poses will converge to their actual pose class.

Figure 13 shows the pose class determination results for the barrel-type drill, which has two almost perfect symmetric planes. We define class I as the standing pose, which is the pose as shown in Figure 1; class II and class III are the poses in which the drill lays down on the table in two different ways.

With the presence of sensor noises, the point cloud data alone may not be enough to determine the pose accurately,

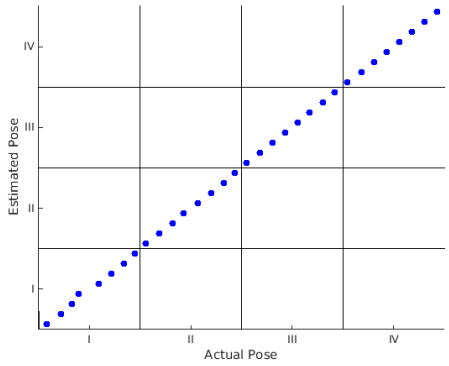

Fig. 12. Pose class determination results for the gun-type drill. The x and y axes represent the estimated and the actual angle of the drill. I, II, III and IV correspond to four pose classes.

and thus MCIPE estimated the pose incorrectly in some test cases. The result shows that MCIPE estimates the pose class wrongly for 14 out of 32 test cases in class II and III, and also for 4 out of 16 test cases in class I, as shown by the red and green dots in Figure 13 respectively. Though MCIPE

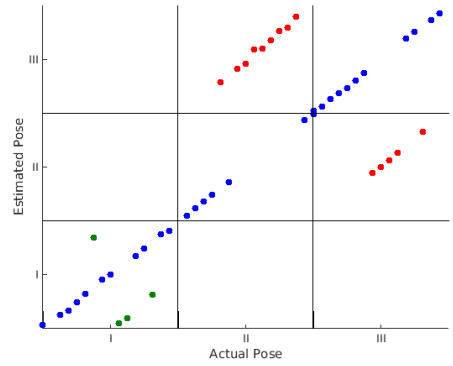

Fig. 13. Pose class determination results for the barrel-type drill. The x and y axes represent the estimated and the actual angle of the drill. I, II and III correspond to three pose classes. Blue points are poses whose pose class estimation is correct. Red points and green points are poses whose pose class estimation is not correct due to symmetry.

fails on some test cases which shows its limitation while handling objects with two perfect symmetric planes, it shall be noted that determining the pose class for these objects is not possible for any algorithm only using texture-less point cloud data. To deal with these objects, we need to combine additional information such as textures for the two candidate solutions provided by our algorithm.

It takes about 770ms and 590ms to determine the pose class for the gun-type drill and the barrel-type drill. This process can be accelerated by GPU acceleration and cloud computing, since running MCIPE on an object is independent of running it on other objects. In addition, we could further accelerate the computation by using the height matching between the model and scene data.

C. Object Classification using MCIPE

The same technique can be applied for object recognition, by considering each pose class of each object as a separate

model data and apply MCIPE for registering all of them against the scene data. For all the four objects in Figure 1, there are 14 different pose classes in total. For each actual pose, 5 randomly placed pose has been tested and the results are shown in Figure 14. Among all 70 test cases, 63 of them are correct, 4 of them are incorrect due to the symmetric problem, and 3 of them are incorrect due to the high level of sensor noise.

			Actual														
	Obj/		G-Drill				B-Drill				F.S.			C. T.		
	Pose		I	II	III	IV	I	II	III	IV	I	II	III	I	II	III
Estimated	G-Drill	I	5													
		II		5												
		III			5	2		1					1			
		IV				3										
	B-Drill	I					5			1						
		II						3								
		III						1	5				1			
		IV								4						
	F.S.	I									5					
		II										5				
		III											3			
	C.T.	I												5		
		II													5	
		III														5

Fig. 14. Object recognition across four different objects and all their pose classes. The gun-type drill (G-Drill) and the barrel-type drill (B-Drill) have four pose classes, while the fire hose standpipe (F.S.) and the cutting tool (C.T.) have three pose classes.

