Dear Part-IV students,

I hope your exams Int Ill and you're enjoying your break.

Just a reminder that the **Mid-year video on results progress is due on 22 July (Friday)**. Here's more information on preparing this video report:

* You must submit one video presentation (3 minutes per student, total 6 minutes per team). It should contain only technical content (e.g., mathematical formulations, algorithms, designs, analyses, experimental results etc.). The project scope, objectives, and literature review should not be included.
* Each student should describe their own progress in their 3-min block.
* Single student projects could use 3min or the entire 6min.
* Only one video file (6min) is to be submitted per team.
* Both students is to submit the 6min video file individually, meaning each team will submit the video twice.
* Please ensure that the 6min video is less that 100MB.
* In case you need it, a tutorial on how to use PoIrPoint for video presentation is shared under [Modules,](https://canvas.auckland.ac.nz/courses/75843/modules/217765)
* Submission is via Canvas [Mid-year Progress Video.](https://canvas.auckland.ac.nz/courses/75843/assignments/276182)

I will be in touch to schedule the remaining two lectures on preparing the final report and for Display Day.

Regards,

Reza

**Mid-Year Progress Video Checklist**

**mmWave Sensor**

* Raw data from mmWave
* Data visualisation on texas instruments site
* Data parser and python file
* Data points exported to excel
* MATLAB grouping algorithm visualisation of 3D points
* MATLAB SLAM (if I can get some output)

**Robot**

* Teleop manual movement
* Teleop control with our stub inputs
* Maybe show Lidar data if I run out of things to include (just to show that I know how it works and that I are aiming to do the same except with the mmWave sensor)

**Next Steps**

* Any refinements to the algorithm?
* Anything wrong with the algorithm?
* Using algorithm output and feeding it to the robot

Talk about:

* Mmwave demo short video
* Describe parser & what I had to change to make it work with our device

**Script**

* Hi, we are project 102 and this is our mid year progress video on developing the Millimetre-wave based SLAM for the turtlebot running the robot operating system. In this progress video, we will be outlining what the sensor raw data looks like, how we parse the data through the COM ports, and how this data is visualised. We will also go over a few of the algorithms we’ve explored and what direction we plan to pursue.
* A simple demo has been prepared by the developers of the millimetre sensor, Texas instruments, which allows us to visualise data points which the sensor is reading from its surroundings. **[Insert video of demo]** Here you can see a variety of parameters such as Doppler values, coordinates, and the number of static and dynamic detected objects. **(Describe plots & movement)**
* When reading the data directly from the sensor, I noticed that it was not in the correct format and was updating too fast for us to see **[insert clip].** The next step was to parse the jiberish-looking data from the COM ports of the laptop. After intensive research, I settled on using an open source parser made in Python for the IWR1443. The parser stores the sensor's parameters in a dictionary object for easy access. It was also modified to visualise the x and y coordinates from the sensor **[insert image]**.
* I chose a Python parser because I couldn't find any other 3D parsers for the IWR1443 by texas instruments. After viewing the data points, I tried a few different approaches to manipulate and process the data. I chose to do this in MATLAB as it was a more familiar language. To do this, the data points were exported to a csv file to access within our MATLAB scripts. In this clip, we have placed the sensor on a chair to showcase how datapoints from the parser look as we turn the chair clockwise. You can see that the data points are slowly shifting towards that direction as it picks up information while the sensor is being moved.
* My first approach was to manually calculate the distance between two data points in 3D space for all coordinates in order to detect clusters **[insert distance algo]**. The reasoning behind detecting clusters was to make the sensor visualisation easier to interpret. I noticed that even when the sensor was in front of an empty wall, the sensor detected over 26 objects. The clustering was intended to group these points together and find the average of them all. However, this proved to be time consuming and not effective at detecting clusters.
* My second approach was to use *kmeans* clustering in MATLAB **[insert kmeans formula and our test image]**. This built-in function calculates the number of clusters given a set of arrays for the x, y, and z coordinates. I noticed that this was excellent at clustering the points together, but the downside is that I need to know how many clusters need to be formed first. This is very often not possible to achieve when streaming real-time data from the sensor. Another issue I encountered was that some points were incorrectly identified as part of the wrong cluster, shown by the green data point close to the blue cluster. I decided to scrap this idea and migrate to exploring ways to visualise coordinates in 3D space.
* I was simultaneously exploring the 3D point cloud approach. We found this example on MATLAB which performed SLAM using 3D Lidar Point clouds **[show example]**. I wanted to explore whether this approach would also work with mmWave sensor data. The 3D point cloud algorithm takes a cell array of n-by-3 matrices where n is the number of points and each column represents the x,y,z coordinates which are obtained from our parser.
* This input data and a set of robot parameters and thresholds are used to create a 3D Pose graph and occupancy map. The algorithm in this example performs a loop closure estimation within a specified radius. When the scans are accepted, a pose graph is created. Outlier points in the trajectory are removed based on the specified parameters. These poses are added to the occupancy map and displayed on a graph. A relative pose is estimated based on the current pose and previous pose. The issue I faced with this approach was that there weren’t enough data points because the mmWave data is less dense than a LiDAR sensor output. So using a SLAM approach with 3D LiDAR point cloud data does not work synonymously with mmWave point clouds.
* We now understand how mmWave Point Clouds are formed and have a rough idea of how to implement a SLAM algorithm but we need to develop an algorithm which is capable of performing SLAM on mmWave point clouds. We wanted to continue our research and development in Python because the parser we are using is in Python and continuing further in MATLAB would introduce issues with real-time data processing by using two different languages concurrently.
* Currently we have implemented visualising the mmWave sensor point clouds in python using the *open3D* Library. This clip shows what the data looks like when we have some objects at different depths in front of the sensor. The points can be seen clearly in 3D space in terms of their x, y, and z values. We still need to explore how to manage the sensitivity of the sensor to produce a clearer visualisation that we can work with. We want to keep open3D graphs *persistent* so the points it sees are saved onto the plot.
* Our next steps involve using the 3D point cloud data to develop a SLAM algorithm for the robot to operate with. We would also like to migrate away from using the keyboard to control the robot, or modify the *teleop* ROS program to work based on our algorithm output. This concludes our mid-year progress video on our project, thank you for listening.