Computer Science - Senior Capstone Project
Spring 2019
May 12, 2019

3D Point Cloud Registration Algorithms for the NYUSH IMA Telewindow Project

Final Report

Zhanghao Chen, Xincheng Huang

Supervised by: *Michael Naimark*
Mentor: *Olivier Marin*

Abstract

The goal of this project is to develop a set of algorithms that geometrically align, or “register”, the 3D point clouds captured by the depth cameras of “Telewindow” into one 3D image. Current registration algorithms usually adopt a pipeline architecture with multiple stages. Plenty of variants are available for each stage and there is not a standardized solution suited for all applications. We build an experimental registration pipeline and investigate different combinations of the algorithms from each stage to find the optimal pipeline under the “Telewindow” project setting. We identify the most potential algorithm combinations as the result of our experiments, which is able to provide an accurate registration result given clean point clouds of a user’s figure.

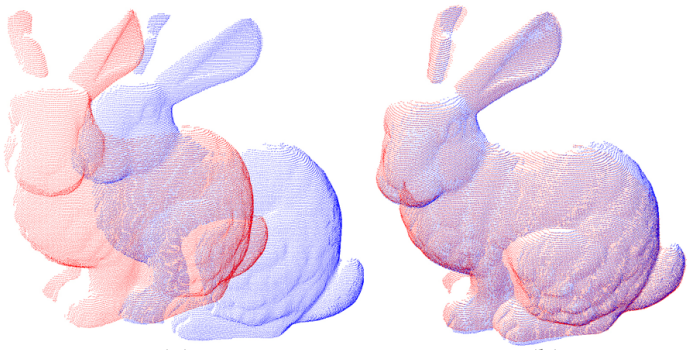
Contents

1 Introduction

This project is one of the tracks of the "Telewindow" project by Interactive Media Arts at New York University Shanghai. The purpose of "Telewindow" is to enhance teleconferencing experience by streaming 3D images constructed with 3D point clouds captured by a set of depth cameras and can be divided into four tasks, namely 3D point registration, merging, rendering, and 3D data streaming. Therefore, the target of our project is to find the optimal solution for the first task, 3D point cloud registration.

To be more specific, 3D point cloud registration deals with two or more point clouds of a certain object or environment and tries to obtain the translation and rotation matrices that can map the clouds into a common coordinate system such that the object or environment can be rebuilt in a three-dimensional space. An example of 3D registration is given in Fig. 1. In general, the 3D point cloud registration problem can be classified into two categories, non-rigid and rigid. The former deals with scenes that involve non-rigid bodies and motions, such as water and animals. And the latter is only concerned with rigid bodies whose deformation can be neglected. In our project, although we are dealing with scenes that involve human, the point clouds we captured are always from same time frames, so there is no non-rigid motion or body deformation. Thus, we only consider rigid point cloud registrations here.

A lot of researches have been done in this area, and most of them take the overall 3D registration problem as a least-square minimization problem, which aligns the point clouds by minimizing the distance between corresponding points [1]. This is done with two stages: coarse alignment and local refinement. In the former process, an initial estimation of the cloud alignment is computed. And in the latter, this initial estimation is refined iteratively until a certain criterion is reached [2]. Each stage contains multiple steps, and we will give a review of each of them below.



(a) (b)

Fig. 1: An example of 3D point cloud registration [3].

2 Related Work

As mentioned, 3D point cloud registration generally contains two stages, namely the coarse alignment process and the local refinement process. In the area of 3D registration, the former process has much more variants and different implementations, whereas the latter process is mostly based on the classic method as it was introduced in [4]. Therefore, despite that the coarse alignment process precedes the local refinement process in the registration pipeline, we are introducing local refinement first as it is fundamental for most registration algorithms. Meanwhile, in terms of metrics, 3D registration methods are generally assessed from three aspects, that is accuracy, robustness, and efficiency. Therefore, in the review below, we will assess most of the methods based on such metrics.

2.1 Local Refinement Methods

2.1.1 Local Registration Assuming Perfect Data

As mentioned, rigid registration can be approached as a least-square minimization problem, by defining a cost function that represents the current matching error between the point clouds [1]. In theory, this is a linear least-square minimization problem and can be directly solved with Singular Value Decomposition (SVD) on the premise that noise-free point cloud data and perfect point correspondence are available [5]. Nevertheless, neither of the two premises is reachable in the real world, and this is why an iterative method comes in.

2.1.2 Iterative Closest Point

Today, most of the state-of-the-art implementations of registration methods employ an algorithm (or its variants) called Iterative Closest Point Algorithm (ICP), introduced by Besl and McKay in 1992 [1] [4]. Given an initial estimation of point correspondence, the ICP algorithm iteratively refines the transformation by running SVD for multiple times and removing outliers and redefining point correspondence along the way [1]. At each iteration, the algorithm estimates new point correspondences based on the transformation computed, and reject the correspondence pairs that are considered outliers. There are plenty of methods in deciding whether a pair of correspondent points are outliers, and in practice, multiple of such correspondence rejection methods are used together to determine whether a point correspondence pair should be kept or rejected [6]. The step of correspondence filtering and rejection is also essential for the process of coarse alignment, and thus we will give it more attention in the next section. Since the introduction of ICP, many

variants have been proposed, and each has their advantages under different circumstances, and here we present them as below.

2.1.3 ICP point-to-point and ICP point-to-surface

Two widely used variants of ICP are ICP point-to-point [1] and ICP point-to-surface algorithms [7]. The main difference between them is the way they define point correspondences. For instance, consider the case of pairwise point cloud registration, the former, ICP point-to-point, simply search for point correspondences between the source and target point clouds by finding the nearest neighbor based on the Euclidean distance. And the latter consider a group of points around the candidate correspondent point in the target cloud and find the point correspondence by aligning the point on the source point cloud to the surface normal **n** of that group of points [1]. Fig. 2 and Fig. 3 give demonstrations for ICP point-to-point and ICP point-to-surface.

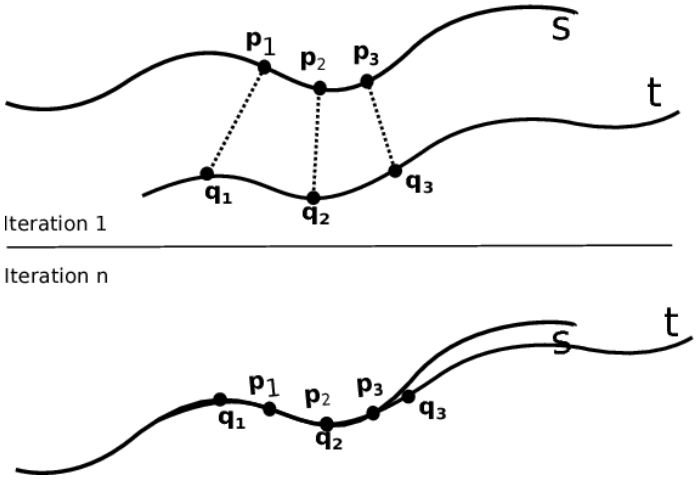


Fig. 2: A demonstration of ICP point-to-point [1].

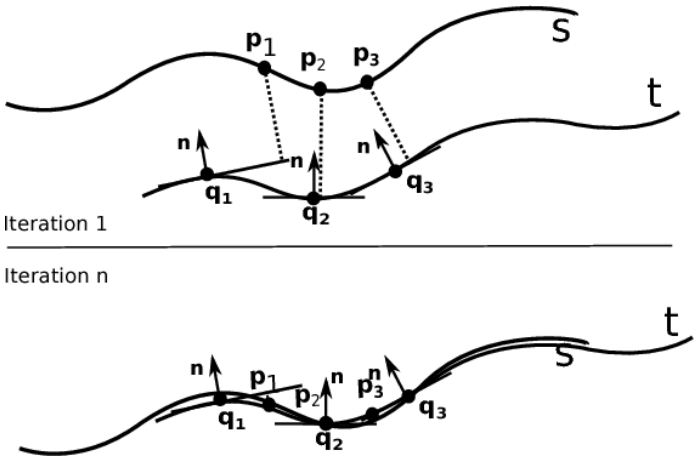


Fig. 3: A demonstration of ICP point-to-surface [1].

