# RSIP: Rail Safety Improvement With Advanced Situational Awareness At Grade Crossings Using Cooperative Perception And Trusted V2x Communications

*Public Rail-Crossing Deployment (March rd. near Carling/Station rd.)*

## Data collection, analysis and sensor evaluation

May 15 2024

| | |
|---|---|
| Sensor Cortek<br>Prepared by: | Area X.O, Invest Ottawa<br>Prepared by: |
| NAME: | NAME: |
| Sensor Cortek<br>Approved by: | Area X.O, Invest Ottawa<br>Approved by: |
| NAME: | NAME: |

## EXECUTIVE SUMMARY

### Overview

This project aims at defining the processes, exploring the challenges, and building the tools necessary to effectively operate a smart rail-crossing. The learnings from this project can help hone requirements, procure suitable hardware, and advance the techniques used in the data collection and analysis with the objective of creating a safe environment for road users around rail crossings; especially for vulnerable road users.

### Goals

The goals of this project is to leverage unique capabilities, expertise, and collaborators to develop a data collection pipeline that collects data from an rail-crossing fitted with a suite of sensors, create a list of critical-cases that breach rules and regulations in the rail-crossing environment, process collected data to find instances of said critical use cases, and finally, develop a dashboard prototype to allow the exploration of the collected data and interaction with the instances of critical-case events.

### Key Learnings

The project produces a wealth of learnings in all its aspects. In the data collection, the key learning was the mechanism to collect data from heterogeneous sensors along with the temporal properties of the data along with the best approaches to transmit the data to the world. From processing the data to find critical-case events, the key learnings were in setting thresholds and the importance of accurate calibration of the sensors in order to map critical events to the physical rail-crossing. Finally, a key takeaway was the approach to designing the dashboard in order to serve appropriate data concerning the critical-case events to stakeholders.

## ACKNOWLEDGEMENTS

The authors would like to acknowledge the following experts, along with many other individuals, for their invaluable contributions, dialogues, insights and suggestions to the authors made during the project and the preparation of this report:

**Team Area X.O:**

Divyanshu Kamboj
Rebecca Thompson
Susanne Cork
Ethan Ross

**City of Ottawa:**

Omar Choudhry

# Table of Contents

# 1. Introduction

## a. Purpose

The purpose of this project is threefold: the first is to design a reliable and efficient data collection pipeline to collect data from the sensors at the rail-crossing, the second is to establish a preliminary case-based analysis of events captured by the sensors, and the third is to develop a dashboard prototype that will serve as a critical tool in summarizing events captured by the suite of sensors installed at an rail-crossing. The dashboard provides an overview of critical events occurring at the rail-crossing, allowing traffic engineers to monitor and manage traffic flow, identify potential issues, and make data-driven decisions. A well-designed dashboard can help traffic engineers to improve traffic flow, reduce the likelihood of accidents, and communicate effectively with other stakeholders. By leveraging the power of real-time data and visualization, a dashboard can help to optimize the performance of an rail-crossing and ensure the safety and efficiency of the transportation system.

## b. Background

This project was led and coordinated by Invest Ottawa which included communications with all project partners and stakeholders. Invest Ottawa's partners included the National Research Council and a range of high-tech companies and specialized testing service providers with expertise in vehicle automation and connectivity.

Transport Canada's Rail Safety Improvement Program (RSIP) supported this testing campaign in Ottawa, Ontario. Transport Canada's RSIP mandate is to support projects which help create nationally consistent tools that address road safety challenges. Results of testing conducted within this testing campaign will be used by Transport Canada to develop national regulations and non-regulatory tools.

Machine learning focuses on getting machines to learn and act like humans do by using one or more perception sensors, hardware and software that analyzes and interprets sensor information, and a system which sends commands to an automated system.

Deep learning technology is used to predict patterns and perform judgment-based applications through the deployment of artificial intelligence (AI) algorithms to teach robots and machines to do what comes naturally to humans: learning by example.

Machine vision and deep learning-based methods are key enablers of many safety services and applications for Connected and Automated Vehicle (CAV) and Intelligent Transportation Systems (ITS).

The focus of this project is on the safety of vulnerable road users (VRUs) through automated detection and classification, i.e., identifying VRUs at, through and around rail-crossing. The term Vulnerable Road User (VRU) has been defined and interpreted by transportation communities worldwide in different ways. For example, the U.S. Department of Transportation’s National Strategy on Highway Safety has defined VRUs as: “road users who are most at risk for serious injury or fatality when involved in a motor-vehicle-related collision. These include pedestrians of all ages, types, and abilities, particularly older pedestrians and people with disabilities. VRUs also include bicyclists and motorcyclists. Older drivers may also be considered to fit into this same user group." [1] Although the problem of object detection and classification is not new to the CAV/ITS community, prior works have overlooked certain categories of objects such as VRUs. Most publicly available datasets have tended to ignore these pedestrian classes, often with little or no samples at all. Example classes of important VRUs include those using wheelchairs, strollers and other mobility devices such as walkers and canes.

## c. Project Scope

The scope of this project is to capture data from a public rail crossing and analyze it to identify critical use cases that violate the scene conditions. The project will utilize cameras (both color and thermal) and lidar sensors to capture traffic data: the number of cars, their speed and direction of travel, etc. The captured data will then be calibrated, analyzed to identify any patterns or anomalies that may indicate potential safety issues at the rail-crossing.

The next step of the project will be to develop algorithms to process the data and identify critical use cases that violate the conditions at the rail-crossing. These critical use cases include drivers running red lights, failing to yield, or speeding through the rail-crossing. Once these critical use cases have been identified, the project will develop a dashboard to display the findings in a user-friendly format. The dashboard will be accessible to stakeholders such as city officials, transportation departments, and law enforcement agencies, to help them understand the data and take appropriate actions to improve the rail-crossing's safety.

## d. Limitations

The limitations in this project are divided in the following categories:

### i. Sensor Limitations

Sensor limitations in a multi-sensor environment is primarily about the size of the field of view (FoV). Ideally, the FoV of all the sensors should be as large as possible and consistent between sensors. However, this is not true in reality. The variation of the fields of view causes a limited overlap between sensors and limits the benefit of analyzing the data from the perspective of multiple sensors and leveraging the strength of each individual sensor in the face of challenging visibility conditions. For instance, if one sensor has a narrow FoV, it may miss objects that another sensor with a wider FoV might detect, leading to incomplete data and potential blind spots.

Lidar, a key component in this analysis, has its own set of limitations. One significant challenge with lidar is its sensitivity to environmental conditions. Factors such as rain, fog, and dust can severely impact its performance, reducing accuracy and reliability. Additionally, when working with two lidars, as is the case in this project, synchronization and calibration issues can arise. Ensuring that both lidars are perfectly aligned and calibrated to produce accurate and consistent data is a complex task. Misalignment can lead to data discrepancies, making it difficult to merge data streams and create a coherent and accurate 3D representation of the environment.

The size of the data generated by lidar sensors is another critical limitation. Lidar systems produce large volumes of data, which can pose challenges for data storage, transmission, and processing. Proprietary implementations of lidar data processing can further complicate matters. These implementations often lack standardization, making it difficult to integrate data from different lidar systems or to use common processing algorithms. This lack of standardization can lead to inefficiencies and increased costs, as custom solutions are needed for each proprietary system.

Data transportation and processing are also significant hurdles. The sheer volume of data generated by multiple lidars requires robust data handling and processing capabilities. Transmitting this data in real-time, especially in bandwidth-constrained environments, can be challenging. Additionally, the computational power needed to process lidar data is substantial, requiring advanced hardware and optimized algorithms. Balancing the need for real-time processing with the available computational resources is a constant challenge, impacting the overall efficiency and effectiveness of the lidar-based system.

### ii. Hardware limitations

Collecting data reliably from a suite of sensors is an essential step in building a robust dataset for further analysis and for processing by machine learning algorithms. The limited space along with the form factor allowed to be installed at the physical location limits the type of edge computer to be deployed at said rail-crossing. These limitations generally restrict the possible processing power of the edge machine; thresholds have to be set in order to accommodate the capabilities of the machine available for deployment at the location of interest.

Storage capacity is another critical hardware limitation. The vast amounts of data generated by sensors, especially high-resolution lidar systems, require significant storage resources. Edge computers with limited storage capabilities may struggle to handle the volume of data, leading to potential data loss or the need for frequent data offloading, which can interrupt real-time processing. Furthermore, the storage solutions must be robust and reliable, capable of withstanding the environmental conditions at the deployment site, such as temperature fluctuations, humidity, and physical vibrations.

Processing power is equally crucial in the context of edge computing at rail-crossings. The edge computers must be powerful enough to handle the real-time data processing demands of multiple sensors, including complex computations required for machine learning algorithms. Limited processing power can lead to delays in data analysis, reducing the system's overall effectiveness and responsiveness. High-performance processors are often required to ensure that the data is processed swiftly and accurately, but the constraints on space and power consumption at the deployment site can limit the choice of suitable hardware.

Networking capabilities are another significant factor that adds to the hardware limitations of the system. Reliable and high-speed network connections are essential for transmitting data from the edge devices to centralized servers or cloud platforms for further analysis and storage.

However, in remote or harsh environments, achieving stable network connectivity can be challenging. Network latency and bandwidth limitations can hinder real-time data transmission, affecting the timeliness and reliability of the system's responses. Additionally, robust networking hardware is necessary to ensure consistent communication between sensors, edge devices, and remote servers, which can be challenging to deploy and maintain in field conditions.

Other factors contributing to hardware limitations include power supply constraints, environmental protection, and maintenance requirements. Edge computers and sensors must be designed to operate on limited power supplies, often relying on solar panels or batteries in remote locations. Ensuring that these devices have sufficient power to operate continuously is a significant challenge. Moreover, the hardware must be ruggedized to withstand environmental hazards such as dust, moisture, and temperature extremes. Regular maintenance and updates are also essential to keep the system running smoothly, but these can be difficult to perform in remote or hard-to-reach locations, further complicating the deployment and operation of edge computing systems at rail-crossings.

#### iii. Rail-Crossing Limitations

Not all rail-crossings are alike. Therefore, the critical-case events may not apply across all rail-crossings. This limitation causes a subset of the identified critical cases to be applied only to the rail-crossing at hand, and similar types of rail-crossings.

#### iv. Connectivity Limitations

The massive amounts of data collected is the first challenge to the connectivity to the system. The current system of connectivity leverages 4G technology to transmit data to the internet. Thus, the system inherits the limitations of the speeds of 4G, along with that of network loads at peak times. These limitations in turn imposed size and time of transmission limitations on the captured data.

