3D Point Cloud Project Report - Generalized ICP

François Darmon
Master MVA
francois.darmon@telecom-paristech.fr

Abstract

In this project, we implement[1] the three registration methods detailed in [4]. These three methods can be expressed using the same probabilistic framework. We details the algorithm and their link with the framework then we analyze the differences of these methods using synthetic data.

1 Introduction

Registration is a key step in the acquisition of point clouds. It is the step that finds the geometric transformation between two point clouds of the same scene that have been acquired from different location. The geometric transformation is a relatively simple one: due to the point cloud nature, the transformation must be a composition of a rotation and a translation. The easiest way to find the transformation from one cloud to another would be to use reference points that match in both clouds. It is then easy to find the geometric transformation from one cloud to the other. However, such reference points are not always available. It can be costly to use targets when acquiring the data or to find reference points manually.

The approach used by most algorithm such as ICP (Iterative Closest Point) does not use such known matches. ICP is an iterative algorithm. It computes at each step the matches to each point and find the best transformation for these correspondances. Then, the procedure is iterated on the computed matches with the new transformation. A pseudo-code of ICP can be found in Algorithm 1.

Algorithm 1: ICP principle

input : Reference point cloud A of size $n_A \cdot d$
Data point cloud B of size $n_B \cdot d$
Initial transformation R_0, t_0 from B to A
output: Transformation R, t from B to A

1 $R \leftarrow R_0$
2 $t \leftarrow t_0$
3 **while** *not converged* **do**
4 **for** $i \leftarrow 1 : n_B$ **do**
5 $m_i \leftarrow$ `FindClosestPointInA`$(R \cdot b_i + t)$
6 **end**
7 $R, t \leftarrow$ `FindBestTransformationBetween`$(m_i, R \cdot b_i + t)$
8 **end**

The important steps in algorithm 1 are at line 5 and 7. At line 5, a matching point m_i in A is found for every point $R \cdot b_i + t$. In order to keep the algorithm simple, the most simple choice is made: select the point that has the smallest euclidian distance. Line 7 consists in finding the best transformation that transform each point b_i into their match m_i. The best transformation means

[1] The Python 3.6 code can be found at `https://github.com/fdarmon/Generalized_ICP`

that the transformation minimizes a specific *cost function*. The choice of the cost function is the difference between the three methods presented in [4].

2 General Framework

2.1 Probabilistic model

In this section, we describe the probabilistic framework used to find the best rigid transformation between two point cloud. For simplicity reasons, we suppose that there is a one-to-one correspondence between the clouds: a_i in point cloud A corresponds to b_i in point cloud B.

We build a probabilistic model for the point clouds. We suppose that each point a_i and b_i has been sampled respectively from $\mathcal{N}(\hat{a}_i, C_i^A)$ and $\mathcal{N}(\hat{b}_i, C_i^B)$.$\hat{a}_i$ and $\hat{b}_i$ are the true positions of the points a_i and b_i. C_i^A and C_i^B are known covariance matrices. Let $d_i^{R,t} = b_i - Ra_i - t$. Then for any transformation (R, t) and for the true transformation (R^*, t^*)

$$\begin{aligned} d_i^{R,t} &\sim \mathcal{N}(\hat{b}_i - R\hat{a}_i - t, C_i^B + RC_i^A R^T) \\ d_i^{R^*,t^*} &\sim \mathcal{N}(0, C_i^B + R^* C_i^A (R^*)^T) \end{aligned}$$

The estimation of R^*, t^* can be done using MLE (Maximum Likelihood Estimation). [4] suggests to use conjugate gradient descent.

