HIGH SCHOOL EDITION
Journal of Student Research
Volume 10 Issue 4 (2021)

Indoor Point Cloud Registration Process Using Iterative Closest Point (ICP) Algorithm

Youngwoo Chang[1] and Nate Barlett[2]

[1]St. Mary's International School
[2]Advisor, Motiv Research co.

ABSTRACT

In this paper, there are three major sections covered. The paper begins with the explanation of the Lidar sensor, which is a device that uses one or more lasers to measure distances from itself to objects in the world. Such a measurement can be represented as a point cloud, which is a set of (x,y,z) values for every surface that the laser(s) irradiate. Then, the theory of rigid transformations, including the covariance matrix, the translation matrix, and how the Singular Value Decomposition (SVD) can decompose the covariance matrix between two point-clouds to obtain the rotation matrix relating them. In addition, the concept of Iterative Closest Point (ICP) is explored and tested, along with the SVD algorithm, by using the Stanford bunny data sets. Finally, the ICP algorithm is used to combine two different Lidar scans of an office room together to see its effectiveness in a real-world environment.

Introduction

It was a sensational experience to watch the movie, "Ready Player One", which is a story about people competing in virtual reality (VR) to inherit the VR gaming system [1]. In this film, people in 2045 are immersed in the virtual world to forget the pain and worries of the real world. The way that the characters blended in the world of VR being able to explore the details of that world fascinated spectators. Currently, the technology of VR evolves greatly and may, in the near future, have a direct impact on people's lives.

Augmented Reality (AR) and Digital Twin are representative examples of those that are being applied in virtual reality. In particular, the technology called Digital Twin has a great advantage in that it can perform simulations that cannot be executed in the real world due to reasons such as time, resources and safety by converting the environment and objects in the real world into digital data [2]. The two main ways to collect data for the Digital Twin technology is to collect the surrounding environment using photos or lasers as point data. In this paper, in particular, the point data collection of the surrounding environment using a laser sensor and the processing and visualization of the collected data will be explained.

By using a laser scanner, also known as Lidar (Light Detection and Ranging), a scanned region can be represented in the form called a 'point-cloud', depicting 3D objects or space [3]. In order to completely scan the environment, the Lidar should be used in multiple locations—this will reduce 'shadowed' locations. ICP (Iterative Closest Point) algorithm is commonly used for point cloud registration, a process that matches scans measured from different lidar locations or orientations together [4]. The process is analogous to stitching together photographs taken at different locations together to form a large photo capturing a full landscape, but at a 3-dimensional scale. The registration process used in this paper consists of obtaining the corresponding points of two point-cloud data sets and using two algorithms, SVD (Singular Value Decomposition) and ICP (Iterative Closest Point), to match the two sets together [5]. The visualization of the resulting registration of point cloud data is included.

Lidar Sensor

Using Lidar scanners, distances, dimensions and the returned light's intensity can be measured—done by sending a laser pulse and measuring the time taken for the laser to reach a surface and return to the scanner. The distance is calculated by: $distance = time \times speed\ of\ light$. Thus, a large point-cloud comprises a set of X, Y and Z coordinates, and, in some cases, additional attributes such as intensity or colors can be obtained. Each pulse containing information about the environment are combined to represent the entire 3D region.

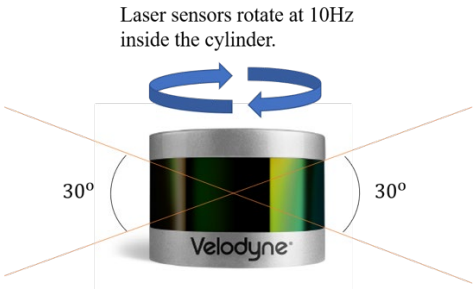

Figure 1. Velodyne VLP-16 Lite

In this study, Velodyne's lidar sensor, VLP-16 Lite as shown in Figure 1, which has 16 laser sensors in vertical axis with 30 degrees vertical view and a 360-degree horizontal scan plane was used. The main features of the VLP-16 Lite can be found in Table 1 [6].

