Recognition and Pose Estimation of 3D Objects Through 3D Features

Tolga Birdal
Ph.D. Candidate, Technical Univesity of Munich
July 6, 2014

This document aims to summarize the developments and the theoretical achievements covered throughout the first term of Google Summer of Code. The developments so far are heavily based on [1].

1 COMPUTATION OF POINT PAIR FEATURES (PPF)

2 INITIAL COMPUTATION OF OBJECT POSE GIVEN PPF

Let me summarize the following notation:

- p_m^i: i^{th} point of the model (p_m^j accordingly)
- n_m^i: Normal of the i^{th} point of the model (n_m^j accordingly)
- p_s^i: i^{th} point of the scene (p_s^j accordingly)
- n_s^i: Normal of the i^{th} point of the scene (n_s^j accordingly)
- $T_{m\to g}$: The transformation required to translate p_m^i to the origin and rotate its normal n_m^i onto the x-axis.
- $R_{m\to g}$: Rotational component of $T_{m\to g}$.
- $t_{m\to g}$: Translational component of $T_{m\to g}$.

- $(p_m^i)'$: i^{th} point of the model transformed by $T_{m\to g}$. ($(p_m^j)'$ accordingly).
- $\mathbf{R_{m\to g}}$: Axis angle representation of rotation $R_{m\to g}$.
- $\theta_{m\to g}$: The angular component of the axis angle representation $\mathbf{R_{m\to g}}$.

2.1 Transforming a Point Pair Onto the Ground Plane

The transformation in a point pair feature is computed by first finding the transformation $T_{m\to g}$ from the first point, and applying the same transformation to the second one. Transforming each point, together with the normal, to the ground plane leaves us with an angle to find out, during a comparison with a new point pair.

We could now simply start writing

$$(p_m^i)' = T_{m\to g} p_m^i$$

where

$$T_{m\to g} = -t_{m\to g} R_{m\to g}$$

Note that this is nothing but a stacked transformation. The translational component $t_{m\to g}$ reads

$$t_{m\to g} = -R_{m\to g} p_m^i$$

and the rotational being

$$\theta_{m\to g} = \cos^{-1}(n_m^i \cdot \mathbf{x})$$
$$\mathbf{R_{m\to g}} = n_m^i \wedge \mathbf{x}$$

in axis angle format. Note that bold refers to the vector form.

When the scene point p_s^i is also transformed on the same plane as $(p_m^i)'$, the points will be misaligned by a rotational component α. For the sake of efficiency, the paper splits it into two components α_s and α_m. Respectively, these denote the rotations from the transformed scene to the x-axis and from the transformed model to the x-axis. Luckily, both α_s and α_m are subject to the same procedure of computation, which reads as follows:

$$\alpha_m = \tan^{-1}\left(\frac{-(p_m^j)'_z}{(p_m^j)'_y}\right) \text{ for model}$$
$$\alpha_s = \tan^{-1}\left(\frac{-(p_s^j)'_z}{(p_s^j)'_y}\right) \quad \text{for scene}$$

using the fact that on x-plane x=0.

in the implementation, alphas are adjusted to be rotating towards 0.

2.2 Hough-like Voting Scheme

After both transformations the difference of the point pair features remain to be $\alpha = \alpha_m - \alpha_s$. This component carries the cue about the object pose. A Hough-like voting scheme is followed over the local model coordinate vector and α, which eventually recovers the object pose.

3 Pose Registration via ICP

The matching process terminates with the attainment of the pose. However, due to the multiple matching points, erroneous hypothesis, pose averaging and etc. such pose is very open to noise and many times is far from being perfect. Although the visual results obtained in that stage are pleasing, the quantitative evaluation shows 10 degrees variation, which is an acceptable threshold. Many times, the requirement might be set well beyond this margin and it is desired to refine the computed pose.

Furthermore, in typical RGBD scenes, the depth maps can capture only less than half of the model due to the visibility in the scene. Therefore, a robust pose refinement algorithm, which can register occluded and partially visible shapes quickly and correctly is not an unrealistic wish.

At this point, a trivial option would be to use the well known iterative closest point algorithm [**?**]. However, utilization of the basic ICP leads to slow convergence, bad registration, outlier sensitivity and failure to register partial shapes. Thus, it is definitely not suited to the problem. For this reason, many variants have been proposed [**?**]. Different variants contribute to different stages of the pose estimation process.

ICP is composed of 6 stages and the improvements I propose for each stage is summarized below.

3.1 Sampling

To improve convergence speed and computation time, it is common to use less points than the model actually has. However, sampling the correct points to register is an issue in itself. The naive way would be to sample uniformly and hope to get a reasonable subset. More smarter ways try to identify the critical points, which are found to highly contribute to the registration process. Gelfand et. al. [**?**] exploit the covariance matrix in order to constrain the eigenspace, so that a set of points which affect both translation and rotation are used. This is a clever way of subsampling, which I will optionally be using in the implementation.

A result of the smart covariance based sampling is shown in Figure 3.1.