VIII. Conclusion

We have proposed a novel algorithm, MCIPE, to produce better initial pose guesses for ICP to solve the 3D registration problem. MCIPE is efficient and is able to provide better pose estimation results than the state-of-the-art approaches. In addition, MCIPE is able to automatically identify the pose class that the target object is currently in. It can further be extended to handle the object recognition problem.

Future extensions that we plan to investigate include: 1) improving the curve fitting technique to deal with multiple contours lying on the same level due to occlusion; 2) automatically identifying the optimal slice contour; 3) modifying the algorithm to solve the object recognition problem with massive number of objects; 4) extending our approach to cluttered scenes with stacked objects by leveraging physcially-based simulators to estimate the stability of stacked objects under a given set of pose estimations and filter out invalid pose combinations.

Acknowledgments

The authors would like to thank Professor Wyatt Newman for his insight and advice in this research.

References

[1] G. Pratt and J. Manzo, "The darpa robotics challenge [competitions]," *IEEE Robotics Automation Magazine*, vol. 20, no. 2, pp. 10–12, 2013.

[2] M. T. Mason, "Manipulator grasping and pushing operations," in *Robot hands and the mechanics of manipulation*, M. T. Mason and J. Salisbury, Eds. Cambridge, MA, USA: Massachusetts Institute of Technology, 1985, pp. 171–294.

[3] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2000, pp. 348–353.

[4] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[5] "Multisense sl," https://carnegierobotics.com/multisense-sl/.

[6] L. Silva, O. R. P. Bellon, and K. L. Boyer, "Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 762–776, 2005.

[7] O. Cordón, S. Damas, and J. Santamaría, "A fast and accurate approach for 3d image registration using the scatter search evolutionary algorithm," *Pattern Recognition Letters*, vol. 27, no. 11, pp. 1191–1200, 2006.

[8] Y. Otake, M. Armand, R. S. Armiger, M. Kutzer, E. Basafa, P. Kazanzides, and R. Taylor, "Intraoperative image-based multiview 2D/3D registration for image-guided orthopaedic surgery: Incorporation of fiducial-based c-arm tracking and GPU-acceleration," *IEEE Transactions on Medical Imaging*, vol. 31, no. 4, pp. 948–962, 2012.

[9] "Least squares 3d surface and curve matching," *ISPRS Journal of Photogrammetry and Remote Sensing*.

[10] A. W. Fitzgibbon, "Robust registration of 2d and 3d point sets," in *British Machine Vision Conference*, 2001.

[11] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust global registration," in *Eurographics Symposium on Geometry Processing*, 2005.

[12] A. Makadia, A. I. Patterson, and K. Daniilidis, "Fully automatic registration of 3d point clouds," in *International Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1297–1304.

[13] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[14] G. Sharp, S. Lee, and D. Wehe, "Icp registration using invariant features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 90–102, 2002.

[15] R. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3d registration," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.

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

[17] A. Nuchter, K. Lingemann, and J. Hertzberg, "Cached K-d tree search for ICP algorithms," in *International Conference on 3-D Digital Imaging and Modeling*, 2007, pp. 419–426.

[18] R. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2155–2162.

[19] D. H. Ballard, "Readings in computer vision: Issues, problems, principles, and paradigms," M. A. Fischler and O. Firschein, Eds., 1987, ch. Generalizing the Hough Transform to Detect Arbitrary Shapes, pp. 714–725.

[20] J. Glover, R. Rusu, and G. Bradski, "Monte carlo pose estimation with quaternion kernels and the bingham distribution," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[21] Y. Cui and D. Stricker, "3d shape scanning with a kinect," in *ACM SIGGRAPH 2011 Posters*, ser. SIGGRAPH '11, 2011, pp. 57:1–57:1.

[22] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[23] E. C. H. Cheung, C. Chao, and W. S. Newman, "Initial pose estimation using cross-section contours," in *IEEE International Conference on Robotics and Biominmetics*, 2014, p. to appear.