6D Object Pose Estimation using RGBD Data and Fast-ICP

Nitin J. Sanket[1] & Daniel D. Lee[2]

***Abstract*— This paper presents an approach for fast 6D pose estimation of the object using noisy RGBD sensor data with a variant of the Iterative Closest Point (ICP) algorithm which is optimized to work for speed while retaining if not improving accuracy. The results presented show that the algorithm is sub degree and centimeter accurate in localizing the object.**

I. INTRODUCTION

Due to large scale development in the gaming sensor industry there have been a splurge of new RGBD sensors into the market in the recent years. Some of them are Microsoft Kinect, Asus Xtion Pro, Structure IO and many more. Also, the development of 3D printers and the cost-effectiveness to 3D print an object for prototyping has increased the need for 3D object scanners and recent advances in grasping control has elevated the need for accurate and fast 6D (3 for translation and 3 for rotation) pose estimation of the object. Fast-object scanners are generally bulky and need a lot of processing power, hence the graphics and vision community has employed a lot of point cloud matching techniques to solve this problem in a small form factor with high-speed.

In this paper, we combine ideas from different variants of ICP to achieve a significant speed boost and the results are compared against Vicon ground truth pose of the camera (by backtracking camera pose using object pose). A comparison of different variants of ICP is also presented in this paper. The rest of the paper is organized as follows: Section II talks about the pre-processing techniques to extract the object from the frame. Section III presents two variants of the ICP used to solve the problem of pose estimation. Methods to speed up ICP are presented in Section IV. Denoising methods for the final RGB model are presented in Section V. Experimental setup and analysis of results are presented in Section VI. The paper is concluded in Section VII and scope for future work is also presented.

The overall pipeline is as follows: The input RGBD data is sub-sampled and fed into a ICP algorithm. The ICP algorithm is run multiple times in a cascaded fashion which truncation between each step. This is done for all the frames. De-noising is performed on the final point cloud to obtain a clean 3D recreated object.

The data obtained from the RGBD point cloud has floor and other non-object points. We need to extract the candidate point cloud.

[1]Nitin J. Sanket is a MSE in Robotics Student at University of Pennsylvania nitinsan@seas.upenn.edu

[2]Daniel D. Lee is a Professor in Electrical and Systems Engineering Department at University of Pennsylvania ddlee@seas.upenn.edu

II. OBJECT DETECTION USING RANSAC AND OUTLIER REJECTION

The object is assumed to be kept on a planar surface with the object needed being the biggest object on the plane. RANSAC is used to find the largest plane in the image - the ground/planar surface. The RANSAC algorithm is described in Algorithm 1.

Data: p (3D point cloud), τ, MaxIter, IR
Result: o (object candidate point cloud)
while *t<MaxIter — InlierRatio>IR* **do**
  % Pick 3 points (A, B, C) at random from p % Fit a plane $(ax + by + cz + d = 0)$ to these 3 points

$$AB = B - A$$

$$AC = C - A$$

$$N = AB \times AC$$

  % N has the values of (a, b, c)

$$d = -A^T * N$$

  % Find outlier points o (object candidate points)

$$f(x) > \tau$$

  % Here, $f(x)$ is plugging in point x into the plane equation divided by the norm of N to measure residual and τ is the threshold
  % Find Inlier Ratio as ratio of number of inlier points to total number of points
end

Algorithm 1: RANSAC for Object Candidate Point Estimation

Sometimes, RANSAC does not remove all the non-object points due to sensor noise or RANSAC not finding the best plane. For this reason a post-filtering (outlier rejection) operation is performed. In the outlier rejection process, distance to each point from the mean of the RANSAC filtered point cloud is calculated, anything below a factor of the size of the model is retained as object candidates. The input image, RANSAC output and the outlier-rejected output are shown in Fig. 1.

