OSU Mechanical Engineering

Smart Products Laboratory

Sensors | Seven

ME Course Number
Spring 2019

Table of Contents

List of Figures

1 Overview

In this lab you will program a LIDAR sensor and a double servo motor joint configuration to scan an object and present the 3D point cloud model.

2 Background

The information below should provide a basic introduction to the concepts.

2.1 Coordinate Transformations

${}^{G}R_{B}$ – Rotation of frame B with respect to frame G, with ${}^{G}\boldsymbol{R}_{B} \in \mathbb{R}^{4\times4}$ or ${}^{G}R_{B} \in \mathbb{R}^{3\times3}$

${}^{G}D_{B}$ – Translation of frame B with respect to frame G, with ${}^{G}\boldsymbol{D}_{B} \in \mathbb{R}^{4\times4}$

${}^{G}T_{B}$ – Transformation of frame B with respect to frame G, with ${}^{G}\boldsymbol{T}_{B} \in \mathbb{R}^{4\times4}$

${}^{G}d_{B}$ – translation of frame B with respect to frame G, with ${}^{G}d_{B} \in \mathbb{R}^{3\times1}$

${}^{G}r_{p}$ – position of point p with respect to frame G, with ${}^{G}r_{p} \in \mathbb{R}^{3\times1}$

Transformations

The general 4x4 homogeneous transformation matrix is,

$$
{}^{G}\boldsymbol{T}_{B} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ \mathrm{r}_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ p_{41} & p_{42} & p_{43} & s_{44} \end{bmatrix} = \begin{bmatrix} rotation & translation \\ perspection & scaling \end{bmatrix}
$$

With the homogeneous rotation and translation matrices being,

$$
{}^{G}\boldsymbol{R}_{B} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ \mathrm{r}_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{G}R_{B} & 0 \\ 0 & 1 \end{bmatrix}
$$

$$
{}^{G}\boldsymbol{D}_{B} = \begin{bmatrix} 1 & 0 & 0 & d_{14} \\ 0 & 1 & 0 & d_{24} \\ 0 & 0 & 1 & d_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} I_{3\times3} & {}^{G}d_{B} \\ 0 & 1 \end{bmatrix}
$$

For Euclidean transformations

$$
{}^{G}\boldsymbol{T}_{B} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_{14} \\ \mathrm{r}_{21} & r_{22} & r_{23} & d_{24} \\ r_{31} & r_{32} & r_{33} & d_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{G}R_{B} & {}^{G}d_{B} \\ 0 & 1 \end{bmatrix} = {}^{G}\boldsymbol{D}_{B}\, {}^{G}\boldsymbol{R}_{B}
$$

Rotation Matrices

Global or equivalently inertial coordinate frames will be denoted in (X, Y, Z) cartensian form. Local or equivalently moving coordinate frames will be in the cartesian form of (x, y, z). Rotations about global or equivalently inertial coordinate axes will be cast in the Greek form of (Φ, Θ, Ψ). Rotations about the local or equivalently moving coordinate axes will take the Greek form of $(\varphi, \vartheta, \psi)$. The fundamental global rotation matrices are,

$$R_{Z,\Psi} = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\Psi & -s_\Psi & 0 \\ s_\Psi & c_\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{Y,\Theta} = \begin{bmatrix} \cos(\Theta) & 0 & \sin(\Theta) \\ 0 & 1 & 0 \\ -\sin(\Theta) & 0 & \cos(\Theta) \end{bmatrix} = \begin{bmatrix} c_\Theta & 0 & s_\Theta \\ 0 & 1 & 0 \\ -s_\Theta & 0 & c_\Theta \end{bmatrix}$$

$$R_{X,\Phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \sin(\Phi) & \cos(\Phi) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\Phi & -s_\Phi \\ 0 & s_\Phi & c_\Phi \end{bmatrix}$$

