Michigan State University

Team Lockheed Martin

LiDAR and Stereo Image Fusion for Autonomous Navigation

Project Plan

Fall 2022

Lockheed Martin Sponsors
Josh Davidson
Dr. Bennie Lewis

Michigan State Capstone Team
Carlo Barths
Matt Anikiej
Nathaniel Ferry
Dominic Mazza

Contents

Executive Summary

Serving both the international and domestic communities, Lockheed Martin provides world-class service in advancing scientific discoveries and improving national security. Lockheed Martin is a Fortune 500 company with business areas in aeronautics, missiles and fire control, rotary and mission systems, and space. They provide solutions to multiple countries and United States (U.S.) agencies including Japan, Germany, the Department of Defense, all five U.S military branches, and the National Aeronautics and Space Administration (NASA).

Lockheed Martin Space is a division of Lockheed Martin that aims to solve the world's challenges by delivering vital technologies to existing and future endeavors in space. Lockheed Martin Space has partnered with General Motors (GM) to explore potential design solutions to the next generation of rovers called the Lunar Mobility Vehicle (LMV). This next generation of rover will enable long-term exploration of the lunar surface, assist astronauts in science, and can optionally be operated autonomously.

Team Lockheed Martin Space is working with Lockheed Martin to develop software that combines data from Light Detection and Ranging (LiDAR) sensors with stereo image sensors on an embedded system. This software will serve as a demonstration that LiDAR pointcloud data can be combined with stereo image data for autonomous vehicle applications.

Additionally, Team Lockheed Martin Space is developing a messaging transport plugin for Lockheed Martin's SmartSat™ framework that allows for embedded systems to utilize MQTT (Message Queuing Telemetry Transport) API (Application Programming Interface) calls in communications with Internet of Things (IOT) devices. This will allow the LMV to communicate with IOT sensors in and around the vehicle and broadcast state of health data across a satellite constellation.

Functional Specifications

When navigating the lunar surface, it is imperative that a vehicular platform can sense and communicate the details of its environment. In contrast to the environmental conditions on earth, vehicles navigating the lunar environment have inconsistent access to light, poor terrain quality, and lack of adequate data to train the neural networks that supply detection information. With advancements in machine learning, there have been numerous innovations in the last decade allowing for use of novel sensor technologies such as LiDAR and software-based techniques such as stereo imaging to detect an environment's objects as well as categorizing the objects around it. Additionally, communication from the LMV to surrounding systems and ground control presents other novel difficulties. As the LMV and its inhabitants will be in a distinct environment relying on satellite communication to transmit data between mission control and the lunar surface, existing communication technologies must be adapted to facilitate transport of data between occupants on the moon as easily as possible.

This project aims to aid Lockheed Martin and GM in their task of developing sensing software for the LMV by utilizing both LiDAR and stereo imaging to detect the lunar environment. As the hardware of the LMV is an embedded system composed of Single-Board Computers, there are hardware constraints that must be adapted to provide real-time detections that are not present in traditional machine learning deployments. To account for this, the team will be developing a fusion technique between two different sensor modalities, stereo-cameras and LiDAR. This fusion technique will occur before any machine learning model, reducing the resource constraints on a system that requires computational and electrical efficiency.

Moreover, the Lockheed Martin Space team will be developing a plugin for Lockheed Martin's existing SmartSat™ framework that allows for message passing between IoT devices on the lunar surface. This plugin will allow SmartSat™ to utilize the MQTT protocol to facilitate this communication with a minimal code footprint and network resource costs.

To facilitate easier integration of both MQTT compatibility and fusion technologies, the team will be building a ROS (Robot Operating System) backend to run the software components on multiple single-board computers emulating the hardware on the LMV. Similar components will be used on the LMV, and development with these components will allow Lockheed Martin Space to assimilate our software with preexisting developments.