III. 3D OBJECT POSE ESTIMATION USING ITERATIVE CLOSEST POINT

The aim of this section is to estimate the 6D pose of the object in some frame of reference. It is assumed that we

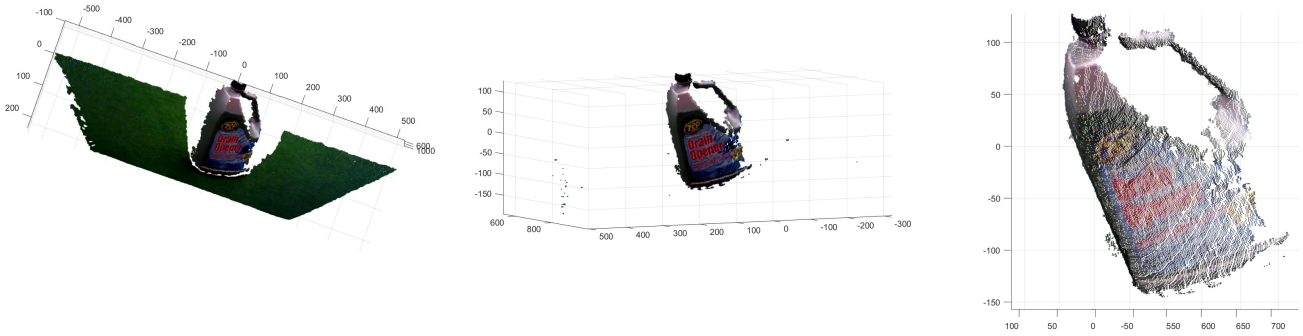

Fig. 1. Left to Right: Input Image, Output after RANSAC (notice stray points), Output after outlier-rejection (the clean object point cloud is obtained).

have the model of the object and we are trying to estimate the pose of the object in current camera frame. This can be formulated as finding the 6D transformation between two 3D point which minimizes the Sum of Square Errors (SSE). Let the two point sets (correspondences established) be p and q, now we are trying to solve the following optimization problem:

$$\min_{R,T} \sum_{\forall i} \|q_i - (Rp_i + T)\|^2$$

when $q_i = Rp_i + T$, here i is the data point number. Let us assume we have N data points in this case. Because the above optimization problem involves a rotation matrix R, the optimization problem is inherently linear. So obtaining a globally optimal solution is expensive and slow. We assume that the point-sets are close enough, i.e., we have a good initial guess and the locally optimal solution is the globally optimal solution. This is a very strong assumption but why this is reasonable in our case will be clear soon. In our case the first point set is the model data and the second point set are the candidate object points (extracted using RANSAC) from the RGBD point cloud. Now, consider solving the optimization problem using the standard ICP as discussed in the next Sub-section.

A. Point to Point ICP (P2P) [1]

First step int P2P is to mean center both point sets as given below:

$$\tilde{q}_i = q_i - \frac{1}{N}\sum_{\forall i} q_i$$
$$\tilde{p}_i = p_i - \frac{1}{N}\sum_{\forall i} p_i$$

Now, the optimization problem becomes

$$\min_{R,T} \sum_{\forall i} \|\tilde{q}_i - R\tilde{p}_i\|^2$$

The translation doesn't come into picture in the above equation because we mean-centered both the point sets.
Now, we need to solve for the rotation $\hat{R}$ (estimated value of R) and recover the translation $\hat{T}$ (estimated value of T) as

$$\hat{T} = q - \hat{R}p$$

To compute $\hat{R}$ using SVD the following steps are used.

$$H = \sum_{\forall i} \tilde{p}_i \tilde{q}_i^T$$

here T denoted the matrix transposition. Decompose H using SVD,

$$H = U\Lambda V^T$$
$$X = VU^T$$

If $\det X = 1 \Rightarrow \hat{R} = X$.
If $\det X = -1 \Rightarrow$, the algorithm failed.
For a texture less object P2P estimates are generally bad. P2P is generally very slow to converge. One has to run P2P multiple times to get a good estimate of the transformation between 2 point sets. The closer the point sets initially (i.e., the closer the transformation to identity) the faster is the convergence. To accelerate the convergence a different error metric than SSE between point to point is used. The new error metric used tries to minimize the squared distance between a point in p and the tangential plane in q. This is discussed in the next Sub-section.

B. Point to Plane ICP (P2Pl) [2]

We will be using the same notation (with p and q are swapped) used previously. Now, consider that the normal vector at q_i is given by the vector $n_i = \begin{bmatrix} n_{ix} & n_{iy} & n_{iz} \end{bmatrix}^T$. The optimization problem is given by (Refer to Fig. 2 for a visualization of the optimization function),

$$\min_{R,T} \sum_{\forall i} \|(q_i - (Rp_i + T)) \bullet n_i\|^2$$

We linearize the problem assuming that the angles are small. The original rotation matrix is given by,

$$R(\alpha,\beta,\gamma) = \begin{bmatrix} c_\gamma c_\beta & -s_\gamma c_\alpha + c_\gamma s_\beta s_\alpha & s_\gamma s_\alpha + c_\gamma s_\beta s_\alpha \\ s_\gamma c_\beta & c_\gamma c_\alpha + s_\gamma s_\beta s_\alpha & -c_\gamma s_\alpha + s_\gamma s_\beta c_\alpha \\ -s_\beta & c_\beta s_\alpha & c_\beta c_\alpha \end{bmatrix}$$

