Point Feature Histogram

Yi-Cheng Liu, liuyiche@umich.edu; Tien-Li Lin, tienli@umich.edu;

***Abstract*—This report implemented Point Feature Histograms (PFH) as an advanced method for point cloud registration and alignment. PFH overcomes the limitations of traditional techniques like Principal Component Analysis (PCA) and Iterative Closest Point (ICP). By considering both point positions and local characteristics which were represented by histogram, PFH offers robustness against noise and outliers. We show our implementation workflow that integrates PFH with ICP, improving alignment accuracy. Our approach involves normal vector estimation, feature descriptor extraction, and PFH calculation for each point. To optimize efficiency, we selectively sample a subset of points for registration. The GitHub code is at https://github.com/yi-cheng-liu/point_feature_histogram.**

I. Introduction

In the realm of computer vision, image data forms a substantial component. In recent years, there has been significant growth in 3D point cloud processing, whether the data is acquired from LiDAR systems or derived from imagery. The application of point clouds in object recognition and robotics perception represents a huge advancement.

Some essential tasks in point cloud data processing include segmentation tasks, feature extraction, and object detection through point clouds. A critical challenge in point cloud processing is aligning or registering multiple point clouds into a consistent coordinate system. Techniques such as Principal Component Analysis (PCA) can help estimate initial transformations, while more advanced methods like Iterative Closest Point (ICP) and Point Feature Histograms (PFH) optimize the alignment. These alignment methods are quite important for data fusion and they enable precise 3D understanding of complex scenes.

II. Related Work

A. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) has emerged as a fundamental technique. It is important to simplify the complexity of point cloud data. PCA works by finding the primary axes of variance in the data, which helps in reducing dimensions while retaining essential features. This reduction is crucial in processing large datasets, as it enables more efficient analysis and interpretation. However, PCA assumes linearity in data, which limits its use cases in datasets with nonlinear dynamics. Additionally, PCA is sensitive to noise and outliers, potentially skewing results.

B. Iterative Closest Point (ICP)

The Iterative Closest Point (ICP) method has emerged as a powerful technique for aligning point clouds. ICP is particularly useful when dealing with point clouds from different viewpoints or time steps. It iteratively calculates the transformation matrix between two sets of point clouds by minimizing the distance between corresponding points. While ICP is quite effective in many cases, it does have some limitations. It can be sensitive to initial guesses, making it very easy to get stuck in local minima. Moreover, it could have a large possibility of not performing well when the pose difference between point clouds is large. For instance, when the rotation is significantly different, it could find the wrong closest point and get stuck as shown in Fig. 1

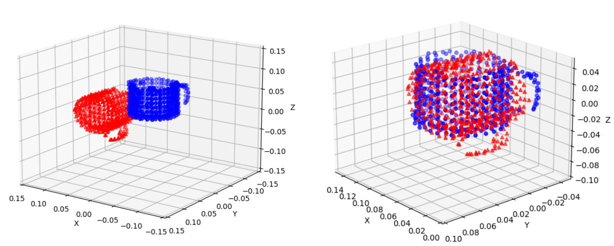


Fig. 1: ICP result. ICP causes the alignment to find the wrong point when finding the closest point.

C. Point Feature Histogram

To address these challenges in PCA and ICP, Point Feature Histograms (PFH) have been introduced as an advanced method for point cloud alignment and registration. PFH considers not only the positions of different sets of points but also their local characteristics, making it pretty robust to noise, and outliers. By calculating local histograms of point features, PFH can provide a more specific representation of the data. The generation of histograms is particularly beneficial when dealing with complex scenes or datasets with nonlinear dynamics. This approach overcomes the limitations of both PCA and traditional ICP. PCA doesn't include the local characteristics of the point clouds, making it sensitive to nonlinear data. Meanwhile, ICP has plenty of issues when noise and outliers are tainting the data. PFH addresses these issues by incorporating local surface characteristics through histograms of point features, providing a more noise-resistance method.

