Street Drone Technical Realization Document

Figure 1: HAN Street Drone Vehicle.

Course: EVML Project

Written By: Hassan Hotait

Class: M-EVML-VT

Supervisor: Mr. Hugo Arends

Date: 24/01/2022

Version 1.0

Table of Contents

System Architecture

System Description:

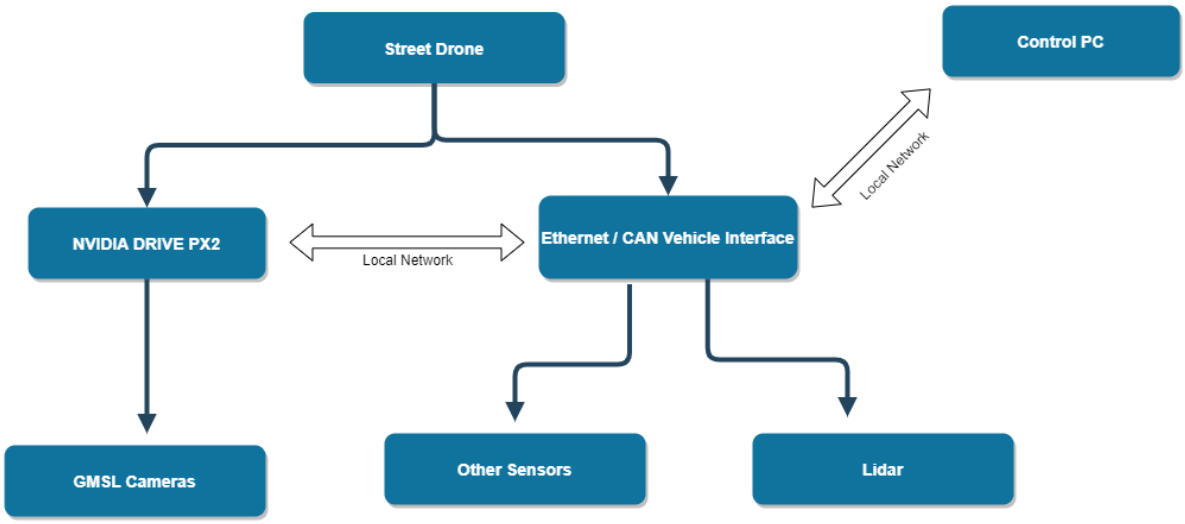

Figure 2: Street Drone System Architecture for Vision Perception.

The Vehicle Interface is responsible for the communication between the Street Drone vehicle and ROS based self-driving software stacks (Autoware.ai) running on the control PC. The interface bridges the gap between ROS kinetic and the OpenCAN vehicle interface of the StreetDrone Xenos Control Unit (XCU) integrated into the SD Twizy R&D and SD ENV200 vehicles.

Figure 3: Nvidia DRIVE PX2.

NVIDIA DRIVE™ PX is the AI car computer that enables automakers, truck makers, tier 1 suppliers, and startups to accelerate production of automated and autonomous vehicles. It scales from a single processor configuration delivering AutoCruise capabilities, to a combination of multiple processors and discrete GPUs designed to drive fully autonomous robotaxis. The Driveworks API designed for NVIDIA DRIVE devices offers a variety of self-driving functionality. However, at the current moment the DRIVE PX2 is going to be used only for Vision related functionality and not to control the vehicle in any way. Autoware.ai is a popular open-source software project that provides a complete set of ROS based self-driving modules, including localization, detection, prediction, planning, and control that are running on the control PC.

Knowing that Lidar and Camera fusion was going to be needed to achieve the desired functionality the Nvidia DRIVE PX2 had to be able to communicate with the control PC. Knowing that the Lidar, Control PC and PX2 are on the same local network (via Ethernet), the ideal choice for communication was via ROS.

ROS is a distributed computing environment. A running ROS system can comprise dozens, even hundreds of nodes, spread across multiple machines. Depending on how the system is configured, any node may need to communicate with any other node, at any time.

As a result, ROS has certain requirements of the network configuration:

- There must be complete, bi-directional connectivity between all pairs of machines, on all ports.
- Each machine must advertise itself by a name that all other machines can resolve.

Nodes can share information across different machines on the same local network by simply publishing or subscribing to a topic. A publisher node sends information on a topic. The subscriber node subscribes to the topic where the publisher sent the information to receive. This was ideal since the Autoware.ai nodes running on the control PC already publish sensor information on different topics.