Table 1. Main features of VLP-16 Lite

Features	Remarks
Channel	16
Wavelength	903 nm
Ranging accuracy	+/-3cm (typical)
Measurement range	Up to 100m
Output data points	300,000 points/s
Vertical view angle	$30^o(+15^o$ to $-15^o)$
Horizontal view angle	360^o
Horizontal angular resolution	$0.1^o \sim 0.4^o$
Laser rotation frequency used in this paper	10Hz (600RPM)
Weight	830g
Dimension	103mm diameter x 72mm height
Power consumption	8W (typical)

The point cloud data collected by the lidar can be exported in different data format, e.g., pcap or rosbag. The rosbag format is commonly used in ROS (Robot Operating System). ROS is an open-source, operating system for robot developments that provides the services the user would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [7]. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. A detailed description on how ROS works is excluded due to the scope of this paper. However, it should be noted that points are innately measured in polar coordinates by the VLP-16 Lite as shown in Figure 2 [6].

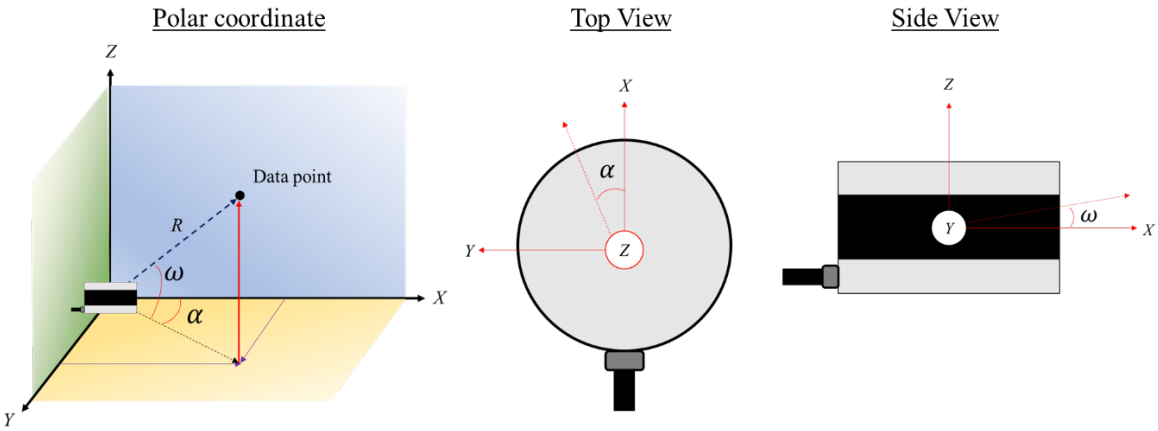

Figure 2: Lidar's dataset coordinate system exported in ROS environment

For an efficient data processing of point cloud registration, it is required to convert the dataset in polar coordinate into a Cartesian coordinate system. The mathematical model of VLP-16 Lite, which calculates the (x, y, z) coordinates from ROS environment is given as follows.

Equation 1: Conversion of polar coordinate to cartesian coordinate:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} R \cdot cos\omega \cdot cos\alpha \\ R \cdot cos\omega \cdot sin\alpha \\ R \cdot sin\omega \end{bmatrix}$$

An example of point cloud data converted to (x, y, z) coordinate of a single data block exported from Lidar is shown in Figure 3.

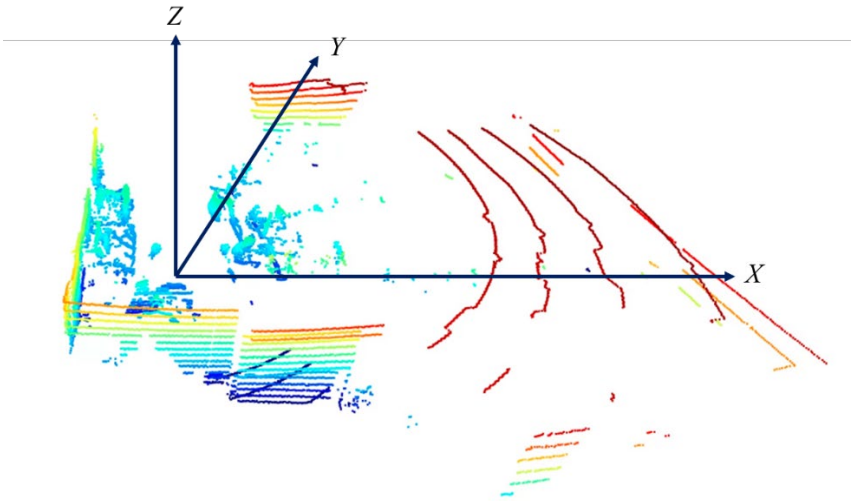

Figure 3. A point cloud set measured by one full 360-degree rotation of the VLP-16 laser sensors

Rigid Transformation for Point Clouds

The basic concept of point-cloud registration is to merge two or more point-cloud sets using an optimal transformation matrix which is a pair of rotation and translation matrices. A transformation that does not alter the size or shape of an object; rotations, reflections, translations are known as rigid transformations. As reflections and scaling are not applied in this exploration, transformations consisting of rotations and translation are considered.