Where $c_\theta = \cos\theta$ and $s_\theta = \sin\theta$. The linearized rotation matrix is given by,

$$\hat{R} = \begin{bmatrix} 1 & \alpha\beta - \gamma & \alpha\gamma + \beta \\ \gamma & \alpha\beta\gamma + 1 & \beta\gamma - \alpha \\ -\beta & \alpha & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix}$$

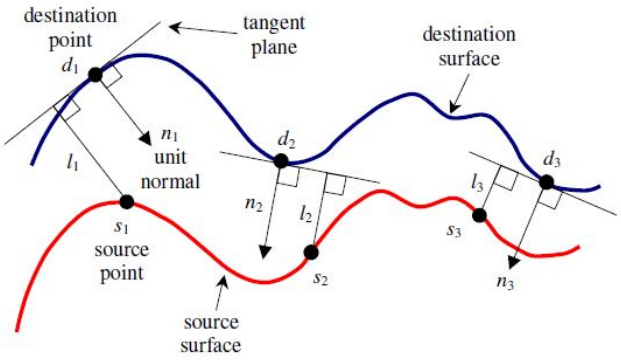

Fig. 2. Visualization of Point 2 Plane error metric [2].

We can replace the actual R in the optimization problem by approximated $\hat{R}$.

$$\min_{\hat{R},T} \sum_{\forall i} \left\| \left(q_i - \left(\hat{R} p_i + T \right) \right) \bullet n_i \right\|^2$$

This can be written as a linear system of equation $Ax = b$ where the terms A, b and x are given below.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & n_{1x} & n_{1y} & n_{1z} \\ a_{21} & a_{22} & a_{23} & n_{2x} & n_{2y} & n_{2z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & n_{Nx} & n_{Ny} & n_{Nz} \end{bmatrix}$$

$$\begin{aligned} a_{i1} &= n_{iz} p_{iy} - n_{iy} p_{iz} \\ a_{i2} &= n_{ix} p_{iz} - n_{iz} p_{ix} \\ a_{i3} &= n_{iy} p_{ix} - n_{ix} p_{iy} \end{aligned}$$

$$b_i = n_{ix} q_{ix} + n_{iy} q_{iy} + n_{iz} q_{iz} - n_{ix} p_{ix} - n_{iy} p_{iy} - n_{iz} p_{iz}$$

$$x = \begin{bmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{bmatrix}^T$$

$$T = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$$

Now, x can be solved using the pseudo-inverse of A i.e., $A^\dagger$. The solution is

$$\hat{x} = A^\dagger b$$

The rotation matrix can be reconstructed using the non-approximated formula and the last three elements of $\hat{x}$ represent the translation. Again, this algorithm is run multiple times which each run having the output of the previous run as the initial guess to obtain the final pose estimate. It is observed that P2Pl method is much faster to converge than P2P method. More quantitative analysis is provided in the results Section. The next section talks about approaches to speed up the ICP algorithm.

IV. SPEEDING UP ICP [3]

A. Sub-sampling in Normal Space

The model point cloud data has a lot of redundant points and all points need not be used for matching. Only the points which are around the edges have highly useful information. This is analogous to edges and corners being better image features. A simple way to sub-sample points is to randomly sample points in clustered normal space. We run K-means clustering (K chosen by cross validation) to cluster the points in normal space or $S2$. The distance metric used is a cosine distance which is given by,

$$d_{pq} = 1 - \frac{pq^T}{\sqrt{(pp^T)\,(qq^T)}}$$

Then a chosen number the points (again obtained from cross-validation) are randomly and uniformly sampled from each cluster. This sub-sampled point set is used as the input to the ICP algorithm. Note that this process is done on both the model point set and the data point set. This gives a huge speed-up and the times are presented in the results. A sample output of K-Means clustering in normal space is shown as $S2$ plot in Fig. 3 (for the drill). In the left sub-plot, different colors show different clusters, 125 clusters were used for the ease of visualization though in reality 512 clusters are used (obtained by cross-validation). In the right sub-plot, the red points represent all the data points and blue points show the sub-sampled data points. Similar outputs for Liquid Container are shown in Fig. 4. The final sub-sampled points using 512 clusters used in the experiments section are shown in Fig. 5.