The transformation **T** that both the ICP point-to-point and ICP point-to-surface define is composed of a rotation **R** and a translation **t**. And the error metrics to be minimized in the two methods are listed respectively below as (1) and (2), where (p_k, q_k) is the kth of the N correspondences from the source point cloud to the target point cloud [6]:

$$E_{point-to-point}(\mathbf{T}) = \sum_{k=1}^{N} \|\mathbf{T}\mathbf{p}_k - \mathbf{q}_k\|_2^2 \tag{1}$$

$$E_{point-to-surface}(\mathbf{T}) = \sum_{k=1}^{N} (\mathbf{T}\mathbf{p}_k - \mathbf{q}_k) \cdot \mathbf{n}_{\mathbf{q}_k})^2 \tag{2}$$

These two algorithms have their advantages in different scenarios. In general, as the ICP point-to-surface algorithm finds the correspondence points in a tangent plane formed by a group of candidate correspondent key points, it is more robust against noises. To be more specific, the point-to-surface algorithm outperforms the point-to-point algorithm when the source point cloud does not contain much noise while the target point cloud is mostly smooth. On the other hand, the point-to-point algorithm performs better when the structures are mostly quadratic or polynomial [1].

2.1.4 Colored Point Cloud Registration

In the modern context of 3D point cloud registration, the scenes that we are dealing with are usually colored. Therefore, based on the traditional point cloud registration method such as ICP point-to-point and ICP point-to-surface, there have been efforts that integrate color in these registration algorithms. One common approach is to consider the color as additional parameters, which extends the original three-dimensional definition of the points to a four- (if the colors are gray-scaled) or six-dimensional one(if the colors are in RGB) [8] [9].

Apart from this common approach, J.Park et al. introduced a method [10] that, instead of parameterizing the color information with the three-dimensional point coordinates, it approaches them separately. In particular, this method defines the colored information as “photometric data”, and the spatial information as “geometric data”, and develops an optimization objective for each of them. Then, the method defines a joint optimization objective that integrates both the geometric term and the photometric term. J.Park et al. claim that this method is more efficient and accurate than both the algorithms that do not take color into account and the common colored registration algorithms that take color as mere additional parameters [10]. Table 2 below, summarized from [10], is a comparison between the performance of J.Park et al.’s Colored

Point Cloud Registration, other common colored registration, and non-colored registration.

Methods	Accuracy	Robustness	Efficiency
ICP point-to-point	Low	Low	High
ICP point-to-Surface	Medium	Medium	Medium
Color 4D ICP	Medium	Low	Medium
Color 6D ICP	Medium	Low	Medium
J.Park et al's Colored ICP	High	Medium	High

TABLE 1: Performance comparison between ICP algorithms.

2.1.5 Beyond Local Refinement

The accuracy and efficiency of the local refinement algorithms as described above rely heavily on the quality of pre-alignment [1]. Without a good pre-alignment, the local refinement algorithms take a longer time to converge and it is easy for them to end up in a local minimum of the cost function and thus yield bad results. Therefore, modern pre-alignment, or "Coarse alignment", provides a pipeline to enhance robustness and efficiency of the ICP algorithms, and we are giving it a review below.

2.2 Coarse Alignment via Feature Matching

Due to the non-convexity of the problem and the local iterative procedure it adopts, ICP requires a reliable initial alignment to avoid the problem of local minima. Typical solutions to this problem adopt a feature-based coarse initial alignment. A set of keypoints are firstly extracted from the input clouds. For each point, a compact vector representation is computed, which is referred to as a feature. Point correspondences are estimated via feature matching, followed by subsequent filtering of outliers. Finally, a coarse transformation is estimated based on the correspondences, which can be further refined by ICP [6]. A number of different approaches have been proposed for each stage, which we will now review.

2.2.1 Keypoint Extraction

A keypoint is simply a point that is identified as relevant for a given task, like registration, object recognition etc. It usually has some special properties that distinguish itself from other points, for example, locating right on the corner of a surface. Keypoint extraction reduces the number of points used for computation. An ideal keypoint detector should also be repeatable with respect to rigid transformation and noise. A great number of 3D keypoint detectors have been proposed, like NARF [11], 3D-SIFT [12], ISS [13],

and FAST [14], to name only a few of the available methods. Some proposals include both a keypoint detector and a companion keypoint feature descriptor, like NARF. However, detectors and descriptors from different proposals can be used in a mixed way, which may even result in a better performance than the original pair [15]. Hansch et al. compared NARF and 3D-SIFT and reported that NARF computes faster but produces fewer keypoints and worse registration results than 3D-SIFT [16]. Besides explicit extraction of keypoints, one may alternatively only use sampling-based methods, e.g., using all available points [4], uniform downsampling [17], or random sampling [18]. While relying on specific keypoint detection schemes improves the repeatability and robustness of the registration against noise [6], there is evidence that this may also hurt the accuracy by a too strong reduction of available points, especially in the multi-view registration case which involves more than two point clouds [16].

2.2.2 Feature Description

A feature is a compact vector representation of a point's local neighborhood. Points in different clouds with a similar feature are likely to represent the same surface point. A good feature descriptor should be robust against noise, invariant against rigid transformation, fast to compute, and fast to compare [19]. As is the case of keypoints, there are a huge number of feature descriptors available. They can be based on only geometric characteristics (e.g. PFH [20] , SHOT [21], 3DSC [22]), or if available, also incorporates photometric characteristics (e.g., PFH-RGB [23], Color-SHOT [24]). It is generally hard to separate the performance of feature descriptor from that of the overall pipeline. Nevertheless, in terms of accuracy on the registration task, SHOT is reported to outperform a number of other methods including KPQ [25], MeshHoG [26], etc., and the pair of ISS as keypoint detector and 3DSC as feature descriptor yields the best result [15]. Hansch et al. suggests that the PFH family are faster to compute and more robust against viewpoint differences than the SHOT family [16]. In particular, FPFH [27], a faster variant of PFH, computes faster while achieving similar accuracy as PFH [16].

Recently, deep learning methods have been studied for feature extraction. In the field of 2D computer vision, deep learning methods, especially convolutional neural networks (CNNs), have shown superiority over hand-coded feature descriptors in extracting more advanced features [28]. Yang et al. adopted 2D convolutional neural networks to extract features for image registration and outperforms the method using SIFT keypoint extractor and feature descriptor [29]. When it comes to 3D point cloud data, although there are several attempts applying deep learning for feature extraction [30] [31], they are originally proposed

for the task of object recognition and the utility of them for the task of 3D point cloud registration has remained mostly elusive.

Methods	Incorporate Color Information	Running Speed
SHOT	No	Slow
Color-SHOT	Yes	Very slow
PFH	No	Fast
FPFH	No	Very fast
PFHRGB	Yes	Medium
3DSC	No	Very slow

TABLE 2: A summary of selected 3D point cloud feature descriptors.

2.2.3 Correspondences Estimation

A number of approximation methods exist for finding the ideal correspondences between different clouds. Typically, it is done by nearest neighbor search in the feature space. A point in the source cloud is paired to their nearest neighbor in the target cloud. The brute-force algorithm is too computationally expensive for real applications with a massive number of points as it takes a linear search time for each point. Various data structures for rapid searches have been proposed, and k-d tree [32] is the most well-known. While very effective in low dimensional spaces, its performance degrades dramatically with increased dimensions [33]. To further improve upon search efficiency, a number of fast approximation algorithms have been proposed, and priority search k-means tree [34] and multiple randomized k-d trees [35] are among the most efficient ones. They provide significant speedups with minor loss of accuracy [34].

