REPORT

Lincoln MKZ Autonomous Project

Date: - 11th May 2024

University at Buffalo The State University of New York

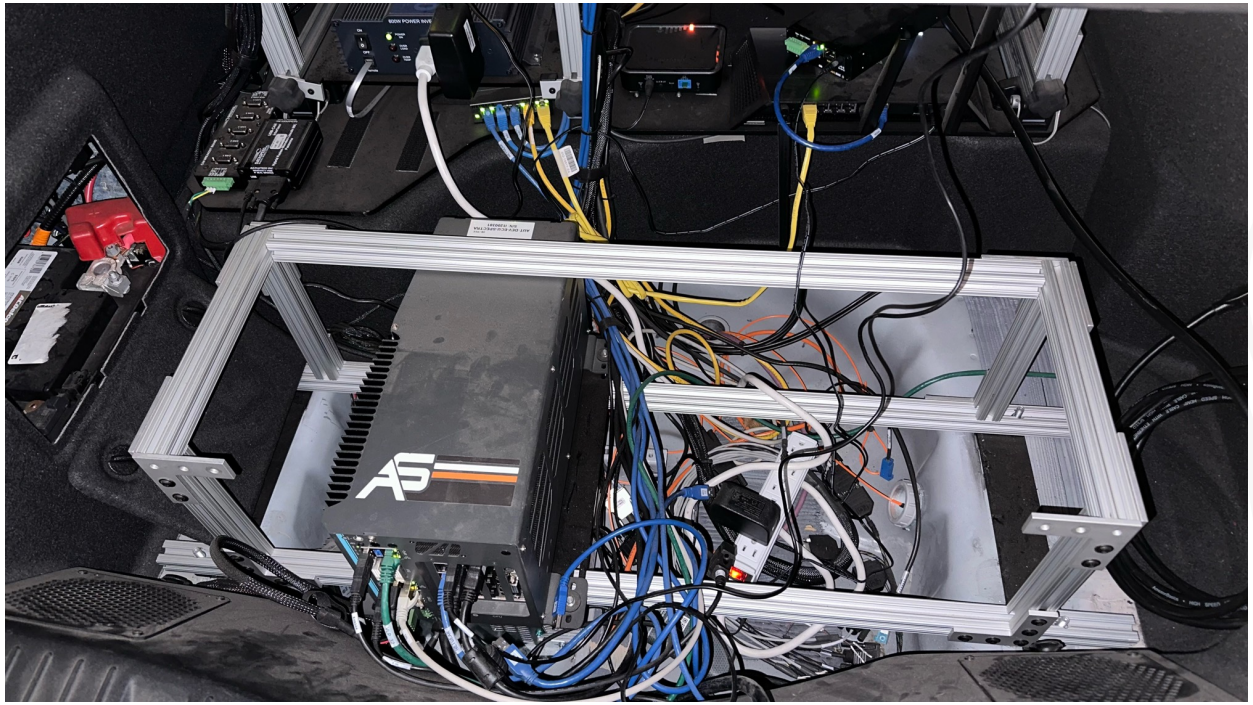
University at Buffalo The State University of New York

