Scale-Invariant Fast Global Registration

Adrian Szatmari

March 25, 2018

Abstract

Histogram 3D feature descriptors are used to find candidate matches that cover the surfaces. The *invFPFH* is created, being a scale invariant version of the FPFH feature descriptor used in Fast Global Registration (FGR).

A single objective function is optimized to align the surfaces and disable false matches. The optimization achieves tight alignment. No correspondence updates or closest point queries are performed in the inner loop, thus significantly cutting running time.

Contents

1 Introduction

Different users and customers will likely provide 3D scans through various means. For instance, the 3D scan provided does not have the same scale as that of the 3D model. Different scans will have different properties, such as sampling density [1, 2, 3], relative scale [4], sampling noise and outliers [5, 6], relative initial position [5], or only sections of objects [7]. Whether to create 3D models or to register existing 3D models to scans, an industrial strength pipeline is necessary.

Aligning two sets $P = \{p_i \in R^3, i \in N_i\}$ and $Q = \{q_j \in R^3, j \in N_j\}$ is not an easy task. As of yet, no method has been completely successful in addressing all of the concerns at once.

Fast Global Registration [5], from 2016, distinguishes itself from the rest. Most methods, with some exceptions [8, 9, 10, 11], work by first establishing a set of most likely correspondences between P and Q. Let's write a correspondence as (p, q). Subsequently, they try to minimize

$$E(T) = \sum_{(p,q)} \|p - Tq\|$$

or an equivalent objective function [12, 13]. FGR does not require correspondence updates or closest-point queries in the minimization loop, saving a lot of computation. Moreover, the authors claim that FGR requires *no particular initial position* and that it is *robust to noise*. Unfortunately, FGR does not find the relative scaling factor between P and Q and it is not scale invariant.

In this report, preliminary steps towards scale invariance are taken for FGR. In Section 2, the original pipeline is presented; in Section 3, it is shown that the method is not scale invariant and some remedies are suggested; in Section 4, the processing of the data is presented as well as the challenges that were faced. In Section 5, a future work flow is suggested as well as some problems dual to point registration.

2 Fast Global Registration

In FGR, the putative matches are found through the popular FPFH descriptors and then by applying some filters; the optimization is done through a Black-Rangarajan decomposition of the objective function.

2.1 FPFH Descriptors

Point descriptors are generally histograms that bin some set of features describing the neighborhood of a point. Scientists often talk of global versus local descriptors, which usually addresses the size of the neighborhood. The computation of local descriptors is typically embarrassingly parallel. Examples are SIFT 3D, NARF, Spin Images, FPFH, and many others [14, 15, 16]. Currently, the most popular seems to be FPFH [16], which captures the curvature of a set of points in 3D.

Let $p_i, p_j \in P$, with corresponding estimated surface normals n_i and n_j, and let $u = n_i, v = (p_j - p_i) \times u$, and $w = u \times v$. Let $SPF(p_i)$ be the histogram of the following three features:

$$\begin{aligned} \alpha &= v \cdot n_j, \\ \phi &= (u \cdot (p_j - p_i))/\|p_j - p_i\|, \\ \theta &= \arctan(w \cdot n_j, u \cdot n_j), \end{aligned} \tag{1}$$

with 5 bins per feature. In other words, a Kd-tree is first built for P, then for each point $p_i \in P$ a certain k-neighborhood $nbh_k(p_i)$ is selected, $\alpha_j, \phi_j, \theta_j$ are computed for $p_j \in nbh_k(p_i)$ and then

binned. Therefore $SPF(p_i) = ([h_1, \ldots, h_5]_\alpha, [h_1, \ldots, h_5]_\phi, [h_1, \ldots, h_5]_\theta)$, where $[h_1, \ldots, h_5]_x$ is the histogram for x with $nbh_k(p_i)$ as support.