2.2.4 Correspondences Filtering

As mentioned, in the local refinement process, the algorithms like ICP takes a correspondence filtering step in each iteration. This is also an important step in the coarse alignment process, through which it roughly filters mismatches in the initial estimation. This step, whether it is in the coarse alignment process or in the refinement process, takes the same set of filtering methods, including filtering based on distances (exclude pairs with distance exceeding the mean distance or a arbitrary threshold), target matching (exclude duplicated target matches), normal compatibility (exclude pairs with large inconsistent surface normals), and surface boundaries (exclude pairs on surface boundaries). Meanwhile, there is a relatively more advanced randomized technique called RANSAC [36]. This method randomly choose a subset of the point and then filter them based on distance. Then, by iterations, each time a different subset of the points is

selected, and the correspondence pairs got filtered becomes different. In this way, it progressively edits the correspondence selection, and it helps to keep the overall registration from ending up in a local minimum since it filters invalid correspondent point randomly and dynamically. The methods above, when applied, are not separated from one another. In general, they are used together to obtain better filtering results.

Methods	Filtering condition
Based on Distance	whether the point pair distance exceeds a given threshold
Based on Median Distance	whether the point pair distance exceeds the distance median
Based on Duplication	filter out the duplicated correspondent points in the source cloud
Based on RANSAC	randomly choose a subset of the points and filter based on distance
Based on Normal Compatibility	whether point pairs have consistent surface normals
Based on Surface Boundaries	when two surfaces overlap, filter surface boundary points

TABLE 3: Correspondence Filtering Methods.

2.2.5 Transformation Estimation

Finally, the same type of transformation estimation used in one single iteration of the ICP methods can be done based on the estimated correspondences. This can produce a coarse alignment which can then be iteratively refined.

2.3 Discussion and Conclusion

In summary, 3D point cloud registration is a very comprehensive area containing multiple steps and choices of algorithms, and here we provide our insights of these algorithms specific to their potential application to "Telewindow".

First, for coarse registration, the algorithms are not separated from each other. Under a general pipeline, they are combined and used in a mixed way. In the meantime, the performance of them is highly dependent on the scenes captured. Considering the setting of "Telewindow", it introduces a relatively new application in terms of real-time conferencing, with almost static positioning of cameras, relatively closer object-to-camera distance, as well as the requirement of being real-time. Thus, the optimal registration pipeline of 3D point clouds for "Telewindow" can be very different from the ones used in common cases. As a result, broad combinations of the algorithms should be explored in our experiment. Therefore, our experiment with coarse registration will cover nearly all the possible combinations of the algorithms as listed above. We will not experiment with PFH, KPQ or MeshDoG for feature description, as they are shown inferior to the other methods in [16] [15].

Second, for local refinement methods, we will take a different approach. Instead of experimenting with multiple combinations of algorithms, we only need to identify the one that performs the best. The reason is that the ICP algorithms are relatively independent, the sole result it relies on is the transformation matrices computed by the coarse registration process. Among the ICP algorithms that we reviewed above, we will experiment with ICP point-to-point, ICP point-to-plane, and J.Park et al.'s Colored ICP [10], as they are more promising than the others in terms of accuracy and efficiency.

In conclusion, the goal of our project is to figure out the optimal registration pipeline for "Telewindow" by experimenting and tuning different combinations of the state-of-art methods in each of the registration steps under the unique project setting of "Telewindow". The details of the construction of such a pipeline will be presented in the next section.

3 Solution

3.1 Solution Overview

In general, we build an experimental pipeline that ensembles multiple registration stages together. At each of the stages, we implement several potential candidate algorithms as discussed in our literature review. Consequently, our experimental pipeline can provide us with several different combinations of the algorithms from each registration stage, thus facilitating our experiments.

The inputs of our pipeline are two or more point clouds of one object captured in the same time frame. The pipeline selects one of the point clouds as the target point cloud, and the rest as source point clouds. Then, for each of the source point clouds, each algorithm combination computes a transformation matrix that maps it to the target point cloud. Therefore, suppose our pipeline provides M combinations of registration algorithms and is input with N point clouds, the output of the pipeline will be $M * (N - 1)$ transformation matrices.

3.2 Pipeline Architecture

Fig. 4 below shows the architecture of this experimental pipeline that we build, which contains four major stages, including preprocessing, keypoint and feature computation, correspondence matching, and refinement using ICPs. The former three stages are implemented using the PCL library, and the refinement stage is implemented using Open3D.

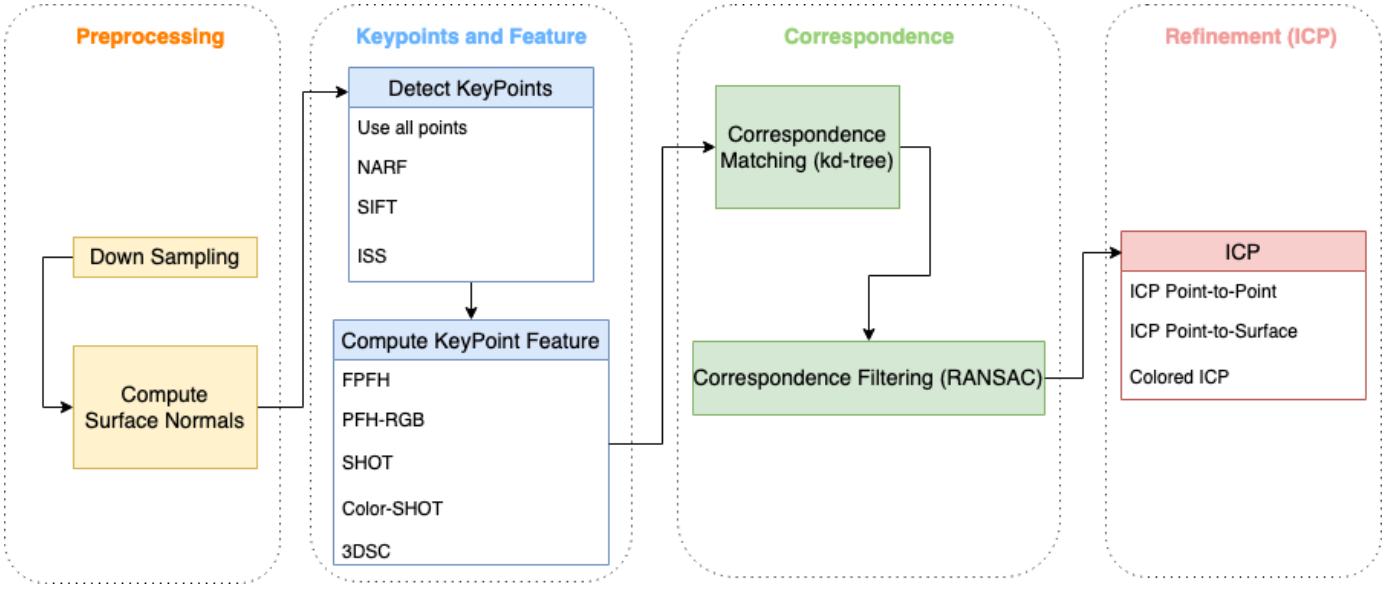


Fig. 4: The experimental pipeline.

For key point detection and feature computation, we are trying to explore as many combinations as possible, since multiple papers have reported that each of the algorithms can triumph in different specific scenarios, and it is hard to determine to which scenario Telewindow's data belong. For correspondence matching and final refinement, however, the performance depends more on the quality of coarse refinement than on the scenes itself and is thus more stable across different applications. Therefore, we select only the most common and popular ones for our experiments.