Advisor: - Dr. Chunming Qiao

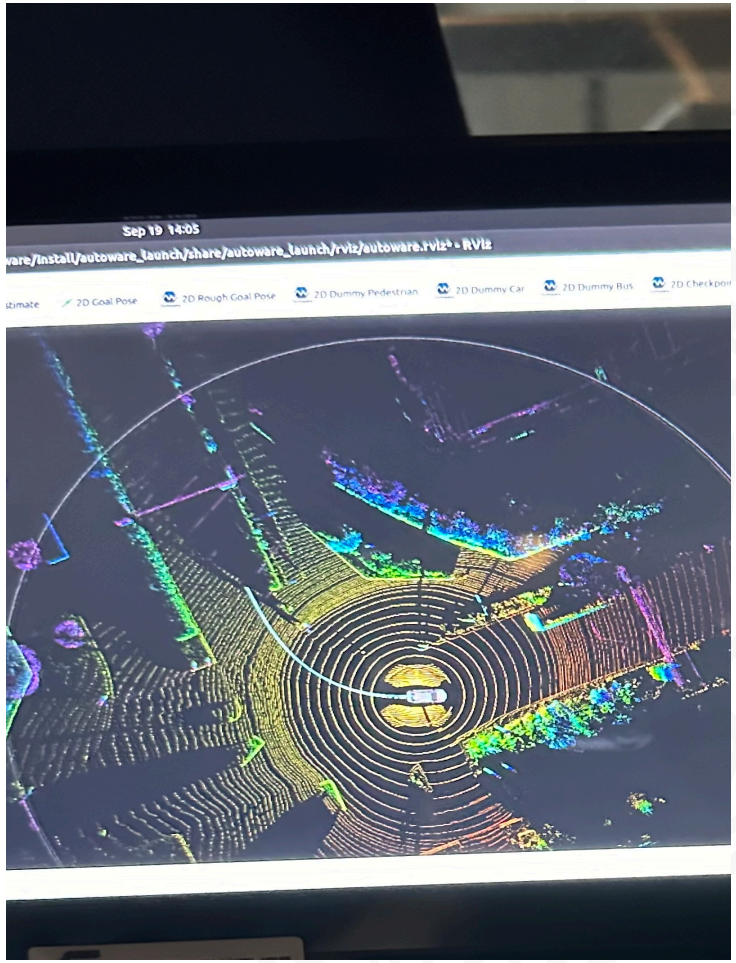
In charge: - Stevens Korzelius

Members: - DeeKay Goswami and Daniel Vadranapu

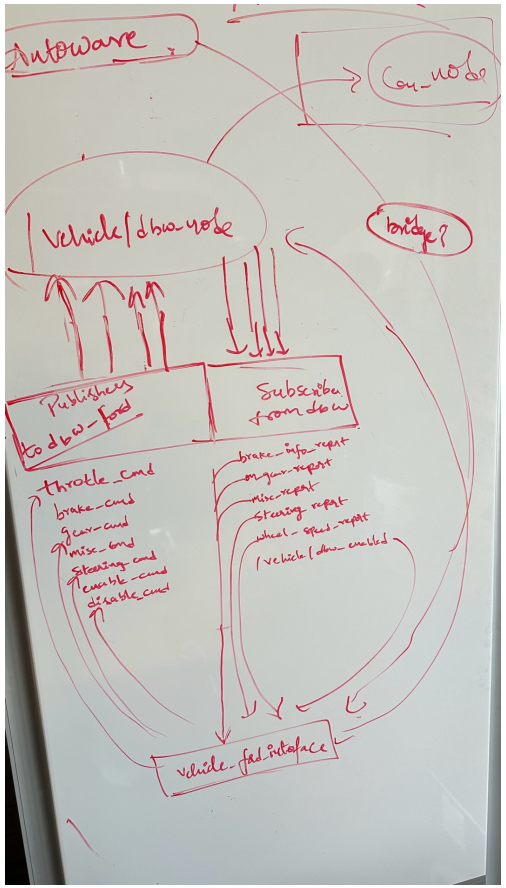
- We worked on getting the vehicle's system up & ready for operation. Installed Latest Ubuntu and built Autoware.Universe from source.