III. Implementation

In our work, we use distance between historgram signatures instead of the Euclidean distance between points to align two point clouds, so our workflow can be separated into two part. Initially, we compute Point Feature Histogram (PFH) for both the source and target point clouds. The PFH provides a comprehensive representation of the local geometric features within the point cloud. Subsequently, the derived histograms are employed within the Iterative Closest Point (ICP) framework. This step involves the utilization of the histograms to compute the rotation and translation matrices required for the transformation of the point clouds. This two-step process is iteratively repeated until either the error margin falls below a threshold or the maximum number of iterations.

A. Point Feature Histogram

For aligning two point clouds, utilizing the normal vector and curvature as feature descriptors is both basic and effective. hese features are straightforward to calculate. However, numerous points possess similar or identical features, leading to insufficient information for accurate point cloud matching. To address this issue, [1] proposes a novel approach named Point Feature Histogram. Unlike basic methods, PFH not only extracts the normal vector but also considers the relative positioning between a central point and its neighboring points, thereby capturing a richer set of information within the point cloud.

In our implementation, the process begins with the calculation of the estimated normal vector for every point in the point cloud. Find the nearest 20 points surrounding a central point and then employing Principal Component Analysis (PCA) to extract the eigenvector corresponding to the smallest eigenvalue. The rationale behind this choice is that the magnitude of the eigenvalue signifies the strength of the eigenvector in representing the data. Since the normal vector is orthogonal to the surface and should exist in the null space of the surface, the eigenvector associated with the smallest eigenvalue provides the most accurate estimation for this null space.

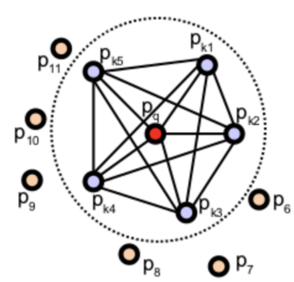


Fig. 2: Considered neighbors of the central point. Red point is central point, blue points are neighbors points, and yellow points are not considered points.

PFH is calculated from deciding which points are considered neighbors of the central point. This is based on whether the Euclidean distance between two points within a predefined radius shown as figure 2. PFH for the central point is derived by evaluating the relationships between every pair of points within this radius, effectively capturing the local geometric structure of the point cloud.

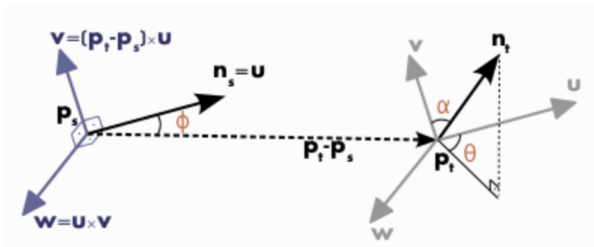


Fig. 3: Using normal vector to define feature descriptor

Fig 3 show how we get the relationship between two points. Given two point p_1, p_2 and their normal vector n_1, n_2. We can define the coordinate frame shown as following at one point.

$$\begin{cases} \mathbf{u} = \mathbf{n}_s \\ \mathbf{v} = \mathbf{u} \times \frac{\mathbf{p}_t - \mathbf{p}_s}{\|\mathbf{p}_t - \mathbf{p}_s\|_2} \\ \mathbf{w} = \mathbf{u} \times \mathbf{v} \end{cases} \tag{1}$$

Based on this u, v, w frame, the relationship between normal vectors n_1, n_2 can be expressed as following eqution:

$$\begin{cases} \alpha = \mathbf{v} \cdot \mathbf{n}_t \\ \phi = \mathbf{u} \cdot \frac{\mathbf{p}_t - \mathbf{p}_s}{d} \\ \theta = \arctan(\mathbf{w} \cdot \mathbf{n}_t, \mathbf{u} \cdot \mathbf{n}_t) \end{cases} \tag{2}$$

Now we have the descriptor α, ϕ, θ for each point pair, the next step is turn α, ϕ, θ into bin histogram. The subsequent step involves converting these values into a bin histogram. This process is accomplished through a methodical approach where each feature is first normalized and then multiplied by a predetermined number of bins. The result of this normalization and multiplication is a series of indices corresponding to the bins in the histogram. Finally, for each point pair, the histogram's values are incremented according to these bin indices, ensuring that each bin accurately reflects the frequency of its corresponding angular range in the point cloud.