Nvidia provides all Driveworks functionality as ROS packages to be compiled to run on the DRIVE Platform. However, the team was not able to successfully complete the cross-compilation of the Nodes using a Linux Host System after 2 weeks of trying. Therefore, in agreement with the Streetdrone Supervisor Mr. Klifman socket communication had to be implemented within the DriveNet Sample application (C++ Sample Application Instead of ROS Based Node). Therefore, ROS based communication between the Control PC and the PX2 was not possible. Hence, Socket Inter-Process Communication was implemented to send information to the PX2. The socket implementation is based on the Client Server IPC Sample provided in the Driveworks API. However, implemented within the Object Detection application.

Hardware:

Figure 4: Street Drone Vehicle.

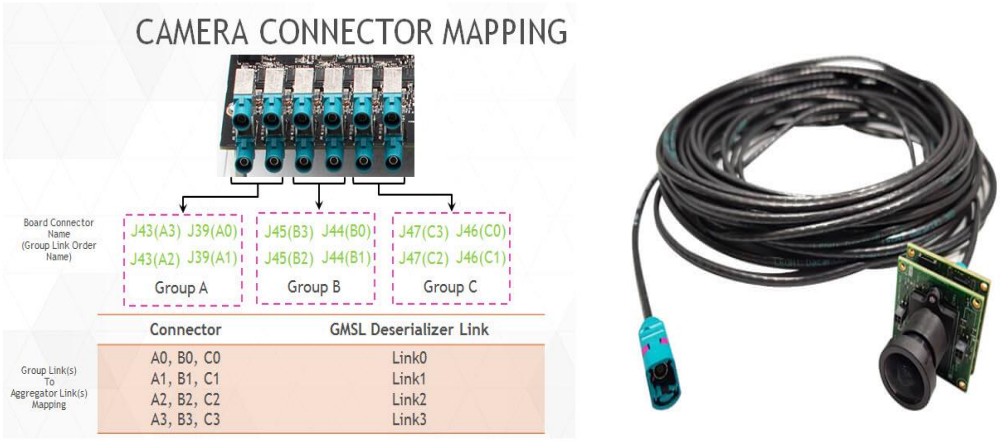

Figure 5: Camera Connector Mapping & Wiring.

Streetdrone Vehicle has 8 cameras in total. Three front ones, two on each side and a rear central camera. You can select cameras individually or max four at a time based on the group and index of each camera.

Object Detection & Classification:

As discussed in research report, Jetson Inference Algorithms were a good out of the box solution since they are provided as ROS Nodes. However, due to the platform incompatibility it was not possible to run them on the PX2. Therefore, DriveNet was the optimal choice since it's created and optimized to run on the PX2 via the Driveworks API.

DriveNet is a camera-based multiclass object detection deep neural network (DNN). It can detect the following object classes, such as:

- Cars and trucks (both labeled as cars)
- Bicycles and motorcycles (both labeled as bicycles)
- Pedestrians
- Road signs
- Traffic lights

Running the sample:

```
./sample_drivenet --input-type=[video|camera]
                  --video=[path/to/video]
                  --camera-type=[camera]
                  --csi-port=[a|b|c]
                  --slave=[0|1]
                  --enableFoveal=[0|1]
                  --dla=[0|1]
                  --dlaEngineNo=[0|1]
                  --precision=[int8|fp16|fp32]
                  --stopFrame=[frame]
```

Figure 6: Running DriveNet from Terminal.

For more information regarding the optional parameters refer to the Driveworks API documentation.

Limitations

Warning

Currently, the DriveNet DNN has limitations that could affect its performance:

- It is optimized for daytime, clear-weather data. As a result, it does not perform well in dark or rainy conditions.
- It is trained primarily on data collected in the United States. As a result, it may have reduced accuracy in other locales, particularly for road sign shapes that do not exist in the U.S.

The DriveNet DNN is trained to support any of the following six camera configurations:

- Front camera location with a 60° field of view
- Rear camera location with a 60° field of view
- Front-left camera location with a 120° field of view
- Front-right camera location with a 120° field of view

- Rear-left camera location with a 120° field of view
- Rear-right camera location with a 120° field of view

The DriveNet DNN works on any of the above camera positions, without additional configuration changes. (NVIDIA Corporation, 2018)

Determine Angle to Object:

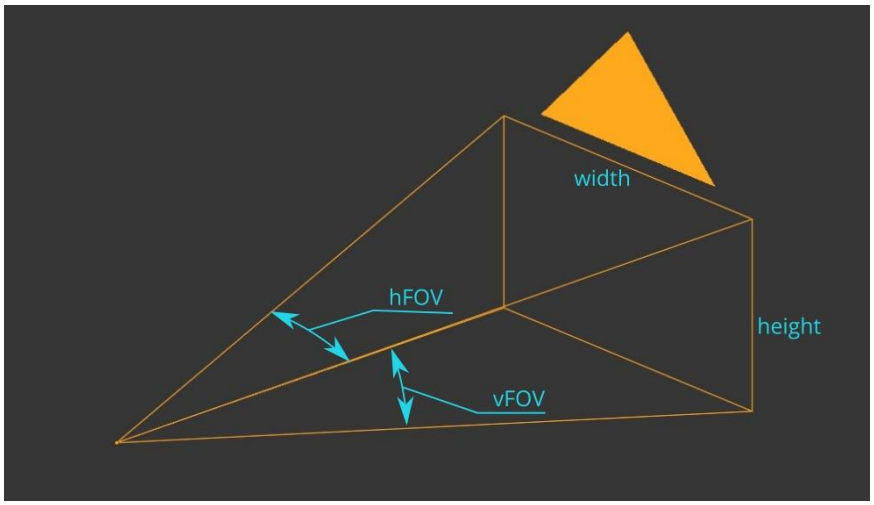

Figure 7: Diagram showing the linear relationship between width and hFOV..

The horizontal FOV is linearly proportional to the width of the image. Therefore, the horizontal angle relative to the center of the camera is defined by:

$$\text{Angle} = \frac{(\text{x-}\frac{\text{Width}}{2})}{\frac{\text{Width}}{2}} * \frac{\text{hFOV}}{2} \qquad \text{when x} > \frac{\text{Width}}{2}\ , \qquad \text{where x: X coordinate of object}$$

$$\text{Angle} = \frac{\text{hFOV}}{2} - \frac{(\text{x-}\frac{\text{Width}}{2})}{\frac{\text{Width}}{2}} * \frac{\text{hFOV}}{2} \qquad \text{when x} < \frac{\text{Width}}{2}\ , \qquad \text{where x: X coordinate of object}$$

Get Distance using Camera Lidar Fusion:

Camera Lidar Fusion has been chosen instead of Stereo Vision since

The gridmap generator outputs a 1D array with high or low value representing occupancy of length width x height. Therefore to calculate the distance, we reshape the array into (height,width) and obtain a grid map representing the parking deck shown below. Free grids are represented as black while occupied ones are grey. (Relative to Lidar Center)

Figure 8: 1D Array reshaped into (height,width).

Since the top view angle from center of camera to object is calculated we can plot a line from the center of the camera at that angle. The 1st grid along that line to be occupied represents the coordinate of the object (get_coordinate function). Knowing the coordinates of the camera in the cost map we can get the distance using Pythagoras Theorem. Knowing the grid resolution, we can convert the distance into meters (get_distance function).

Figure 9: Distance to person standing in front of vehicle.