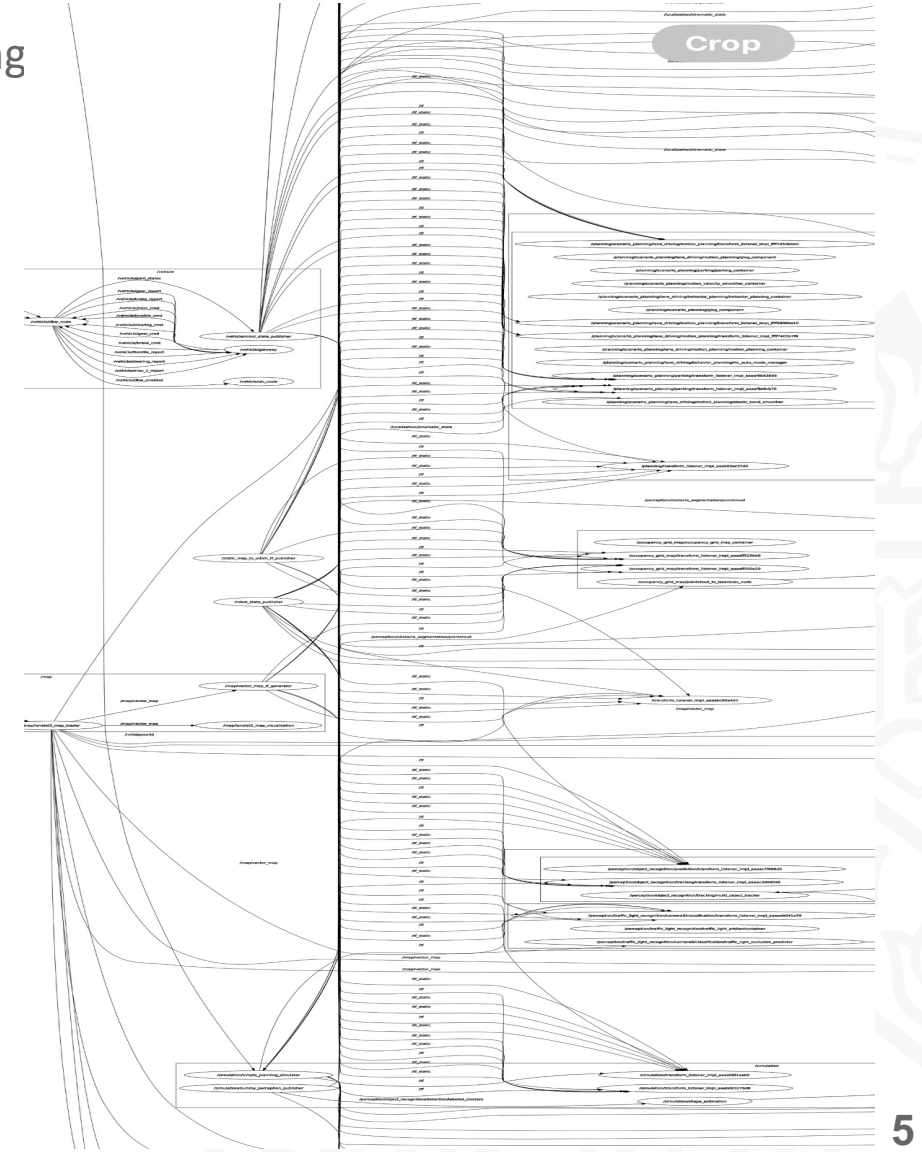


- We tested the system by running the planning simulation and it was performing very well.
- Set 2D pose and goal point in rviz and initiated autonomous functionalities to drive the car on the planned path and reaching destination.
- Noted down all the “ROS2 Topics” in action.



- We studied the Autoware.Universe, it's packages and started working towards the root cause that why the car is not getting controlled by the latest version of autoware in Ubuntu 22.04 based on ROS2.





- We built a ROS2 based C++ bridge between autoware and DataSpeed's DBW controller to listen to autoware commands and control the car autonomously directly through autoware. GitHub!

Click here for code.