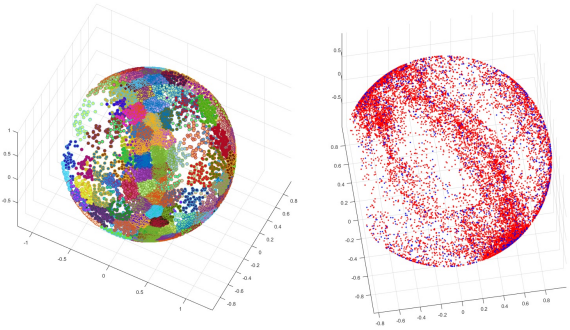

Fig. 3. Left: $S2$ Plot of 125 clusters clustered using K-Means algorithm, Right: $S2$ Plot of data points and sub-sampled data-points (red: all, blue: selected) for Drill Model.

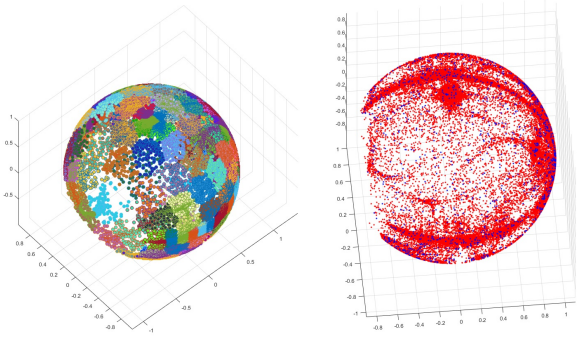

Fig. 4. Left: $S2$ Plot of 125 clusters clustered using K-Means algorithm, Right: $S2$ Plot of data points and sub-sampled data-points (red: all, blue: selected) for Liquid Container Model.

B. Truncation based on Ranking in Euclidian Space

Truncation is the process of discarding some data which is deemed useless. After the point correspondences are found using KNN, the points are arranged in ascending order of the euclidian distances in 3D. The points corresponding to highest (bottom most) 15% (found using cross-validation) distances are discarded. This makes the algorithm slightly faster and more robust to outliers. More comparisons with and without using truncation are presented in the results section. Fig. 6 shows how truncation removes points which might be outliers. The green points indicate the sub-sampled model points, the blue points indicate the object candidate points and the red points show the points removed by truncation. We can observe that the far side of the handle are removed which the algorithm thinks as noise because it is not consistent with the current orientation.

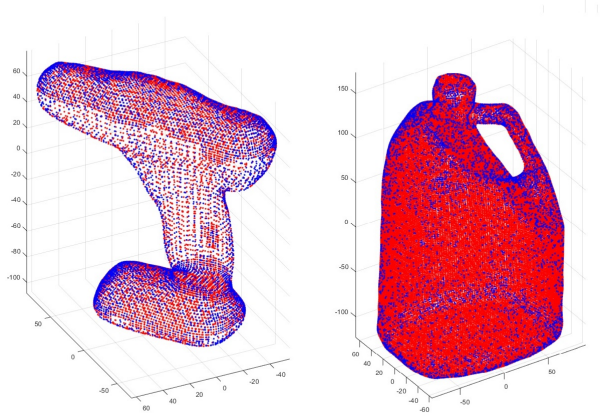

Fig. 5. Left: Sub-sampled Drill Model, Right: Sub-sampled Liquid Container Model (512 clusters used in both cases and 10 samples per class are retained).

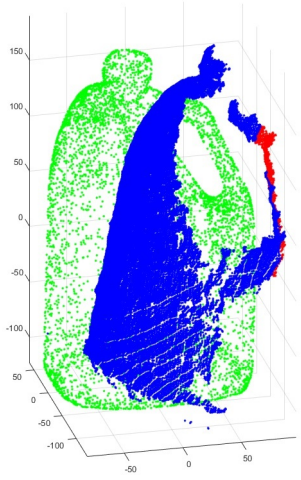

Fig. 6. Truncation procedure output, Green: Sub-sampled model, Blue: Retained Points, Red: Points rejected by truncation.