4 Experiment Settings

The experiment is conducted on three common scenes that can occur in teleconferencing to test our algorithms. In the first scene, we have the testers sitting straight. Hypothetically, this should be the most ideal case that has the highest potential to yield the best results. For this scene, we test on both male and female testers to see whether our algorithms are stable across different people with different faces and figures. After the first scene, we start to add potential noise to our data. In the second scene, we have the tester hold a bottle, and in the third scene, the tester sits sideways. The initial point clouds of these three scenes will be presented in section 5 along with the registered point clouds. We use the same pipeline configuration for all the scenes and generate the results for each algorithm combination.

4.1 Data

The testing point cloud data are captured by the four Intel® RealSense™ Depth Camera D415s on the experimental platform of the "Telewindow" project. The four cameras are installed at the top, left, right and bottom of the display screens. The cameras have several built-in filters and we enable all of them as this yields the best result. We also filter all the points whose depths are more than 1.5 m to remove the noisy background.

Fig. 5: Experimental platform of "Telewindow".

Fig. 6: Example of unregistered point clouds.

4.2 Metrics

We evaluate our registration pipeline from two perspectives, running speed and accuracy. We run the same pipeline for 10 times and take the mean running time as a measure for running speed. Due to the lack of ground truth registration results, we cannot directly compare our results with the ground truth. We use the Root Mean Squared Correspondence Error (RMSCE) and the Maximum Correspondence Error (MCE) to objectively measure the accuracy of the registration results. They represent the mean and the maximum matching error in terms of aligning the registered source clouds to the target cloud. Formally, let $\mathscr{C}$ denote a list of N 3D point clouds that we want to register. We use $\mathscr{C}_i$ to represent the i-th point cloud of the list and M_i is the number of points in $\mathscr{C}_i$. We use $\mathbf{p}_{ij}$ to denote the 3D homogeneous coordinate of the j-th point in the i-th point cloud. For each point cloud $\mathscr{C}_i$, we estimate the transformation $\mathbf{T}_i$ to align $\mathscr{C}_i$ with $\mathscr{C}_1$, the target cloud. Let $NN_{\mathscr{C}_1}(\mathbf{p})$ denote the nearest neighbor point of $\mathbf{p}$ in $\mathscr{C}_1$ computed with the Euclidean distance function. The Root Mean Squared Correspondence Error (RMSCE) and the Maximum

Correspondence Error (MCE) for $\mathscr{C}$ are computed as follows:

$$RMSCE_{\mathscr{C}} = \sqrt{\frac{1}{\sum_{i=2}^{N} M_i} \sum_{i=2}^{N} \sum_{j=1}^{M_i} \left\| \mathbf{T}_i \mathbf{p}_{ij} - NN_{\mathscr{C}_1}(\mathbf{T}_i \mathbf{p}_{ij}) \right\|_2^2} \tag{3}$$

$$MCE_{\mathscr{C}} = \max_{2 \le i \le N, 1 \le j \le M_i} \left\| \mathbf{T}_i \mathbf{p}_{ij} - NN_{\mathscr{C}_1}(\mathbf{T}_i \mathbf{p}_{ij}) \right\|_2 . \tag{4}$$

We run the same pipeline for 10 times and report the mean error score.

4.3 Parameter Tuning

We tune the parameters of the algorithms used in each stage beforehand. Downsampling is performed with a uniform 3D grid size of 1 cm. For each point in the cloud, its surface normal and feature descriptor are estimated using all neighbor points in a sphere of radius 2 cm. Correspondence estimation is carried out in a reciprocal manner (searches for correspondences from cloud A to cloud B as well as from B to A and only use the intersection). The RANSAC correspondence filter takes a maximum iteration of 1000 and a maximum correspondence points-pair distance of 3 cm. For more details on the configuration of the keypoint detectors, feature descriptors, and ICP refinement algorithms, see Appendix A, B and C.

5 Results

In this section, we provide a detailed description of our experimental results for each scene that we mentioned in section 4. For each scene, we report the coarse registration results in tables. The top 3 algorithms in terms of lowest RMSCE (shaded in light grey) are picked for testing different ICP algorithms upon the coarse alignment transformations computed by them. We then report the ICP registration results in tables and visually present the initial point clouds and the registered point clouds of lowest RMSCE. The tables of registration results are sorted in the ascending order according to firstly RMSCE, then time and lastly MCE. The lowest values in each column of the table are marked with red.

5.1 Sitting Up Straight

We test on two users, one male, and one female, sitting up straight towards the cameras.

5.1.1 Test User 1 (Male)

In Table 4, we list the coarse registration results for test user 1 sitting up straight. It is clear from the table that for keypoint detectors, Use All Points outperforms SIFT, NARF, and ISS in terms of registration

accuracy at the cost of slower running speed. For feature descriptors, SHOT, FPFH, and PFH-RGB are among the best three in terms of registration accuracy, reaching an RMSCE of about 2.5 cm. SHOT performs slightly better than the other two in accuracy but is significantly slower than them, especially FPFH.

In Table 5, we list the ICP registration results. All the algorithm combinations perform similarly in terms of registration accuracy, reaching an RMSCE of about 1.5 cm, but color-ICP is slightly better. Point-to-plane ICP is slightly faster than color-ICP and point-to-point ICP, but overall, the time for ICP refinement is negligible compared to the time for coarse alignment.

Algorithms	Time (s)	RMSCE (m)	MCE (m)
All Points+SHOT	42.3949	0.0240	0.2858
All Points+FPFH	8.5297	0.0244	0.3065
All Points+PFHRGB	11.045	0.0272	0.3405
SIFT+FPFH	6.3637	0.0359	0.2974
SIFT+PFHRGB	6.2544	0.0471	0.385
ISS+PFHRGB	4.4588	0.0561	0.2993
All Points+3DSC	262.2276	0.0577	0.418
All Points+SHOTColor	227.2644	0.0589	0.3249
SIFT+SHOTColor	8.9224	0.0697	0.4362
ISS+SHOTColor	4.4091	0.0703	0.3144
ISS+SHOT	4.3602	0.0707	0.3641
ISS+FPFH	4.4822	0.0759	0.3879
SIFT+SHOT	6.8108	0.0875	0.3291
SIFT+3DSC	10.0681	0.1031	0.3843
NARF+SHOTColor	3.4788	0.1239	0.4203
NARF+SHOT	3.4844	0.1239	0.4203
NARF+PFHRGB	3.4891	0.1239	0.4203
NARF+FPFH	3.4934	0.1239	0.4203
NARF+3DSC	3.5209	0.1239	0.4203
ISS+3DSC	4.3523	0.1239	0.4203

TABLE 4: Coarse registration results for test user 1 sitting up straight.

Algorithms	ICP time	Total time	ICP RMSCE	ICP MCE
All Points+FPFH+color-ICP	0.3838952	8.7973752	0.0150694	0.277851
All Points+PFHRGB+color-ICP	0.3855574	11.1381574	0.0150694	0.277851
All Points+SHOT+color-ICP	0.3838651	42.2366651	0.0150694	0.277851
All Points+FPFH+point-to-point	0.3539190	8.6842290	0.0150694	0.277851
All Points+PFHRGB+point-to-point	0.3520682	11.0670682	0.0151460	0.283751
All Points+FPFH+point-to-plane	0.1733897	8.4972697	0.0155357	0.284039
All Points+PFHRGB+point-to-plane	0.172739	10.8989390	0.0155357	0.284039
All Points+SHOT+point-to-plane	0.1716471	42.1088471	0.0155388	0.284038
All Points+SHOT+point-to-point	0.4255302	42.4977302	0.0156969	0.282453

TABLE 5: ICP registration results for test user 1 sitting up straight.

Fig. 7: Registration results for test user 1 sitting up straight using All Points+FPFH+color-ICP. Left: initial point clouds; Middle: after coarse registration; Right: after ICP registration.