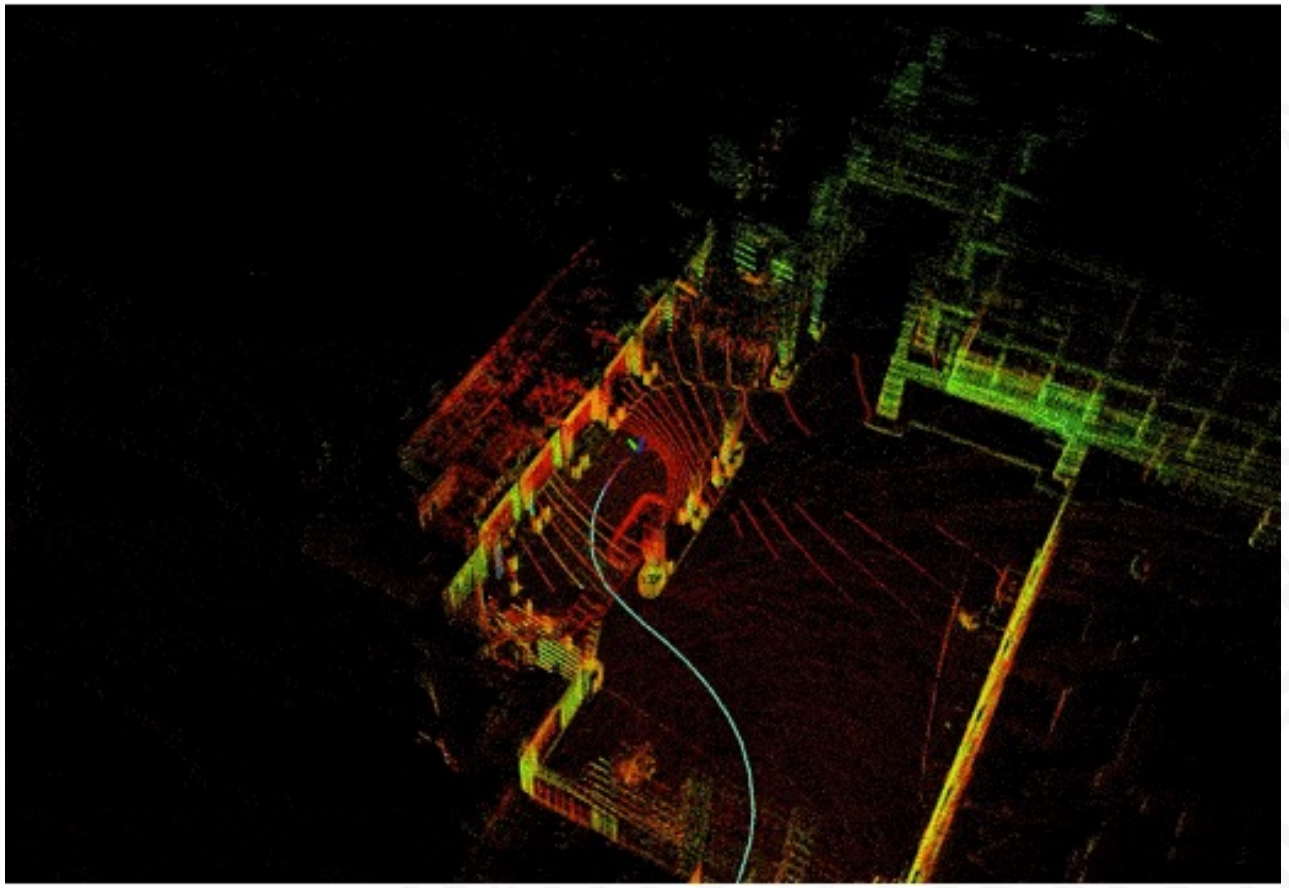
```cpp
// Author: DeeKay Goswami

#include "rclcpp/rclcpp.hpp"
#include "autoware_auto_vehicle_msgs/msg/gear_command.hpp"
#include "autoware_auto_control_msgs/msg/ackermann_control_command.hpp"
#include "autoware_auto_vehicle_msgs/msg/gear_report.hpp"
#include "autoware_auto_vehicle_msgs/msg/steering_report.hpp"

#include "dbw_ford_msgs/msg/gear_cmd.hpp"
#include "dbw_ford_msgs/msg/brake_cmd.hpp"
#include "dbw_ford_msgs/msg/throttle_cmd.hpp"
#include "dbw_ford_msgs/msg/steering_cmd.hpp"
#include "dbw_ford_msgs/msg/gear_report.hpp"
#include "dbw_ford_msgs/msg/steering_report.hpp"
#include "sensor_msgs/msg/joy.hpp"

class Lincoln_MKZ_Bridge : public rclcpp::Node
{
public:
    Lincoln_MKZ_Bridge() : Node("Lincoln_MKZ_Bridge")
    {
        //Output to DBW Node
        gear_publisher_ = this->create_publisher<dbw_ford_msgs::msg::GearCmd>("/vehicle/gear_cmd", 10);
        throttle_publisher_ = this->create_publisher<dbw_ford_msgs::msg::ThrottleCmd>("/vehicle/throttle_cmd", 10);
        brake_publisher_ = this->create_publisher<dbw_ford_msgs::msg::BrakeCmd>("/vehicle/brake_cmd", 10);
        steering_publisher_ = this->create_publisher<dbw_ford_msgs::msg::SteeringCmd>("/vehicle/steering_cmd", 10);

        //Output to Autoware
        gear_report_publisher_ = this->create_publisher<autoware_auto_vehicle_msgs::msg::GearReport>("/vehicle/status/gear_status", 10);
        steering_report_publisher_ = this->create_publisher<autoware_auto_vehicle_msgs::msg::SteeringReport>("/vehicle/status/steering_status", 10);

        //Input from DBW Node
        gear_report_subscription_ = this->create_subscription<dbw_ford_msgs::msg::GearReport>(
            "/vehicle/gear_report", 10,
            std::bind(&Lincoln_MKZ_Bridge::gearReportCallback, this, std::placeholders::_1));
        steering_report_subscription_ = this->create_subscription<dbw_ford_msgs::msg::SteeringReport>(
            "/vehicle/steering_report", 10,
```

