Advanced Vision Practical

Reconstruction of office from a set of 3D point clouds

s1451142, s1459898

March 18, 2019

1 Introduction

In this paper we reconstructed an office room from a set of 3D point clouds acquired from an Intel RealSense depth sensor that is moved to scan different views. The core of this paper is to estimate how the 3D points captured in each frame can be transformed into points in one selected frame. Once all points are transformed, we can see a complete view of the room.

Data set:
Our data comprises of $N = 40$ frames of 3D point cloud data, $\{P_1\ ,\ \ldots\ ,\ P_{40}\}$, where P_i is a 3D point cloud at frame i, for $i = 1, \ldots, N$. Each frame is from a different viewpoint. The data is acquired by an Intel RealSense depth sensor. The data is stored in a cell structure with N cells, each with 2 entries:

- P.Color, the RGB value of each point, and
- P.Location, the XYZ location of each point, where X and Y coordinates horizontally and vertically, and Z going away from the camera.

The rest of the article is organised as follow. Section 2 describes and explains the methods used for each stage of the process to reconstruct the room tin details. Section 3 provides a brief discussion of the results and factors that influence the performance of the algorithm.

2 Algorithm

In this section we describe reconstruction systems based on point clouds from laser scans. The list of thresholds values involved in each of the approaches are listed in the Table 1.

2.1 Extract the relevant data from each point cloud

The raw point clouds are noisy and contains outliers: 1) irrelevant points outside the window; 2)a person in one frame; 3) effective depth data due to specular reflections; 4) deformed data near the

PROCESS	*THRESHOLD VALUE*
Task 3.1: Extract the relevant data from each point cloud	
Irrelevant points outside the window	$z_{thresh} = 3.5$
Defective depth data (flying pixel)	$l = 30$ (number of nearest neighbours
	$Distance\_threshold = 0.05$
Deformed data near the edge	$Edge\_threshold = 6$
Task 3.2: Estimating the reference frame transformation between consecutive frames	
Extracting and matching interest points using SIFT desciptors	$Peak\_threshold = 0$
	$Match\_threshold = 5$
Estimating reference 3D transformation using RANSAC	$k = 1000$ (number of iterations)
	$m = 25$ (number of sample points)
	$I_{min} = 50$ (initial minimum inlier count required)
	$I_{decay} = 0.9$ (inlier count decay rate)
	$d_{thresh} = 0.01$
Task 3.3: Build as complete a 3D model as possible and characterise the surfaces	
Identification of reconstructing the wall	Maximum distance between inliers and plane = 0.3
	Maximum absolute angular distance = 5°

Table 1: Main thresholds involved in the developed approach and default values.

edges of the image and 5) noisy ripples in the data. Therefore, the point clouds are preprocessed to remove noise from the point cloud.

2.2 Estimate the reference transformation linking each consecutive pair of frames

Extracting and matching interest points using `SIFT` descriptors
The determination of correspondences is typically a challenging aspect of the overall registration process, which directly affects the quality of the estimated transformation parameters and has the potential to cause divergence of the optimization algorithm. The corresponding points in the two point clouds are determined using `SIFT` (Lowe, 1999). Once the features are extracted at the keypoints, they are matched across the two point clouds to find the corresponding points. We used the implementation of the `VL Feat` (Vedaldi & Fulkerson, 2010) library for both extracting and matching the interest points

Estimating 3D rigid transformations using SVD of a matrix
Once the corresponding points are obtained, least square algorithms are used to calculate the 3D transformation. Two corresponded point sets $\{t_i\}$ and $\{s_i\}$, where each element is a 3×1 vector, i.e. $t_i, s_i \in \mathbb{R}^3$ for $i = 1, \dots M$ are related by

$$t_i = Rs_i + T + \epsilon_i, \tag{1}$$

where, $R \in \mathbb{R}^{3\times3}$ is a rotation matrix, $T \in \mathbb{R}^3$ is a translation vector and $\epsilon_i \in \mathbb{R}^3$ is a noise vector for each point sets (Arun et al., 1987). The correlation matrix, $H \in \mathbb{R}^{3\times3}$, between the two

corresponded point sets is computed using the centered corresponded point set,

$$\begin{aligned} \bar{t} = \frac{1}{M}\sum_{i=1}^{M} t_i, \qquad \bar{s} = \frac{1}{M}\sum_{i=1}^{M} s_i, \\ H = \sum_{i=1}^{M} \bar{s}_i \bar{t}_i^T. \end{aligned} \tag{2}$$