5.1.2 Test User 2 (Female)

In Table 6 and 7, we list the coarse and ICP registration results for test user 2 sitting up straight. The list of top algorithms in terms of accuracy is similar to that for test user 1. However, the ICP algorithms only improve the coarse alignment slightly, and the ICP registration results are worse than for test user 1.

Fig. 8: Registration results for test user 2 sitting up straight using All Points+FPFH+point-to-point. Left: initial point clouds; Middle: after coarse registration; Right: after ICP registration.

Algorithms	Time (s)	RMSCE (m)	MCE (m)
All Points+FPFH	6.5532	0.0253	0.4593
All Points+SHOTColor	95.2939	0.0295	0.4849
All Points+PFHRGB	7.8707	0.0325	0.4769
All Points+SHOT	21.4263	0.0339	0.1976
SIFT+FPFH	5.1938	0.0458	0.4550
ISS+PFHRGB	3.9790	0.0545	0.2180
SIFT+SHOTColor	6.3588	0.0586	0.2390
ISS+SHOTColor	3.9631	0.0648	0.2282
SIFT+PFHRGB	5.1448	0.0651	0.4783
SIFT+SHOT	5.4594	0.0722	0.2646
ISS+SHOT	3.9235	0.0775	0.2765
All Points+3DSC	113.8454	0.0777	0.4517
ISS+FPFH	4.0968	0.0810	0.2442
SIFT+3DSC	6.7184	0.0829	0.2719
NARF+3DSC	3.3331	0.1084	0.4276
NARF+PFHRGB	3.3808	0.1084	0.4276
NARF+SHOT	3.4350	0.1084	0.4276
NARF+SHOTColor	3.4782	0.1084	0.4276
NARF+FPFH	3.4922	0.1084	0.4276
ISS+3DSC	3.8989	0.1084	0.4276

TABLE 6: Coarse registration results for test user 2 sitting up straight.

Algorithms	ICP time (s)	Total time (s)	ICP RMSCE (m)	ICP MCE (m)
All Points+FPFH+point-to-point	0.2657542	6.9294542	0.0241943	0.451891
All Points+FPFH+color-ICP	0.2293787	6.5925787	0.0252828	0.166874
All Points+FPFH+point-to-plane	0.1758761	6.6765461	0.0275143	0.450531
All Points+PFHRGB+point-to-point	0.2359052	8.2850152	0.0278369	0.468062
All Points+SHOTColor+point-to-point	0.2359350	95.3924350	0.0279272	0.462050
All Points+SHOTColor+color-ICP	0.2037940	97.0842940	0.0281132	0.473025
All Points+PFHRGB+color-ICP	0.2123878	8.4493578	0.0281810	0.472483
All Points+PFHRGB+point-to-plane	0.1389971	8.0945771	0.0283307	0.464510
All Points+SHOTColor+point-to-plane	0.1065028	96.4375028	0.0283822	0.455710

TABLE 7: ICP registration results for test user 2 sitting up straight.

5.2 Holding Object

In Table 8, we list the coarse registration results for test user 1 holding an object. For keypoint detectors, Use All Points still generally outperforms SIFT, NARF, and ISS in terms of registration accuracy but ISS works pretty good when paired with PFH-RGB. For feature descriptors, there is no obvious winner and the same descriptor paired with different keypoint detector can lead to very different results.

In Table 9, we list the ICP registration results. All the algorithm combinations perform similarly in

terms of registration accuracy, reaching an RMSCE of about 8.8 cm. The ICP refined alignments do not improve much upon the coarse alignments.

Algorithms	Time (s)	RMSCE (m)	MCE (m)
All Points+3DSC	288.2582	0.0842	0.3958
ISS+PFHRGB	4.4138	0.0960	0.4080
All Points+SHOT	45.5421	0.0965	0.3570
SIFT+3DSC	10.2706	0.1026	0.3970
All Points+FPFH	8.4364	0.1030	0.4368
All Points+PFHRGB	11.1968	0.1075	0.4472
SIFT+PFHRGB	6.3519	0.1077	0.4427
SIFT+SHOTColor	9.1327	0.1101	0.3905
All Points+SHOTColor	239.9269	0.1124	0.4005
ISS+FPFH	4.4942	0.1127	0.4347
SIFT+FPFH	6.3545	0.1176	0.3629
SIFT+SHOT	6.8911	0.1333	0.3857
ISS+SHOT	4.3883	0.1370	0.3716
ISS+SHOTColor	4.4232	0.1464	0.5000
NARF+SHOT	3.4897	0.1478	0.4270
NARF+3DSC	3.5411	0.1478	0.4270
NARF+SHOTColor	3.5613	0.1478	0.4270
NARF+FPFH	3.6059	0.1478	0.4270
NARF+PFHRGB	3.6358	0.1478	0.4270
ISS+3DSC	4.3986	0.1478	0.4270

TABLE 8: Coarse registration results for test user 1 holding an object.

Algorithms	ICP time	Total time	ICP RMSCE	ICP MCE
All Points+SHOT+color-ICP	0.3613048	45.4901048	0.0878607	0.348174
All Points+3DSC+point-to-point	0.4449792	286.8779792	0.0880570	0.373715
All Points+3DSC+color-ICP	0.3197620	284.3807620	0.0884108	0.348174
All Points+SHOT+point-to-plane	0.2574489	45.3856489	0.0889373	0.351296
ISS+PFHRGB+point-to-point	0.2994981	4.7011881	0.0891415	0.351056
ISS+PFHRGB+point-to-plane	0.2603290	4.6380190	0.0891687	0.351296
All Points+3DSC+point-to-plane	0.2828851	284.8028851	0.0893555	0.351276
All Points+SHOT+point-to-point	0.3452160	45.5648160	0.0893731	0.351056
ISS+PFHRGB+color-ICP	0.3584950	4.7095750	0.0895911	0.348174

TABLE 9: ICP registration results for test user 1 holding an object.

Fig. 9: Registration results for test user 1 holding an object using All Points+SHOT+color-ICP. Left: initial point clouds; Middle: after coarse registration; Right: after ICP registration.

5.3 Sitting Sideways

In Table 10, we list the coarse registration results for test user 1 sitting sideways. The coarse registration results are worse than those of the sitting up straight scene but are comparable to those of the holding object scene. In terms of registration accuracy, Use All Points outperforms SIFT, NARF, and ISS for keypoint detection; PFH-RGB and Color-SHOT, the two descriptors that incorporate color information, outperform all the other descriptors.

In Table 11, we list the ICP registration results. All the algorithm combinations perform similarly in terms of registration accuracy, reaching an RMSCE of about 8.5 cm. Also, it is worth noting that the ICP refined alignments do not improve upon the coarse alignments.

Fig. 10: Registration results for test user 1 sitting sideways using All Points+PFHRGB+point-to-point. Left: initial point clouds; Middle: after coarse registration; Right: after ICP registration.

Algorithms	Time (s)	RMSCE (m)	MCE (m)
All Points+PFHRGB	9.9959	0.0753	0.4303
All Points+SHOTColor	196.8858	0.0800	0.4269
All Points+3DSC	269.6997	0.0873	0.3457
SIFT+PFHRGB	6.0237	0.0876	0.3070
All Points+SHOT	38.5081	0.0886	0.4144
All Points+FPFH	7.7910	0.1007	0.3702
SIFT+FPFH	5.9326	0.1041	0.4621
SIFT+3DSC	8.9177	0.1128	0.4038
ISS+SHOT	4.2407	0.1129	0.3780
SIFT+SHOTColor	8.2242	0.1140	0.4868
ISS+PFHRGB	4.2388	0.1229	0.3898
SIFT+SHOT	6.3399	0.1328	0.4718
ISS+SHOTColor	4.2871	0.1529	0.5000
ISS+FPFH	4.2954	0.1652	0.5000
NARF+3DSC	3.3400	0.1959	0.5000
NARF+SHOTColor	3.3641	0.1959	0.5000
NARF+SHOT	3.3656	0.1959	0.5000
NARF+PFHRGB	3.4277	0.1959	0.5000
NARF+FPFH	3.4669	0.1959	0.5000
ISS+3DSC	4.1301	0.1959	0.5000

TABLE 10: Coarse registration results for test user 1 sitting sideways.