The fundamental local (Euler) rotation matrices are,

$$R_{z,\psi} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{y,\vartheta} = \begin{bmatrix} \cos(\vartheta) & 0 & -\sin(\vartheta) \\ 0 & 1 & 0 \\ \sin(\vartheta) & 0 & \cos(\vartheta) \end{bmatrix} = \begin{bmatrix} c_\vartheta & 0 & -s_\vartheta \\ 0 & 1 & 0 \\ s_\vartheta & 0 & c_\vartheta \end{bmatrix}$$

$$R_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\varphi & s_\varphi \\ 0 & -s_\varphi & c_\varphi \end{bmatrix}$$

2.2 LiDAR Scanner Kinematic Model

The scanner setup includes two servos represented as controllable joints with parameters θ_1 and θ_2. Both servos have a range of 90° to −90° however the servo2 shouldn't be actuated below 20°. The system can be modeled by two rigid linkages each of which is rotated by a servo motor.To be able to use this model in terms of coordinate transformation matrices you will need to measure the indicated "linkages" indicated in the figures below.

The measurements include:

$$d_1, a_{x_1}, d_2, a_{y_2}, a_{x_2}$$

You will also need to determine the set rotations, α, from one coordinate frame to another to correctly the linkage.

The reason we do this is to transform the given lidar measurement, referred to as x_{meas} or in vector form ${}^2r_p$, which will be purely along the X_2 coordinate axis, into a global coordinate point given the angle of servo1 and servo2. To solve for the global coordinates of the lidar measurement ${}^0r_p$ you should derive the matrix ${}^0T_2$ using the following equations.

$${}^0r_p = {}^0T_2\, {}^2r_p$$

$$ {}^{0}T_2 = {}^{0}T_1\, {}^{1}T_2 $$

$$ {}^{0}r_p = \begin{bmatrix} {}^{0}x_p \\ {}^{0}y_p \\ {}^{0}z_p \\ 1 \end{bmatrix} = {}^{0}T_2 \begin{bmatrix} {}^{2}x_p \\ {}^{2}y_p \\ {}^{2}z_p \\ 1 \end{bmatrix} = {}^{0}T_2 \begin{bmatrix} x_{meas} \\ 0 \\ 0 \\ 1 \end{bmatrix} $$

The link transformation matrices are calculated with the following transformations

$$ {}^{i-1}T_i = D_{i-1,d_i} R_{z_{i-1},\theta_i} D_{i-1,a_i} R_{x_{i-1},\alpha_i} $$

$$ D_{i-1,d_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix},\ D_{i-1,a_i} = \begin{bmatrix} 1 & 0 & 0 & a_{x_i} \\ 0 & 1 & 0 & a_{y_i} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ R_{z_{i-1},\theta_i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_{x_{i-1},\alpha_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