- We tested the vehicle on-road and resulted in working of “Gear” and “Throttle” of the vehicle successfully. However, car was unable to apply “Brake” and turn through “Steering”.

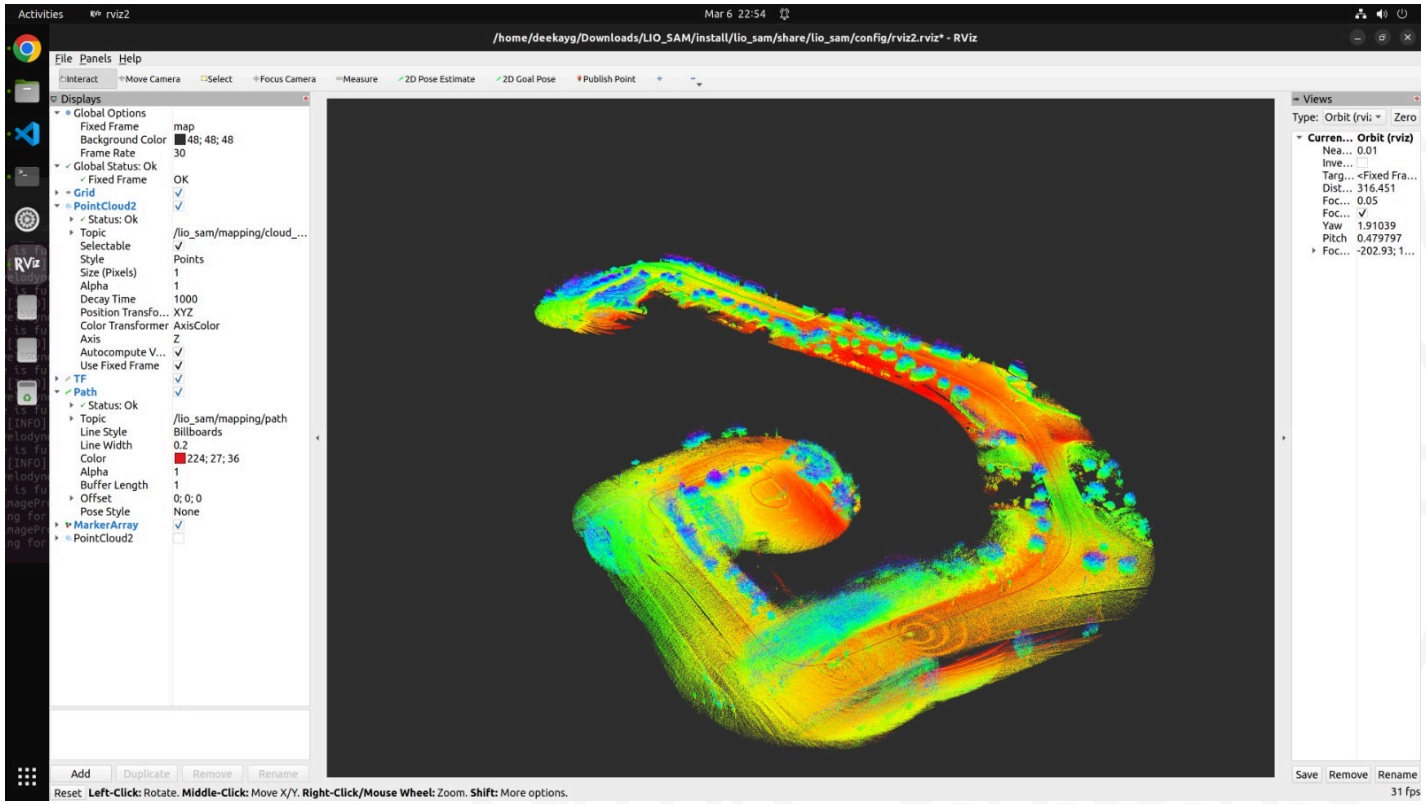
- We tried numerous approaches to rectify the problem between the bridge and car's DBW system.
- Additionally, we also worked on vehicle's GPS component and built ROS2 based Novatel GPS package.