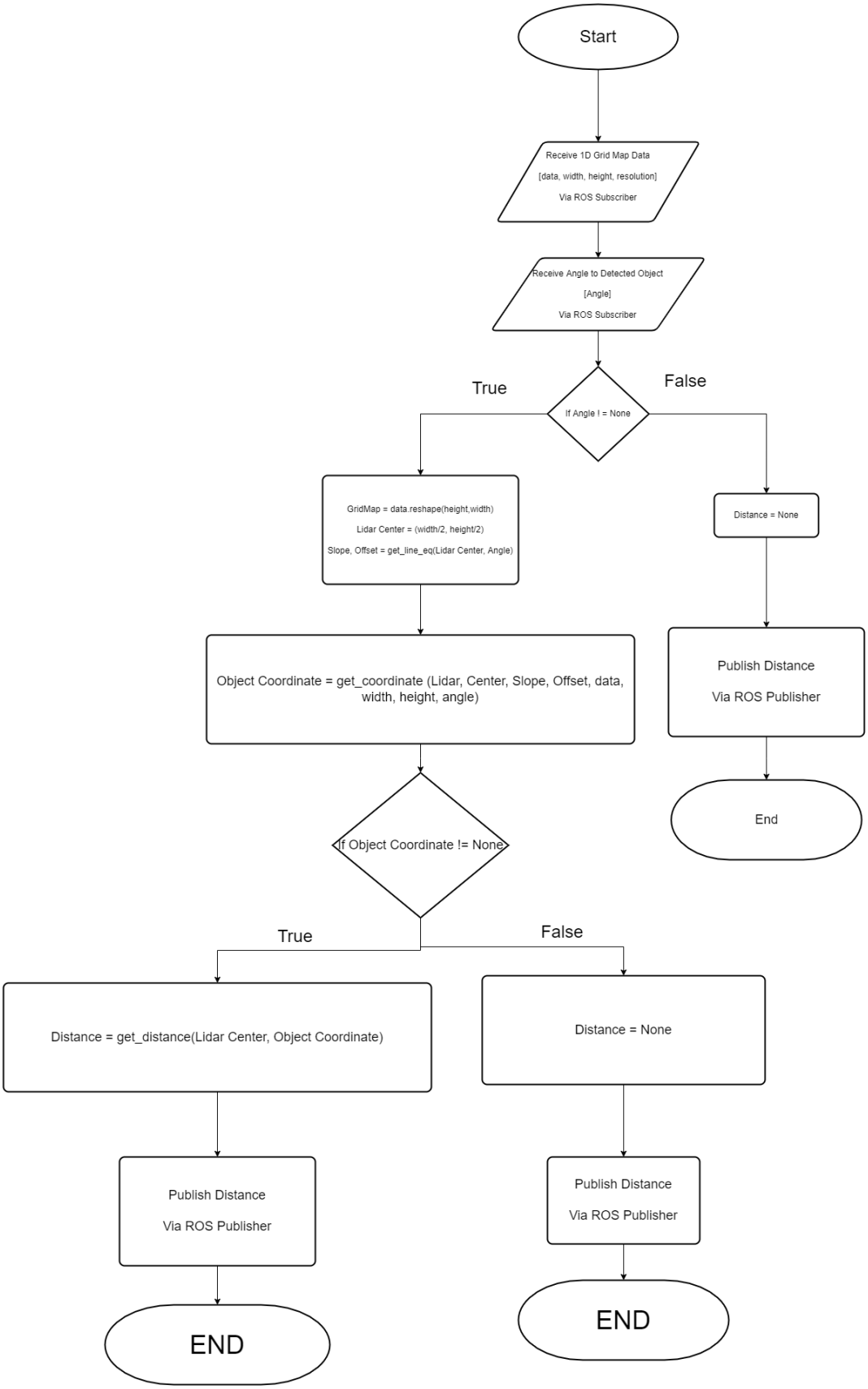

Figure 10: Flowchart of Lidar_Camera_Fusion Node.

Predict Pedestrian Trajectory:

The prediction will be represented in the Lidar Grid Map instead of in the image for several reasons. First of all, the best fit line method prediction on the image investigated in the research report was tested and resulted in decent performance. However, not when the vehicle was in motion. It was not possible to correct the pixel coordinates based on the change in vehicle position. Whereas, in the Grid Map the object coordinates can be converted into World Coordinates knowing the vehicle pose. This allows to get the global direction vector needed for trajectory prediction. In addition to that, the prediction on the image is only a visual representation. It does help the vehicle make better decisions in the real world. Whereas, trajectory prediction in the Grid Map can be integrated with path planning to dynamically avoid objects. This is because the path planner uses the Grid Map to plan the optimal path.

Once the coordinate of the object is obtained in the Grid Map (get_coordinate function) it must be converted into Object World Map Coordinate, the object most recent world position history is stored in a list of length given by the user. This will allow the user to determine how many position samples the prediction is based on. Once we have enough points in the list, we plot the best fit line and determine the direction vector based on it. Then a line is drawn from the object with the obtained direction vector. The advantage of a best-fit line is that it's less susceptible to noise than simple taking the direction vector based on the last two points.

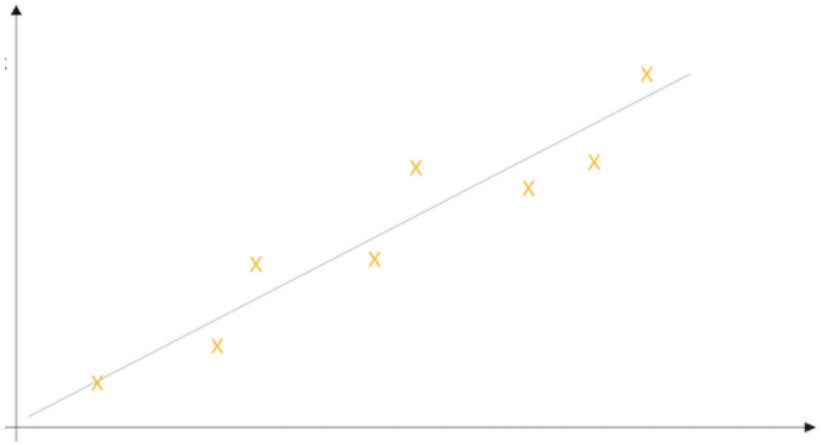

Figure 11: Diagram showing best-fit of scatter points..