Algorithms	ICP time	Total time	ICP RMSCE	ICP MCE
All Points+PFHRGB+point-to-point	0.3405781	10.3644781	0.0773697	0.444735
All Points+PFHRGB+color-ICP	0.3305154	10.3074454	0.0841944	0.444831
All Points+SHOTColor+color-ICP	0.3319890	194.8319890	0.0841944	0.444831
All Points+3DSC+color-ICP	0.3312569	224.9312569	0.0841944	0.444831
All Points+SHOTColor+point-to-point	0.2810199	194.9990199	0.0887514	0.444631
All Points+3DSC+point-to-point	0.2796409	224.6506409	0.0887514	0.444631
All Points+PFHRGB+point-to-plane	0.2069662	10.5309662	0.0907480	0.446678
All Points+SHOTColor+point-to-plane	0.1569500	195.0429500	0.0907525	0.446686
All Points+3DSC+point-to-plane	0.1576631	225.6326631	0.0907525	0.446686

TABLE 11: ICP registration results for test user 1 sitting sideways.

6 Discussion

6.1 Result Analysis

6.1.1 Effect of Keypoint Detectors

In all three scenes, using all points as keypoints outperforms the explicit keypoint detection algorithms 3D-SIFT, NARF, and ISS in terms of registration accuracy. This suggests that the explicit keypoint detectors fail to detect highly distinctive keypoints under the ”Telewindow” project setting and further hurt the coarse registration accuracy by reducing the number of keypoints for correspondence matching. In Fig. 11, we plot the average number of detected keypoints and the average RMSCE/MCE of coarse registration using different keypoint detectors. It is clear that when the number of detected keypoints reduces, the RMSCE increases. This suggests that either we need a more powerful keypoint detector that is capable to detect highly distinctive keypoints, or we should just use all points from the downsampled clouds as keypoints. The human facial keypoint detectors used for facial recognition might be helpful.

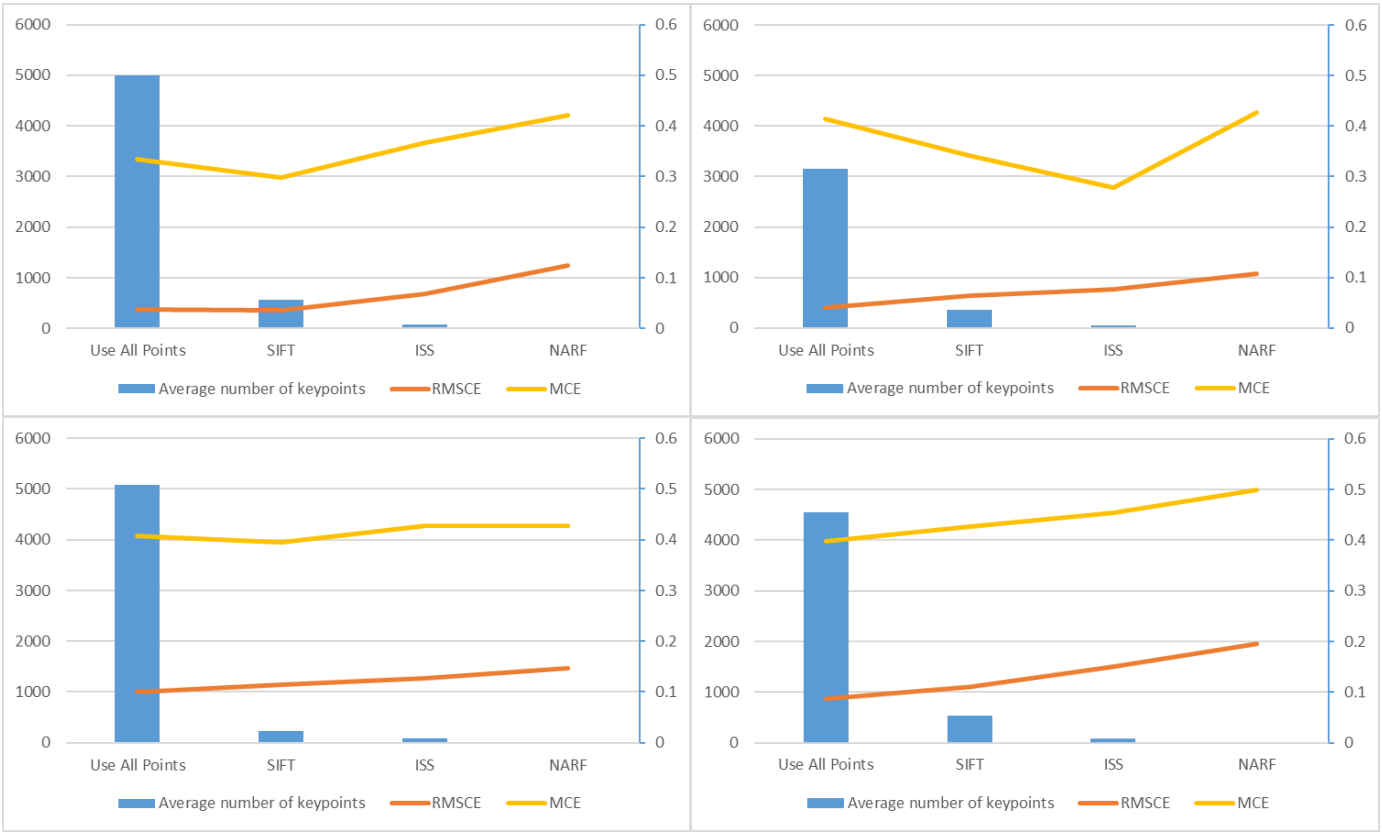


Fig. 11: Average number of detected keypoints and the average RMSCE/MCE of coarse registration using different keypoint detectors. Top-left: Test user 1 sitting up straight; Top-right: Test user 2 sitting up straight; Bottom-left: Test user 1 holding object; Bottom-right: Test user 1 sitting sideways.

6.1.2 Effect of Feature Descriptors

There is no dominating feature descriptor in our experiments. In terms of coarse registration accuracy, FPFH, PFHRGB and SHOT are the best three. However, SHOT is much slower than the other two, as it computes a much higher-dimensional feature vector for each keypoint (see Appendix B). Although our point clouds are colored, the two feature descriptors incorporating color information, PFH-RGB, and Color-SHOT, do not show superiority over the others except for the sitting sideways scene. [15] reports that ISS + 3DSC computes the most accurate registration results among the combinations they tested, but in our experiments, this pair produces the worst registration results.

6.1.3 Effect of ICP Algorithms

J.Park et al. [10] claims that the color-ICP they proposed produces more accurate registration results and runs faster than point-to-point and point-to-plane ICP. This is not supported by our experiments though. The three tested ICP algorithms produce similar registration results in all scenes, but color-ICP is slightly better. Point-to-plane ICP runs faster than point-to-point ICP and color-ICP. However, all of them can finish running within 1 s, much faster than the coarse registration algorithms. Therefore, in practice, we think color-ICP is the optimal one to use for refinement.

6.1.4 Effect of Different Scenes

Our registration pipeline works best when the test user is sitting up straight. Holding objects or sitting sideways worsens the registration results. When the test user sits sideways towards one side, part of him/her body will be blocked from the other side's depth camera. This will potentially hurt the registration results. We do not know why holding a bottle would worsen the registration results. A potential explanation is that both the shape and the color of the bottle are similar to that of the test user's arm, which confuses the algorithms. Also, our registration pipeline produces less accurate alignment for test user 2 sitting up straight than for test user 1. In Table 12, it is clear that the point clouds of test user 2 have much fewer points. Since we observe a positive correlation between the number of points and registration accuracy in section 6.1.1, this might account for the worsened result. However, we are not knowledgeable about why the point clouds of test user 2 have fewer points.