- Analyzed the throttle outcome from the autoware.
- Worked on acceleration part to get “Brake” and “Throttle” working.
- Integrated **ULC** functionality into our **Universe_DBW_Bridge** and tested on-road and found that acceleration and deceleration/brake was working expectedly.

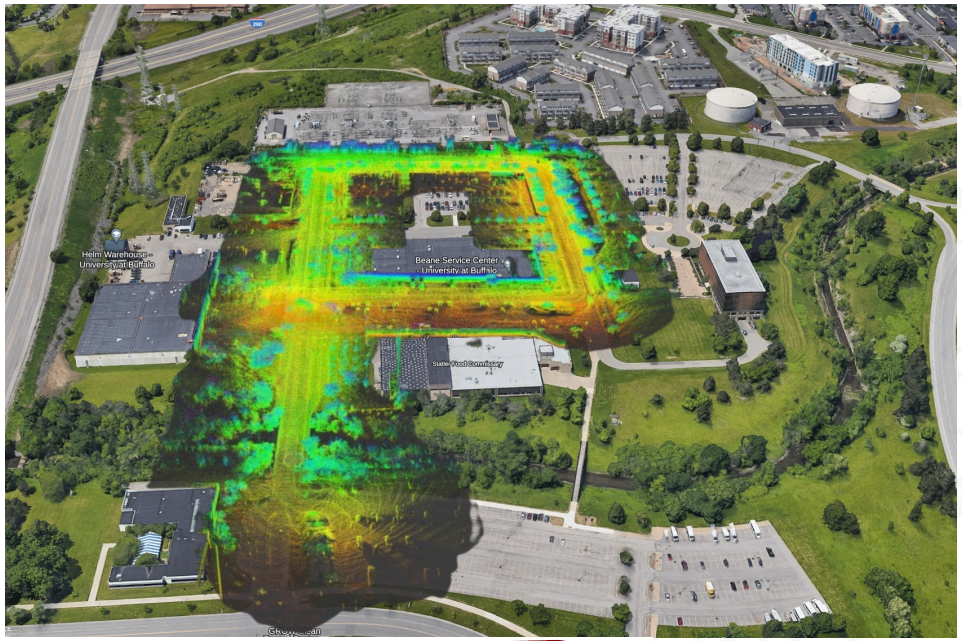
- Worked on integrating LiDAR and explored the available packages.
- After integration, we started exploring several **SLAM** Algorithms to successfully build a campus map.
- We read several SLAM papers like **FAST-LIO**, **LEGO-LOAM, LIO-SAM** etc.
- Only **LIO-SAM** is available in ROS2 so we directly implemented that.

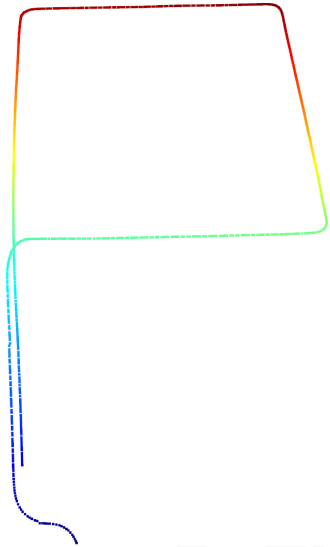


- We started recording the ROS2 bags to later utilize for campus mapping using ROS2 based LIO-SAM algorithm.
- After building the map, it was found that algorithm was having difficulties in consistently matching LiDAR and IMU frames.
- This led to failure in loop-closure which is a crucial part as well as step in building any map.

