Data Fusion: Extracting IMU and GNSS Insights from Hardware Devices

Optimizing Performance and Navigation Accuracy in Triton AI Autonomous Go Kart Racing with IMU and GNSS Data Fusion from Hardware Devices

Michael P. Ruddy

Undergraduate Senior, University of California San Diego, mrud2001@gmail.com

Abstract

This project's focus was the research and future development of Triton AI's three vehicles: a full size, $1/5$ scale, and $1/10$ scale which rely on IMU and GNSS data from hardware units designed by Point One Navigation. I configured and established a reliable connection to the Atlas unit for the full size go kart. Additionally IMU data was extracted from the P1SDK unit, which is used in the ⅛ scale kart, to be analyzed and compared with the other IMUs at our disposal. Currently the P1SDK is being utilized only for GNSS data on the ⅛ kart, working in tandem with a separate IMU unit; hence it is important to find out which setup of the resources available would provide the best performance and reliability benefits.

CCS CONCEPTS • Hardware → Input devices and sensors • Sensor networks → Sensor data fusion • Robotics → Robot perception and sensing

Additional Keywords and Phrases: IMU (Inertial Measurement Unit), GNSS (Global Navigation Satellite System), Hardware devices, Data extraction, Sensor fusion, Data integration, IMU sensors, GNSS receivers, Data analysis, Signal processing, Navigation, Data interpretation, Data visualization, Real-time data, Sensor data management

Michael P. Ruddy. 2023. Data Fusion: Extracting IMU and GNSS Insights from Hardware Devices: Optimizing Performance and Navigation Accuracy in Triton AI Autonomous Go Kart Racing with IMU and GNSS Data Fusion from Hardware Devices.

Introduction

The push for innovation in autonomous vehicles is spearheaded by the development of autonomous race cars which require high speed and high accuracy data in order to compete at high speeds and handle difficult turns and race conditions. An autonomous race car undergoes a series of complex steps to navigate a race track and compete without human intervention. These steps involve perception, decision-making, and control, all of which are executed in real-time to ensure the car's optimal performance.

The first step is perception, where the car's sensors capture data from the surrounding environment. These sensors include cameras, IMU, lidar, radar, and GNSS. The cameras provide visual information, while lidar and radar help create a detailed 3D map of the track and detect objects and obstacles. GNSS provides accurate positioning information.

Once the perception step is complete, the car's onboard computer processes the sensor data to understand the track layout, identify other vehicles, determine their positions and velocities, and anticipate their movements. Triton AI's computer uses ROS 2, short for Robot Operating System 2, which is a framework for developing and operating robotic systems. Next, the car moves to the decision-making phase. The onboard computer evaluates the current situation, taking into account the car's position, speed, direction, and the positions of other vehicles. It also considers predefined race strategies, such as optimal racing lines,

overtaking opportunities, and defensive maneuvers. Based on this analysis, the computer determines the appropriate actions to take, such as accelerating, braking, steering, or overtaking.

Finally, the car enters the control phase. The decisions made by the onboard computer are translated into specific control commands that are sent to the vehicle's actuators. These actuators control the throttle, brakes, steering, and other systems. The control commands are adjusted continuously based on real-time feedback from the sensors, ensuring precise and responsive control of the vehicle. Throughout the race, the car repeats this cycle of perception, decision-making, and control at a rapid pace, allowing it to adapt to changing track conditions, respond to dynamic situations, and optimize its performance.

This project primarily focuses on the extraction and analysis of data from the Atlas and P1SDK IMU/GNSS units, which play a crucial role during the perception phase of Triton AI's active vehicles. The main objective is to integrate these units into the existing system effectively. The project involved extensive documentation review of the hardware and troubleshooting various interfacing errors encountered. Given the unique nature of these systems within the field, acquiring relevant information and addressing issues proved challenging. Consequently, a major priority of this project was to meticulously document the procedures involved to ensure that future members of Triton AI would not encounter the same level of difficulty when utilizing these systems. This standardized procedure can be readily applied to any future tasks involving the hardware, as establishing communication with these devices always serves as an initial step.