V. DE-NOISING OBTAINED OBJECT MODEL

The ICP algorithm with sub-sampling and truncation are run for all the frames. The point cloud output from each frame is transformed into a common frame (Vicon frame in our case) and collated. To obtain a 'clean' object model a simple de-noising approach is used. A plane is constructed for points in a neighborhood of 10 points (chosen from cross-validation). Any point in this neighborhood whose average distance to its 10 neighbors is above 2 times the standard deviation is considered an outlier an is discarded. This gives us a 'cleaner' looking output. Figs. Figs. 12, 11, 14 and 13 show the denoised outputs with raw outputs for different methods.

VI. EXPERIMENTS AND RESULTS

The experimental setup was frames form a video sequence, a full color reconstruction of the object with object and camera pose tracking was performed. The estimated camera poses were compared against Vicon camera poses. Because the object did not move, translation error does not make sense. Hence to visualize drift in translation the full reconstruction of the point cloud is presented, drift, if any will be easily noticeable. Also, a standard deviation measure for the distance is presented (this should be bounded within the object size accounting for different faces of the object). The problem setup is treated such that we have a good initial pose estimate for the first frame (obtained by coarse search) and the subsequent key-frames (every 10 frames) use the previous pose output as the initialization to make the convergence fast.

All the videos have 2 sub-plots, the left plot shows the final collated RGB point cloud (Updated every keyframe), the right sub-plot shows the iteration-wise convergence using ICP (red is the sub-sampled model and blue is the transformed candidate points). Videos (Figs.) 15, 16, 17 and 18 show the outputs of P2P (drill), P2Pl (drill), P2P (liquid container) and P2Pl (liquid container) for every 50 frame skip with cascade initialization (first frame initialized to same value for a particular dataset and all methods).

A comparison of Vicon's estimate of camera pose and pose estimated using ICP for Drill using P2P and P2Pl are shown in Figs. 7 and 8 respectively. The same plots for liquid container are shown in Figs. 9 and 10. The reconstructed object for Drill dataset using P2P and P2Pl are shown in Figs. 11 and 12 respectively. One can easily observe in Fig. 11 that the drill object is blurry as the pose estimates are not very good in the P2P approach (Refer Fig. 7). This is due to the fact that drill has very few points and it is easy for P2P to get stuck in local minima. Fig. 12 shows a crisp point cloud generated with P2Pl showing its robustness to fewer points and this is also demonstrated by good pose estimation in Fig. 8. The reconstructed object for Liquid Container dataset using P2P and P2Pl are shown in Figs. 13 and 14 respectively. In this dataset, due to the presence of huge number of points both P2P and P2Pl perform well. Though the point cloud obtained by P2Pl is much crisper than P2P again

demonstrating the robustness of P2Pl algorithm. This is the same trend observed in the pose estimates in Figs. 9 and 10. P2Pl follows vicon much closer than P2P approach.

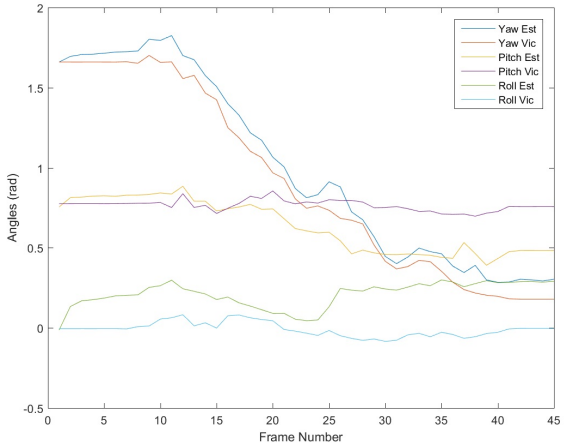

Fig. 7. Comparison of Vicon estimates of camera pose and pose estimated using ICP for Drill dataset using P2P.

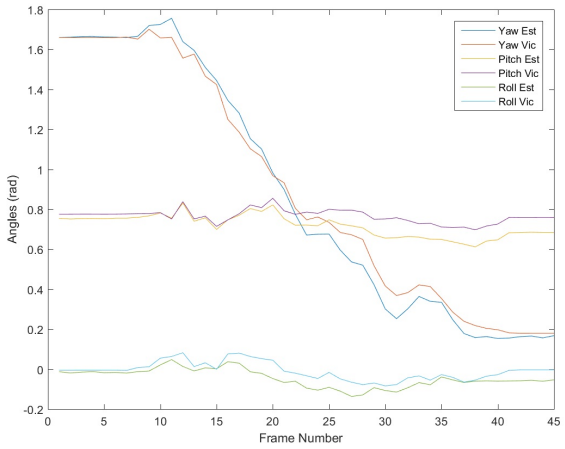

Fig. 8. Comparison of Vicon estimates of camera pose and pose estimated using ICP for Drill dataset using P2Pl.