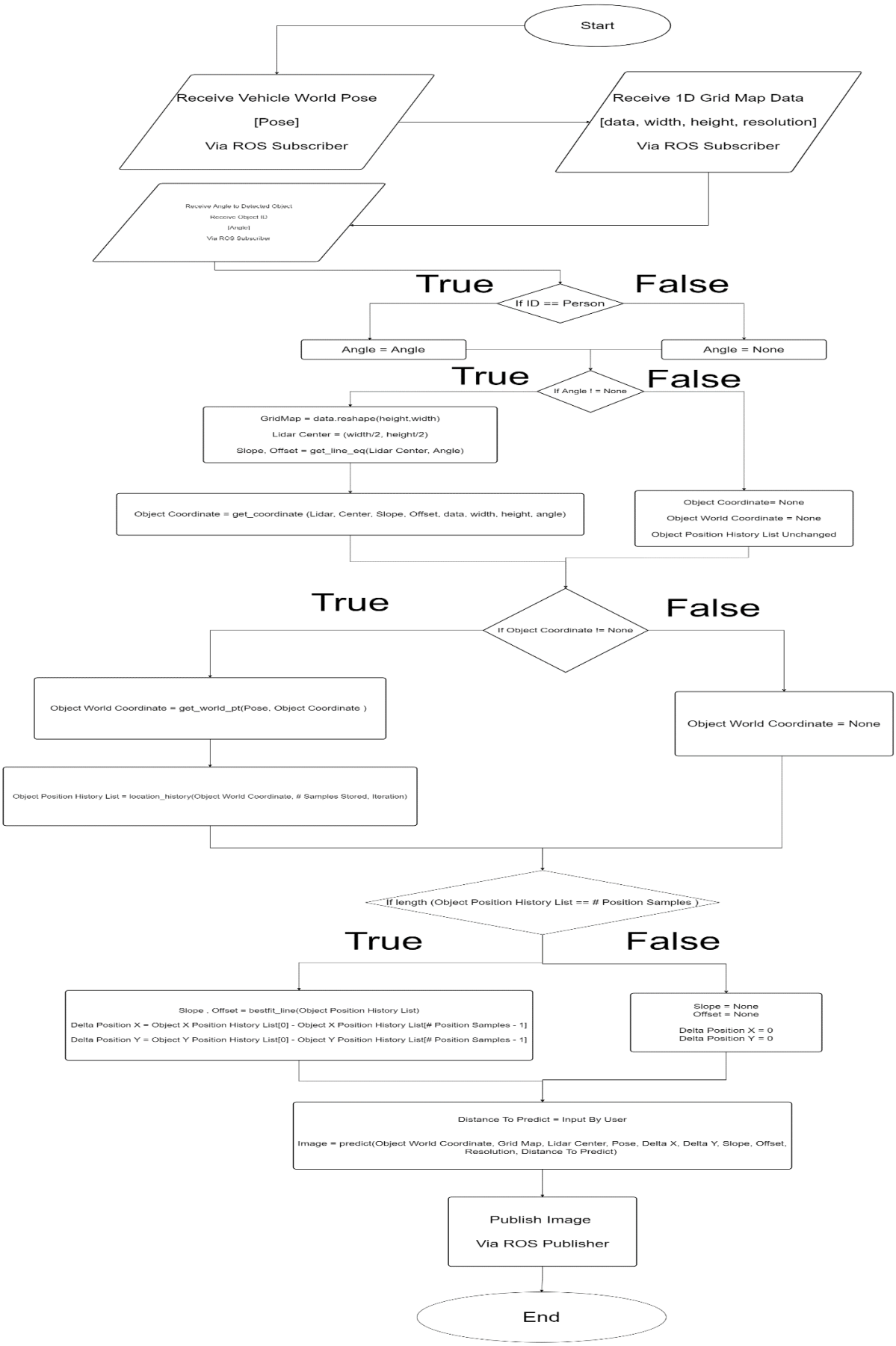

Figure 12: Flowchart of Trajectory Prediction.

Detect Parking Spot:

Since the Jetson Inference Algorithms were incompatible with the PX2, the Semantic Segmentation Neural Net “SegNet” could not be used to detect an empty parking spot as discussed in the research report. Therefore, the parking detection was done using the grid map.