Finally $FPFH(p_i)$ is defined as a weighted mean of the $SPF(p_j)$ with $p_j \in nbh_k(p_i)$:

$$FPFH(p_i) = SPF(p_i) + \sum_{p_j \in nbh_k(p_i)} \frac{SPF(p_j)}{k\|p_j - p_i\|}. \tag{2}$$

2.2 Descriptor Matching

1. Most likely correspondences are established using $FPFH(P)$ and $FPFH(Q)$. FGR proposes to do a nearest neighbor search from $FPFH(P)$ to $FPFH(Q)$ and then vice versa. These two searches are done by building Kd-trees for both $FPFH(P)$ and $FPFH(Q)$. Note that in general $card(P) \neq card(Q)$. The authors call these initial correspondences K_1.

2. K_1 is first refined by the "Reciprocity Test" into K_2. The Reciprocity Test simply ensures that if $FPFH(p_i)$ is closest to $FPFH(q_j)$ as searched in the Kd-tree of $FPFH(Q)$, then $FPFH(q_j)$ is closest to $FPFH(p_i)$ in the Kd-tree of $FPFH(P)$. This is a good common sense requirement and it enforces the correspondences to be one-to-one.

3. K_2 is afterward refined by the "Tuple Test" into K_3. The Tuple Test is some sort of global coherence test. Namely for 3 randomly picked correspondence pairs $(p_1, q_1), (p_2, q_2), (p_3, q_3) \in K_2$, the authors require the following to be true:

$$\tau < \frac{\|p_i - p_j\|}{\|q_i - q_j\|} < 1/\tau, \quad \tau = 0.9, \quad \forall i \neq j, \tag{3}$$

   and successful triples are saved in K_3. This set is then fed to the optimization procedure.

2.3 Black-Rangarajan Decomposition

FGR minimizes the following equation for T:

$$E(T) = \sum_{(p,q) \in K_3} \rho(\|p - Tq\|), \tag{4}$$

where $\rho(x) = \frac{\mu x^2}{\mu + x^2}$ is the scaled German-McClure estimator [5]. This estimator penalizes small residuals in the least squares sense and rapidly neutralizes outliers. Objective 4 is hard to optimize directly, but the Black-Rangarajan duality permits the following decomposition:

$$E(T, L) = \sum_{(p,q) \in K_3} l_{p,q}\|p - Tq\|^2 + \sum_{(p,q) \in K_3} \mu(\sqrt{l_{p,q}} - 1)^2, \quad L = \{l_{p,q}\}. \tag{5}$$

In order to minimize $E(T, L)$, take the partial derivatives w.r.t. $l_{p.q}$ and solve for zero:

$$\frac{\partial E}{\partial l_{p,q}} = \|p - Tq\|^2 + \mu \frac{\sqrt{l_{p,q}} - 1}{\sqrt{l_{p,q}}} = 0, \tag{6}$$

which gives

$$l_{p,q} = \left(\frac{\mu}{\mu + \|p - Tq\|^2} \right)^2. \tag{7}$$

The optimization loop works by alternating between optimizing T and $L = \{l_{p,q}\}$. Since both optimization steps minimize the same global objective, the alternating loop converges to a minimum. When L is fixed, then T can be solved via the Gauss-Newton method and a linearization of the rigid motion element in $SE(3)$. When T is fixed, L can be updated via Eq. 7. For more details on implementation or theory look into the Matlab code or into the original paper.