### e. Milestones

Below are the milestones for this project:

- **Sensor and Data Configuration**

    - Milestone: Complete sensor operationalization, initial data collection, field testing, sensor configuration validation, and data storage setup
- **Data Collection and Classification**
    - Milestone: Complete dataset cleaning, building, and annotation for Lidar and Radar
- **Model Design and Training**
    - Milestone: Complete model design and initial training
- **Data Analysis and Dashboard Development**
    - Milestone: Update dashboard for object detection and implement tracking of unsafe behaviors
- **Testing and Final Report**
    - Milestone: Deliver final report by June 30th 2024

## f. Deliverables

Below are the deliverables for this project:

- **Sensor and Data Configuration**
    - Operationalize sensors
    - Initial data collection including field testing
    - Validate sensor configuration
    - Establish data storage protocols
- **Data Collection and Classification**
    - Clean and build datasets
    - Annotate data for Lidar and Radar
- **Model Design and Training**
    - Design models
    - Train models
- **Data Analysis and Dashboard Development**
    - Update the dashboard for object detection
    - Implement tracking of unsafe behaviors at and around at-grade crossings for Lidar
- **Testing and Final Report**

- Deliver draft report
- Complete edits and finalize report by June 30, 2024

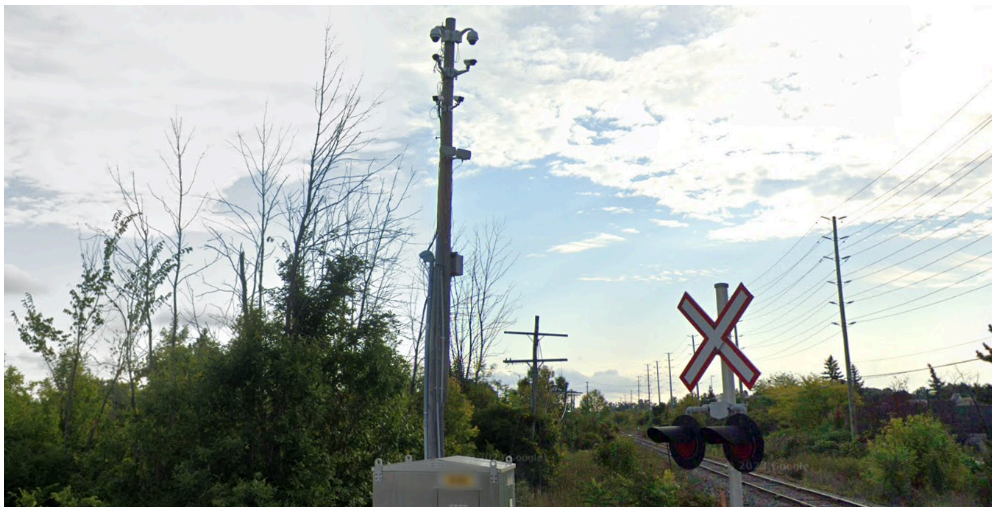
### g. Project Team

Below is a table highlighting the project team:

| Sensor Cortek | AreaX.O / Invest Ottawa |
|---|---|
| Robert Laganiere - Prj. Mgmt. | Divyanshu Kamboj - Prj. Mgmt. |
| Fahed Hassanat - Prj. Mgmt. - Dev. | Rebecca Thompson - Prj. Mgmt. |
| Antoine Huet - Dev. | Ethan Ross- Tech. Dply. |
| Morteza Pasandi - Dev. | |

## 2. Hardware Framework

The hardware framework consists of the sensors, the network devices, the supporting devices, and an edge computer. The images below show the sensors mounted on poles at the rail-crossing. For detailed information about the collection framework, please refer to section 3 part a.



## a. Sensors

Three types of sensors are present at the rail-crossing scene: LiDAR, thermal (infrared-spectrum) camera and RGB (visible-spectrum) camera. The two lidar sensors are Cepton Vista P90 laser scanners installed at a height of 18’, with a range of 200m at 30% reflectivity, fields of view (FOV) of 60° horizontal and 22° vertical, and angular resolutions of 0.25°x0.25° (10 Hz) and 0.27°x0.27° (16 Hz).
The two thermal cameras are Bosch Dinion IP Thermal 8000 cameras installed at a height of 20’, with a resolution of 640x480 at a framerate up to 30 fps, and a thermal sensitivity < 50 mK.
The two RGB pan-tilt-zoom (PTZ) camera are Bosch Autodome IP Starlight 7000i camera at a height of 30’, with a resolution of 1920x1080 at a framerate up to 30 fps and a dynamic range of 120 db to improve the quality of low-light images.

Vista P90 Lidar

Bosch Dinion IP Thermal 8000

Bosch Autodome IP Starlight 7000i

### b. Supporting Hardware

The supporting hardware in the data collection system consists of two MHCorbin devices that capture meta-data from all the sensors at the rail-crossing and report it to a cloud-based database. Two MHCorbin devices are required to handle the load that is produced from two lidars; it was found that using only a single MHCorbin device is not sufficient to handle both lidars present at the rail-crossing.

### c. Connectivity

The system uses unmanaged switches to connect the components internally while a 4G modem provides connectivity to the outside world. The system was provisioned with IP rules that controlled access to the system from outside. These provisions allow the connection to individual sensors to monitor or modify settings, as well as connectivity to the edge computer for the formal data collection process.

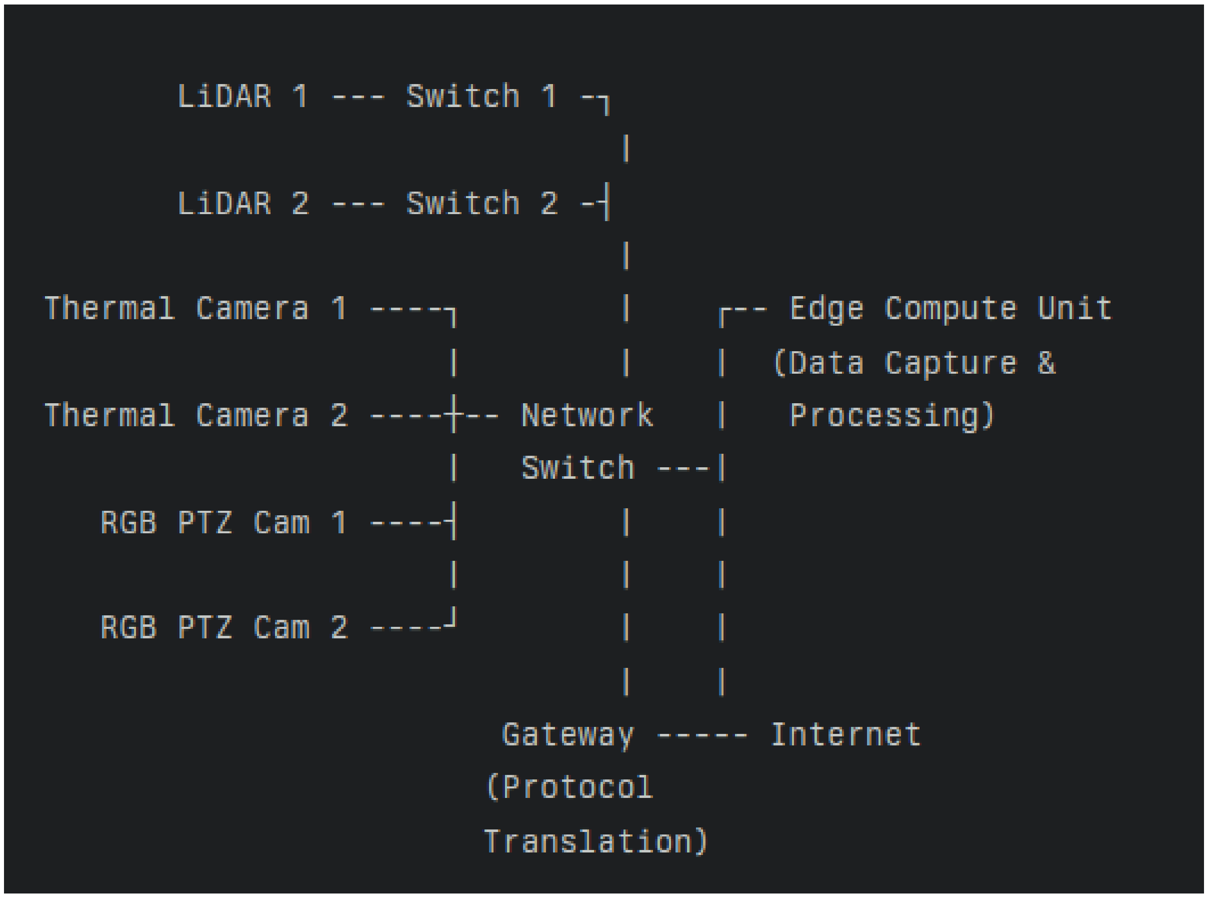
## 3. Data Collection

Data collection is at the core of this project. The data captured from the rail-crossing using thermal, LiDAR, and RGB sensors provides a wealth of information for offline processing to identify critical events and unusual anomalies. The thermal sensors capture the temperature of objects, allowing for the detection of vehicles, pedestrians, and other objects. The LIDAR sensors use laser beams to create a 3D point cloud of the rail-crossing, enabling the detection of the location, speed, and direction of objects. The RGB sensors capture color images of the rail-crossing, providing a visual representation of the scene. By combining data from these sensors, it is possible to create a comprehensive dataset that can be analyzed offline to identify critical events and unusual anomalies, such as accidents, near-misses, and traffic congestion.

The thermal, LiDAR, and RGB sensors are mounted on the same pole near the rail-crossing for data collection. The sensors are synchronized to capture data at the same time, and the data is stored on a compute module that is installed at the poles, then transmitted wirelessly to a remote server for offline processing. Once the data is collected, it can be pre-processed to remove noise and correct any sensor biases or errors. The pre-processed data is then analyzed using machine learning algorithms and computer vision techniques to identify critical events and unusual anomalies. For example, object detection algorithms can be used to detect and track vehicles and pedestrians, while anomaly detection algorithms can be used to identify unusual patterns in the traffic flow or behavior. Overall, collecting data from a rail-crossing using thermal, LIDAR, and RGB sensors can provide valuable insights into traffic patterns and behavior, enabling transportation planners and engineers to make informed decisions about traffic management and safety.

## a. Hardware Framework

The hardware framework consists of the sensor, the network devices, the supporting devices, and an edge computer. The figure below shows the system architecture that allows for collecting data from all available sensors.



```
       LiDAR 1 --- Switch 1 -┐
                             |
       LiDAR 2 --- Switch 2 -┤
                             |
Thermal Camera 1 ----┐       |    ┌-- Edge Compute Unit
                     |       |    |  (Data Capture &
Thermal Camera 2 ----┼-- Network  |   Processing)
                     |   Switch ---|
   RGB PTZ Cam 1 ----┤       |    |
                     |       |    |
   RGB PTZ Cam 2 ----┘       |    |
                             |    |
                        Gateway ----- Internet
                       (Protocol
                       Translation)
```

Hardware Connection Diagram

## b. Software Framework

The software framework used for data collection from the sensors at the rail-crossing was composed of multiple components due to heterogeneity of the sensors at hand. The following is a decomposition of the various components used to capture data from the different sensor:

i. LiDAR data collection: The Cepton lidar connects to the system of sensors via a client server architecture. The lidar takes the role of the server, while the device connected to the lidar initiates the connection and plays the role of the server. The Cepton is made such that it broadcasts the lidar frames to all devices on the same subnet. It was important to take this into consideration, such that other devices on the network do not leak out the information from the lidar. The Robot Operating System (ROS) was the best choice of a sub-framework for collection of lidar frames. ROS enables the design of processes that will listen to the broadcast from the Cepton lidar and capture that information using scripts within the ROS environment. The architecture of the system had to take into account the presence of two separate Cepton lidars, and thus the arrival of messages from two different devices.

ii. Thermal and RGB data collection: To capture data from the cameras, both thermal and PTZ, the system takes advantage of the presence of a streaming server available natively. The stream is initiated by a client script that makes a request to the RTSP server available on the cameras and extracts frames from the stream while pairing these frames with the timestamp of their arrival.
To enhance the data collection process, a separate procedure is initiated to handle the metadata associated with each frame. This procedure employs FFMPEG, a powerful multimedia framework, to establish a session that captures not only the video stream but also the accompanying metadata. The metadata, which includes information about the detections made by the on-board AI, is crucial for understanding the context of the captured frames. By initiating an FFMPEG session, the

system can continuously record both the visual data and the AI-generated metadata in real-time, ensuring that each frame is accurately annotated with relevant detection information.