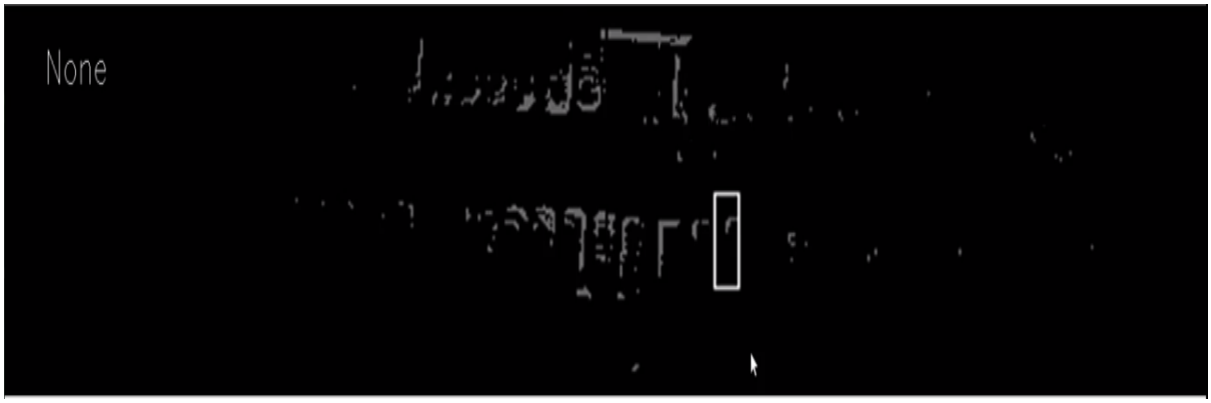

Figure 13: Figure showing occupied parking spot.

A rectangle the representing the space the vehicle will occupy is defined in the cost map perpendicular to the driving direction (left to right). A front right spot is shown above. The % of occupancy in that rectangle is calculated based on the number of occupied grids. If the occupancy is below the threshold than the center of the rectangle is considered a free parking spot.

However, since it’s not possible to find if the found position is in the middle parking markings method was modified to only look predefined spots that the team made sure are in the middle of the parking markings.The closest spot to the vehicle that is free will be the chosen one to park. The spots are defined in a map of the parking deck obtained from a Lidar Scan.

Figure 14: Vehicle parking into one of the defined parking spots.

Test Results

Object Detection & Classification:

Test	Result
Drive Street Drone on HAN Parking deck and observe what objects are detected and their classification.	The algorithm can classify vehicles, pedestrians, bikes, and traffic signs accurately. However, a traffic light isn't available at the HAN to test on. Very few objects are misclassified on the parking deck.
Stand in front of the vehicle and walk away until the person isn't detected anymore to determine the max distance for detection.	Pedestrians are detected reliably at a max distance of 15 m.
Drive through the parking deck and determine the distance to the furthest detected vehicle.	Cars are detected at a distance of 25 meters.

Figure 15: DriveNet output at HAN parking deck.

Get Angle to Object:

To validate that the calculated angle is correct a test plan was followed. Nadir our Colleague had to stand on the marked positions shown below relative to the front camera center. After the vehicle was centered and the measurements were taken, the circles in the figure were marked as tape on the floor. Since at that point in time the Object Detection algorithms were not implemented yet, Nadir was tracked using his red top.

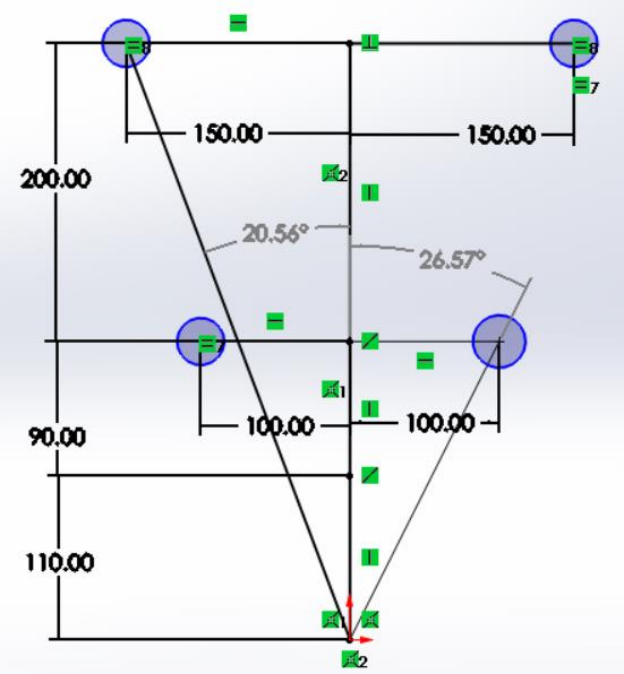

Figure 16: Figure showing marked positions on the floor for test plan.

It is shown below that angle formula used is accurate enough in determining the real angle. At the rear a deviation of 1.4 degrees is observed compared to 0.6 degrees at the front. This deviation is acceptable knowing that the center of the rectangle isn't perfectly in the middle of Nadir. Also, taking in account the vehicle could be not perfectly straight as well.

Figure 17: Figure showing angle to person is -19.2 degrees.

