Local and Cooperative Autonomous Vehicle Perception from Synthetic Datasets

by

Braden Hurl

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master's of Math
in
Computer Science

Waterloo, Ontario, Canada, 2019

Abstract

The purpose of this work is to increase the performance of autonomous vehicle 3D object detection using synthetic data. This work introduces the Precise Synthetic Image and LiDAR (PreSIL) dataset for autonomous vehicle perception. Grand Theft Auto V (GTA V), a commercial video game, has a large, detailed world with realistic graphics, which provides a diverse data collection environment. Existing works creating synthetic Light Detection and Ranging (LiDAR) data for autonomous driving with GTA V have not released their datasets, rely on an in-game raycasting function which represents people as cylinders, and can fail to capture vehicles past 30 metres. This work describes a novel LiDAR simulator within GTA V which collides with detailed models for all entities no matter the type or position. The PreSIL dataset consists of over 50,000 frames and includes high-definition images with full resolution depth information, semantic segmentation (images), point-wise segmentation (point clouds), and detailed annotations for all vehicles and people. Collecting additional data with the PreSIL framework is entirely automatic and requires no human intervention of any kind. The effectiveness of the PreSIL dataset is demonstrated by showing an improvement of up to 5% average precision on the KITTI 3D Object Detection benchmark challenge when state-of-the-art 3D object detection networks are pre-trained with the PreSIL dataset. The PreSIL dataset and generation code are available at https://tinyurl.com/y3tb9sxy

Synthetic data also enables data generation which is genuinely hard to create in the real world. In the next major chapter of this thesis, a new synthetic dataset, the TruPercept dataset, is created with perceptual information from multiple viewpoints. A novel system is proposed for cooperative perception, perception including information from multiple viewpoints. The TruPercept model is presented. TruPercept integrates trust modelling for vehicular ad hoc networks (VANETs) with information from perception, with a focus on 3D object detection. A discussion is presented on how this might create a safer driving experience for fully autonomous vehicles. The TruPercept dataset is used to experimentally evaluate the TruPercept model against traditional local perception (single viewpoint) models. The TruPercept model is also contrasted with existing methods for trust modeling used in ad hoc network environments. This thesis also offers insights into how V2V communication for perception can be managed through trust modeling, aiming to improve object detection accuracy, across contexts with varying ease of observability.

Acknowledgements

I would like to first thank my supervisors, Professor Krzysztof Czarnecki and Professor Steven Waslander. Their weekly guidance has helped steer the project, sparked my curiosity, and fuelled my passion for this work. Professor Robin Cohen also provided invaluable guidance and editing for the third chapter of this thesis. Her expertise in trust modelling inspired the chapter and paved the way for the research project. I would also like to thank all of the members of the WISE and WAVE labs, particularly Matthew Angus, Mohamed El Balkini, and Samin Khan, who provided me with invaluable guidance to working with Grand Theft Auto V.

Dedication

This thesis is dedicated to my parents, Doug and Lisa Hurl. I would not be where I am today without their continuous support and dedication to my well-being and personal development.

Table of Contents

List of Tables

List of Figures

Abbreviations

ADS	Automated Driving System 1
AP	Average Precision 2
APICC	Average Per Image Containing Class 25
AV	Autonomous Vehicle 1
FOV	Field of View 9
GNSS	Global Navigation Satellite System 49
GPU	Graphics Processing Unit 21
GTA V	Grand Theft Auto V 5
IoU	Intersection Over Union 3
IV	Intelligent Vehicle 34
LiDAR	Light Detection And Ranging 1
MANET	Mobile Ad Hoc 41
RSU	Road Side Unit 34
V2V	Vehicle-to-Vehicle 5
VANET	Vehicular Ad Hoc Network 34

Chapter 1

Introduction

Fully Autonomous Vehicles (AVs) are vehicles which do not require a human operator; they use an Automated Driving System (ADS) to drive autonomously. AVs are becoming increasingly prevalent on roadways. Waymo vehicles have now driven autonomously for over 10 million miles [68] and BMW has been testing autonomous vehicles since 2011 [1]. The list of companies racing to offer full AVs to customers has grown to be substantial. As AV development continues, it is essential to prioritize road safety.

AVs need to be able to locate where they are in the environment, where important roadway features lie, and where other objects on the road are located. These are important primary tasks for collision avoidance. AVs are typically outfitted with cameras and one or more Light Detection And Ranging (LiDAR). These sensors are the base of the perceptual systems and extremely important as a missed detection could result in a collision. A fatal crash between an autonomous Uber vehicle and a pedestrian occurred in March 2018. The National Traffic Safety Board (NTSB) released a preliminary report stating that "the system first registered radar and LiDAR observations of the pedestrian about 6 seconds before impact" [8]. A major factor in the collision was that the "software classified the pedestrian as an unknown object, as a vehicle, and then as a bicycle with varying expectations of future travel path." The perception algorithms today are far from perfect and any improvements in perceptual accuracy could save lives.

Dynamic object detection is a fundamental capability required for AV navigation and operation in dynamic environments. Significant progress has been made on 2D object detection in the last decade, with seminal works such as AlexNet [41] and ResNet [30] leading the way. Some datasets provide data specifically for AVs as the detection problems are limited to driving scenes. On the KITTI Object Detection Benchmark for autonomous driving [24], 2D object detection methods that identify bounding boxes of objects in the image plane are now consistently over 90% Average Precision (AP). For 3D AV perception challenges, which must identify the position and extend an object bounding box in three dimensions, LiDAR point clouds are commonly used as a source of depth information to augment image data. LiDAR point clouds are created by pulsing lasers and sensing the return strength and time to create a set of points in 3D space. Training deep learning methods on large, sparse, unordered point clouds is challenging and there remains a large gap between 2D and 3D object detection scores.

For the purpose of this thesis, the focus is on the 3D object detection task of the perceptual pipeline. 3D object detection is the task of positioning a 3D bounding box around objects. The Karlsruhe Institue of Technology has created a benchmark, the KITTI vision benchmark suite [24], to compare research for AV perception tasks such as 3D object detection. For KITTI, the object classes are vehicles, cyclists, and pedestrians. The dataset is comprised of 2D images, 3D point clouds obtained from a Velodyne LiDAR, and detection labels. A sample of an image is provided in Figure 1.1 with the corresponding point cloud (image colours are projected onto each 3D point). Detections are also present as 3D bounding boxes in orange (pedestrians), red (vehicles), and burgundy (cyclists).

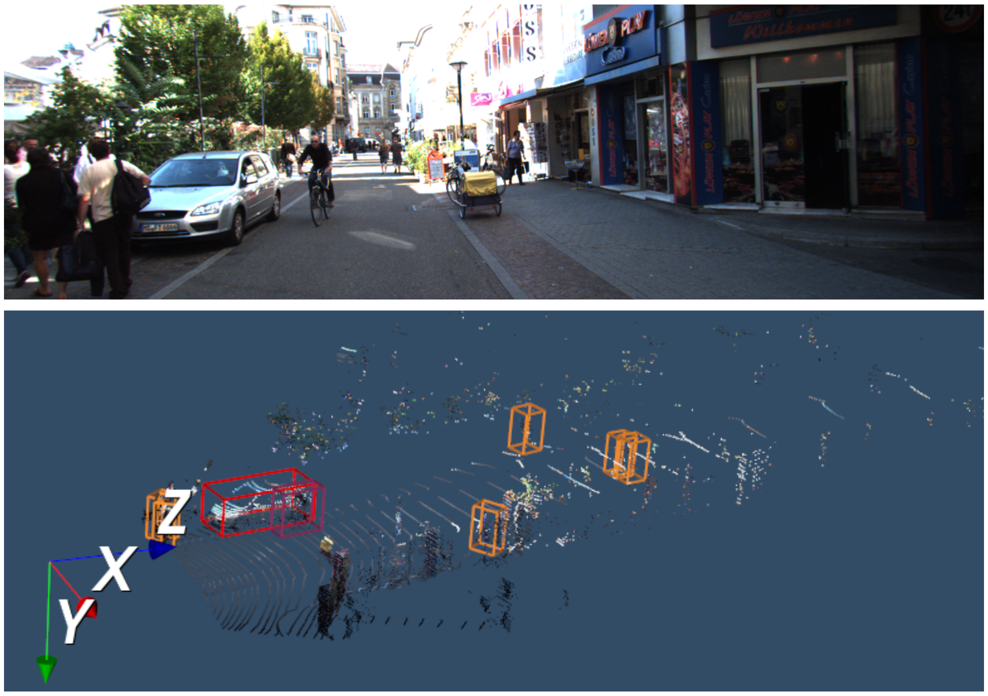

Figure 1.1: KITTI 2D image and LiDAR point cloud. Part of the data provided for the KITTI 3D Object Detection benchmark challenge [24]. Image colours are projected onto the point cloud. 3D bounding boxes are shown for vehicles (red), pedestrians (orange), and cyclists (burgundy).

The scores of the top submissions on the KITTI 3D object detection challenge are provided in Table 1.1. The scores are measured with AP, a metric which balances precision and recall scores where a detection is considered a true positive if the 3D bounding box has at least a 50 percent Intersection Over Union (IoU) with a ground truth bounding box (one per ground truth bounding box). The classes are easy, moderate (mod.), and hard, which are determined based on object occlusion, object truncation, and bounding box height in the 2D image. As can be seen, there is substantial room for improvement. The pedestrian class results are particularly poor compared to the car class.

Table 1.1: KITTI 3D Object Detection Top Results as of August 11, 2019 [24], [67], [62], [71]

Benchmark	Easy	Mod.	Hard
Car	88 %	78 %	76 %
Pedestrian	57 %	47 %	42 %
Cyclist	80 %	65 %	58 %

With more publicly available training data this gap could be narrowed. The data that is publicly available has large class-imbalances; the KITTI 3D Object detection training dataset consists of 28,742 cars and only 4,487 pedestrians and 1,627 cyclists. This significant class imbalance is likely playing a role in the disparity between car and pedestrian 3D detection performance, as the difference between top results on the two classes is over 30 percent AP. This is in large part attributed to a lack of training samples. Additionally, pedestrians are smaller objects (fewer LiDAR points), change shape while in motion, resemble poles, and are more varied in their appearance.

For modern deep learning methods, more data typically leads to increased performance. Unfortunately, large datasets can be time-consuming and expensive to create, partially a result of the resource intensive process of human-annotating 3D point clouds from LiDAR scanners. Bounding and orienting objects in 3D space is a difficult and time-consuming task; the SUN RGB-D dataset [63] reported an average of 114 seconds of annotation time per 3D object instance. Lee et al. [43] use pre-trained models to help speed up labelling of 3D object detection datasets for autonomous driving. They reduce the annotation process time by almost 95 percent with only a slight reduction in quality. The dearth of large, publicly available datasets is now becoming a problem of the past as new datasets such as ApolloScape [32], nuScenes [9], Argoverse [10], and the Waymo Open Dataset[1]) become available. However, each dataset tries to define slightly different perception challenges and does not offer anything substantially novel. These datasets do not have complete manipulability of the data, a tool which synthetic data (data collected from a source other than the real world - such as video games or simulators) provides. With synthetic data, hard case creations become easy, weather insertion is trivial, and coordinated detections from multiple vehicles can be obtained. Synthetic data opens up possibilities for data which is genuinely hard to create in the real world.

Currently, synthetic data has been used as an inexpensive means to augment real-world datasets. For example, the URSA dataset [3], Driving in the Matrix [37], and Yue et al.

[1]https://waymo.com/open/

[72] have used the video game Grand Theft Auto V (GTA V) to produce synthetic datasets for various autonomous driving tasks. GTA V is selected for synthetic autonomous driving datasets due to the availability of numerous detailed object models, its large world, realistic graphics, and large modding (changing game source-code) community which facilitates the development of data generation tools. Once a synthetic data capturing process is established, annotated data can be generated on a computer with no human supervision at a fraction of the cost required for manual annotation. Furthermore, synthetically generated data opens up possibilities for annotations which are nearly impossible for human annotators to achieve.

In the second chapter of this thesis one of the main contributions is detailed, creating a precise synthetic LiDAR generator within the video game GTA V. Creating a precise LiDAR generator is difficult due to game engines often simplifying models to reduce computational complexity for physics calculations. The synthetic LiDAR generator presented in this thesis captures 3D shapes similar to real-world LiDARs and is accompanied by synchronized images and annotation data. The dataset is named the Precise Synthetic Image and LiDAR (PreSIL) dataset for autonomous vehicle perception. It is shown through experimentation how the PreSIL dataset can be used to boost the performance on state-of-the-art 3D object detection networks for AVs.

PreSIL lays the ground work for obtaining precise synthetic 3D object detection data, introducing a new level of fidelity based on the quality of the game engine. However, as stated above, where synthetic data really proves its worth is in the ability to produce data which is virtually unattainable in the real-world. In the third chapter of this thesis, the PreSIL collection environment is modified to collect a new dataset for cooperative perception. Cooperative perception incorporates information from nearby vehicles by utilizing Vehicle-to-Vehicle (V2V) communication [12]. To create a dataset for cooperative perception, a large portion or all of the vehicles in the collection environment need to have autonomous driving sensor suites, be synchronized and localized in the world space, and then all of the collected data, from all agents, would need to be annotated. This presents an immense challenge in the real world. Synthetic environments provide an omniscient view of the world, enabling the creation of a cooperative perception dataset with little extra effort. In this thesis, ego-vehicle is used to define the main vehicle. The main purposes of cooperative perception are to:

- Extend the range of typical "local" perception networks (networks only using sensor data from the ego-vehicle).
- Detect objects which are occluded from the perspective of the ego-vehicle.

- Reduce uncertainty in local detections and increase perceptual accuracy.

Unfortunately, malicious agents could potentially take advantage of the inter-vehicle information flow to cause harm. It is ill-advised to rely on all actors being benevolent as this leaves the system susceptible to attack. To prevent the consequences of such attacks, trust modelling techniques can be used. Trust modelling is being applied in other areas of V2V communication, but not with a focus on cooperative perception. In this thesis, trust modelling techniques are analyzed by their propensity to be integrated into a cooperative perception system. Finally, the TruPercept dataset is used to experiment with an array of cooperative perception and trust modelling techniques.

To summarize, the main contributions are:

- A novel method to generate precise LiDAR point clouds which accurately represent people (pedestrians and cyclists) using the video game GTA V. A large (50,000+ frames) dataset is provided in the KITTI format with extended information and labels. It is demonstrated that the PreSIL data can boost the performance of a state-of-the-art object detection network on the KITTI 3D Object Detection benchmark challenge [24].

- The first, to the best of our knowledge, multi-vehicle perception dataset for AVs which enables information fusion at any stage of perception. The PreSIL dataset is not solely scenario-based but encompasses regular urban driving in a synthetic world. This will create a testbed for researchers to benchmark models which could drive improvements in cooperative perception.

- An analysis on available trust modelling techniques and their potential usefulness for cooperative perception. Several examples are shown where not using trust modelling could prove to be perilous. Several baseline and novel methods are provided for integrating trust modelling with cooperative perception into an end-to-end distributed perception model. Experiments show up to 5% increase in AP from local perception methods.

Resources

The following resources are applicable to this thesis:

- The PreSIL paper: `https://arxiv.org/abs/1905.00160`

- The PreSIL dataset: https://tinyurl.com/y3tb9sxy
- PreSIL and TruPercept data generation code:
  https://github.com/bradenhurl/DeepGTAV-PreSIL
- TruPercept paper and dataset will be released upon publication at
  https://tinyurl.com/y2nwy52o
- TruPercept code: to be released upon publication at
  https://github.com/bradenhurl/av_trust_and_perception

Chapter 2

Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception

2.1 Contribution

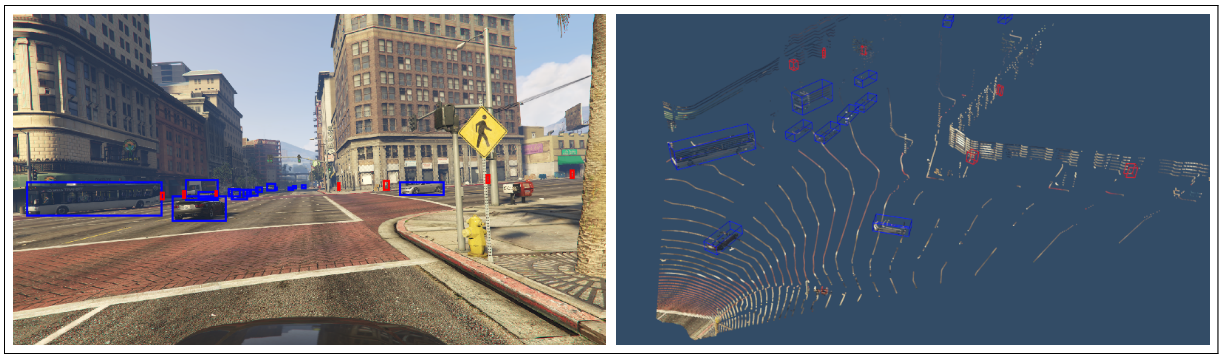

Figure 2.1: Annotated image and point cloud pair from the PreSIL dataset

A sample from the PreSIL dataset is shown in Figure 2.1. The PreSIL dataset consists of both 2D and 3D labels for object detection. Also included is object segmentation information for both 2D images and 3D point clouds. The data generation code can be used to generate a dataset of any size. Publicly available is a large (50,000+ frames) dataset in the KITTI format (detailed in [23]) with extended information and labels obtained from in-game functions. For each data frame, the dataset contains:

- A 1920 x 1080 resolution color image.
- A 1920 x 1080 resolution depth map.
- For each object in the image:
  - A label in the KITTI 3D Object Detection format (2D and 3D bounding boxes, occlusion, truncation, and class information).
  - An augmented label with entity ID, the number of 2D points in the image associated with the object, speed, pitch, roll, and a model name.
- A 1920 x 1080 instance segmentation image for vehicles and people (corresponding to entity ID in label), semantic segmentation (based on GTA V stencil labels e.g. vegetation, sky) for all other pixels.
- An instantaneous point cloud without any motion distortion in a forward-facing 90 degree Field of View (FOV) and similar to the KITTI format but without reflectance values. For each point there is also an associated entity ID if the point corresponds to a vehicle or person.
- Intrinsic and extrinsic calibration information for all sensors.

2.2 Related Work

Many works have already successfully applied synthetic data to various autonomous driving perception tasks. In this section, we will outline related works in various domains of autonomous driving perception.

2.2.1 2D Object Detection

Driving in the Matrix [37] focuses on 2D object detection using GTA V. The 2D bounding boxes from GTA V are produced from 3D bounding boxes transformed to the 2D perspective and are thus loose-fitting. The authors describe a method to capture depth and stencil (class-wise segmentation) buffers from the graphics pipeline which contain detailed depth information and the class of each pixel respectively. They use the stencil buffers to tighten 2D bounding boxes around classes and use the depth information to separate instances from the same stencil class. With this method they are able to obtain tight 2D bounding boxes for over 200,000 images. They show how they have obtained a larger coverage of data than Cityscapes [14]. When only training on their synthetic data, they receive better results on KITTI than when only training on Cityscapes. Martinez et al. [47] also use 2D images from GTA V to detect distance to objects. They generate over 480,000 instances to train a CNN with 2D images for various tasks such as vehicle distance and lane markings. To summarize, 2D data from GTA V has provided notable performance improvements for training detection networks.

2.2.2 3D Point Cloud Segmentation

Yue et al. [72] and SqueezeSeg [69] have leveraged 3D data from GTA V by creating an in-game LiDAR using ray casting. Ray casting can be done in-game through a GTA V native function. The benefit of using GTA V for point cloud segmentation is that each point has an instance level segmentation annotation for any object class (i.e. vehicles, pedestrians, and cyclists), meaning each point can be attributed to the precise object it is a part of. This is a difficult and time-consuming task for a human annotator to achieve as each 3D point needs to be labelled (there can be over a million points per LiDAR scan). In these works, for the KITTI data, individual points are annotated with instance level object information using 3D bounding boxes since the true annotations are unavailable. Augmenting the KITTI data (without reflectance) with GTA V point clouds increased accuracy by about 9 percent IoU. These experiments were only conducted on cars.

Ray casting may seem like a great solution for creating point clouds in GTA V but there are major downsides with the underlying function mechanisms. SqueezeSeg [69] notes that “GTA V uses very simple physical models for pedestrians, often reducing people to cylinders.” Furthermore, some vehicles are simply not hit by ray casting, which could be a major problem for image and LiDAR fusion methods. The PreSIL data generation process addresses both of these limitations.

Fang et al. [20] use deep learning to remove object points from real-world point clouds then insert simulated points from 3D object models. Using this process they are able to augment real-world data with new training samples containing different objects in the same environment without having to collect or annotate more data. They increase accuracy by adding these augmentations to their training dataset. This method does not create images that match, which is an important data element for sensor fusion methods.

2.2.3 3D Scene Understanding

Playing for Data [58] uses GTA V to generate a semantic segmentation dataset comprised of 25,000 video frames. The experiments show that synthetic data significantly increases accuracy when supplementing smaller real-world datasets. Furthermore, it significantly reduces the amount of time and resources needed for obtaining larger amounts of data. Playing for Benchmarks [57] generates 250,000 video frames of semantic segmentation information and includes annotations for optical flow, instance-level segmentation, 2D object detection, and tracking. However, these methods use a technique called detouring, which avoids interacting with GTA V code and only allows coarse semantic segmentation labelling.

The URSA dataset [3] increases the quality of semantic segmentation labels generated from GTA V by labelling each texture which is encountered in the game. This allows unlimited data generation without any extra annotation time. They also increase the size of the released dataset to contain over 1,000,000 images. They show further increases compared to Playing for Benchmarks and demonstrate how more synthetic data can increase network performance.