The use of FFMPEG also allows for advanced processing capabilities such as encoding, decoding, and streaming, making it a versatile tool for handling complex data streams. This approach ensures that the data collected is comprehensive and rich in detail, facilitating better analysis and interpretation. By capturing both the video and the metadata, the system can leverage the AI's capabilities to identify and track objects, providing valuable insights into the behavior and movements within the monitored area. This dual-stream approach not only enhances the accuracy of the data but also enables more sophisticated post-processing and analysis, leading to more effective monitoring and decision-making.

## c. Methodologies

The process of the data collection culminated in using scripts to put together the different components of the collection system in order to initiate a recording session. Bash scripts were written to call the different scripts and to start them as different processes on the edge system. The main script contained a customizable variable that controlled the length of the capture process and terminated all recording processes gracefully once the time limit was reached.

It was important to be conscious of the size of the data so as not to exceed the storage capacity of the edge device. The data coming from the sensors was saved in directories corresponding to the different sensors in the system. The exception to this is the lidar sensors, as the data stream resulting from the capture architecture was merging the messages from both lidar.

At the end of the data capture session, the directories containing the captured data were compressed to allow for a robust and efficient transmission of data.

## d. Delivery and Distribution

The system at the rail-crossing hosts a 4G modem that allows for internet connectivity to the rest of the world via a static IP address. Using the IP address to the rail-crossing system, the edge computer can be accessed through the port forwarding rules set up on the system. With connectivity to the internet and accessibility to the edge machine, the data can be transmitted to a machine offsite for post processing and distribution. The protocol of choice was to use SCP to transfer the data since it provides a layer of security on the transmission pipeline. However, due to the large amount of data and the instability of the network, it was decided that a drop-in swappable storage is the best option to collect data. Through using controlled data collection sessions, the size of data can be determined beforehand to best suit the storage requirements.

The lidar data was processed to split the messages and group them first by sensor, then by groups of one minute duration, and finally compressed. Data from all the sensors was then uploaded to a secure cloud repository where the development teams have access to them.

## e. Challenges and Limitations

As mentioned earlier, one of the most critical challenges in the data capture process is the limitation in connectivity using the 4G modem. Every hour captured at the rail-crossing results in approximately 250 GB of compressed data. The transfer of this amount of data takes a long time over 4G connectivity. There is also the issue of frequent disconnection of the network, which causes corruption in the currently transmitted file and requires retransmission after the connection is re-established, thus the decision to move to swappable on site storage for data exchange.

The second challenge was in the use of a passively cooled edge machine that relies on dissipating heat via an attached heat sink. The intensity of the collection processes causes the system to heat up significantly, faster than the dissipation system can discharge generated heat. This causes the system to resort to throttling down the cycles of the CPU and jeopardizing the health of the collection processes. It was noticed that the longer the recording session, the more the quality of the collected frames degraded, showing a direct relation between this degradation and the down-throttling of the CPU cycles due to heat buildup.

# 4. Data

Data is collected from a suite of 3 complimentary types of sensors: two Cepton LiDAR scanners, two infrared-spectrum (thermal) Bosch cameras and a visible-spectrum (RGB) Bosch camera. Each of these sensors has its own set of disadvantages that restricts the situations in which they can be used with high accuracy.

Each sensor captures the rail-crossing scene in its own format. The resulting data is used to detect agents moving at the rail-crossing; those detections are in turn analyzed to identify dangerous situations for road users.

## a. Sensor Data

### i. LiDAR

The Cepton Vista P90 LiDAR is used to capture 3D point clouds of the rail-crossing scene. LiDAR sensors are resilient to poor weather conditions, but the processing of point clouds is computationally expensive. It is the only sensor in the suite to capture 3-dimensional information about the scene. This sensor would be used to detect pedestrians walking on the road when they shouldn't, triggering an alarm with a low false-positive rate.

### ii. Thermal

The Bosch Dinion IP Thermal 8000 Cameras capture data in the infrared spectrum at a resolution of 640x480. This makes them resilient to changes in visible lighting and in weather conditions at the cost of lower image resolution. This sensor can be used to detect people and animals on the road in poor weather and lighting conditions (e.g. night-time, heavy storms).

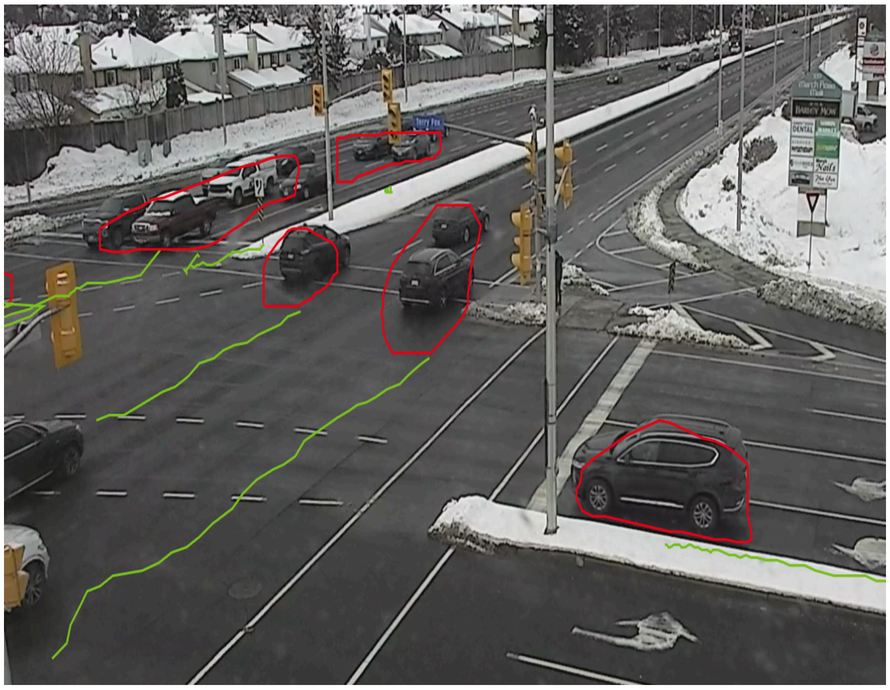
### iii. RGB

The Bosch Autodome IP Starlight 7000i camera captures light in the visible-spectrum at a high definition resolution of 1920x1080. This makes it the most understandable sensor type and the most susceptible to poor lighting and weather conditions. It can be used to track anything moving at the rail-crossing and to detect any kind of resulting unsafe behavior.

## b. Analysis

### i. Metadata from InfluxDB

Aside from the raw data collected from the sensors, the MHCorbin device collects and transmits information about the detections that were captured by the sensors' onboard AI. This data is transmitted to a cloud-based database. In general, the meta-data includes information about the position of the detected object, its classification, the times of detections and an ID for the tracked object. Meta-data from the Cepton lidars also includes information about the size and speed of the object.  The image below shows a sample of the detections performed by the onboard AI on a frame from the PTZ camera at the rail-crossing.



### ii. Metadata from inhouse AI processing

The inhouse AI processing used proprietary algorithms developed by Sensor Cortek to process the raw data from the Lidar sensor and produce information about the objects in the scene. This was essential for detecting the critical-case events identified for this project, as the information reported to the database was not sufficient to get a granular level of details for extracting instances of said events.

In order to perform the proper analysis to detect the event cases, the data will undergo three stages of analysis:

1. Detection: this stage involves an AI model detecting an object in the scene. The detection is used to identify the space the object occupies in 2d or 3d representation of the scene from the sensor data. For example, for cameras, the detection is going to be represented by a two dimensional bounding box the encapsulates the object and for Lidar a three dimensional bounding box in x,y,z direction to perform the same task.

2. Classification: this stage involves classifying the detected object under a specific class of objects. For example, in this study, the detected objects of interest are Person, Car, Bus, Truck and Cyclist.

3. Tracking: this step involves identifying the same object across multiple frames while it is in view of the sensor. Tracking classified objects is essential because actions performed in the scene are performed by an actor, and to attribute the actions to the corresponding actor it is necessary to identify correctly who performed the action.

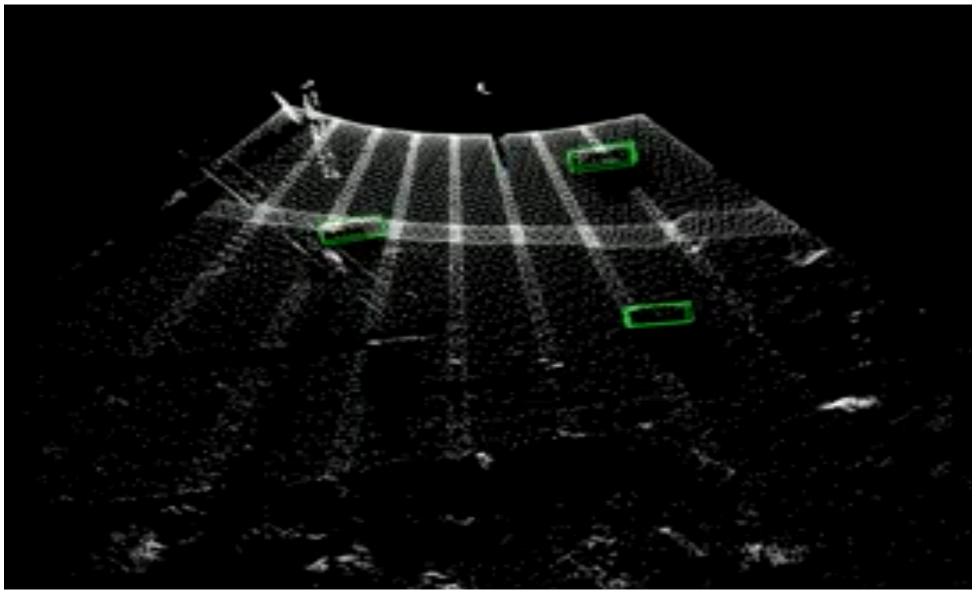
### iii. Lidar Object Detection

OpenPCDet served as the starting point for identifying a baseline for object detection using LiDAR, demonstrating its fundamental role in advancing the capabilities of LiDAR-based detection systems. OpenPCDet is an advanced, open-source project for LiDAR-based 3D object detection, developed under the OpenMMLab initiative. It is built on the PyTorch framework and supports various state-of-the-art 3D detection models, providing a flexible and scalable solution for point cloud data processing. The toolbox includes highly refactored codes for both one-stage and two-stage detection frameworks, making it adaptable for diverse detection tasks. One of its significant achievements is winning the Waymo Open Dataset challenge in multiple categories, showcasing its robustness and effectiveness in real-world applications.

The design of OpenPCDet emphasizes modularity and flexibility. It separates data and model handling with a unified point cloud coordinate system, which simplifies the extension to custom datasets. This separation allows for a more straightforward integration of new models and datasets. The toolbox supports a wide range of models, including PointPillar, SECOND, PartA2, and PV-RCNN, among others. This versatility is further enhanced by the support for distributed training and testing, enabling the use of multiple GPUs and machines to handle large datasets efficiently.

Moreover, OpenPCDet includes comprehensive features such as adaptive training sample selection (ATSS), RoI-aware point cloud pooling, and GPU-based 3D IoU calculation. These features improve the precision and efficiency of 3D object detection tasks. The project is continuously updated, with ongoing contributions that add support for new models and datasets, enhancing its capabilities and keeping it at the forefront of LiDAR-based 3D object detection technology. This makes OpenPCDet a valuable resource for researchers and developers working on advanced 3D perception projects.