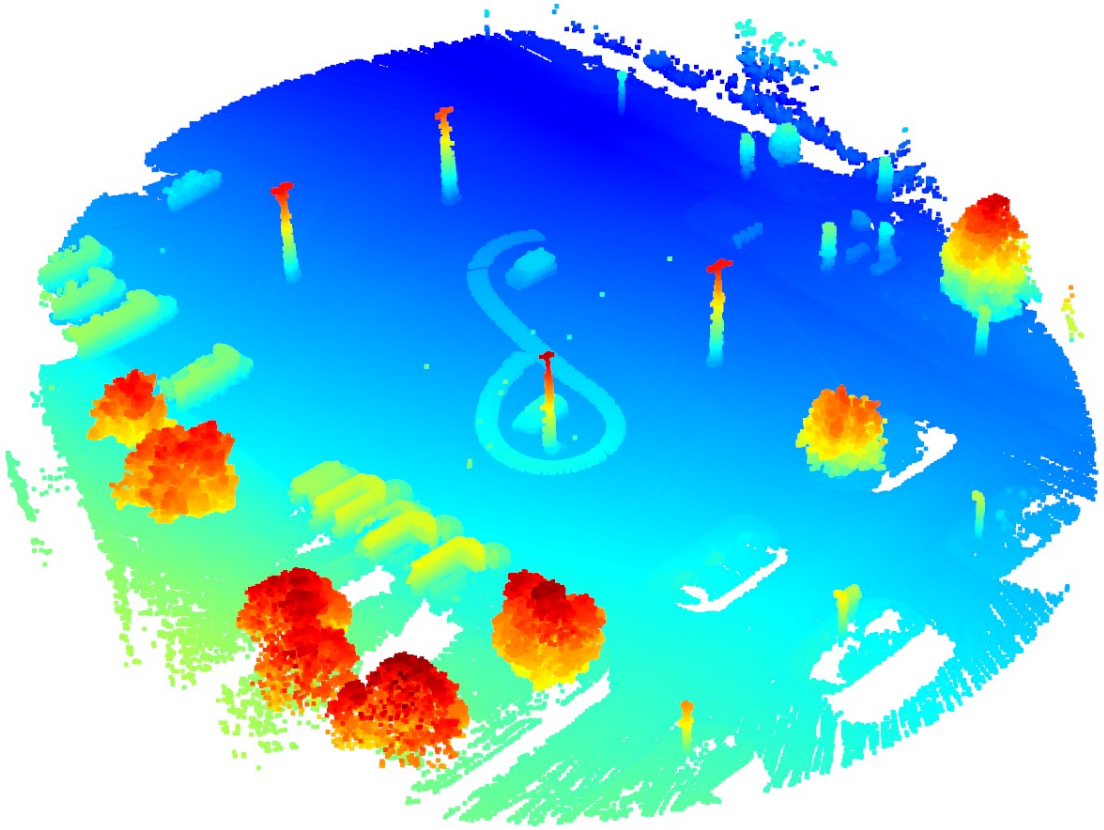


- We changed the location for mapping and recorded the bag on different spot of service road.
- Although mapping went good but after carefully analyzing the map, we observed that the **trajectory** was slightly **dragged** off which was due to **Z-axis error** while mapping.
- Hence, to verify this, we **overlapped** the map on top of the google map using google earth. As you can see from the image on the right, **start** and **end** points are matching with the location but trajectory is not straight.