It is time consuming to calculate PFH for all points in every iteration. To optimize the process, we selectively sample a subset of points from the source point cloud. Specifically, we randomly choose 400 points and use this subset as the input for the Iterative Closest Point (ICP) algorithm. This method reduces computational cost while maintaining the integrity and effectiveness of the point cloud alignment process.

B. *Iterative Closest Point*

After deriving PFH for all points, the next step is using ICP to align two point cloud. The approach diverges from the traditional ICP method, which employs Euclidean distance in 3D space. Instead, for each selected source point, we identify the most similar counterpart in the target point cloud by calculating the Euclidean distance between their respective PFHs. This strategy enables us to pinpoint points in the target cloud that possess the most similar features in terms of their PFH representation.

After getting correspondence sequence of points, we employ Singular Value Decomposition (SVD) to determine the rotation and translation matrices that align the two point clouds. This alignment is achieved by minimizing the distance between these corresponding point pairs.

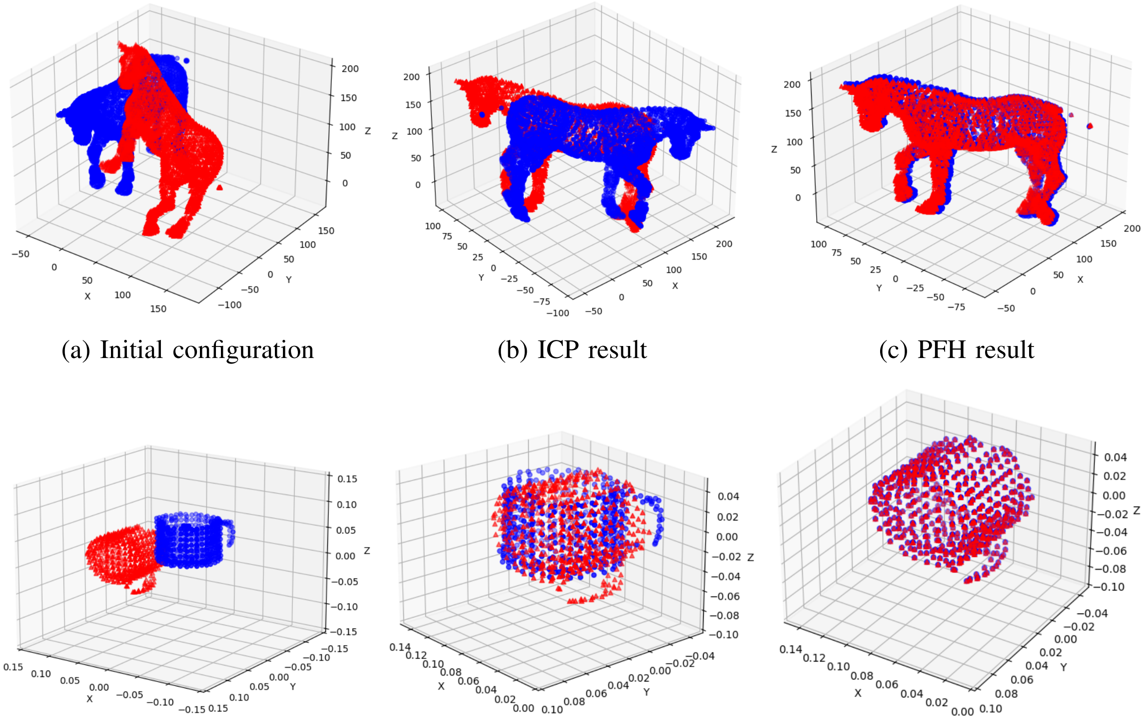
This process is iteratively repeated, continuously refining the alignment. The iteration ceases either when the error margin falls below a predetermined threshold, or when the number of iterations reaches a set limit.