1. Object Detection:



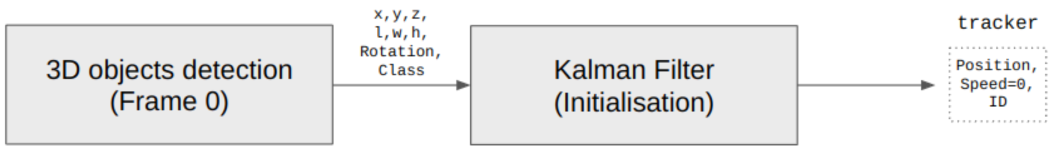
Using the PointPillar model that was trained on Kitti dataset provided insight on the preprocessing step of the data and the expected results from an off the shelf analysis algorithm. The results were good in some cases, but overall sporadic and not fit to perform accurate object tracking. In general, the finding from this part of the experiment is the following:

- Reasonable results for vehicles.
- No large vehicle detection (i.e. bus).
- More than expected false positive results.

2. Object tracking:

    a. Initialization

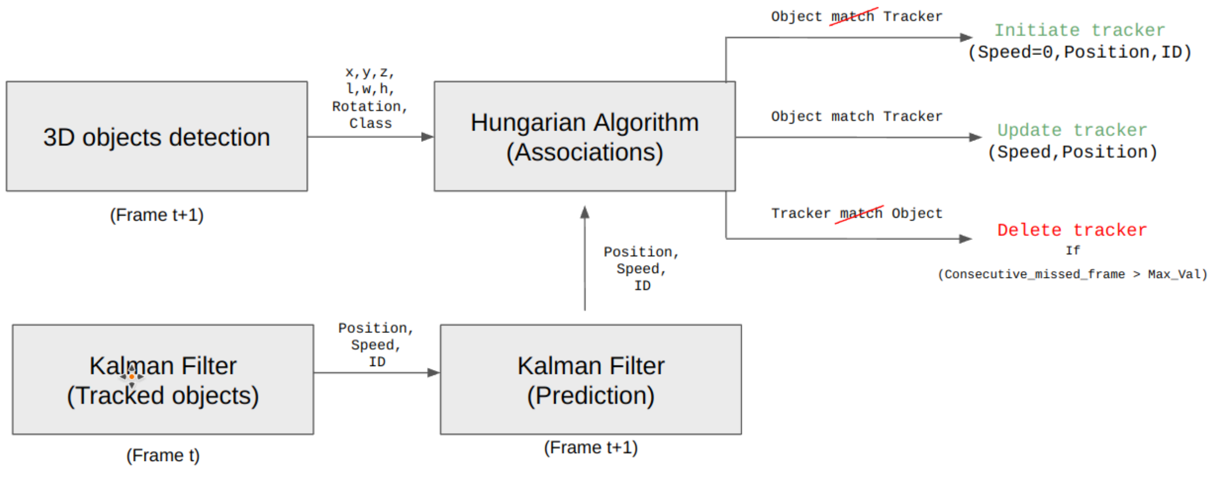
The initial step in the tracking process is using the Hungarian Algorithm and Kalman Filter for tracking 3D objects. It begins where objects are detected and their attributes, such as position (x, y, z), dimensions (length, width, height), rotation, and class, are identified. This information is then fed into the Kalman Filter (Initialization) block. The Kalman Filter uses these attributes to initialize the tracker, which outputs the position, initial speed (set to zero), and a unique ID for each tracked object. This process sets up the framework for subsequent tracking and updating the state of each object across subsequent frames.


3D objects detection (Frame 0)
x,y,z, l,w,h, Rotation, Class
Kalman Filter (Initialisation)
tracker
Position, Speed=0, ID

    a. Propagation

The second step of the algorithm is propagation, which involves updating and maintaining the state of tracked objects over time. In this step, "\3D objects detection for frame t+1 detects new objects and provides their attributes (x, y, z, length, width, height, rotation, and class). These attributes are fed into the Hungarian Algorithm (Associations) to associate detected objects with existing tracked objects. Simultaneously, the Tracked objects (using Kalman Filter) from frame t provide the predicted position, speed, and ID of each tracked object for frame t+1.

The Hungarian Algorithm matches detected objects with existing trackers. If a detected object matches a tracker, the tracker is updated with the new position and speed. If a detected object does not match any existing tracker, a new tracker is initiated with the object's position and an initial speed of zero. Conversely, if a tracker does not match any detected object, it is deleted if the number of consecutive missed frames exceeds a predefined threshold, for instance the maximum possible value. This process ensures that the system accurately tracks objects, updating their states or removing them if they are no longer detected over a specified period.



In essence, We perform a Lidar object detection using DSCAN from point cloud filtered by removing the road and static objects. Once done we use our identifier tracking algorithm to track the objects and their evolution during their movement across the rail-crossing. With that information we orient the bounding box around the points previously classified, this getting distinct bounding boxes with ID.

4. Preliminary results:

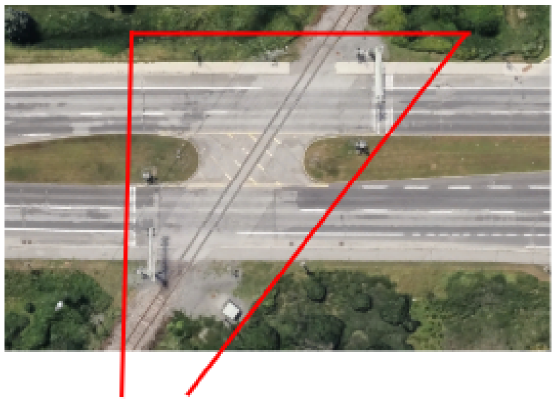
The preliminary results with a pretrained model showed the following:

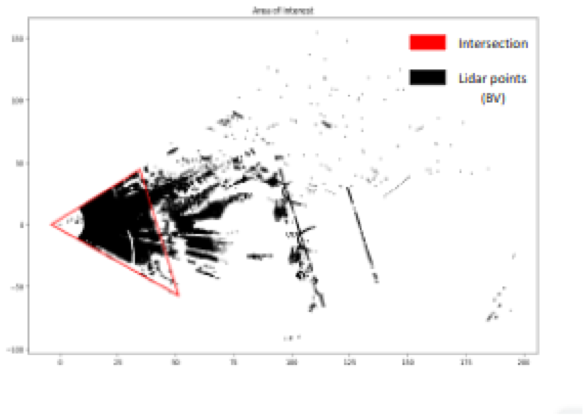
a. Bad ID tracking for pedestrians
b. No bus tracking
c. Noticeable false detections
d. Reasonable ID tracking for cars.
e. Standard data format for capturing detection and tracking info from lidar:
   Timestamp, label, x, y, z , l, w,h, rotation, object_ID

5. Generating ground-truth for Fine-tuning the DNN model:

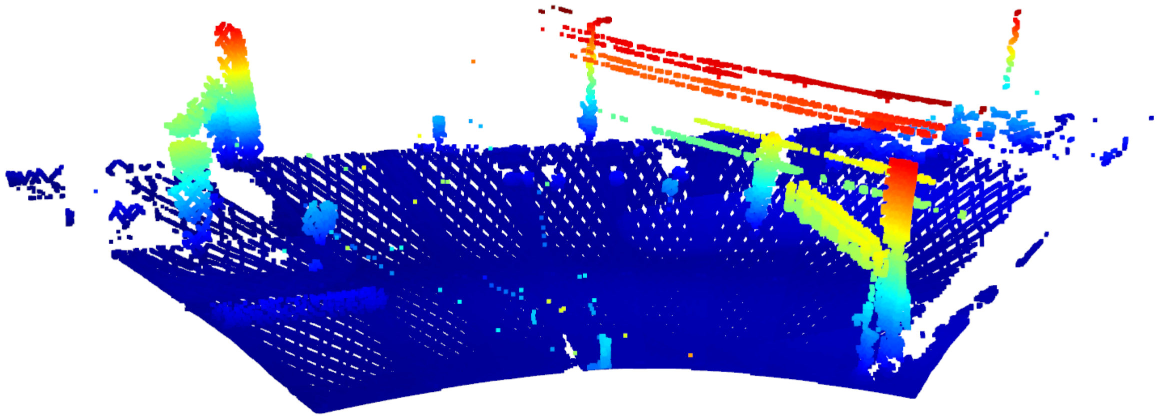
The process of fine-tuning the detection model requires understanding the input data from the sensor at the rail-crossing in order to generate the ground truth data. Below are the steps taken to generate the ground-truth data:

a. Region of interest: a significant portion of the lidar pointcloud is not usable as it falls outside of the region of interest. Therefore the first step is to create boundaries for the area of interest and use it to crop/filter the data in order to obtain only what is of interest.

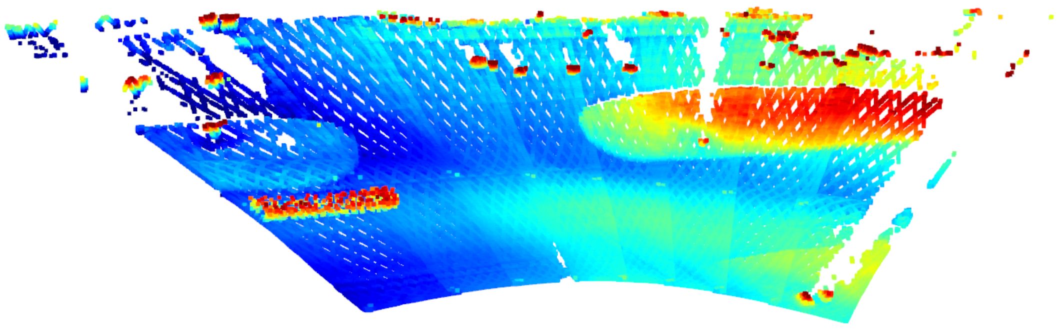




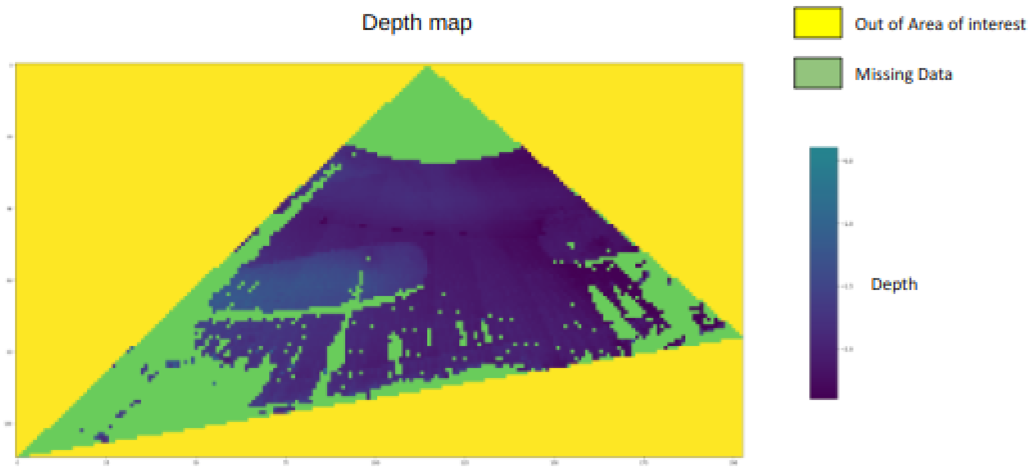
b. Static and dynamic objects separation: To detect moving objects effectively we must separate them from static objects (road,infrastructures …). We performed the merge lidar frames (100x) with an empty intersection to get a perfect scan of the surrounding.