We want to find the optimal transformation model that maps the set $\{s_i\}$ to $\{m_i\}$, which minimise the least squares error criterion:

$$\Sigma^2 = \sum_{i=1}^{M} \parallel t_i - (Rs_i + T) \parallel^2 . \tag{3}$$

The optimal rotation matrix, $\hat{R}$ and translation, $\hat{T}$ can be found using the singular value decomposition (SVD) of H, that is:

$$\begin{aligned} \text{SVD}(H) &= U\Lambda V^T, \\ \hat{R} &= VU^T, \\ \hat{T} &= \bar{t} - \hat{R}\bar{s}, \end{aligned} \tag{4}$$

where, U and V are $\mathbb{R}^{3\times3}$ orthonormal matrix and Λ is a $\mathbb{R}^3$ diagnonal matrix (Lorusso et al., 1995; Arun et al., 1987).

Estimating reference 3D transformation using `RANSAC`

We could estimate the rotation and translation by using all corresponding points found above. However,it is possible that the points get matched incorrectly due to noise in the scanning, overlapping shapes only over parts of their extent, inaccuracies in the previous stages and etc. `RANSAC` algorithm is more robust to noise (Fischler & Bolles, 1981). `RANSAC` algorithm randomly selects m data point pairs from $\{t_i\}$ and $\{s_i\}$, which are used to estimate the 3D transformation model, i.e. a translation vector and 3D rotation matrix that links the two point clouds. 3D Transformation model is then applied to the source point in each of the corresponding points. The euclidean distance between the target point, $\{t_i\}$, and the transformed source point for each of the point pairs, $\{\tilde{s}_i\}$, is computed. The estimated 3D transformation model is validated by checking the number of correct matches (inliers) falls within a distance threshold. Then, the final optimal 3D transformation model, $(\hat{R}, \hat{T})$, which gives the maximum number of inliers is calculated. A pseudo-code of out Matlab implementation of `RANSAC` is presented in Algorithm 1.

In RANSAC, we iterate until there are enough number of inlier counts which is defined by I_{min}. I_{min} is scheduled to decay with a rate of I_{decay} when it did not manage to get the required number of inlier counts in the given amount of iterations, k. These stopping criterion will ensures that a good transformation model, with less computation time needed as compared with the normal RANSAC algorithm.

2.3 Fuse all of the points into a single 3D coordinate system

We represent both rotation, R and translation, T matrices in one 4×4 transformation matrix, A.

Algorithm 1 *RANSAC*

Input:
$\{t_i\}$: Target 3D SIFT Points
$\{s_i\}$: Source 3D SIFT Points
Hyperparameters:
k: Number of iterations
m: Number of samples
I_{min}: Minimum number of inliers required
I_{decay}: Inlier count decay rate
d_{thresh}: Distance threshold to be considered as an inlier.
Output: T_{best}, R_{best}
Initialize: $Loop\_count = 0, inlier\_count = 0$
repeat
  $\{t'_i\}, \{s'_i\} \leftarrow Sample\ m\ points\ from\ \{t_i\}, \{s_i\}$
  $T, R \leftarrow 3D\_Rigid\_Estimation(\{t'_i\}, \{s'_i\})$
  $\{\tilde{s}_i\} \leftarrow 3D\_Transformation(\{s_i\}, T, R)$
  $d \leftarrow euclidean\_distance(\{\tilde{s}_i\}, \{t_i\})$
  $count \leftarrow count(d \leq d_{thresh})$

  if $count \geq inlier\_count$ **then**
    $inlier\_count \leftarrow count$
    $T_{best}, R_{best} \leftarrow T, R$
  end if

  $Loop\_count \leftarrow Loop\_count + 1$
  if $Loop\_count \geq k$ **then**
    $I_{min} \leftarrow I_{min} * I_{decay}$
    $Loop\_count \leftarrow 0$
  end if
until $inlier\_count \geq I_{min}$

return T_{best}, R_{best}

$$R = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix},\ T = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix} \longrightarrow A = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, we have the following equation:

$$A_{i,j} P_j = \tilde{P}_{i,j}, \tag{5}$$

where $P_j \in \mathbb{R}^{4\times \mathrm{M}}$ is a point cloud at frame j with each column has the value $[x, y, z, 1]^T$, $\tilde{P}_{i,j} \in \mathbb{R}^{4\times \mathrm{M}}$ is a transformed point cloud ,and $A_{i,j} \in \mathbb{R}^{4\times 4}$ is the transformation matrix that performs the mapping on points P_j from frame j to frame i.

