VIETNAM NATIONAL UNIVERSITY, HANOI

UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Nguyen Hung Thinh

3D REGISTRATION

Major: *computer science*

HA NOI - 2015

VIETNAM NATIONAL UNIVERSITY, HANOI

UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Nguyen Hung Thinh

3D REGISTRATION

Major: **computer science**

Supervisor: Assoc. Prof. Bui The Duy

Co-Supervisor: Dr. Ma Thi Chau

HA NOI - 2015

AUTHORSHIP

"I hereby declare that the work contained in this thesis is of my own and has not been previously submitted for a degree or diploma at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no materials previously published or written by another person except where due reference or acknowledgement is made."

Signature:……………………………………………

SUPERVISOR’S APPROVAL

“I hereby approve that the thesis in its current form is ready for committee examination as a requirement for the Bachelor of Computer Science degree at the University of Engineering and Technology.”

Signature:………………………………………………

ACKNOWLEDGEMENT

I would like to express my great appreciation to Prof. Bui The Duy and Dr. Ma Thi Chau for their invaluable guidance, comprehensive encourage and useful critiques. This thesis would not have been possible without their help.

I consider it an honor to work with my colleagues: Tran Minh Luyen, Nguyen Tuan Anh, Phan Dang Thanh. They go along with me since the beginning of this work, giving me many useful advices to overcome many obstacles, both in study and life.

I also would like to thank University of Engineering And Technology, faculty of Information Technology and Human Machine Interaction Lab for all their support.

ABSTRACT

In Computer Vision, 3D registration problem or aligning multiple 3D data sets to single coordinate system problem is fundamental and essential. Its applications vary from reconstructing 3D models to shapes matching, which can be applied to medical visualization, environment reconstruction or robot navigation. This problem has attracted many researchers' attention. Many methodologies have been proposed: searching for global optimization transformation by finding correct correspondences, local optimization by minimizing an error metric that measures the closeness of two input data sets or combining both methods. In our thesis, we adapt the local optimization methodology, specifically, the Iterative Closest Point algorithm to solve the registration problem. However, as a local optimization process, the initial positions of point cloud data are crucial for algorithm performance. So, we proposed a semi-automatic process to aid this weakness and apply the algorithm to register a set of point cloud data. Three real world data sets have been used for testing this process and the result shows that the method is efficient and can be used for a real world application.

TABLE OF CONTENT

List of Figures

ABBREVATIONS

ICP	Iterative Closest Point algorithm
GICP	Generalized ICP
PCA	Principle Component Analysis
KD-tree	K dimensions tree
RGB	Red-Green-Blue
SIFT	Scale-invariant feature transform
FPFH	Fast Point Feature Histograms
RVM	Root Mean Square

Chapter 1

INTRODUCTION

Nowadays, advancements in computer vision have brought depth data closer and closer to our daily life. Depth data exists in every corner, not just in computer vision for robot navigation or medical visualizer, but also in application for interaction between end users and the environment, such as Kinect fusion (Shahram, et al., 2011) from Microsoft, and many other applications.

The appearance of cheap and convenient 3D cameras like Microsoft Kinect makes sampling 3D data widely available for mainstream users, students and researchers. However, those devices mostly provide raw point clouds data one perspective at a time, which is not enough for reconstructing 3D models. Therefore, the registration process comes in to play. Point clouds are taken at multiple views and are aligned together to achieve complete, higher quality cloud for reconstruction process.

For that reason, registration arises as a subtask in many different applications, and attracts attention from researcher community. Many approaches have been proposed. One of them is Iterative Closest Point algorithm (Yang & Gerard, 1991) (Paul & Neil, 1992). Despite the fact that ICP is widely used, it has some significant drawbacks, which come from the algorithm's assumptions. Firstly, point clouds participate in registration process must contain overlap, and the overlap must be large proportion to number of points in those point clouds. Secondly, the relative position between point clouds is assumed to be close initially. In addition, another drawback comes from the characteristic of the process of registration a list of point clouds. Registration means to find the relation between point clouds to transform them to unique coordinate, hence, one broken transformation will result an unusable combination for further processing.

This thesis proposes a semi-automatic system that utilizes some simple actions from human to eliminate those drawbacks from original ICP algorithm. Users will control the initiation step, which is aligning point clouds to satisfy the constraint of close distance. Moreover, users decide what result is good, what result is bad and decide whether realigning current result is needed. Also, this thesis adapts the pair-wise approach. Consecutive point clouds will be aligned pair by pair to maximize the proportion of overlapping region compared to the cumulative approach: align-merge-align.

The rest of the thesis is structured as follows. Chapter 2 provides detailed description about background works that our method depends on. Firstly, it introduces briefly the general concept of registration and then examines more specific topics for this thesis: KD-tree, normal space sampling, normal estimation and phases in ICP algorithm. Chapter 3 contains the proposed method for improving the ICP algorithm and the process of registering a set of point cloud data. In chapter 4, there are results of several experiments on real world data. All data were archived by Microsoft Kinect and each scene contains different geometry features for testing our method.

Chapter 2

BACKGROUND KNOWLEDGE

This chapter first introduces some basic terminology of registration in general and then provides theoretical description of background works, which are KD-tree, normal estimation with PCA, sampling methods and Iterative Closest Point algorithm.

2.1 Registration in general

This section introduces some basic definitions in 3D registration problem.

Point Cloud data

Point Cloud data is a set of 3-dimensional points achieved by depth sensors, e.g. Microsoft Kinect.

Registration of point clouds

Registration of point clouds is the process of aligning point clouds to correct position in a particular coordinate system. More specifically, the goal of registration process is to find a homogenous transformation matrix that transforms the coordinate system of a point cloud to other's to make two point clouds have correct relative position with each other in physical world.

Homogenous transformation

The Homogenous transformation matrix for 3D case is a 4x4 matrix.

$$H = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

$$R: [3x3]\ matrix\ that\ represents\ rotation$$

$$T: [3x1]\ matrix\ that\ represents\ translation$$

The registration method in the context of this thesis only deals with rigid transformation. So the scale factor is set to constant 1.

2.2 Data preprocessing

At the beginning of registration process, preprocessing data is very important because it improves both speed and correctness. It will be described in this section.

2.2.1 KD-tree

In this thesis, various phases depend on computing the nearest neighbors of points (section 2.2.3, 2.3.1), which is expensive for unstructured point clouds. So the method of KD-tree is used for organizing point cloud. And its direct application is computing nearest neighbor. This section provides basis of KD-tree (Jon, 1975) and the nearest neighbor finding.

First, defining KD-tree. KD-tree is a binary tree, which is constructed as follow:

Algorithm: construct KD-tree

Input: a set of points X measured in n dimensions

Output: a KD-tree that represents X

Parameter: Y is current data set, initially, Y = X. i is current dimension

```
1    If i > n then i = 1
2    current_node = find median of data set Y in dimension i
3    Separate Y to $Y_1$ and $Y_2$ which contain remain items in Y having smaller
     and bigger value in dimension i respect to current_node
4    current_node.left = build_kd_tree ($Y_1, i + 1$)
5    current_node.right = build_kd_tree ($Y_2, i + 1$)
```

Below is a small example of KD-tree in 2 dimensions. The case of 3 dimensions is analogous.

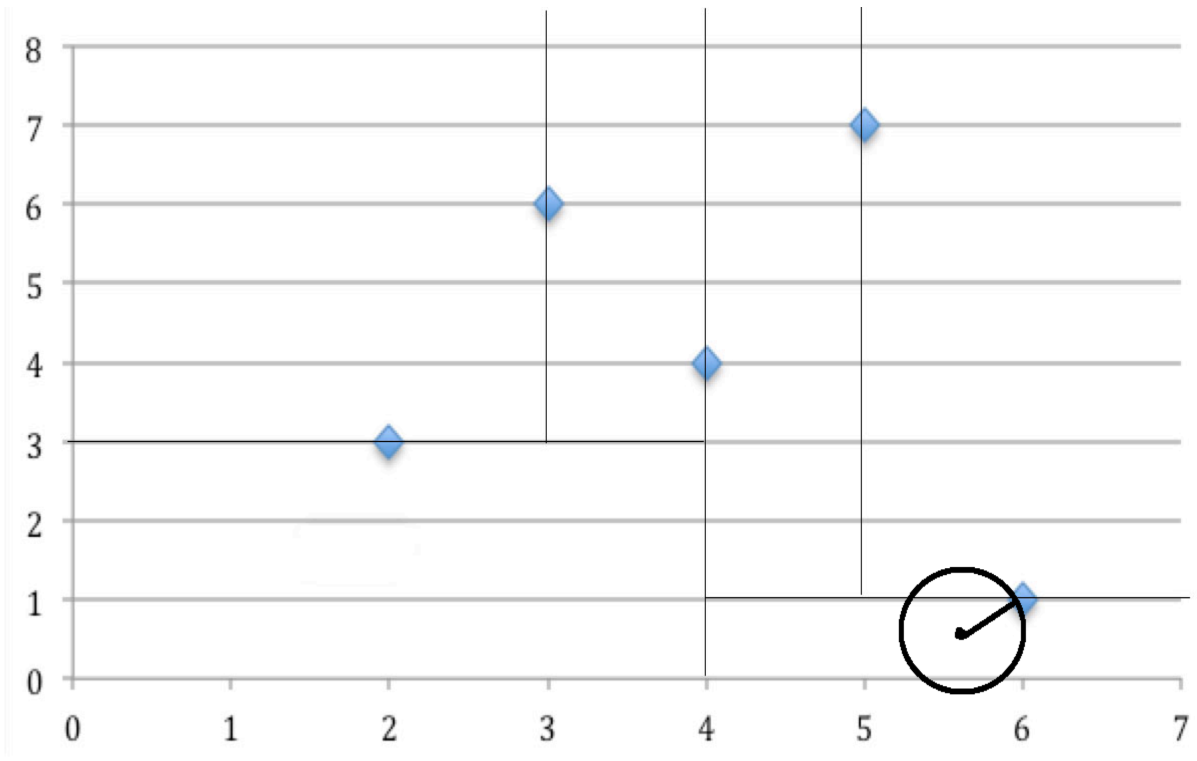


Figure 1. Visualize KD-tree in two dimensions case

We have a set of data point (4, 4), (3, 6), (2, 3), (5, 7), (6,1)

Choose x coordinate: and the median is 4, choose point (4, 4) as first node. (3,6) and (2,3) are on the left side of (4, 4). (5, 7) and (6,1) are on the right side of (4, 4)

After that, next coordinate to examine is y. On the left the median in y coordinate is 3. So (2, 3) will be the next node. (3,6) fall into the right side of (2,3). The right side of (4, 4) is similar.

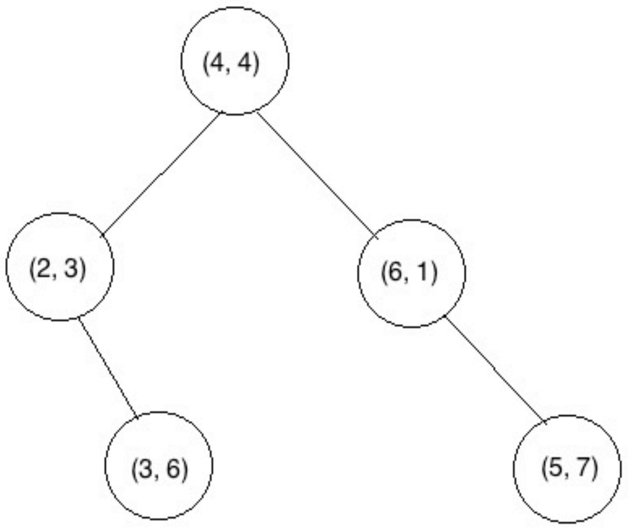


Figure 2. The result of KD-tree in binary tree form

Second, computing the nearest neighbor. In figure 1, the black dot is a test point, and the goal is to find nearest neighbor of the test point in the tree. The test point is lie on the lower half of point (6, 1). So, first, it travels all the way from the root to that node and got the current smallest distance, which is illustrated by black line (note that, in the first time it travel, there are nodes in other side of some nodes it did not travel to). However, to make sure the distance it found is smallest, the algorithm draws a circle with radius is the current smallest distance, if that circle intersects with the borderline then there is possibility that other half of border line contains a smaller distance, so it must travel to the other half. The intersection between the circle and the borderline can be expressed by comparing the radius with distance of the test point to borderline.

Algorithm: find closest point in KD-tree

Input: KD-tree, a point x

Output: closest point n to x

```
1    Initialize n = null, distance  = infinity
2          Start travel through tree with root node
```

```
3     If distance > distance (current_node, x)
4         Distance  = distance (current_node, x)
          n  = current_node
5     If   (x >= current_node)
6         //only compare the value in dimension used for splitting points
7         Travel recursively right sub tree of current node
8     Else
9         Travel recursively left sub tree of current node
10    If   current_node – x < distance
11        //minus between the value in dimension used for splitting points
12        Travel recuresively remain sub tree of current node
```

2.2.2 Normal estimation

Normal is important feature used for sampling and ICP. However, the computation in the context of point cloud is not straightforward. It needs to adapt the Principle Component Analysis method (SVANTE, KIM, & PAUL, 1987). This section first introduced the principle component analysis and then describes the usage of PCA in finding normal of points in point clouds.

A. Principle component analysis

Principle component analysis or PCA is a statistical tool. Its main goal is to find a basis that describes a set of data best. PCA has many applications in various fields including this thesis domain: computer vision.

The input of PCA is a set of n data, which is sampled in m dimensions. Denote each vector of data by:

$$X_i = [x_1 \quad \cdots \quad x_m]^T$$

And the matrix X represents all data we have:

$$X = [X_1 \quad \ldots. \quad X_n]$$

Assume that the current basis is:

$$B = \begin{bmatrix} 1 & .. & 0 \\ .. & .. & .. \\ 0 & .. & 1 \end{bmatrix}$$

To find a new basis that describes data best is equivalent to finding a transformation P that transforms data matrix X to new data matrix Y. And Y has following characteristics; the redundancy between dimensions is minimized and the variance in each dimension, which is considered interesting, should be maximized. The redundancy is determined by value of covariance between dimensions. Denote the covariance matrix of a data set Z is: Cov_Z. Note that, means of X and Y is assumed to be 0, if they are not 0 then data of X and Y must subtract their means before the PCA process. The problem can be formulated as:

$$Y = PX$$

We have to find P to make Cov_Y a diagonal matrix. P is called the principle components of X.

We have:

$$Cov_Y = \frac{1}{n} YY^T$$

$$= \frac{1}{n} (PX)(PX)^T$$

$$= \frac{1}{n} PXX^T P^T$$

$$= PCov_X P^T$$

The matrix Cov_X is symmetric matrix so:

$$Cov_X = E.D.E^T$$

$$E\ is\ matrix\ of\ eigen\ vectors\ of\ X$$

$$D\ is\ a\ diagonal\ matrix$$

So:

$$Cov_Y = P(EDE^T)P^T$$

Choose $P = E^T$ and since the E is orthonormal then $P = E^{-1}$

$$Cov_Y = E^{-1}EDE^{-1}E$$

$$= D$$

The conclusion is P = E^T is the principle component of X.

B. Normal estimation using PCA

After applying PCA to a specific point and its neighbors, we will have three eigenvectors with their eigenvalues. Two of them have larger eigenvalue than the third eigenvector. They represent the approximate plane that points lie in. The third eigenvector with smallest eigenvalue is normal of that point's surface.