Algorithm 1 Fast Global Registration

Input: A pair of surfaces P and Q
Output Transformation T that aligns Q to P

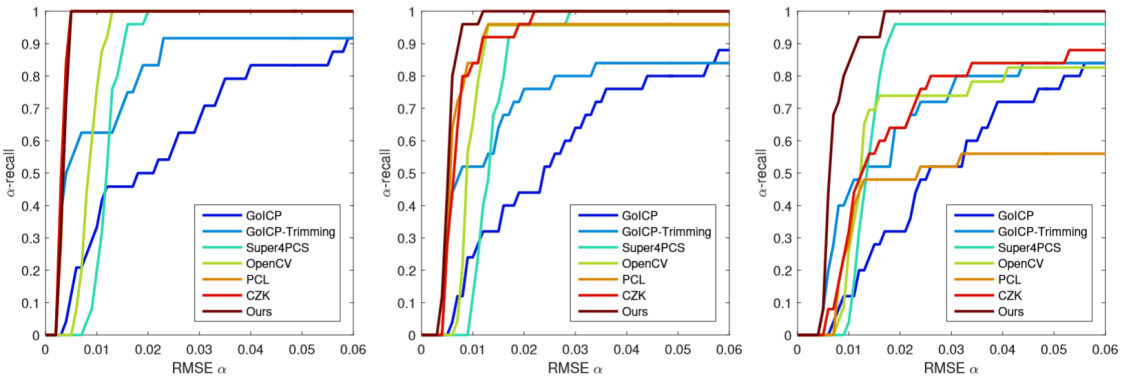
Compute normals n_P and n_Q;
Compute FPFH features $F(P)$ and $F(Q)$;
Build K_1 by computing nearest neighbors between $F(P)$ and $F(Q)$;
Apply "Reciprocity Test" on K_1 to get K_2;
Apply "Tuple Test" on K_2 to get K_3;
Initialize $T \leftarrow I$ and $\mu \leftarrow D^2$;
while not converged or $\mu >$ threshold **do**
    Every four iterations, $\mu \leftarrow \mu/2$;
    Compute and update L using K_3 via Eq. 7;
    Set up the linear system for T and solve it ($Ax = b$);
    Update T;
end while
Verify whether T aligns Q to P

The algorithm does not need to update K_3 directly. The vector L acts as an indicator function for the correspondence pairs. When $l_{p,q}$ is close to 0, the pair (p, q) contributes little to the optimization and when $l_{p,q}$ is close to 1 it contributes more. The authors suggest to initialize μ to the square of the diameter of the point set, and to halve it a every fourth iteration. This parameter acts as a sliding knot on the size of neighborhoods.

2.4 Performance

FGR's performances shine compared to other approaches. The graphs from the paper are presented. FGR is denoted as "Ours" in the figures. Note that the citations in the figures are from the FGR paper, and thus do not correspond to the references in this report.

Here are the graphs for the behavior w.r.t. noise on synthetic data.



(a) No noise (b) Gaussian noise, $\sigma = 0.0025$ (c) Gaussian noise, $\sigma = 0.005$

Figure 1: Controlled experiments on synthetic data. α-recall is the fraction of tests for which a given method achieves an RMSE $< \alpha$. Higher is better. The RMSE unit is the diameter of the surface. FGR algorithm is more robust to noise and is more accurate than prior approaches.

Here are the performances w.r.t. to noise and the precision of the registration.

	$\sigma = 0$		$\sigma = 0.0025$		$\sigma = 0.005$	
	Average RMSE	Maximal RMSE	Average RMSE	Maximal RMSE	Average RMSE	Maximal RMSE
GoICP [42]	0.029	0.130	0.032	0.133	0.037	0.127
GoICP-Trimming [42]	0.035	0.473	0.039	0.475	0.044	0.478
Super 4PCS [26]	0.012	0.019	0.014	0.029	0.017	0.095
OpenCV [8]	0.009	0.013	0.018	0.212	0.032	0.242
PCL [34, 19]	**0.003**	**0.005**	0.009	0.061	0.111	0.414
CZK [7]	**0.003**	**0.005**	0.008	0.022	0.035	0.274
Our approach	**0.003**	**0.005**	**0.006**	**0.011**	**0.008**	**0.017**

Figure 2: Average and maximal RMSE achieved by global registration algorithms on synthetic range images with noise level α. Maximal RMSE is the maximum among the 25 RMSE values obtained for individual pairwise registration tests. FGR outperforms other methods by a large margin when noise is present.

Here are the performance w.r.t. to the relative initial positioning of the point clouds.