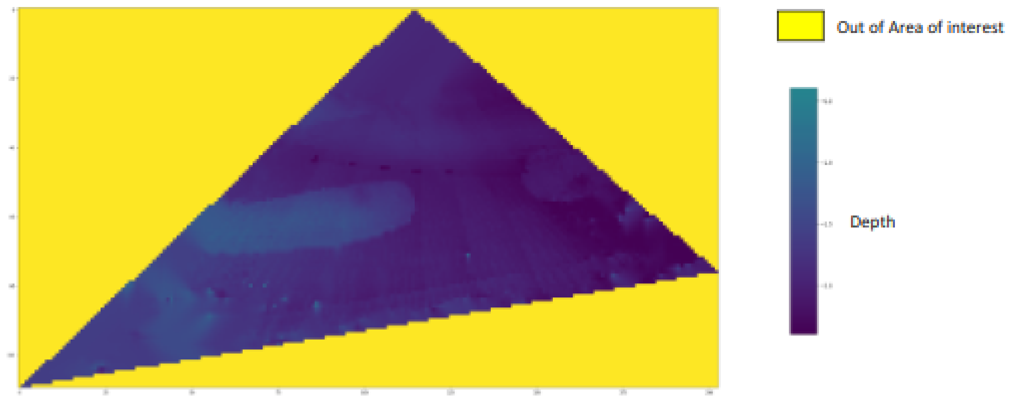
c. Road extraction: To perform a DBScan and detect each structure that is moving or not. We need to remove the road points and thus isolate each group of points. We notice that the topography of the road can't be assimilated to a plan



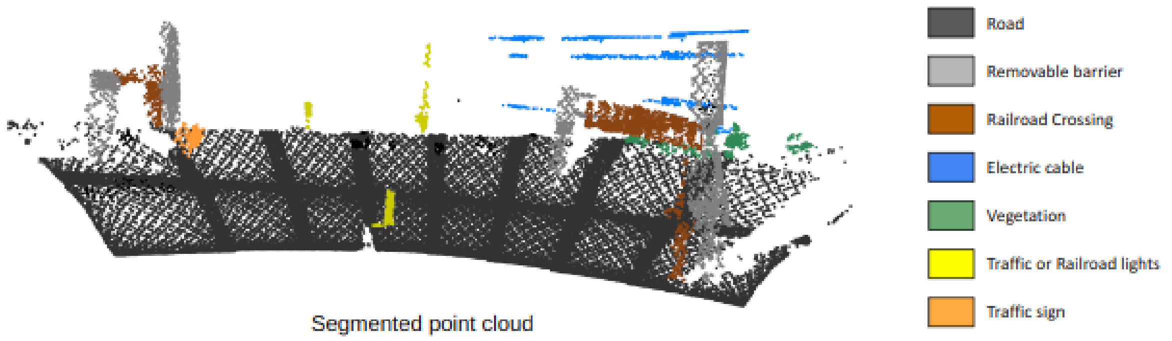
d. Road pointcloud: The existing road pointcloud is extracted from the data. In said data, there exists points outside of the area of interest, data of interest and missing data based on the delineated area from the rail-crossing.



e. Filling the missing data: Bilinear interpolation is used to estimate the missing data with sufficient accuracy such that we have a complete depth map of the road.

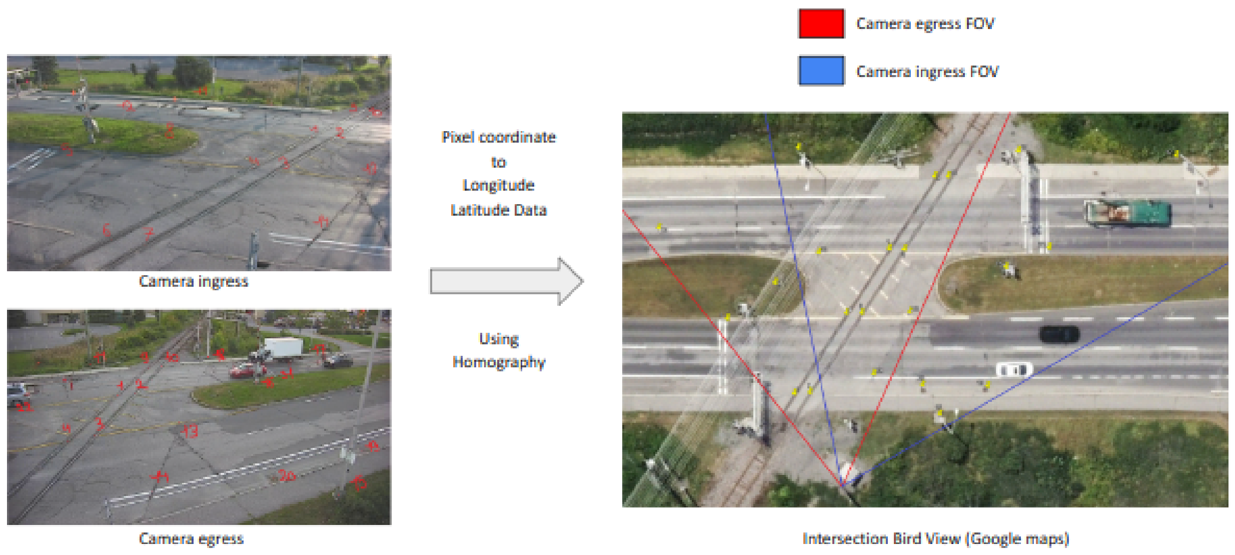


f. Static pointcloud segmentation: DBSCAN is employed to extract the separate static objects in the scene.

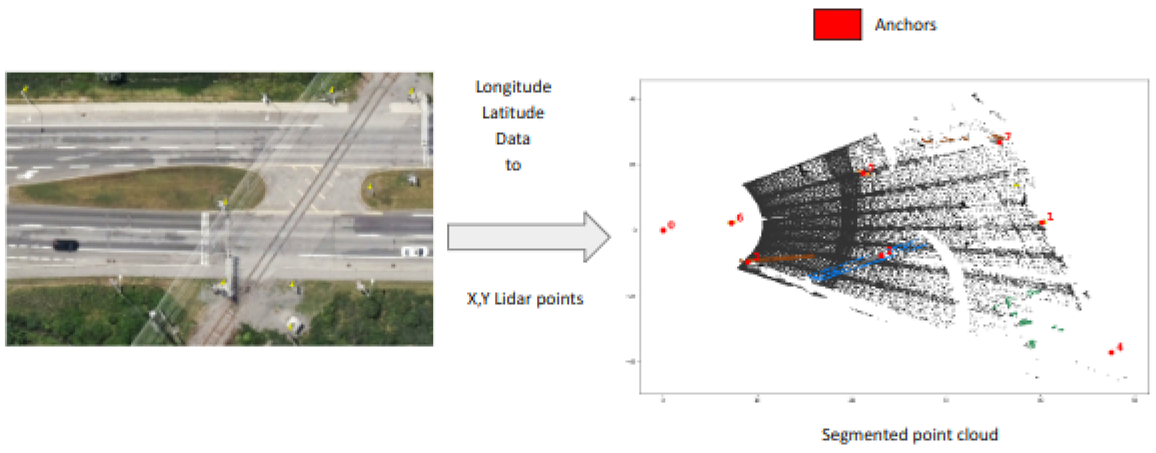


Segmented point cloud

g. Camera Detections: Classification of moving object data from RGB cameras at the rail-crossing


Camera egress FOV
Camera ingress FOV
Pixel coordinate to Longitude Latitude Data
Using Homography
Camera ingress
Camera egress
Intersection Bird View (Google maps)

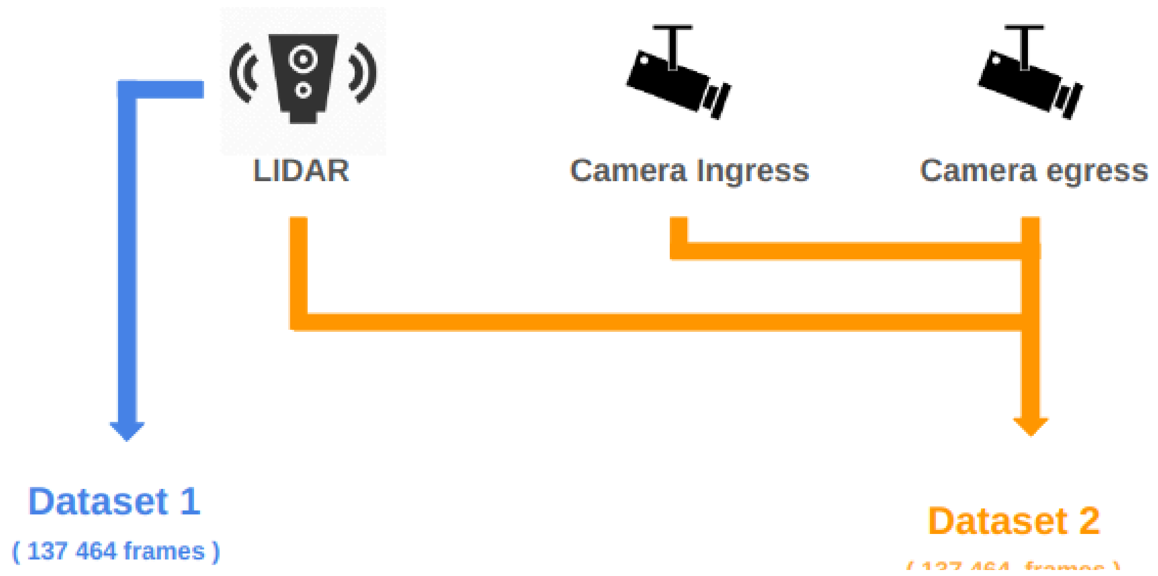
h. Camera domain to Lidar domain: The next step is to move the ground-truth from the camera domain to the Lidar domain



i. Camera detections results: Running a robust detection model on the camera data to generate the detections. It was found there was a significant number of frames missing from the Egress camera. Below are the results of the detections from the camera data.

| | **Camera Ingress** | | **Camera Egress** | |
|---|---|---|---|---|
| | Nb of objects | (%) | Nb of objects | (%) |
| Car | 1619730 | 97.61 | 58989 | 98.49 |
| Bus,Truck | 22375 | 1.35 | 587 | 0.98 |
| Person | 12219 | 0.74 | 180 | 0.30 |
| Cyclist | 3152 | 0.19 | 102 | 0.17 |
| Motorcyclist | 1847 | 0.11 | 38 | 0.06 |
| **Nb Frames** | **777 508** | | **36 010** | |

j. The generated dataset:



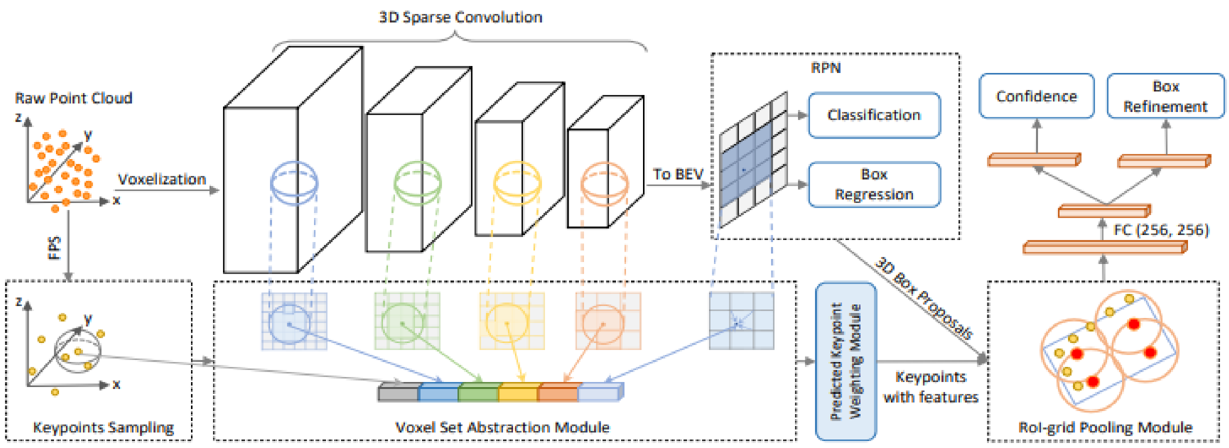
The dataset consists of two subsets, each obtained through different analysis approaches. While both subsets are extracted from the same number of frames , in this case, 137,464, the first was extracted from the dataset by using the object detector on the lidar data directly. The data was sorted by the size of the pointcloud to make the distinction between the large vehicles and small cars. This subset contained 1,071,825 small and 231,759 large vehicles. The large vehicles contributed to 17.8% of the total number of detected objects in this subdataset. Notice that this approach did not contain people or cyclists as the model failed to detect them.