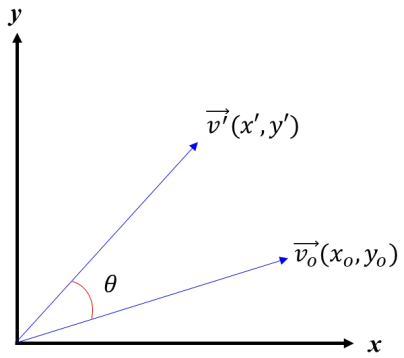

Figure 4. Rotation of a vector by θ in 2-D space

Consider a matrix that rotates a given vector, $\overrightarrow{v_o}$, by a counterclockwise angle, θ in a fixed coordinate system. Then the matrix rotating $\overrightarrow{v_o}$ by θ is expressed as:

Equation 2: Rotation matrix for angle, θ:

$$R_\theta = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

So, $\overrightarrow{v'} = R_\theta \cdot \overrightarrow{v_o}$.

When extending a 2-dimensional rotation matrix to a 3-dimensional space, the matrices for counterclockwise rotation by angles alpha, beta, and gamma around the x, y, and z axes are expressed:

Equation 3: Rotation matrix for each axis x, y, and z:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix},\ R_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix}, R_z(\gamma) = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

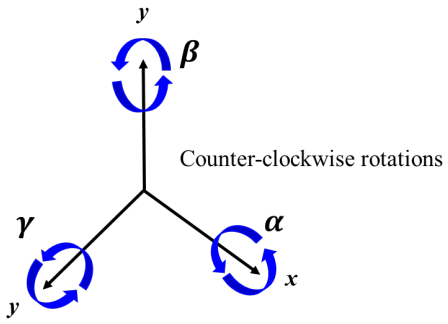

Figure 5. Rotation along x,y, and z-axis in 3-D space

When the rotations along x, y and z axes simultaneously, the overall rotation matrix, *R* is obtained by matrix dot product [8]:

Equation 4: Combined rotation matrix

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha) = \begin{bmatrix} cos\alpha \cdot cos\beta & cos\alpha \cdot sin\beta \cdot sin\gamma - sin\alpha \cdot cos\gamma & cos\alpha \cdot sin\beta \cdot cos\gamma + sin\alpha \cdot sin\gamma \\ sin\alpha \cdot cos\beta & sin\alpha \cdot sin\beta \cdot sin\gamma + cos\alpha \cdot cos\gamma & sin\alpha \cdot sin\beta \cdot cos\gamma - cos\alpha \cdot sin\gamma \\ -sin\beta & cos\beta \cdot sin\gamma & cos\beta \cdot cos\gamma \end{bmatrix}$$

The translation operation in 3-dimensional space is illustrated in Figure 6.

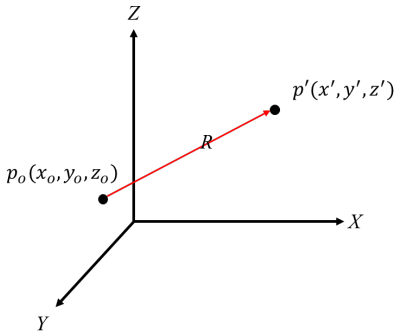

Figure 6. Translation operation in 3-D space

Then, the translation operation is unfolded in the equation below.
Equation 5: Translation operations:

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

The translations can be rewritten into a matrix as shown in Equation 6.
Equation 6: Translation operation in matrix form.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix}$$

Now, the transformation matrix can be considered simultaneously with the rotation matrix, and manipulate the transformation matrix so that the two operations take place together. In this case, the transformed point, $p_t(x_t, y_t, z_t)$ from $p_o(x_o, y_o, z_o)$, can be expressed by:

Equation 7: Rotation and translation operation.

$$p_t = R \cdot p_o + t.$$

In a matrix expression, it can be rewritten in the following equation.
Equation 8: Rotation and translation operation in matrix form.

$$\begin{bmatrix} R_{3\times3} & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \end{bmatrix}$$

$R_{3\times3}$ is the rotation matrix in 3-dimensional space. Here, the transformation matrix is not a square matrix but a 3x4 matrix. An inverse matrix can transform the point, $p_t(x_t, y_t, z_t)$, back to its original point, $p_o(x_o, y_o, z_o)$, but the transformation matrix above is not inversible as it is not a square matrix. To make this possible, an additional row that does not affect the point – by adding '0's and a '1' – is added so that the transformation matrix becomes 4x4 square matrix as shown in Equation 9.

Equation 9: Transformation matrix, *T*.

$$T = \begin{bmatrix} & R_{3\times3} & & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$

The transformation matrix can be easily tested even without Lidar's point-cloud data sets. A well-known open point cloud data source, Stanford bunny dataset was used [9]. The original Stanford bunny data consists of 35,947 points. The point-cloud data set was sampled down to 3,023 points as it is unnecessary to use all the points given to test the effectiveness of the transformation matrix. Using Python's open3d library, the down sampled bunny dataset was rotated by $-45^o(45^o$ clockwise) and translated along the x-axis by 0.1 as shown in Figure 7.

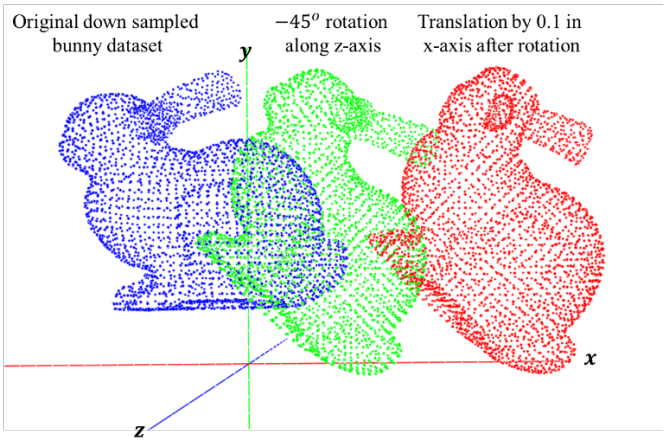

Figure 7. Transformation of Stanford bunny dataset by -45 degrees rotation and 0.1 distance translation

Identification of Transformation Matrix

In the previous section, it is described how a point-cloud data set can be rotated and translated via the transformation matrix. This process can be reversed to bring the transformed matrix back to its original point-cloud data set by obtaining the transformed matrix. This can be done by calculating the covariance matrix of the two sets and then applying the SVD. However, a fundamental assumption for the covariance matrix and SVD is that the number of points comprising the two data sets must be equal well structured, meaning that the corresponding points of the two sets are known.

Covariance Matrix

A covariance matrix is a square matrix giving the covariance between each pair of elements of a given random vector. There is a similar named statistical measure, variance. Variance measures the variation of a single random variable for example, the math score of a student in a large class, whereas covariance is a measure of how much two random variables vary together, for example, the math score and English score of a student in a large class. The formula for variance is given by:

Equation 10: Variance.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Where n is the number of data samples and $\bar{x}$ is the mean of the random variable x. The covariance is an extension of the variance where another variable is introduced, which is expressed by:

Equation 11: Covariance of two random variables.

$$\sigma^2(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

Where $\bar{y}$ is the mean of the random variable y.

Assuming there are two point-cloud data sets, $\boldsymbol{P}$ and $\boldsymbol{Q}$, the covariance of $\boldsymbol{P}$ and $\boldsymbol{Q}$ is given by the following equation [10]:

Equation 12: Covariance matrix.

$$\boldsymbol{C_{3\times3}} = \frac{1}{n}(\boldsymbol{P} - \bar{\boldsymbol{P}})(\boldsymbol{Q} - \bar{\boldsymbol{Q}})^{\boldsymbol{T}}$$

Where $\bar{P}$ and $\bar{Q}$ are the centroids of the point-cloud data set, $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. $\boldsymbol{P}$ and $\boldsymbol{Q}$ are the point sets in 3-dimensional space and defined as $3 \times n$ matrices as shown below.