Algorithm: normal estimation

Input: a point cloud X

Output: normal for each points in X

```
1   for each x in X
2       neighbor = Find n nearest neighbor of x //n can be 20 or any number
3       COV = compute covariance matrix base on neighbor
4       Find eigenvectors and eigenvalue of COV
5       Normal at x = eigenvector with smallest eigenvalue
6   End
```

2.2.3 Sampling

A typical point cloud data which is achieved from Kinect often contains around 100,000 to 400,000 points or even larger, so if point clouds are passed directly to the process of registration, the computation complexity will be significant. Another problem is outliers. They will affect the algorithm to find the wrong transformation. So, sampling the point cloud to decrease the amount of points and outliers is needed. Two simple methods are uniform sampling or random sampling. They do decrease the amount of points. However, when applying those two sampling methods in ICP, another problem arises. In some cases, when input clouds are "geometrically hard" for ICP to process, for example the symmetric case, small features in the cloud is essential for computing the correct alignment, those features are not considered in any two methods above. For that reason, the Normal Space Sampling is proposed in (Szymon & Marc, 2001). Normal Space Sampling selects points that maximize the distribution of points' normal. For the implementation, first thing to do is constructing a 3d histogram of normal with a specific number of bins in x, y, z dimension, then go through all bins sequentially until the desired number of sample is reached.

Algorithm: normal space sampling

```
Input: a set of points X and set normal N
       Number of desired points
       Number of bins in each dimension
Output: a set of sampled points
1   Construct histogram of normal N
2   For each n in N
3       put n in to appropriate bin
4   end
5   current_bin = 0
6   While not picked enough point and histogram not empty
7       If  current_bin not empty
8           Random pick one point
9           Delete picked point in current bin
10      current_bin++
11      If current_bin >= histogram.size
12          current_bin = 0
13  end
```

2.3 Iterative Closest Point algorithm

This thesis studies the problem of registering multiple views. And we use ICP (Yang & Gerard, 1991) (Paul & Neil, 1992) as a core for this process, so this section will introduce the concept of ICP.

Problem statement:

Given two point clouds X = $\{x_i\}$ (i = 1 … m) and Y = $\{y_i\}$ (i = 1 … n) in 3-dimensonal space. The initial transformation is assumed being known (initial system will be discuss in chapter 3). There exists a transformation T, if T is the correct transformation from X to Y then the function F below will be minimized:

$$F(T) = \sum_1^m q_i d(Tx_i,\ Y) \tag{1}$$

Where:

$$q_i = 0\ if\ x_i\ is\ an\ outlier, \qquad q_i = 1\ if\ x_i\ is\ not\ an\ outlier.$$

$$And\ d(x, Y)\ denote\ the\ distance\ function\ from\ point\ x\ to\ Point\ Cloud\ Y$$

This thesis only solves the registration problem between point clouds, so, using the definition Y = $\{y_i\}$ with equation (1) we can rewrite function d as:

$$d(Tx_i, Y) = d(Tx_i, y_i) \tag{2}$$

$$x_i\ and\ y_i\ are\ pair\ point\ represent\ the\ same\ surface\ point$$

And re-formulate the problem as:

$$\underset{T}{argmin} \sum_1^m q_i d(Tx_i, y_i) \tag{3}$$

From its first introduction by (Yang & Gerard, 1991) and (Paul & Neil, 1992), many variants have been proposed to resolve those problems (Szymon & Marc, 2001) (Aleksandr, Dirk, & Sebastian, 2009). In the iterative process below, ICP can be break down into three phrases: matching, rejecting, error metric and minimization. The next three sections will examine each phrase in detail.

The iterative process can be described as follow:

Algorithm: Iterative closest point

```
    Input: source cloud $X = \{x_i\}$, target cloud $Y = \{y_j\}$
    Output: A transformation matrix from $X\ to\ Y$
1   $T \leftarrow T_0$
2   While (not converged or max iterations not reached)
3       For $x_i$ in $X$
4           Correspond pair $(x_i, y_i)$ = Find matching point in Y
5           Reject invalid pair point
6       End
7       $T = \underset{T}{argmin} \sum_i d(Tx_i, y_i)$
8   End
9   return T
```

2.3.1 Matching

(Szymon & Marc, 2001) provided a comparison between several methods: closest point, closest compatible point, normal shoot, normal shoot compatible, project, project and walk. According to their research, the "project" like methods have lowest convergence rate, the "closest point" method is more sensitive to error than others methods. They hypothesize that when meshes are still relatively far from each other, the closet point-matching algorithm generates large numbers of incorrect pairings. However, they also concluded that closest point is the most robust for "difficult" geometry. This thesis adapts the closest point method as matching strategy. The reasons are:

+ The problem with incorrect pairing, there is a rejecter to reject fail correspondences. And user does the initialization in this thesis, so initial position is assumed to be close.

+ In real world, "difficult" geometry will likely to appear often, so a robust strategy is very important to get a good result.

The distance for computing "closest point" in this context is the Euclid distance of two points $d\ (x, y)$.

A naive method to find the closest point of a point x in a point cloud Y is searching all the points in point cloud Y. However, using K-d tree to organize the point cloud (see section 2.2.1 for more information), searching for closest point is accelerated.

2.3.2 Rejection

To transform equation (1) to equation (2), the distance of point x_i to cloud Y is transformed to distance between x_i and individual point y_i of cloud Y. However, the perfect point pairs are not known yet, so the algorithm find point pairs base on the Euclid distance in previous section. So, there exist fail point pairs, which affect the result of ICP. For that reason, those fail pairs should be filtered. In equation (1), q_i is used for rejecting point outlier may cause by data acquisition, for convenient we also use it for rejecting fail point pairs in equation (2). So the definition of outlier is:

+ Point that caused by data acquisition error

+ Invalid point pairs

The first problem was handled by sampling data in section 2.2.3. This rejection phrase of ICP handles invalid point pairs case.

The purpose of registration is to merge two point clouds to produce a point cloud with more information than an individual point cloud. So two point clouds contain overlap region to give information to registration process and also contain region that have extra information. That region does not contribute to the registration process and the distance of point pairs that contain those additional points should have a large value compared to pair point in the overlap region. So, the distance rejecter will reject any point pairs that have distance beyond a specific threshold. Users can set this threshold.

2.3.3 Error metric

As stated, ICP is an optimization problem and can fall into the local optimum trap, all the methods mentioned above is to enhance the chance to reach a global optimum result and also decrease the convergence rate. And error metric plays a very important role in both cases. (Paul & Neil, 1992) proposed the point-to-point metric, a metric that computes transformation by minimizing the distance between points, while (Yang & Gerard, 1991) used point-to-plane error metric. The distance function in point-to-plane is computed by distance from a point and the tangent plane of its correspondence. (Szymon & Marc, 2001) discussed about both methods.

Point to point error metric, the equation (2) becomes:

$$d(Tx_i, y_i) = (Tx_i - b_i)^2 \tag{4}$$

Point to plane error metric, the equation (2) becomes:

$$d(Tx_i, y_i) = (n_i(Tx_i - b_i))^2 \tag{5}$$

$$n_i \ is \ normal \ vector \ at \ point \ b_i$$

$$note \ that \ x_i \ and \ y_i \ are \ correspondence \ pair, which \ was \ found \ in \ matching \ step$$

(Aleksandr, Dirk, & Sebastian, 2009) proposed a different approach by apply a probabilistic model to cope with the problem. And they showed that both pervious methods are special case of their method. That method is called "Generalized ICP". This thesis use Generalized ICP as error metric.

The output of pervious two steps is two sets of points X = $\{x_i\}_{i=1\dots n}$ and Y = $\{y_i\}_{i=1\dots n}$, x_i and y_i are correspondence. The method assumed that there exists two perfect correspondence sets $\hat{X} = \{\hat{x_i}\}_{i=1\dots n}$ and $\hat{Y} = \{\hat{y_i}\}_{i=1\dots n}$ generate X and Y independently by 3-dimensional multivariate normal distribution.

$$x_i \sim \mathcal{N}(\hat{x_i}, C_i^X)$$

$$y_i \sim \mathcal{N}(\hat{y_i}, C_i^Y)$$

$$C_i^X \ and \ C_i^Y \ are \ two \ covariance \ matrices$$

Denote T^* is the correct transformation between two point clouds, we have:

$$\hat{y_i} = T^* . \hat{x_i} \tag{6}$$

Denote $d_i^T = y_i - Tx_i$ we have:

$$d_i^{T^*} = y_i - T^*.x_i$$

Since x_i and y_i generated by two normal distribution:

$$d_i^{T^*} \sim \mathcal{N}(\hat{y_i} - T^*.\hat{x_i}, C_i^Y + T^*.C_i^X.(T^*)^T)$$

Apply (6):

$$d_i^{T^*} \sim \mathcal{N}(0, C_i^Y + T^*.C_i^X.(T^*)^T)$$

The transformation of T will be estimated by:

$$T = \underset{T}{\operatorname{argmax}} \prod_i p(d_i^T) = \underset{T}{\operatorname{argmax}} \sum_i \log(p(d_i^T))$$

Simplified to:

$$T = \operatorname{argmin}_T \textstyle\sum_i (d_i^T)^T (C_i^Y + T.C_i^X.(T)^T)^{-1} (d_i^T) \tag{7}$$

If we use:

$$C_i^Y = I$$

$$C_i^X = 0$$

Then (5) turns into (4)

And for the point-to-plane the setting was:

$$C_i^Y = P_i^{-1}$$

$$C_i^X = 0$$

$$where\ P^i\ is\ the\ projection\ matrix\ onto\ surface\ normal\ of\ b_i$$

Two settings above are point-to-point, and point-to-plane error metric. (Aleksandr, Dirk, & Sebastian, 2009) proposed a different setting with the name of plane-to-plane error metric. The error metric assumes that points have large variance in their local plane and vary little in their surface normal direction. Consider a point with normal vector $n_x = [1, 0, 0]$, to model the assumption above, the point-to-plane algorithm assigns the covariance of that point to:

$$Cov = \begin{bmatrix} \varepsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\varepsilon\ is\ a\ very\ small\ number\ which\ represent\ low\ uncertainty\ along\ normal\ direction$$

$$e.g\ \varepsilon = 0.001\ or\ maybe\ smaller$$

For a random point, we have to rotate above Cov matrix to the point surface normal. In section 2.2.2, surface normal is computed by PCA, which is the eigenvector that have smallest eigenvalue. Denote the rotation matrix R that rotates n_x to that normal vector.

With that, we compute C_i^Y and C_i^X by:

$$C_i^X = R_i^X . Cov . (R_i^X)^T$$

$$C_i^Y = R_i^Y . Cov . (R_i^Y)^T$$

Then substitute to (7) to compute T.

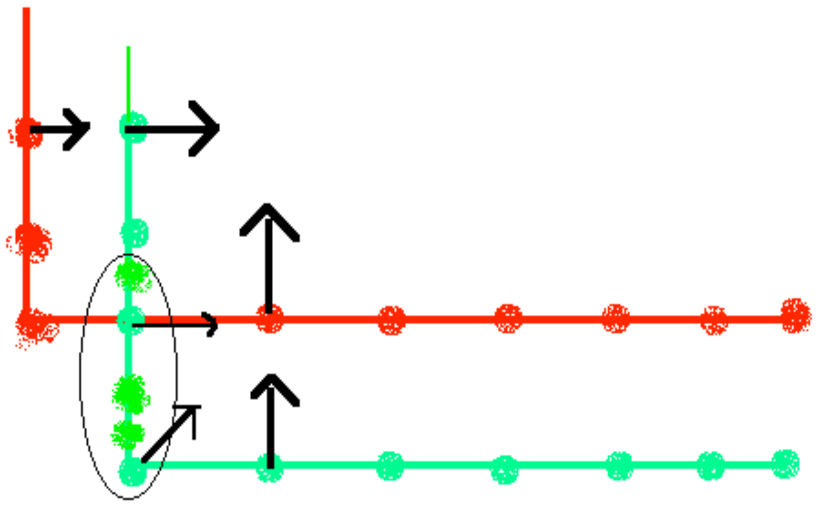


Figure 3. An example of plane-to-plane method

The figure above illustrates an example where there are several fail matches (inside the ellipse). With plane-to-plane configuration, the sum of two covariance matrices will act as a penalty for incompatible normal direction. Hence reduce the influence of those fail matches to overall minimization.

Chapter 3

OUR PROPOSED METHOD

This chapter presents a semi-automatic process to solve the registration problem and also discuss two processes of registering a set of point clouds.

3.1 Semi-automatic registration

The core problem we tackle in our method is producing initialization for ICP. ICP is based on local optimization approach, and for this category, initial position is critical. This problem is hard, but with a little help of human, the obstacle is greatly reduced. The initial position between two point clouds will be manually adjusted. With this simple action, the initial position for point cloud is much more reliable and dynamic. Also, as we mentioned before, pair-wise method is sensitive to error, every link should be monitored carefully or else, failure is likely to occur. As far as we know, there isn't any mechanism to evaluate error automatically since the correct alignment is unknown. Hence, user's confirmation to accept the result or re-register them with different initial position is needed. For these reasons, we proposed a semi-automatic mechanism for register point clouds.

Inputs of the process are two point clouds: point cloud 1 and point cloud 2. First, two point cloud are passed to an interaction window with their initial position is raw data achieved from scan. At this stage, users can freely move the data to create a good initialization position for two point clouds. After that, the initial pose will be used for ICP algorithm. Note that users can move clouds very close together but the result needs to be refined by ICP to achieve the best output. The ICP algorithm in this thesis uses the closest point matching, distance rejecter and the generalized error metric.

Finally, two point clouds are merged and showed to user. If users confirm that the result are bad then the process resets and user can set up another initial position. If the result is confirmed as good result then process produces the transformation between two point clouds. This process is illustrated in figure 4.

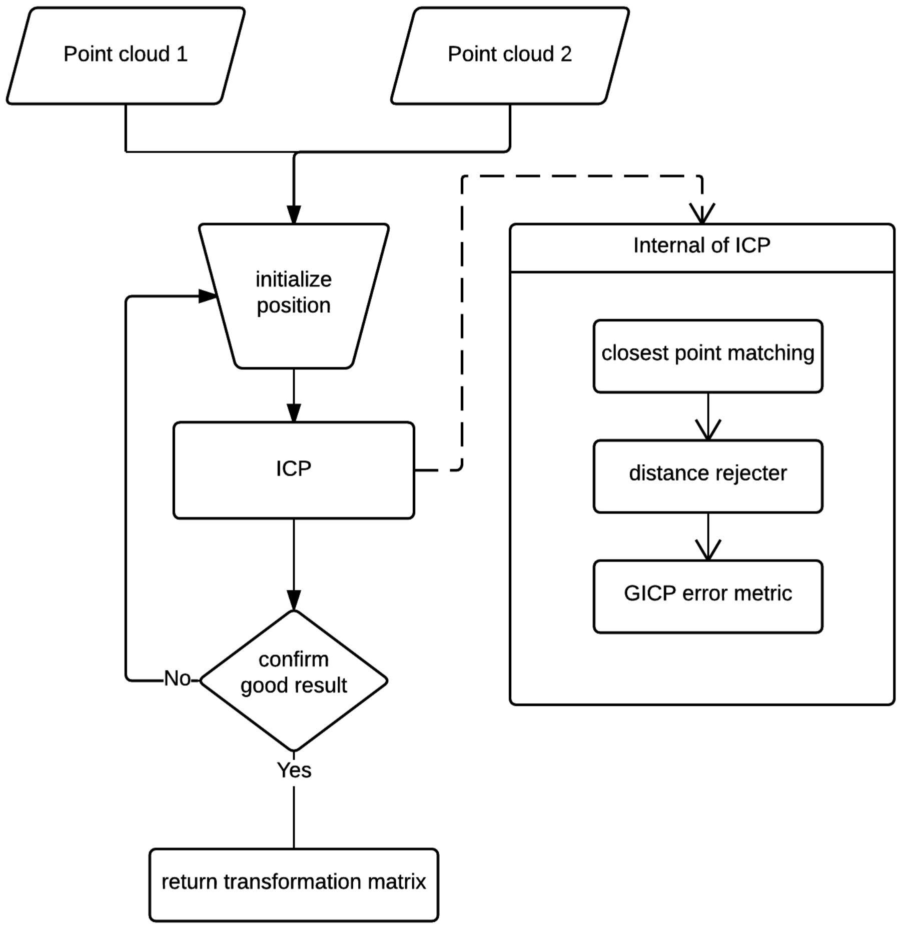


Figure 4. Semiautomatic process for solving registration

3.2 Process of registration

The registration process of a set of point clouds is broken down to the pair-wise registration problem. There are two choices to implement this process, the first one is to start with two raw point clouds, register and merge them to one point cloud, then use that output as input of next registration, the second one only use original input point clouds, register to find the transformations between them and merge them after all transformation have been found to produce the output cloud (figure 5).