A comparison of all the approaches with their times is shown in Table I. We can observe that in the drill dataset (small dataset/lesser number of points) we achieve a speed-up of 4.7x compared to simple P2P approach. We also achieve a speed up of 7.8x for the Liquid Container dataset (larger dataset/more number of points) as compared to the simple P2P approach. We can clearly see how the speed-up helps when the number of points grow.

The translation error was bounded within 5cms in all dimensions and for both dataset. 5cms being much smaller than out model size, we can see that ICP handled translation very well.

I tried to use factor graphs and UKF to filter the pose

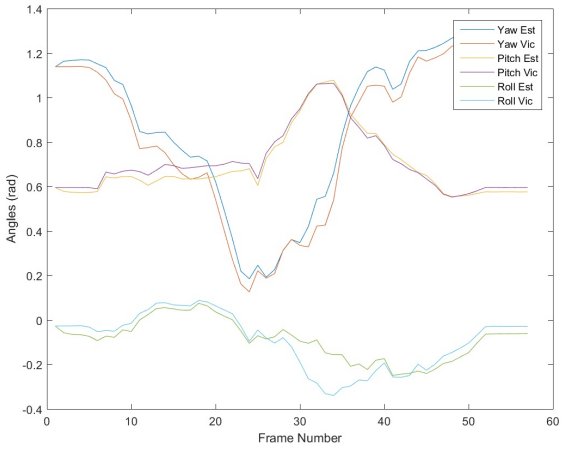

Fig. 9. Comparison of Vicon estimates of camera pose and pose estimated using ICP for Liquid Container dataset using P2P.

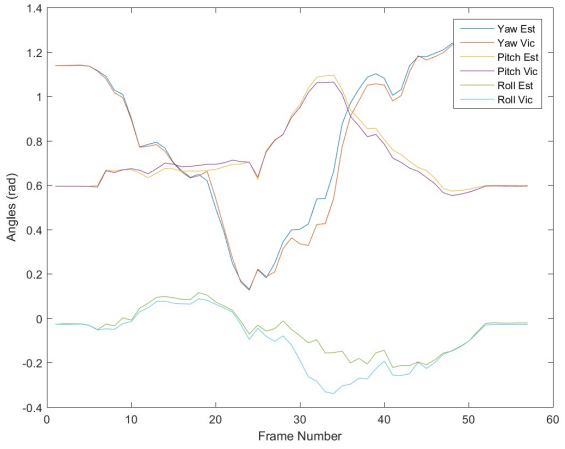

Fig. 10. Comparison of Vicon estimates of camera pose and pose estimated using ICP for Liquid Container dataset using P2Pl.

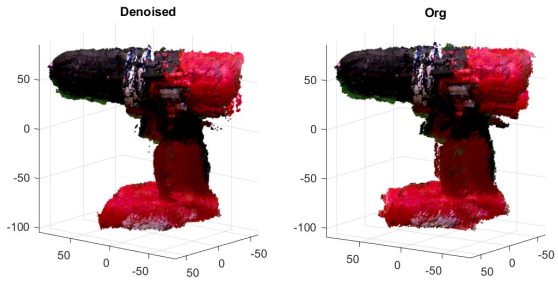

Fig. 11. Left: Denoised output, Right: Raw Output for Drill dataset using P2P (Frame Skip of 50).

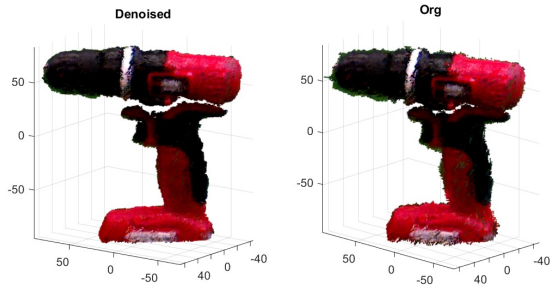

Fig. 12. Left: Denoised output, Right: Raw Output for Drill dataset using P2Pl (Frame Skip of 50).