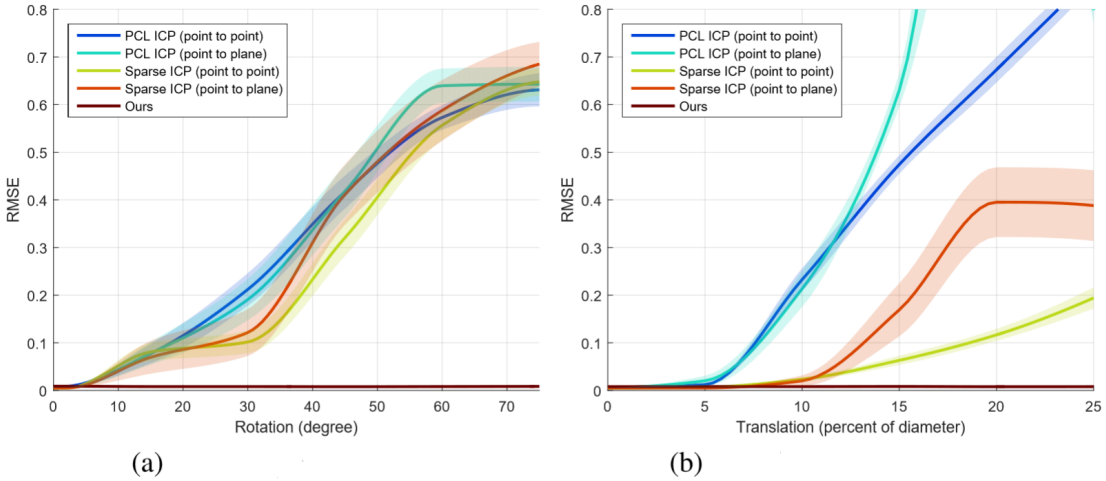


Figure 3: Controlled comparison with local methods. Local registration algorithms are initialized with a transformation generated by adding a perturbation in rotation (left) or translation (right) to the ground-truth alignment. The plots show the mean (bold curve) and standard deviation (shaded region) of the RMSE of each method. Lower is better.

Finally here are some more registration statistics for FGR.

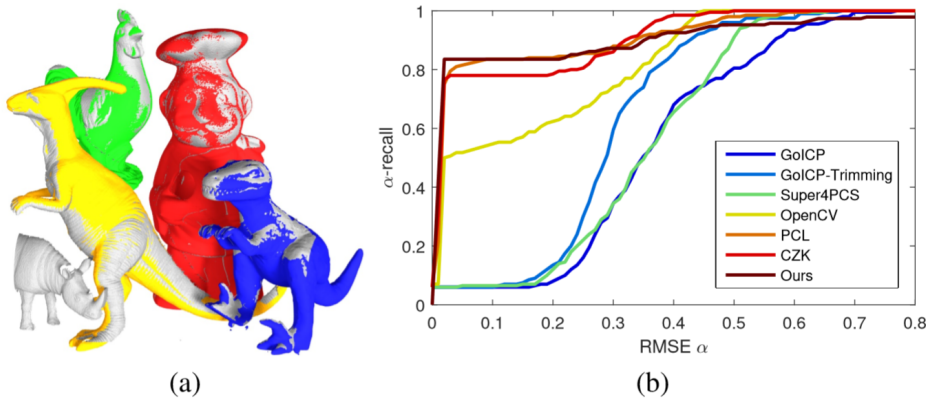


Figure 4: Global registration results on the UWA benchmark. (a) FGR on one of the 188 tests.The scene is colored white and objects aligned to the scene have distinct colors. (b) α-recall plot comparing our method and prior global registration algorithms. (Higher is better.)

In the original paper, timings are also reported. Since timings depend also on implementation, they will not be reproduced here. However, the results were all faster for FGR than for other algorithms by at least an order of magnitude.

3 Scale Invariance Check

FGR is not scale invariant and does not find the scaling factor in the registration. This is shown here and some remedies are suggested.

3.1 FPFH and Normal Flips

In general, it is hard to find the orientation of estimated normals. Note that FPFH necessitates n_P, the set of estimated normals of P. For $p_i \in P$, the normal n_{p_i} is computed by applying PCA onto $nbh_6(p_i)$ and then by selecting the normal as the smallest principal component. However, if v is an eigenvector, then so is $-v$. The normals estimated from PCA are always ambiguous.

It is possible to get oriented normals from a mesh, which is what was done in the Section "Development History". It is also possible to orient the normals given a viewpoint of the 3D point cloud, but in the case of this data no natural viewpoint comes to mind. If $n_i \to -n_i$, then the features in Eq. 1 become

$$\alpha \to -\alpha, \phi \to -\phi, \theta \to -\theta.$$

It is clear that FPFH is not normal flip invariant, but it is much less clear how to solve this without a loss in descriptiveness. The problem is not tackled in this report.