The first method seems natural, however, it has some critical drawbacks:

1. As the process run, the input data grow in size, which slows all the consecutive computation. And also, the input data were changed at runtime. So

preprocessing step has to be re-do each iteration. Those two reasons make the processing time increase dramatically.

2. Overlap region between two inputs will be smaller and smaller, and vice versa with outlier, which will affect the correctness of register result. This problem can be omitted in most cases if users initialize the two inputs close to correct alignment, and using closest point matching in ICP.

The second method has an upper hand. The input data is original input data. So the speed is unchanged as well as overlap region between two point clouds. But both methods are sensitive to error because of the dependent characteristic of registration, one registration fail (large error in alignment) in half of overall process will result failure in the whole process. That's the case of large error. However, in the situation of small errors, it may be acceptable and usable for small errors occur in the first situation but in the second, the small errors can become significant. As we described, the second method computes the transformation between original input point clouds and merge them as final result after all transformation have been computed, if small errors occur in several dependence transformations, they accumulate.

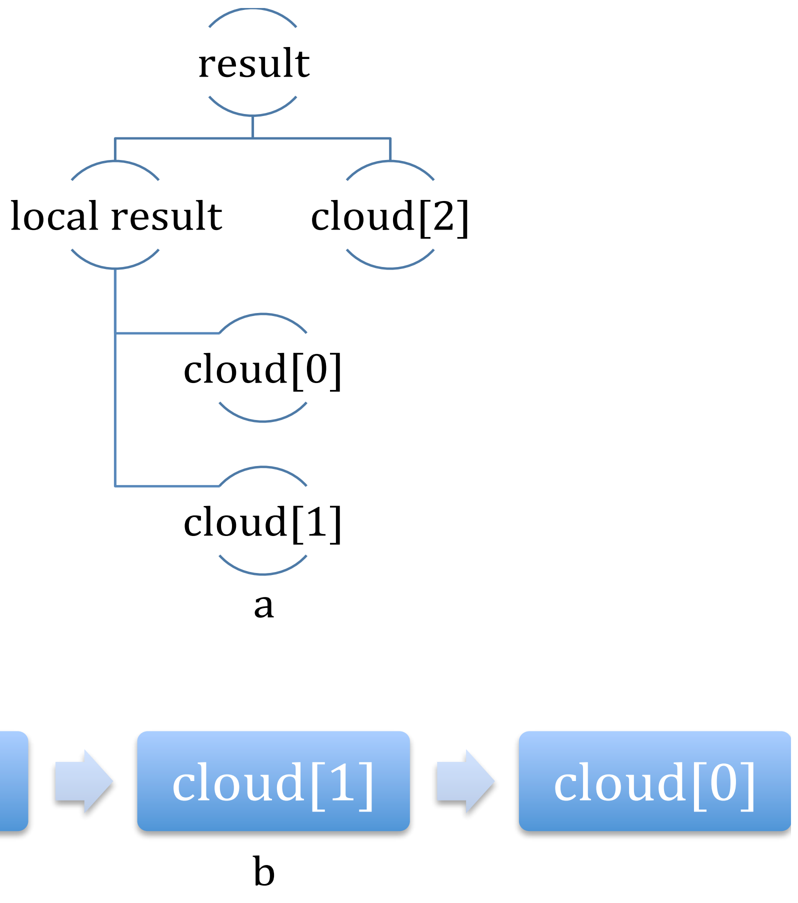


Figure 5. Two merging procedures

Figure 5 (a) shows the first method, cloud[0] and cloud[1] are registered and merged to local result then local result registered with cloud[2] to produce result. (b) shows the second method, cloud[1] is aligned to cloud[0], cloud[2] aligned to cloud[1], cloud[2] can later be aligned to cloud[0] by those two relationships.

Despite the fact that the second method is more sensitive to error, the first method computational cost is much more significant if the data set consists of about 20 or higher point clouds, each point cloud contain approximately 300000 points, which is normal for a small room data sets. For that reason, we choose the second method for our process.

To be more specific, the registration process is illustrated in figure 6. Consider a set of point clouds of multiple views, each time, two consecutive point clouds i and i+1 will be registered in pairwise way. Output will be a transformation from point cloud i+1 to i. And after registrations between all pairs, we have transformation matrices from next point cloud to previous one and we can transform all point cloud to the first point cloud by multiply those matrices (figure 5).

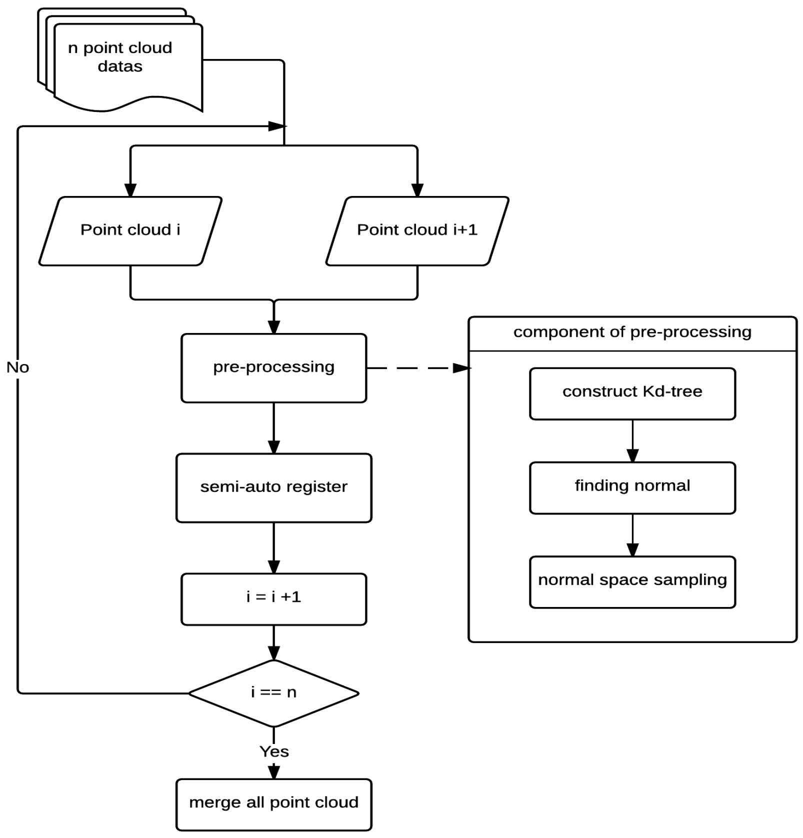


Figure 6. Pair-wise registration process

Chapter 4

RESULTS AND DISCUSSIONS

This chapter describes set up and results of experiments we performed to evaluate performance of our proposed method. It consists of a section to introduce data sets, an experiment to show the importance of different initial position, a comparison between our system with an automatic system and an experiment to test the performance of registering a set of point clouds.

4.1 Data test

We have prepared three different data sets for our experiment.

First data set is “room1” data set, which consists of 21 point clouds. It contains simple objects with smooth surfaces. It is showed in figure 7.

Second data set is “Tree” data set, which consists of 9 point clouds. “Tree” has random geometry with leaves. “Tree” data set is shown in figure 8.

Third data set is “Stair” data set, which consists of 5 point clouds. It is a hard data set for ICP to find a good result. Objects are all similar. So it is very hard for ICP to find the good correspondence and the error metric to penalize the wrong matches “Stair” data set is shown in figure 9.

All the data sets were archived by using Microsoft Kinect, each point cloud consist of approximately 300,000 points