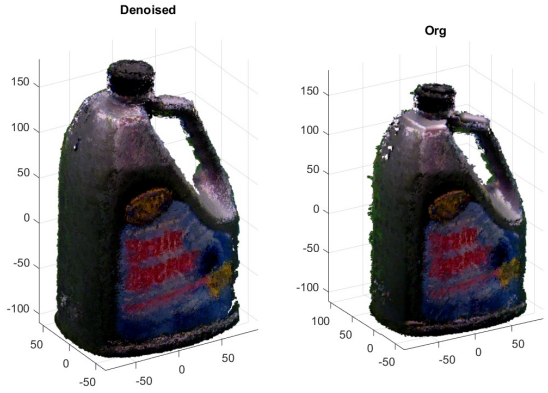

Fig. 13. Left: Denoised output, Right: Raw Output for Liquid Container dataset using P2P (Frame Skip of 50).

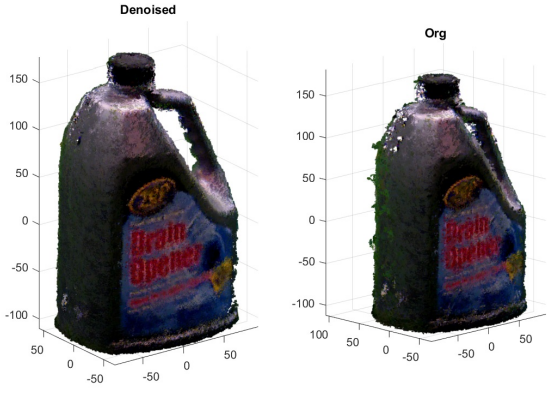

Fig. 14. Left: Denoised output, Right: Raw Output for Liquid Container dataset using P2P (Frame Skip of 50).

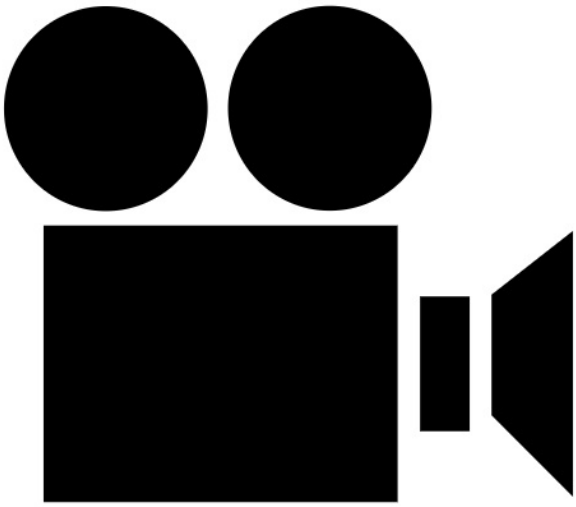

Fig. 15. Video showing the execution of object reconstruction for Drill dataset using P2P (0.35x speed).

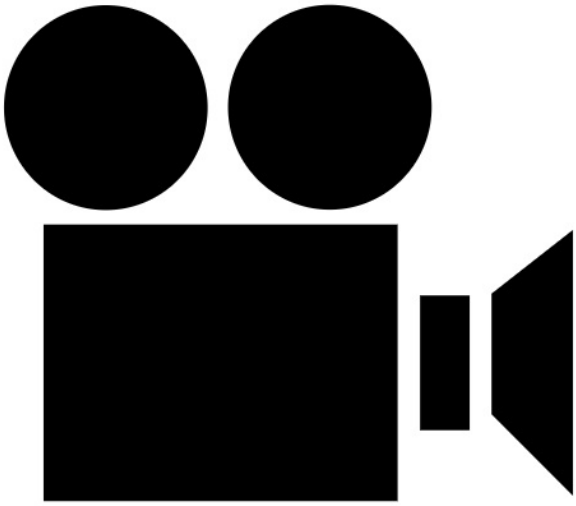

Fig. 16. Video showing the execution of object reconstruction for Drill dataset using P2Pl (0.125x speed).