2.2.4 Simulators

Another way to generate synthetic data is through the use of simulators. SYNTHIA [60] and CARLA [17] use commercial game-engines to create virtual worlds for AV simulations. Simulators are often easier to work with than video games such as GTA V since they are purpose built for autonomous driving development, their source code is often open source, and there is documentation readily available. Unfortunately, realistic worlds take large amounts of resources to develop and currently available simulators fail to reach the same level of realism that is available in commercial video games. CARLA has recently added LiDAR simulator capabilities with the use of ray-casting methods. However, the LiDAR simulated in CARLA is done through ray-casting onto basic collision meshes. CARLA plans to eventually release a LiDAR simulator similar to the one we have produced in GTA V[1] (based on detailed depth buffers instead of ray casting), but it is not yet available.

[1]https://waffle.io/carla-simulator/carla/cards/5b9b8f959dc4a8001bdc5a87

2.3 Synthetic Data

To increase the usability of PreSIL for researchers, the data is formatted to be compatible with the KITTI dataset [24], a widely used dataset for AV perception. New labelling conventions are created whenever it is necessary. When there is an opportunity to create new annotations and sensory data not provided in KITTI, these are also included. Unfortunately, due to the limitations of what is possible (either through functionality or knowledge of the functionality) in GTA V, some data elements need to be approximated. This section details the capturing process, challenges encountered, limitations of the tools provided, and the format of new data.

2.3.1 Capturing Process

Our data capturing process is based on DeepGTAV's approach[2] for creating a scenario, driving autonomously, and retrieving object information. We also use the native code section of GTAVisionExport[3] for collecting depth and stencil information. Both of these repositories use Script Hook V[4] to interact with GTA V.

The camera is placed in the same position as the LiDAR scanner. This is not consistent with a realistic sensor configuration but is necessary when obtaining depth information for LiDAR point clouds, as the depth buffer is generated from the camera perspective. The vehicle used is the "ingot" model; a station wagon which closely resembles the Volkswagon Passat used in the KITTI dataset. The position of the in-game camera and LiDAR is set to match the positioning of the Velodyne LiDAR scanner from the KITTI setup [23] as closely as possible.

Position, dimensions, heading, type, and vehicle model are obtained directly from native in-game functions, which have been documented by the GTA V modding community. 2D bounding boxes can be obtained by transforming the corners of the 3D object to the 2D image plane. This provides loose 2D bounding boxes, which can be tightened using instance-wise semantic segmentation labels. Pixels are segmented in a two-step process. The first step is to project the associated depth pixel into 3D space and calculate if it is situated in a 3D bounding box of the same class as the stencil pixel. If it is not, or it falls into multiple 3D bounding boxes we use a process similar to Driving in the Matrix [37],

[2]https://github.com/aitorzip/DeepGTAV
[3]https://github.com/umautobots/GTAVisionExport
[4]http://www.dev-c.com/gtav/scripthookv/

which uses the stencil buffer to segment object classes then depth disparity to separate instances.

2.3.2 Precise LiDAR point clouds

Previous LiDAR point cloud generation methods in GTA V have used the native ray casting function. As mentioned in section 2.2, this function uses collision meshes which are simplified models (e.g. cylinders for pedestrians). SqueezeSeg [69] has noted this and avoided using pedestrians for this reason. Furthermore, some objects are not hit with the ray casting function beyond approximately 30 metres. User-made documentation notes that ray casting "will not register for far away entities. The range seems to be about 30 metres."[5]

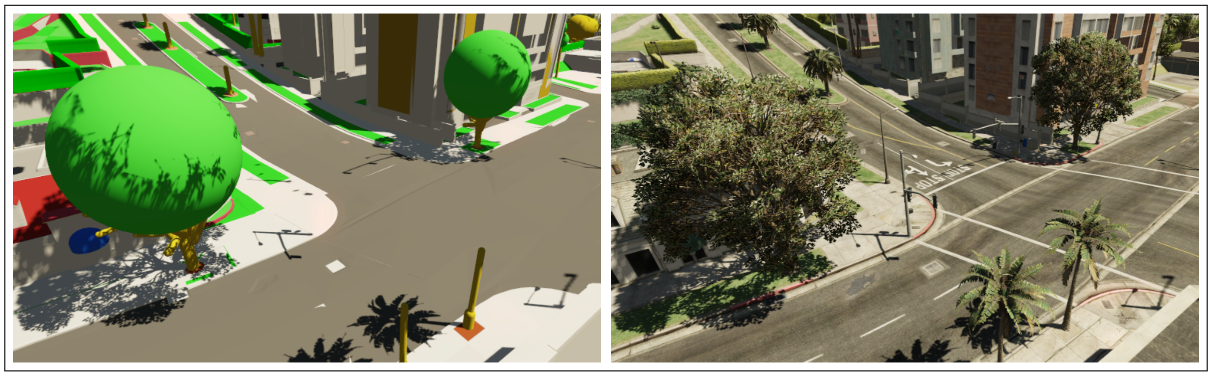

Figure 2.2: Collision meshes simplify physics calculations. A screenshot from GTA V shows the difference between the collision meshes (left) and the final rendering (right).

It is possible to observe the details of collision meshes in Codewalker, a GTA V mod. Figure 2.2 shows that the collision meshes (used for simplifying physics calculations) of some models are not as accurate as the rendered image portrays.[6] Unlike ray casting, the depth buffer is generated using accurate models, which can be used to create a point cloud generation process with precise depth information.

A point cloud, P, is defined as a set of n points, $p_i \in P = (x_i, y_i, z_i) \in \mathbb{R}^3, i = \{1, \ldots, n\}$. The coordinate system from KITTI is used where x is right, y is down, and z is forward (with $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$ the respective unit vectors). An image, I is an array of pixels at locations

[5]http://dev-c.com/nativedb/

[6]https://www.gta5-mods.com/tools/codewalker-gtav-interactive-3d-map

(u, v) with associated RGB values (R_{uv}, G_{uv}, B_{uv}). A depth buffer, D, is an array of pixels at locations (u, v) with associated depth buffer values D_{uv}, where the value at (u, v) corresponds to the same pixel in the image I. The depth buffer, D, is stored in a non-linear format. A transformation function, denoted T_D, is required to go from the depth buffer format to a depth value $d_{uv} = T_D(D_{uv})$. The function $T_D(D_{uv})$ is detailed in section 2.3.3.

An instance segmentation image, S, is an array of pixels at locations (u, v), each with an associated entity ID. Each point p_i is also given an associated entity ID e_i if the point corresponds to an object of interest (e.g. pedestrians, vehicles). A projection transformation function, denoted T_P, is required to go from a vector in $\mathbb{R}^3$ to a value (u, v) corresponding to a point on I, D, and S. T_P uses camera and in-game parameters and is described in detail on page 33, equation 3.2 of *3D Map Estimation from a Single RGB Image* by Racinsky [55]. A transformation using T_P gives decimal u and v values. The resulting point (u, v) is located between four pixels which form a square. If ceil and floor represent rounding up and down to the nearest integers respectively, then the square of four pixels surrounding the point (u, v) are given by (ceil(u),ceil(v)), (floor(u),ceil(v)), (ceil(u),floor(v)), (floor(u),floor(v)). Let $(u_{nearest}, v_{nearest})$ be the closest pixel which is a corner of the square and $(u_{farthest}, v_{farthest})$ be the furthest of these pixels and $GetNear(u, v)$ be a function to obtain these pixels.

To generate a point cloud, unit vectors $\vec{r}_{\alpha\beta}$ are created at various horizontal (α) and vertical (β) angular resolutions to simulate a Velodyne HDL-64E. Each unit vector is projected to the image plane and a depth value is obtained through an interpolation process. This method relies on the LiDAR and camera being located in the same position. Horizontal angular resolution α_r of 0.09 degrees is used when rotating around the y (down) axis of the vehicle. The minimum and maximum horizontal angles are defined as $\alpha_{min} = -45°$ and $\alpha_{max} = 45°$. Vertical angular resolution β_r of 0.42 degrees is used corresponding to a rotation around the x (right) axis. Minimum and maximum vertical angles are defined as $\beta_{min} = -24.9°$ and $\beta_{max} = 2°$, which corresponds to 64 vertical beams. Let d_{max} be the maximum range (120 metres for the Velodyne HDL-64E). Let $R(\alpha) \in SO(3)$ be a rotation of α degrees about the y-axis. Let $R(\beta) \in SO(3)$ be a rotation of β degrees about the x-axis.

Algorithm 1 defines the PreSIL process of generating a point cloud. Without interpolation the point cloud consists of straight lines of points. By performing step 10 it eliminates points with large disparity being interpolated which can cause artifacts (points in between objects). This issue can be seen in Figure 2.3. The ratio of 1.08 was determined to work well through visual inspection of point clouds. A more robust solution could be to only interpolate pixels of the same object instance (only on the pedestrian), or type (e.g. vegetation) using the instance-wise segmentation or semantic segmentation maps.

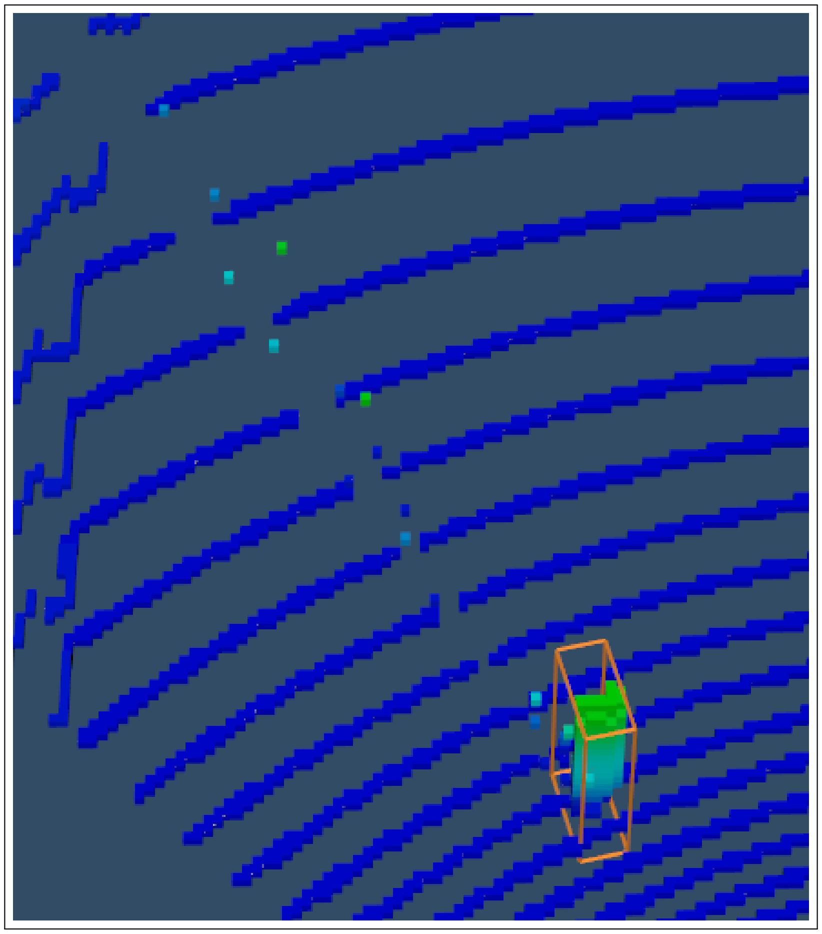

Figure 2.3: A zoomed in LiDAR point cloud where points are coloured based on height from ground. When foreground object pixels are interpolated with background pixels it produces artifacts (the light green/cyan pixels not within the orange pedestrian 3D bounding box). These can be reduced by limiting the difference in depth of pixels which are interpolated together. A more robust solution could be to only interpolate pixels of the same object instance (only on the pedestrian), or type (e.g. vegetation) using the instance-wise segmentation or semantic segmentation maps.

Algorithm 1 Algorithm for generating a point cloud

```
    Input: $\alpha_{min}, \alpha_{max}, \alpha_r, \beta_{min}, \beta_{max}, \beta_r$
    Output: $P$
1:  $i \leftarrow 0$
2:  for $\alpha | \alpha_{min} < \alpha < \alpha_{max}, \alpha \mod \alpha_r = 0$ do
3:      for $\beta | \beta_{min} \leq \beta \leq \beta_{max}, \beta \mod \beta_r = 0$ do
4:          $\vec{r}_{\alpha\beta} \leftarrow R(\alpha)R(\beta)\hat{\mathbf{k}}$
5:          $u, v \leftarrow T_P(\vec{r}_{\alpha\beta})$
6:          $(u_j, v_j) | 1 \leq j \leq 4 \leftarrow GetNear(u, v)$
7:          for $j | 1 \leq j \leq 4$ do
8:              $d_j \leftarrow d_{u_j, v_j}$
9:          end for
10:         if $\max(d_{nearest}, \ldots, d_{farthest}) < 1.08 \cdot \min(d_{nearest}, \ldots, d_{farthest})$ then
11:             $d_r \leftarrow BilinearInterpolation(d_{nearest}, ..., d_{farthest})$
12:         else
13:             $d_r \leftarrow d_{nearest}$
14:         end if
15:         if $d_r < d_{max}$ then
16:             $p \leftarrow \vec{r}_{\alpha\beta} \cdot d_r$
17:             $e_i \leftarrow S_{(u_{nearest}, v_{nearest})}$
18:             $p_i \leftarrow \{p, e_i\}$
19:             $P \leftarrow P \cap p_i$
20:             $i \leftarrow i + 1$
21:         end if
22:     end for
23: end for
24: $n \leftarrow i$
```

This results in precise point clouds which accurately represent all object classes. A comparison of a point cloud generated using ray casting and the PreSIL method is displayed in Figure 2.4. The image focuses on pedestrians and bicycles as these two classes are drastically improved by the PreSIL method. Figure 2.5 demonstrates how a ray casting point cloud may not hit all objects and, captured simultaneously, shows how the PreSIL method hits the vehicles which ray casting missed.

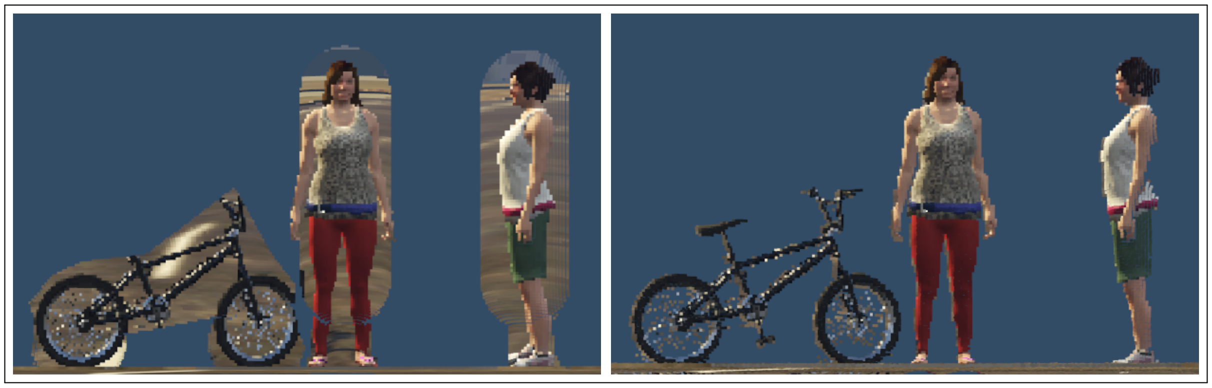

Figure 2.4: Image colors projected onto a point cloud display the difference between ray casting (left) and the PreSIL point cloud generation method (right) with a view of pedestrians and a bicycle. Note: For display purposes a higher LiDAR resolution was used for this graphic.

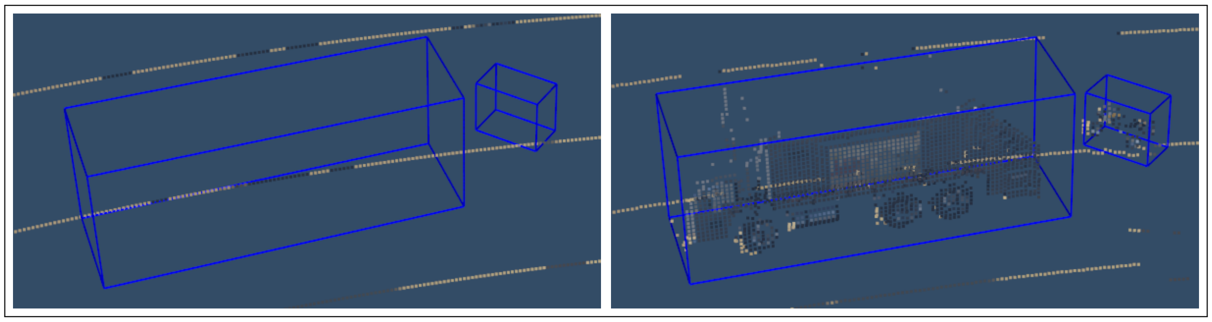

Figure 2.5: Comparison between point clouds generated using ray casting (left) and the PreSIL method (right). Some vehicles past 30 metres are not hit with ray casting but are captured with the PreSIL method.

Gaussian noise is added to each point consistent with the accuracy (2 centimetres) of the Velodyne HDL-64E used for the KITTI dataset [23] in an attempt to mimic the real sensor noise. The Gaussian Noise is added with a standard deviation of 6 millimetres, which has approximately a 99.9 percent chance of being under 2 centimetres. The similarities between the line compositions can be seen in Figure 2.6. In the future, noise could be changed to closer replicate sensor noise. Potential changes could include altering the noise based off of distance or lateral noise instead of only depth. Actual sensor noise should be investigated before implementing these changes.

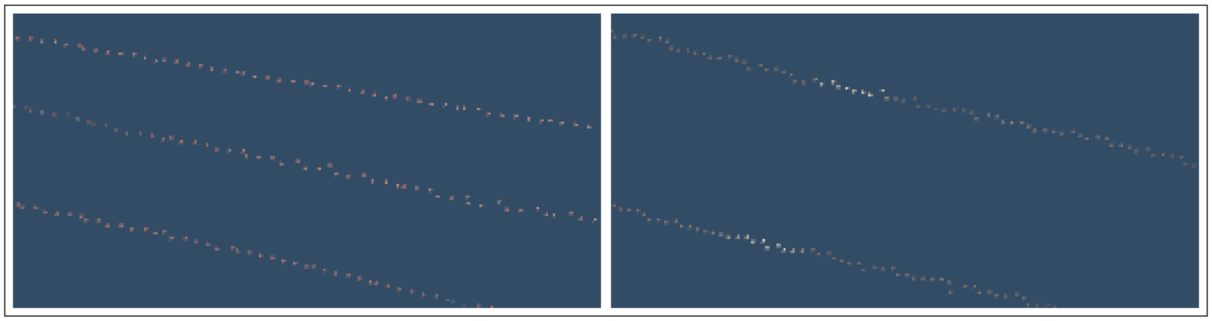

Figure 2.6: Similarity between depth noise in KITTI (left) and PreSIL (right).

2.3.3 Depth Calculation

Using information from Racinsky [55] as a starting point, the function T_D was reverse engineered by examining object points at various distances until points associated with objects were within the 3D bounding boxes. The near and far clipping planes are the planes parallel to the screen before and beyond, respectively, which no object is captured by the in-game camera. The equation for the transformation T_D to obtain a depth value d_{uv} for a pixel (u, v) is

$$d_{uv} = \frac{d_{nc}}{D_{uv} + \dfrac{N_z \cdot d_{nc}}{2 \cdot F_z}} \tag{2.1}$$

where d_{nc} is the distance to the point on the near clipping plane for a pixel (u,v) on the image given by the equation

$$d_{nc} = \sqrt{N_x^2 + N_y^2 + N_z^2} \tag{2.2}$$

The variables N_z and F_z represent the near and far clipping plane distances respectively and are obtained through in-game functions. N_x and N_y are the horizontal and vertical near clipping plane distances, respectively, from the center of the clipping plane to the point (u, v) on the image. These are defined as

$$N_x = \left| \frac{2 \cdot u}{N_W - 1} - 1 \right| \frac{N_W}{2} \tag{2.3}$$

$$N_y = \left| \frac{2 \cdot v}{N_H - 1} - 1 \right| \frac{N_H}{2} \tag{2.4}$$

The near clipping plane width and height are denoted by N_W and N_H respectively. They are calculated by using the screen aspect ratio (denoted as ζ), near clipping plane distance, and the vertical FoV (ψ_V) (all of which can be obtained by in-game functions). These values remain the same unless graphics settings are changed.

$$N_H = 2N_z \tan\left(\frac{\psi_V \frac{180}{\pi}}{2}\right) \tag{2.5}$$

$$N_W = \zeta N_H \tag{2.6}$$

A 3D view of the clipping planes is shown in Figure 2.7. A top-down view labelled with variables is shown in Figure 2.8 to illustrate the concepts. A side-view would show similar parameters for height and the y-axis.