3.2 FPFH and Scale Invariance

Let $\lambda P = \{(\lambda x_i, \lambda y_i, \lambda z_i) | (x_i, y_i, z_i) = p_i \in P, \lambda > 0\}$, the scaled version of P. First note that the normals of n_P and $n_{\lambda P}$ are the same. On the other hand $\lambda p_j - \lambda p_i = \lambda(p_j - p_i), p_j, p_i \in P$. The features $\alpha_\lambda, \psi_\lambda, \theta_\lambda$ for λP, become

$$\begin{aligned}
\alpha_\lambda &= ((\lambda p_j - \lambda p_i) \times u) \cdot n_j = \lambda((p_j - p_i) \times u) \cdot n_j = \lambda\alpha, \\
\phi_\lambda &= (u \cdot (\lambda p_j - \lambda p_i))/\|\lambda p_j - \lambda p_i\| = (u \cdot (p_j - p_i))/\|p_j - p_i\| = \phi, \\
\theta_\lambda &= \arctan(\lambda w \cdot n_j, u \cdot n_j) = \arctan\left(\frac{\lambda w \cdot n_j}{u \cdot n_j}\right).
\end{aligned} \tag{8}$$

The α features scale linearly with λ, the ϕ features are invariant with respect to scale. The θ features do not have a closed form with respect to $\lambda > 0$. Let's redefine v as follows:

$$v = \frac{p_j - p_i}{\|p_j - p_i\|} \times u,$$

instead of $v = (p_j - p_i) \times u$. This will leave u, v, and w invariant under positive scaling. Therefore, both the α and the θ features become scale invariant. This seemingly innocent change still produces descriptive histograms. Note that the bin edges of FPFH need to be adjusted. This modification allows to compare point clouds at different scales while using FPFH. Call this new process *invFPFH*.

3.3 Matching Filters

In order to build a set of putative matches, the authors of FGR propose two nearest neighbor searches followed by the Reciprocity Test and the Tuple Test. The nearest neighbor search and the Reciprocity Test both operate on the FPFH descriptors, thus they can also operate on the invFPFH descriptors. By switching from the FPFH to the invFPFH, the first two tests become scale invariant.

However, the Tuple Test is clearly not scale invariant since

$$\frac{\|p_i - p_j\|}{\|q_i - q_j\|} \neq \frac{\|\lambda p_i - \lambda q_j\|}{\|q_i - q_j\|}, \quad \forall \lambda \neq 0.$$

A tuple that passes the test on one scale might fail on another scale. This filter cannot remain in a scale invariant version of FGR.

Note the similarity between the Tuple Test and Lowe's Ratio Test [17]. Generally speaking, Lowe's Ratio serves to filter putative matches. Let $FPFH(p_i)$ be the set of features of point p_i, and let $FPFH(p_{j_1})$ and $FPFH(p_{j_2})$ be the first two features closest to $FPFH(p_i)$ in feature space. Then Lowe's Ratio ρ is

$$\rho = \frac{\|FPFH(p_i) - FPFH(p_{j_1})\|}{\|FPFH(p_i) - FPFH(p_{j_2})\|}. \tag{9}$$

If ρ is smaller than a certain threshold, typically 0.8, then the correspondence (p_i, p_{j_1}) is deemed of good quality. As ρ operates on descriptors, having invariant feature descriptors ensure that Lowe's Ratio Test is also invariant. This is used in the current Matlab implementation.

There are other ways to match points based on their descriptors. The Earth Mover's Distance (EMD) works well with histograms and can be easily computed through the Hungarian algorithm for bipartite graph matching [18]. This approach seems promising, although it was not explored in the current report.

3.4 Optimization Loop

The optimization loop in its current form cannot find the relative scaling factor in the final transformation T. However, the approach pioneered in the FGR paper seems strong enough to accommodate for that modification. Further research in that direction is necessary.

4 Development History

4.1 Project Initialization

First I did a literature review. Second, I wanted to use the Point Cloud Library and C++, but I ran into two nasty bugs.