The second subset was generated using ground-truth from lidar in conjunction with camera data. The process used in generating this dataset was more restrictive in order to eliminate noise in the data. Using the same amount of frames, the algorithm generated information for 498,067 cars, 92,559 buses and trucks, 6748 persons and 568 cyclists. The table below shows the same detailed with their percentage:

| | Nb of objects | (%) |
|---|---|---|
| Car | 498 067 | 83.3 |
| Bus,Truck | 92 559 | 15.48 |
| Person | 6 748 | 1.12 |
| Cyclist | 568 | 0.1 |

You'll notice that pedestrians and cyclists contributed to less than 1.5% of the dataset, and that reflected on the use case analysis as will be seen in the coming sections.

k. The DNN model: PV-RCNN, Point-Voxel Feature Set Abstraction for 3D Object Detection

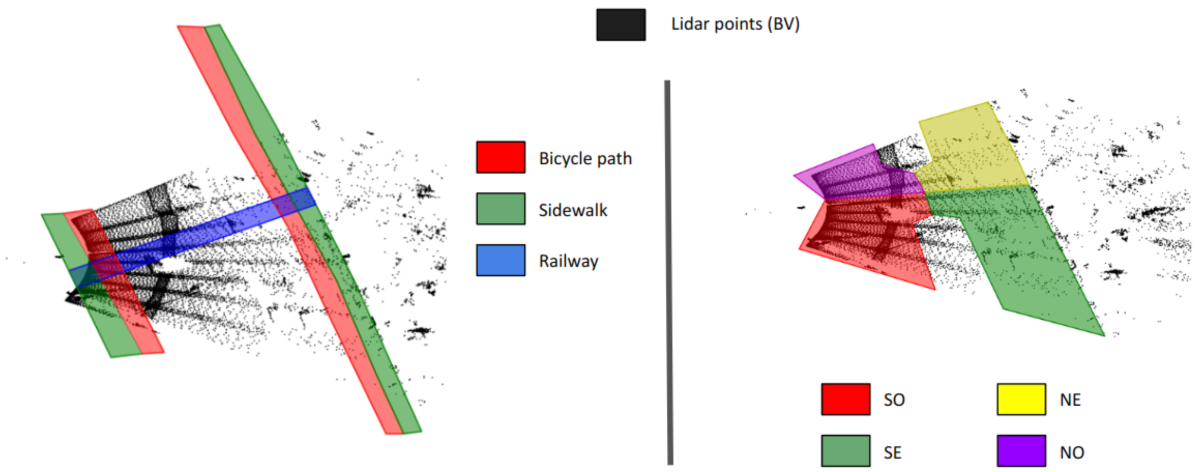


l. Fine-tuning results: Below are the results of fine tuning the detection model.

| Category | Metric | bbox | bev | 3D | aos |
|---|---|---|---|---|---|
| Small Vehicles | AP @0.70 | 90.2795 | 53.816 | 55.5547 | 75.18 |
| Small Vehicles | AP R40@0.70 | 96.1766 | 53.5797 | 58.1811 | 60.01 |
| Large Vehicles | AP @0.70 | 81.854 | 68.4443 | 62.3698 | 76.65 |
| Large Vehicles | AP R40@0.70 | 86.4675 | 67.4572 | 65.4272 | 81.24 |

## 6. Event Tracking

After establishing a strong object detector model through the fine-tuning process described above, the next step is to establish the bases to identify events through the actions of the detected and tracked objects. The area of the rail-crossing requires division to logical areas in order to address the events of interest. The image below identifies the division of the area used in the analysis in this document.



## 7. Summary

The use of 3D object detection models based on LiDAR, pre-trained on databases such as KITTI, Waymo, and NuScenes, provides a foundation for initial detection results. However, these models exhibit limitations in precision, with some clearly visible objects not detected, certain classes such as buses or bicycles missing, and poor performance in detecting pedestrians.

To address these issues, we developed a semi-automatic point cloud labeling tool to create our own training database, which includes cars, heavy vehicles, pedestrians, and bicycles. This approach enabled us to test the feasibility of the system for event detection more efficiently.

Our database consists of 137,464 annotated LiDAR frames, containing:

- 498,067 detected cars
- 92,559 detected heavy vehicles
- 568 detected bicycles
- 6,748 detected pedestrians

We trained the PointPillars model on this dataset and achieved significantly improved detection results compared to the pre-trained models.

Object tracking yielded promising outcomes, as anticipated:

- Larger objects are easier to detect.
- Objects closer to the LiDAR are more reliably detected.
- Pedestrians and bicycles are easily detected when they are in designated areas (sidewalks, cycle paths), but are often mislabeled as cars when on the road.

We integrated an identifier tracking program with the LiDAR detection model, which allows us to track vehicles and detect events effectively. The identifier tracking performs well, even in complex scenarios where a vehicle's detection is intermittent due to obstructions. As long as the detection gap does not exceed eight frames, the vehicle is usually tracked successfully.

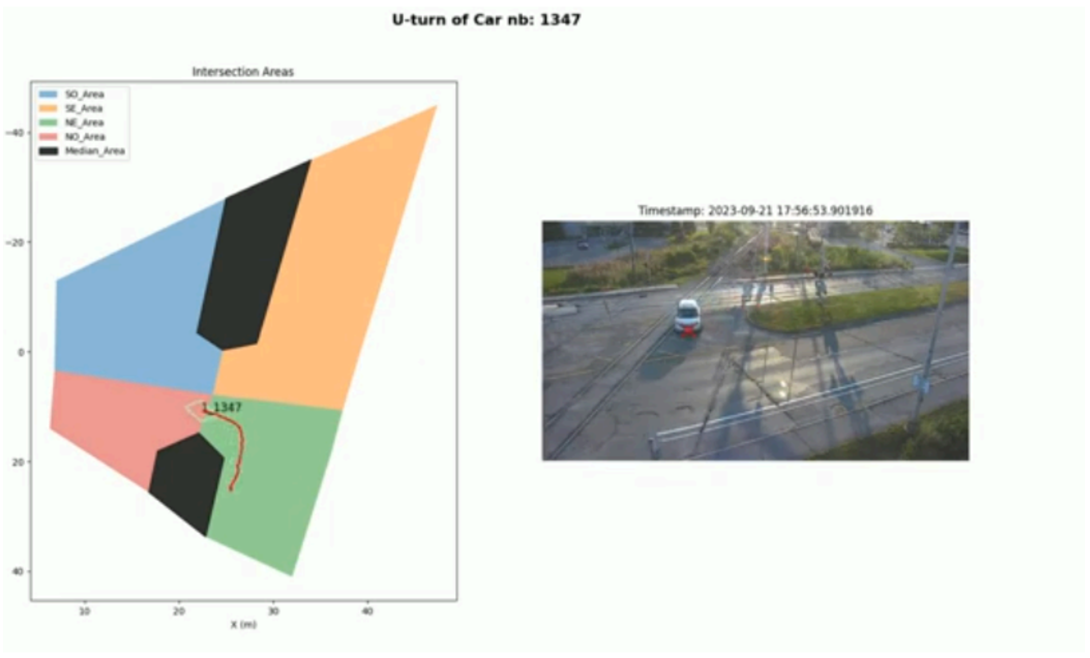
We believe that combining camera and LiDAR data could provide the most precise detection. This integration would leverage the tracking accuracy of cameras and mitigate the LiDAR's issues with poor or non-detection, while maintaining the high spatial precision crucial for event detection.

### iv. Proposed Event Cases

The table below lists the identified critical-case events identified for this project:

| Description | Actor | Condition |
|---|---|---|
| Jaywalking | Pedestrian | pedestrian crossing in danger areas |
| U-turnt | Vehicle | Vehicle Performing illegal u-turn |
| Dangerous-Stop | Vehicle | Vehicle stopping in dangerous area |
| Queuing | Vehicle | Vehicles queuing near/on the rail-crossing |

The image below shows an example of our analysis in capturing a U-turn event. The process captured all U-turn in the controlled ground-truth as well as the incidental one:



The table below shows the number of events captured by the analysis compared to the ground-truth:

| Description | # GT | # Detections |
|---|---|---|
| Jaywalking | 17 | 0 |
| U-turnt | 5 | 5 |
| Dangerous-Stop | 2 | 2 |
| Queuing | 14 | 14 |

The system failed to capture pedestrian data due to the low number of lidar points attributed to the pedestrian representation in the scene. This may be alleviated with more training involving pedestrian ground-truth from the scene, but ultimately, a sensor fusion approach that combines Lidar and camera data.

## v. Summary

The use of 3D object detection models based on LiDAR, pre-trained on databases such as KITTI, Waymo, and NuScenes, provides a foundation for initial detection results. However, these models exhibit limitations in precision, with some clearly visible objects not detected, certain classes such as buses or bicycles missing, and poor performance in detecting pedestrians.

To address these issues, we developed an automatic point cloud labeling tool to create our own training database, which includes cars, heavy vehicles, pedestrians, and bicycles. Although the annotations are less precise than manual annotation, this approach enabled us to test the feasibility of the equipment for event detection more efficiently.

Our database consists of 137,464 annotated LiDAR frames, containing:

- 498,067 detected cars
- 92,559 detected heavy vehicles
- 568 detected bicycles
- 6,748 detected pedestrians

We trained the PointPillars model on this dataset and achieved significantly improved detection results compared to the pre-trained models.

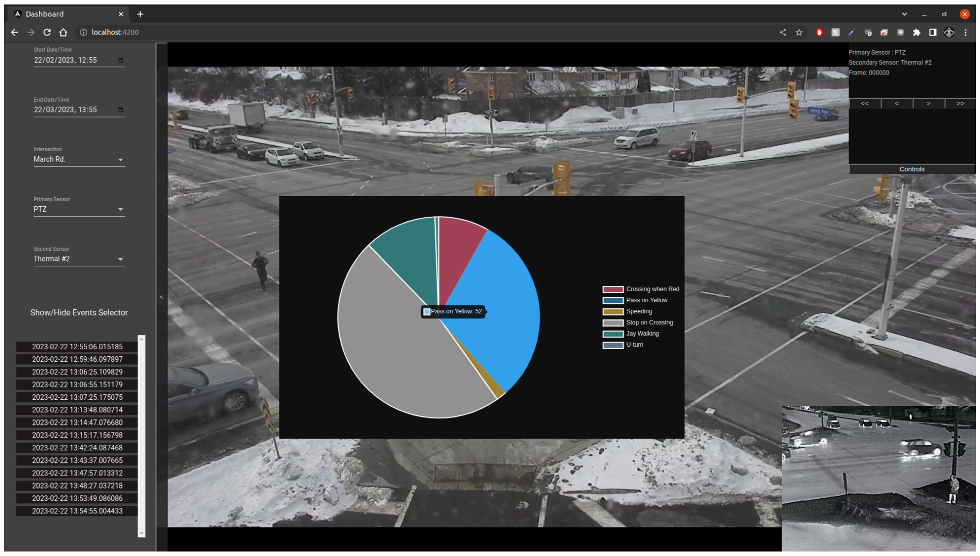
Object tracking yielded promising outcomes, as anticipated:

- Larger objects are easier to detect.
- Objects closer to the LiDAR are more reliably detected.
- Pedestrians and bicycles are easily detected when they are in designated areas (sidewalks, cycle paths), but are often mislabeled as cars when on the road.