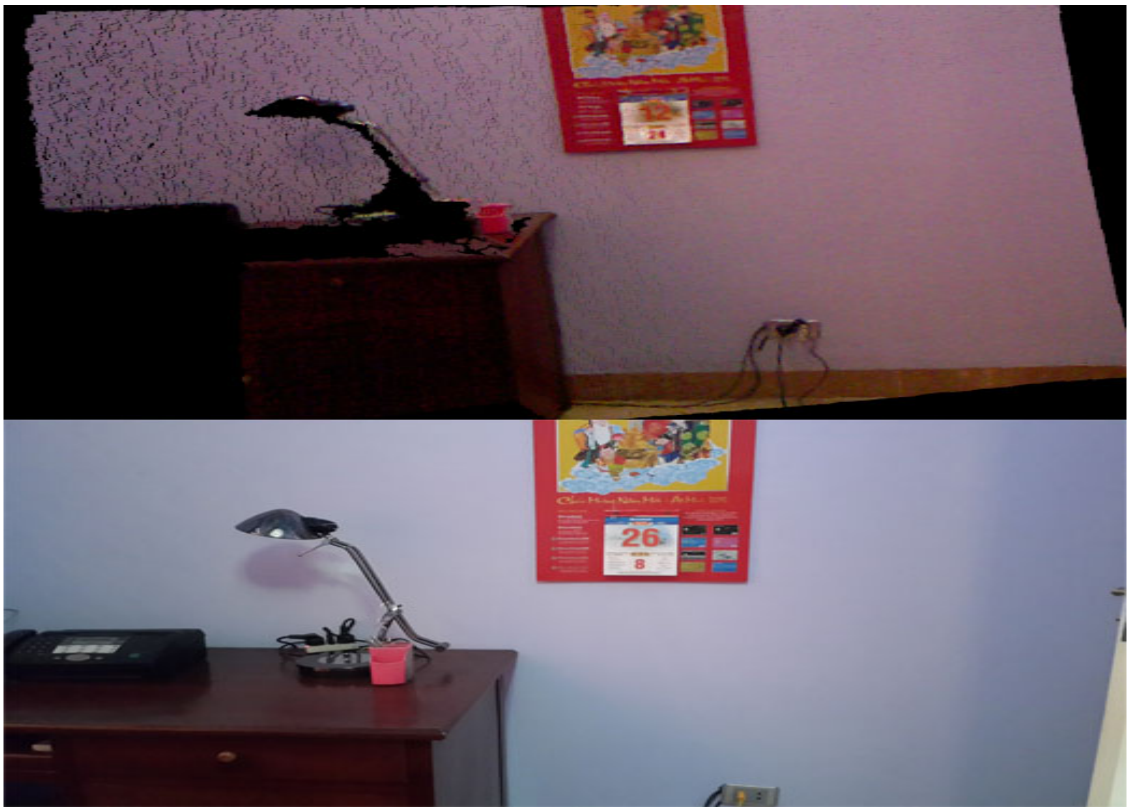


Figure 7. "Room1" data set. Upper image: 3D point cloud. Lower image: 2D image

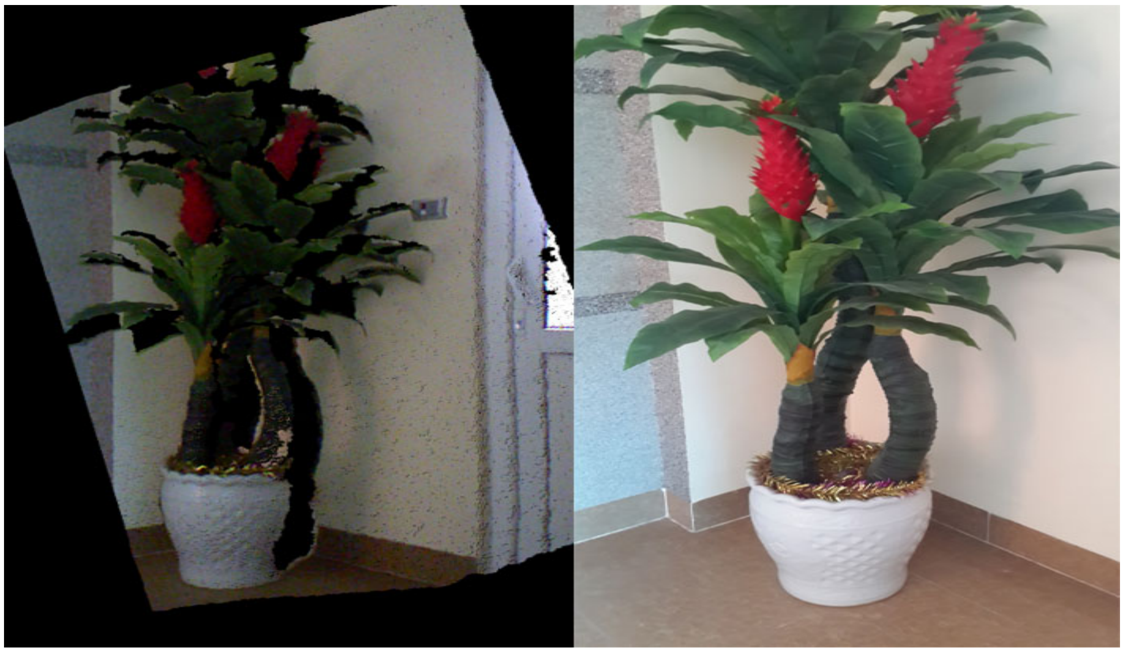


Figure 8. "Tree" data set. Left image: 3D point cloud. Right image: 2D image

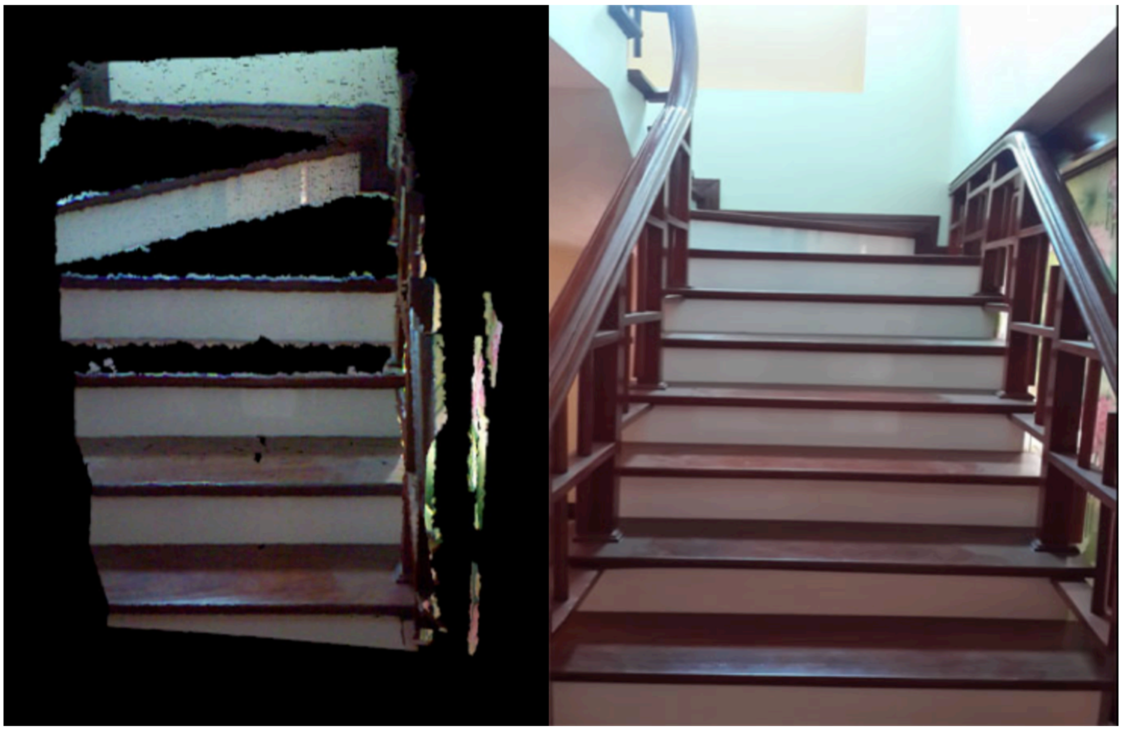


Figure 9. "Stair" data set. Left image: 3D point cloud. Right image: 2D image

4.2 The importance of initialization

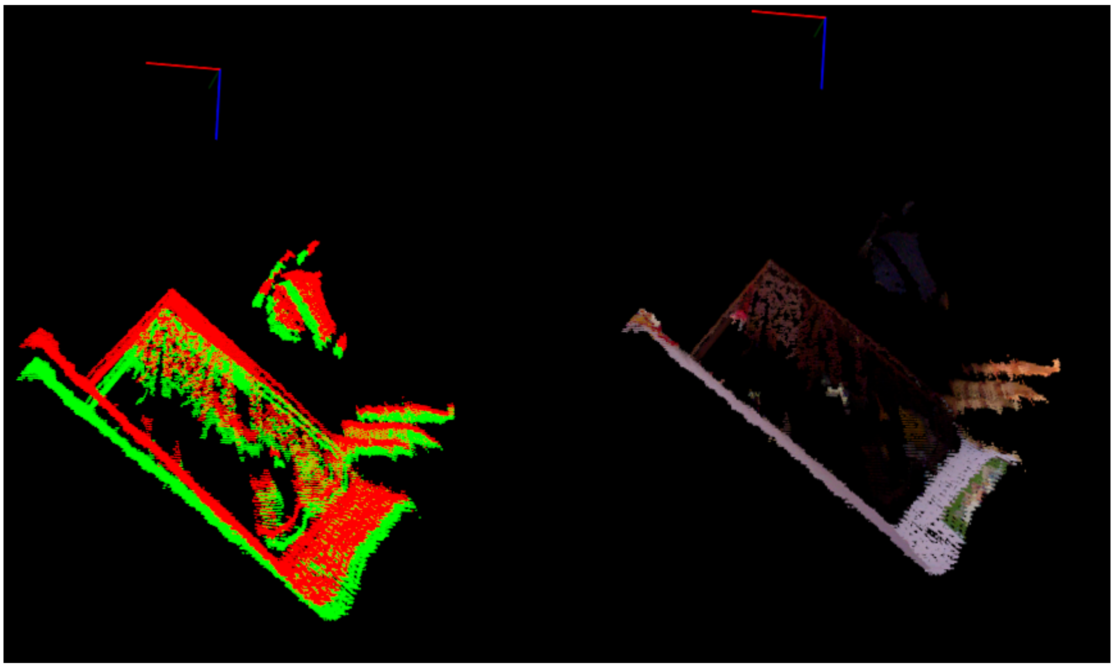
To illustrate the importance of initialization, we did a simple experiment. To measure a correct error of process, we take a point cloud with two different positions as two input clouds. So, the real point pairs are known. The error is estimated by root mean square (RVM) of distance between points in real pairs.

$$E = \sqrt{\frac{\sum_i (x_i - y_i)^2}{n}}$$

$$(x_i, y_i)\ is\ real\ point\ pair$$

$$n\ is\ number\ of\ point\ pairs$$

Figure 10, 12 and 14 illustrate two different initial positions given a different result. All three figures contain one initial position leads to good result (above) and one initial position lead to fail result (below). Also, figures 11, 13 and 15 illustrate RVM error for those scenes respectively. A small change will make the algorithm fail to find a result. So, with the option of custom the initial position in our approach, user can have the ability to find best initialization to compute correct solution for their data.



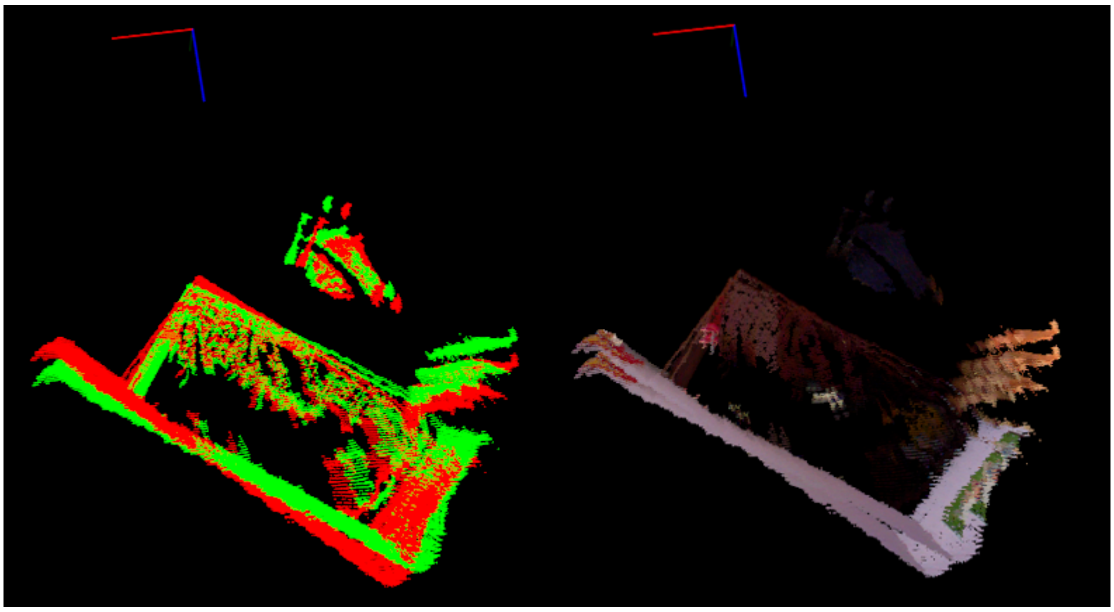


Figure 10. Two different initial positions. Upper image shows the initial position lead to good result, lower image shows the initial position lead to bad result. Left: initial position, right: result

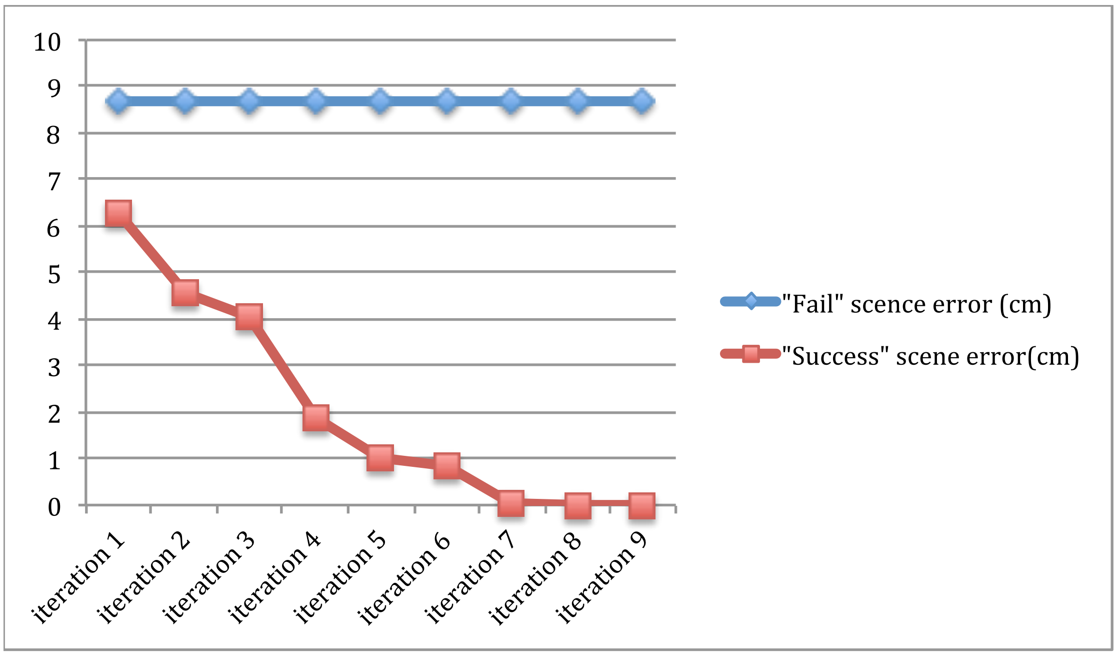


Figure 11. Errors in iterations of data set "room1"

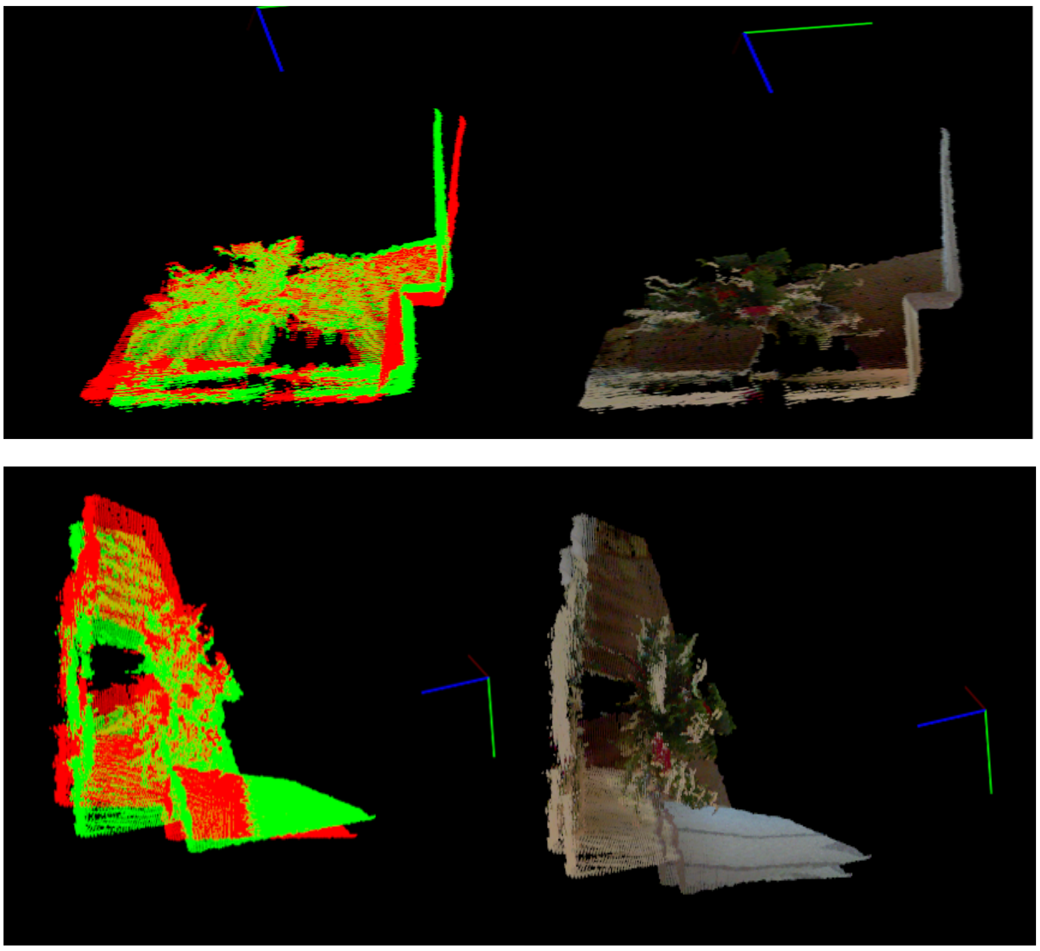


Figure 12. Two different initial positions in data set "tree". Upper image shows initial position leads to good result, lower image shows initial position leads to bad result. Left: initial position. Right: result

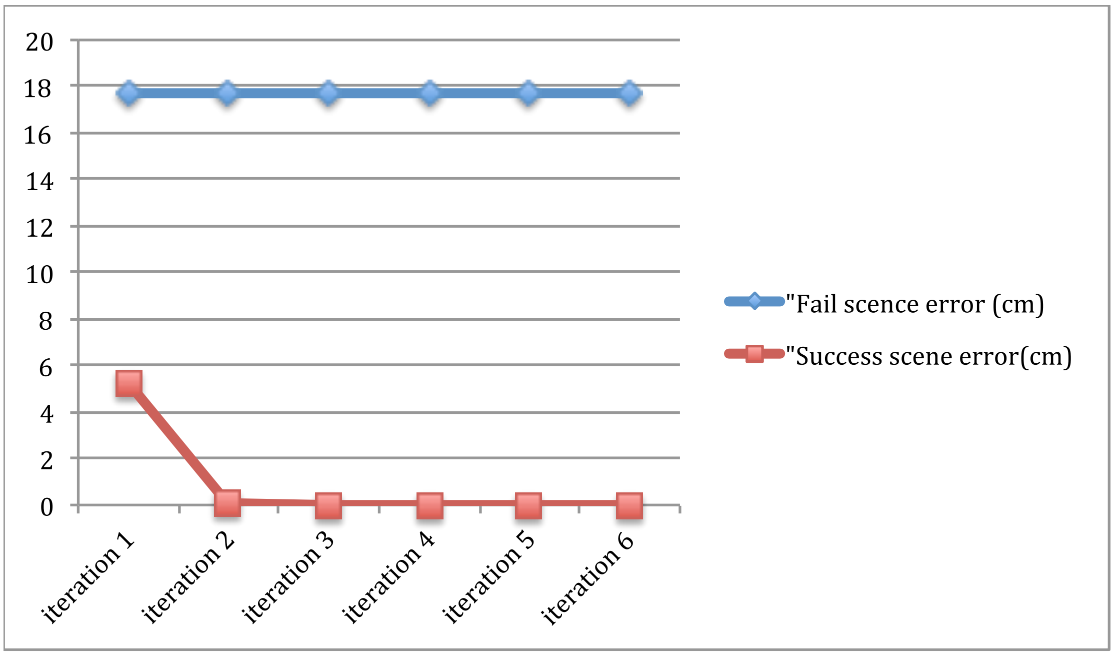


Figure 13. Errors in iterations of data set "tree"