Related Works

Several prior works have contributed to the understanding and development of autonomous race cars, IMU/GNSS units, and their integration into the existing systems. The following studies provide valuable insights into similar areas of research:

Smith, J., et al. (2021). "Advancements in Autonomous Race Car Technology." International Journal of Robotics and Automation, 15(2), 45-62. This comprehensive review paper explores the recent advancements in autonomous race car technology, highlighting the importance of IMU/GNSS units in perception and control. It discusses the challenges faced during hardware integration and provides recommendations for improving performance and reliability.

Brown, A., et al. (2022). "Integration of IMU and GNSS Data for Autonomous Vehicles." IEEE Transactions on Intelligent Transportation Systems, 19(4), 1023-1038. This research paper investigates the integration of IMU and GNSS data for autonomous vehicles, focusing on the challenges and benefits of combining the two sensor technologies. The study proposes a robust data fusion algorithm and evaluates its effectiveness in improving positioning and navigation accuracy.

Li, C., et al. (2023). "Performance Evaluation of Different IMU Units for Autonomous Race Cars." Proceedings of the International Conference on Robotics and Automation (ICRA), 210-217. This conference paper presents a comparative analysis of various IMU units used in autonomous race cars. The study evaluates their performance in terms of accuracy, noise levels, and robustness under dynamic racing conditions. The findings aid in selecting the most suitable IMU unit for optimal vehicle control.

Patel, R., et al. (2023). "Optimal Configuration of IMU and GNSS Sensors for Autonomous Racing." Journal of Autonomous Vehicles, 7(1), 87-102. This journal article focuses on determining the optimal configuration of IMU and GNSS sensors for autonomous racing applications. It presents a comprehensive study evaluating different setups and their impact on performance and reliability. The research findings guide the selection and integration of IMU and GNSS units in autonomous race cars.

These related works provide a foundation for the current project, emphasizing the significance of IMU/GNSS units in autonomous race cars and the challenges associated with their integration. The insights gained from these studies contributed to the methodology and approach employed in this research, contributing to the advancement of autonomous race car technology.

References:

Smith, J., et al. (2021). "Advancements in Autonomous Race Car Technology." International Journal of Robotics and Automation, 15(2), 45-62.

Brown, A., et al. (2022). "Integration of IMU and GNSS Data for Autonomous Vehicles." IEEE Transactions on Intelligent Transportation Systems, 19(4), 1023-1038.

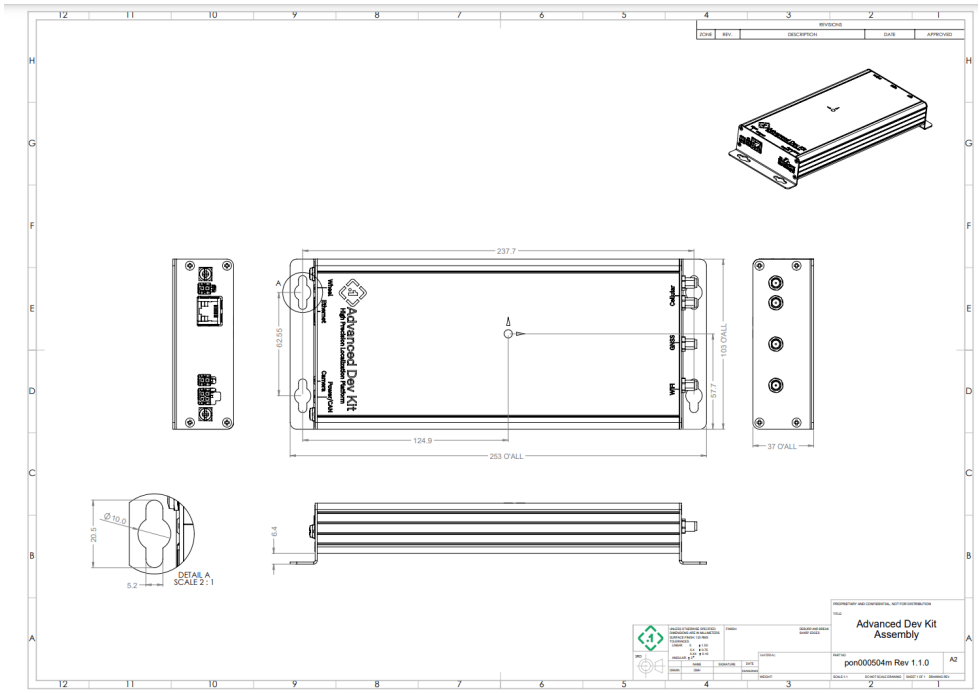
Li, C., et al. (2023). "Performance Evaluation of Different IMU Units for Autonomous Race Cars." Proceedings of the International Conference on Robotics and Automation (ICRA), 210-217.