We integrated an identifier tracking program with the LiDAR detection model, which allows us to track vehicles and detect events effectively. The identifier tracking performs well, even in complex scenarios where a vehicle's detection is intermittent due to obstructions. As long as the detection gap does not exceed eight frames, the vehicle is usually tracked successfully.

We believe that combining camera and LiDAR data could provide the most precise detection. This integration would leverage the tracking accuracy of cameras and mitigate the LiDAR's issues with poor or non-detection, while maintaining the high spatial precision crucial for event detection.

# 5. Dashboard



Start Date/Time: 22/02/2023, 12:55

End Date/Time: 22/03/2023, 13:55

Intersection: March Rd.

Primary Sensor: PTZ

Second Sensor: Thermal #2

```
2023-02-22 12:55:06.015185
2023-02-22 12:59:46.097897
2023-02-22 13:06:25.109829
2023-02-22 13:06:55.151179
2023-02-22 13:07:25.175075
2023-02-22 13:13:48.080714
2023-02-22 13:14:47.076680
2023-02-22 13:15:17.156798
2023-02-22 13:42:24.087468
2023-02-22 13:43:37.007665
2023-02-22 13:47:57.013312
2023-02-22 13:48:27.037218
2023-02-22 13:53:49.086086
2023-02-22 13:54:55.004433
```

Primary Sensor : PTZ
Secondary Sensor: Thermal #2
Frame: 000000

## a. Purpose

A dashboard is an essential tool that provides a quick overview of the events captured by a suite of sensors installed at a rail-crossing. The purpose of a dashboard is to display real/non-real-time data collected by these sensors, allowing traffic engineers to monitor and manage traffic flow, identify potential issues, and make data-driven decisions. The dashboard provides a summary of key performance indicators such as traffic volume, average speed, and number of accidents, helping traffic engineers to identify patterns, trends, and anomalies that could affect the overall performance of the rail-crossing.

## b. Functionality

The functionality of the dashboard falls into the following categories:

i. Time Frame selection: The dashboard offers the ability to select a start time and end time down to the minute .

ii. Sensor selection: The dashboard offers the ability to select which sensor to display on the main area of the screen as a primary sensor and which to display as a secondary sensor to show in the bottom right corner.

iii. Frame navigation: The dashboard offers the ability to navigate between frames one at a time, or a configurable "skip" number in which a certain number of frames is skipped between the viewable frames. This number is configurable, but defaults to 5.

iv. Frame information: The dashboard displays information about the current sensors, primary and secondary as well as the current frame name or number.

v. Critical events: The dashboard offers the ability to display critical events detected in a selected timeframe through two mechanism:

   1. Summary: the dashboard summarizes all critical events in one graphical representation from which the user can obtain general information about the critical events that happened within the selected period.

   2. List: The dashboard displays specific information about occurrences of a specific critical event category.

   3. Single event: the dashboard displays an individual instance of a critical event category on the main display with specific annotation to highlight and localize the detected action in the frame.

vi. Viewability: all panels and dialogue boxes are collapsible/hideable in order to permit maximum viewability of the frames.

vii. The dashboard for this project is a static demonstration. There is no link to a database. However, the selection from within the predetermined time frame is dynamic and the critical events are actual critical events that were identified in the data for that period.

## c. Framework

The underlying technology of the dashboard consists of two components: a front-end and a back-end.

### i. Frontend

Angular was selected as the framework to build the front end as it is very suitable for interactive applications such as the dashboard. Angular uses a combination of the Typescript, Javascript and HTML languages.

### ii. Backend

The back end for this iteration of the demo was an apache http server modified to serve the large quantity of data captured by the sensors.

## d. Data formats

The dashboard accepts specific formats for the data it depicts. These formats cover the frames of the sensor, the file types, and the data structure of how the data relate to each other. The following table shows the different formats for each sensor.

| Sensor | Format |
|---|---|
| LiDAR | PCD |
| Thermal | PNG, JPG |
| PTZ | PNG, JPG |

The file that communicates the temporal property of the data and the relationship between the different outputs of the different sensor is specified in a JSON format. Appendix A  is the schema expected of the JSON file describing the critical events.

## e. Data synchronization

All the sensors are producing frames at a rate of 10 frames per second[1]. Therefore, the temporal synchronization between the frames is done by assigning a principal sensor to be used as a reference. For every sensor, its synchronized frame is then the one with the smallest temporal difference from the reference frame of the principal sensor.

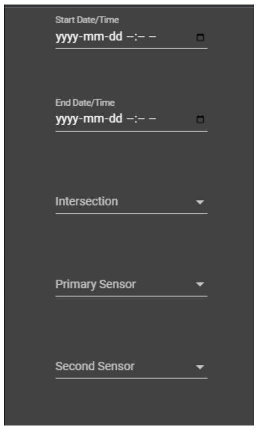
## f. UI Elements

### i. Navigation panel

The navigation panel of the dashboard serves two main functions. The first is to provide a filtration mechanism to target a specific period of events and select the desired sensors for the primary and secondary display (described below). The second is to control the visibility of the critical event's selection and viewing found in the timeframe which was selected using the filtration mechanism. The navigation panel is collapsable for a full screen view of the displayed frame.

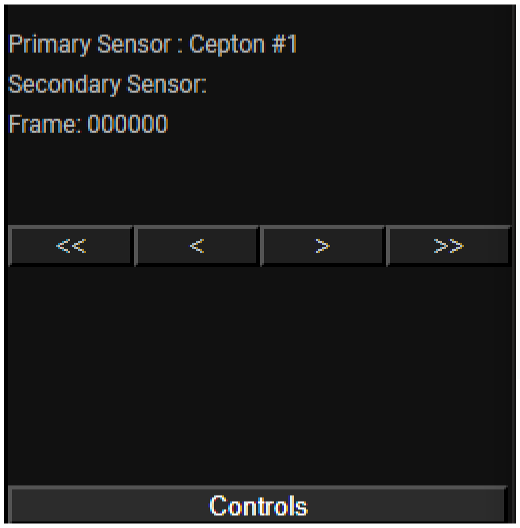
Below is an illustration of the dashboard's navigation panel.

[1] The PTZ sensor is not producing data at a consistent 10 fps rate, but more around 10 to 13 fps.



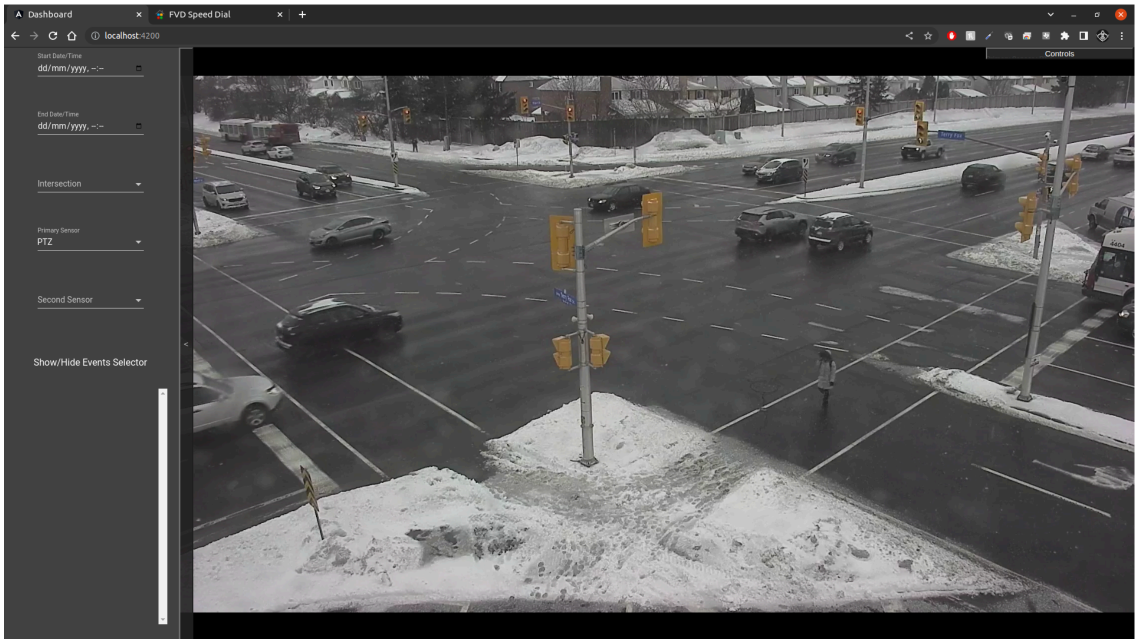
## ii. Control panel

The control panel contains information about the selected sensor for primary display, the selected sensor for the secondary display and the current frame being displayed. It also hosts the control buttons that allow navigation between frames. The navigation buttons allow for movement between adjacent frames, as well as moving by a certain amount of frames forward or backward. The “skipping” portion of the navigation is configurable with a default value of 5. The control panel is collapsable for a full screen view of the displayed frame.
Below is a depiction of the control panel.


Primary Sensor : Cepton #1
Secondary Sensor:
Frame: 000000

### iii. Main display

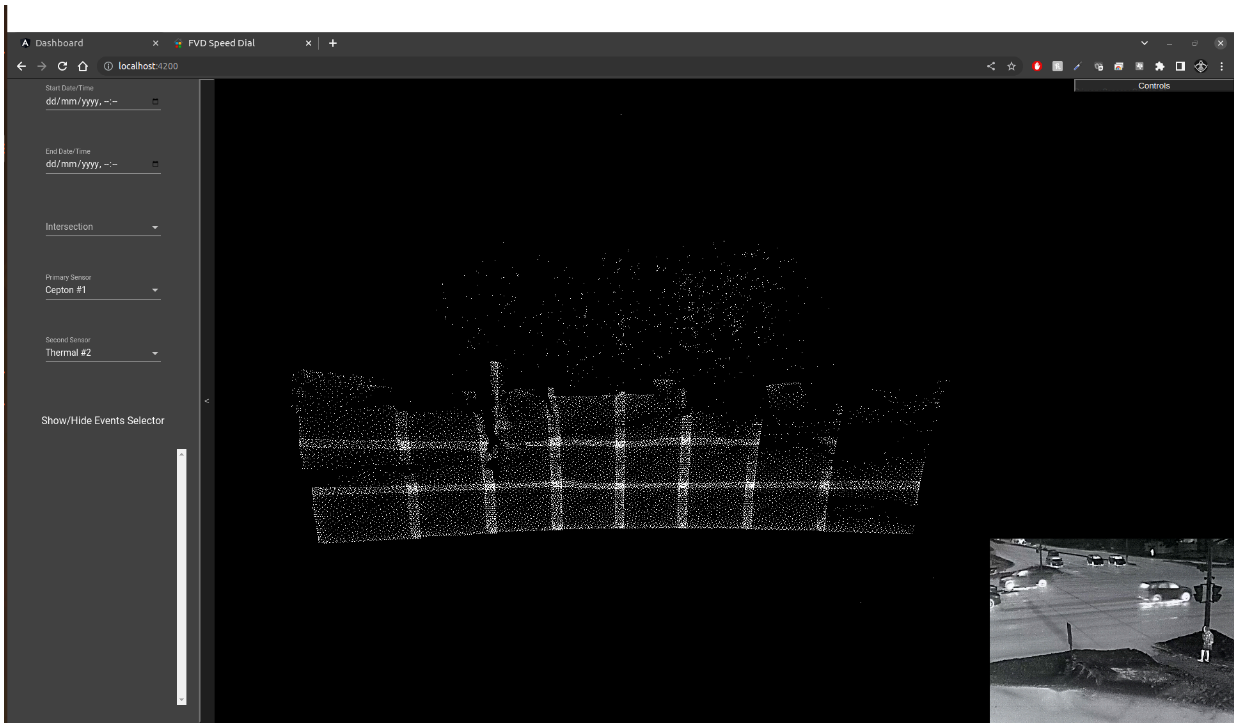
The main display refers to the major area of the screen occupied by the frame from the sensor that was selected as primary in the navigation panel. The purpose of designating a sensor as primary is to allow its frames to occupy the maximum viewable area for studying, examining or analyzing. Below, only the primary sensor is displayed (the PTZ camera, in this instance).