Technical Specifications

System Architecture

Camera-LiDAR Fusion and Detection

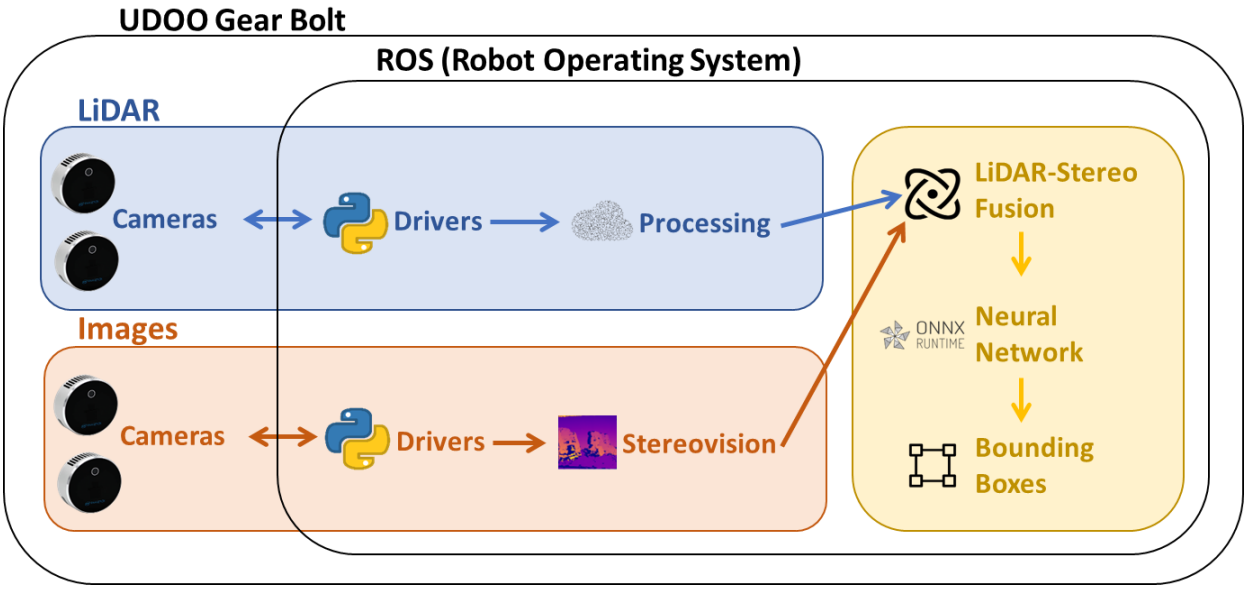

Figure 1: System architecture diagram of LiDAR-Stereo Fusion

The cameras for the LiDAR-Stereo Fusion will be connected to the UDOO Bolt via a network switch. Once connected, both LiDAR and images will be propagated through the system and undergo preprocessing. The LiDAR data will undergo integrity checks and validation while the images are fused into a stereo-disparity map. The resulting stereo-disparity map and validated point clouds will be fused into a singular pointcloud using our fusion algorithm. The fused pointcloud will be fed into the neural network for 3D object detection. The neural network will output three-dimensional bounding boxes around pertinent objects in the environment.