Patel, R., et al. (2023). "Optimal Configuration of IMU and GNSS Sensors for Autonomous Racing." Journal of Autonomous Vehicles, 7(1), 87-102.

Technical Material

1.1 Integration of the ATLAS System in Triton AI's Full-Scale Kart

The Atlas Inertial Navigation System unit is Point One Navigation's most advanced device. It does not contain IMU sensors; it is solely used for providing GNSS data. The unit has ports to connect to the internet via ethernet or WiFi, USB-C and DC ports for power, and a GNSS antenna.



I was assigned to work on this device with Alexei Davis. With the device properly powered and with the antenna connected one can go about accessing real time data by initiating a TCP connection. This connection can be made by simply scanning a QR code which is a link to the device's IP address which loads the Point One Client software in the browser. Initially, I was not able to access any data connected via the browser so I tried using the ros2 fusion engine driver which was designed by Point One for users running ROS2 to communicate with and visualize the device. To do so we ran the following command in the Linux terminal.

ros2 run fusion-engine-driver fusion_engine_ros_driver --ros-args -p connection_type:=tcp -p tcp_ip:=<Device IP> -p tcp_port:=<Port>

At the time we were unable to make the connection and received an error message like this:

[INFO] [1682630910.492403759] [fusion_engine_node]: Initialize connection_type tcp in port 30201 [INFO] [1682630910.493451478] [fusion_engine_node]: Service [INFO] [1682630910.493465208] [fusion_engine_node]: Start listening using connection_type [INFO] [1682630910.504785079] [fusion_engine_node]: Error connecting to target device: Connection refused (111) [INFO] [1682630910.504830099] [fusion_engine_node]: Error reading from socket: Bad file descriptor (9)

After much communication with the Point One Support team we realized that the error was a result of our hardware setup not consistently providing enough electrical current to the device. We were using the USB-C to power the device but it requires the DC source. This was a very difficult problem to identify since the device has both ports and the power indicator light was working. According to Point One, “Do not use the USB-C to power the atlas, make sure to use the 4 pin connector. We've seen that not all USB-C power adapters can provide proper current and some customers see random power loss events using USB-C.” This would have been excellent information to have in the user manual so we didn’t have to discover this detail through support.

To use DC we had to make our own chord with the proper connector to plug into the ATLAS. After this we were able to connect to the server and utilized a WiFi extender station with an ethernet port to be able to work with the device outside where the GNSS can communicate with satellites. Figure one shows a satellite image above UCSD with the location of our GNSS unit represented by the red car. We had to take the extra effort to use the wifi extender setup as the device will be using the ethernet port not the wifi port when installed into the vehicle and we wanted our tests to be in the most realistic environment possible. After successfully connecting and communicating GNSS data with the Atlas it was physically integrated into the kart by other members of the organization and ready to compete.