Equation 13: $\boldsymbol{P}$ and $\boldsymbol{Q}$ in matrix.

$$\boldsymbol{P} = \begin{bmatrix} x_{p,1} & \cdots & x_{p,n} \\ y_{p,1} & \cdots & y_{p,n} \\ z_{p,1} & \cdots & z_{p,n} \end{bmatrix}, \ \boldsymbol{Q} = \begin{bmatrix} x_{q,1} & \cdots & x_{q,n} \\ y_{q,1} & \cdots & y_{q,n} \\ z_{q,1} & \cdots & z_{q,n} \end{bmatrix}$$

In order to perform the matrix multiplication of $\boldsymbol{Q}$ and **P**, $\boldsymbol{Q}$ needs to be transposed into a $n \times 3$ matrix. The resulting covariance matrix will give a $\boldsymbol{C_{3\times3}}$ for 3-dimensional point cloud data sets (Equation 14). Using the covariance matrix is important as it can be used to extract the rotation matrix [12].

Equation 14: Elements of covariance matrix.

$$\boldsymbol{C_{3\times3}} = \frac{1}{n}\begin{bmatrix} \sum(x_p - \bar{x}_p)(x_q - \bar{x}_q) & \sum(x_p - \bar{x}_p)(y_q - \bar{y}_q) & \sum(x_p - \bar{x}_p)(z_q - \bar{z}_q) \\ \sum(y_p - \bar{y}_p)(x_q - \bar{x}_q) & \sum(y_p - \bar{y}_p)(y_q - \bar{y}_q) & \sum(y_p - \bar{y}_p)(z_q - \bar{z}_q) \\ \sum(z_p - z_p)(x_q - \bar{x}_q) & \sum(z_p - \bar{z}_p)(y_q - \bar{y}_q) & \sum(z_p - \bar{z}_p)(z_q - \bar{z}_q) \end{bmatrix}$$

Singular Value Decomposition (SVD)

The singular value decomposition of an arbitrary $m \times n$ matrix, $\boldsymbol{A}$ is:
Equation 15: Singular value decomposition equation.

$$A = USV^T$$

Where $\boldsymbol{U}$ is $m \times m$ and $\boldsymbol{V}$ is $n \times n$ orthogonal matrices, and $\boldsymbol{S}$ is $m \times n$ diagonal matrix containing the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ with assumption of $n < m$. If we apply SVD to 3×3 covariance matrix, Equation 15 becomes the following matrix form:
Equation 16: SVD for covariance matrix

$$C_{3\times3} = [U_{3\times3}] \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} [V_{3\times3}]^T$$

The SVD calculation was performed by using NumPy library in Python. With the $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices from SVD calculation, the rotation matrix is given by the following equation [11].
Equation 17: Rotation matrix.

$$R = VU^T$$

The translation aligns the centroid of the point-cloud data set, $\boldsymbol{Q}$, with the rotated centroid of the point-cloud data set $\boldsymbol{P}$ (Equation 18) [11].
Equation 18: Translation matrix.

$$t = \bar{Q} - R\bar{P}$$

As a setup, the original Stanford bunny point cloud dataset was translated by -45^o rotation and 0.1 translation in x-axis and was treated as if this transformation was unknown. Now, the transformation matrix of the original data and the transformed data was calculated by using SVD. Now, the transformation matrix is inversed and applied to the transformed bunny data to match the original bunny data and allow the user to verify its level of correspondence (Figure 8).

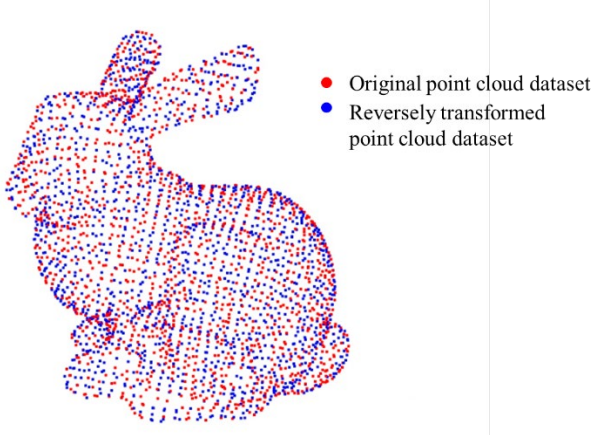

Figure 8. Reverse transformation back to the original bunny point-cloud data set.

