ROB 101: Computational Linear Algebra
Project #1 Map Building from LiDAR Data

Tribhi Kathuria, Maani Ghaffari, and Jessy Grizzle
University of Michigan Robotics Institute

September 17, 2020

This problem set counts for 20% of your course grade. You are encouraged to talk at the conceptual level with fellow students on Piazza, but you should complete all work individually and you should not share any non-trivial code or solution steps. If you have any doubts about what you can share, please check with one of the instructors or assistants. **We ask that you only discuss the project over Piazza so that everyone has access to the same information. That seems fair enough, right?** See the ROB 101 Canvas Site for the full collaboration policy.

Submission Instructions

The submission involves only code submission that should be done through Illumidesk, just like we do for your Julia Homeworks.

Notebooks vs. PDF

The notebooks are meant to be self-explanatory, and this PDF should be looked at as a supplement, providing extra information that is not in the notebook, and might help better understand the problem.

Objective

The objective of this course is to get you to deal with huge amounts of data and manipulate it in a meaningful manner using Matrix algebra. As you progress through the Jupyter notebook, you will learn to code and appreciate different kinds and forms of *linear transformations*. So let's dig in!

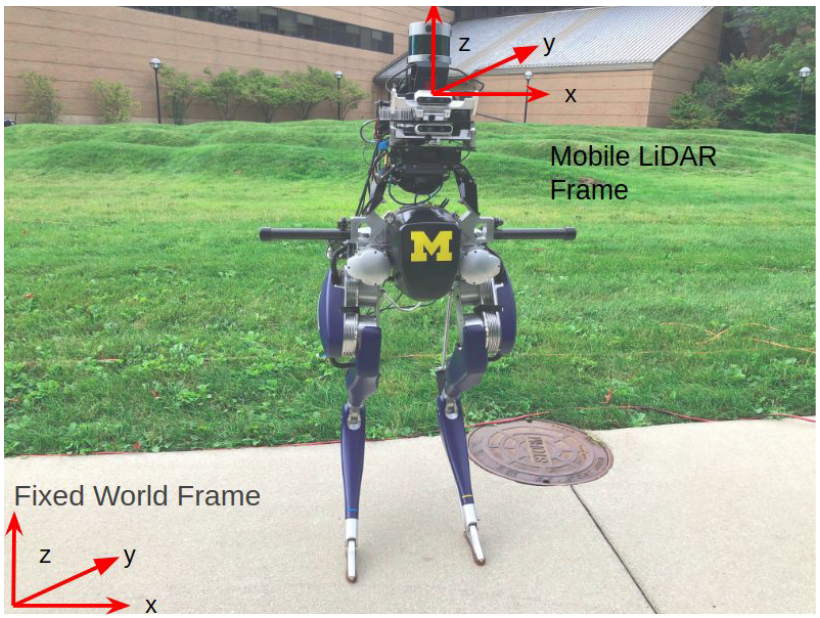


Figure 1: The image shows two frames used by the Cassie bipedal robot, one that moves with the robot and one that is fixed in 3D space. A LiDAR is a laser-based device for measuring the position of objects around a mobile robot. The LiDAR is mounted to the robot. As the robot moves, so does the LiDAR. Each data scan collected by the LiDAR, which happens 10 times per second (10 Hz), is with respect to a different position in 3D space, namely the LiDAR's position at that exact moment in time. To assemble the individual data scan into a global "map" that can be used for navigation, the LiDAR points must be transformed into a common fixed frame called *world or global frame*.

1 Problem Statement

For this project you are given data collected on the Cassie Blue bipedal robot during an experiment on the North Campus Grove and are asked to transform the data for building a map and visualizing it using Julia. The underlying work you will do is very similar to what is done in real-time[1] on Cassie in order to build a map for autonomous navigation. You may wish to view the video https://www.youtube.com/watch?v=pNyXsZ5zVZk to see mapping and navigation done in real-time. Yes, this is very similar to what is done by AVs (Autonomous Vehicles). One difference is that an AV has 200 Kg of electronics in its trunk to process all the data, while Cassie's entire autonomy package weighs in at 9 Kg and runs off a hobbyist LiPo battery.