Once we have the reference frame transformation matrix between each consecutive pair of frames, $\{A_{1,2}, A_{2,3}, \ldots, A_{39,40}\}$, where $A_{i,j}$ indicates mapping points in frame j to frame i, then we can

map all the points from each frame into the initial frame since $A_{j,j+2} = A_{j,j+1} * A_{j+1,j+2}$. The algorithm for merging 40 frames into one single 3D coordinate system is given in Algorithm 2. The next section we will talk about how we will be evaluating the reconstruction of the room.

Algorithm 2 Merge

Input:
P_i where $i \in [1, 40]$: 3D point cloud data
$A_{1,j}$ where $j \in [2, 40]$: Transformation matrix to transform frame j to 1
Initialize:
$Final\_PC \leftarrow P_1$
for $k = 2$ **to** 40 **do**
    $\tilde{P}_{1,k} \leftarrow 3D\_Transformation(P_k, A_{1,k})$
    $Final\_PC \leftarrow PC\_Merge(Final\_PC, \tilde{P}_{1,k})$
end for

2.4 Build as complete a 3D model as possible and characterise the surfaces

Surface plane of the merged 3D model of the room are then extracted using the MATLAB vision toolbox, `pcfitplane` function (Mat, 2017). The function fits a plane to a point cloud that has a maximum allowable distance from an inlier point to the plane. It uses another variant of `RANSAC` which is M-estimator SAmple Consensus(`MSAC`). The function returns a point cloud of a plane randomly by default. We selected random points for all the walls and assign it to the function to return the normal vector for the plane that is mapped to left, right and front wall. A visual inspection is performed to check if the objects extracted correctly describe the room.

Then, analysis based on both angles and distance between each pair of planes are carried out by doing the following. We measure (a) the angles between both the left, right walls and the front wall; and (b) the separation between the left and right walls, using a plane point near the center-of-mass of the wall points. The best performance would be getting 90° between them.

3 Results and Discussion

3.1 Extract the relevant data from each point cloud

Irrelevant points outside the window

We cleared the irrelevant points by removing points that have Z values of more than 3.5 as shown in Figure 1. We can see that in the Figure 1 we manage to remove the points outside the window which might cause misalignment in reconstructing the room. However, some useful information about the features of the room are lost in this process, for instance the window in this frame.

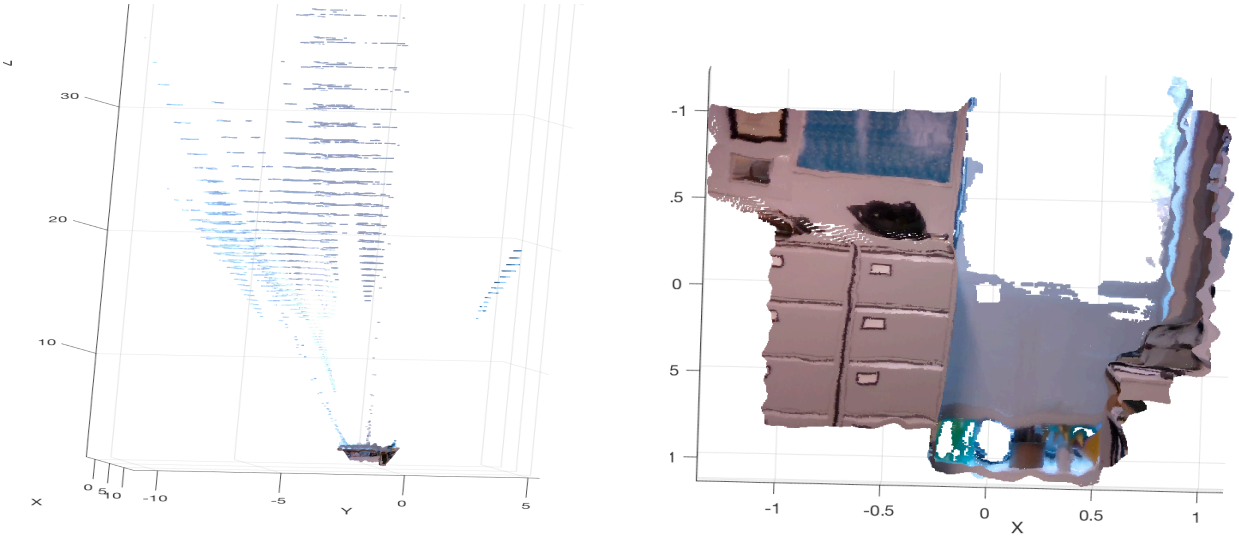

Figure 1: Left: Original point cloud at frame {19}. Right: Cleaned point cloud after removing the irrelevant points.