Upon displaying both data sets, it was observed that the two were exactly matched, making them indiscernible. Therefore, to illustrate that the two data sets' correlation, even number indices of the original bunny data and odd number indices of the transformed data set were used.

ICP (Iterative Closest Point)

Under the condition that the correspondence between points of two different data sets are known, SVD is a powerful algorithm that can find accurate transformation matrices. However, keeping this condition is nigh impossible in real world circumstances. The point-clouds collected by a Lidar is not structured, meaning that the information pertaining to which points measured from one Lidar position correspond to the same points measured from another Lidar position is unknown. To solve this problem, ICP algorithm is commonly used [12]. The ICP algorithm uses SVD to find the transformation between two-point cloud datasets, but makes a key assumption because the point correspondences are unknown as mentioned. ICP assumes that the corresponding points between two datasets are the ones with the smallest distance, meaning, given a point, P, in one point cloud, its corresponding point in the other point cloud is the point Q, which is closest in distance to P. The ICP algorithm begins by identifying corresponding points in terms of the distance. Then SVD is applied to calculate an intermediary transformation matrix to bring the target point-cloud set close to its original point cloud set. The corresponding points between the two sets and the intermediary transformation matrix are calculated and applied to the target set to being it even closer to the original data set. This process is repeated until the average distance between the corresponding points of the target source and the original source is under a certain threshold, hence the word 'iterative' in its name [13]. Though powerful, the ICP algorithm is applicable when there is a relatively good starting point–meaning that the transformation between two point-clouds is not large to begin with. Otherwise, it can get 'trapped' in a local minimum and the resulting solution will be useless.

ICP algorithm is a time-consuming process as distances between points in one set and points in the other have to be calculated in order to determine which points correspond with each other. In order to accelerate the ICP algorithm, K-d tree (short for k-dimensional tree) method is used to find the correspondences between the point datasets. K-d tree is a space-partitioning data structure for organizing points in a k-dimensional space. K-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key like nearest neighbor searches [14]. In this paper, the ICP algorithm is implemented in Python with scikit-learn library supporting K-d tree function in handy [15]. The code was tested using Stanford bunny data – bunny data from 0° view angle and 45° view angle. The test shows a good registration result.

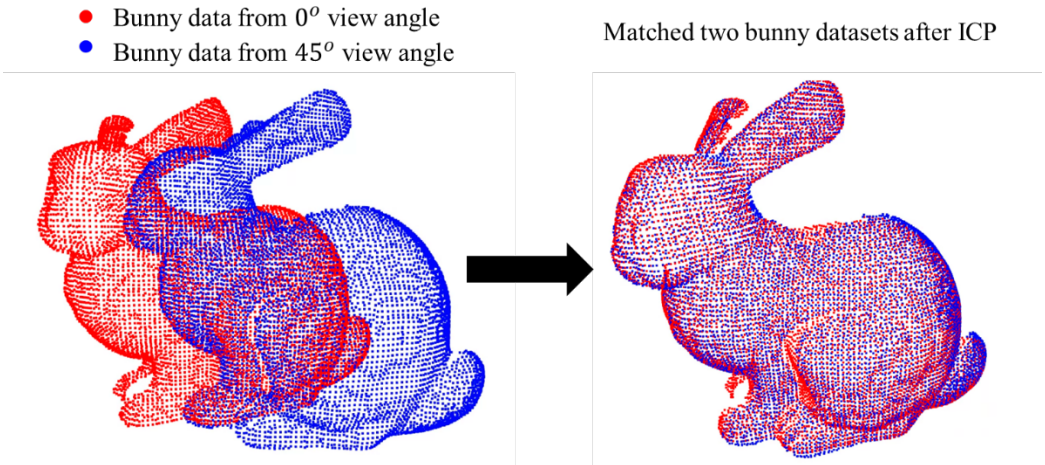

Figure 9. Matched result of two different bunny datasets using ICP.

Results: Point Cloud Registration

Using the ICP algorithm, registrations of multiple point-cloud data sets from a real environment was performed using a Lidar scanning robot system (Figure 10).