Start Date/Time: dd/mm/yyyy, --:--
End Date/Time: dd/mm/yyyy, --:--
Intersection
Primary Sensor: PTZ
Second Sensor

### iv. Secondary display

As with the primary display, the user can choose a secondary sensor to display its frame in the bottom right corner of the screen. The secondary sensor display offers another view of the scene with a different sensor technology to help study actions or events at the rail-crossing. The user can select the secondary sensor to display from the navigation panel. Due to limitations within the browser, the user is limited in the choice of the secondary sensor display to the thermal and RGB sensors. Below is shown an example with the lidar as the primary sensor and a thermal camera as the secondary sensor in the bottom right corner.

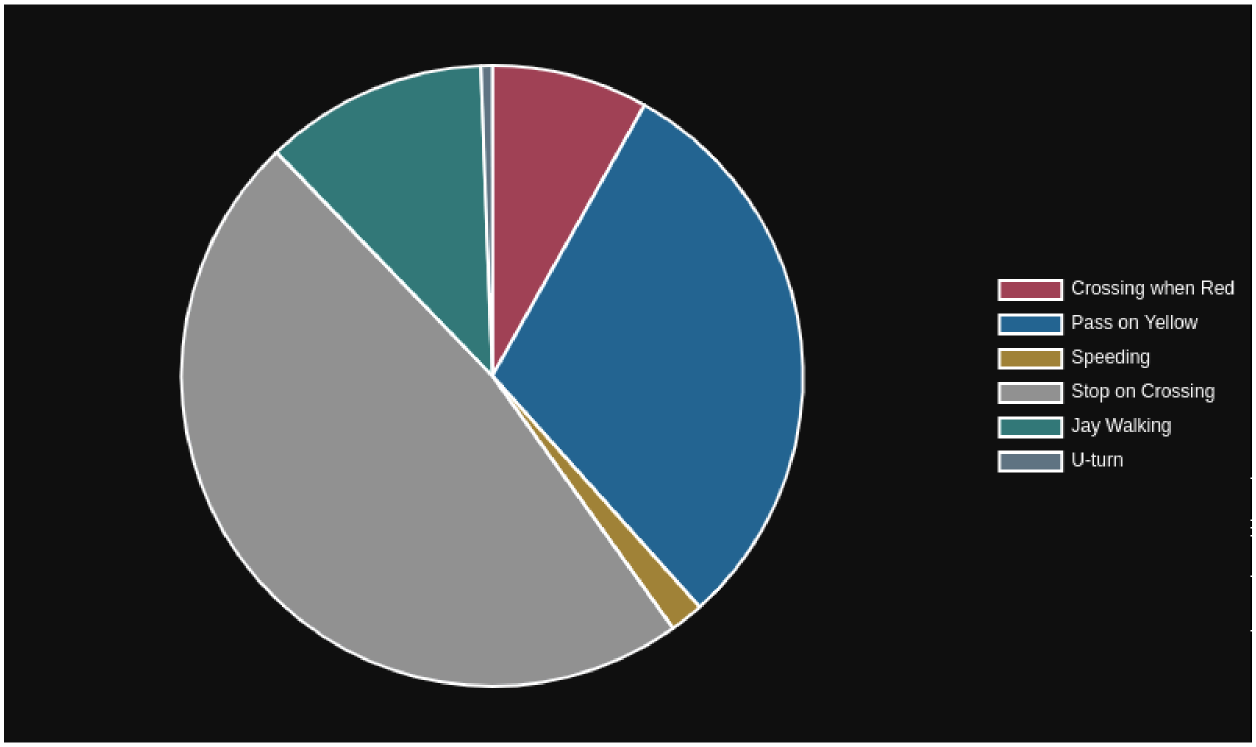


## v. Critical Events Display

Apart from displaying synchronized sensor frames, the dashboard can show and navigate between identified critical events captured by the sensors at the rail-crossing. The dashboard can do this in two steps through the components discussed herein:

### 1. Pie-Chart Selector

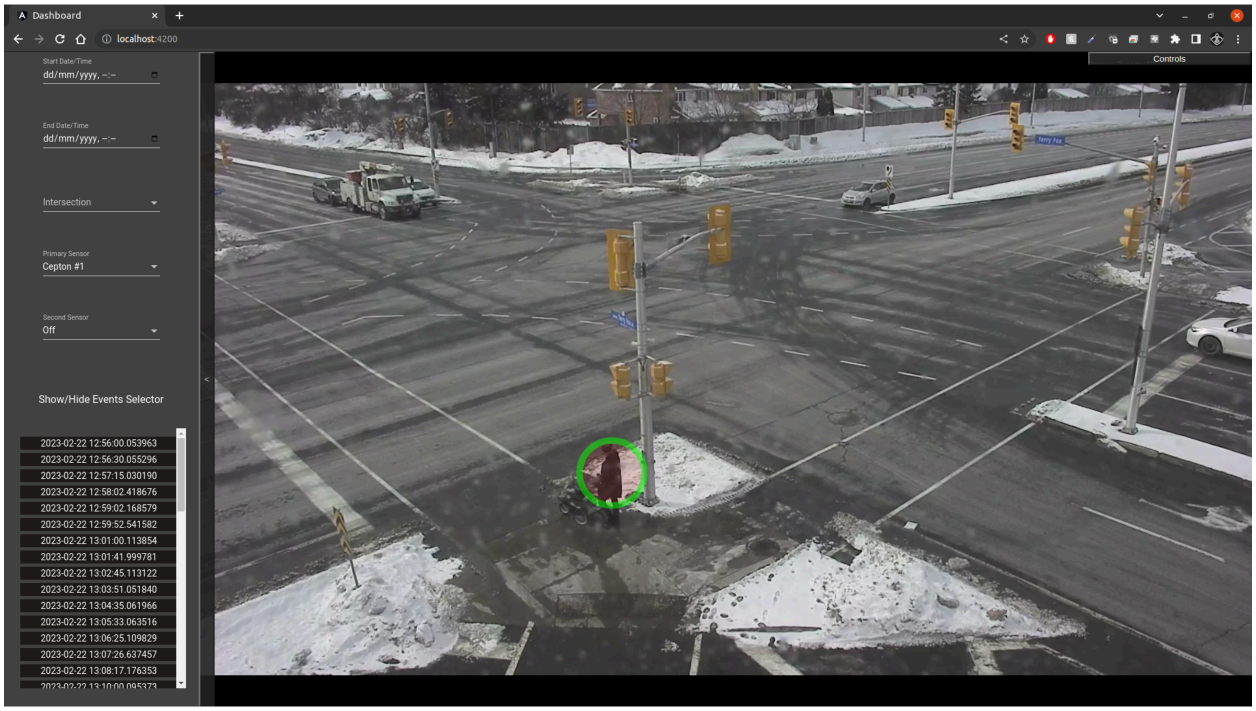
The event selector is a dynamic pie chart summarizing the critical events recorded at the rail-crossing. The user can hover over a part of the pie chart to get information about the number of occurrences of a specific category. The user can also click on any element in the legend to hide or show the related representation on the graph. This action permits the user to hide events that overwhelm the display and focus on events that occur less frequently. Below is shown the pie chart of critical events.



## 2. Event List

The event list is the second step in exploring the details of a specific critical event. While the pie chart displays the general information about all the different categories of events within a timeframe, the event list shows all the instances of a specific category within the same timeframe. The event list is populated when the user clicks on a section of the pie chart.

When the user selects an event from the list, the dashboard displays the corresponding frame on the main sensor display, and it highlights the specific event with a circle that is drawn by the dashboard on the frame. Below is shown an example of a selected event and the corresponding frame on the main display.



Start Date/Time
dd/mm/yyyy, --:--

End Date/Time
dd/mm/yyyy, --:--

Intersection

Primary Sensor
Cepton #1

Second Sensor
Off

```
2023-02-22 12:56:00.053963
2023-02-22 12:56:30.055296
2023-02-22 12:57:15.030190
2023-02-22 12:58:02.418676
2023-02-22 12:59:02.168579
2023-02-22 12:59:52.541582
2023-02-22 13:01:00.113854
2023-02-22 13:01:41.999781
2023-02-22 13:02:45.113122
2023-02-22 13:03:51.051840
2023-02-22 13:04:35.061966
2023-02-22 13:05:33.063516
2023-02-22 13:06:25.109829
2023-02-22 13:07:26.637457
2023-02-22 13:08:17.176353
2023-02-22 13:10:00.095373
```

## g. Future considerations

Future consideration for the dashboard involve two aspects:

### i. Dashboard development:

This aspect involves the introduction of configuration files that will allow the user to customize the interface in many ways, but mainly, adding sensors, sensor data format, and critical events categories.

### ii. Backend development

This aspect will focus on developing a node js server to manage the connection between the database containing the metadata , the sensor data repositories and the frontend. In a web application, a Node.js server can play a critical role in connecting a database of metadata, a data repository, and a frontend. The server acts as a middleware layer that facilitates communication between the frontend and the backend systems. The database of metadata can contain information about the application's data models, event analysis from sensors, and schema. The data repository, on the other hand, can store the sensor data, such as the frames that are produced from each sensor, sensor profiles, sensor debug messages, and other content. The frontend, which was described earlier, provides the user interface for interacting with the application.

The Node.js server is responsible for processing user requests, fetching data from the database and data repository, and returning the requested data to the frontend. The server can use various frameworks and libraries, such as Express.js and Mongoose, to handle HTTP requests, route the requests to the appropriate endpoints, and interact with the database and data repository. By connecting the frontend, database of metadata, and data repository, the Node.js server enables developers to build dynamic, responsive, and data-driven web applications that can handle large amounts of traffic and user interactions.

# Appendix A : JSON Schema for Critical Events

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "1": {
      "type": "array",
      "items": [
        {
          "type": "object",
          "properties": {
            "timestamp": {
              "type": "string"
            },
            "coords": {
              "type": "array",
              "items": [
                {
                  "type": "integer"
                },
                {
                  "type": "integer"
                }
              ]
            },
            "objID": {
              "type": "integer"
            },
            "caseID": {
              "type": "integer"
            }
          },
          "required": [
            "timestamp",
            "coords",
            "objID",
            "caseID"
          ]
        }
      ]
    }
  },
  "required": [
    "1"
  ]
}
```