A person in one frame

In frame {27}, the scene is occluded by a person. After applying the bounding box, we removed the undesirable person as shown in Figure 2. However, there is not much points left which would caused some issues in extracting interest points in later tasks.

Defective depth data due to specular reflections

Defective depth data caused by specular reflections of smooth surfaces are easily removed by counting the number of neighbouring pixels in a given radius as shown in Figure 3

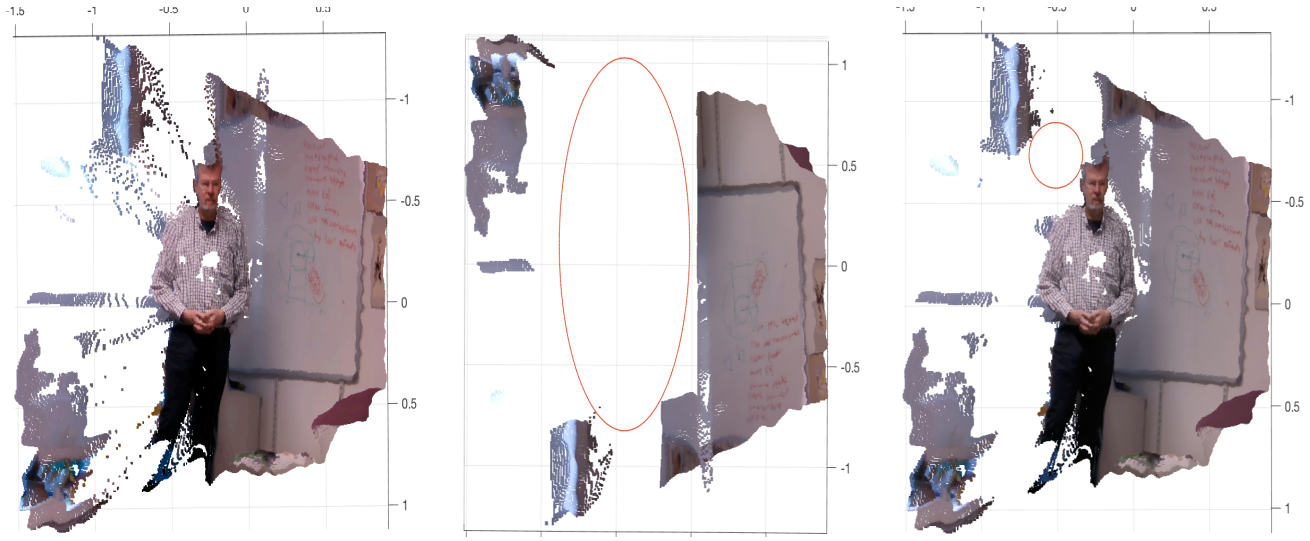

Figure 2: First: Original point cloud at frame {27}. Second: Cleaned point cloud after removing the person. Third: Cleaned point cloud after removing the flying pixels.

Deformed data near the edges of the image

We can see that the edge points, which are mostly noisy are removed. However, the final reconstructed point cloud are not affected significantly with the presence of the edge points.

Figure 3: First: Colour image at frame {12} after applying mask on the edges. Second: Original point cloud. Third: Cleaned point cloud after removing the edges of the images.

3.2 Estimate the reference transformation linking each consecutive pair of frames

The results of detecting the `SIFT` points of some of the scan pairs are shown in Figure 4. The overall performance of `SIFT` is very good due to the properties of invariance to translation, rotation, illumination and scale. Accurate results obtained here help the subsequent steps to converge to a precise solution. Figure 4 on the right illustrates an example of failed `SIFT` keypoint matching: there is only one matched SIFT keypoint between frame {24} and frame {25}.