2.2 Implementation

The log-likelihood to maximize is

$$L(R,t) = \sum_{i=1}^{n} \log\left(\det\left(C_i^B + RC_i^A R^T\right)\right) - \tfrac{1}{2}\left(d_i^{R,t}\right)^T \left(C_i^B + RC_i^A R^T\right)^{-1} d_i^{R,t})$$

However, it is non convex and hard to maximize. A way to deal with this issue is to adopt an EM-like approach and fix the current estimation of (R, t) in the covariance in order to compute new MLE estimation of (R, t). Using this approach, optimization problem can be simplified to the minimization of the loss function

$$l(R,t) = \sum_{i=1}^{n} \left(\left(d_i^{R,t}\right)^T M_i d_i^{R,t}\right) \tag{1}$$

where $M_i = \left(C_i^B + RC_i^A (R)^T\right)^{-1}$. This also simplifies the gradient computation. Using the following notation for the residual $r_i = b_i - Ra_i - t$, the gradient is:

$$\begin{aligned} \frac{\partial l}{\partial t} &= -2\sum_{i=1}^{n} M_i r_i \\ \frac{\partial l}{\partial R} &= -2\sum_{i=1}^{n} M_i r_i a_i^T \end{aligned}$$

The conjugate gradient descent cannot be applied as is because it is a constrained optimization problem with the constraint that R must be a rotation matrix. We used the Euler parametrization of the rotation to address this issue. The advantage of this parametrization is that it takes 3 independent parameters, $(\theta_x, \theta_y, \theta_z)$. Using this formalism, the rotation matrix can be written as $R = R_{\theta_z} R_{\theta_y} R_{\theta_x}$

where: R_{θ_x},R_{θ_y},and R_{θ_z} are elementary rotations around axis x,y and z

$$R_{\theta_x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix}$$
$$R_{\theta_y} = \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix}$$
$$R_{\theta_z} = \begin{pmatrix} \cos\theta_x & -\sin\theta_x & 0 \\ \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Using this notation, the gradient of R with respect to θ can be computed :

$$\frac{\partial R}{\partial \theta_x} = R_z R_y \frac{\partial R_x}{\partial \theta_x}$$
$$\frac{\partial R}{\partial \theta_y} = R_z \frac{\partial R_y}{\partial \theta_y} R_z$$
$$\frac{\partial R}{\partial \theta_z} = \frac{\partial R_z}{\partial \theta_z} R_y R_x$$

And then combine it with the previous gradient using chain rule :

$$\frac{\partial l}{\partial \theta} = \sum_{i,j=1}^{3} \frac{\partial l}{\partial R_{ij}} \frac{\partial R_{ij}}{\partial \theta}$$

Most of the formulas of this section were found in [3]. We implemented the MLE estimation using Scikit Learn [1] conjugate gradient optimization function. In our case we minimize a function with dimension 6 parameter $x = (t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$

3 Applications

Three methods are presented in [4]. *Point to point* and *Point to plane* are preexisting methods but the probabilistic framework encompasses these two methods. It also allows to create a new method *Plane to plane*.

3.1 Point to point

Point to point is the method that is used in standard ICP for the best rigid transformation computation. This method relies on the minimization of a simple function: the average squared distance between a point and its corresponding point in the other point cloud.

$$l = \sum_{i=1}^{n} d_i^{R,t} (d_i^{R,t})^T$$

This expression is really similar to Equation 1. In this particular case $M_i = 1$ so $C_i^B = 1$ and $C_i^A = 0$. This means that the assumption underlying *point to point* method is an isotropic distribution of b_i around its true value and a deterministic value of a_i. This model seems really simplistic but it performs well in practice.

One interesting aspect to note is that M matrix does not depend on the current transformation. In this setting, there is no need to alternate the EM-like steps for minimizing the loss function. It is already convex and the gradient is easy to compute.

There exist a closed form solution for the minimization of the loss but for the sake of comparability, we performed a conjugate gradient method for the minimization.

3.2 Point to plane

Point to plane method [2] uses the structure of the point cloud. Since a point cloud is a collection of points sampled from a surface, the data has a locally planar structure that can be used.

Instead of trying to minimize the distance between each point b_i and its corresponding point Ra_i+t, *Point to plane* assumes that the reference cloud is locally planar and tries to minimize the distance between each transformed point $Ra_i + t$ and the local plane of b_i. The distance between a point x and the plane corresponding to another point y is $||n \cdot (x - y)||^2$ where n is the normal at point y.