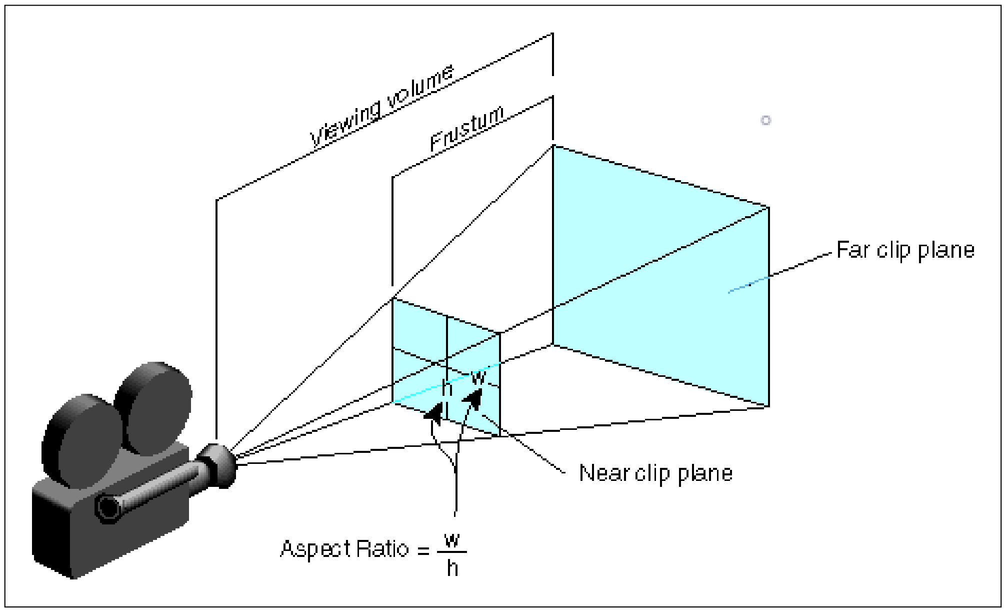

Figure 2.7: A 3D view of the clipping planes and viewing frustum [27].

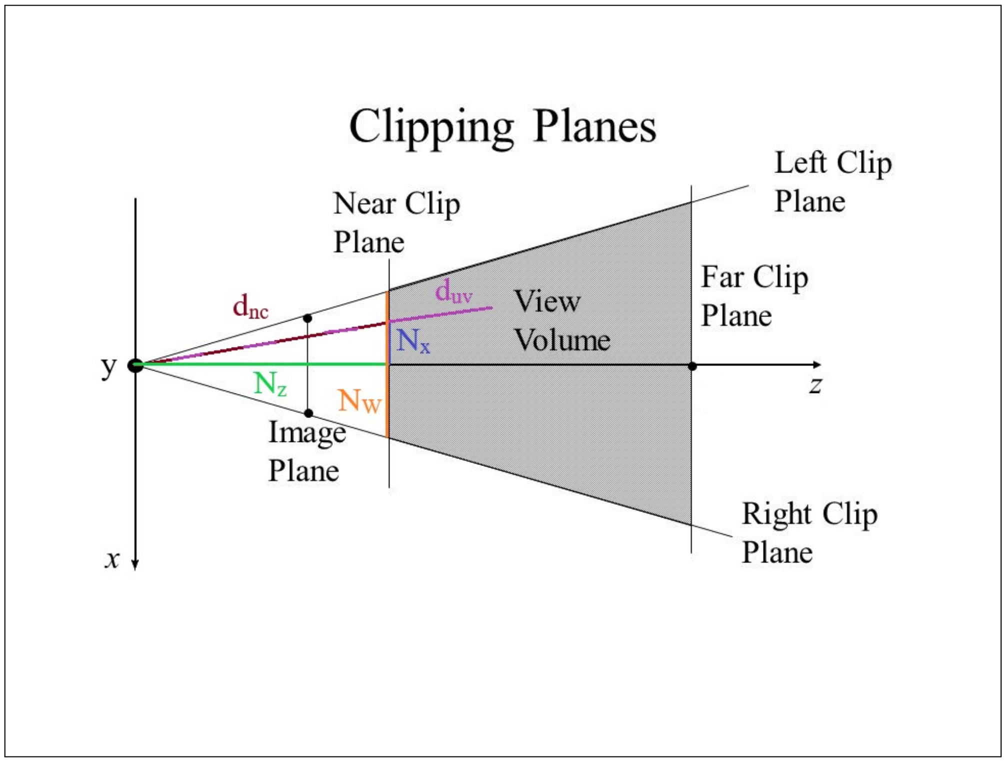

Figure 2.8: A top-down view of the clipping planes and variables used for the depth buffer transformation [75]. Best viewed in colour.

2.3.4 Synchronization

Our data collection process uses both native functions and DirectX buffers to collect data. Synchronization between the data sources proved to be a major challenge. Since the source code is not available, it is impossible to know exactly how the game engine functions. The depth and stencil buffers are obtained from the DirectX buffers, whereas native functions are used to obtain information about the object and to perform ray casting. When the buffers arrive from DirectX, a callback function can be used to time the collection of all other data. However, the native functions are at a different time-step than the buffers when the callback is signaled. This problem can be viewed by projecting image data onto a high-resolution point cloud. If aligned, the bounding box should perfectly encompass the object and point colours should correspond to their associated image colour. It is easiest to visually inspect if objects (especially fast-moving objects) of a colour different from the background are aligned due to high contrast and larger misalignment. The DirectX buffers from one time-step after using the native functions usually align with the annotation data sources; however, there are some instances where this method does not produce aligned data. The method of synchronization used for PreSIL, which always results in all data sources being aligned, uses native functions to pause the game, flush the graphics buffers, collect synchronized data, then resume the game. This procedure is completed in the following steps:

1. Pause game
2. Two calls to render the script cameras
3. Collect data (from native functions and DirectX)
4. Resume the game

This procedure has been tested on a Graphics Processing Unit (GPU) from both Nvidia and AMD and has been visually inspected to ensure point clouds align. The visual inspection was completed throughout a predetermined route during which the vehicle regularly maneuvers around corners. Synchronization and alignment problems are easily seen when the camera is rapidly rotating as it creates greater inter-frame disparity.

2.3.5 Limitations

GTA V does not contain reflectivity information, a value which typically accompanies each LiDAR point and is relative to the reflectance of the material hit as well as the incidence angle of the laser pulse. This value is commonly used for detecting objects but is omitted from the PreSIL point clouds. The reflectivity value also determines the range at which a LiDAR scanner can detect a surface. If there is low reflectivity, the detectable range of a surface will be lower. This representation of reflectivity was not included in the PreSIL point cloud generation method. In the future, it could potentially be incorporated by labelling reflectivity of texture values in a similar manner to the URSA dataset [3] texture labelling approach used for semantic segmentation. Another potential solution could be to train a generative network on real-world reflectivity data (such as KITTI) so that the network could generate reflectivity information for synthetic point clouds.

A second challenge lies in the continuous data collection nature of true rotating LiDAR systems. The Velodyne HDL-64E contains a row of lasers and rotates to allow the lasers to capture data in a periodic 360 degree sweep around the vehicle. During KITTI data collection, the scanner is rotating at 10 Hertz. This affects the point cloud in that there can be up to 100 ms difference between point capture times in the same point cloud. This is a difficult behaviour to replicate in synthetic data as there is not enough control over the passage of time in-game. Interestingly, however, LiDAR scans in the KITTI dataset are motion corrected for the movement of the ego-vehicle, removing much of this effect based on GNSS/INS motion estimates, although dynamic objects continue to exhibit distortion in the corrected scans. For the PreSIL synthetic LiDAR scanner, all points are captured at the same instant in time, which is an idealization that is not possible in rotating LiDAR data collection in dynamic environments.

Another limitation of PreSIL is that the LiDAR scanner only captures points which are in the field of view of the camera (45 degrees to either side of forward). Future work will attempt to build a 360 degree LiDAR generator using multiple cameras with different viewing perspectives.

Velodyne HDL-64E point clouds can contain other sources of noise around edges, from high incidence angles, from varied reflectivity of surfaces, and from rays hitting the edge of the internal mirrors [26]. One effect of noise is that some points within the maximum range are not captured. SqueezeSeg [69] projects points onto a sphere to show that KITTI point clouds contain this type of noise. The PreSIL synthetic LiDAR does not attempt to approximate these types of noise.

2.3.6 Extensions

While synthetic data from GTA V has some limitations, it also has a few advantages. Instead of reflectivity values for each point in the point cloud, PreSIL contains new pieces of information. First, an entity ID is provided if the point lies on an object, or the stencil class if it does not. The elevation changes also lead to further label parameters being created. As can be seen in Figure 2.9, pitch can be necessary to complete the object information otherwise the bounding box may not encompass the entire object. Roll is also included to complement pitch.

The extended annotations are contained in a separate file to keep labels compliant with the KITTI format. For each object, the augmented annotations also include:

- Entity ID (to relate entity ID of points to a detection box)
- Count of pixels which are part of the object's 2D instance-based segmentation mask
- Speed

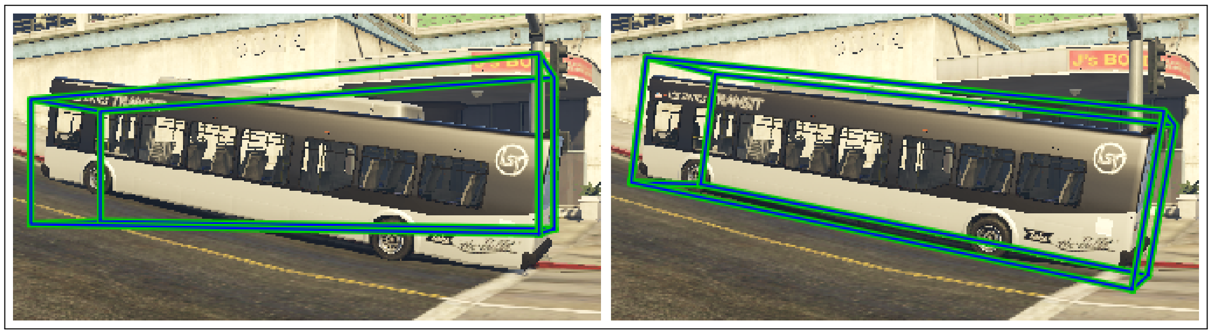
Figure 2.9: On steep slopes vehicles appear out of their ground truth 3D bounding boxes using the KITTI data format (left), but the PreSIL augmented annotations correct this problem (right).

- The object's model string identifier

Model strings could enable interesting experiments by using vehicle models as classes instead of broader vehicle classes such as cars and trucks. Images are captured with a resolution of 1920 x 1080 pixels. Full resolution depth, stencil (class-wise segmentation), and instance-level segmentation images have also been included. These additional files are useful for semantic segmentation and depth completion tasks. Specific formatting for these files can be found in the data format document included with the dataset download.

2.3.7 Dataset Statistics

The count for each object class and the average count per image which contains the object class are outlined in Table 2.1. The PreSIL dataset has an abundance of pedestrians and vehicles but fewer cyclists (due to the game spawn rate) compared to KITTI. The PreSIL dataset also includes important classes such as Bus, Motorbike, and Trailer, which are not included in the KITTI dataset. Figure 2.10 shows a bird's eye view (BEV) heatmap comparison of the KITTI and PreSIL datasets for the pedestrian class. As can be seen, the PreSIL dataset has a larger coverage area than the KITTI dataset. A large part of the discrepancy is likely due to GTA V having wider roads than the KITTI collection area. Pedestrians are encountered much further to the sides and there is a gap in the middle (the roadway) where there are fewer pedestrians in the PreSIL dataset.

We also added the capability to create user configurable scenes. User configurable scenes can be used to test the coverage of a model and increase the robustness by training

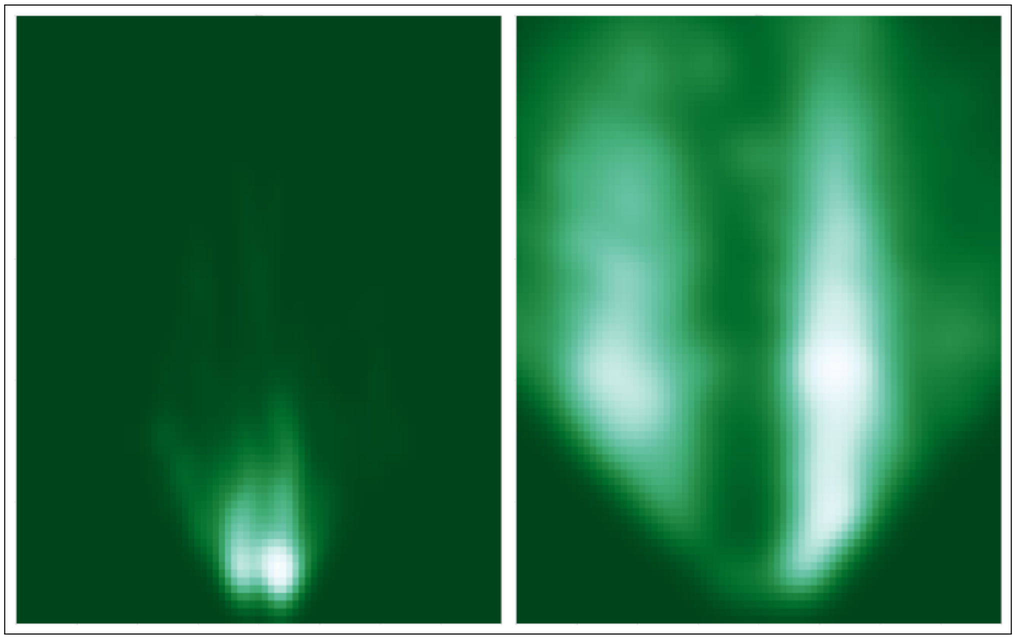

Figure 2.10: Birds eye view (BEV) Heatmaps of KITTI (left) and PreSIL (right) showing coverage of pedestrian locations. Heatmaps extend 40 metres to the left and right, and 100 metres forward.

Table 2.1: Comparison of class statistics for the PreSIL and KITTI datasets where Average Per Image Containing Class (APICC) is provided to compare class densities of images which have at least one instance of the class.

Dataset	Type	KITTI	PreSIL	KITTI	PreSIL
		Count		APICC	
Both	Car	28,742	334,066	4.30	7.27
	Pedestrian	4,487	147,077	2.52	6.17
	Truck	1,094	105,316	1.06	2.90
	Person_sitting	222	6,511	2.24	2.09
	Cyclist	1,627	516	1.43	1.23
PreSIL	Motorbike	NA	15,022	NA	1.37
	Bus	NA	7,792	NA	1.14
	Trailer	NA	4,530	NA	1.20
	Railed	NA	1,226	NA	3.04
	Boat	NA	606	NA	1.02
	Airplane	NA	255	NA	1.00
	Animal	NA	21	NA	1.00
KITTI	DontCare	11,295	NA	2.11	NA
	Van	2,914	NA	1.36	NA
	Misc	973	NA	1.25	NA
	Tram	511	NA	1.46	NA

on the data which reveals holes in the coverage. Examples of this can be seen from Yue et al. [72]. Future work may use these scenes to improve coverage of pedestrians or cyclists.

2.3.8 Dataset Samples

Different data from a single frame is presented as a sample. An annotated image in Figure 2.11, an annotated point cloud in Figure 2.12, linearized depth image in Figure 2.13, semantic segmentation of objects in Figure 2.14, and a visualization of the raw stencil buffer in Figure 2.15.

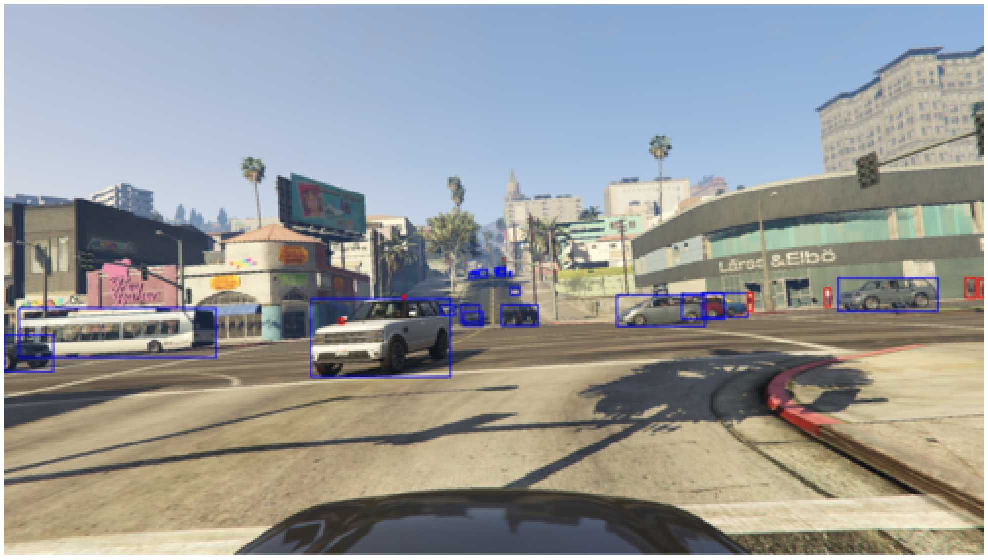

Figure 2.11: Sample image: annotations are blue for vehicles and red for pedestrians.

2.4 Experiments and Results

Experiments were constructed for two primary purposes:

1. To prove the efficacy of the PreSIL synthetic data capturing process at improving accuracy for autonomous driving perception benchmarks.
2. To understand the impact of pre-training with synthetic data on state-of-the-art pedestrian detection algorithms for autonomous driving.

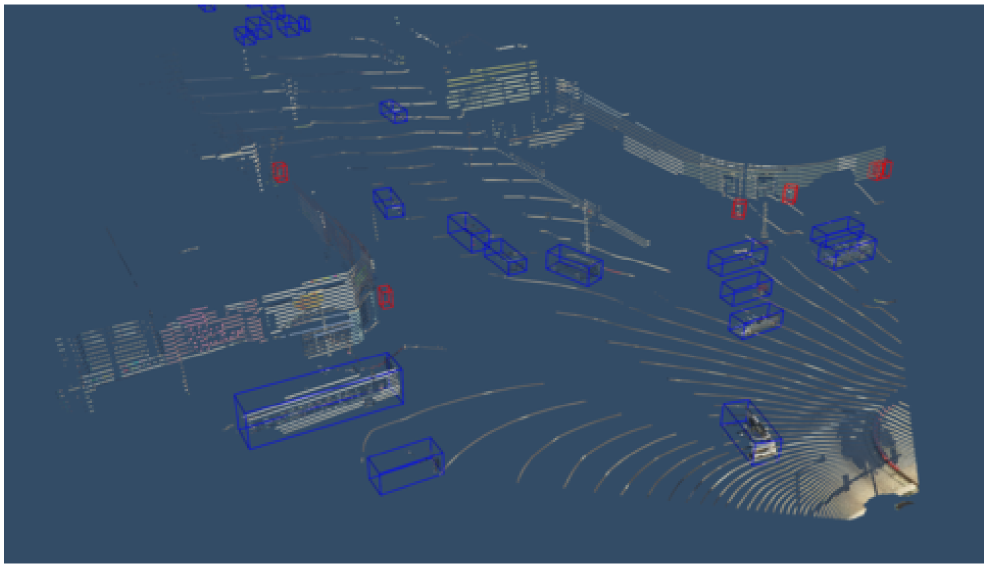
Figure 2.12: Sample point cloud: annotations are blue for vehicles and red for pedestrians.