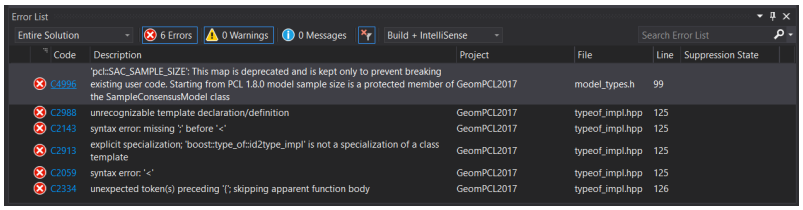


Figure 5: This was eventually solved by commenting out 3 lines in typeof-impl.hpp.

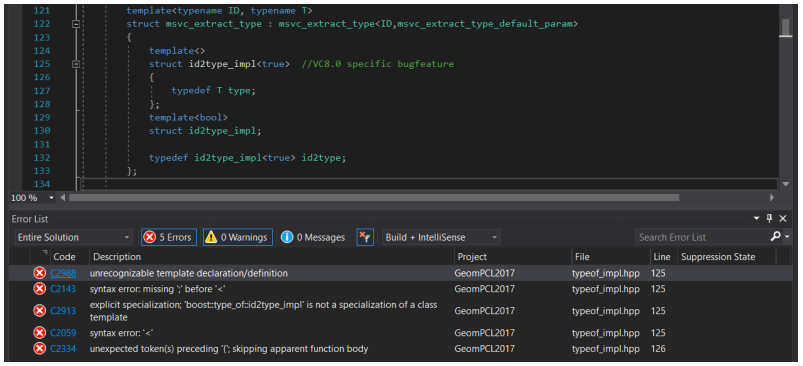


Figure 6: This was an error in the Boost library.

Therefore, I turned to Matlab and its easy visualization tools, the plan being to implement FPFH and then FGR.

4.2 Data Preprocessing

4.2.1 Remeshing and Downsampling

To isolate the behavior of FGR, the 3D mesh of the chair and the point cloud data from the scan were normalized. The point cloud was downsampled with

```
ptCloud = pcdownsample(ptCloud,'gridAverage',0.5);
```

which uses a block filter [19].

The mesh was first upsampled with a wavelet remeshing strategy [20], giving:

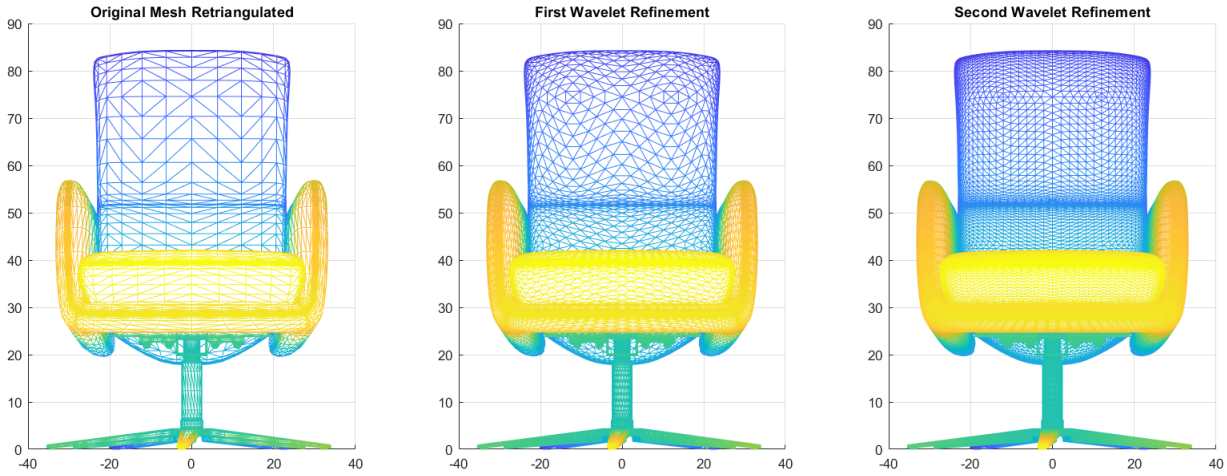


Figure 7: Successive remeshing iterations using the wavelet remeshing strategy. The precise steps are in "test-precomp.m".

and then downsampled in the same way as the point cloud, using.

```
ptCloudMesh = ptCloud(vertices);
ptCloudMesh = pcdownsample(ptCloudMesh,'gridAverage',0.5);
```

4.2.2 Normal Extraction

The FPFH descriptors need surface normals as input. As described in Section 3.1, the normals extracted with PCA were of poor quality. To solve this, the provided point cloud was first meshed using MeshLab [21]. The mesh was then used to extract nice normals.