The loss function is then:

$$l = \sum_{i=1}^{n} ||n_i^T (Rb_i + t - a_i)||^2$$

For every point, the distance can be reformulated using the projection matrix P_i on the normal n_i at a_i:

$$\begin{aligned} l &= \sum_{i=1}^{n} ||n_i \cdot (Rb_i + t - a_i)||^2 \\ &= \sum_{i=1}^{n} (P_i d_i) (P_i d_i)^T \\ &= \sum_{i=1}^{n} d_i P_i d_i^T \end{aligned}$$

We used the fact that P_i is a projection matrix so $P_i P_i = P_i$. Equation 1 can be recognized with $M_i = P_i$. This method can then be implemented using the Generalized ICP framework. P_i matrix is computed in our implementation as $P_i = n_i n_i^T$ with n_i the normal of the point cloud at a_i. The normals are computed with a Singular Value Decomposition (SVD) of the neighboring points covariance for each point. This step is really computationally expensive because we perform a neighbor search and a SVD for every point of the reference cloud. In practice, this step usually takes as much time as the optimization of the cost function.

Similar to *Point to point*, matrices M_i do not depend on the current transformation parameters. The conjugate gradient can be performed once and for all without the need to alternate an expectation part and a maximization part.

3.3 Plane to plane

Plane to plane method is introduced in [4]. The idea is to use a similar planar modelisation as *Point to plane* but to take into account the symmetry of the problem. Instead of using the planarity of the reference plane only, this new method uses the planarity of both clouds.

In Generalized ICP framework, the planarity of the data is modelled in the covariance matrices. As the data is assumed to be sampled from a plane, one can expect the data to have high variance in the plane direction in comparison to the normal direction. We can model this with a small ϵ variance in the normal direction and a variance equal to 1 in the other directions:

$$C_i = R_{n_i} \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} R_{n_i}^T$$

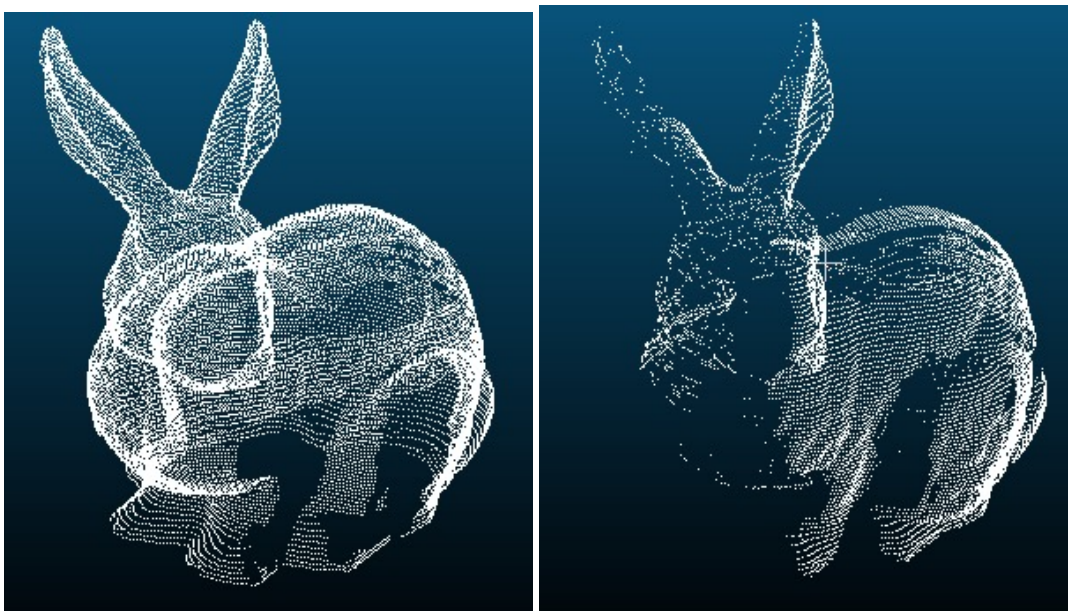
R_{n_i} is the rotation that bring e_1 the first axis of global coordinates to n_i. In practice, this rotation matrix is obtained by computing the Singular Value Decomposition (SVD) of the covariance of the points in the neighborhood of the point. The rotation is simply the matrix U of the SVD UDV^T.

This method needs the most processing power. Unlike *Point to plane*, the SVD on the neighbouring covariance have to be performed on every point of *both* point clouds. Also, contrary to the two other methods, M_i matrix of equation 1 depends on the current transformation. This adds complexity to the method because the estimation process has to alternate a computation of M_i matrices for every pairs of points and a conjugate gradient minimization of the cost function.