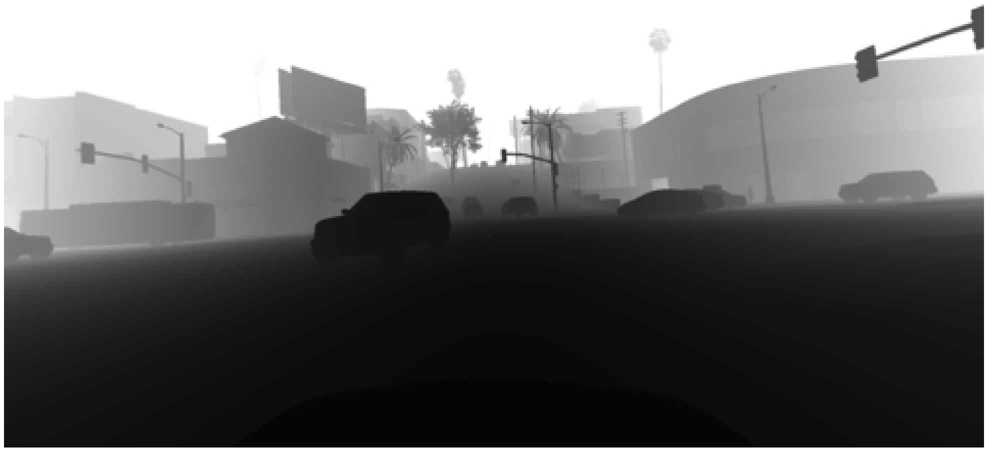
Figure 2.13: Sample linearized depth map

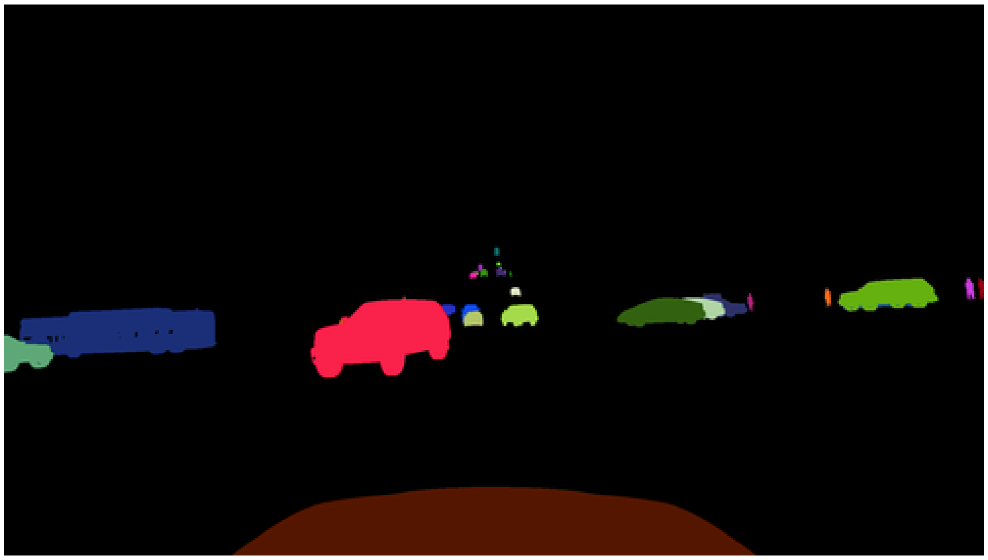
Figure 2.14: Sample object segmentation image

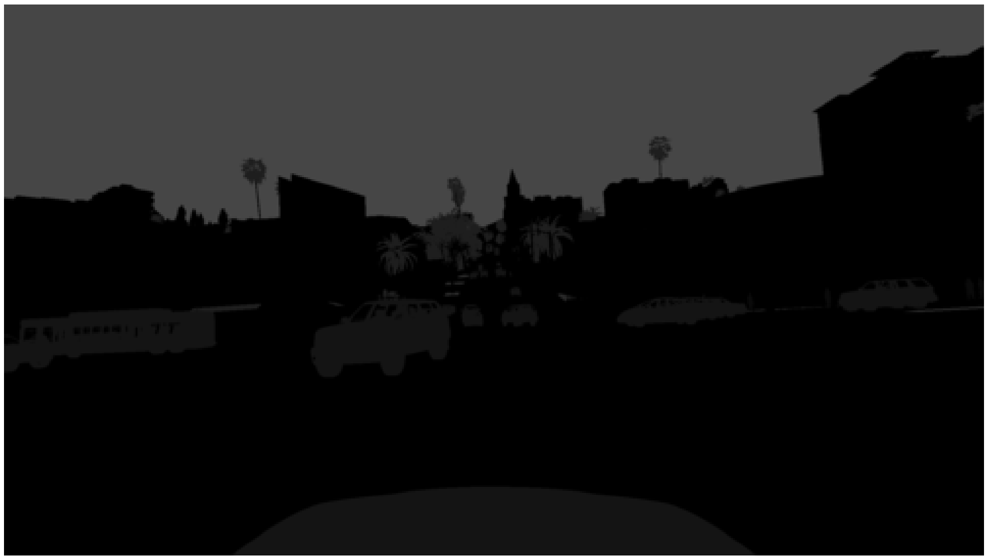
Figure 2.15: Sample stencil image

3D Object Detection was used for evaluation as it is a primary task for autonomous driving perception and, to the best of our knowledge, results for 3D object detection for autonomous driving with synthetic and real data have never been published. We chose to use the AVOD [42] architecture with the feature pyramid network (AVOD-FPN) for testing. AVOD-FPN is a top-10 performer on the KITTI 3D object detection benchmark challenge [24] under the pedestrian category, has publicly available code, and runs in real-time (0.1 second runtime). Pedestrians are the focus of the evaluation as there is a smaller proportion of pedestrians in the KITTI dataset and the PreSIL novel LiDAR generator drastically improves the realism of pedestrians. Cyclists are not included in the experiments as there is a much smaller cyclist count in PreSIL than in the KITTI dataset. If desired, additional scenarios could be constructed where cyclists are spawned in abundance.

2.4.1 Evaluation Metrics

The trained 3D object detection network is evaluated using the standard KITTI metrics for 3D object detection. This is defined as average precision, which is simply the precision averaged over different levels of recall. The KITTI 3D object detection challenge is divided into three categories (easy, moderate, and hard) based on 2D bounding box height, occlusion, and truncation levels.

2.4.2 Experimental Setup

To confirm that the PreSIL data improves the performance of state-of-the-art object detectors, we compare the results of training and evaluating AVOD-FPN using different combinations of synthetic and real training data. The KITTI 3D Object Detection dataset is used for both training and evaluation. The KITTI 3D Object Detection training dataset is split approximately 50/50 to create which will be referred to as the KITTI training/testing splits. These splits resulted in 3,712 train instances and 3,769 test instances. Validation will not be used when training on KITTI but instead the results will be taken from the best evaluated checkpoint for each training configuration. The PreSIL data is split into an 80/20 training/validation split resulting in approximately 40,000 and 10,000 instances, respectively. A subset is created of 10,000 instances from the 40,000 instance training split for use in a different training configuration.

As a baseline, AVOD-FPN is trained using only the KITTI training split. For the next set of training configurations a subset of 10,000 or the full 40,000 instances of PreSIL is used for training. First, AVOD-FPN is trained on the PreSIL train subset, then evaluated with the PreSIL validation split. Next, fine-tuning on the KITTI training split is completed by training from the highest performing checkpoint from the PreSIL training. Finally, all checkpoints are evaluated using the KITTI testing split. The results shown are from the single best-performing checkpoint of each configuration.

As a secondary experiment AVOD-FPN is trained with subsets of the real data to show the effectiveness of synthetic data in the presence of less real data. Training configurations with half (approximately 2,000 instances) and no real-world training data are completed.

2.4.3 Experimental Results

Results are displayed in Table 2.2. The notation 'x-y' is used to specify the number of PreSIL and KITTI data instances (in thousands) used to train AVOD-FPN. For example, 40-4k signifies 40,000 PreSIL instances and 4,000 KITTI instances were used for training.

Table 2.2: Experimental Results: Average Precision on the Pedestrian class of the KITTI 3D Object Detection Benchmark.

Data	Moderate	Easy	Hard
0-4k	51.31 %	57.40 %	44.70 %
10-4k	51.45 %	57.57 %	47.72 %
40-4k	**55.57** %	**59.47** %	**49.20** %
40-2k	48.04 %	54.14 %	44.62 %
40-0k	11.85 %	14.13 %	11.62 %

The results are promising as there is a significant improvement (1-5% AP) at all difficulties when pretraining with 40,000 PreSIL instances. An interesting observation is that the improvement is largest for the hard difficulty. This could be due to the disparity between the distributions of pedestrian locations in the PreSIL and KITTI datasets. Shown in Figure 2.10, PreSIL has a larger average distance than KITTI and therefore a higher probability of training data better representing the hard difficulty level. The results in Table 2.2 are significantly ($\approx$ 10%) higher than on the KITTI benchmark results page since the networks are evaluated on a test split of the training data. The training data with labels provided by KITTI is likely closer in distribution within the training set than to the testing set, and therefore results in better performance.

The car class was also compared using the same setup as with pedestrians, results are shown in Table 2.3. An almost 7% gain was seen with the moderate class whereas the easy and hard difficulty levels remained almost identical. It is unknown why this discrepancy in training occurred. Difficulty level composition remained similar throughout all training, validation, and test splits. However, the distribution of the difficulty levels could be skewed from differences in KITTI labeling of occlusion and truncation levels compared to the PreSIL generation process.

Table 2.3: Experimental Results: Average Precision on the Car class of the KITTI 3D Object Detection Benchmark.

Data	Moderate	Easy	Hard
0-4k	78.56 %	74.54 %	67.90 %
40-4k	**85.39** %	74.95 %	67.84 %

No comparison to the impact of other synthetic datasets on network performance is possible at this time due to their current lack of availability. An area of future for work could be to collect data from a simulation environment such as CARLA for comparing performance with the PreSIL dataset.

2.5 Conclusions

We have created a synthetic dataset with standard annotations for autonomous driving perception tasks at a fraction of the cost of a real-world dataset. We have given a qualitative analysis on why the PreSIL data collection process is superior to any other synthetic data collection methods. We have also added new annotations. The new annotations create exciting opportunities for new ways to train perception networks for various autonomous driving tasks. Future work will include adding a tracking dataset as well as 360 degree coverage of images and LiDAR. We have demonstrated how synthetic data can be used to improve state-of-the-art 3D object detection performance and have provided a quantitative analysis on why some difficulty levels show larger improvements. The PreSIL dataset and code is publicly available[7] and can be used by anyone to improve the performance of their perception algorithms. Future work could also include transfer learning and methods for adapting data from one distribution to another to improve training performance. These results only scratch the surface of how synthetic data can be used to achieve more robust autonomous driving perception networks.

[7]https://tinyurl.com/y3tb9sxy

2.5.1 Limitations

GTA V was chosen as the synthetic data collection environment due to its large world and commercial grade graphics which are highly funded compared to open source simulators. However, working with GTA V comes with many disadvantages. Source code is not available and in-game functions need to be accessed through Script Hook V[8]. These functions are used by accessing hash values, some of which have been documented by the modding community. Function parameters (even types), are sometimes unknown and documentation is user-made through experimentation and examination of binary files. This results in poor documentation for many functions. Furthermore, unexpected behaviours can be encountered, such as vehicles sporadically disappearing or not being hit by in-game raycasting functions. These problems were difficult to circumvent and made development tedious. It is therefore recommended to not invest vast amounts of time in GTA V, especially for scenario creation, as controlling entity behaviour is not straightforward. Another major limitation to this work is that it likely cannot be used commercially due to copyright restrictions, meaning it is only useful for research purposes.

2.5.2 Future Work

As more real-world datasets are becoming available (ApolloScape [32], nuScenes [9], Argoverse [10], Waymo Open Dataset[9]), the usefulness of synthetic data for training 3D object detection networks may be diminished. Feng et al. [21] provide a summary of datasets for AV object detection and segmentation tasks. Other interesting extensions to this work include investigating different domain adaptation techniques. Pre-training was used to validate the PreSIL dataset; it would be interesting to investigate the effects of other methods such as training using a fraction of the training data as frames from the PreSIL dataset and the remaining frames from real-world data. SG-GAN [45] and Savkin et al. [61] use GANs to increase the similarity of synthetic images from GTA V to real-world images for the semantic segmentation task. They show large performance gains when only training with synthetic data. Other work could include using different combinations of domain adaptation methods to improve performance on real-world tasks. Training simultaneously with synthetic data and real data is a promising direction for domain adaptation. This can be done by creating different ratios of synthetic data to real data in the training set. This would remove the need to pre-train and fine-tune.

[8]http://www.dev-c.com/gtav/scripthookv/

[9]https://waymo.com/open/

One downside of GTA V point clouds is that there is no reflectance value available. Typical LiDAR scanners produce x, y, and z positional values as well as a reflectance measurement. Yue et al. [72] elected to simply use depth information without a reflectance value for their synthetic LiDAR generator. When only training with the KITTI dataset they show a decrease in performance when omitting the reflectance values, a drop of approximately 8 percent Intersection over Union (IoU) in their experiments. This signifies that reflectance values play a significant role in point cloud segmentation, so it is likely also important for detection. Future work could attempt to generate reflectance values for PreSIL point clouds using a GAN and in-game properties.

Chapter 3

TruPercept: Synthetic Data and Trust Modelling for Autonomous Vehicle Cooperative Perception

This chapter proposes an approach for AVs to communicate perceptual observations, tempered by trust modelling of peers providing reports. Also presented is a new synthetic dataset which can be used to test cooperative perception. The TruPercept dataset includes unreliable and malicious behaviour scenarios to experiment with some challenges cooperative perception introduces. The TruPercept runtime and evaluation framework allows modular component replacement to facilitate ablation studies as well as the creation of new trust scenarios for experimentation. The dataset, including all trust scenarios, as well as all tools, evaluation, and model code are publicly available at `https://tinyurl.com/y2nwy52o`.

3.1 Introduction

As AVs become increasingly prevalent on public roadways, the need for communication between vehicles, V2V communication, increases. Vehicular Ad Hoc Networks (VANETs) are spontaneous networks utilized for intelligent vehicle communication. VANETs can be comprised solely of Intelligent Vehicles (IVs), however also commonly include Road Side Units (RSUs) to extend communication range and a central server for tasks such as moderating agents or information aggregation and evaluation. VANETs can be utilized to transfer information such as traffic congestion, road conditions, and even collision warnings.

Cooperative Perception, sometimes referred to as collaborative perception or collective perception, is a perception domain which focuses on fusing information from multiple agents. Cooperative perception is grounded in the idea that two or more agents can combine information from their respective viewpoints to increase perceptual range and accuracy and reduce blind spots caused by occlusions from one perspective.

Most state-of-the-art cooperative perception methods assume information received is correct. In real-world situations this assumption does not hold, network failures may results in false detections or malicious attackers could take advantage of models which do not incorporate mechanisms to detect and eliminate false information. Trust modelling is a field of study which aims to identify untrustworthy agents. Trust modelling has been studied for VANETs but never, to the best of our knowledge, with a focus on cooperative perception.

This chapter will address this topic and propose a novel system model in an attempt to improve perceptual accuracy through the use of VANETs and trust modelling. Cooperative perception aims to have three main benefits:

1. Extend perceptual range: data can be shared which goes beyond the range of the ego-vehicle sensors. Longer ranges leads to longer warning times for hazardous situations.

2. Perceive occluded objects: different vehicles may have a clear view of locations that are occluded from the ego-vehicle. A typical example is when trying to pass on a 2-way highway, the leading vehicle can provide the following vehicle with a complete view past the leading view. Occluded and truncated objects are a major factor in densely populated urban environments. Pedestrians may walk between vehicles when crossing traffic or vehicles running red lights may appear suddenly from behind a corner. These situations can cause accidents that humans will have little warning to avoid. However, with communication between vehicles, objects which were previously obstructed from view can become known. By adding objects out of view it might not increase detection accuracy (as these objects are not typically scored) but it will surely increase safety.

3. Increase perceptual accuracy: cooperative perception has the potential to also increase accuracy over local perception methods. Higher perceptual accuracy increases safety by making it less likely an object is not detected.

The motivation behind using cooperative perception to increase perceptual accuracy, even in situations where the object is at least partially visible to the ego-vehicle, arises from two main ideas:

(i) *Ensemble Methods*: Ensemble methods in machine learning are ways in which multiple classifiers can have their predictions aggregated together to produce the final output prediction [16]. Ensembles often produce better results than a single classifier. By aggregating the detections from multiple vehicles, the final predictions act similar to that of ensemble methods. The classifiers may or may not be identical (depending on the software in the vehicle), but the incoming data is different. A hypothesis is introduced that using detections from multiple vehicles could improve detection accuracy through similar principles of ensemble learning.

(ii) *Viewpoint Diversity*: Different viewpoints may increase detection accuracy of objects in view of the ego-vehicle as a closer, less obstructed, or non-truncated view of an object can increase the probability of a true positive detection. Some vehicles may also include sensor modalities that others are missing (like infrared camera or radar).

It is from these two main points that the topic in this chapter is motivated. The goal is to increase object detection accuracy, even of objects that are beyond the typical sensor range, which will in turn increase road user safety. The solution depends on parties being willing to share information and must not degrade performance (i.e., the solution must run in real-time).

Existing work in this area is limited, likely due to the lack of testing data. Current state-of-the-art cooperative perception methods focus on limited test scenarios. This is problematic as the effect these models have on the entire perceptual pipeline cannot be analyzed. Furthermore, the methods which have been published typically do not even address the possibility of false or malicious information being disseminated. By introducing cooperative perception without conducting a thorough analysis on the potential implications of malicious agents, autonomous vehicles could become significantly more dangerous. Perception is a critical component of autonomous vehicles and failures could result in serious injury or loss of life.

In an attempt to push forward the capabilities and safety of cooperative perception, this thesis chapter offers three primary contributions:

- The TruPercept dataset, which is, to the best of my knowledge, the first multi-vehicle perspective perception dataset for AVs gathered in a realistic environment. Only limited scenarios were explored in the past, the TruPercept dataset is encompasses regular urban driving in a synthetic world over multiple routes. The TruPercept data collection program also allows easy and inexpensive collection of additional data within the GTA V world. This will allow researchers to benchmark models for improving cooperative perception.

- An analysis on available trust modelling techniques and the capacity in which they can be integrated into a cooperative perception system model.
- Several baseline and novel methods for integrating trust modelling with cooperative perception into an end-to-end distributed perception model. The best model increases performance by up to 5% on the TruPercept dataset.

These building blocks for effectively understanding trust modelling with perceptual information are essential for moving forward with safe cooperative perception models.

3.2 Related Work

Existing methods for cooperative vehicles utilize various information fusion approaches, VANETs, and trust modelling techniques. A background study on the state-of-the-art will be introduced in this section.

3.2.1 VANETs

VANETs can be used to share different types of information, some examples are traffic congestion information and road condition warnings. Cooperative perception systems will also use VANETs to share perceptual information between vehicles.

Xu et al. [70] analyze network capabilities and protocols to ensure messages can be reliably received between vehicles with low latency (< 100 milliseconds) and an acceptable range (300 metres) for applications which are used for safety. This makes it possible for vehicles to transfer critical messages in a timely manner. Communication between vehicles can make driving maneuvers safer and more efficient through cooperative maneuvers. Correa et al. [15] show how smart infrastructure can support these maneuvers. They also discuss how RSUs can enhance perceptual range by using standardized messaging systems. Riedl et al. [59] estimate the network coverage for VANETs to rate the availability of the network for IVs and to potentially allow RSUs to be installed in optimized locations.

Since VANETs are spontaneous and vehicles communicate directly between each other, there exists a possibility that a vehicle disseminates false information. Vehicles may act for selfish reasons, such as spreading information that routes on their path are experiencing traffic congestion in an attempt to reduce their own travel time. Vehicles could also attempt to change their identity. To prevent this possibility a central server or authority can be included in VANET system models. The role of the central server or authority in many VANET systems ([13], [46]) is to provide a trust certificate to help vehicles identify which information is trustworthy. Many trust models include information being relayed through vehicles to RSUs then onto the central server. The central server can then determine the accuracy of information if it receives delayed ground truth information, such as authority-provided collision information. This information can be used by the server to calculate the trustworthiness value of each vehicle. A trustworthiness certificate can then be compiled with the vehicle ID and trustworthiness value. In some system models ([13], [46]) these certificates are attached to each message. However, in other systems ([11], [48]), there is a lack of central authority or an inability for the central authority to obtain ground truth information. The trust modelling component of the system must deal with this information or lack thereof through various techniques.

3.2.2 Cooperative Perception

In this section, a background study on cooperative perception is provided. Perception methods which use only the ego-vehicle viewpoint will be referred to as local perception.

Feng et al. show, in their summary of modern object detection and segmentation networks [21], methods which utilize information fusion at various stages in the detection network. Typical fusion methods in 3D object detection fuse image and LiDAR data. Cooperative perception similarly has to fuse information from multiple sources except the number of sources can be much greater as information sources are nearby vehicles. With cooperative perception the information shared can range from raw image and point cloud data to final object detections or tracks (detections over time).

Several researchers have done work on the latency bounds required for cooperative perception. ReLaDec: Reliable Latency Decision Algorithm for Connected Vehicle Applications [66] proposes an algorithm to determine whether incoming information is within the latency requirement for specific applications. Allig et al. [2] propose a new method for aligning detections shared between vehicles for cooperative perception.

Kim et al. [40] use map merging techniques to align all views together. They suggest that sensors are only needed on the fronts of vehicles and all other information can be shared between vehicles. They also provide several techniques for see-through visualizations which could allow human drivers to visualize blind spots caused by leading vehicles when trying to pass. They test their methods on several pre-defined scenarios with up to four vehicles in real-world situations.

Protocols, latency, and alignment issues are important concepts when dealing with cooperative perception but are beyond the scope of this thesis. For the rest of this chapter the assumption will be that protocol, latency, and alignment problems are solved.

Raw Sensor Data Fusion

Arnold et al. [4] use images from multiple perspectives to reconstruct 3D objects which are commonly found in autonomous driving perception scenes. They show how it could be possible to use images from multiple vehicles to better detect relevant 3D objects.

Chen et al. [12] fuse raw point cloud data to improve 3D object detection accuracy for AV perception. They argue that using point clouds is better than images since there is no need for overlap to perform convolutions. Point clouds can simply be concatenated and 3D LiDAR detection networks run normally. Chen et al. define a model for cooperative perception but only test it by concatenating point clouds from the same vehicle of a KITTI scene at two different times. The authors mention their testing process is due to a lack of LiDAR datasets, especially for cooperative perception. However, they are able to show that the detection of stationary vehicles is improved and the range of detections is extended. The authors also provide their own dataset, which is limited to only 16-beam LiDAR sensors, golf-cart test vehicles, and is only collected in parking lots.

The benefits of fusing raw sensor data is that more information is provided and that networks can fully take advantage of that information in their detection process. The disadvantage is that more information has to pass between vehicles than in late fusion, increasing the bandwidth requirement and potentially the latency of information dissemination. Kim et al. [39] test the latency of transferring raw, compressed, and processed perceptual information. They use the IEEE 802.11N standard for information transfer and show an average latency of 854 milliseconds for the raw data, 37 milliseconds for compressed data, and 17 milliseconds for the processed data. They used a 640 x 360 resolution camera and compression time took 8 milliseconds on average. They also include results with LiDAR data, however, only for a system with 721 points per point cloud (the Velodyne HDL 64-E can produce upwards of a million points). These methods also assume that shared raw data is compatible with the ego-vehicle raw data, which may not be the case.

Late Fusion

Correa et al. [15] say perception performance can be enhanced by transporting detection results (late fusion) instead of raw data. They point to two standardized message transportation protocols; the Cooperative Awareness Message (CAM) [18] and the Collective Perception Message (CPM) [19] (which is currently being developed). Hobert et al. [31] discuss some of the limitations and point to future recommendations for the use of CAMs [18]. These messaging systems can provide a means for standardized communication of late fusion for cooperative perception.

Obst et al. [52] combine tracks from multiple vehicles to attempt to increase detection of vehicles which are out of sensor range. This can facilitate detecting and tracking vehicles which enter into view.