Point Cloud	Test User 1	Test User 2
1	7283	4750
2	5711	3721
3	2477	1568
4	4531	2592

TABLE 12: The number of points in each of the 4 point clouds of Test User 1 and 2.

6.1.5 Optimal Pipeline

Based on the previous analysis, we identify the optimal algorithm combinations under the "Telewindow" project setting, shown in Table 13.

Stages	Optimal Algorithm
Keypoint Detection	Use All Points
Feature Description	FPFH/PFH-RGB
ICP Refinement	color-ICP

TABLE 13: Optimal registration pipeline.

6.2 Issues and Challenges

During the development of the project, we encountered several issues and key challenges. For some of them, we were able to find a solution, and for the rest, we found alternative approaches. We list these challenges below and discuss the decisions we made when attempting to solve them.

6.2.1 Building the Pipeline Architecture

Constructing the pipeline architecture of the project is fundamental to our project, and was also the first challenge we encounter.

Firstly, since our pipeline is relatively complicated, it was hard to build it from scratch. Therefore, we found and forked an open-sourced third-party implementation of a registration pipeline from GitHub (see Acknowledgement). This pipeline was not able to be run properly because it was implemented based on an obsolete version of the Point Cloud Library and does not include some of the algorithms that we wanted to experiment with. However, it provides us an idea of how a registration pipeline is implemented so that we could proceed by rebuilding and extending their implementation.

Secondly, our pipeline encompasses so many stages and algorithms that no single library is able to provide support for all of them. Therefore, we implemented the pipeline with two libraries. On the one hand,

for the implementation of coarse registration, we need to construct various combinations of algorithms. Therefore, we implemented it with the Point Cloud Library, which does not include some state of the art algorithms but provides an almost exhaustive list of the coarse registration algorithms that we need. On the other hand, for the implementation of local refinement, we aimed for better performance and efficiency of the algorithms as we only need to experiment with three different ICP algorithms. Therefore, we used Open3D, which is more advanced, as several newest versions of ICP were actually published by the authors of Open3D.

6.2.2 Image Preprocessing

It was an unexpected fact for us that how much impact image preprocessing has on the accuracy and performance of our pipeline. Initially, we fed our pipeline with the raw point cloud data captured by the cameras, which includes both the user and the background. We hoped that including more data could allow the feature detectors to detect more meaningful feature points. However, the fact that the background is messy and changes over time actually create more noises that creating more meaning for features. Therefore, we decided to eliminate backgrounds and register only based on human figures.

After eliminating the background, we assumed that our algorithm should become more efficient since there are fewer points to process. However, in the beginning, the time that it took to process our data stayed the same. We identified the problem by investigating the structure of our point cloud, and it turned out that it kept all the points that we eliminated as empty points with a zero coordinate and were still computed by our pipeline. After filtering the points with zero coordinates, the speed of our algorithms improved significantly.

6.2.3 Parameter Tuning

Finding the best parameters of our algorithms played an important role in our project. Initially, we mostly adopted the suggested parameters offered by the two libraries, but the result was that the algorithms either end up with an error or performs badly. After that, we discovered that the suggested parameters by the libraries all assumed a much higher camera resolution than the ones Telewindow uses. For instance, one key parameter was ”search radius”, which is used by almost all coarse registration algorithms for computing point features. Since the cameras that we used have a lower resolution, in other words, have a larger distance between the points, searching with the suggested radius often lead to inaccurate point features. To solve this problem, we investigated the hardware configuration of our cameras and tuned the parameters to it

such that our algorithms could return meaningful results. For an exhaustive list of the tuned parameters, please see the appendices.

6.2.4 Metrics

As mentioned in section 4.2, we calculated the matching errors between the registered source point clouds and the target point clouds as our metrics. In fact, this was not the original plan. Initially, we attempted to obtain the ground truth for the transformation matrices with a checkerboard (a.k.a a calibration board). A checkerboard provides a camera with extra significant features (see Fig. 12) such that, theoretically the algorithm can obtain the ground truth.

However, feeding images that involve a checkerboard directly to our pipeline does not yield the perfect results as we expected. In fact, our further investigations indicate that we need to adopt extra algorithms from the OpenCV library to actually obtain a ground truth with a checkerboard. We considered doing this out of the scope of our project.

As a result, we had to find an alternative approach, which is the one we described in section 4.2. It was actually inspired by how the ICP method provided by Open3D computes its internal error. Therefore, we adopted this idea and implemented a program computing the metrics for both the coarse registration process and the local refinement process.

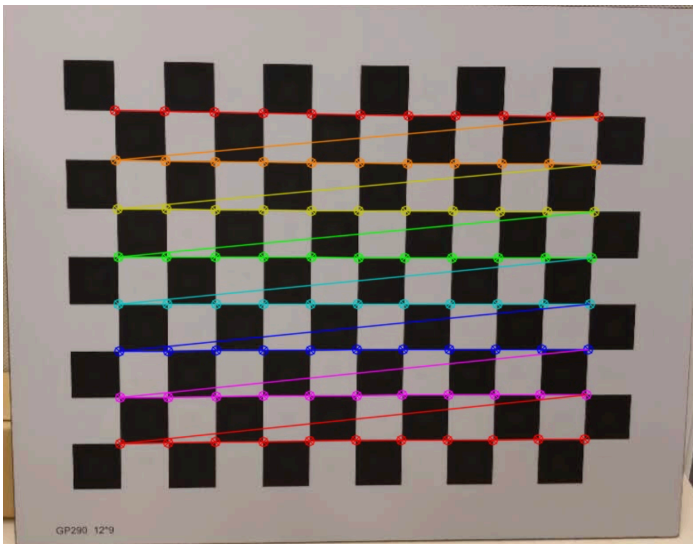


Fig. 12: A checker board and its features identified.

7 Conclusion

In conclusion, this project proposes an algorithm combination (Use All Points for keypoint detection, FPFH/PFH-RGB for feature description, color-ICP for refinement) for 3D registration that is both efficient and accurate given clean point clouds of a human figure (i.e. with the user sitting straight up without holding any extra object). Here, we conclude this report by summarizing our contributions and state some possible improvements and future works.

7.1 Summary of Contributions

- Reviewed multiple highlighted algorithms for 3D registration and selected the most potential ones for experiments.
- Built an experimental pipeline for the 3D point cloud registration under the “Telewindow” project setting by rebuilding and extending an open source pipeline that we forked.
- Tuned each of the algorithms at different stages of our pipeline by investigating the specific point cloud data captured and the hardware configuration of ”Telewindow”.
- Conducted experiments under different scenes with different test users and analyzed the result for each test cases.
- Identified some promising pipelines that are able to finish the registration within 10 seconds and yield accurate registration results given clean enough point cloud data.

7.2 Possible improvements and Future Works

Although our proposed algorithm combinations yield satisfactory results given clean point clouds, there is still much room for improvement.

Currently, the performance of our solution relies heavily on the data given. However, our pipeline should still be feasible for “Telewindow” as we can require a user to sit straight-up at the start-up phase of the “Telewindow” software. However, this is not user-friendly enough and denies the possibility of validating the registration by running the algorithm periodically during teleconferencing.

Moreover, it is still possible to improve the performance of our algorithms. As mentioned, some papers have pointed out that the combination of the ISS keypoint detector and the 3DSC feature descriptor should have a very good performance. However, in our experiments, it was entirely the opposite. Therefore, it is possible that we have not tuned the parameters for these two algorithms to the best. Meanwhile,

for correspondence filters, we are currently using RANSAC, which is arguably the most powerful one as suggested by our literature review. However, some papers also point out that stacking it with other correspondence filters might work even better. Due to the time limit, we did not include this in our experiments.