4 Experiments

4.1 Data

We used the data from the bunny point cloud. This point cloud has few points compared to other point clouds so it was possible to have results without heavy computation. The point cloud was split into two clouds with small overlapping. One cloud was randomly translated and rotated. The two point clouds are shown in Figure 1.



(a) Reference point cloud (b) Point cloud to align

Figure 1: Point cloud used for the experiments

4.2 Results

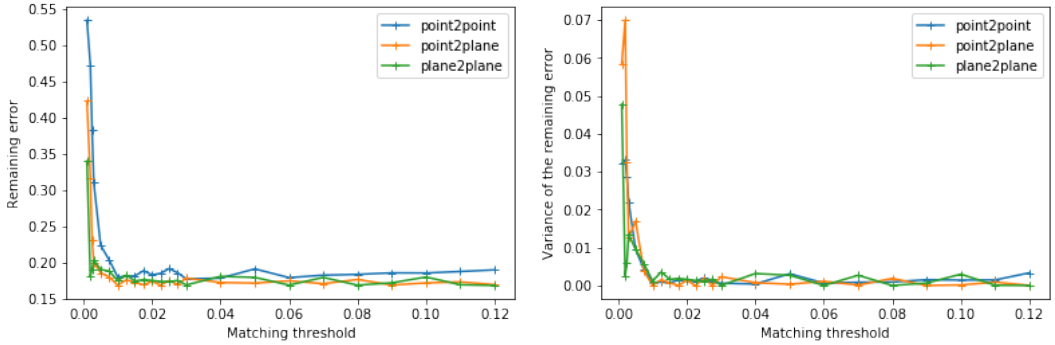
We compared the performance of the 3 methods in the alignmenent of the two clouds. In order to avarage the randomness of the initial transformation parameters, we performed the experiments 50 times. To assess the quality of a registration, we analyzed the RMS, the average distance between a point and its nearest neighbour. Figure 2 shows the mean and variance of RMS after convergence of the ICP as a function of the matching threshold. This threshold is used to discard pairs of points when their distance is too high.

One can notice in Figure 2 that *Point to point* method has generally worse performances than the two more complex methods. However, there is no clear tendency regarding the randomness of the result: each method has the same variance in the final RMS. As noted in [4] Plane to plane method seems to be the method that is the least sensible to the matching threshold value.

However, the result are not as impressive as in [4]. This may be due to the data chosen, the bunny point cloud is small and has no large planar section. Such planar section may advantage the two methods that use planar modelisation. Also, there are no occlusion or large missing part between the two point clouds. This makes the matching threshold useless. In that situation, assessing the sensitivity of a method to the matching threshold can be difficult to do.

5 Conclusion

Point to point remains a really useful method because it is the fastest to compute. There is no computation of the normals of the clouds and there exist a closed form solution of the optimization



(a) Mean RMS after convergence

(b) Variance of the RMS after convergence

Figure 2: Link between RMS distribution and matching threshold

problem. However, in some case, more complex methods that use the structure of the data can be useful.

Point to plane is a simple extension of *Point to point* that has the advantage of being simple and reasonably fast. *Plane to plane* is the slowest method for no huge performance improvements in our experiments. This method is still theoretically appealing because of the symmetry of the treatment of the reference cloud and the cloud to align. This method also has an interesting theoretical framework with the use of a probabilistic model of the data acquisition.

It would be interesting to test the methods on more diverse data like in [4] in order to exhibit the advantages of the modelisation of *Plane to plane*.

References

[1] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[2] Y. Chen and G. Medioni. "Object modeling by registration of multiple range images". In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. Apr. 1991, 2724–2729 vol.3. DOI: 10.1109/ROBOT.1991.132043.

[3] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20121115. Nov. 2012. URL: http://www2.imm.dtu.dk/pubdb/p.php?3274.

[4] A. Segal, D. Haehnel, and S. Thrun. "Generalized-ICP". In: *Proceedings of Robotics: Science and Systems*. Seattle, USA, June 2009. DOI: 10.15607/RSS.2009.V.021.