There are multiple benefits of fusing information at the detection or tracking level:

- The amount of data which is shared is much smaller.
- Local perception can occur immediately after sensor data is procured. This allows for faster perception results instead of waiting to receive raw data then producing detections.

A disadvantage is that the raw information being condensed into detections can result in information loss (when detections are imperfect). The system also becomes more complex since instead of running the already present detection networks on more information there is now a local perception stage and then an aggregation stage. The aggregation stage is when local and received detections are combined to form final detections.

Data Correctness

Cooperative perception requires data sharing between vehicles which introduces security risks. Mondal and Mitra describe a system setup for vehicle identification [51] so that the identity of the vehicle can be reliably determined. However, the correctness of the information that a vehicle shares is not guaranteed. A vehicle could disseminate false information unintentionally, for example by using a poorly performing perceptual network or miscalibrated sensors. Unfortunately, malicious information could also be spread with the intention of causing harm. Incorrect data which is received and believed could potentially cause fatal accidents. For example, if the ego-vehicle is travelling at highway speeds and a malicious agent sends a false detection of a stopped vehicle a few metres in front of the ego-vehicle position, the vehicle control system may initiate a collision avoidance procedure which could put the ego-vehicle occupants, or other third party members, at risk.

Obst et al. [52] present a method for checking plausibility of V2V data by using local perception at the tracking level. They base their model on sequential probability ratio testing (SPRT), which is essentially a discrete Bayes filter which uses tracks and detections over a time period to estimate the probability of existence. Their system is based off the assumption that objects have a high probability of being detected if they are in the FoV and are not heavily occluded. They show through experimentation that their system is able to detect ghost vehicles (false vehicles added through V2V communication) which begin outside the FoV then enter into the FoV through a passing maneuver. They set up a real-world experiment with a few limited test cases.

3.2.3 Trust Modelling Techniques

In this section various trust modelling techniques used in VANETs and other similar ad hoc networks are described and evaluated based on their ability to meet the aforementioned goal of improving cooperative perception. There are an abundance of papers on trust modelling for VANETs, Mobile Ad Hocs (MANETs), and other similar multi-agent systems. The following papers were selected to provide some background on the topic or to present ideas which could be incorporated into a final solution. In the papers which are presented it is either assumed, or a secure system is presented (such as in [46]), that messages can be securely delivered and the sender can be properly identified. As an example, identification can be accomplished through a private and public key pair. This is not the focus of this thesis and these parts of the models will not be discussed; assume vehicles can be identified and messages securely delivered.

Threshold Methods

Li et al. propose a simple model [46] for dealing with malicious entities in VANETs. They label their model reputation based announcement (RBA). They assume a central server is present which contains the reputation of each vehicle. The message is simply believed if the vehicle which sent the message has a reputation above a certain threshold. When vehicles receive a message from a vehicle, they first request the reputation certificate from the server. Reputation certificates can be stored for a time amount t which is an input parameter. The central server computes the reputation based on feedback from vehicles. Binary feedback is used in the experiments for simplicity. If the vehicle experiences an event from received messages it may report to the server if the messages were correct or incorrect. The central server stores these messages in a database for another period of time (input parameter) until the messages are discarded. The central server updates the reputation score for a vehicle every time a reported experience on them is discarded or added. The score is computed by first aggregating the current experiences per vehicles, using a weighted average based on how old the messages are (newer messages are weighted higher). Next, scores are aggregated equally over all vehicles which have provided feedback to obtain the final reputation score. By aggregating messages first by reporter, then between reporters, one reporting vehicle cannot dominate the reputation score. There is another input parameter which acts as a threshold; if a vehicle's reputation score falls under this value the vehicle is permanently revoked from the system and all future messages it sends will be disbelieved.

This system puts more effort into the message transfer and security than the reputation or trust score calculation. If the threshold values of the system are discovered than an entity could easily game the system by lying only so much to remain under the thresholds or by building up trust and then lying. With their work, Li et al. provide a good introduction to the concept of security within VANETs and the idea of entity reputation.

Raya et al. [56] propose a method general to ad hoc networks and test its performance on VANETs. Their scheme aggregates messages from multiple sources based on their trustworthiness score. The system measures events which are evaluated based on a message aggregation score. The message aggregation score is created by using either Bayesian inference, or Dempster-Shafer Theory (DST). Bayesian inference takes into account the prior probability along with a weighted probability that new messages provides that the event is true based on the message and the sender's trustworthiness. The calculations for DST add in plausibility which is calculated as the sum of evidence which does not directly disprove the event. If the event belief score is above a threshold parameter it is believed, if it is below another threshold parameter it is not believed, and if it is between these two thresholds more information is requested. This system model introduces the concept of not knowing if the event should be believed or not by having two thresholds. However, having to request more information for an event would not be possible when dealing with a perceptual system as it would dramatically increase latency.

POMDP Methods

Partially observable Markov decision processes (POMDPs) are used by a few authors for trust modelling in VANETs, MANETs, and other applications. Most methods, such as [34] use each entity and their trustworthiness score as states in the model. This can create an immense state space which does not scale well for VANETs. Zhang et al. solve this issue by creating one POMDP per event [13]. The authors assume the server has access to real traffic events, such as authority provided collision information, after a delayed period. It uses this information as the ground truth to compare with uploaded messages for calculating trustworthiness. The observations consist of the report (occurred or $\neg$ occurred), trustworthiness, and freshness (fresh/old depending on time since message). To avoid having to set threshold parameters the observation function is learned. The authors have a complex method of learning the observation function through adding experience as a Dirichlet distribution to find the posterior, computing a vector based on the posterior, clustering these vectors, then using the cluster center which contains the new experience as the observation function for the POMDP model for the current event. They use a reward function which returns negative values for wrong decisions, a cost for requesting more information, and positive values for a correct decision.

The experiments showed that their methods perform faster than the threshold methods outlined in the previous section. The authors also concluded that dynamic behaviours can be learned. They proved this by having vehicles switch behaviours throughout an experiment and show that after an initial dip the accuracy reaches similar levels. However, the authors found that there was a warm-up period since the behaviour patterns and vehicle trust need to be learned which takes time.

It would be interesting to experiment with this model using each potential object as a new POMDP. As it stands, the warmup period seems to make this model unusable as vehicles are only in proximity of each other for short periods of time. The model is more complex than other methods and would be more difficult to reproduce and integrate with perception.

Application Classes: Safety Focus

Souissi et al. [64] propose a self-adaptive trust management model which chooses to believe a message based on different information depending on the application class. The authors break the information into three primary classes (examples are provided for each class but there are many potential uses):

- Comfort: file transfers, streaming services
- Efficiency: platooning, routing
- Safety: emergency vehicle warnings, collision avoidance

The authors outline how most methods use previous experience to build a trust score; however, depending on the application class, there is other information which may prove useful. They state that vehicles are fast moving and rarely come within proximity to communicate more than once, exchanging information for a brief period. They use this as validation for basing trust evaluation not only on a vehicle's reputation. The authors add a similarity component to the trust evaluation which measures the similarity between a received message and other messages which have recently been received. The motivation behind this is that if there is an event such as a traffic collision or roadwork, many vehicles will be sending messages which are similar. They base the similarity on validity (occurrence or $\neg$ occurrence) and location. The score is calculated from the number of similar events. The authors also propose other methods:

- *Behaviour assessment*: Measuring if attributes such as speed match the report of an accident or traffic flow.
- *Contextual information*: Incorporating weather information (such as snow) for event alerts such as dangerous turns to determine if the context could be a factor in this event.
- *Risk assessment*: Determining risks associated with believing or disbelieving the report. For example, if there is a collision ahead and traffic is stopped disbelieving the event would have a much higher risk value.

The authors discuss how safety applications need to perform with much lower latency than the other two application classes. They envision a system where different combinations can be used for different applications. They also suggest that doing a risk assessment can be used to determine if other methods need to be used to decrease the uncertainty in the belief.

The authors note that sensor failures could be a cause of unintended false information. This is especially true for perception and a trust model which integrates perceptual information from multiple vehicles should take this possible error source into account.

Uncertainty

Balkrishnan et al. introduce uncertainty to trust modelling for mobile ad hoc networks (MANETs) [5]. Their motivation behind this work was that trust models for these networks fail to portray the ignorance value within the trustworthiness score. Examining how a new vehicle is given a trust score reveals why having a single value for trustworthiness is not sufficient. Should a new vehicle be given a low trust score, an average trust score, or a perfect trust score? A low trust score means the vehicle is untrustworthy but may truly mean that it is unknown if the vehicle is trustworthy. A high trust score may signal the vehicle has done nothing wrong but information may be readily believed when it should not be. Average trust may seem the most plausible but Balakrishnan suggest this is not the best option since there is nothing known about this vehicle at this point in time. They suggest using an uncertainty score in combination with a trustworthiness score.

Role vs. Experienced Based Trust

Zhang et al. [74] and Minhas et al. [50] address the issue of initializing a vehicle's trust level through the means of roles. They define three different roles from highest to lowest trust as:

1. *Authority*: Police cars, traffic controllers, RSUs
2. *Public Services*: Ambulances, fire trucks, public transit, etc.
3. *Professional cars*: Driver-training vehicles, drivers with more than 10 years of safe driving experience

These roles improve trustworthiness accuracy, especially for a vehicle which first enters the network. An example where this would benefit would be a police car pulling out of a station. This vehicle has the need to travel fast as soon as it is started and needs other vehicles to pull over quickly. By giving it a role other vehicles will readily trust the police car. A weakness with role based trust is when or if a vehicle with a trusted role becomes compromised by a third party with malicious intent. Vehicles may be susceptible to trusting these vehicles whereas if roles were not involved they may have been able to identify that the vehicle has become untrustworthy. However, drivers can be incentivized to report truthfully ([49], [50]) through a system whereby those with poor reputation will lose the opportunity to receive reports from peers.

Relay Control and Clustering

Zhang et al. also propose a novel clustering method in [74] which helps reduce false information spread between the cluster nodes. They note other clustering methods have been used but theirs expands upon these in two ways:

1. The trust opinions are also aggregated along with messages to reduce quantity of information between cluster nodes.

2. Majority opinion is used as a relay control to decide if information should be disseminated.

This method seems to coincide extremely well with integrating VANETs with perception. There are three reasons for this:

1. Geographic clustering: Vehicles near to each other are more likely to perceive the same objects. By clustering vehicles which are nearby it is likely the vehicles will end up with a similar detections which then reduces the complexity when trying to determine which detections are of the same object.

2. Many repeat objects: Many vehicles will perceive the same objects and in dense urban areas there can be many objects perceived at one instance in time. By only allowing one score per object per cluster the information flow can be drastically reduced.

3. Stronger cluster beliefs: By aggregating the detection scores between vehicles in the cluster it will likely lead to a more accurate score for each detection.

3.3 The TruPercept Dataset

Ideally, data would be collected from the real-world, as it would most accurately represent autonomous driving conditions; however, collecting data from the real-world for cooperative perception has resulted only in limited test-case scenarios. The models are usually tested for specific attacker scenarios and the effect of the cooperative perception model on the overall perceptual results is unknown. To the best of our knowledge, there are no existing publicly available real-world or synthetic cooperative perception datasets for autonomous driving. Creating a real-world dataset would require a fleet of vehicles equipped with autonomous sensor systems to collect data simultaneously. This would require large amounts of resources which were beyond our budget.

Trust modelling for V2V are often evaluated through the use of simulation software. Chen et al. [13] use OMNET++ [65] for V2V communication and SUMO [7] for road traffic simulation. However, these simulators do not contain sensory level perceptual information. Experiments with the TruPercept model require perceptual data for nearby vehicles. Simulators such as CARLA [17] are a viable options as they can obtain synthetic perceptual data and attempt to simulate traffic flow. Since perception algorithms need to be run from multiple view points this would require high amounts of compute power and is unlikely to be possible in real-time.

Alternatively, as discussed in the previous chapter, some autonomous driving datasets have been created from video games. Data can be collected ahead of time and be processed, manipulated, and evaluated after collection. GTA V is a video game which contains realistic graphics and has been used to generate synthetic data in a diverse environment for AV perception [37]. Recent efforts have even created an in-game LiDAR point cloud generator [72]. In the previous chapter, and my published work [33], I extend the range of [72] and improve the accuracy of generated point clouds. I show that the data is similar to real-world driving conditions and is suitable for evaluating the TruPercept method.

The TruPercept dataset consists of two series of captures, spanning approximately 300 and 400 seconds respectively. The data is captured at 1 Hertz and contains identical data to the PreSIL dataset for the ego-vehicle and each vehicle within a 100 metre radius. A 100 metre radius is used to restrict the total number of vehicles. Ideally, 160 metres will be used as the LiDAR scanner used for the PreSIL dataset has a max range of 80 metres so vehicles must be within double the max LiDAR range to have overlapping point coverage areas and thus obtain matching detections. The data is captured over two routes through urban environments. The dataset and all code used to generate the data is available at `https://tinyurl.com/y2nwy52o`.

3.3.1 Data Limitations

The synthetic data is easily attainable and attempts to mimic real-world vehicle behaviour and perceptual difficulty accurately. Unfortunately, due to the game engine of GTA V, the data collection between vehicles could not be perfectly synchronized. The camera position cannot be switched and rendered while the game is paused, instead the game speed is set to the minimal value. This results in perspectives being captured with time discrepancies and alignment issues with some of the data. An example of the alignment problems can be seen in Figure 3.1. Due to this problem, systems should seek to identify time offsets and compensate for them. In some cases, there is no overlap between 3D ground truth bounding boxes of the same object from different vehicle perspectives. This can severely impact any cooperative perception method and reduce or eliminate any potential performance gains.

To mitigate the effects during experimentation, if there was overlap with a detection from the ego-vehicle, the position and heading from the ego-vehicle was typically used. This improves performance as the evaluation is performed with ground truth data captured at the same time as the ego-vehicle sensor data.

This problem could potentially be used to simulate the inability to completely synchronize data sources between vehicles in real-world situations.

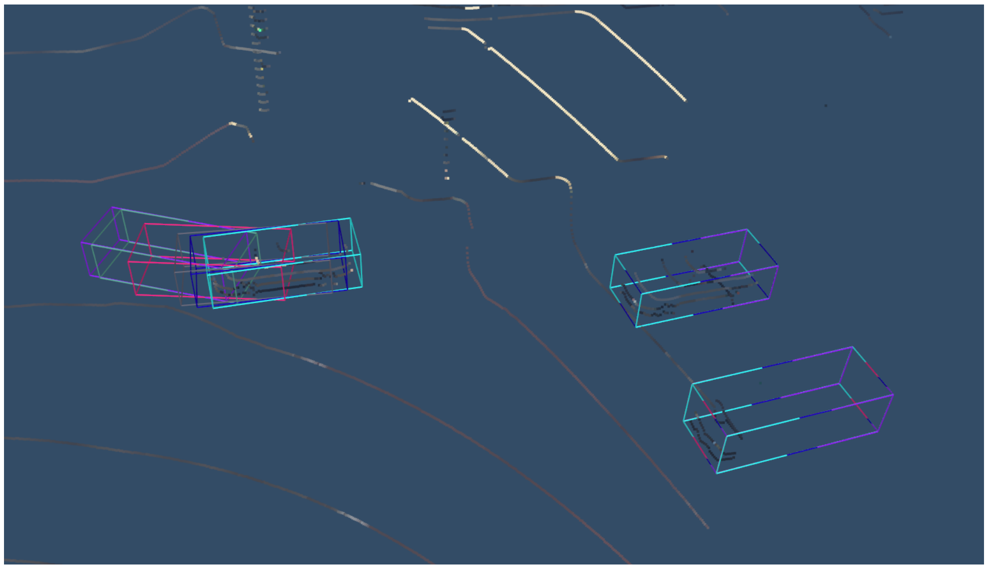

Figure 3.1: Ego-vehicle point cloud annotated with ground truth data from all nearby perspectives. Ground truth bounding boxes are annotated with a different colour for each perspective. Note that the bounding boxes for the moving vehicle on the left are not perfectly aligned, whereas the stationary vehicles on the right have perfect alignment resulting in the multi-coloured boxes. (Best viewed in colour)

Only perspectives from cars were captured. Perspectives from motorcycles and large vehicles should be included in future work. Large vehicles will likely require different LiDAR vertical angles as the tops of the vehicles are much higher. They could likely benefit from a perception network trained with data captured from the alternate viewpoint. Motorcycles were excluded since the camera positioning sometimes ended up inside the motorcyclist (busses also experience this issue). They will require a different camera/LiDAR positioning than other entities.

3.3.2 Data Synchronization

Since the data becomes nearly useless in some frames with larger time discrepancies an attempt to synchronize the data was made. The synchronization corrections are for first-order motion. For each perspective, the detections are matched to the respective ground truth objects. If matched, a velocity can be obtained. The game does not provide accurate time-stamps. To obtain a time-difference estimate the fastest moving object is used. The difference in ground-truth positions is used to obtain a distance which is then divided by the speed to obtain the time passage. This can then be used to correct positions for the remaining detections which were matched to ground truth objects. Detections which are not matched are not synchronized.

For example, all objects in Figure 3.2 are matched with the respective ground truth object from their detecting vehicle. Velocity-corrected bounding boxes for all of these objects can be seen in Figure 3.3.

This is not perfect as some objects (such as false positives which don't have an associated ground truth object) will not be matched, speed can change over the time period, and orientation of the object can also differ. This type of synchronization is possible in the real-world since velocity can be obtained from tracking systems and time discrepancies could be obtained if vehicles time-stamp data from Global Navigation Satellite System (GNSS) clocks. GNSS clock drift and GPS denied areas are a source of error which are more difficult to correct, research on this problem is being done by Hasan [29].

To improve upon this synchronization method, the orientation difference was also used. The change in orientation can be determined for matched objects by measuring the difference between the ground truth objects. Half of this is applied to the detection, then the distance based correction is applied, finally the remaining half of the rotational synchronization is performed. The rationale behind this method is that it accounts for the change of the velocity vector orientation over time. A top-down view of moving vehicles is used to demonstrate the effectiveness of this mechanism. The velocity and orientation corrected bounding boxes from the same example frame can be seen in Figure 3.4.

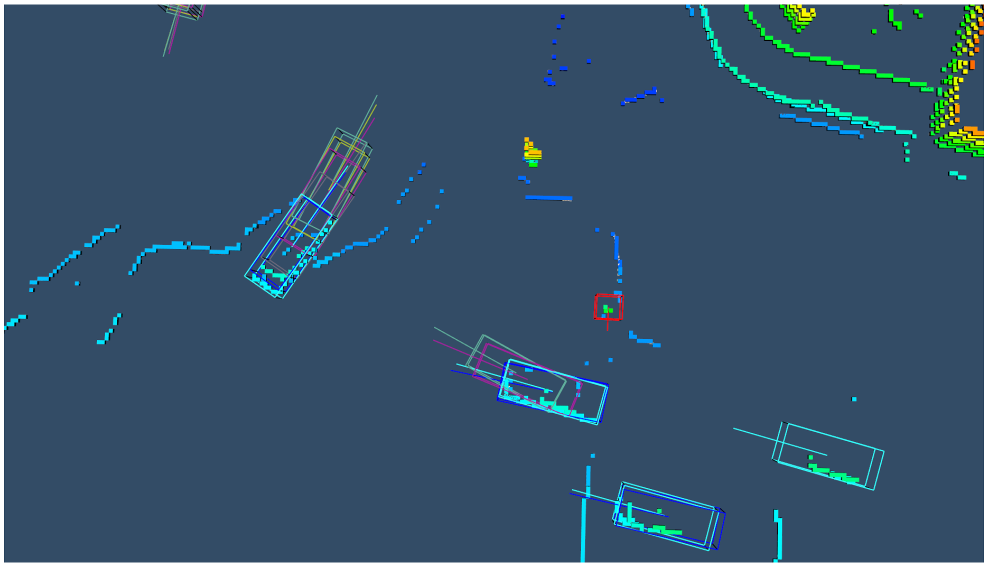

Figure 3.2: Ego-vehicle point cloud annotated with ground truth data from nearby perspectives. Ground truth bounding boxes are annotated with a different colour for each perspective. No synchronization method is used.

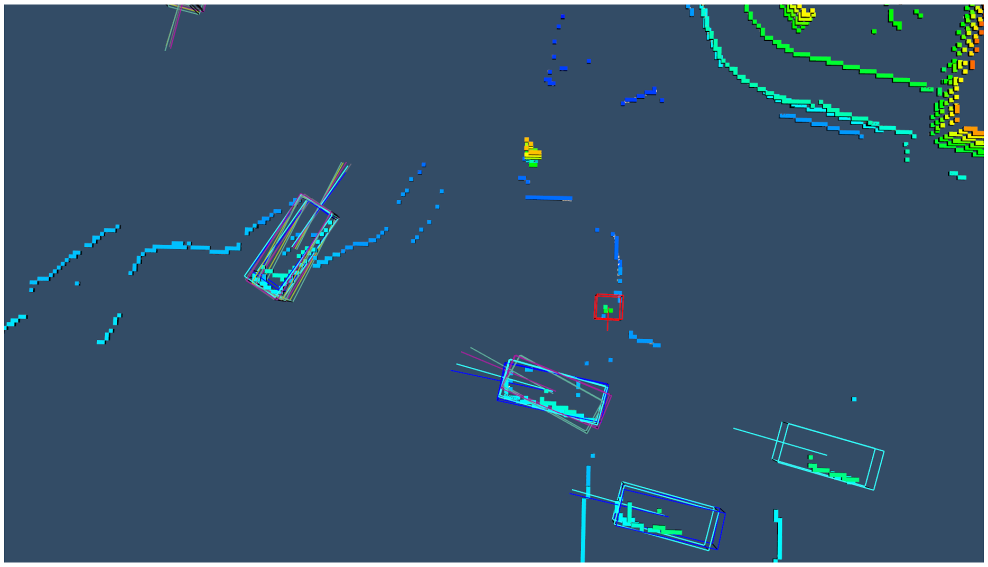

Figure 3.3: Velocity synchronization is used.