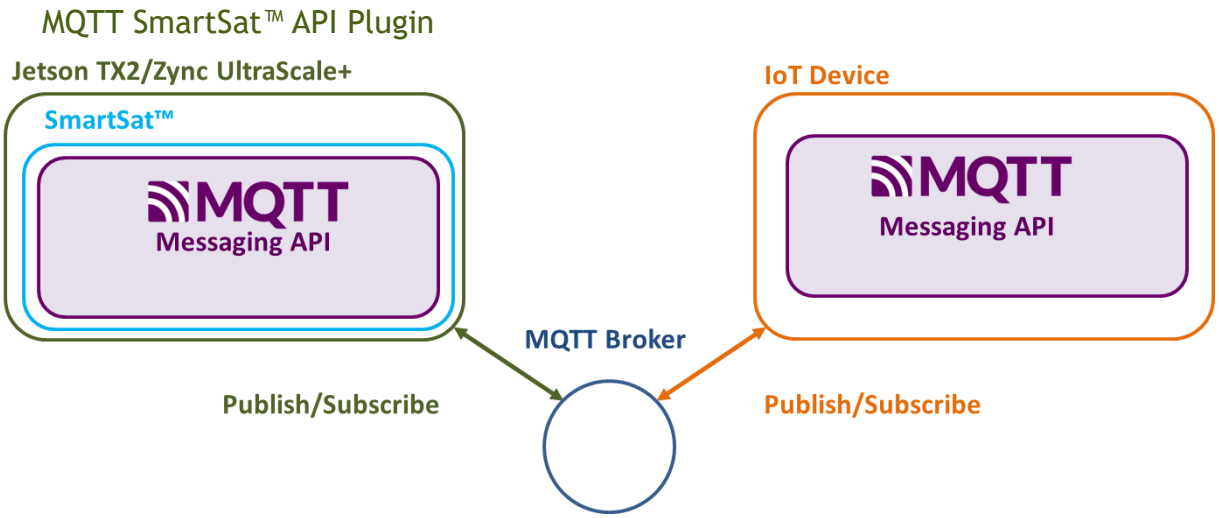

Figure 2: System architecture diagram for MQTT SmartSat™ Plugin

The SmartSat™ MQTT plugin utilizes the SmartSat™ SDK and an MQTT API. The MQTT API plugin acts as a component that can enable MQTT support for SmartSat™ applications. An MQTT client can be any IoT device with an MQTT library installed on it. MQTT clients can publish or subscribe to messages on a specific topic to a broker. The broker is responsible for receiving messages sent from clients and directing them to the correct client for message receipt.

Hardware Components

UDOO Bolt Gear Developer Kit

Figure 3: UDOO Bolt Gear

The UDOO Bolt Gear Developer Kit is a Single-Board Computer (SBC) meant to accelerate AI (Artificial Intelligence) applications on embedded systems. The system includes a Amd Ryzen Embedded V1000 CPU, an AMD Radeon Vega 8 GPU, and 16GB of RAM. This system will serve as the deployment platform for our LiDAR and stereo-image fusion algorithm and will facilitate its connection to our LiDAR and camera sensors.