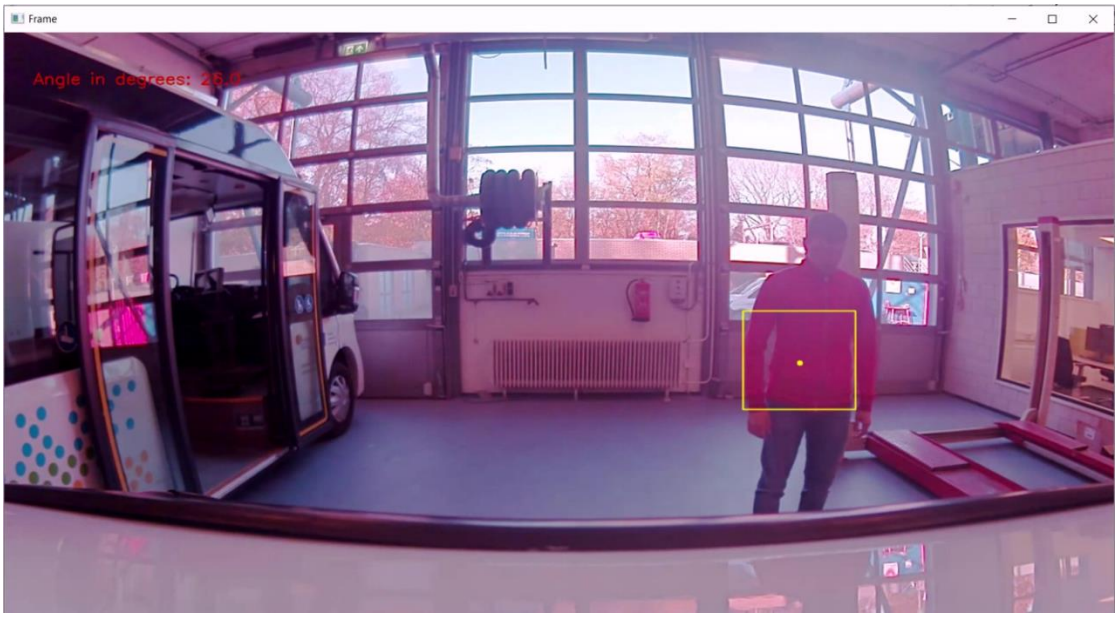

Figure 18: Figure showing angle to person is 26 degrees.

Get distance to object:

The distance has been validated in a similar manner to the angle validation test plan. A person stood at a known distance and the calculated distance was compared with the real distance.

Figure 19: Distance to person standing in front of vehicle.

The calculations are accurate down to +/- 20 cm due to the grid resolution.

The maximum distance that can be detected is dependant on how far DriveNet can detect the object. The dimensions of the Lidar Grid Map are editable and therefore do not impose any limits. (Other than the maximum range which is 200 m x 200 m)

The algorithm has tested also when both the vehicle and the targets and moving and it still performs well. Refer to Video Get Distance.

However, it is observed that sometimes a line is drawn to a random angle where there is a faulty prediction by DriveNet.

Predict Pedestrian Trajectory:

Refer to videos Trajectory Prediction.mp4 and Trajectory Prediction 2.mp4.

Figure 20: Figure showing person moving to right and prediction to the right..