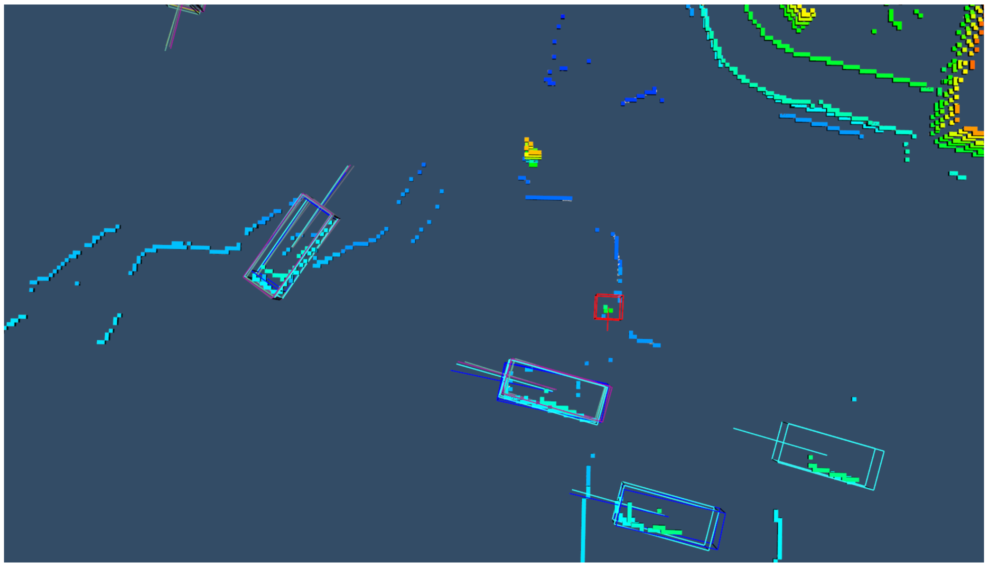

Figure 3.4: Velocity and orientation synchronization is used.

3.4 TruPercept Model

Inspired by some of the existing related work, this section outlines a novel integrative approach to perception and trust for AVs.

3.4.1 Perception

There are many computer vision models designed for 3D object detection in AVs such as [42] and [54]. 3D Object detection models typically output a set of objects (Δ). Each object instance will be denoted as θ and is detailed with a type (vehicle, pedestrian, cyclist), 3D position relative to on-board GPS (x,y,z), 3D bounding box dimensions (width, height, length), heading (rotation around up axis $\in \{-\pi, \pi\}$), and a confidence score (thresholding the confidence scores produces a list of detections). Objects are passed along to other systems of the AV such as for object tracking or for decision making (behaviour planner). This model aims to improve the accuracy of 3D object detections by incorporating detections from other vehicles. Different 3D object detection models can simply be interchanged within this system.

The following notation will be used while formally defining the model:

- θ: An object detection instance in the KITTI format [24]
- Δ_V: A list of all detections from a vehicle V
- $\Delta_V(3)$: An instance θ which is the third detection of vehicle V

Each AV will send a message containing the local 3D object detection network detections every instance they are produced. Each message (list of detections) will also contain a time specifying when the raw data was captured. Ideally, vehicles will be synchronized (e.g., by using GPS clocks) so that all sensors are triggered to capture with as much temporal-proximity as possible. Messages will be broadcast to nearby vehicles which can determine to rebroadcast (to extend message range) if the message is within a preset range and time period.

3.4.2 Ground Truth Data

Experience based trust is usually calculated by evaluating messages which are received from nearby vehicles. Messages are compared with the ground truth data so that a trust value can be produced for each message. Ground truth data for 3D object detection encompasses true values for each object (θ) within a range of the ego-vehicle. This is a major problem with trust modelling for perception as ground truth data would require the exact information (θ) for each vehicle, person, and cyclist. Ground truth data is only available in a controlled environment. Approximate true data could be obtained by having all connected vehicles report their own position but this would be subject to the same issues as the reports of the sensed environment. Realistically, receiving complete ground truth information for 3D object detection in the real world is infeasible. Therefore, for current solutions, a method must be created to evaluate received 3D object detection instances (θs) without access to the ground truth data.

3.4.3 Detection Evaluation

This section discusses how a vehicle, the ego-vehicle, will evaluate detections received from nearby AVs. The ego-vehicle will produce a score, denoted as the evaluation score, for each detection it receives.

Evaluation by Matching

The ego-vehicle is already evaluating whether detections are present or not, within the FoV of onboard sensors, by means of a 3D object detection network. It is therefore a natural idea to evaluate received detections on whether or not they match the local detections of the ego-vehicle. This section proposes several modules to evaluate object detection instances received from other vehicles. The following notation will be used:

- IoU: $\Theta \times \Theta \to [0, 1]$: Returns the Intersection over Union (IoU) of the 3D bounding boxes of θ and θ'. For this model, typically $\theta \in \Theta^{v1}$ and $\theta' \in \Theta^{v2}$. This function is used to match detections between vehicles. Detections with an IoU greater than 0 are at least partially overlapping. A 2D example of IoU can be seen in Figure 3.5 and the equation for IoU is

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} \tag{3.1}$$

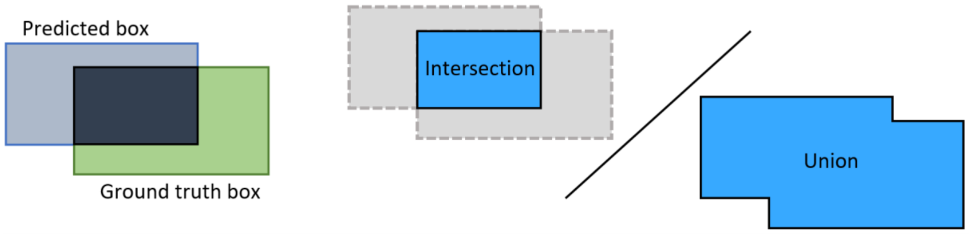

Figure 3.5: Visualization of IoU [35]

- $s^v : \Theta \to [0, 1]$: The local network score of a detection θ from a vehicle v. Network scores are produced by object detection networks and attempt to represent confidence [42] or probability [28] the detection is correct. Typically, any network score over a threshold value is taken as a detection, thus the network score indicates how likely it is the detection exists.

- $e^v : \Theta \to [0, 1] \cup -1$: The evaluation score from a vehicle v evaluating a detection θ which it received. Evaluation scores should ideally be probabilistic between 0 and 1, where 0 represents that the detection is incorrect and 1 signifies the vehicle v is 100% certain the detection is correct.

Comparing to detections

The ego-vehicle, denoted as α, is already evaluating whether detections are present or not, within the FoV of on-board sensors, from a local 3DOD network. It follows naturally to evaluate received detections on whether or not they match Θ^{α}. Detections are compared between vehicles by transforming from a vehicle's camera coordinates to world space using each vehicle's GPS coordinates and intrinsic and extrinsic sensor calibration files. Methods for localization between vehicles in the case of GPS failures or to increase accuracy have also been studied by Kim et al. [39], Jimenez et al. [36], and Li [44].

Firstly, all received detections are filtered by local detection area (70 metres 90°FoV for AVOD [42] default configuration). Next, the received detections Θ^{v} (from a vehicle v) are matched with Θ^{α}, producing a set $\mathcal{M}$ of sets of matches Θ_m. Any unmatched received detections are each put into a new set Θ_m (containing only one unmatched object per set), which the next set of received detections $\Theta^{v'}$ (from a vehicle v') attempts to match against in addition to the ego-vehicle detections Θ^{α}. A matching set Θ_m has a detection from α (if matched), followed by all matching received detections. Two detections are considered a match if the IoU is over a threshold parameter τ and are the same object class (e.g. car, pedestrian, cyclist). IoU for matching is computed with the first object in the set. Each received detection will be matched to the ego-vehicle detection (or first object in the set) with which it has the highest IoU. Only one detection per received vehicle detection list is contained in any matching list Θ_m. Any detection θ which α receives which matches one of its own detections θ' will have e^{α} equal to $s^{\alpha}(\theta')$. If there is no match with θ, α will set $e^{\alpha}(\theta)$ to η, a negative evaluation constant. Experiments are conducted with η set to either 0 or -1. For each received detection θ, and the detection of α labelled θ', where θ' is the object instance in Θ^{α} which has the maximum IoU with θ, the following applies

$$e^{\alpha}(\theta) = \begin{cases} s^{\alpha}(\theta') & \mathrm{IoU}\,(\theta, \theta') > \tau \\ \eta & \text{otherwise} \end{cases} \tag{3.2}$$

Using only the local network score to evaluate detections is not sufficient. Objects which are further away, truncated, or occluded may have lower or η evaluation scores. For example, if an object θ is completely occluded from the ego-vehicle perspective, even though it is there, any received detection of this object will have $e^{\alpha}(\theta) = \eta$. To solve this problem, another value which encompasses the visibility of the detection is proposed. Several function definitions need to be defined:

- $p^{v} : \Theta \rightarrow \mathbf{Z}^{\geq}$: the number of points from the perspective vehicle v LiDAR point cloud $\mathcal{P}^{v}$ which are within the boundaries of an object θ.

- $\Phi^v : \Theta \to [0,1]$: the visibility value of a detection θ from a vehicle v. 1 is completely visible and 0 signifies no knowledge from the current viewpoint. For example, if a vehicle is evaluating a detection from another vehicle, it can have a visibility value of zero if the received detection is occluded from its sensors.

The true visibility of an object θ from a perspective v is difficult to calculate and will be approximated by $p^v(\theta)$. Let γ_l and γ_u be the lower and upper $p^v(\theta)$ limits for min/max visibility respectively. Visibility is calculated as

$$\Phi^v(\theta) = \min\left(1, \frac{\max\left(0, p^v(\theta) - \gamma_l\right)}{\gamma_u - \gamma_l}\right) \tag{3.3}$$

If there are more than γ_u points within the 3D bounding box of θ it will return a value of one. If there are no points residing in the 3D bounding box, then the object is likely obstructed and the visibility value will be set to 0 (the vehicle is completely unsure as to whether the received detection is correct or not). This follows the assumption that the more LiDAR points that return from an object, the higher the chance the 3DOD algorithm has of detecting it. Thus, the visibility score can be used as a means to estimate the confidence that there is no object for a negative (i.e. unmatched) detection evaluation.

After being calculated, the network confidence and visibility evaluations are sent to nearby vehicles attached to the original detection IDs to aid 3D object detection for others.

3.4.4 Trust

It can also be beneficial to evaluate the total information flow from a vehicle. For example, if a vehicle is broadcasting malicious information, the vehicle should be identified, so it can be ignored and appropriate measures can be taken. Trust calculation can be done centrally or by each vehicle. Central aggregation introduces a strong system requirement (central server), but is better as vehicles only enter within proximity of each other for short time periods ($\approx$ 15 seconds average in the TruPercept data). Trust values are calculated for each object and then vehicle on the central server and then periodically broadcast so vehicles may use the information while integrating detections from other vehicles.

The trust $\mathcal{T}$ for a detection θ^α will be denoted as $\mathcal{T}(\theta^\alpha)$ and will be aggregated using evaluations from all nearby vehicles V which received the detection θ^α (V excludes α). The evaluation from each vehicle $v \in V$ will be aggregated in proportion to how visible the detection is from each evaluator perspective v (i.e., $\Phi^v(\theta)$). Let the trust function for a detection from the ego-vehicle θ^α be $\mathcal{T} : \Theta \to [0,1]$.

$$\mathcal{T}(\theta^\alpha) = \frac{\sum\limits_{v \in V} \Phi^v(\theta^\alpha) \cdot e^v(\theta^\alpha)}{\sum\limits_{v \in V} \Phi^v(\theta^\alpha)} \tag{3.4}$$

The trust value for a vehicle v is calculated on a central server by aggregating the trust feedback from all detections that vehicle has sent in proportion to the local network score for each detection. This is done so that detections it is not confident in do not penalize its trust value too heavily. For example, if a vehicle is attempting to be malicious it could broadcast a false detection with a high network score (say 1), to try to trick other vehicles. This detection should have a higher weight in the trust calculation than a detection which it broadcasts with a low network score since the low network score is less likely to fool nearby vehicles. Let $\mathcal{B}^v$ be the set of all $\theta \in \Theta^v$ broadcast within a freshness period f. A freshness period is used so that old information does not substantially outweigh old behaviour. If a freshness period were not used, then a vehicle could build up trust and then start behaving maliciously. Let the trust function for a vehicle be $\mathcal{T}^v : \mathcal{B}^v \to [0,1]$.

$$\mathcal{T}^v(\mathcal{B}^v) = \frac{\sum\limits_{\theta \in \mathcal{B}^v} s^v(\theta) \cdot \mathcal{T}(\theta)}{\sum\limits_{\theta \in \mathcal{B}^v} s^v(\theta)} \tag{3.5}$$

3.4.5 Detection Aggregation

At this stage detections have arrived from nearby vehicles, and each detection has been matched and evaluated. Now, a final step needs to occur to aggregate matching detections into a set of detections which will be output by the system.

A final aggregation step (at each vehicle) produces a score ω for every matching set $\Theta_m \in \mathcal{M}$ where θ_{mi} is the ith detection $\theta \in \Theta_m$. There are several inputs to this stage: evaluations $e^v(\theta)$, visibility $\Phi^v(\theta)$, and trust of evaluator $\mathcal{T}(\mathcal{B}^v)$. Let the score function for a final detection be $\omega : \Theta_m \to [0,1]$.

Weighted Average

A simple aggregation system which uses a weighted average calculates final scores as

$$\omega^{\alpha}(\Theta_m) = \frac{\sum\limits_{\theta^v \in \theta_m} \Phi^v(\theta^v) \cdot \mathcal{T}(\mathcal{B}^v) \cdot e^v(\theta^v)}{\sum\limits_{\theta^v \in \Theta_m} \Phi^v(\theta^v) \cdot \mathcal{T}(\mathcal{B}^v)} \tag{3.6}$$

$\eta = 0$ for this method so that $\omega(\Theta_m) \in [0, 1]$. As an example, for an object θ (which exists), let us create the scenario where the ego-vehicle has a bad view of θ, $\Phi^{\alpha}(\theta) = 0.3$, and a low network score (and thus low evaluation score), $e^{\alpha}(\theta) = 0.3$. A nearby vehicle v has high visibility of the object $\Phi^v(\theta) = 1$, a high matching network score (and thus evaluation score) $e^v(\theta) = 0.9$, and a high trust score $\mathcal{T}^v = 0.8$. The final detection score from α perspective of the set of matching objects Θ_m for the object θ on the ego-vehicle will be

$$\omega^{\alpha}(\Theta_m) = \frac{0.3 \cdot 1 \cdot 0.3 + 1 \cdot 0.8 \cdot 0.9}{0.3 \cdot 1 + 1 \cdot 0.8} = \frac{0.81}{1.1} \approx 0.74 \tag{3.7}$$

The score of the object on the ego-vehicle has been increased due to the additional information received from the nearby vehicle v.

Additive (Positive and Negative)

The weighted average method is able to quantify the belief an object is present relative to the trust and visibility of the object. However, it does not account for the quantity of vehicles which perceived the object. It is expected that the higher the ratio of vehicles which perceive a visible object, the likelier the object is to exist. The additive aggregation adds to the final detection score for every vehicle which perceived the object and subtracts for every vehicle from which the object was not perceived (unmatched). $\eta = -1$ to create the positive/negative addition/subtraction mechanism. The aggregation is weighted by the object visibility $\Phi^v(\theta)$ and trustworthiness $\mathcal{T}^v$ of each evaluator v. The final score is

$$\omega^{\alpha}(\Theta_m) = \sum_{\theta^v \in \Theta_m} \Phi^v(\theta^v) \cdot \mathcal{T}(\mathcal{B}^v) \cdot e^v(\theta^v) \tag{3.8}$$

The resulting value of equation 3.8 is not bounded by 0 or 1; it is later bounded using

$$\omega^{\alpha}(\Theta_m) = min\left(1, max\left(0, \omega^{\alpha}(\Theta_m)\right)\right) \tag{3.9}$$

The trust value for the ego-vehicle should be set to 1 (since aggregation is occurring on the ego-vehicle), or higher than 1 if bias towards the ego-vehicle detections is desired.

Let us take the same scenario as used for Equation 3.7. The additive score is:

$$\omega^{\alpha}(\Theta_m) = 0.3 \cdot 1 \cdot 0.3 + 1 \cdot 0.8 \cdot 0.9 = 0.81$$

3.4.6 Plausibility Checker

Thus far, the message evaluation relies on visibility of objects. This could be taken advantage of if a malicious agent were to insert false detections where there are no points. For example, false detections are inserted half a meter off the ground directly in front of vehicles where there is open air. No LiDAR points would be registered and the visibility value would be calculated as zero even though the non-object is visible. If there are points behind the object, and no points in front or within the object bounding box, then the existence of the object is false with high confidence. Obst et al. [52] create a mechanism for determining false tracks for cooperative perception and use the term plausibility checker.

The TruPercept system incorporates a novel plausibility checker which performs a frustum cull on the point cloud. The frustum is centered to the object center and extends min(width, length)/4 perpendicular to the vector from camera to object center. This 2D box size is used as the end of the frustum so that it is highly likely to only contain points from the desired object if it is not occluded. If more than 10% of the points are closer than the object center (i.e., the LiDAR hit the object in question or something ahead of it), the object is considered plausible. Realistically, 10% is a very low percentage of points likely to be in front of object center, but if there is any occlusion in this small frustum the object is deemed plausible. This stems from the fact that poor localization could shift the detected object center from the real object center. This could then lead to a frustum which is mostly unoccupied by the object (e.g., between an extended arm and the body of a walking pedestrian). Figure 3.6 shows a point cloud before and Figure 3.7 after the frustum cull of the plausibility check from multiple viewpoints. Since there is a vehicle in front of the pedestrian in this example, the plausibility checker would return plausible (>10% of remaining points are in front of the pedestrian).

The plausibility check can be performed in two key spots.

- During message evaluation. Can provide strong negative evaluations instead of not contributing to the evaluation due to low/zero visibility.

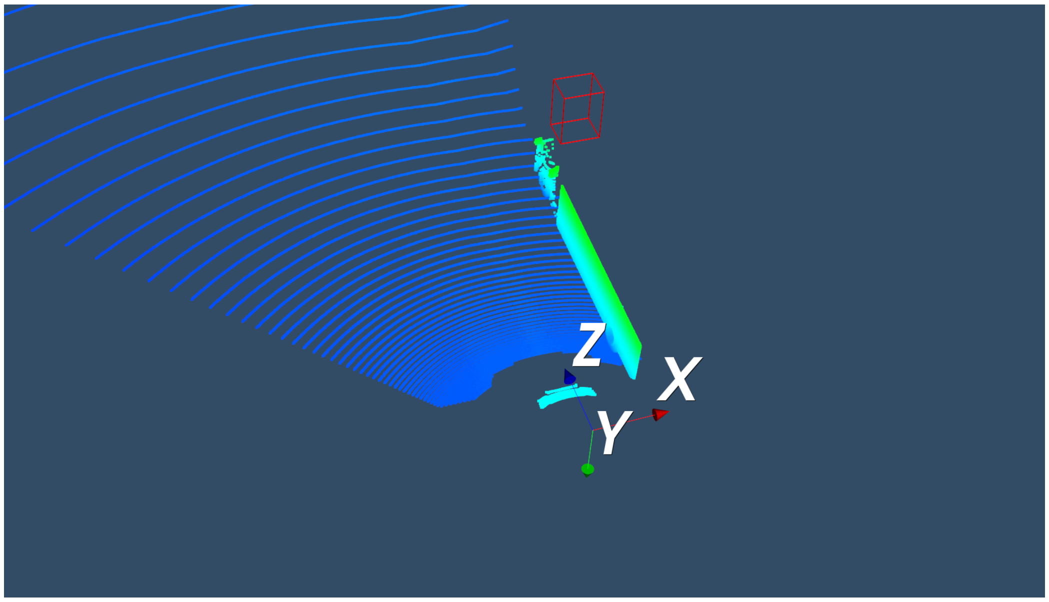

Figure 3.6: Example point cloud with occluded pedestrian which is used for the plausibility check in Figure 3.7. Z-axis is forward, X-axis is right.

- After aggregating detections. Vehicles can perform a plausibility check as a final measure to remove detections which were erroneously aggregated to scores higher than zero (i.e. false positives in the ego-vehicles line of sight which do not contain points between ego-vehicle and object center).

Vehicles should perform a plausibility check on any detection it did not match with and which does not have any points in its 3D bounding box (from the evaluating vehicle perspective). If the plausibility checker determines that the detection is invalid it is given a strong negative score, i.e. visibility of 1 and the negative evaluation score, η, of 0 or -1 (experiments are conducted with both values).