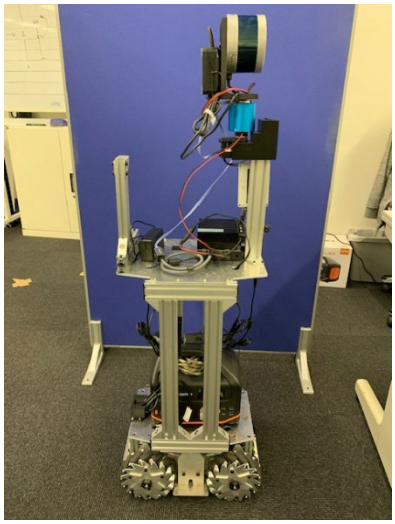

Figure 10. Lidar and robot system used to collect point-cloud data sets

The robot system was developed by Motiv Research co. This system scans 3-dimensional point clouds and is able to store information pertaining to the scanner's position and orientation. The Lidar scanning was done in two different locations within an office room.

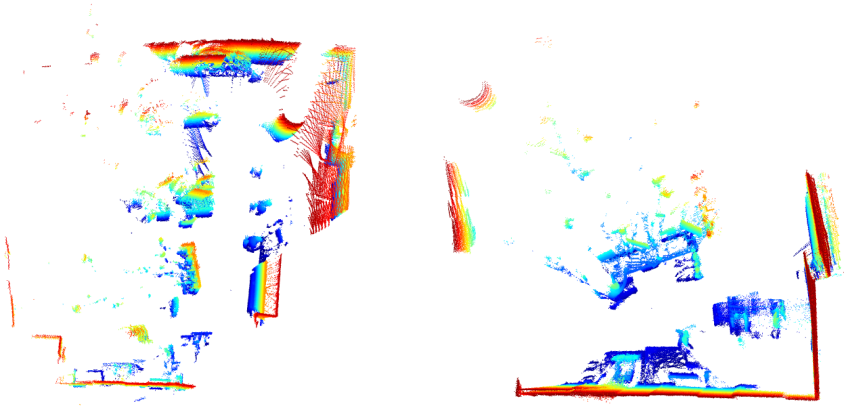

Figure 11. Two point-cloud data sets scanned at two different locations

As mentioned previously, the ICP algorithm fails to correctly correlate pairs of two data sets if the source and target point cloud datasets have a relatively large transformation between them. Hence, the scanner's position and orientation information–information pertaining to the Lidar's location and angle–were used to aid the algorithm. Theoretically, the position and orientation information are enough to create a transformation matrix of the target point-cloud–the position information can be used to calculate translation while the orientation information can be used to calculate the rotation matrix–but the robot's pose estimation is not accurate; thus, this information can only be used as an aid to reduce error for the ICP algorithm. This can be done by using the position and orientation information as

a 'guess' to bring the point-cloud set close enough to the original point-cloud set. Then ICP algorithm can be performed to accurately register the two point-cloud data sets. The registered result is presented in Figure 12.