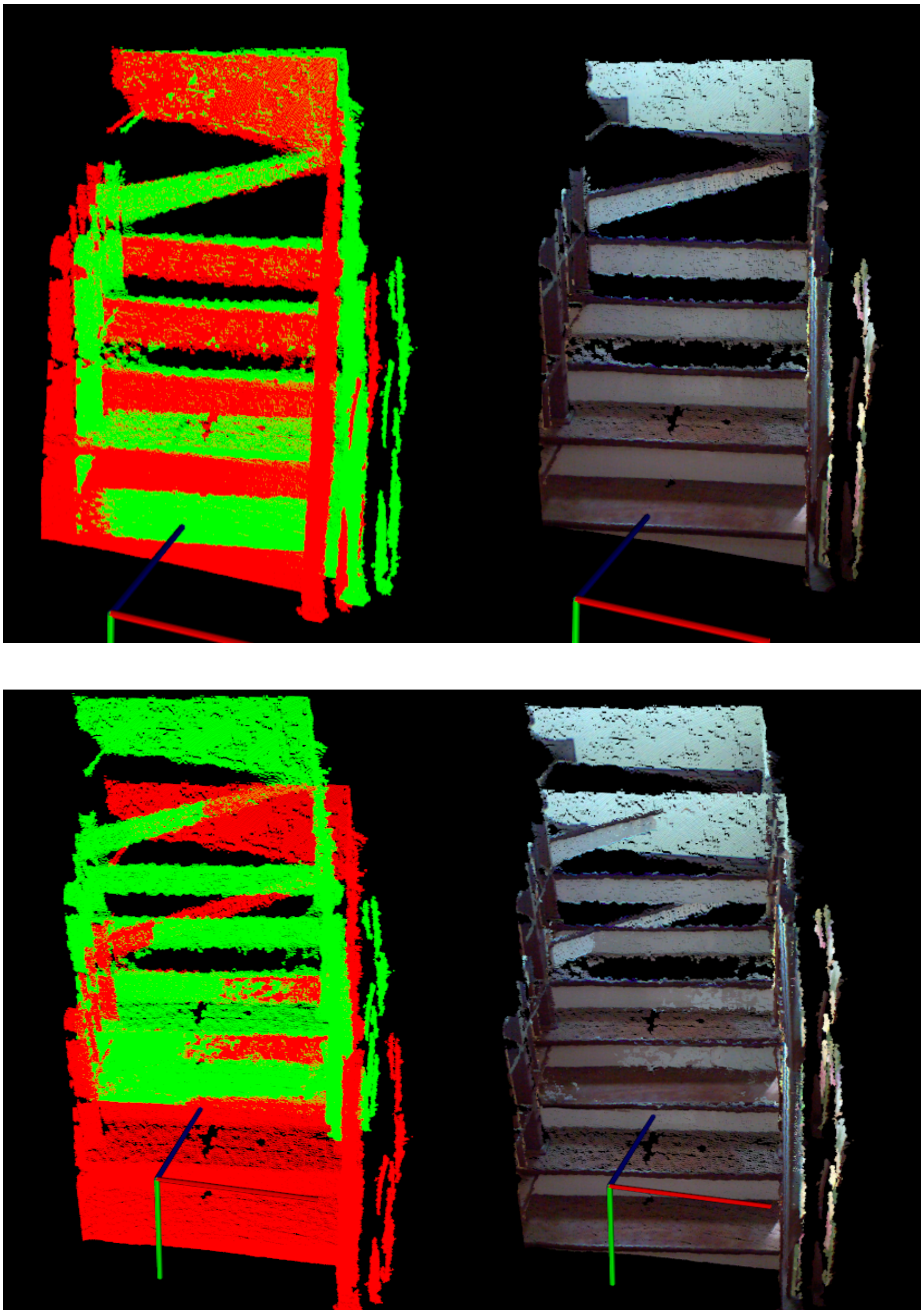


Figure 14. Two initial position of "stair" data. Upper image shows initial position leads to good result, lower image shows initial position leads to bad result. Left: initial position. Right: result

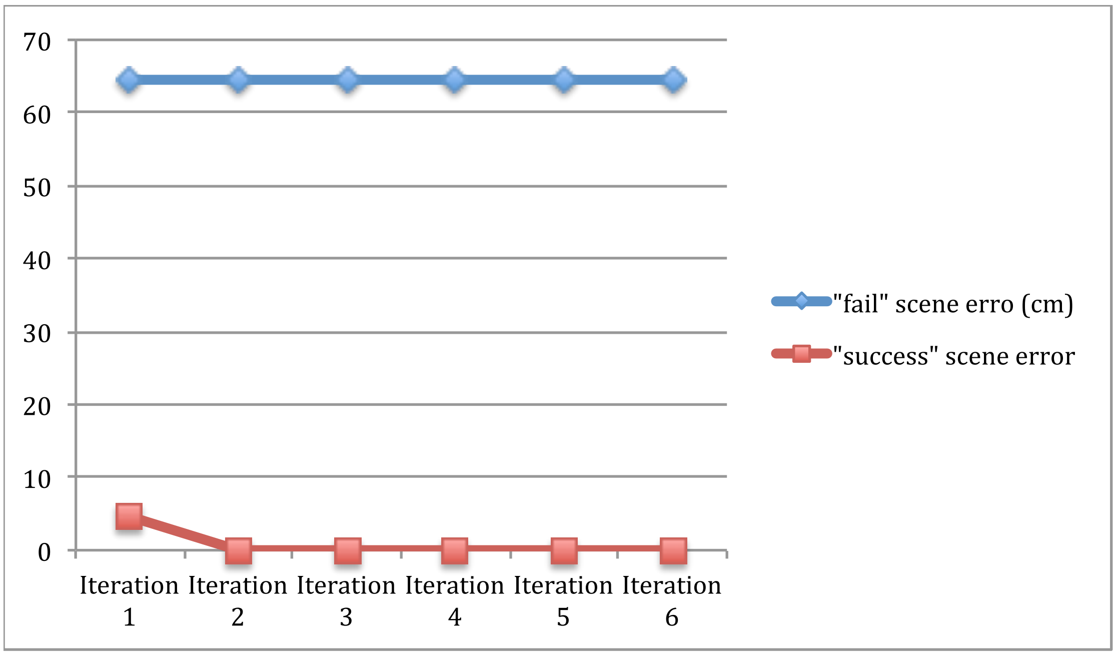


Figure 15. Errors in iterations of data set "stair"

4.3 Comparing with an auto-initial system

In this section, we compare our semi-automatic system with an automatic system.

4.3.1 Automatic system

The automatic system automates the initialization step then passes the result to GICP algorithm just like semi-automatic one. The automatic initializer is based on the process described in figure 16. A more detail description can be found at (Point Cloud Library Comunity)

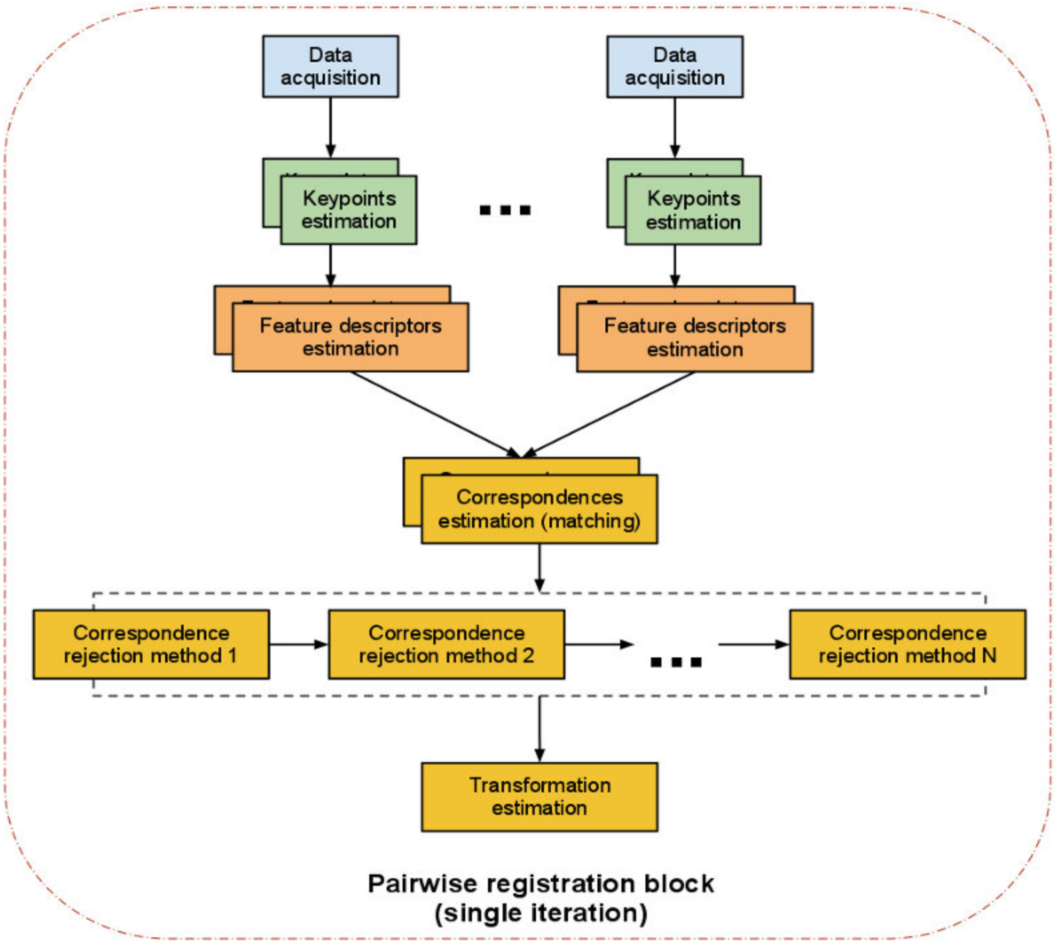

Data acquisition
Data acquisition
Keypoints estimation
Keypoints estimation
Feature descriptors estimation
Feature descriptors estimation
Correspondences estimation (matching)
Correspondence rejection method 1
Correspondence rejection method 2
Correspondence rejection method N
Transformation estimation
Pairwise registration block (single iteration)

Figure 16. Pairwise registration (Point Cloud Library Comunity)

To be more specific, we use SIFT (Lowe, 2004) to detect key points, FPFH (Radu, Nico, & Michael, 2009) as descriptors and Sample Consensus Initial Alignment (Radu, Nico, & Michael, 2009) for remaining steps.

4.3.2 Comparison

To compare two methods, we apply them to three registration situations, which are taken from three data sets in section 4.1.

Figure 17 shows the original position of two “room1” point clouds. One point cloud is colored in red and one point cloud is colored in green. The initial position is computed automatically or manually set. The point clouds on the left side of figure 18 are two initial positions. Figure 18 (a) is initialization of automatic system, (b) is initialization of semi-automatic system and the results of two initializations after passed to GICP are on the right side. The test case of “tree” data is shown in figure 19 and figure 20. “Stair” test case is shown in figure 21 and figure 22. The structured of those two test cases is similar with “room1” case.

In three cases, we can see that automatic initialization failed to find good initial positions for ICP; the “room1” data set has the best result, however, error is still observed around the calendar. In two others cases, the situation is much worse, especially in “stair” data, the relative position of two point clouds are further even comparing with raw position. In our semi-automatic system, the results after GICP steps of all three test cases are good.

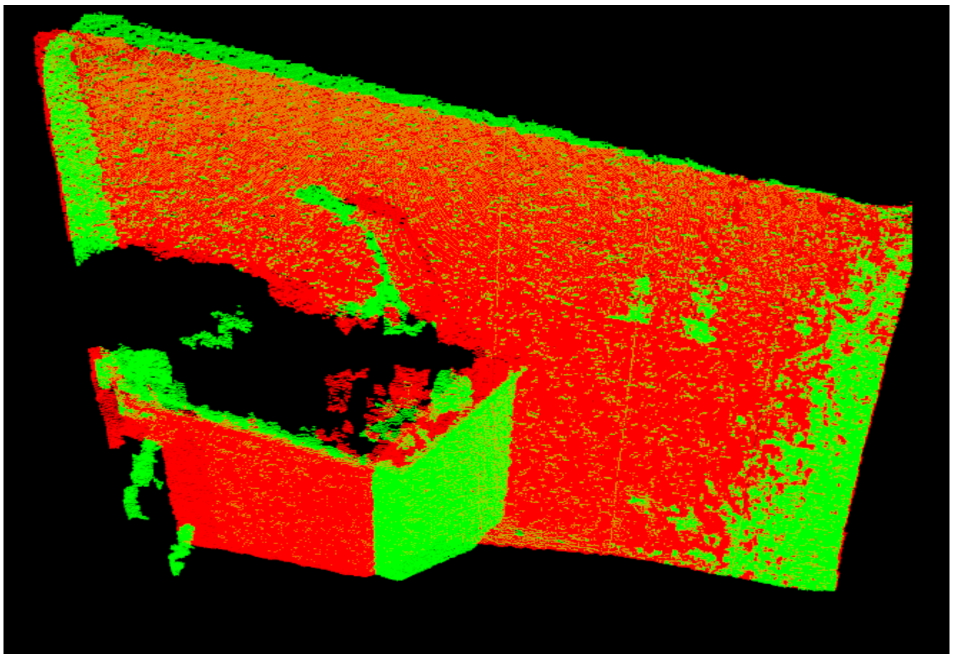
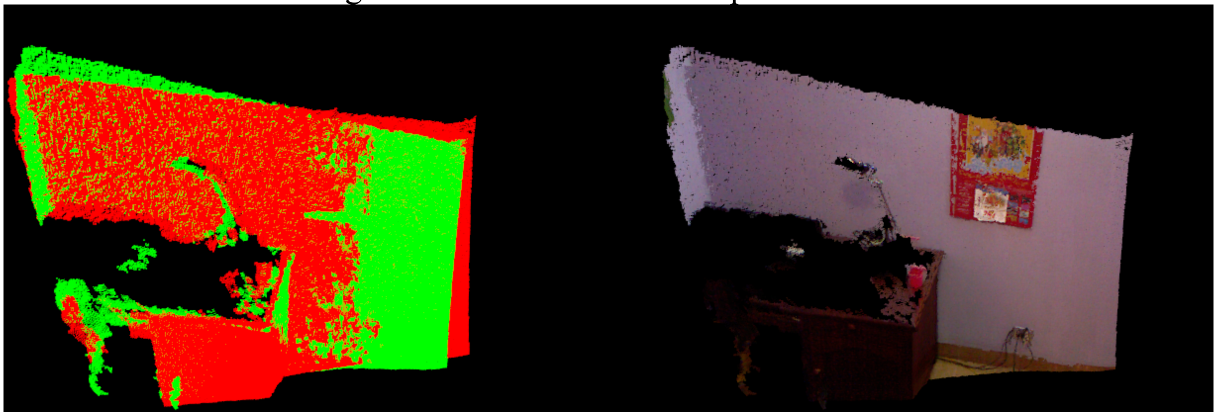
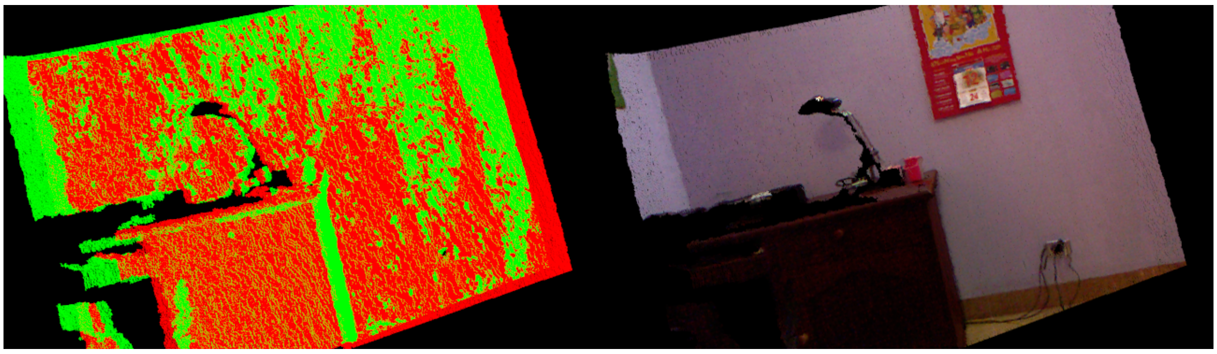


Figure 17. Two raw "room1" point clouds



a. automatic system



b. semi-automatic system

Figure 18. "tree" data (a) Left: automatic initialization. Right: result after ICP (b) Left: manual initialization. Right: result after ICP