Random sampling in various iterations in `RANSAC` results in a large number of models to be tested and this may lead to local optimization of good models. This results in a more accurate estimation of the 3D transformation model. However, this introduces high computation time required to compute the optimal 3D transformation model. By having a moving inlier counts and the number of iterations as our stopping criterion, we are able to reduce the computation time significantly, and still produce good results. Each frame takes around 5 seconds to be able to output an optimal transformation. This can be seen from the results obtained as shown in Figure 5.

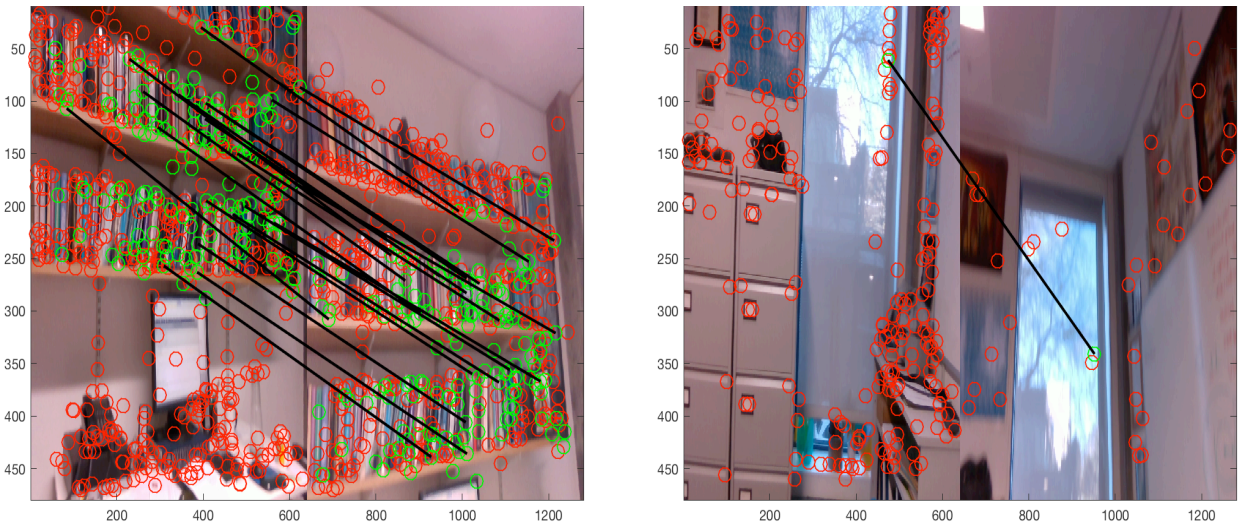

Figure 4: Matching of SIFT descriptors with matched points in green and unmatched points in red.

3.3 Build as complete a 3D model as possible and characterise the surfaces

Figure 6 shows the results of the 3D reconstruction of room in three different perspective view by fusing all of the points into a single 3D coordinate system. Results of the assessment of the room reconstruction are presented in Table 2.

For the extracted walls, we can see that the angle between the left/front wall and right/front wall are slightly off by the ground truth (90°). For the right wall, the slight deviation might be due to the fact that a lot of points (occluded object) have been removed in frame 27. This decreases the number of interest available in the frame that is used to match between frames later on. On the other hand, for the left wall, points belonging to the bookshelf and the desk embedded in left wall were falsely classified as left walls. This is caused by the fact that they are located in the borders of room as shown in Figure 7. This explains the high deviation from the ground truth.

	Results	Ground Truth
Left wall angle	102.7904°	90°
Right wall angle	81.7996°	90°
Perpendicular distance between wall	3.2494 units	N/A
Total computation time	6 minutes	

Table 2: Results of conducted experiment on room reconstruction.