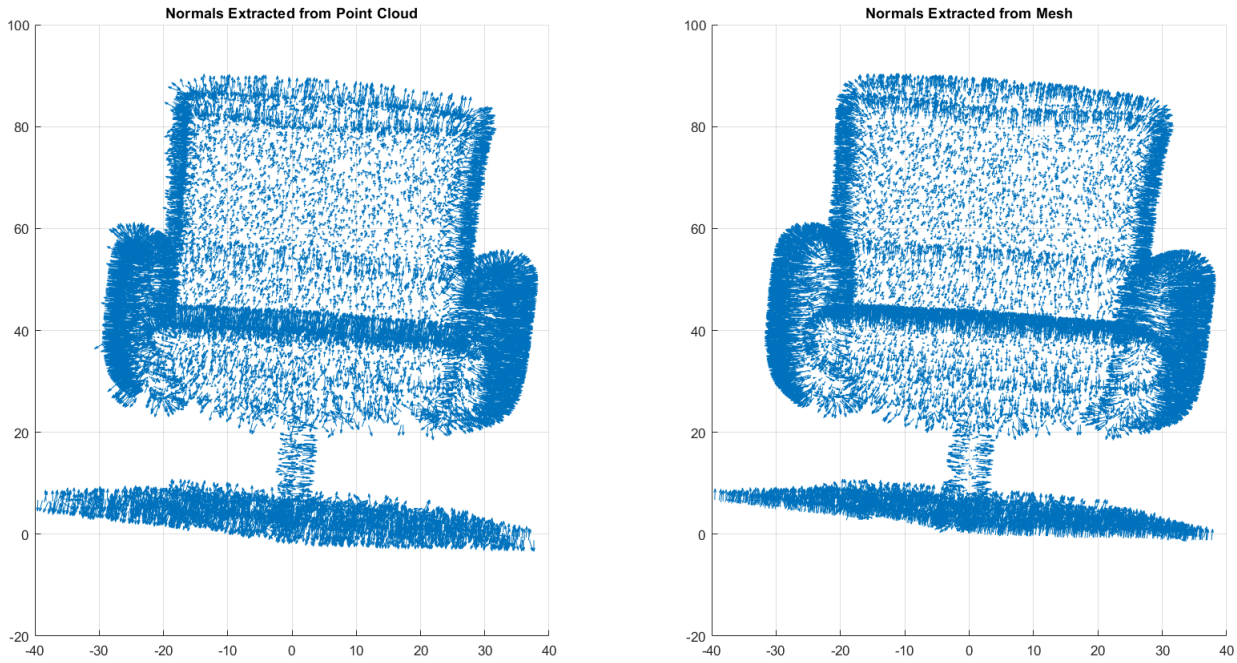


Figure 8: Note that the normals on the arm on the left are badly oriented in the first figure, but not in the second figure.

4.3 FPFH and FGR

I have a lot of experience with histograms from my project on Shape Contexts on Github [22]. Thus the implementation of FPFH was just a matter of understanding the paper. The implementation of invFPFH was more a matter of theoretical work than technical difficulty.

The implementation of FGR was done by converting the C++ code of the authors from their Github page [23]. At first I made some minor conceptual mistakes, but then it went quite smoothly.

5 Conclusion

The FGR method can be split into two parts: the 3D point feature extraction and point matching; the optimization loop and solving for T. The main contribution of FGR is in the optimization loop. Let's posit that the results from the original paper are correct.

Needed Improvements

- Unclear how it behaves with changes in sampling density.
- Not yet fully scale invariant.
- Does not find the scaling factor for T.
- Better matching routines are needed.