Xilinx Zync Ultrascale+

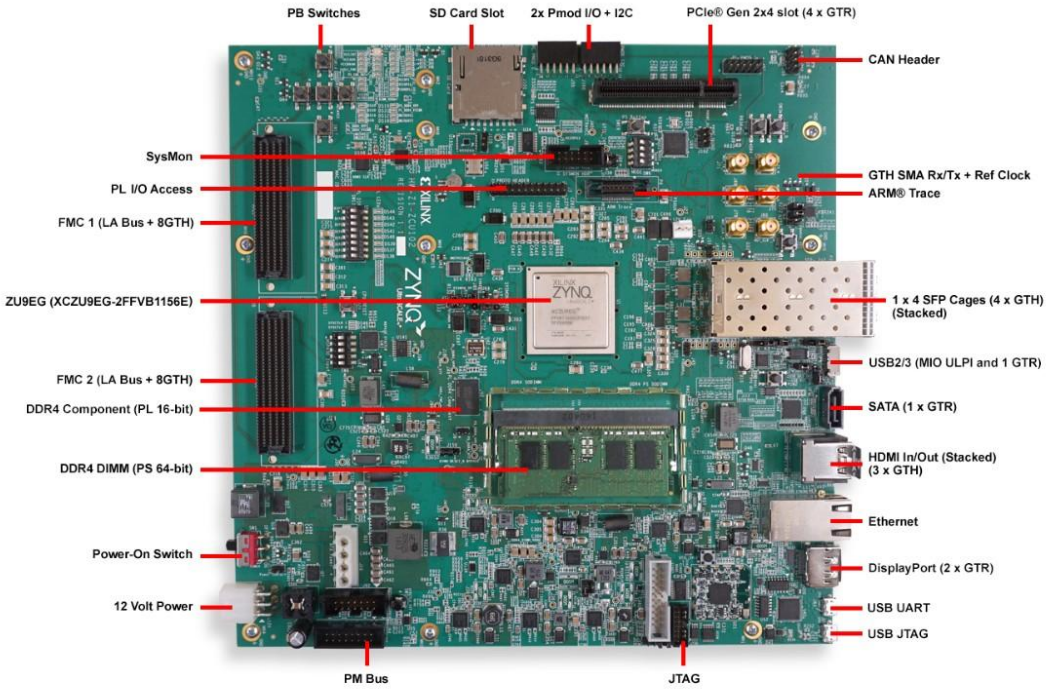

Figure 4: Zync Ultrascale+ module

The Zync Ultrascale+ features a 1.5GHz quad-core Arm Cortex-A53 application processing unit (APU), dual-core Cortex-R5F real-time processing unit (RPU), and a Mali-400 MP2 graphics processing unit (GPU). The combination of central processing units (CPUs) and field-programmable gate arrays (FPGAs) enabled the benefits of the CPU's data driven decision making and the FPGA's ability to: control processing flow, interact with high I/O, and its ability to perform massively parallel algorithms at high speeds.

Intel RealSense LiDAR and Stereo Camera

Figure 5: Intel RealSense LiDAR and Stereo Camera device

The Intel RealSense LiDAR camera is a LiDAR sensor designed to output pointcloud data with power efficiency and speed in mind. There will be two cameras in our system supplying pointcloud data to the UDOO Bolt Gear for processing, fusion, and onboard object detection. The RealSense also comes equipped with two stereo cameras providing a disparity map simultaneously with the LiDAR data.

Software Components

ROS (Robot Operating System)

The Robot Operating System (ROS) is an open-source robotics framework that allows for modular robotics system development. The version of ROS that the team is utilizing is ROS Melodic, which is the ROS version compatible with Ubuntu 18.04. ROS is built with a publisher-subscriber framework, allowing for easy message passing and fast integration of system components. Additionally, ROS is developed in C++ and Python, which will be used interchangeably by the team depending on memory-constraints and library availability. The team will use ROS to house all our software components in the final deliverable and serve as the runtime environment for our real-time sensor deployment.

KITTI Dataset

To validate our system on a large variety of data, an external dataset must be used. In the field of earth-based autonomous vehicles, the Kitti dataset is used to test and validate novel neural network architectures for object detection using stereo cameras and Lidar, as well as benchmark networks against other methods. The team will utilize the Kitti dataset for training and testing in parallel with development of the ROS system, allowing for more thorough development of the neural networks in viability.

Stereovision

Stereovision is the technique of processing two images of the same object and extracting 3D data from the images. This data is generated through two cameras that have an offset on one axis and are oriented towards the same object. This offset allows for a software system to calculate the disparity between the two images, which allows for a depth mapping of the environment to be created. The depth mapping can then be converted into the pointcloud format which allows for fusion with LiDAR pointcloud data.

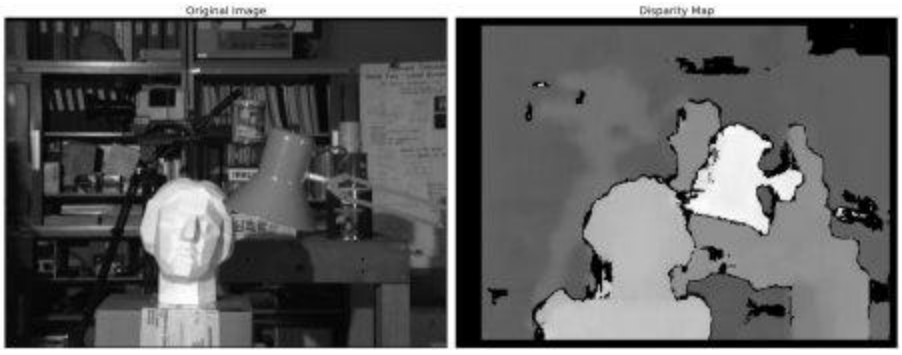

Figure 7: Example image of stereovision

Camera/LiDAR Fusion

Fusing the data obtained from the stereo and LiDAR cameras first involves generating a point cloud from stereo data. ROS provides this functionality, producing a point cloud from a rectified color image and disparity image. When data from both cameras are properly formatted, the two data sets will be concatenated to form a singular point cloud.