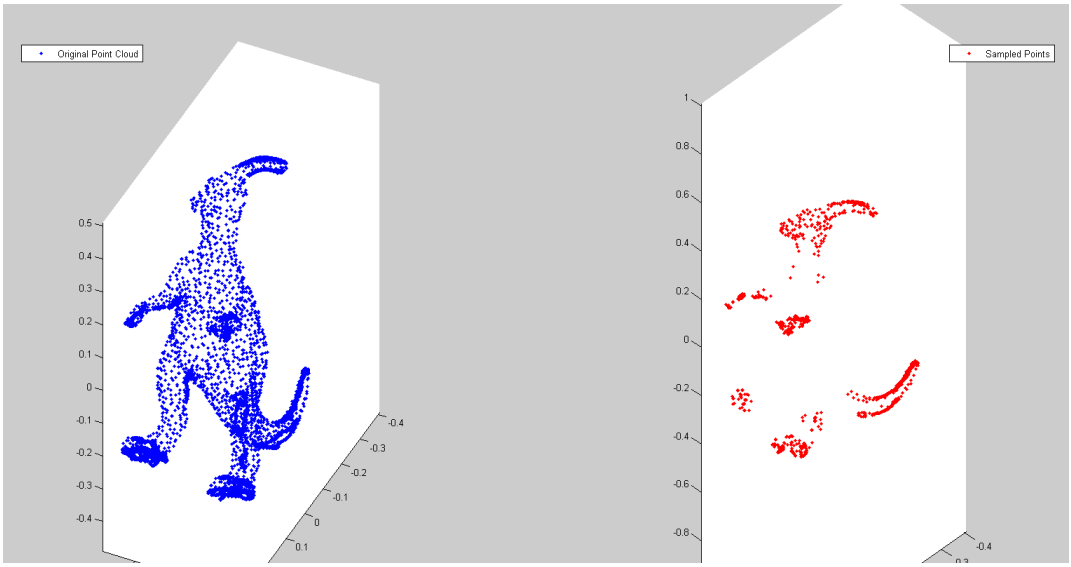

Figure 3.1: Registration in presence of occlusion, noise and large pose variation

3.2 Correspondence Search

As the name implies, this step is actually the assignment of the points in the data and the model in a closest point fashion. Correct assignments will lead to a correct pose, where wrong assignments strongly degrade the result. In general, KD-trees are used in the search of nearest neighbors, to increase the speed. However this is not an optimality guarantee and many times causes wrong points to be matched. Luckily the assignments are corrected over iterations.

To overcome some of the limitations, Picky ICP [**?**] and BC-ICP (ICP using bi-unique correspondences) [**?**] are two well-known methods. Picky ICP first finds the correspondences in the old-fashioned way and then among the resulting corresponding pairs, if more than one scene point p_i is assigned to the same model point m_j, it selects p_i that corresponds to the minimum distance. BC-ICP on the other hand, allows multiple correspondences first and then resolves the assignments by establishing bi-unique correspondences. It also defines a novel no-correspondence outlier, which intrinsically eases the process of identifying outliers.

For reference, both methods are used. Because P-ICP is a bit faster, with not-so-significant performance drawback, it will be the method of choice in refinment of correspondences.

3.3 Weighting of Pairs

In my implementation, I currently do not use a weighting scheme. But the common approaches involve *normal compatibility* ($w_i = n_i^1 \cdot n_j^2$) or assigning lower weights to point pairs with greater distances ($w = 1 - \frac{||dist(m_i,s_i)||_2}{dist_{max}}$) [**?**].

3.4 Rejection of Pairs

The rejections are done using a dynamic thresholding based on a robust estimate of the standard deviation. In other words, in each iteration, I find the MAD estimate of the Std. Dev. I denote this as mad_i. I reject the pairs with distances $d_i > \tau mad_i$. Here τ is the threshold of rejection and by default set to 3. The weighting is applied prior to Picky refinement, explained in the previous stage.

3.5 Error Metric

As described in [**?**], a linearization of point to plane error metric is used. This both speeds up the registration process and improves convergence.

3.6 Minimization

Even though many non-linear optimizers (such as Levenberg Mardquardt) are proposed, due to the linearization in the previous step, pose estimation reduces to solving a linear system of equations. This is what I do exactly.

3.7 ICP Algorithm

Having described the steps above, here I summarize the layout of the icp algorithm.

3.8 Efficient ICP Through Point Cloud Pyramids

3.9 Visual Results

3.9.1 Results on Synthetic Data

Figure 3.2 shows the result of the registration when the scene is partially observed and subject to random uniform noise. More over the initial pose set out to be $[\theta_x, \theta_y, \theta_z, t_x, t_y, t_z] = []$

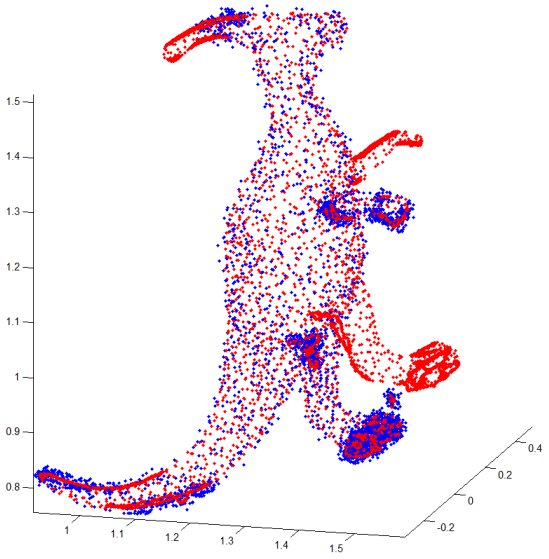

Figure 3.2: Registration in presence of occlusion, noise and large pose variation

REFERENCES

[1] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition." in *CVPR*. IEEE, 2010, pp. 998–1005. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#DrostUNI10