2 Understanding LiDAR Point Cloud Data

LiDAR stands for Light Detection And Ranging and can be thought of as "light-based radar". It is an optical remote-sensing technique that uses laser beams to scan the surfaces of objects, producing highly accurate x, y, z measurements of their position with respect to the LiDAR. In a typical LiDAR sensor, light pulses are emitted from a bank of rapidly firing lasers. The LiDAR on Cassie has a bank of 32 laser beams! You can imagine a LiDAR functioning similar to a strobe light on a camera (when you use flash to snap a photo), except the light pulse is much shorter in duration and lower in intensity. Also, its frequency is chosen to be in the infrared where it cannot harm humans or animals' eyes.

[1] Real-time means the computations are done quickly enough on the robot that they can be used almost immediately. This is opposed to "off-line" where you collect data on the robot and then do the processing on a desktop in the lab. You are clearly doing offline data processing, which is easier, because there are no computation time requirements.

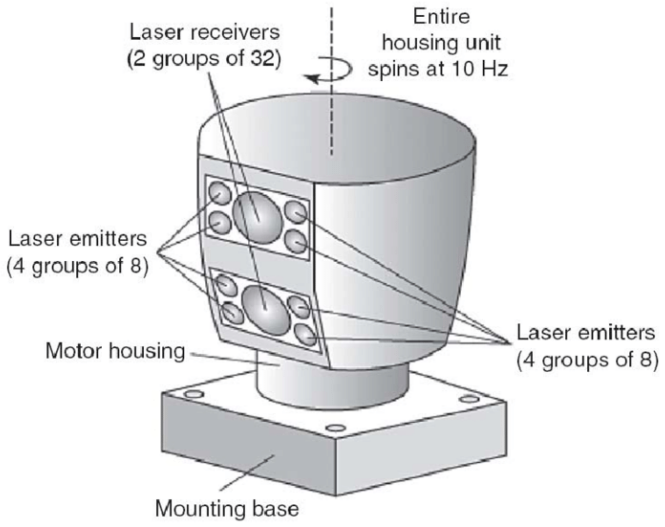


Figure 2: An example of a 3D LiDAR.

As the light pulse "flies" from the laser, it will reflect off anything it encounters, such as the ground, buildings, tree branches, humans, or other robots. The reflected light pulse is then measured by the LiDAR sensor where it does the following things:

- It measures the "time of flight" of individual light pulses, that is, the duration of time from the sending of the light pulse and the return of its reflection to the LiDAR. The light travels to the object and back to the LiDAR, hence it travels twice the distance of the object from the LiDAR. If you call the time of flight δT and the speed of light, c, then the distance d from the LiDAR to the object is $\frac{1}{2}\delta T c$. For an object that is 10 meters away (33 feet), the time of flight is 6.66710^{-8} seconds or approximately 67 nanoseconds. Hence, the electronics to measure such small numbers with high accuracy is quite amazing in and of itself.

- The sensor also measures and records the "intensity" of each returned laser pulse. The intensity is roughly the "brightness" of the returned pulse, or more precisely, the "energy" in the returned pulse. Dark-colored and matte surfaces tend to absorb more of the energy, while light-colored and smooth surfaces tend to reflect more of the energy. Roboticists often use the intensity to add "color" to LiDAR "images" (that is, to plots of LiDAR data) as in Fig. 3.

The LiDAR does all of its measurements as if it were the "center of the universe", meaning, it reports the data in its own coordinate system. As the robot walks, the LiDAR moves with it, and hence the position of the LiDAR's coordinate system is moving. The LiDAR leaves you to deal with that motion! Hence, to join two successive LiDAR scans in a proper manner, we must understand how the robot has moved.

The transformations that you will be given involve "translations" and "rotations" of points in $\mathbb{R}^3$, that is, 3D space. The *translations* account for the motion of every point by a fixed amount in x, in y and z, which you can think about as moving a coffee cup[2] in 3D space, with the handle always facing the same direction. The *rotations* account for relative angular motions of groups of points, which you can think of as turning the coffee cup over and pouring its contents on the floor.

[2]The coffee cup is our way of making a point cloud seem like a physical object that you can imagine translating and rotating in space.