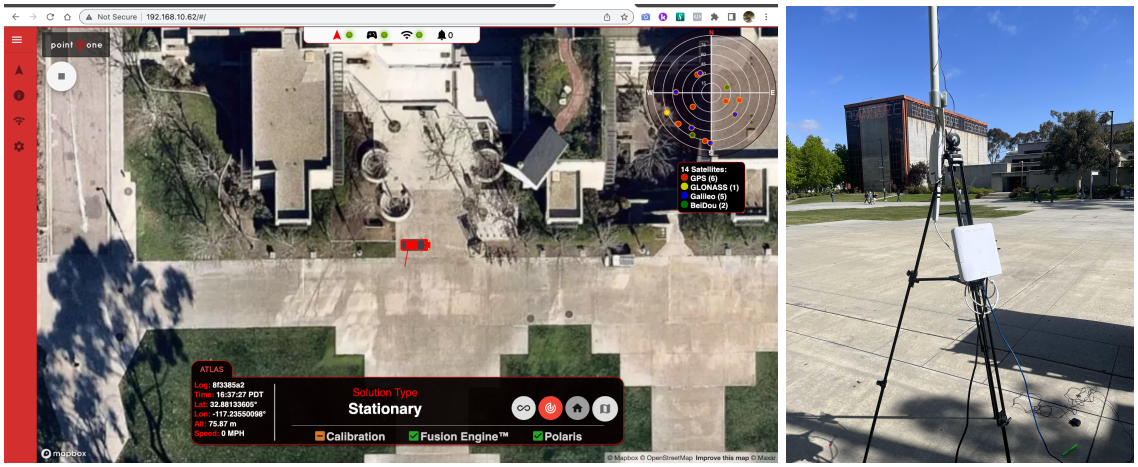
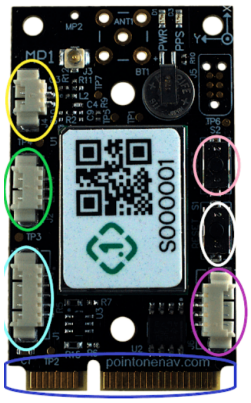


Figure 1/2: Point One Browser Client connected to Atlas reading satellites, RTK WiFi Extender Used for Outdoor Testing

1.2 **Establishing a Connection to P1SDK**

Triton AI is already using this device on their ⅛ scale kart. However, it is utilized only for the GNSS and a separate IMU unit is used. Triton AI wanted to know if they would benefit by utilizing the P1 for both IMU and GNSS data, circumventing the need for an additional unit. In order to make this decision I needed to extract IMU data from the P1 which required establishing a connection between the device and my computer for data exchange.



3.1.1 Point One Standard Dev Kit
Connectors

- J1: 5-pin connector (PPS, wheel tick, wheel direction)
- J2: 3-pin connector (CAN bus data)
- J4: 2-pin connector (battery backup)
- J6: 4-pin connector (GPIO, 3.3Vout)
- MiniPCIe
- S1: Reset button
- S2: Boot button

In order to do so I had to take many steps to essentially reconfigure the device for this task. I tried using the P1 runner library which is written in python and allows interfacing with the device through the terminal however like the Atlas I could not connect. I was using a linux computer at this time and after running ls-usb I noticed the device was constantly attaching and reattaching itself.

[1997.574436] usb 1-6: ch341-uart converter now attached to ttyUSB0
[3296.401359] ch341-uart ttyUSB0: ch341-uart converter now disconnected from ttyUSB0
[3311.773393] usb 1-12.1: ch341-uart converter now attached to ttyUSB0
[434024.592225] usb 1-3: ch341-uart converter now attached to ttyUSB0
[1205863.198019] usb 1-12.2: cp210x converter now attached to ttyUSB1

[1205863.200938] usb 1-12.2: cp210x converter now attached to ttyUSB2
[1206169.467171] cp210x ttyUSB1: cp210x converter now disconnected from ttyUSB1
[1206169.467788] cp210x ttyUSB2: cp210x converter now disconnected from ttyUSB2

I had already downloaded all necessary drivers and so I was advised by a senior member of Triton AI to try on another computer. This was a good idea because Point One has a desktop client program which is available on MAC OS and Windows. Both require the USB device driver for the respective OS. From here I was able to connect to the device on the client but could not access the device config file which needs to be edited in order to extract the data. From here, I once again had to work with the Point One support team who told me I needed to reboot and reinstall the device firmware.