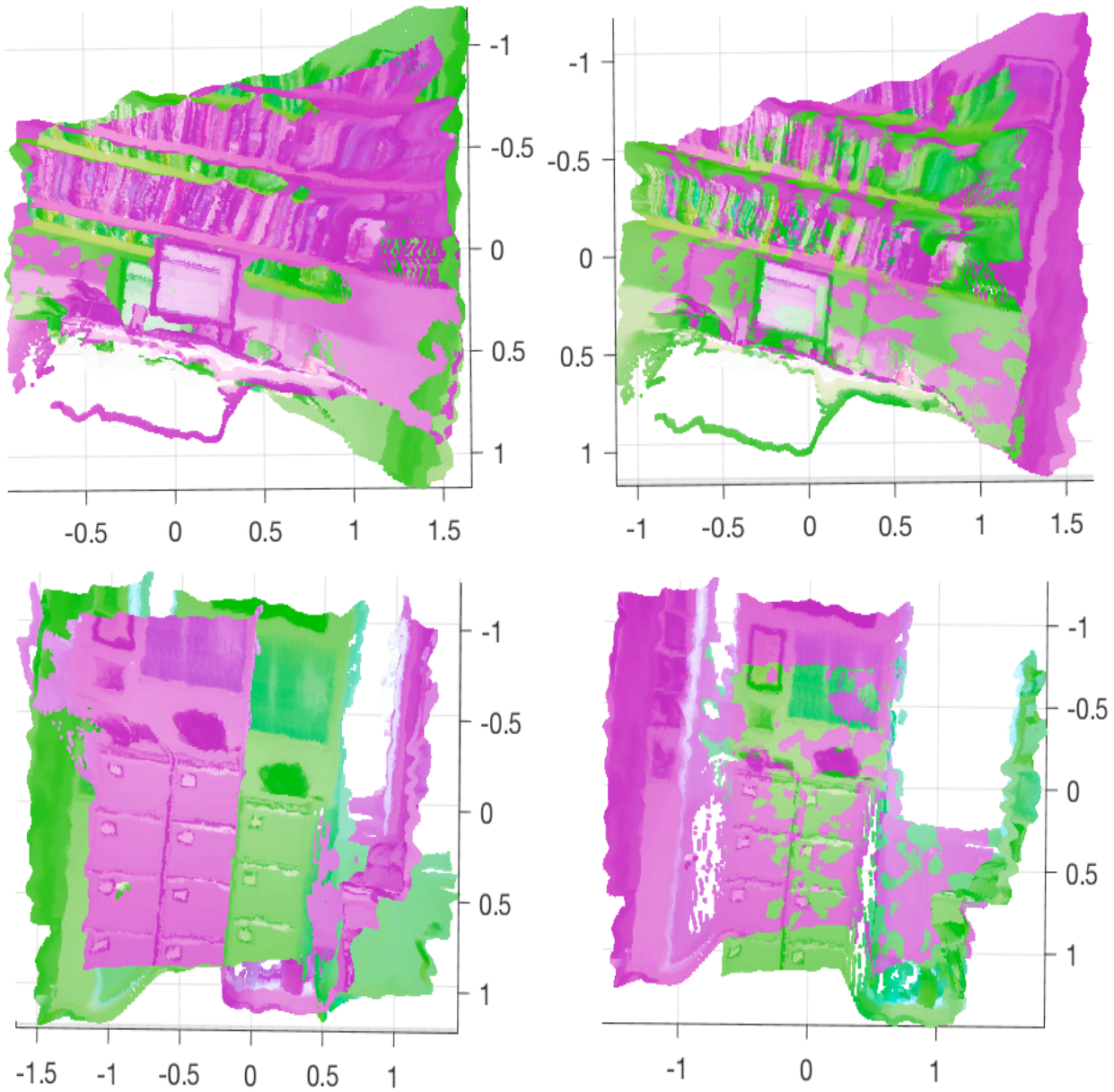

Figure 5: Consecutive pair of frames before (Left) and after (Right) registration by RANSAC.