Fusing the data this way will facilitate easy integration of the data with an established neural network architecture, resulting in faster development of a network that can provide accurate detections. Additionally, this data fusion technique is agnostic to the sensor configuration present in the system, allowing for use of this fusion algorithm with prerecorded data in training the underlying neural networks.

ONNX Runtime

The ONNX runtime is an inference engine designed with performance and modularity in mind. The runtime utilizes a universal format for saving and loading models, the ONNX format, and exporting to this universal format is possible in every industry standard machine learning/deep learning library. This runtime allows for the deployment of neural networks in the system to be more efficient due to its library-side optimizations, and it will allow the team to hot-swap models trained in different libraries for rapid-testing.

3D YOLO

One neural network that has shown tremendous success at LiDAR Processing is 3D YOLO (You Only Look Once). 3D YOLO works by being able to break down the point cloud into voxels, and then extracting the information it can from each point within a voxel. It first passed the point cloud through a feature learning network to extract all relevant features out of a point within a voxel and transforms it to a new feature space. YOLO then takes this new feature space and returns the bounding box coordinates and the class scores for each bounding box.

3D YOLO has slightly lower accuracy scores compared to other popular methods such as VoxelNet or MV3D, but the model is more capable of real-time performance than both methods. The runtime speed of the network is especially important as it will ensure the rover will be able to detect changes in the environment as they occur.

PointNet

PointNet is an industry standard in classifying point clouds. It takes an unordered point set and returns either a label or a new segmented point set. PointNet has the advantage of being efficient and effective. The network allows for a variable number of points which makes it easy for the team to get working with different image resolutions.

PointDAN

Domain adaptation consists of training a machine learning model for one source domain and then using that model to learn from a different source. PointDAN implements that concept, but specifically for a point cloud model. PointDAN uses a PointNet++ encoder but is unique in that it establishes Self-Adaptive (SA) nodes to learn from a domain. The team will be running the LiDAR data through the PointDAN model to allow the system to be transferred from a typical environment on earth to a completely new lunar environment.

SmartSat™ SDK

The SmartSat™ SDK (Software Development Kit) is a proprietary SDK developed by Lockheed Martin Space. The SDK provides a framework of APIs and abstraction layers that allow modular apps to be developed for any architecture requirement. This framework allows applications and services to be developed for a SmartSat™ enabled device like the lunar rover or a satellite.

MQTT

Message Queuing Telemetry Transport (MQTT) is a messaging protocol designed for the Internet of Things (IoT). MQTT is designed to connect devices with minimal code and network bandwidth together using a publish/subscribe pattern. MQTT is lightweight, power-saving, and guarantees message delivery. These benefits make MQTT ideal for harsh and low power environments such as a lunar rover.

VxWorks

VxWorks is a real-time operating system (RTOS) developed by Wind River. RTOSes are operating systems that can switch between tasks rapidly and are able to provide deterministic real time response to events. VxWorks also provides a small memory footprint for memory-constrained systems such as a lunar rover. These benefits make VxWorks common for airplanes, rovers, and other robots.

Risks

Switching over from prerecorded LiDAR data to live sensor data

Difficulty: Easy

Description: Initially the stereo images and LiDAR data will come from the Kitti database so that the team has time to receive the LiDAR sensor and set up the ROS wrapper. The models will be tested on the Kitti database first and then later trained on the sensor data.

Mitigation: ROS provides tools allowing sensor data to be easily read and integrated. The data format of the sensor data will be the same as the Kitti data, allowing for easy integration.

Development of a new system for fusion of LiDAR and stereo data

Difficulty: Medium

Description: The output from stereovision techniques and LiDAR sensors conveys similar information, but in different formats, a stereo disparity map and pointcloud data. As the team is developing an early-stage fusion architecture, these two data types must be fused prior to network input. This is theoretically possible, but not widely documented.

Mitigation: The team will be converting the stereo disparity map to a pointcloud and combining the two point clouds as both the conversion from disparity maps and fusion of multiple point clouds is more defined.