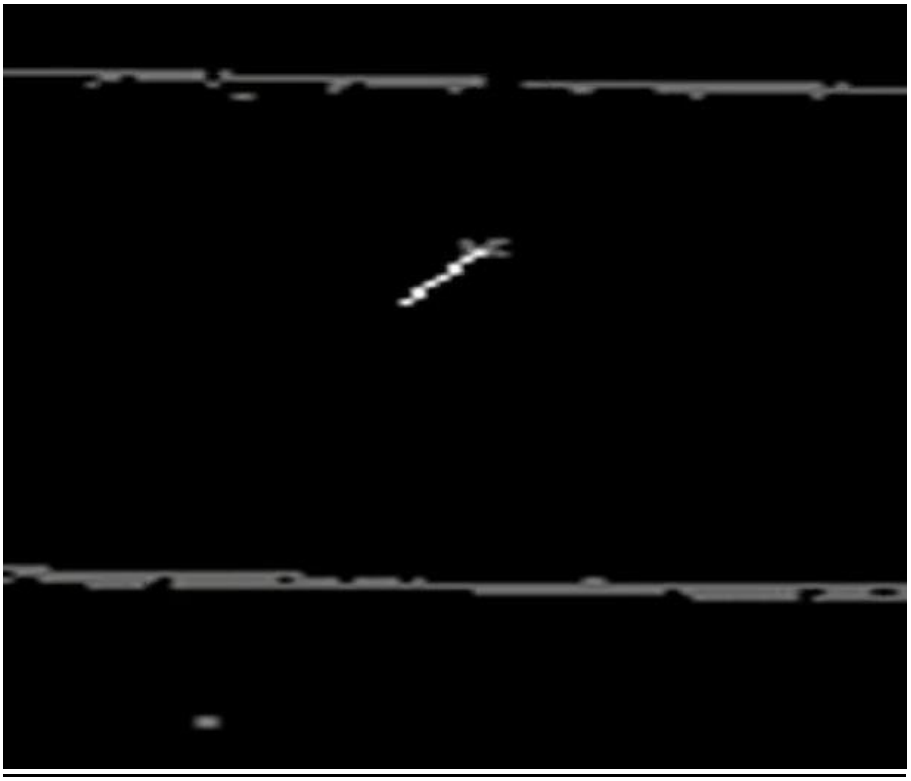

Figure 21: Figure showing predicted trajectory to bottom left.

The prediction algorithm was tested on a person after he was instructed. to move in a zig-zag manner. The predictions are decent and are better visualized in the videos. One issue is that this only works when we have one person in the frame. Since it's currently not possible to know which points correspond to which human. However this is discussed in the recommendations.

Parking Spot Detector:

Figure 22: Figure showing vehicle parking in free parking spot.

By testing this functionality several times while the parking spot is completely occupied to barely occupied the detection was consistent and reliable.

Conclusion

What Computer Vision techniques allow the Street Drone to detect road users and their distance from camera, estimate pedestrian trajectories and detect free parking spaces?

Objects are detected using Deep Learning Perception algorithms and the distance to objects and estimated trajectories are obtained using a Lidar and Camera Fusion workflow. Where objects including vehicles and pedestrians are classified using DL Detectors and then the (Top View) angle to that object is obtained based on the horizontal position of the center of the object and the FOV of the camera. When the angle is calculated it's processed alongside the Lidar Top View 2D Occupancy Grid Map. Knowing the position of the camera in the grid map it's possible to find the 1st occupied grid along the line passing the center of camera and drawn at the angle where object is detected that represents the coordinate of the object. Knowing the resolution of the grid map distance is calculated.

The coordinate of the objects are then sampled as a function of time to estimate the direction vector of the object using a best fit line. However, the coordinates of the object are relative to the vehicle. Therefore, when vehicle is in motion this would lead to inaccurate direction vectors. Thus, the grid map coordinates are transformed to world coordinates knowing the vehicle center position and offset distance to Lidar Center.

Unfortunately, parking spot detection using Computer Vision techniques could not be investigated since the SegNet Neural Network was not compatible with the PX2. Hence, a simple solution was proposed based on the percentage of occupancy of a region where the vehicle is supposed to park.

Recommendations:

System Architecture Recommendations:

- Since the team gave up on cross compiling the Drive work samples into ROS nodes due to urgency and lack of time, future students can investigate this further. This opens the possibilities to new control/perception algorithms other than those provided in Autoware.
- Those ROS Samples are important because they will allow the integration of the PX2/AGX in controlling the vehicle. Currently, all control algorithms are running on a control PC which has significantly lower performance. This would make huge difference on algorithms like path planning.
- PX2 & AGX are underutilized.

Communication Recommendations:

- If DriveNet is successfully cross compiled into a ROS node, then ROS based communication is a huge advantage due to information being shared by anyone listening to the topic unlike sockets that send the information individually to anyone who needs it.
- If DriveNet cross compilation to ROS node is unsuccessful, the data sent by the socket must contain more meaningful information. Currently only the class and object ID are shared. However, object centre image coordinates and number of objects in the frame would open the possibilities of identifying the same object in the next frame and allow Lidar Fusion Node to run at higher rate. (Currently the distance to a single object is obtained from the Cost Map per frame whereas it's possible to get n distances / frame).

Node Source Code Recommendations:

- Get Object Coordinate function (Refer to Lidar Camera Fusion Flow Chart) implementation is too complicated and can be simplified.
- In Lidar Fusion node allow the distance calculation of multiple objects per frame.
- Modify the nodes to take parameters for easier use and customisability.
- Create Launch File to start the complete Vision Perception stack.
- Trajectory Prediction is limited to when only 1 person is detected in the image. It is easily modifiable to multi-person prediction if we can ID identify the same people in the next frames. This is achieved by following the recommendation under Communications.
- Obtain position and pose of 8 cameras relative to the LIDAR to integrate them into Lidar Camera Fusion. Contact StreetDrone for CAD drawings or develop plan for measurement

References

NVIDIA Corporation. (2018). *Driveworks SDK Reference 1.2.400.* From Nvidia.