IV. Results

We now present the experimental result, the data sources are collected from the course or open-sourced data from the point cloud library, more data can be found in [2]. The main result would be presented with the mug from the course data and the horse from the PCL data.

A. *Quantitative Reuslt*

Starting with the Mug data, the ICP method completed its process significantly faster, taking approximately 1.42 seconds, compared to the PFH method, which took about 29.28 seconds. The reason for this is that the Mug has a comparatively small number of point clouds, around 400 points. Therefore, when calculating the distance between points rather than histograms, the traditional ICP process speed could be much faster. However, the final error of the ICP method, at 25.72, was a lot higher than the PFH method which is close to zero. Results are shown in Table I.



(a) Initial configuration (b) ICP result (c) PFH result

(d) Initial configuration (e) ICP result (f) PFH result

Fig. 4: Results of two sets of data. Top row: Horse (3400 points), Bottom row: Mug (400 points).

For the Horse data, since there are 3400 points in the data, the time difference between ICP and PFH methods was less pronounced, with ICP taking about 14.67 seconds and PFH around 32 seconds. The final error for the Horse data using ICP was extremely high, at 507,081.14, while PFH achieved a significantly lower error of 36,532.42. Although PFH took more than double the time of ICP, its error was substantially lower, suggesting a better quality of alignment. Results are shown in Table II.

Metric	**ICP**	**PFH**
Time (seconds)	1.4196	29.2803
Final Error	25.7236	3.7882×10^{-13}

TABLE I: ICP and PFH Comparison in Computation Time and Error in Mug Data

In the Point Feature Histogram (PFH) method, Table III illustrates a clear trend: as the number of bins increases, the error in point cloud matching decreases. With just 8 bins, the error is highest, indicating that a coarser histogram lacks detail. As the bin number grows to 27, 64, and finally 125, the error progressively reduces, signifying that finer histograms capture more detailed features, and improve accuracy. Moreover, the increased precision with higher bins doesn't increase cost of computational time. We assume that a larger bin number equates to higher resolution for the algorithm, allowing it to extract more detail and information from the descriptor. This enhanced detail extraction is likely the reason for the algorithm's more rapid convergence.

Metric	**ICP**	**PFH**
Time (seconds)	14.6733	32.0034
Final Error	507081.1376	36532.4223

TABLE II: ICP and PFH Comparison in Computation Time and Error Horse Data

TABLE III: Error Metrics for Different Bin Numbers

Bin Number	Error after first iteration
8	148440.0804
27	60955.6991
64	41567.6784
125	18931.5701

B. Qualitative Result

As demonstrated in Fig. 4, the PFH method maintains high accuracy and reliability regardless of the point cloud size, whether it is as few as 400 points such as the mug in 4d or as many as 3400 points such as the horse in 4a. This consistency extends to situations with minimal or substantial variance in initial guess positioning. Even in cases of large initial rotational misalignments and large distances, the PFH method effectively identifies the closest histogram match. The results show that PFH still consistently computes the appropriate rotational adjustments, ensuring precise alignment of the point clouds, compared with the traditional ICP method.

V. Conclusion

In conclusion, our implementation in the approach to point cloud alignment by integrating the Point Feature Histogram (PFH) with the Iterative Closest Point (ICP) method. This technique shows results in error that it outperforms traditional ICP methods in accuracy, particularly with complex and larger point clouds, by focusing on feature-based histogram similarity rather than direct Euclidean distance. The quantitative results validate its effectiveness in achieving precise alignments.

References

[1] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Learning informative point classes for the acquisition of object model maps," in *2008 10th International Conference on Control, Automation, Robotics and Vision*, 2008, pp. 643–650.

[2] PCL-Github. Point cloud library data. [Online]. Available: https://github.com/PointCloudLibrary/data/tree/master

[3] PCL. Estimating surface normals in a pointcloud. [Online]. Available: https://pcl.readthedocs.io/projects/tutorials/en/latest/normal_estimation.html#normal-estimation

[4] ——. Point feature histograms (pfh) descriptors. [Online]. Available: https://pcl.readthedocs.io/projects/tutorials/en/latest/pfh_estimation.html