Training neural networks is very computationally expensive and time consuming

Difficulty: *Solved*

Description: Training models are time consuming on a CPU, but much easier on a GPU due to their SIMD (Single Instruction Multiple Data) architecture. However, the resources provided to us in the capstone labs do not have any GPUs.

Mitigation: DECS server is set up with 10 RTX 8000 GPUs, 96 worker threads and 768 GB RAM.

Creation of a neural network to process fused point clouds

Difficulty: Hard

Description: The current popular neural networks for 3D image processing mostly use only either LiDAR or stereo data. However, we need to adapt these models to process the new fused data.

Mitigation: The team will be adapting stereo data to established pointcloud data format. Then the model will be trained on the fused data instead of using a pretrained model to make sure it runs well with the different data format.

Make model architecture size-efficient and high performance

Difficulty: Hard

Description: The model architecture needs to be small enough that the rover can process data fast enough to react to it, rather than when it passes whatever it sees.

Mitigation: The team will benchmark and test the pretrained model to determine the theoretical effectiveness of the jetson in terms of real-time inference. If the model is performant, the model will be used as-is in the system. If the model is not performant, the team will shrink its input size, and utilize integer quantization to drastically shrink the model size and increase performance.

SmartSat™ MQTT plugin multiple platform support

Difficulty: Easy

Description: The SmartSat™ MQTT plugin is required to run on multiple different platforms, with multiple different architectures and multiple operating systems.

Mitigation: The team has already scoped out and installed several MQTT APIs to ensure that the plugin will be compatible on multiple operating systems and platforms.

Testing Plan

For testing and quality assurance, the team will be utilizing these techniques:

Google Test (Unit Testing)

Google Test is desired for the testing of the MQTT SmartSat™ messaging plugin application. Integration tests will be developed that benchmark the performance of the MQTT plugin application with all existing plugin applications. Integration tests will be run on all configurations of hardware and software that the MQTT plugin application will be used on. Google Test is preferred for its portability, it works on several niche operating systems that this project uses.

ROS Test (Integration Testing)

ROS test is a library for ROS integration testing. Using .launch files, our system can run individual components to test integration success node-by-node, as well as testing the health of the entire system. ROS test also allows for automation of testing with synthetic and prerecorded data, which will greatly aid the team in checking and rechecking system integrity.

Human in the Loop Verification (Neural Networks)

Human in the loop verification means that the team will be involved during model training and testing. This testing involves evaluating and correcting the model's outputs. By combining human involvement and traditional learning the team will be able to train a more efficient, performant model.

Data Standardization

Since the system will be dealing with different sensor modalities and their integration/combination, the data flowing through it must be checked to verify integrity. This is especially important in our system, as the most important component of computer vision is the quality of your data. Both camera frames and LiDAR frames will be actively sanitized and standardized to prevent error propagation throughout the system.

Testing will occur in parallel on each individual part of the system. Both the LiDAR fusion and the models will first be tested with data from the Kitti database. After LiDAR fusion is successfully implemented the fused data will be tested on the models. At the same time the ROS wrapper will be getting tested on the sensor data. Once everything is working the models will be tested on the data the ROS wrapper is collecting from the sensors after it. The MQTT will also be tested in parallel throughout all of development.

Schedule

Week 1: 09/05 – 09/11

- *Team*: Setup of Development Environments and Resources
- *Team*: ROS Tutorial/Hello World
- *Team*: Initial meeting with sponsor

Week 2: 09/12 – 09/18

- *Team*: Drafting of Project Plan Document
- *Dominic Mazza, Nate Ferry*: Setup of NVIDIA Jetson TX2
- *Carlo Barths, Matt Anikiej*: Setup and testing of ML models

Week 3: 09/19 – 09/25

- *Team*: Initial draft of Project Plan completed
- *Team*: Project Plan presentations
- *Dominic Mazza*: Configuration of camera sensors in ROS
- *Nate Ferry*: Setup of RealSense LiDAR drivers
- *Carlo Barths*: Setup Pointnet and trained it
- *Matt Anikiej:* Testing of NN architectures