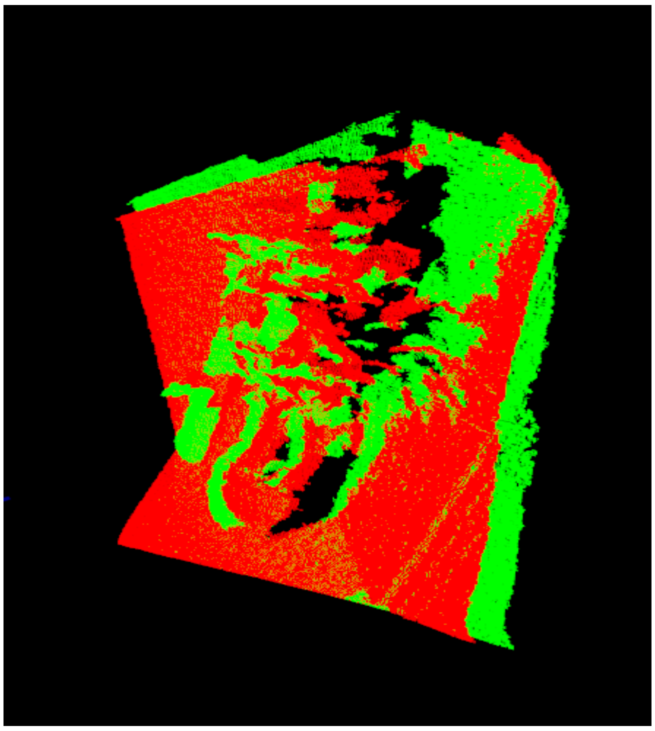
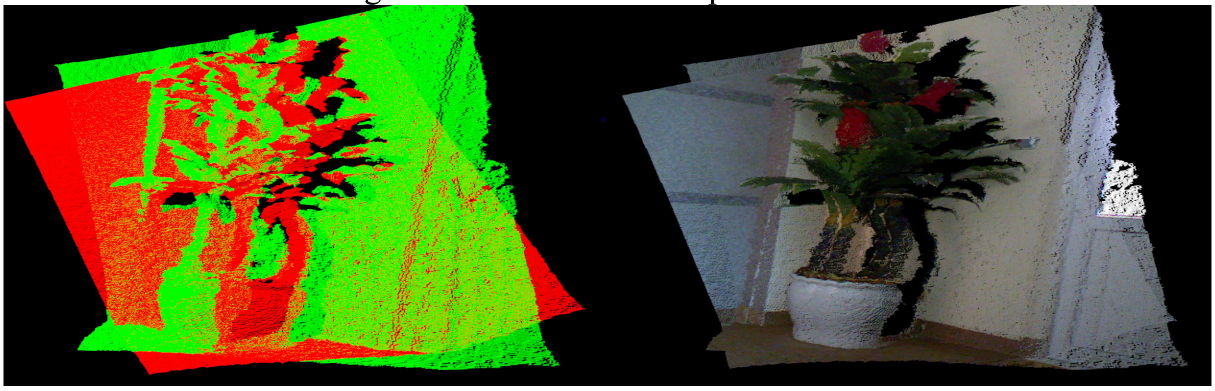
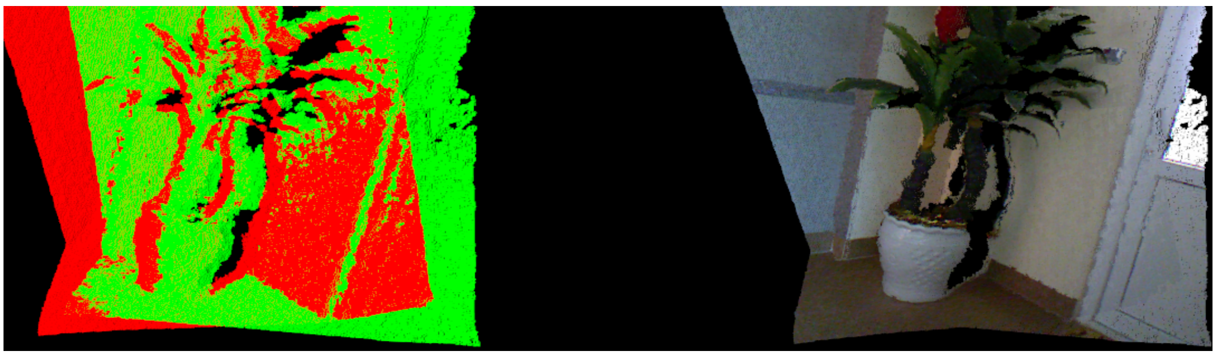


Figure 19. Two raw "tree" point clouds



a. automatic system



b. semi automatic system

Figure 20. "tree" data (a) Left: automatic initialization. Right: result after ICP (b)Left: manual initialization. Right: result after ICP

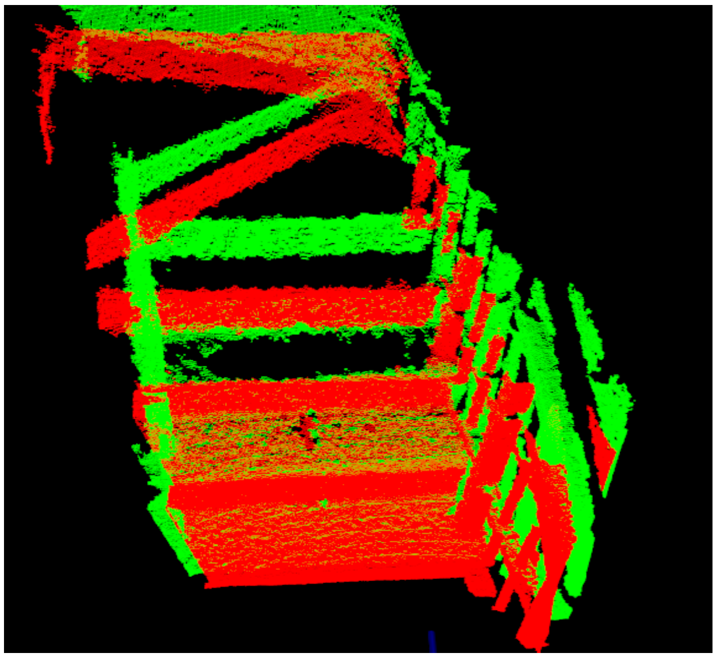
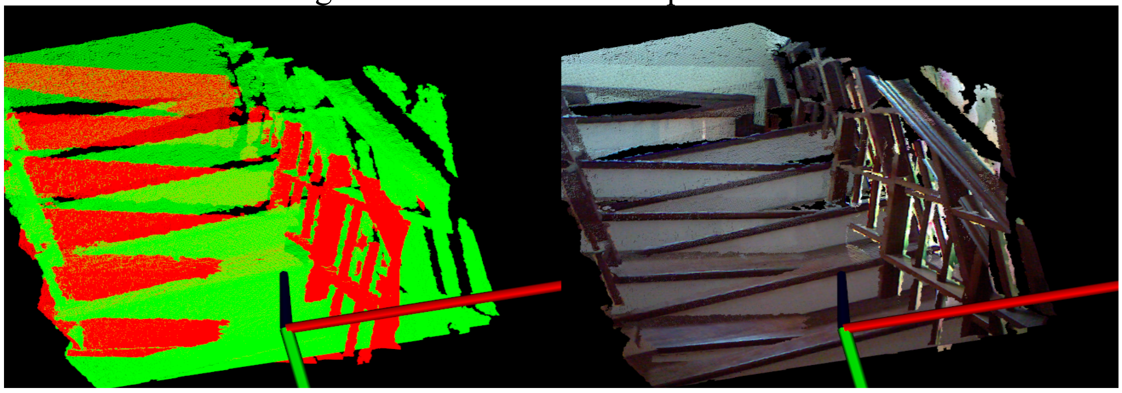


Figure 21. Two raw "stair" point clouds


a. automatic system

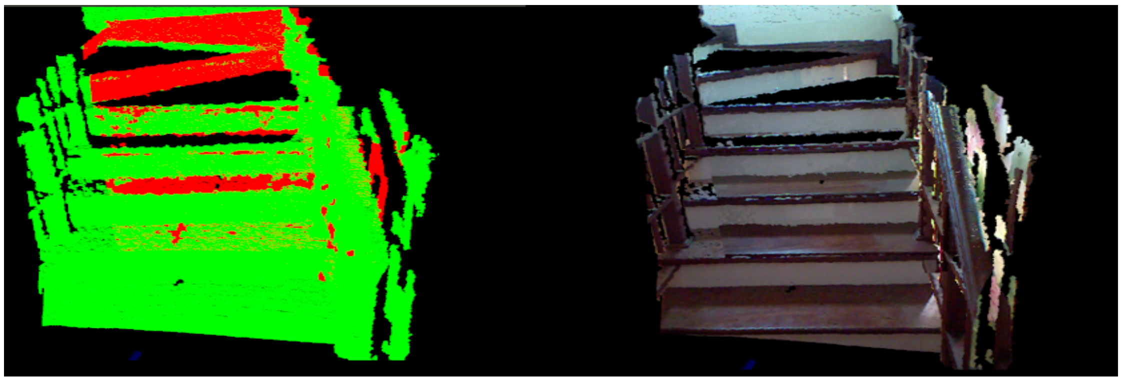

b. semi automatic system

Figure 22. "stair" data. (a) Left: automatic initialization. Right: result after ICP (b) Left: manual initialization Right: result after ICP

4.4 Testing with three sets of point clouds

In this section, we perform registration in three data sets to produce three combined results.

4.4.1 Room1 data set

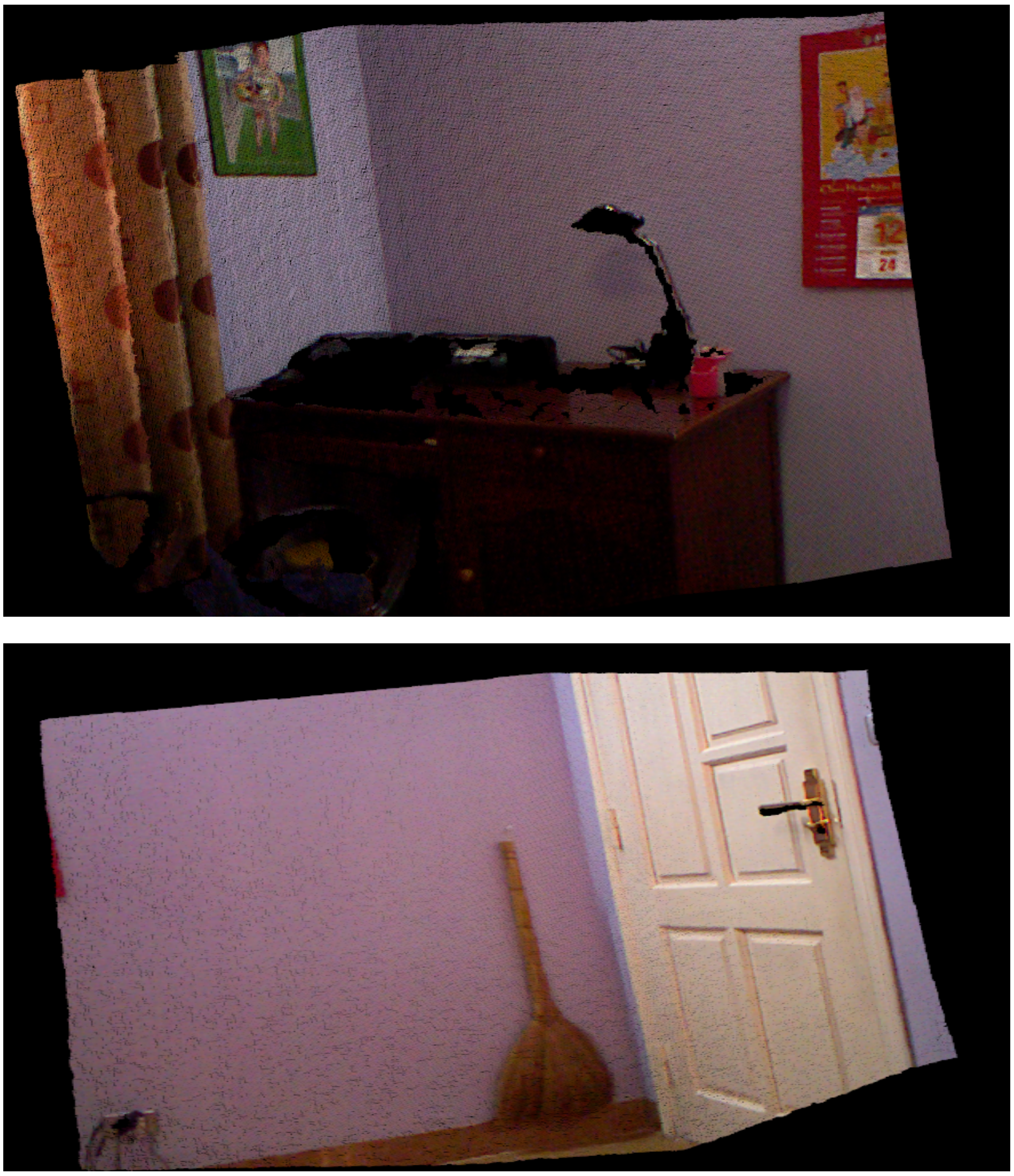
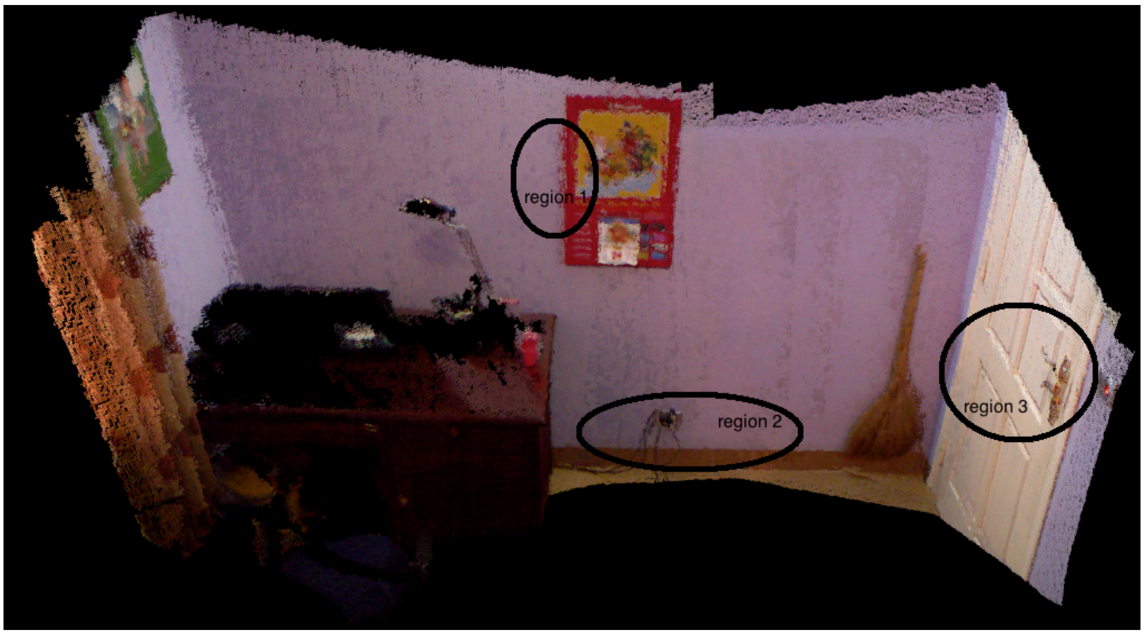