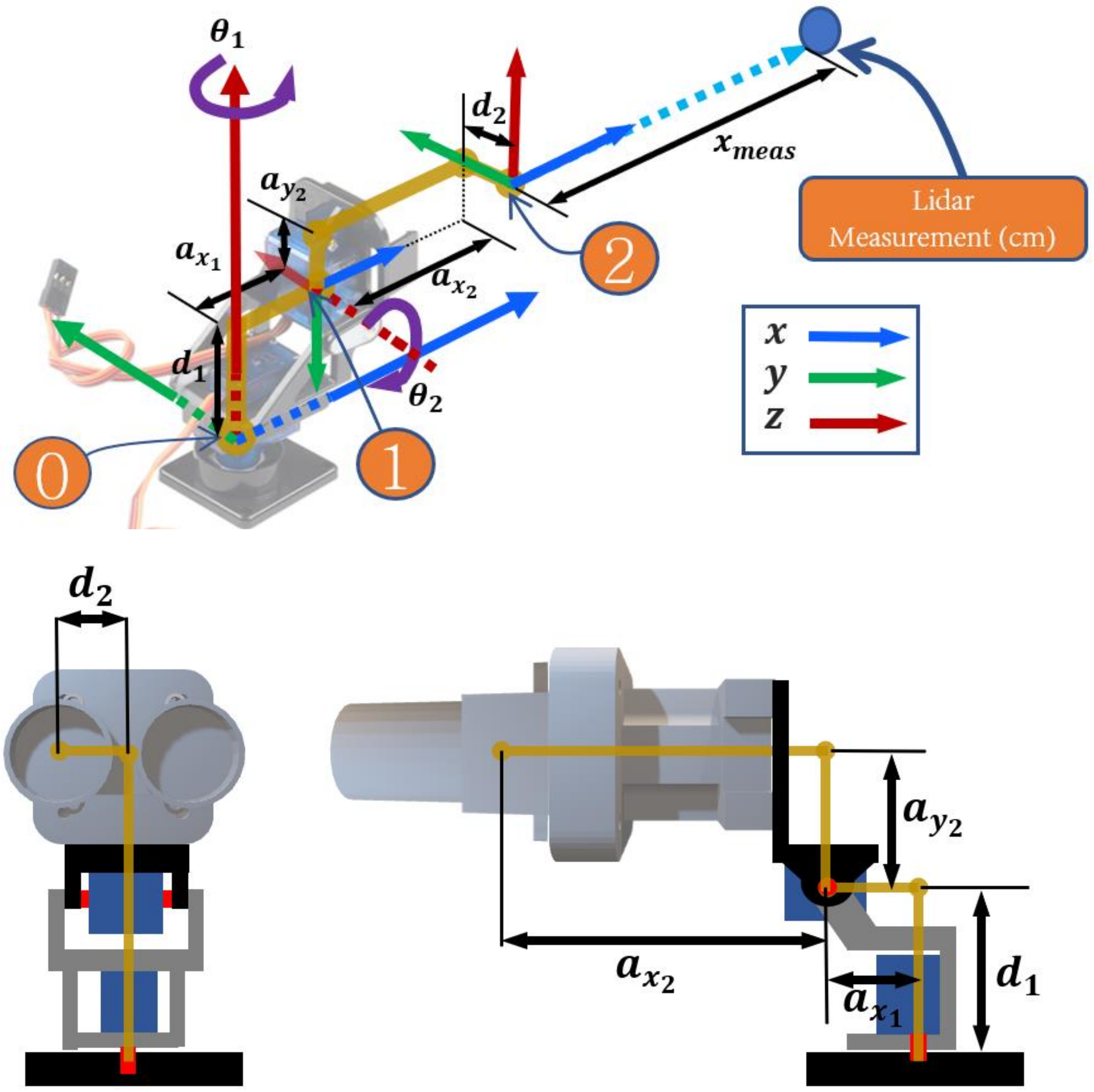

Figure 1 – LiDAR Scanner Model

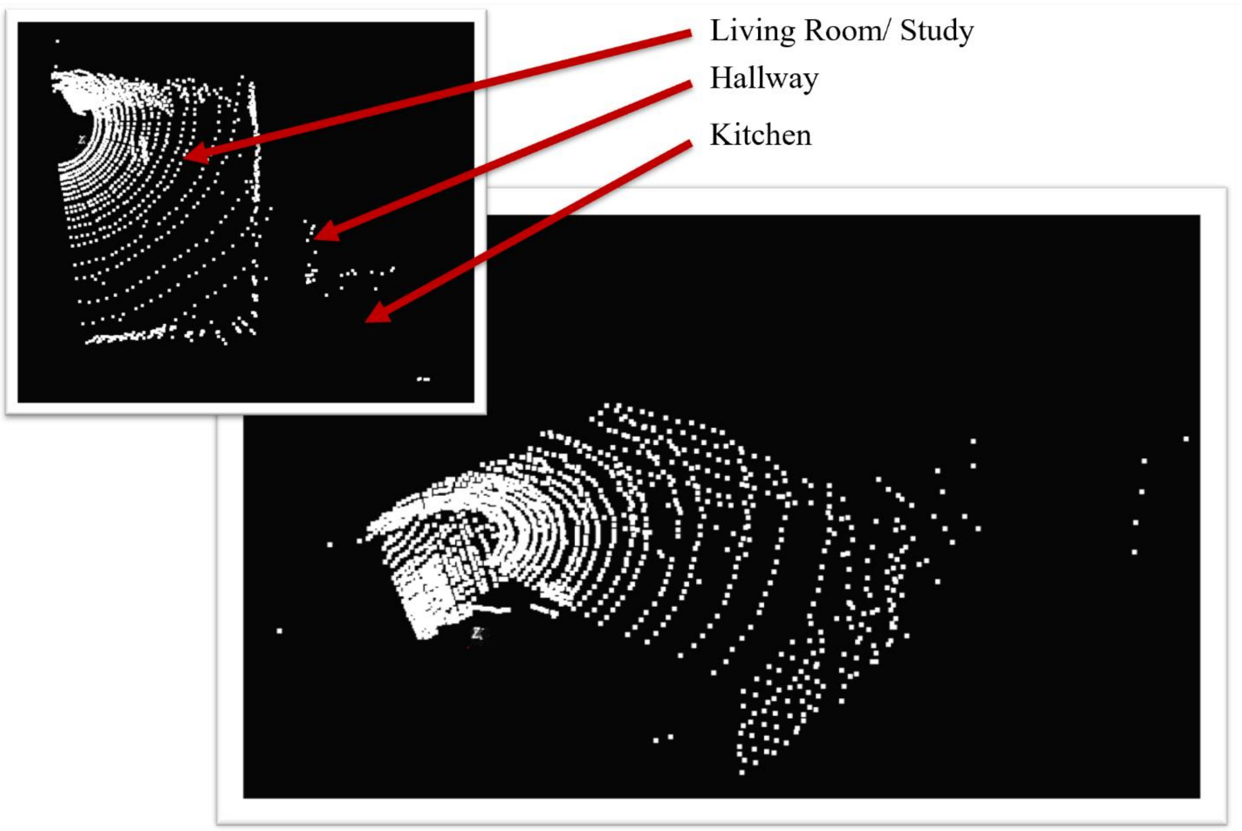

Figure 2 – Example Scanning Result

3 References & Resources

3.1 References

The following references may be helpful to complete the lab.

1. https://en.cppreference.com/w/cpp/io/basic_fstream
2. https://github.com/pimoroni/pantilt-hat
3. http://docs.pimoroni.com/pantilthat/index.html#pantilthat.PanTilt
4. https://www.learnpython.org/en/Welcome
5. https://pinout.xyz/pinout/pan_tilt_hat
6. https://shop.pimoroni.com/products/pan-tilt-hat
7. https://mobiusstripblog.wordpress.com/2016/12/26/first-blog-post/
8. https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial
9. https://github.com/garmin/LIDARLite_Arduino_Library
10. http://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf

4 Pre-lab

None.

5 Laboratory

Save all code, data, and solutions as you will need them to complete the final lab report.

5.1 Requirements

Bring your raspberry pi (RPI) so that you can develop your code. You can grab a PanTilt hat to practice your scanning routine. Currently there is only one LIDAR Lite sensor, so you must coordinate with other groups to test out your code.

5.2 Procedure

1. Measure the unknown parameters of the scanner.
2. Derive the transformation equation of the scanner to calculate the x, y, z coordinates of the LIDAR measurements.
3. Develop code to control the PanTilt hat and the LiDAR Lite V3 sensor (both of which use I2C to communicate).
   a. Go to http://docs.pimoroni.com/pantilthat/index.html#pantilthat.PanTilt to see how to control the PanTilt hat in python and translate the code into a C++ class which inherits your I2C class. (Ignore functions that are solely concerned with the light and RGB lights).
   b. Go to http://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf and create a LidarLite class that inherits your I2C class and can control and read data from the Lidar Lite sensor.
   c. Create a Scanner application class that inherits your LidarLite and PanTilt classes.
      i. Use https://en.cppreference.com/w/cpp/io/basic_fstream to format and save your data (x, y, z coordinates of each measurement) in a text file.

    ii. Create a scanning routine which can scan an object with enough resolution that you can clearly identify it from the reconstructed point cloud.
    iii. Implement your derived transformation equations to obtain the (x, y, z) coordinates of each measured lidar point in your scanning routine.

4. After developing your code, select a recognizable object and implement your scanning routine, saving your data in a text file.
5. Export your text file to MATLAB and write a MATLAB script to extract and plot your data in a point cloud as shown in the previous figures.
6. Demonstrate your working solution to the GTA.