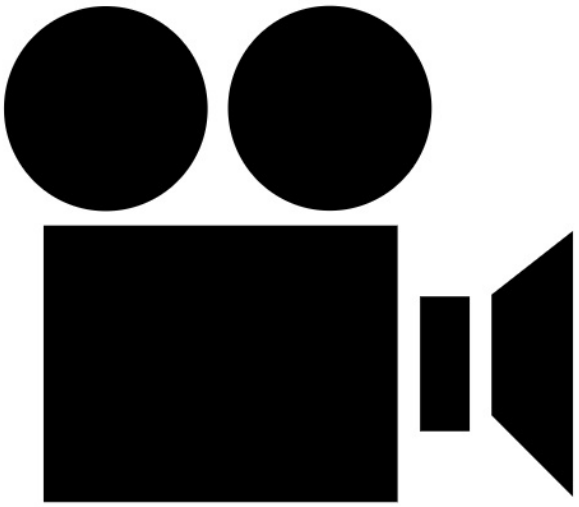

Fig. 17. Video showing the execution of object reconstruction for Liquid Container dataset using P2P.

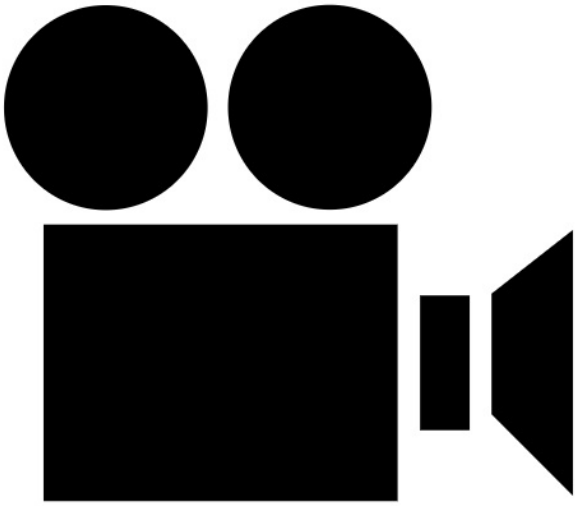

Fig. 18. Video showing the execution of object reconstruction for Liquid Container dataset using P2P.

TABLE I

COMPARISON OF MULTIPLE METHODS AND THEIR TIMING, DATASET 1 REFERS TO DRILL AND DATASET 2 REFERS TO LIQUID CONTAINER.

Method	Time/Iteration (s)	Dataset
P2P	3.01	1
P2P+Sub-sampling	1.69	1
P2P+Sub-sampling+Truncation	1.70	1
P2Pl	1.08	1
P2Pl+Sub-sampling	0.72	1
P2Pl+Sub-sampling+Truncation	0.64	1
P2P	7.83	2
P2P+Sub-sampling	2.41	2
P2P+Sub-sampling+Truncation	2.37	2
P2Pl	3.12	2
P2Pl+Sub-sampling	1.04	2
P2Pl+Sub-sampling+Truncation	1.01	2

estimates but they did not seem to work. They sort of acted like a low pass filter.

VII. CONCLUSIONS AND FUTURE WORK

A method for estimating 3D object pose and reconstruction using RGBD data and a variant of ICP is presented in this paper. The results are compared with vicon ground truth and have been shown to be sub degree and sub centimeter accurate. Different ways to speed up the execution like sub-sampling or truncation are presented which achieve a speed up of at-least 4.5x as compared to the standard approach while retaining the same accuracy if not better (due to robust selection of points). The P2Pl approach outperforms P2P approach in all aspects. Sub-sampling and truncation not only help in speed up but also add to the robustness of the algorithm. The algorithm can operate at 4-5fps on a i7 laptop when tracking using MATLAB.

Some of the aspects we would like to tackle in the future would be to use a globally optimal approach instead of coarse search for first frame initialization to make the algorithm faster and more reliable. The authors would also like to explore mean shift clustering instead of K-means clustering for sub-sampling.

ACKNOWLEDGMENT

The authors would like to thank Bhoram Lee and Jinwook Huh for their immense help in accomplishing this project.

REFERENCES

[1] Arun, K. Somani, Thomas S. Huang, and Steven D. Blostein, *Least-squares fitting of two 3-D point sets*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 5, pp. 698–700, 1987.

[2] Low, Kok-Lim, *Linear least-squares optimization for point-to-plane icp surface registration*, University of North Carolina Chapel Hill, Vol. 4, 2004.

[3] Rusinkiewicz, Szymon, and Marc Levoy, *Efficient variants of the ICP algorithm*, Proceedings of 3-D Digital Imaging and Modeling, 2001.