Figure 23. Two example point clouds



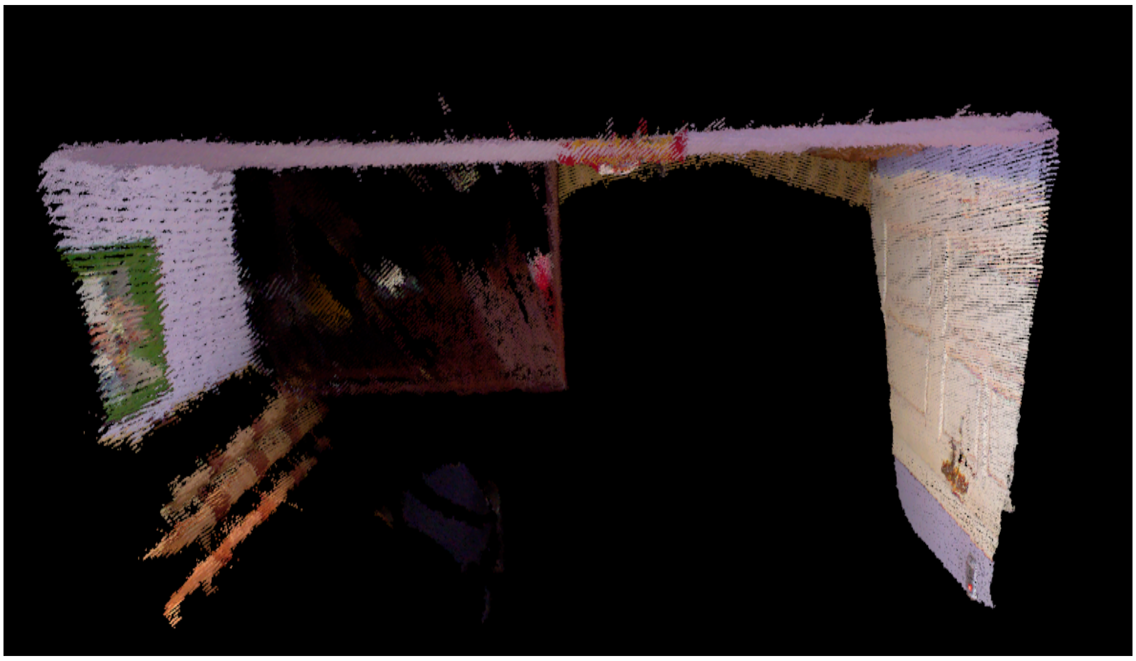


Figure 24. Two perspectives of result point cloud of data set "tree"

Figure 23 contains two examples of point clouds in two different angles of the actual scene and the result is shown in Figure 24. The first image in figure 24 is front view of total point cloud. The second image shows the top view of the point cloud.

The first image indicates a good result, the region 2 and region 3, lines are straight, small objects like the plug and the door holder can be clearly seen. However, in region 1, borders of the calendar contain errors. As stated in section 3.1, pair-wise method have a risk of accumulate errors, but the error only occur in region1, not in others, so we predict that, this is not caused by alignment error but by the calibration error. Calibration is the mapping between RGB-value and depth value of camera. The depth alignment is true but the color is not.

4.4.2 Tree data sets

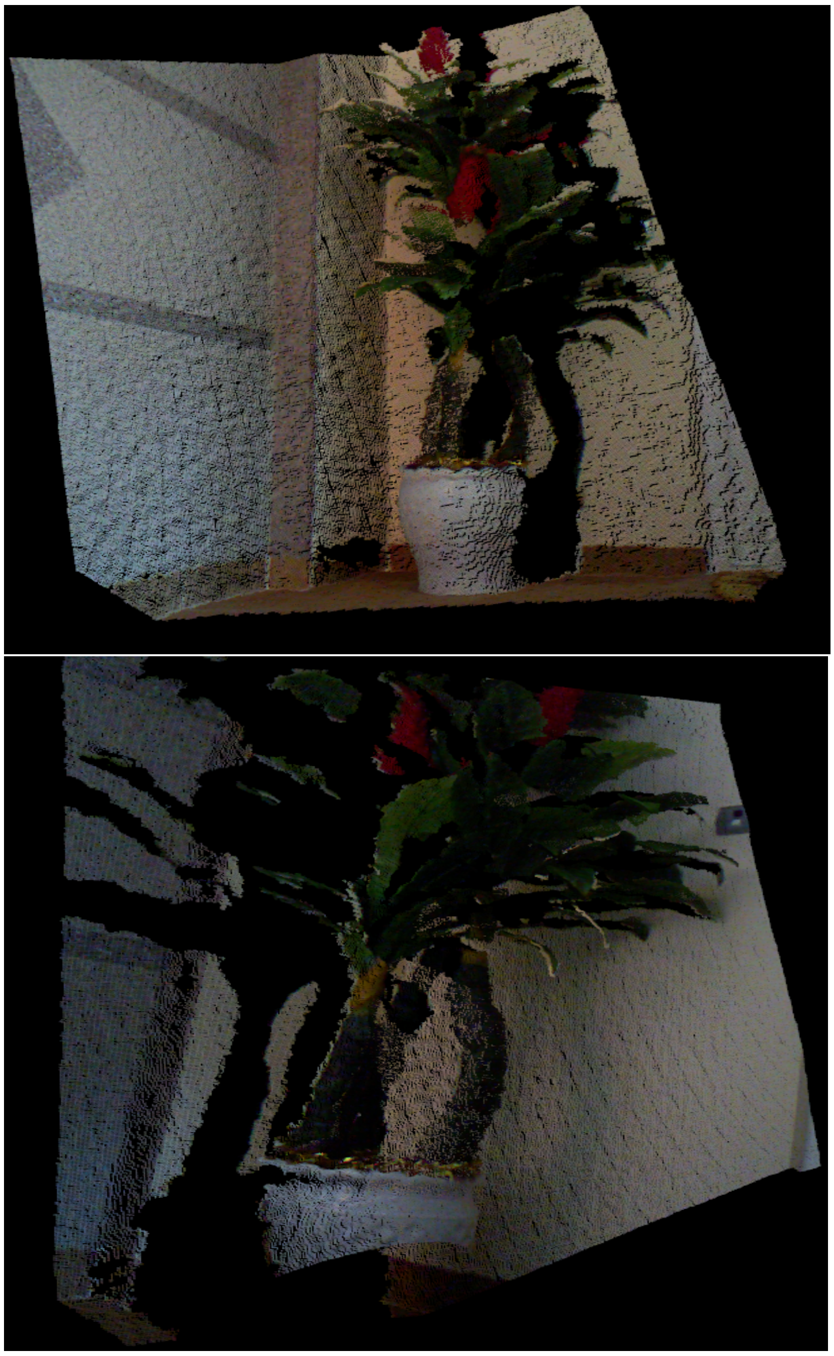


Figure 25. Two example point clouds

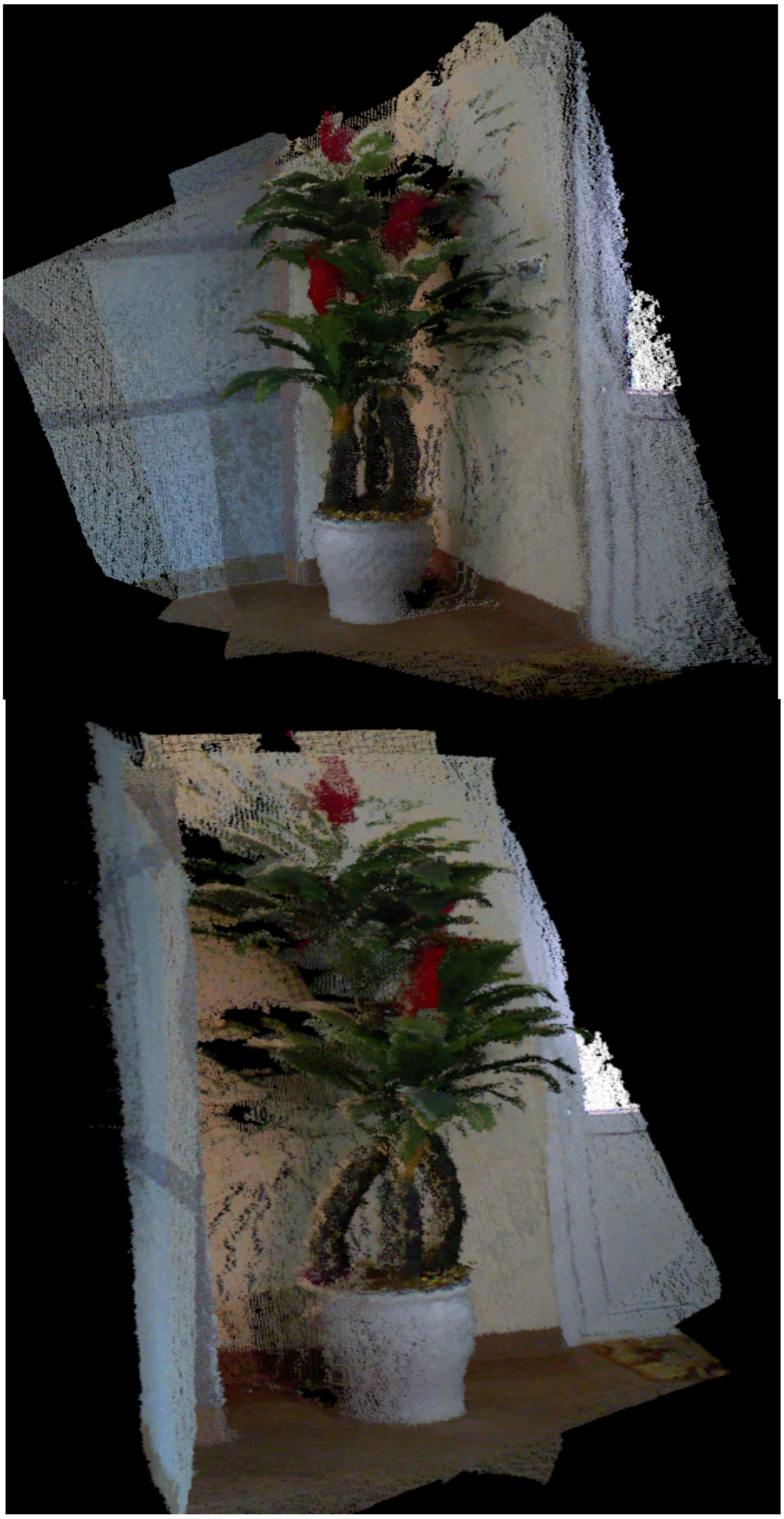
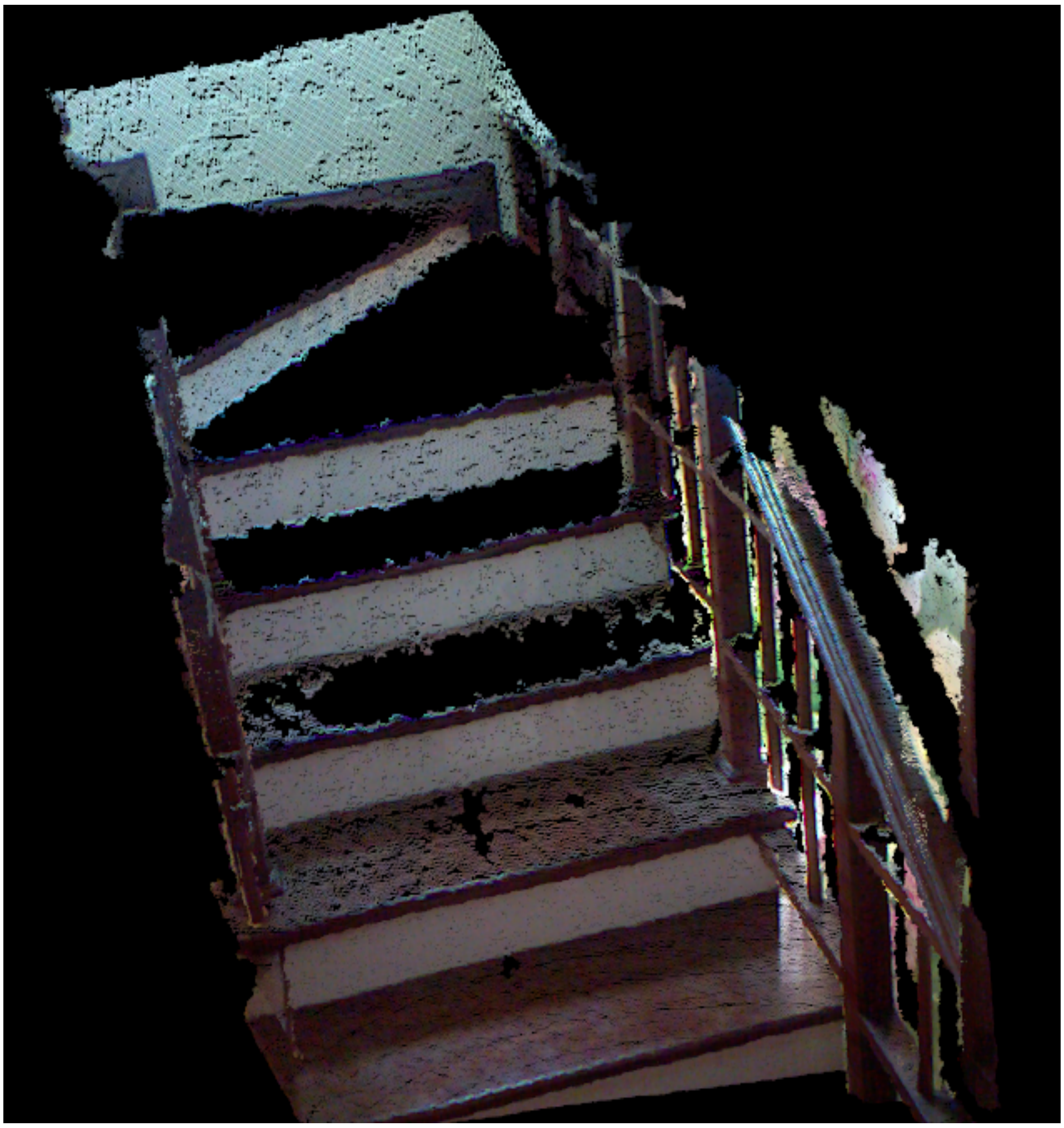


Figure 26. Two perspectives of result point cloud of data set "tree"

Figure 25 shows two example point clouds in different views before registration. Figure 26 shows two different perspective of “tree-result”. Same as “room1” result, the lines remain straight after registrations; the leave and the flowerpot are in place. The result does not have significant visual errors.