Week 4: 09/26 – 10/02

- *Team:* Risk mitigation
- *Team*: Receiving LiDAR sensors
- *Dominic Mazza:* LiDAR driver setup
- *Nate Ferry:* Setup skeleton code for ROS fusion node
- *Carlo Barths:* Setup Pointdan and trained it
- *Matt Anikiej:* Optimization of NN architectures for embedded system

Week 5: 10/03 – 10/09

- *Team:* Preparing Alpha Presentation
- *Dominic Mazza:* Configuration of LiDAR processing
- *Nate Ferry:* Setup of stereo to point cloud processing in ROS
- *Carlo Barths:* Train both models on Kitti and optimize neural networks
- *Matt Anikiej:* Train models with fused data

Week 6: 10/10 – 10/16

- *Team:* Alpha Presentation
- *Dominic Mazza:* Sensor development
- *Nate Ferry:* Start developing subscription and publishing in same ROS node
- *Carlo Barths: Continue to optimize models*
- *Matt Anikiej: Install paho libraries and smartsat sdk on x86*

Week 7: 10/17 – 10/23

- *Team:* Get models to accept fused data format
- *Dominic Mazza:* Migration to Realsense only system
- *Nate Ferry:* Transitioned from mono stereo cameras to RealSense stereo image
- *Carlo Barths:* Convert both models to ONNX
- *Matt Anikiej:* Create a smartsat messaging app capable of sending and receiving messages

Week 8: 10/24 – 10/30

- *Team:* Test out models
- *Dominic Mazza:* Stereo processing development
- *Nate Ferry:* More development on fusion node
- *Carlo Barths:* Both models converted to ONNX
- *Matt Anikiej:* Create MQTT Plugin capable of sending messages

Week 9: 10/31 – 11/06

- *Team:* Beta Presentations
- *Dominic Mazza:* GPU stereo processing and Jetson optimizations
- *Nate Ferry:* Troubleshooting on fusion node development
- *Carlo Barths:* Evaluate converted models
- *Matt Anikiej:* Add receiving capabilities to MQTT plugin

Week 10: 11/07 – 11/13

- *Team:* Beta Presentations
- *Dominic Mazza:* Docker image migration, bolt-gear setup
- *Nate Ferry:* Finished fusion node development
- *Carlo Barths:* Get models to work in ROS
- *Matt Anikiej:* Port app to ZCU102

Week 11: 11/14 – 11/20

- *Team:* Beta Presentations
- *Dominic Mazza:* Fusion optimization and inference integration
- *Nate Ferry:* Research into point cloud alignment
- *Carlo Barths:* Evaluate ROS models
- *Matt Anikiej:* Port app to VxWorks

Week 12: 11/21 – 11/27

- *Team:* Test Models with real data
- *Dominic Mazza:* Documentation and bugfixes
- *Nate Ferry:* Implement point cloud alignment
- *Carlo Barths:* Bugfix models

- *Matt Anikiej:* Create benchmarks tests

Week 13: 11/28 – 12/04

- *Team:* Project videos
- *Dominic Mazza:* Documentation and bugfixes
- *Nate Ferry:* Assist in connecting 3DYolo and PointDAN with ROS
- *Carlo Barths:* Optimize final model
- *Matt Anikiej:* Create benchmark tests

Week 14: 12/05 – 12/11

- *Team:* Deploy models within simulated rover environment
- *Dominic Mazza:* Demonstrate final ROS architecture to customer
- *Nate Ferry:* Research into any possible optimizations
- *Carlo Barths:* Demonstrate final model to the customer
- *Matt Anikiej:* Improve performance of benchmarks

Week 15: 12/12 – 12/18

- *Team:* Demonstrate project to Lockheed
- *Dominic Mazza:* Documentation and preparation for delivery
- *Nate Ferry:* Perform any last necessary optimizations
- *Carlo Barths:* Any last bug fixing before delivery
- *Matt Anikiej:* Bugfixes