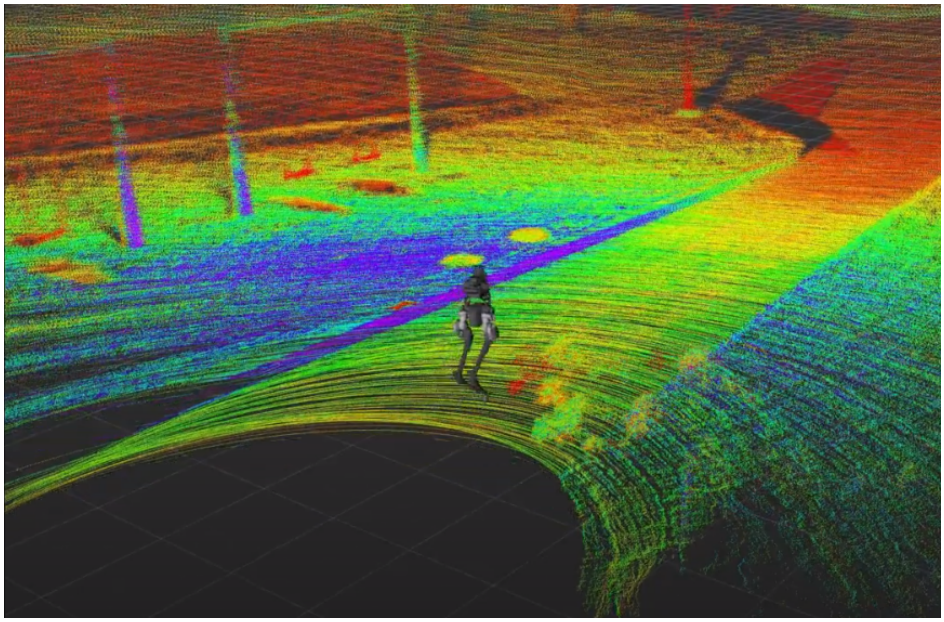


Figure 3: Sample point clouds as seen by Cassie robot. The different colors are based on the intensity the scan returned. The higher the intensity the higher up in the rainbow is the point

Each LiDAR scan contains anywhere from hundreds to tens of thousands of four-tuples (x, y, z, I)- where (x, y, z) are coordinates at which the laser pulse reflected off an object and I is the intensity of the reflected pulse. The collection of such points is a *point cloud*. In the LiDAR data provided to you in the form of CSV (Comma Separated Values) files, the first three columns are X,Y,Z data of each point and the last column is the intensity data of each point.

3 Your Tasks

This first project is broken down into quite small tasks to make it more manageable. Also, you will be given data against which you can check your solutions as you progress through the Project.

Task 0 (0 Points)

Obtain from the Illumidesk Assignments Tab the Jupyter notebook titled "project_1.ipynb" and the folder containing the CSV files called "data" you will need for Project 1. You will see that Project 1 has been released and is ready to be fetched. The Fetch button will then change to a blue Submit that you can use to submit your project. In fact, you are encouraged to keep saving your progress and submitting incrementally so you're not as pressured near the deadline!

Task 1 (10 Points) 2D Square in the Plane

For this Task, we will start with a (green) square defined in the notebook and apply different transformations to arrive at the various objects and their positions shown in Fig. 4. The different "squares" represent the different steps of the transformations we will utilize to go from the initial green square to the final blue diamond shown in Fig 4.

Working from the original green square we should have the following sequence of Transformations intuitively:

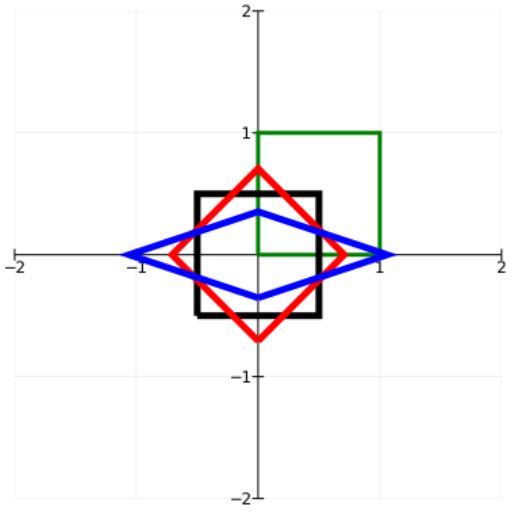


Figure 4: The different squares and the final diamond for Task 1.

1. **Translation:** Get the black square, by Translating your x and y co-ordinates towards the origin. (Part A, Part B)
2. **Rotation:** Rotate the points in black square $\pi/4 rad$ to get the red square. (Part C)
3. **Scaling:** Scale your points in x (by 1.5) and y (by 1.5) axis to get the squished Blue square that now looks like a rhombus. (Part D)

Finally, you should be able to generate the plot given in Fig. 4. In different parts of this task, you *transform* the square (a collection of four points) into all the different shapes we see by applying a *linear transformation*, which is done by matrix multiplications.

However, if you ever ran a block twice or did not follow the sequence, start again to reinitialize your `points` vector. This problem is resolved in Part E, which takes us from the green to the blue square in a single step using *affine transformations*.