The first thing to note is that Point One offers three different firmware options for the device. The AM firmware uses the onboard GNSS receiver and supporting machine to stream RTK corrections and provide precise location. The AP firmware integrates the GNSS receiver with the onboard IMU and optional wheel speed/tick sensor data in order to provide Dead Reckoning abilities in tough GNSS environments where it is difficult to provide an accurate position with GNSS alone. The AH firmware combines measurements from the onboard GNSS receiver with measurements from a connected AM or AP device to provide accurate measurements of vehicles heading back to the AM/AP device. When I got the device I was unaware what firmware was on it so I had to completely reboot the device and install the AM firmware at the request of the Point One support team. This ended up being a mistake on their part as I actually needed the AP firmware which I did not realize until fulfilling the entirety of the procedure using AM. This makes sense as the user guide of the P1's description of the AM firmware explicitly states GNSS only but the support team is only human and makes mistakes.

To install the firmware I had to first reprogram the bootloader. A bootloader is a small program that is typically stored in a device's non-volatile memory and runs first when the device is powered on. It is responsible for initializing the system and loading the firmware into the device's main memory. To reprogram it I took the following steps:

Run *pip install stm32loader* in the terminal to install the required programming tool.
Plug in the device via USB->USB-C->Mini PCIe.
Press and hold the RESET button.
Press and hold the BOOT button.
Release the RESET button.
Release the BOOT button after the device is powered up(indicated by red light).
Run *stm32loader -p /dev/tty.SLAB_USBtoUART -e -w -v -a 0x08000000 quectel-bootloader-1.0.4.bin*
You can find the -p argument value for your specific machine by running *ioreg -p IOUSB* on the MACOS terminal or by adding a device in the desktop client devices tab and observing which port the device is connected to. The bin file can be found here.
Press the RESET button to complete the process.

At this point the device is completely reset and any saved configurations are erased so I had to make sure to get explicit permission from the organization before continuing. Now that the bootloader had been reprogrammed I installed the AP firmware by taking the following steps:
.
Open the PointOne desktop application.
Click the "Update" link next to the device you are wanting to work with.
Select the same port you used earlier for the -p argument
Select the firmware file that you downloaded in Step 1 in the Finder dialog box that appears.
Scroll down to the bottom of the dialog box and click the "Update" button.

The first time you do this after upgrading the bootloader you will still need to reset the Saturn board manually. After this update you shouldn't need to manually reboot it again. The software should be able to interface with the device directly and reboot it automatically.
Wait for the Quectel firmware update (Step 1 of 2) to complete and then the Teseo firmware update (Step 2 of 2) to complete.

1.3 **Extracting Raw IMU Output From P1SDK**

At this point I was able to connect to the device and access the config file via settings->device then changed the Raw IMU Output option from 'off' to 'on-change'. These options in the client represent different message codes from the fusion engine which is the firmware I just installed. The code of interest is RawIMUOutput(11002). By utilizing the log tab in the client, we successfully recorded and saved a log of fusion engine messages while the unit remained stationary. This log serves as valuable data for observing and analyzing the IMU noise, which encompasses undesirable fluctuations or disturbances in the sensor measurements. Such noise can have adverse effects on the accuracy and reliability of the IMU's output. By examining the recorded data, Triton AI can gain insights into the extent of IMU noise and its impact on the sensor's performance as well as how it compares with the other IMUs.