3.4.7 Additional Checks

Additional checks can also be added at this stage. AVs are complex systems and behaviour could become erratic or dangerous if incorrect detections are passed through to behaviour planning systems (for example if an incorrect detection suddenly shows a vehicle a few metres ahead in the direction of motion). Methods suggested by Souissi et al. [64] and

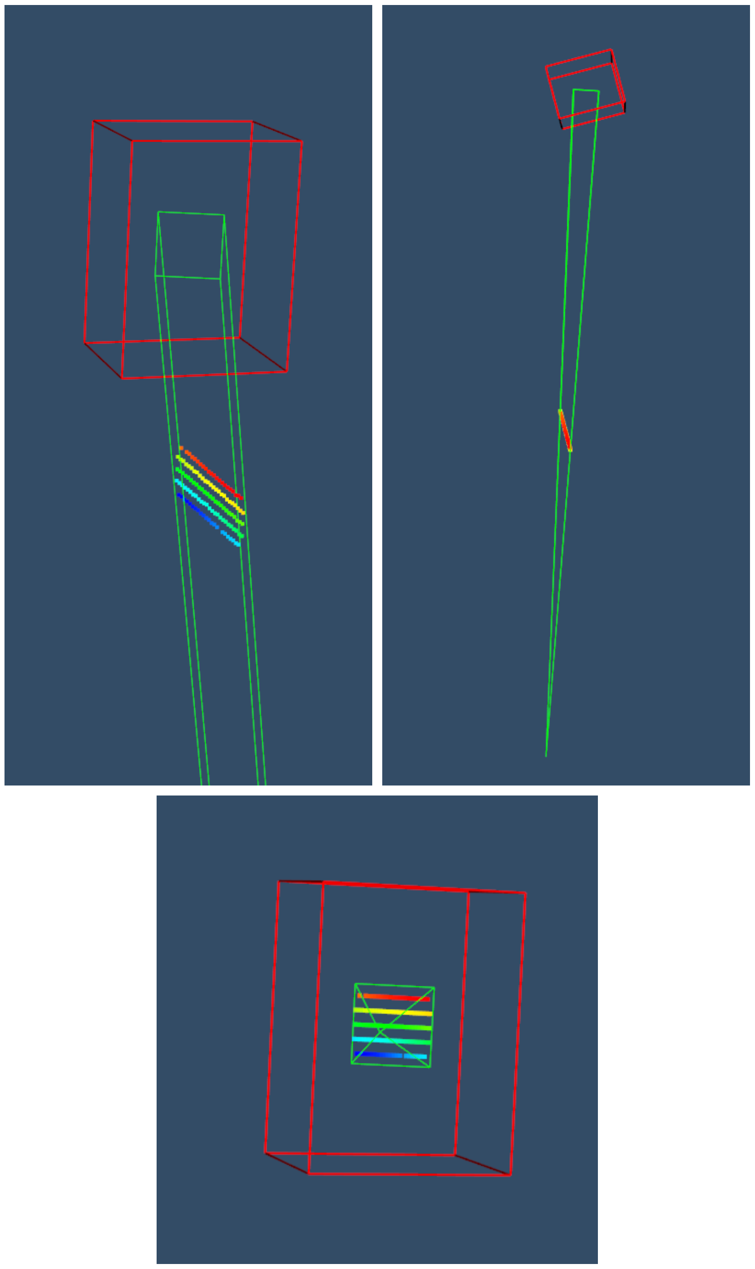

Figure 3.7: Green lines indicate boundaries of the frustum cull. Pedestrian bounding box in red. Points are coloured based on height. Top left: Regular view. Top right: Top-down view. Bottom: View along camera to object center vector. These images contain points from the vehicle occluding the pedestrian which are located in the frustum.

described in section 3.2.3 could be integrated at this point to secure the system further and reduce uncertainty. As an example, evaluating risk based on the detection type and location could be integrated at this stage of the model. The behaviour of the ego-vehicle should likely change if it is predicting an imminent collision, even if the final score in the object is low.

3.4.8 Clustering and Relay Control

Perception for autonomous driving is time-critical and minimizing processing time is a major design factor. The clustering algorithm and relay control proposed by Zhang et al. [74] has the potential to reduce total processing time and latency of information flow between vehicles. The time between sensing and final detections may be reduced by integrating this clustering method as it would:

- Decrease vehicle instance matching by only propagating one instance per vehicle from the cluster level.
- Decrease total information flow between clusters (potential reduction in bandwidth).
- Restrict feedback to vehicles which have views on similar objects, limiting unnecessary matching attempts.

One negative effect of clustering and relay control is that it may limit feedback from alternative viewpoints, thus potentially reducing the quality of detections for unseen vehicles. The method is detailed in [74] and outlined in section 3.2.3. Little modification would be necessary therefore details will not be provided in this thesis. Experimentation with these techniques would be worthwhile if implementing the system for real-time use. As these experiments were not conducted on a time-sensitive system they were omitted from experimentation.

3.4.9 System Summary

Each vehicle perceives, broadcasts to peers and receives broadcasts from peers, in order to derive its model of what it is perceiving. The system follows a repetitive cycle where vehicles detect objects, broadcast their detections, evaluate received detections, broadcast their evaluations, then aggregate detections to produce final detection scores (where sufficiently trustworthy reports are integrated). In detailed steps this translates to

1. Obtain sensor data
2. Run detection network to obtain Θ^α
3. Broadcast Θ^α
4. Receive $\Theta^v \; \forall \, v \in V$ or stop at timeout t
5. Calculate $\mathcal{M}$
6. Calculate and broadcast $p(\theta), e(\theta) \; \forall \, \theta \in \Theta^v, v \in V$
7. Receive $p(\theta), e(\theta) \; \forall \, \theta \in \Theta^v, v \in V$ or stop at timeout t
8. Calculate $\omega(\Theta_m) \; \forall \, \Theta_m \in \mathcal{M}$

The central server receives vehicle broadcasts of $e(\theta)$ and updates $\mathcal{T}(\theta)$ and $\mathcal{T}(\mathcal{B}^v) \; \forall \, \theta \in \mathcal{B}^v, v \in V \cup \alpha$ which it then periodically broadcasts.

3.4.10 System Example

In this section a simple example is presented to clarify calculations. The image from the ego-vehicle perspective α and a top-down view of the $\mathcal{P}^\alpha$ is presented in Figure 3.8. The vehicle directly in front of α will be labelled β. There is also a vehicle in the distance (not included in the point cloud) which will be labelled κ.

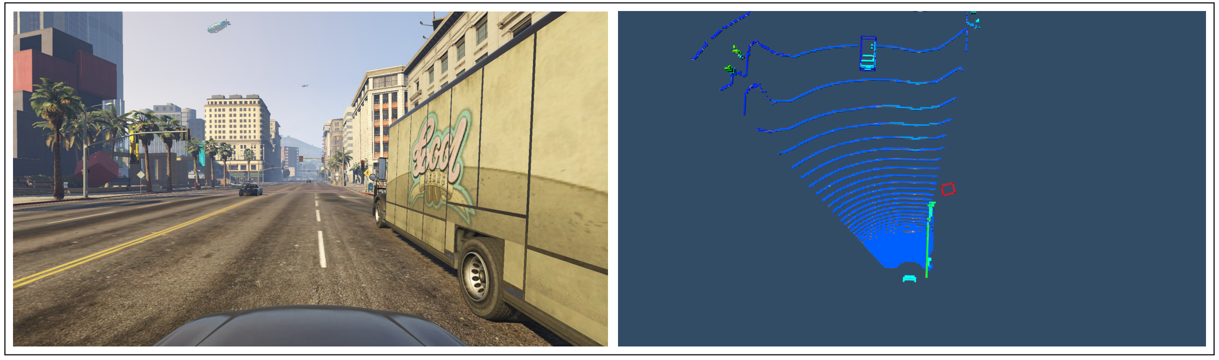

Figure 3.8: Ego-vehicle image and point cloud of scenario 1

For simplicity, the only detections which will be investigated are the detection of β denoted as θ_β, the detection of the pedestrian ρ denoted as θ_ρ, and a false detection χ

introduced by the vehicle β. The initial local outputs for each detection are reasonably contrived to simplify calculations for demonstration and are denoted in Table 3.1

Table 3.1: Local values for the example scenario. Each column in this table corresponds to a matched set Θ_m (assuming the detections all have an IoU $> \tau$)

Perspective	$s(\theta_\beta)$	$s(\theta_\rho)$	$s(\theta_\chi)$	$\Phi(\theta_\beta)$	$\Phi(\theta_\rho)$	$\Phi(\theta_\chi)$
α	0.9	-	-	1	-	-
β	1	0.9	1	1	1	1
κ	0.9	0.8	-	1	1	-

Detections are then broadcast to other vehicles, and vehicles create their reports about the detections they received. These evaluations are shown in Table 3.2. Notice these values largely remain the same, there were only a few additional calculations which needed to be computed for the unmatched received detections.

Table 3.2: Evaluation values for the example scenario. Each column in this table corresponds to a matched set Θ_m (assuming the detections all have an IoU $> \tau$)

Perspective	$e(\theta_\beta)$	$e(\theta_\rho)$	$e(\theta_\chi)$	$\Phi(\theta_\beta)$	$\Phi(\theta_\rho)$	$\Phi(\theta_\chi)$
α	0.9	η	η	1	0	0.3
β	1	0.9	1	1	1	1
κ	0.9	0.8	η	1	1	0

Next, final scores are calculated for each matching list from the ego-vehicle perspective α. Trust values for β and κ are initialized to 0.5. The scores of the two aggregation methods with and without the plausibility checker are shown in Table 3.3.

Table 3.3: Final values for the example scenario

Aggregation Method	Without Plausibility			With Plausibility		
	$\omega(\theta_\beta)$	$\omega(\theta_\rho)$	$\omega(\theta_\chi)$	$\omega(\theta_\beta)$	$\omega(\theta_\rho)$	$\omega(\theta_\chi)$
Average	0.925	0.9	0.625	0.925	0.9	0
Additive	1	0.85	0.2	1	0.85	0

Default trust value of 0.5 was used in the previous calculations. At this point, the central server will have received all detection and evaluation messages from the vehicles and can update the trust values accordingly. First, the trust for each object is calculated, the calculation for the detection of β from the α perspective (i.e., θ^α_β) is

$$\begin{aligned}
\mathcal{T}(\theta^\alpha_\beta) &= \frac{\sum\limits_{v \in V} \Phi^v(\theta^\alpha) \cdot e^v(\theta^\alpha)}{\sum\limits_{v \in V} \Phi^v(\theta^\alpha)} \\
&= \frac{\Phi^\beta(\theta^\alpha_\beta) \cdot e^\beta(\theta^\alpha_\beta) + \Phi^\kappa(\theta^\alpha_\beta) \cdot e^\kappa(\theta^\alpha_\beta)}{\Phi^\beta(\theta^\alpha_\beta) + \Phi^\kappa(\theta^\alpha_\beta)} \\
&= \frac{1 \cdot 1 + 1 \cdot 0.9}{1 + 1} \\
&= 0.95
\end{aligned}$$

The trust for all detections is shown in Table 3.4.

Table 3.4: Detection trust values for the example scenario

Detection	θ^α_β	θ^β_β	θ^β_ρ	θ^β_χ	θ^κ_β	θ^κ_ρ
Trust $\mathcal{T}(\theta)$	0.95	0.9	0.85	0	0.95	0.9

Once the trust of all objects has been calculated the central server can calculate the vehicle trust value. Assuming there are no previous trust calculations on the server (for simplicity), the trust for β will be calculated as

$$\begin{aligned}
\mathcal{T}(\mathcal{B}^{\beta}) &= \frac{\sum\limits_{\theta \in \mathcal{B}^{\beta}} s^{\beta}(\theta) \cdot \mathcal{T}(\theta)}{\sum\limits_{\theta \in \mathcal{B}^{\beta}} s^{\beta}(\theta)} \\
&= \frac{s^{\beta}(\theta^{\beta}_{\beta}) \cdot \mathcal{T}(\theta^{\beta}_{\beta}) + s^{\beta}(\theta^{\beta}_{\rho}) \cdot \mathcal{T}(\theta^{\beta}_{\rho}) + s^{\beta}(\theta^{\beta}_{\chi}) \cdot \mathcal{T}(\theta^{\beta}_{\chi})}{s^{\beta}(\theta^{\beta}_{\beta}) + s^{\beta}(\theta^{\beta}_{\rho}) + s^{\beta}(\theta^{\beta}_{\chi})} \\
&= \frac{1 \cdot 0.9 + 1 \cdot 0.85 + 1 \cdot 0}{1 + 1 + 1} \\
&= 0.583
\end{aligned}$$

Instead of being recalculated on every occurrence, a running total of the numerator and denominator can be kept on the central server as well as the total value added for each frame in the freshness period. The trust for all vehicles is displayed in Table 3.5

Table 3.5: Vehicle trust values for the example scenario

Vehicle	α	β	κ
Trust $\mathcal{T}(v)$	0.95	0.583	0.925

3.5 Experiments

Several experiments were designed to evaluate the TruPercept method and answer some primary research questions:

- Q1: Can the TruPercept model detect objects which are not visible?
- Q2: Can the TruPercept model identify erroneous detections and exclude them?
- Q3: Can TruPercept trust values properly identify a malicious vehicle?
- Q4: Can the TruPercept model improve upon local perception for visible objects?
- Q5: How is performance affected when faced with unreliable or malicious agents?

These experiments are detailed in this section.

3.5.1 Qualitative Analysis

Scenario 1: Algorithm Analysis Using a Single-Frame

A scenario was constructed to evaluate qualitatively whether the novel cooperative perception solution can meet two primary goals:

1. Accurately detect objects which are not visible to the ego-vehicle (Q1)
2. Filter out erroneous and malicious detections (Q2)

The scenario contains the ego-vehicle passing by a parked truck. A pedestrian, occluded from the ego-vehicle perspective, is walking out from behind the truck attempting to illegally cross the road. Another vehicle moving in the opposite direction of the ego-vehicle has a clear line of sight to the pedestrian. The image and a top-down view of the LiDAR point cloud, obtained from the ego-vehicle perspective, is displayed in Figure 3.9. The LiDAR point cloud is annotated with ground truth bounding boxes (blue for cars and red for pedestrians).

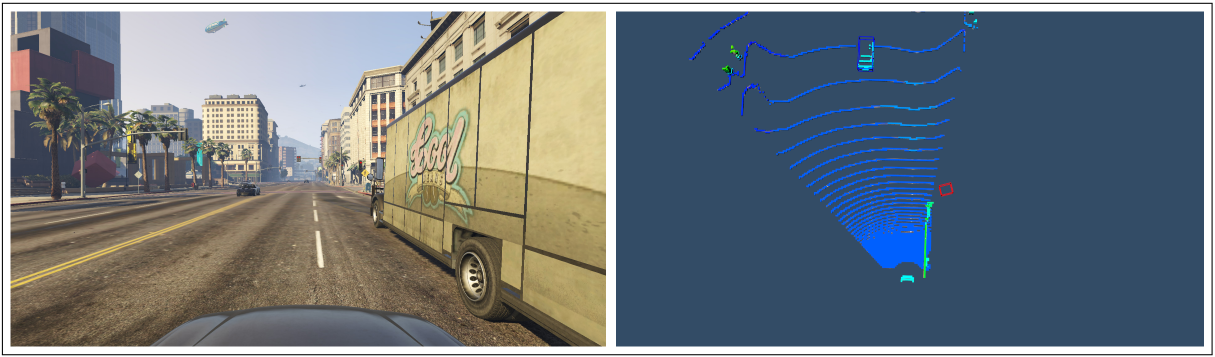

Figure 3.9: Ego-vehicle image and point cloud of scenario 1

However, the base scene is missing a means to test the ability to correctly identify detections which are erroneous. A false detection is inserted into the broadcast of the oncoming vehicle. The false detection is in a dangerous location for the ego-vehicle; less than 10 metres away, directly in its trajectory, and oncoming. The false detection will be considered malicious and is given a high network score (1.0) in an attempt to fool the ego-vehicle. Figure 3.10 shows the detections, with network scores, from the ego-vehicle (left), and the oncoming vehicle (right).

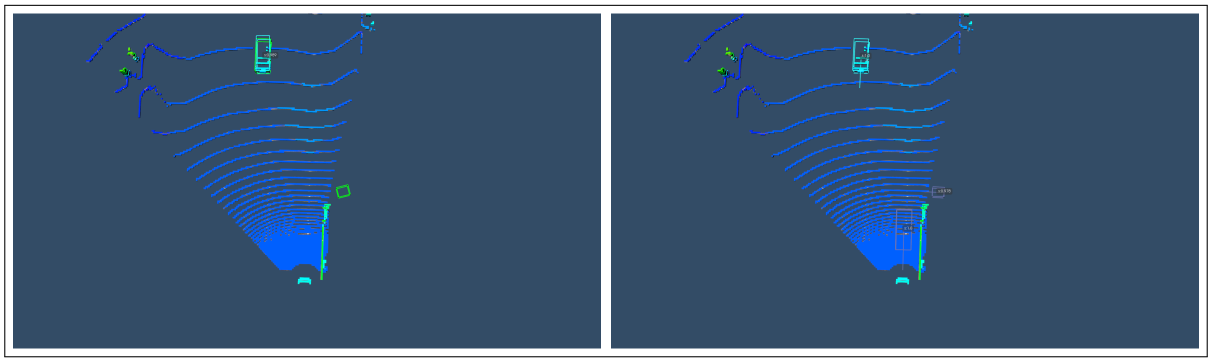

Figure 3.10: Ego-vehicle (left) and oncoming vehicle detections (right). Oncoming vehicle has inserted a false detection in front of the ego-vehicle. Ground truth is shown in green on the ego-vehicle perspective (left), local detection score ($s(\theta)$) on boxes. The vehicle on the right includes its own object with a score of 1. Detections of the ego-vehicle have been filtered out.

When using the TruPercept cooperative message evaluation mechanism as the final perception result, the system is able to correctly detect the pedestrian which is occluded from the ego-vehicle perspective (Q1). Unfortunately the false detection, although the score is decreased, is still present. This is extremely dangerous as belief in a false detection at close proximity to the ego-vehicle could cause erratic behaviour. However, if the plausibility checker is also run, it can eliminate the possibility that the false detection exists while still maintaining the pedestrian which is occluded (Q2). The results from the TruPercept algorithm without and with the plausibility checker added in as a final check after producing final scores ω can be seen in Figure 3.11. The false detection is completely removed and the occluded pedestrian is still detected.

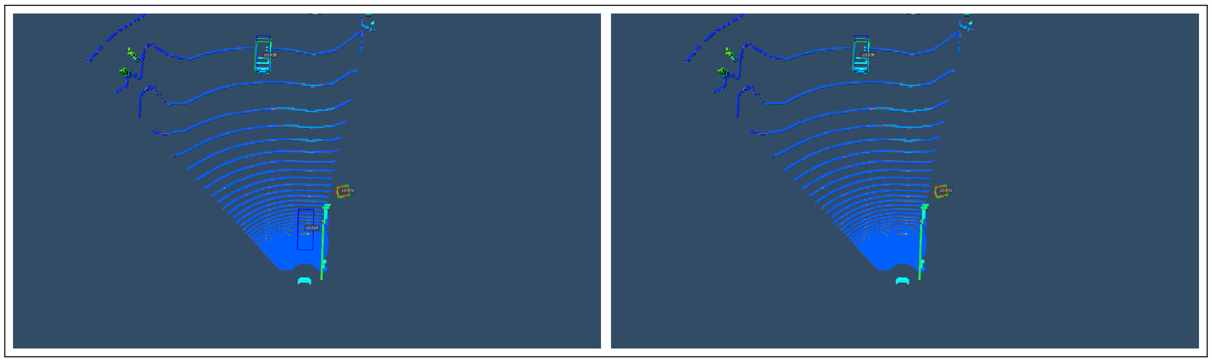
Figure 3.11: Cooperative perception results without (left) and with (right) the plausibility checker. Ground truth in green, final detection score (ω) on boxes.

Scenario 2: Vehicle Trust

This experiment attempts to show that the TruPercept model can correctly identify a malicious vehicle (Q3). Trust is accrued over time by the evaluations of peers. To mimic this, the single frame from the previous scenario is replicated 100 times and the trust score is examined at various instances in time. The false detection is inserted after 50 frames so the vehicle has time to build up trust. A freshness timeout of 50 frames is used so that any evaluation further than 50 frames in the past is discarded and not used towards the vehicle trust.

A secondary situation is introduced where the oncoming vehicle places three false detections in front of the ego-vehicle, also starting at frame 51. This scenario is depicted in Figure 3.12. The vehicle trust values are shown for certain instances in time in Table 3.6.

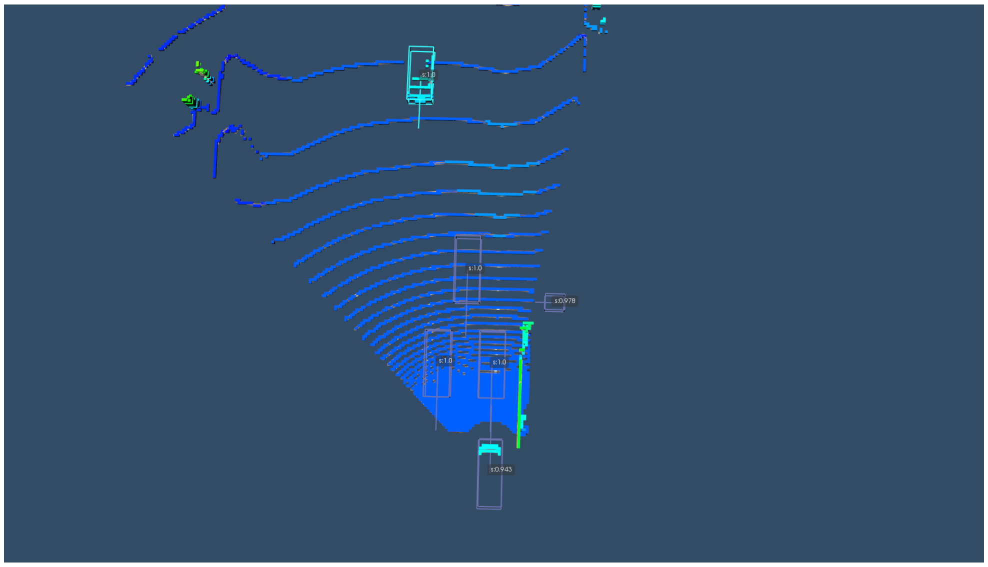

Figure 3.12: Ego-vehicle point cloud with detections ($s(\theta)$) from the oncoming vehicle (marked with the cyan bounding box). Three false detections are added in front of the ego-vehicle to show comparison with only adding a single false detection.

