**README**

import numpy as np

import pandas as pd

left\_sensor = … (location on computer)

right\_sensor = … “ “

sensor1\_data = pd.read\_csv(left\_sensor)

sensor2\_data = pd.read\_csv(right\_sensor)

I then use the read\_csv() function from the pandas library to read in the sensor data from the csv files and store them as dataframes sensor1\_data and sensor2\_data.

sensor1\_pos = sensor1\_data[['pose.position.x', 'pose.position.y', 'pose.position.z']].values

sensor1\_quat = sensor1\_data[['pose.orientation.x', 'pose.orientation.y', 'pose.orientation.z', 'pose.orientation.w']].values

sensor2\_pos = sensor2\_data[['pose.position.x', 'pose.position.y', 'pose.position.z']].values

sensor2\_quat = sensor2\_data[['pose.orientation.x', 'pose.orientation.y', 'pose.orientation.z', 'pose.orientation.w']].values

I then extract the position and quaternion data from both sensors. The [['pose.position.x', 'pose.position.y', 'pose.position.z']] and [['pose.orientation.x', 'pose.orientation.y', 'pose.orientation.z', 'pose.orientation.w']] syntax is used to select the columns containing position and quaternion data, respectively. The .values attribute is then used to convert the selected columns into numpy arrays.

calibration\_matrix = np.array([[0.0, 0.0, 1.0, 0.0],

[-1.0, 0.0, 0.0, 0.0],

[0.0, -1.0, 0.0, 0.0],

[0.0, 0.0, 0.0, 1.0]])

I then define the calibration matrix between the two sensors. This matrix represents the transformation required to convert the data from sensor 2 to sensor 1.

transformation\_matrix = np.dot(np.linalg.inv(calibration\_matrix), np.array([[1, 0, 0, -sensor1\_pos[0][0]],

[0, 1, 0, -sensor1\_pos[0][1]],

[0, 0, 1, -sensor1\_pos[0][2]],

[0, 0, 0, 1]]))

I then calculate the transformation matrix required to convert the data from sensor 2 to sensor 1. This is done by taking the inverse of the calibration matrix and multiplying it with a 4x4 translation matrix that translates the position of sensor 2 to the position of sensor 1. The np.dot() function is used for matrix multiplication.

sensor2\_rot = sensor2\_quat[:, :3]

sensor1\_quat = np.dot(sensor2\_rot, calibration\_matrix[:3, :3].T)

I extract the rotation matrix from the quaternion data of sensor 2 by selecting the first three columns and then multiply this rotation matrix with the transpose of the first three rows and columns of the calibration matrix. The resulting quaternion is the quaternion of sensor 1.

sensor1\_pos = np.dot(transformation\_matrix, np.concatenate((sensor2\_pos.T, np.ones((1, sensor2\_pos.shape[0])))))

Finally, I use the transformation matrix to convert the position data of sensor