In terms of the positive correlation between the number of points and registration accuracy as we mentioned previously in section 6.1.1, there is a potential solution that we did not experiment. We can actually add point clouds captured from consecutive frames together to create more points, which might help the coarse registration to have a better performance.

Last but not least, despite the fact that we gave up on obtaining ground truth with a checkerboard as the work required for this lies outside of the registration algorithms that we have reviewed, we still think it is a very promising approach. It not only provides a potentially better way to computing errors for our pipeline but also yields a nearly perfect alignment (but would require regular calibration).

With the above being said, we list some possible future works on this project as the following:

- Continue to tune the parameters for some currently unsatisfactory, but theoretically potential algorithms.
- Experiment with stacking correspondence filters.
- Experiment with adding point clouds from multiple consecutive frames to provide more data samples for feature detectors.
- Use a checker/calibration board to obtain accurate registration results that can be hard-coded into the machine. This will yield nearly perfect alignment yet requiring regular calibration.
- Incorporate human facial keypoint detectors to obtain highly distinctive keypoints for registration such that the algorithm can be more robust against noises.

Acknowledgement

We would like to thank Prof. Olivier Marin, Prof. Michael Naimark, Cameron Ballard and Bruce Luo for their professional advice and assistance. We also thank the author of the pipeline we forked, Tiancheng Xu from the University of Rochester. Our sincere thanks also go to Wenqian Hu for participating in our experiments. Finally, this study would not have been made possible without the support from New York University Shanghai. We acknowledge all the NYUSH staff and faculty members working on the CS Senior Project class.

References

[1] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn, "A survey of rigid 3d pointcloud registration algorithms," in *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies*, Rome, Italy, Aug. 2014, pp. 8–13.

[2] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision*, Amsterdam, Netherlands, Oct. 2016, pp. 766–782.

[3] P. Li, J. Wwang, Y. Zhao, Y. Wang, and Y. Yao, "Improved algorithm for point cloud registration based on fast point feature histograms," *Journal of Applied Remote Sensing*, vol. 10, no. 4, 2016.

[4] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," *P. S. Schenker, Ed.*, pp. 586–606, Apr. 1992.

[5] S. M. J. Guivant, "Improving the performance of icp for real-time applications using an approximate nearest neighbour search," in *Australasian Conference on Robotics and Automation*, ACRA, New Zealand, 2012, pp. 3–5.

[6] D. Holz, A.-E. Ichim, F. T. R. B. Rusu, and S. Behnke, "Registration with the point cloud library: A modular framework for aligning in 3-d," *IEEE Robot. Autom. Mag.*, vol. 20, no. 4, pp. 110–124, Dec. 2015.

[7] Y. Chen and G. Medioni, "Object modeling by registration of multiple images," *International Journal of Image and Vision Computing*, vol. 10, no. 3, pp. 239–256, 1992.

[8] M. Korn, M. Holzkothen, and J. Pauli, "Color supported generalized-icp," in *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal, Jan 2014.

[9] H. Men, B. Gebre, and K. Pochiraju, "Color point cloud registration with 4d icp algorithm," in *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[10] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct 2017.

[11] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Narf: 3d range image features for object recognition," in *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2010.

[12] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM International Conference on Multimedia*, Sep. 2007, pp. 357–360.

[13] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," in *2009 12th IEEE International Conference on Computer Vision Workshop*, 2009, pp. 689–696.

[14] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, Graz, Austria, May 2006, pp. 430–443.

[15] S. Salti, A. Petrelli, F. Tombari, and L. D. Stefano, "On the affinity between 3d detectors and descriptors," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012, pp. 421–431.

[16] R. Hansch, T. Weber, and O. Hellwich, "Comparison of 3d interest point detectors and descriptor for point cloud fusion," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 3, pp. 57–64, 2014.

[17] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, New York, USA, Jul. 1994, pp. 311–318.

[18] T. Masuda, K. Sakaue, and N. Yokoya, "Registration and integration of multiple range images for 3-d model construction," in *Proceedings of 13th International Conference on Pattern Recognition*, Aug. 1996, pp. 879–883.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[20] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, Sep. 2008, pp. 3384–3391.

[21] F. Tombari, S. Salti, and L. D. Stefano, "Unique signatures of histograms for local surface description," in *European Conference on Computer Vision*, Sep. 2010, pp. 356–359.

[22] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *European Conference on Computer Vision*, May 2004, pp. 224–237.

[23] R. B. Rusu and S. Cousins, "Point cloud library (pcl)," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 1–4.

[24] F. Tombari, S. Salti, and L. D. Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching," in *2011 18th IEEE International Conference on Image Processing*, Sep. 2011, pp. 809–812.

[25] A. Mian, M. Bennamoun, and R. Owens, "On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes," *International Journal of Computer Vision*, vol. 89, no. 2-3, pp. 348–361, 2010.

[26] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud, "Surface feature detection and description with applications to mesh matching," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 373–380.

[27] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 3212–3217.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NIPS)*, 2012, pp. 1097–1105.

[29] Z. Yang, T. Dan, and Y. Yang, "Multi-temporal remote sensing image registration using deep convolutional features," *IEEE Access*, vol. 6, pp. 38 544–38 545, 2018.

[30] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2015, pp. 922–928.

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[32] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software*, vol. 3, pp. 209–226, 1977.

[33] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *VISAPP International Conference on Computer Vision Theory and Application*, 2009, pp. 331–340.

[34] M. Muja and D. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 2227–2240, May 2014.

[35] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2008.

[36] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 26, no. 6, pp. 381–395, 1981.

Appendix A

Parameters of Keypoint Detectors

Keypoint Detectors	Parameters
NARF	angular_resolution=0.5 support_size=0.03
3D-SIFT	min_scale=0.01 n_octaves=6 n_scales_per_octave=10 min_contrast=0.05
ISS	iss_salient_radius=6*model_resolution iss_non_max_radius=4*model_resolution iss_border_radius=1*model_resolution iss_normal_radius=4*model_resolution iss_gamma_21=0.975 iss_gamma_32=0.975 iss_min_neighbors=5

TABLE 14: Parameters of Keypoint Detectors.

Appendix B

Parameters of Feature Descriptors

We use a search radius of 2 cm (twice of the grid size of downsampling) for all feature descriptors. For 3DSC, we set its MinimalRadius = 0.2 cm, PointDensityRadius = 0.4cm. The dimensions of all feature descriptors are listed in Table 15.

Feature Descriptors	Descriptor Dimension
FPFH	33
PFH-RGB	250
SHOT	352
Color-SHOT	1344
3DSC	1980

TABLE 15: Dimensions of Feature Descriptors.

Appendix C

Parameters of ICP Algorithms

We downsample the clouds with a uniform grid size of 1 cm before applying point-to-point and point-to-plane ICP. Normal estimation for point-to-plane ICP is performed by KDTreeSearchParamHybrid with radius = 1 cm and max_nn = 30. For both point-to-point and point-to-plane ICP, we set the max correspondence distance to 2 cm and use the default termination criteria: relative_fitness = 0.000001, relative_rmse = 0.000001 and max_iteration = 30.

For color-ICP, following the tutorial in Open3D Document, we apply the algorithm on three different scales of point clouds. On each scale, normal estimation is performed by KDTreeSearchParamHybrid with radius = 2*downsampling_grid_size and max_nn = 30. We use the following termination criteria: relative_fitness = 0.000001, relative_rmse = 0.000001 and max_iteration varies with scales.

Scale	Grid size of downsampling	Maximum iteration allowed
Rough	8 cm	200
Medium	4 cm	100
Fine	2 cm	50

TABLE 16: Three different scales on which color-ICP is applied.