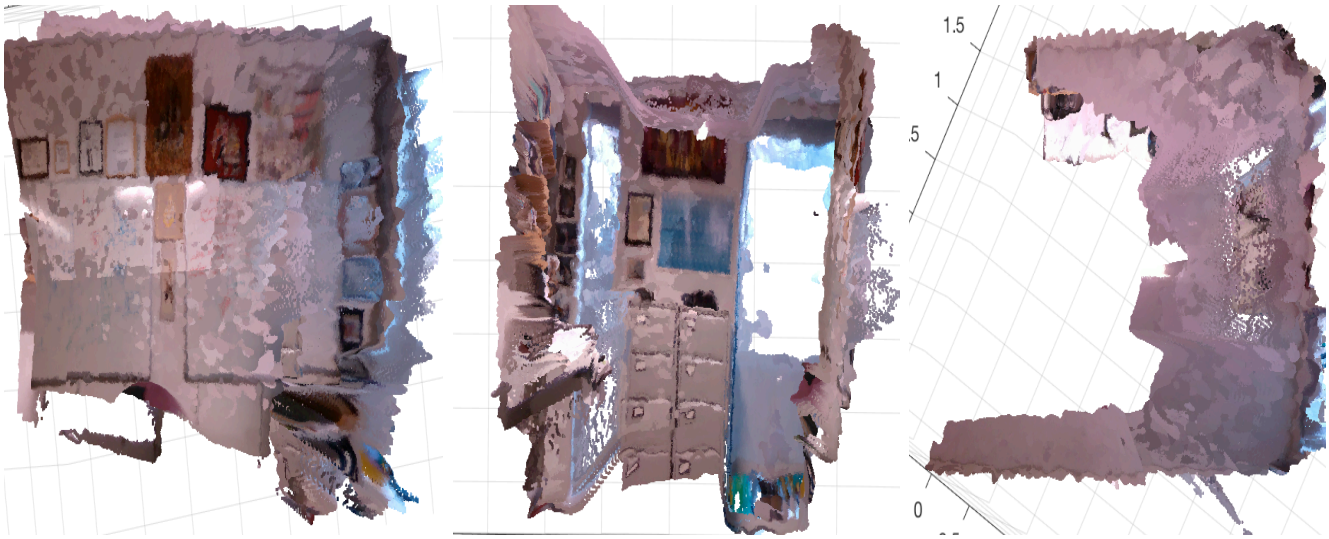

Figure 6: A complete point cloud model after merging 40 frames

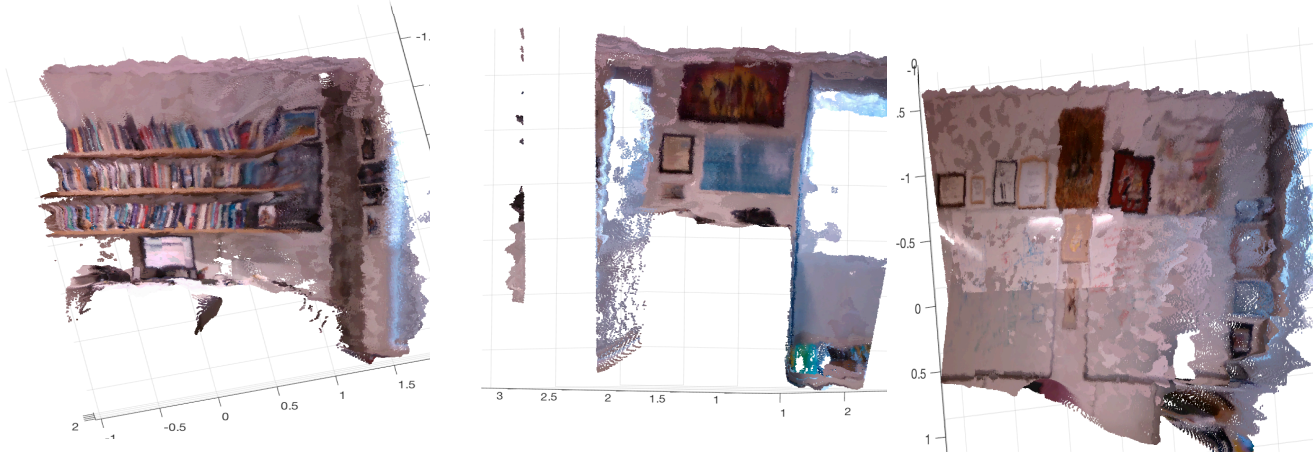

Figure 7: Left, front and right wall extraction from the room.

References

Arun, K. S., Huang, T. S., and Blostein, S. D. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, May 1987. ISSN 0162-8828. doi: 10.1109/TPAMI.1987.4767965. URL `https://doi.org/10.1109/TPAMI.1987.4767965`.

Fischler, Martin A. and Bolles, Robert C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `http://doi.acm.org/10.1145/358669.358692`.

Lorusso, Adele, Eggert, David W., and Fisher, Robert B. A comparison of four algorithms for estimating 3-d rigid transformations. In *BMVC*, 1995.

Lowe, David. Object recognition from local scale-invariant features. volume 2, pp. 1150 – 1157 vol.2, 02 1999. ISBN 0-7695-0164-8. doi: 10.1109/ICCV.1999.790410.

MATLAB version 9.3.0.713579 (R2017b). The Mathworks, Inc., Natick, Massachusetts, 2017.

Vedaldi, A. and Fulkerson, B. VLFeat - an open and portable library of computer vision algorithms. In *ACM International Conference on Multimedia*, 2010.