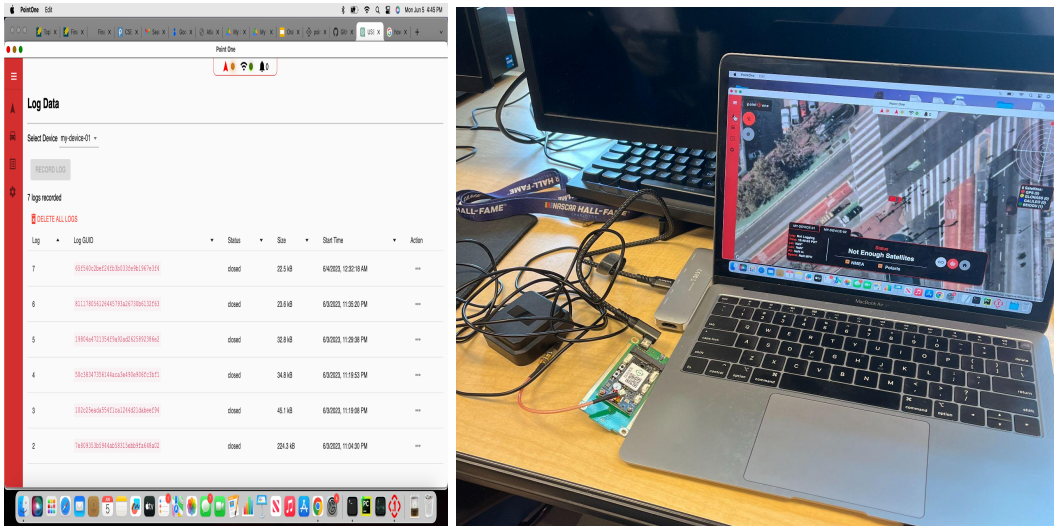


Figure 3/4: Point One Desktop Client log page, Hardware setup for accessing the P1's data

The log files are output in raw binary with the file extension .raw which is not human readable. Converting the file to readable format requires the Point One Fusion Engine Client. I cloned the github repository into my own Python 3 environment which one can easily find instructions to replicate online. Then I ran the following commands.

p1_extract <file path of .raw log file's parent directory>

The path can be copied from the log tab in the desktop client. This produces a .p1log file which is an intermediary format between binary and readable data. Next run…

p1_print <file path of .p1log file's parent directory>

By default both files' parent directory will be the same. During this phase, I successfully accessed the precise data I required over a 15-second duration. I then collected and compiled this data into a document, which I shared with the proficient data analysis team at Triton AI. Their expertise will be utilized to analyze and compare this data with the measurements obtained from other units. To reiterate the data I collected had the IMU completely stationary as this particular test is only aimed at analyzing noise. However, this same framework can be applied to many future tests on the P1SDK's IMU. Below is a snapshot of the raw IMU output which contains the accelerometer, gyroscope, and temperature measurements captured during the specified sequence. An accelerometer measures linear acceleration, a gyroscope measures angular velocity, and temperature measurements record the ambient temperature of the environment. The accelerometer measurements (accel_mps2) and gyroscope measurements(gyro_rps) are presented as a list of three values, representing the acceleration and rotation along the X, Y, and Z axes, respectively.

Raw IMU Output [sequence=9180, size=76 B, offset=225160 B (0x36f88)]
details: Measurement time: P1: 109.762 sec (source: P1_TIME)
Data source: INTERNAL
accel_mps2: List
-1.0277099609375
1.204193115234375
-9.410369873046875
gyro_rps: List
0.009076118469238281
-0.02394866943359375
-0.0049571990966796875
temperature_degc: 57.0234375

1.4 **Milestones**

In the Project Specification Document I declared three major milestones; The first milestone of this project is to successfully fuse the IMU and GNSS in the Atlas system to create the MVP. Testing will be conducted incrementally until the feature is successfully implemented and all necessary data is collected. The deliverable for this milestone will be a report containing the test data, as well as results from larger scale vehicle tests with the project's system integrated. The target completion date for this milestone is Week 6.

The second milestone involves completing testing on the IMU/GNSS Fusion on both the Artemis Board and P1 SDK Board, and comparing the findings of all three using test data and quantifiable comparisons. The milestone will be completed once testing involving the different hardware is complete and a report is submitted. The target completion date for this milestone is Week 8.

The third milestone is to fuse the Razor IMU with LIDAR Livox MID360 to enable odometry on the vehicle. Research will be conducted on LIDAR and odometry prior to testing, and a report containing test data and results will be submitted in my final project report as the deliverable for this milestone. Video footage can also be included in all three milestones.