Current Advantages

- Faster than other methods.
- Noise resistant.
- Initial pose resistant.
- Can handle different point clouds at once, see paper.

The FGR framework is powerful enough to find the scaling factor in the optimization loop. Indeed, it relies on a linearization of the transformation elements in $SE(3)$, but a scaling factor could easily be added.

The point matching is where there is most room for improvement. In particular, a scale invariant and flip invariant descriptor is needed. The invFPFH takes care of scale invariance but more testing is necessary. Moreover, many papers explore the ideas of a spectral approach in order to find "persistent" points. This could be especially useful for point clouds with big holes or heterogeneous sampling. The approach was not implemented in the current project.

Figure 9: A spectral analysis for FPFH, from [16]. Could be applied to invFPFH as well. Reducing the set of point prior matching improves performance and accuracy.

Finally, the dual problem will be to find which models to align. In other words, some sort of global shape signature would have to be computed for each 3D model and then matched with the global signature of the point cloud.

References

[1] Natasha Gelfand, Niloy J Mitra, Leonidas J Guibas, and Helmut Pottmann. Robust global registration. In *Symposium on geometry processing*, volume 2, page 5, 2005.

[2] Sébastien Granger and Xavier Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. In *European Conference on Computer Vision*, pages 418–432. Springer, 2002.

[3] Peter Meer, Doron Mintz, Azriel Rosenfeld, and Dong Yoon Kim. Robust regression methods for computer vision: A review. *International journal of computer vision*, 6(1):59–70, 1991.

[4] Baowei Lin, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda, and Koji Ichii. Scale ratio icp for 3d point clouds with different scales. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 2217–2221. IEEE, 2013.

[5] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016.

[6] Dmitry Chetverikov, Dmitry Stepanov, and Pavel Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and Vision Computing*, 23(3):299–309, 2005.

[7] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33, pages 205–215. Wiley Online Library, 2014.

[8] Siheng Chen, Dong Tian, Chen Feng, Anthony Vetro, and Jelena Kovačević. Fast resampling of 3d point clouds via graphs. *arXiv preprint arXiv:1702.06397*, 2017.

[9] Anuj Sehgal, Daniel Cernea, and Milena Makaveeva. Real-time scale invariant 3d range point cloud registration. In *International Conference Image Analysis and Recognition*, pages 220–229. Springer, 2010.

[10] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[11] Lin Li, Fan Yang, Haihong Zhu, Dalin Li, You Li, and Lei Tang. An improved ransac for 3d point cloud plane segmentation based on normal distribution transformation cells. *Remote Sensing*, 9(5):433, 2017.

[12] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[13] David W Eggert, Adele Lorusso, and Robert B Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine vision and applications*, 9(5-6):272–290, 1997.

[14] Overview and comparison of features Â· pointcloudlibrary/pcl wiki Â· github. `https://github.com/PointCloudLibrary/pcl/wiki/Overview-and-Comparison-of-Features`. (Accessed on 02/06/2018).

[15] Ronny Hänsch, Thomas Weber, and Olaf Hellwich. Comparison of 3d interest point detectors and descriptors for point cloud fusion. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):57, 2014.

[16] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[17] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[18] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Advances in neural information processing systems*, pages 831–837, 2001.

[19] Downsample a 3-d point cloud - matlab pcdownsample. `https://www.mathworks.com/help/vision/ref/pcdownsample.html`. (Accessed on 02/06/2018).

[20] Toolbox wavelets on meshes - file exchange - matlab central. `https://www.mathworks.com/matlabcentral/fileexchange/17577-toolbox-wavelets-on-meshes`. (Accessed on 02/06/2018).

[21] Meshlab stuff: Meshing point clouds. `http://meshlabstuff.blogspot.ca/2009/09/meshing-point-clouds.html`. (Accessed on 02/06/2018).

[22] Github - szat/shapecontexts: Shape contexts for shape matching and point correspondence. `https://github.com/szat/ShapeContexts`. (Accessed on 02/06/2018).

[23] Github - intelvcl/fastglobalregistration: Fast global registration. `https://github.com/IntelVCL/FastGlobalRegistration`. (Accessed on 02/06/2018).