4.4.3 Stair data sets



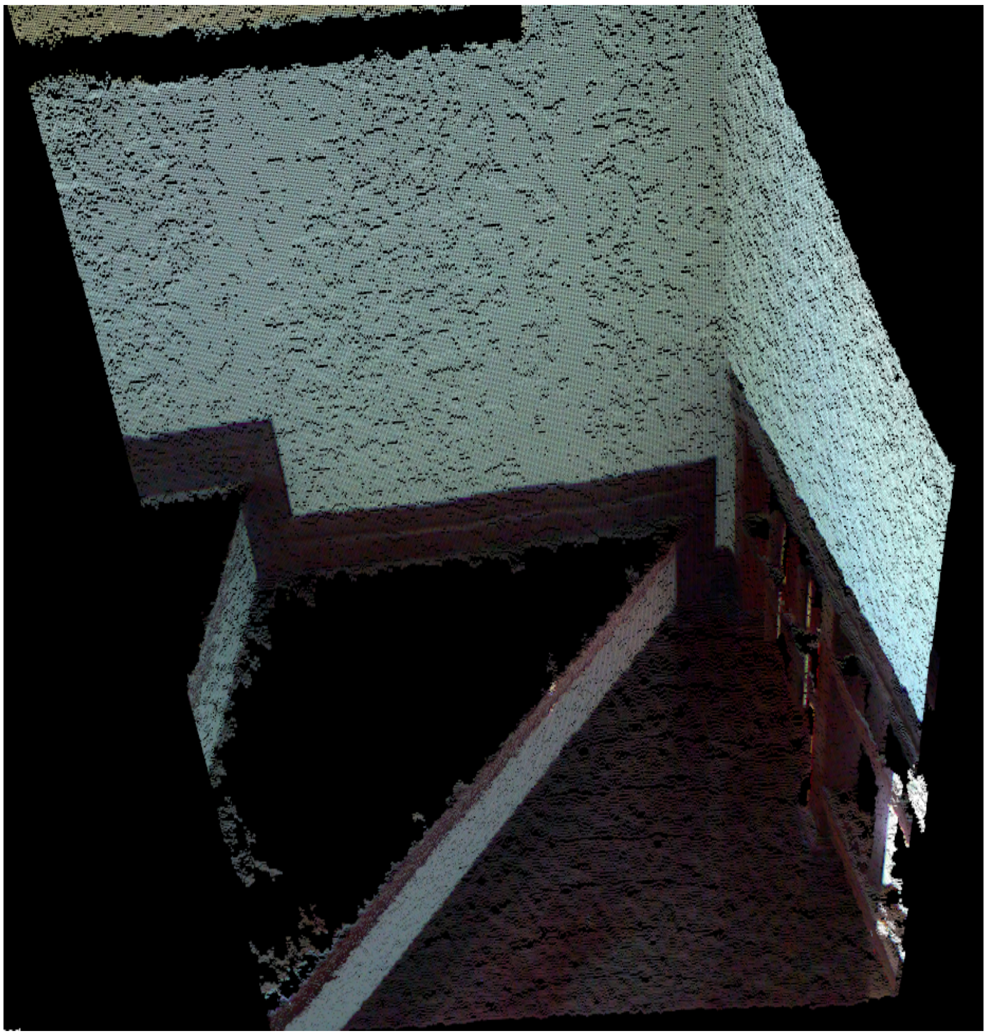
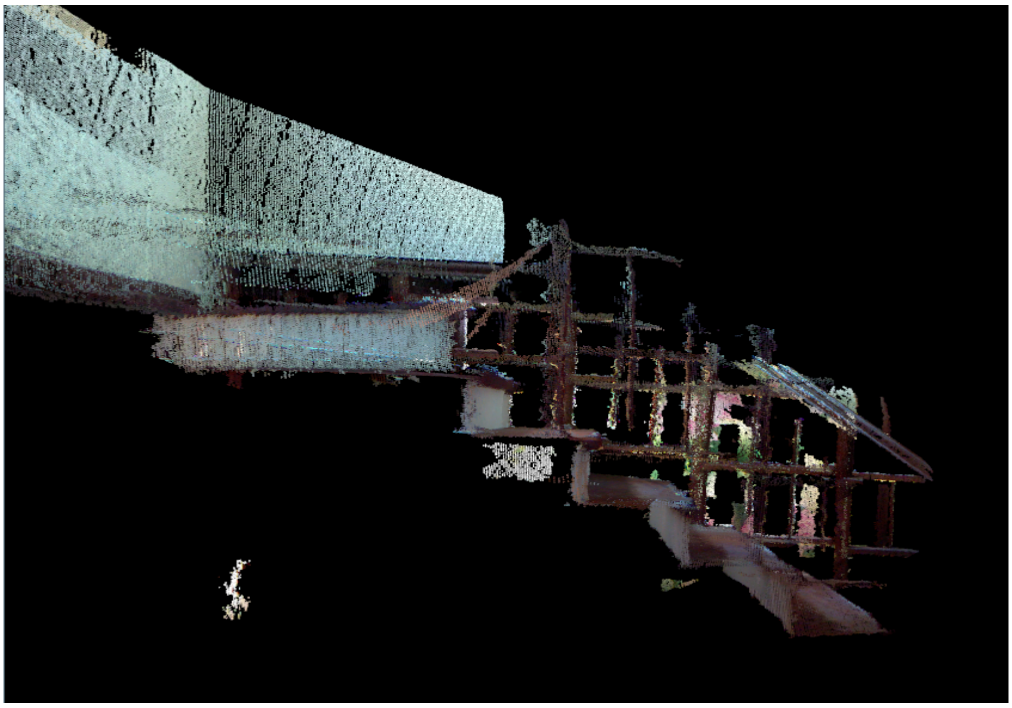
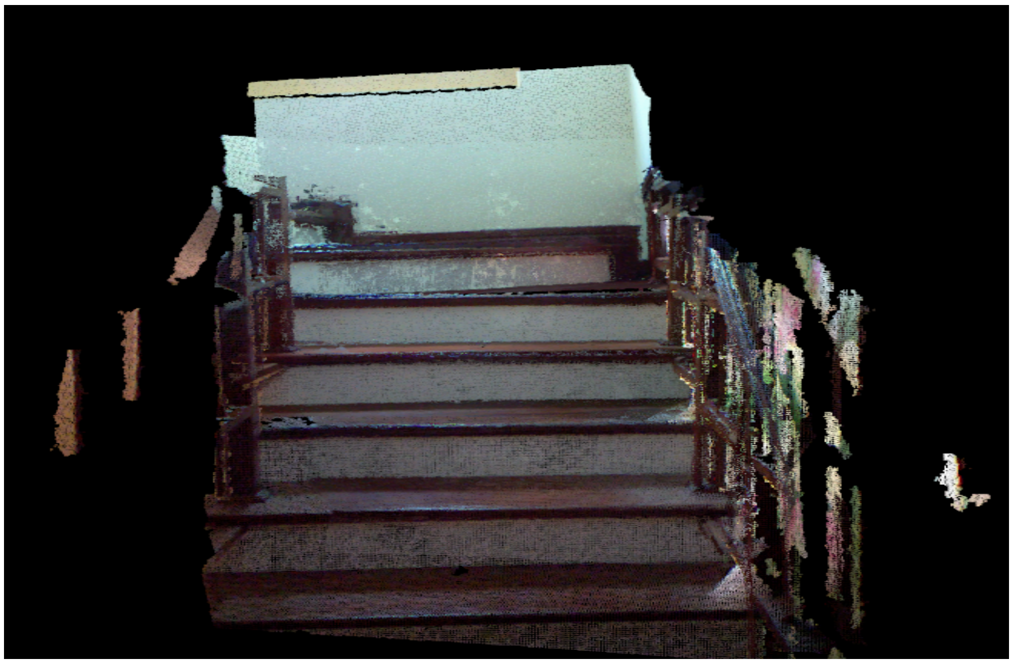


Figure 27. Two example point clouds





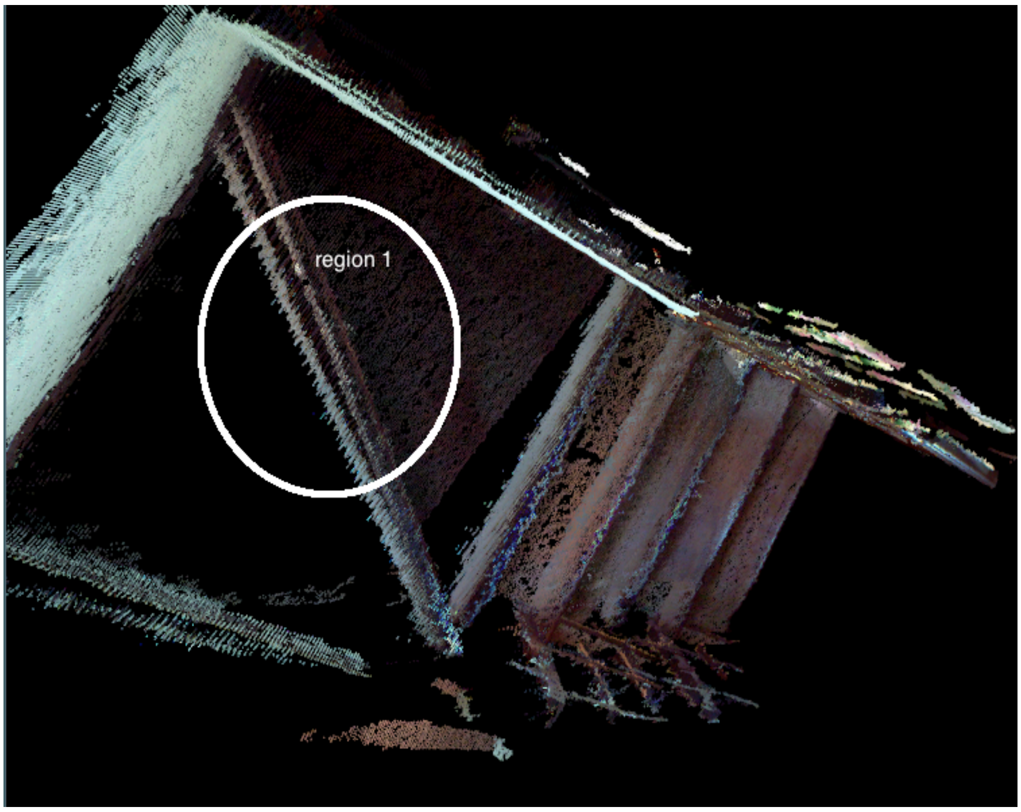


Figure 28. Three different perspectives of result point cloud of data set "stair"

Figure 27 shows two examples of point clouds in “stair” data set. From first two pictures in figure 28, the alignments of depth and horizontal coordinate are good (from front view). However, the third picture, in region 1, there is an error between the stairs. Stairs only contain smooth surfaces geometry, however, there are many points that have similar normal direction, which is used for weighting point pairs in generalized

ICP error metric. The point pairs that are fail matches but still are evaluated as true, the weighting is still big (GICP only penalize pairs with the incompatibility in normal direction) and out-numbered the pair needed to be align to give correct alignment (a small 2D example in figure 29). So for the stair data set or some data sets that similar pattern, they require a precise initialization. (Figure 30)

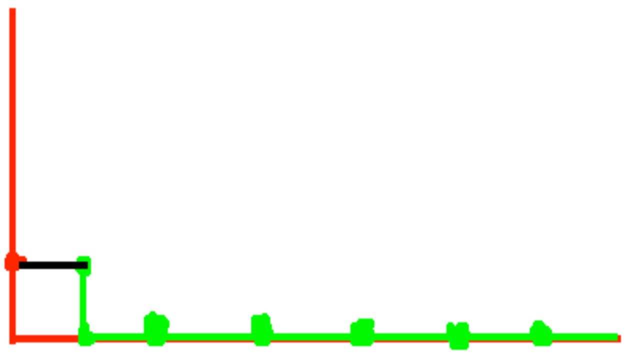


Figure 29. An example that shows the local minima trap

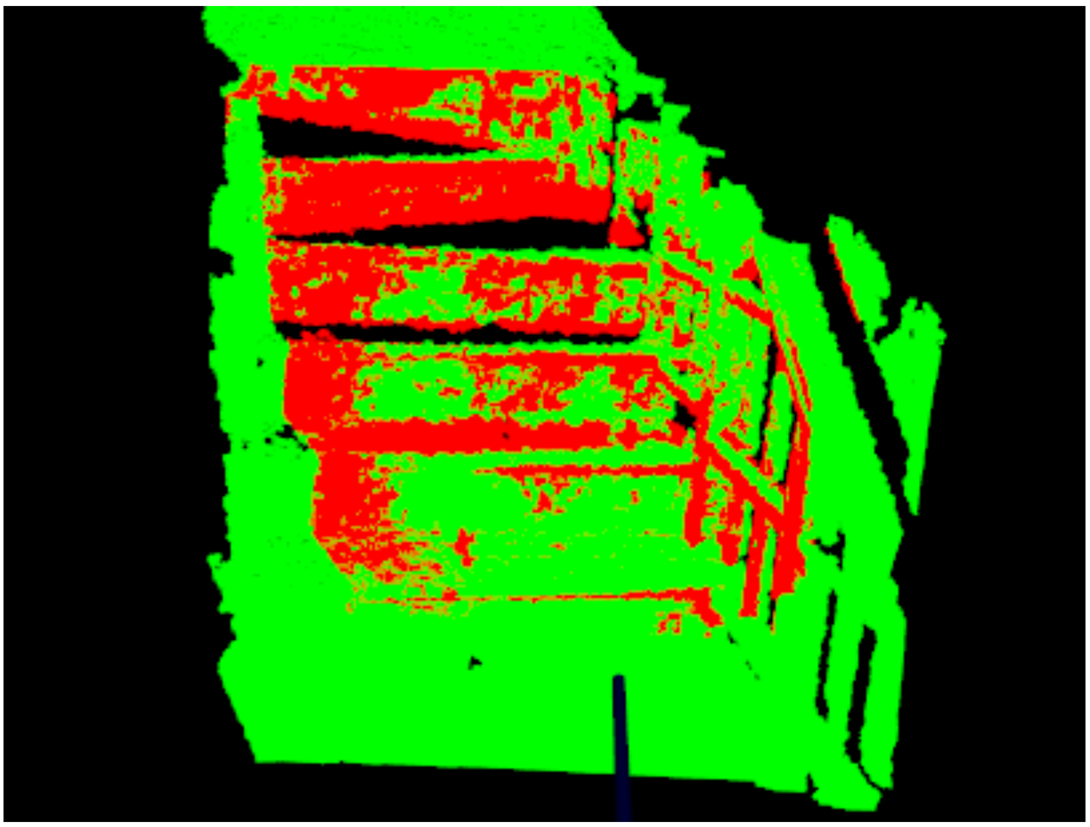


Figure 30. An example of initialization of "stair" data set

CONCLUSIONS AND FUTURE WORK

To sum up, this thesis examines the 3D registration problem base on the state-of-the-art ICP algorithm. We proposed a semi-automatic mechanism for creating initial positions of point clouds as the input for ICP algorithm and also a process for register a set of point clouds. We have tested our method with three different data sets; each contains a specific geometry pattern. The registration results of two data sets with smooth and free form surfaces are good. So, with small aid from human, we can have a registration application program for most general scene. E.g. constructing rooms or object models. However, in the experiment section, errors are also been observed. The method may not work well with situation point clouds with large portion of points have similar surface normal. In addition, the ICP in this thesis used closest point matching, which is fast with using KD-tree, but the only information it used is the distance between two points. So, in the future, we will try to find better methods that best adapt with GICP and generate more precise point pairs.

References

Aleksandr, S. V., Dirk, H., & Sebastian, T. (2009). Generalized-ICP. *Robotics: Science and Systems.*

Jon, L. B. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM* , 509-517.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International journal of computer vision 60.2* , 91-110.

Paul, B. J., & Neil, M. D. (1992). Method for registration of 3-D shapes. *Robotics-DL tentative* (pp. 586-606). International Society for Optics and Photonics.

Point Cloud Library Comunity. (n.d.). *The PCL Registration API.* Retrieved May 13, 2015, from pointclouds.org: http://pointclouds.org/documentation/tutorials/registration_api.php#registration-api

Radu, R. B., Nico, B., & Michael, B. (2009). Fast Point Feature Histograms (FPFH) for 3D Registration. *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on.* IEEE.

Shahram, I., David, K., Otmar, H., David, M., Richard, N., Pushmeet, K., et al. (2011). KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. *Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 559-568). ACM.

SVANTE, W., KIM, E., & PAUL, G. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, (pp. 37-52).

Szymon, R., & Marc, L. (2001). Efficient variants of the ICP algorithm. *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on* (pp. 145-152). IEEE.

Yang, C., & Gerard, M. (1991). Object modeling by registration of multiple range images. *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on* (pp. 2724-2729). IEEE.