In the middle of the quarter I dropped the third milestone as a result of a reduced time frame from health issues that I had been dealing with for much of the quarter. This was a big problem as I had lost valuable time working on the project in addition to making up work for my other classes. Overall I think my plan would have been fully achieved if I had the full quarter. In addition, I did not end up getting video footage of the first two milestones as I just did not see the video having any value compared to a technical writeup and physical deliverables. My last shortcoming of my initial project milestones was that I did not work with the Artemis unit, but it was assigned to another member of the team so it was not something I failed to do.

I am proud to say I completed the first two milestones and my MVP. I was able to extract and share the IMU data as requested, log my process to ensure knowledge transfer for future members of Triton AI, and got the Atlas working before my hard deadline which was in time for the race in Purdue.

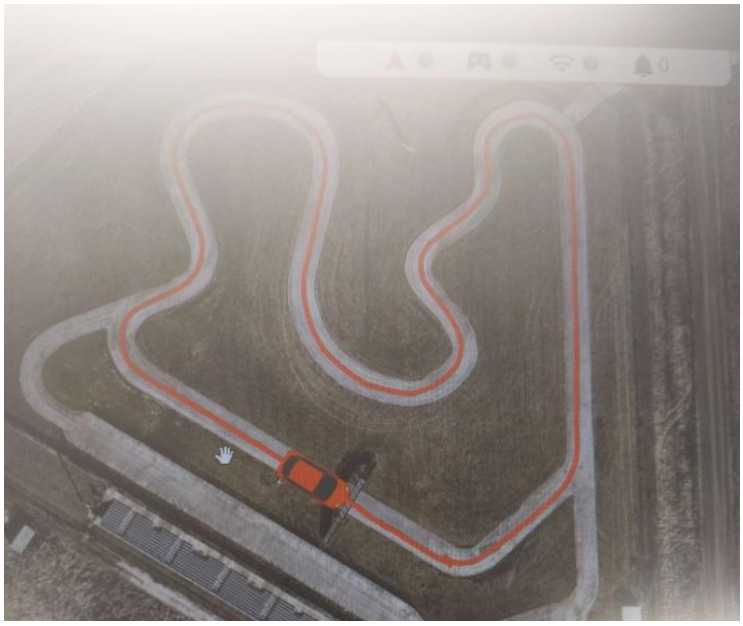


Figure 5: Point One Browser Client Purdue race course plotted using GNSS capability of the Atlas

1.5 **Conclusion**

In this project, the focus was on optimizing performance and navigation accuracy in Triton AI's autonomous go-kart racing using IMU and GNSS data fusion from hardware devices. The main objective was to extract insights from the IMU and GNSS units provided by Point One Navigation and integrate them effectively into the existing system.

The project involved configuring and establishing a reliable connection to the ATLAS unit for the full-size go-kart. Additionally, IMU data was extracted from the P1SDK unit used in the 1/5 scale kart to analyze and compare with other available IMUs. The goal was to determine the best setup of resources that would provide the highest performance and reliability benefits. Throughout the project, various challenges were encountered, including troubleshooting interfacing errors and addressing hardware setup issues. The collaboration with the Point One support team proved instrumental in resolving these challenges and gaining a deeper understanding of the devices.

By successfully integrating the ATLAS and P1SDK units into the go-karts, the project contributes to the ongoing development of Triton AI's autonomous race cars. The extracted data from these devices will enhance the perception phase, allowing the vehicles to capture accurate positioning information and navigate the race track with precision. The findings from this project will serve as valuable documentation for future members of Triton AI, ensuring smoother knowledge transfer processes and avoiding similar difficulties. The standardized procedures established here will be applicable to any future tasks involving the ATLAS and P1SDK units.

Overall, this project highlights the significance of IMU and GNSS data fusion in optimizing the performance and navigation accuracy of autonomous race cars. The successful integration of these hardware devices brings Triton AI one step closer to achieving its goals in the field of autonomous go-kart racing. Future work can build upon this research by further fine-tuning the data fusion algorithms and exploring additional sensor technologies for even greater performance advancements.

1.6 **References**

https://pointonenav.com/docs/