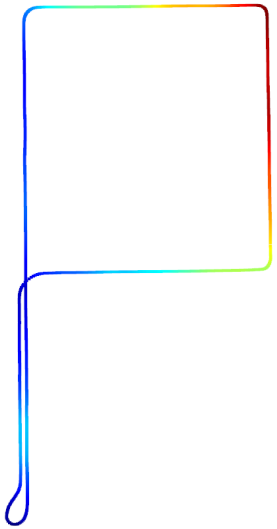


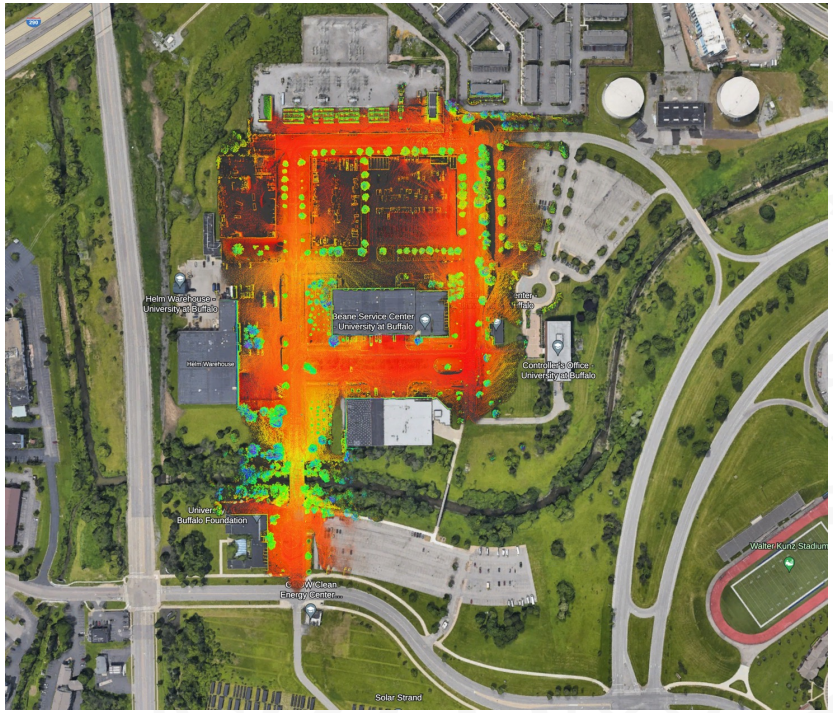
- To resolve this inconsistencies, **Sensor Calibration** is required.
- After exploring tons of calibration techniques and algorithms, we found that forming an **8_Loop** yields better results.
- **Why 8_Loop?** Because an 8_Loop ensures comprehensive coverage of data across all axes of sensor operation, facilitating accurate measurements in multi-dimensional spaces by capturing points at various positions within the sensor's range.
- Image on right shows final **NDT_Map PCD** with proper alignment of features like pole, trees, cars etc after calibration.



- Image on the right shows final calibration parameters.
- **P_LinI** and **p_IinL** are extrinsic sensor parameters from/to LiDAR and initial frame.
- **Euler** values are degrees in radians which later used to calculate **Rotation Matrix.**

```
====== Iteration 13 ======
        [Refinement] costs 779.95 ms
        [DataAssociationInRefinement] starts ...
get refined map done
Plane type  :214 264   4; Plane number: 482
Surfel point number: 4871
        [DataAssociationInRefinement] costs 8999.15 ms
        [Refinement] starts ...
Ceres Solver Report: Iterations: 8, Initial cost: 4.659717e+02,
 Final cost: 4.659717e+02, Termination: CONVERGENCE
Iteration, 13
8_Loop
         P_LinI, 1.905, -1.926, -1.311
     euler_LtoI, 2.75, -1.87, 75.87
         p_IinL, 1.471, 2.293, 1.278
     euler_ItoL, 1.14, 3.13, -75.85
=========================
segment id  : 0
    time offset, -0.0200
      g_refine, -0.04, -3.70
       gravity, -0.633, 0.006, 9.777
     accel bias, -2.5258, -1.4354, -7.7248
      gyro bias, 0.0000, 0.0012, -0.0454
      plane cov, 0.980, 0.012, 0.008
```





- These are the final mapping results after sensor calibration.
- Verified with google map, the trajectory is now straight due to successful loop-closure with **cm accurate** alignment.
- The next step involves launching all the required nodes, including Autoware, Vehicle Control Interface, Map Load (Vector Map from PCD) and localization node to perform autonomous driving goal.

Thank You