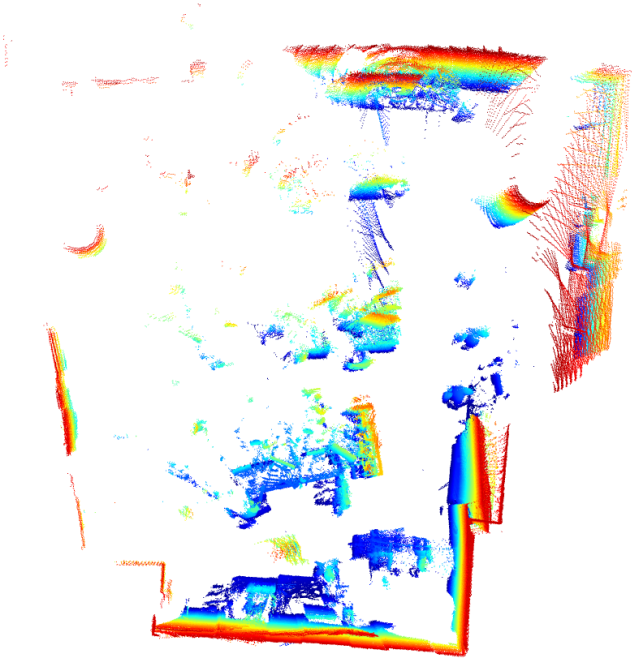

Figure 12. Registration result

Conclusion

The SVD algorithm and the ICP algorithm are applicable when trying to match two point-cloud data sets together. The main issue of the SVD algorithm is how limiting it is in terms of its applications—it can only be used in specific circumstances where the correspondences of source and target point cloud sets are known, rendering it impractical for general use. However, the transformation matrix can be obtained by repeatedly computing the intermediary transformation matrices along with the K-d tree method, which is the essence of the ICP algorithm. The effectiveness of the ICP algorithm was observed to be successful when tested using the Stanford bunny data sets. Furthermore, the application of the ICP algorithm was expanded to a real environment through the use of different scans of the office. Though the registration accurately combined the two different perspectives of the room, there was a noticeable error. Moreover, an aspect of the exploration that was not addressed was the presence of noise; in the Stanford data set and the data set from the office, noise was not a huge factor to consider. However, in many other locations, especially in the streets where there are moving people and objects, there may be enough noise to ruin the accuracy of transformation matrices. In future exploration, the two factors, error reduction and consistence in performance in different environments, should be addressed.

Acknowledgements

I would like to thank Motiv-research co. for not only allowing me to use their Lidar system but also offering me guidance to make this paper possible.

References

[1] S. Spielberg, Director, *Ready Player One.* [Film]. USA: Warner Bros. Pictures, 2018.

[2] "What is Digital Twin? How does it work?," YouTube, 10 January 2019. [Online]. Available: https://www.youtube.com/watch?v=iVS-AuSjpOQ&ab_channel=GeospatialWorld.

[3] "What is LiDAR? (& Why is It on Apple Devices All of a Sudden)," YouTube, 17 May 2020. [Online]. Available: https://www.youtube.com/watch?v=FOxxqVzDaaA&ab_channel=TheUnlockr.

[4] W. Burgard and C. Stachniss, "Introduction to Mobile Robotics Interative Closest Point Algorithm," [Online]. Available: https://cs.gmu.edu/~kosecka/cs685/cs685-icp.pdf.

[5] D. Kalman, "A Singularly Valuable Decomposition: The SVD of a Matrix," *The College Mathematics Journal,* vol. 27, no. 1, 1996.

[6] "Velodyne Lidar VLP-16 User Manual," 26 February 2019. [Online]. Available: https://velodynelidar.com/wp-content/uploads/2019/12/63-9243-Rev-E-VLP-16-User-Manual.pdf.

[7] "ROS," ROS.ORG, [Online]. Available: https://www.ros.org/.

[8] P. Evans, "Rotations and rotation matrices," *Biological Crystallography,* p. 1355~1359, 12 June 2011.

[9] "The Stanford 3D Scanning Repository," Stanford University, 14 August 2014. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/.

[10] N. Janakiev, "Understanding the Covariance Matrix," datascience+, 3 August 2018. [Online]. Available: https://datascienceplus.com/understanding-the-covariance-matrix/.

[11] D. Eggert, A. Lorusso and R. Fisher, "Estimating 3-D rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications,* pp. 272-290, 1997.

[12] P. Jana and M. Dalibor, "Notes on Iterative Closest Point Algorithm," in *17th Conference on Applied Mathematics*, Slovak University of Technology in Bratislava, 2018.

[13] D. Holz, A. Ichim, F. Tombari, R. Rusu and S. Behnke, "Registration with the Point Cloud Library," *IEEE Robotics & Automation Magazine,* vol. 22, no. 4, pp. 110-124, 2015.

[14] H. Marius, "Tree algorithms explained: Ball Tree Algorithm vs. KD Tree vs. Brute Force," towards data science, 15 June 2020. [Online]. Available: https://towardsdatascience.com/tree-algorithms-explained-ball-tree-algorithm-vs-kd-tree-vs-brute-force-9746debcd940.

[15] "sklearn.neighbors.KDTree," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html.

[16] J. Shlens, "A Tutorial on Principal Component Analysis," 7 April 2014. [Online]. Available: https://arxiv.org/pdf/1404.1100.pdf.

[17] "numpy.linalg.svd," Numpy.org, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html.

[18] "YouTube," dronegenuity, 2 March 2020. [Online]. Available: https://www.youtube.com/watch?v=yXCkyuo8bcs&ab_channel=dronegenuity.