Task 2 (2 points) 2D Point Clouds

Follow the assignment in the notebook to plot a 2D point cloud that has been distorted, and fix it using the given transformation matrix. The data is in `data/question.csv` as seen in Fig. 5. The different points are arranged in a N total columns. Figure 6 describes how the data is arranged. We use the function `readdlm` and pass the argument to the path of the CSV file, i.e. `data/question.csv`. We then extract the x,y coordinate of the point cloud and the also the color data in a separate variable that is used to get a colorful plot for our point cloud.

Follow along with the instructions in the notebook to get a final plot, where you should be able to read the text in the picture!

Task 3 (8 points)

Now for the real problem! We are not going to go into much detail over building Transformation in 3D like we did for the 2D case. In case of LiDAR scans our Kalman Filter algorithm helps us essentially generate a Rotation and Translation matrix for our Transformations, that will be directly used for your 3D Point clouds.

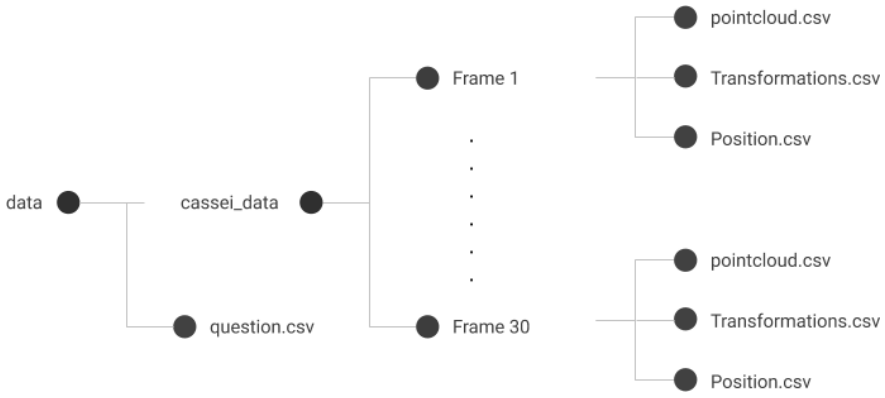
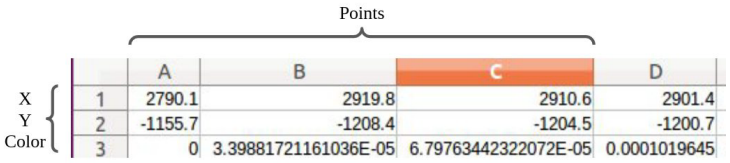


Figure 5: Arrangement of Data in the zip file.



	A	B	C	D
1	2790.1	2919.8	2910.6	2901.4
2	-1155.7	-1208.4	-1204.5	-1200.7
3	0	3.39881721161036E-05	6.79763442322072E-05	0.0001019645

Figure 6: Data arrangement for a 2D point cloud

Understanding how the data is arranged

The data has been organised as different csv files for different Frames of the video that we will finally generate! Each of these folders has data generated over 1 second of data collection. For a LiDAR that works at 10 Hz (10 scans every seconds), it means combining ten point clouds and saving them into a csv file called, `"pointcloud.csv"`.

Point Cloud Data: Each column in the `"pointcloud.csv"` consists of points specified by their (x,y,z) coordinate in the LiDAR frame, that is to say in the view of LiDAR. The data is arranged such that the first three rows give the point information and the last row contains the intensity information that will be used to plot the LiDAR data with the colors of the rainbow scaled according to the intensity, which is the standard of their representation.

Robot Pose: We also give you the pose of the robot at the instance of data collection, that will be plotted to see an estimate of where our physical robot is in the world frame. The robot pose data is saved in the `"position.csv"` file. The first column has the (x,y,z) coordinate in the World Frame and the second column has the orientation of the pose which is based on where the robot is looking.

Transformation Data: Finally, we give the Rotation (R) and Translation (t) matrices in the `"Transformation.csv"` file. These should be used to get the Affine Transformation to be applied to LiDAR point cloud data, to bring it in the World Frame.

Fig. 5 gives the arrangement of data in the different csv files that we will use. The "data_parser" function written for you in the notebooks will parse this data, which can be directly saved into arrays that will be manipulated throughout this task. You do not need to worry about the file arrangement described above as such. As long as you send the right time interval, which can be between 9 - 13 seconds, as an argument to the function it will parse it for you.

Ask any questions on Piazza. Good luck!