Table 3.6: Results comparing vehicle trust values over time (measured in frames)

Scenario	Vehicle	Time (frame number)					
		0	25	50	60	75	100
One false	Ego	0.5	0.82	0.82	0.82	0.82	0.82
	Malicious	0.5	0.81	0.81	0.67	0.47	0.15
	Far-Oncoming	0.5	0.97	0.97	0.97	0.97	0.97
Three false	Ego	0.5	0.82	0.82	0.82	0.82	0.82
	Malicious	0.5	0.81	0.81	0.52	0.18	0.0
	Far-Oncoming	0.5	0.97	0.97	0.97	0.97	0.97

As can be seen, initially the trust values start at 0.5, then over time the trust value for the malicious vehicle first increases, then becomes low. The trust value decreases faster and balances out at a lower score when the vehicle outputs more false detections. Notice that the trust of the vehicles which do not provide malicious detections remain unchanged, thus the TruPercept algorithm was able to correctly identify the malicious vehicle (Q3).

Note that there is a third vehicle not depicted in the scene images (labelled as far-oncoming). This vehicle has a clear view of the occluded pedestrian and all other objects in the scene but is further away. It contributes to the positive detection of the occluded pedestrian.

3.5.2 Quantitative Experiments

This analysis shows for the first time an attempt to quantify the effects of using cooperative perception on the general perceptual pipeline. The TruPercept dataset is, to the best of our knowledge, the first of its kind and expands the possibilities for research. Constructed scenarios were an initial attempt to evaluate cooperative perception models. Now, the ability to evaluate models on scenes of regular driving will allow cooperative perception methods to progress in ways that were not possible until now.

Baseline Cooperative Perception Solutions

To evaluate the efficacy of various approaches and understand the quantitative effects they have on the evaluation criteria a few baseline solutions have been created for cooperative perception. These approaches are simple and are meant to show differences in several simple cases:

- **Believe All (BA) 1:** Believes all detections. If detections are overlapping they are aggregated into a single object with the parameters from the ego-vehicle detection if present, otherwise the highest confidence detection. Every object is output with a confidence score of 1.
- **BA 2:** Same as the BA 1 solution except when the match list length is only 1, the object is included with the confidence score of the vehicle which detected the object.
- **BA 3:** Same as the BA 1 solution except when the match list length is only 1, the object is included with the confidence score of the vehicle which detected the object only if the detecting vehicle is the ego-vehicle, otherwise it is excluded.
- **Believe Ego (BE+):** Only ego-vehicle detections are included. If the match-list for the ego-vehicle is greater than 1, the score is increased to 1, otherwise the confidence score remains the score of the ego-vehicle detection.

Evaluation Criteria

The KITTI evaluation criteria of easy, moderate, and hard are defined by limiting values on occlusion, truncation, and minimum height (in pixels) of objects. This is too narrow of a scope for cooperative perception, as it allows detection of objects which are completely occluded, truncated, or are far in the distance. To provide a deeper inspection into the TruPercept system a new evaluation criteria is defined. The ‘All’ criteria encompasses any object which is within the forward facing 90 degree FoV area up to 140 metres away. Evaluation is restricted to the 90 degree FoV area since autonomous vehicles can use a 360 degree FoV sensor suite therefore it is unfair to include objects which could be detected but are not due to the limited FoV of the dataset.

Cooperative vs. Local Perception

The first experiment is designed to evaluate if the TruPercept model improves upon single perspective methods (Q4). AVOD [42], a state-of-the-art 3D object detector, is used as a baseline for detections. AVOD, with the feature pyramid, is trained using the synthetic data from the PreSIL dataset [33]. The performance of AVOD is evaluated for the ego-vehicle using the KITTI 3D Object detection [24] criteria, and the average precision (AP) measure. The mean AP score is taken between all routes of the TruPercept dataset. Next, AVOD is run for every nearby vehicle (within a 100 metre radius), and the TruPercept model is run to obtain updated scores. The scores of these methods are displayed in Table 3.7. Several variations of the TruPercept model were also included after failing to achieve improved performance compared to local perception. The TruPercept models are outlined below:

- **TruPercept 1**: Weighted Average, $\eta = 0$
- **TruPercept 2**: Summed Positive and Negative, $\eta = -1$
- **TruPercept 3**: Local detections were used primarily. Any non-matching TruPercept 2 detection θ is used if, from the ego-vehicle perspective V, $\Phi_V(\theta) < 10$ and the plausibility check returns true.

The parameters used for the TruPercept models were:

- For cars: $\gamma_l = 0$ $\gamma_u = 100$

- For pedestrians: $\gamma_l = 0$ $\gamma_u = 40$
- IoU threshold $\tau = 0.1$

Table 3.7: Results comparing detection accuracy with several different models (% AP)

	Car				Pedestrian			
Method	Easy	Mod.	Hard	All	Easy	Mod.	Hard	All
AVOD	66.8 %	62.8 %	52.1 %	36.3 %	84.2 %	78.7 %	75.1 %	57.3 %
BA 1	15.1 %	13.0 %	11.3 %	9.5 %	24.8 %	24.0 %	26.3 %	22.2 %
BA 2	43.6 %	36.6 %	30.8 %	23.9 %	67.7 %	61.8 %	62.3 %	49.4 %
BA 3	45.6 %	34.8 %	32.5 %	24.8 %	73.3 %	67.5 %	66.5 %	52.4 %
BE+	53.5 %	42.2 %	38.7 %	28.4 %	81.7 %	72.1 %	73.8 %	56.9 %
TruPercept 1	54.6 %	48.9 %	43.9 %	31.5 %	76.7 %	68.7 %	70.0 %	54.0 %
TruPercept 2	57.2 %	51.4 %	46.0 %	32.8 %	78.6 %	70.4 %	71.4 %	55.0 %
TruPercept 3	64.8 %	62.1 %	51.1 %	36.7 %	83.4 %	78.8 %	75.8 %	58.5 %

Several insights can be gained from these results. The most important result is that none of the tested methods improve the perceptual accuracy by a meaningful margin over the local perception methods. This shows the importance of testing any modules which are added to the perceptual track with the entire system.

The believe all methods performed particularly poor. The three Trupercept models improved over the baseline and managed to improve upon local perception in several categories, primarily for pedestrians and lower visibility categories (Q4). This is likely due to pedestrians being smaller objects and cooperative perception improving accuracy of difficult classes more by introducing closer and alternative viewpoints.

There are several plausible explanations for the poor results output from the cooperative perception models:

- As stated in section 3.3, there are some alignment issues with the synthetic data which could cause performance issues for cooperative perception but not local perception. For example, if a detection is unmatched with ground truth in the synchronization process due to a large time disparity between frame collections, it will not get properly aligned. This could create problems with matching which in turn could create a false positive and negative evaluations even though it should be a matching object with positive evaluations.

- Each vehicle can potentially introduce false detections, when aggregating between multiple vehicles this could result in many more erroneous detections. If this is occurring, it would reduce the precision. Figure 3.13 shows the precision-recall curves for several of the models. It is evident that for many of the poorly performing cooperative perception models the precision is much lower. This is especially true for lower recall values, likely caused by false detections from nearby vehicles being inserted with high confidence scores. For example, if only one vehicle can see an area, if it outputs false positives from its local network then these will be introduced in the final output since there are no possible negative detections.

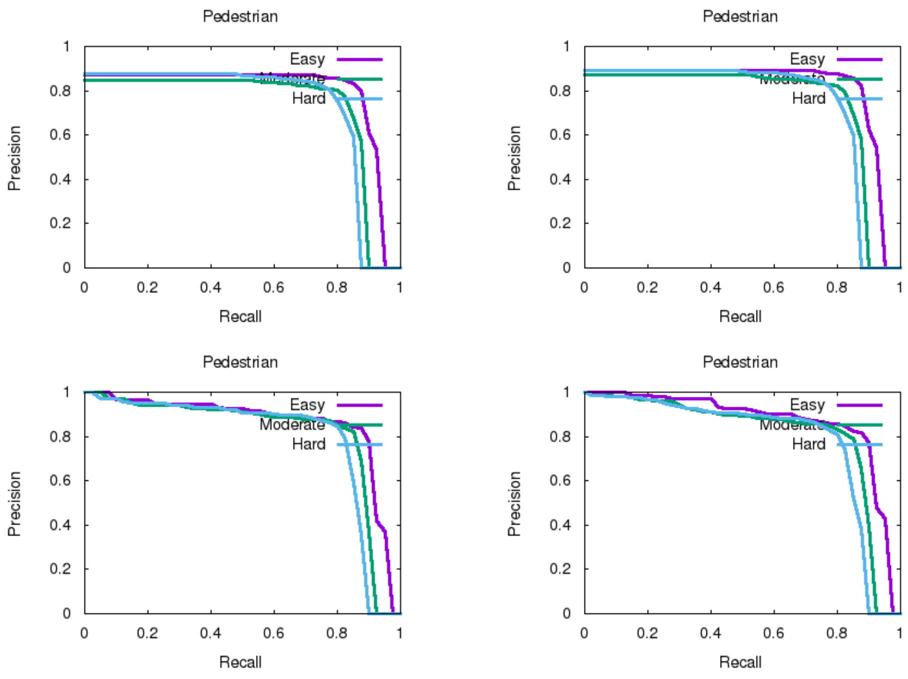

Figure 3.13: Precision-Recall curves for several methods. TruPercept 1 (top-left), TruPercept 2 (top-right), TruPercept 3 (bottom-left), AVOD (bottom-right). The precision-recall curves show the precision for all possible recall values. The higher the curve the better since precision is defined as true positives over true positives and false positives.

One point to note is that pedestrians have higher detection scores than cars. These detection rates are reversed in real-world data. This could be due to several factors such as:

- The bounding boxes for pedestrians are identical sizes in the synthetic data. This could make the regression task for pedestrians easier.
- There is an increased presence of larger vehicles such as trucks, busses, and SUVs which are not part of the 'Car' class. This could cause confusion for a detection network which is only attempting to label cars, resulting in more false positives. This could be rectified by training a network to detect these other classes and taking the class with the highest detection score when there are overlaps.

This could also be why the TruPercept models perform worse for cars. The more false detections the model has as input, the less likely it is to perform well.

Position and Orientation (TruPercept 4)

Position and orientation were originally taken from the ego-vehicle detections due to the synchronization issues with the data. After correcting for synchronization, as described in section 3.3.2, orientation and positioning modifications were investigated. The hypothesis is that vehicles which are closer to an object will provide more accurate positioning and orientation information. This could lead to increased accuracy if objects which are detected are not meeting the KITTI standard of a 0.5 IoU to be considered a true positive. To experiment, the AVOD-FPN detections are taken then their position and orientation is set to that of the closest detecting vehicle in each matching detections list. This resulted in an increase in AP of up to 5% for some criteria. The results can be seen in Table 3.8, labelled as TruPercept 4. It is revealing that the largest improvement to local perception is gained simply by modifying the position and orientation of local detections (Q4). This enables the model to ignore false positives from nearby vehicles and yet still improve the accuracy of local detections.

Table 3.8: Results comparing local perception to cooperative perception with only position and rotation updated from local detections (% AP)

	Car				Pedestrian			
Method	Easy	Mod.	Hard	All	Easy	Mod.	Hard	All
AVOD	66.8 %	62.8 %	52.1 %	36.3 %	84.2 %	78.7 %	75.1 %	57.3 %
TruPercept 4	69.5 %	61.0 %	54.1 %	37.8 %	89.0 %	80.6 %	80.1 %	61.0 %

IoU Threshold

The IoU threshold for two detections to be considered a match also plays an important role in cooperative perception. For higher IoU levels duplicate detections or missed evaluations are more likely to occur. For lower IoU threshold values nearby objects more likely be falsely grouped together and there is a higher probability of false evaluations. Experimentation with the IoU threshold level on the TruPercept3 method was completed. The results are in Table 3.9. The lower IoU threshold corresponded to a higher score.

Table 3.9: Results comparing detection accuracy with various IoU threshold values for matching detections (% AP)

	Car				Pedestrian			
Threshold	Easy	Mod.	Hard	All	Easy	Mod.	Hard	All
0.1	64.8 %	62.1 %	51.1 %	36.7 %	83.4 %	78.8 %	75.8 %	58.5 %
0.3	62.2 %	59.6 %	49.2 %	35.6 %	80.2 %	76.1 %	73.7 %	56.9 %
0.5	58.9 %	56.7 %	51.8 %	35.0 %	72.3 %	69.4 %	67.8 %	52.6 %
0.7	57.2 %	55.0 %	52.7 %	35.7 %	65.5 %	63.5 %	65.6 %	48.3 %
0.9	55.0 %	52.5 %	51.4 %	37.9 %	53.9 %	51.5 %	53.8 %	40.8 %

Figure 3.14 shows the precision/recall curves for the 0.1, 0.5, and 0.9 threshold values. The lower threshold value likely has higher scores due to two reasons:

1. Detections which are of the same object but overlapping less (due to poor localization) will be split into two detections at higher IoU thresholds. This would be seen with less precision at higher IoU thresholds.

2. Detections which are of the same object but overlapping less (due to poor localization) and are not matched due to a higher IoU threshold will cause vehicles to give poor evaluations on the object even though they are detecting the same object.

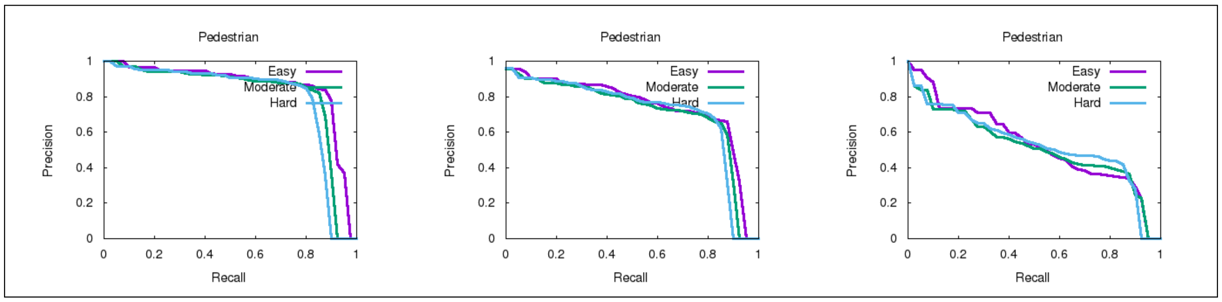

Figure 3.14: Precision-Recall curves for several IoU threshold values using the TruPercept 3 method.

These results show the importance of localization for 3D object detection. If detection locations are not accurate, it can affect cooperative perception algorithms downstream due to poor matching performance. In the future, alternative matching techniques could be investigated (such as also using distance metrics or appearance features) to attempt to improve the overall cooperative perception performance.

Trust Levels

The previous experiment evaluated detections from honest (although potentially not correct) vehicles. This experiment augments the dataset with malicious and unreliable vehicles to evaluate the performance of the TruPercept models while being exposed to entities with non-optimal behaviours (Q5). Behaviours experimented with are:

- Trustworthy: Base detections are taken from the output of the AVOD 3D Object detection network. This is the baseline with results in Table 3.7.
- Unreliable: 10% chance to add an incorrect extra detection for each local detection and a 10% chance to remove each of the local detections.
- Malicious: a vehicle and pedestrian are inserted in front of the ego-vehicle for every frame from 10% of vehicles.

Table 3.10 shows the performance differences of the main methods when detections are randomly added and removed (unreliable scenario). The car class shows a significant performance reduction when introducing the unreliable behaviours. The BE+ method reduces the least as it only accepts ego-detections. The pedestrian class also shows a reduction in performance for the baseline methods. Strangely, the pedestrian class actually improves in all categories for the TruPercept models. There could be several explanations for this:

- The unreliable vehicles receive lower detection scores, lowering the amount they contribute to the solution. If they had bad detections to begin with, the final score of these detections could be reduced.
- False detections also have a chance to be removed, these could be harder to identify as false detections since they will be located where points are, instead of random locations which could contain no points.

Table 3.10: Results comparing detection accuracy with the untrustworthy behaviour scenario (% AP)

	Car				Pedestrian			
Method	Easy	Mod.	Hard	All	Easy	Mod.	Hard	All
AVOD	66.8 %	62.8 %	52.1 %	36.3 %	84.2 %	78.7 %	75.1 %	57.3 %
BA 1	13.5 %	11.5 %	10.1 %	9.2 %	22.2 %	21.3 %	23.6 %	20.4 %
BA 2	35.6 %	30.7 %	26.1 %	22.1 %	68.6 %	61.3 %	62.1 %	49.7 %
BA 3	37.7 %	29.8 %	27.9 %	22.7 %	70.5 %	64.6 %	64.1 %	51.0 %
BE+	53.5 %	42.2 %	38.6 %	28.3 %	80.9 %	76.1 %	73.5 %	56.7 %
TruPercept 1	49.6 %	43.7 %	39.5 %	30.5 %	79.9 %	71.5 %	72.1 %	56.3 %
TruPercept 2	50.7 %	44.7 %	40.3 %	31.2 %	81.3 %	72.8 %	73.2 %	57.0 %
TruPercept 3	52.1 %	49.9 %	41.4 %	32.7 %	86.4 %	80.0 %	77.0 %	60.1 %

Table 3.11 shows the results when adding the malicious detections. The malicious detections decrease the performance significantly, even for pedestrians. The malicious detections behaviour represents a coordinated attack between 10% of the vehicles specifically targeted towards the ego-vehicle. This shows a weakness in the model towards coordinated attacks.

Table 3.11: Results comparing detection accuracy with the malicious behaviour scenario (% AP)

	Car				Pedestrian			
Method	Easy	Mod.	Hard	All	Easy	Mod.	Hard	All
AVOD	66.8 %	62.8 %	52.1 %	36.3 %	84.2 %	78.7 %	75.1 %	57.3 %
BA 1	13.6 %	11.5 %	10.1 %	11.0 %	21.7 %	20.9 %	23.1 %	20.1 %
BA 2	38.9 %	32.8 %	27.8 %	26.2 %	65.2 %	59.2 %	60.2 %	48.2 %
BA 3	41.2 %	32.0 %	30.0 %	23.6 %	71.0 %	65.3 %	64.7 %	51.5 %
BE+	53.5 %	42.1 %	38.6 %	28.2 %	81.7 %	76.3 %	73.8 %	56.9 %
TruPercept 1	49.0 %	44.2 %	40.0 %	30.8 %	76.1 %	68.4 %	69.6 %	54.6 %
TruPercept 2	50.9 %	45.9 %	41.4 %	31.9 %	78.0 %	70.1 %	70.9 %	55.5 %
TruPercept 3	53.4 %	51.5 %	42.7 %	33.7 %	83.1 %	78.5 %	75.7 %	59.2 %

The mean trust values at the termination of the experiments can be seen in Table 3.12. The trust value for unreliable vehicles is not significantly different than for vehicles which were left unmodified for the unreliable behaviour. Behaviours such as the unreliable behaviour, which mostly present true detections, will likely have a higher chance of fooling the current system (Q5). This is a major weakness of the trust modelling component. The trust model was able to detect blatantly malicious behaviour with a much higher success. This can be seen as the average trust value for malicious vehicles was less than half whereas the unreliable vehicles saw a minor decrease in their trust value.

Table 3.12: Comparing the mean vehicle trust value by vehicle trustworthiness types

Scenario	Vehicle type	Trust value
Trustworthy	Trustworthy	0.27
Malicious	Malicious	0.13
	Trustworthy	0.27
Unreliable	Unreliable	0.25
	Trustworthy	0.27

It should also be noted that the average trust value is fairly low, likely due to the amount of false positives which are being output, and alignment issues which are causing poor matching. Reducing false positives could greatly improve performance since false positives are being introduced by each nearby vehicle's local detection network instead of only the ego-vehicle detection network (there are many more false positives). If local detection accuracy increases, this value would be expected to increase.

Detections output by detection networks are typically accepted if above a threshold value. Originally, a threshold value of 0.1 was used (detections with a score of less than 0.1 were ignored). This reduces processing power as there can be many low scoring detections, but it was hypothesized that since these detections have low probability of being correct it would not affect performance. An interesting finding is that when more detections were filtered out, the overall trust value decreased dramatically. This caused problems as vehicles which had a detection with a low score which was removed would then evaluate any matching detections with the negative value η. When changing the 0.1 threshold value to 0.01 the average trust value changed from 0.03 to 0.27.

Data containing the IDs of malicious entities and the full detection lists for randomly added/removed detection methods will be provided so researchers can test performance with identical data. The described behaviours are simply a glimpse at strategies malicious entities could employ. Potential tactics could be coordinated and devised to circumvent trust modelling techniques. In the future, great effort should be placed on creating and testing behaviours which could cause dangerous situations.

3.6 Discussion

The TruPercept model introduces an apparatus to evaluate 3D object detections which are received from nearby vehicles. This enables new avenues of exploratory research. For example, this method could facilitate the collection of difficult training samples while driving autonomously. Cooperative perception could identify objects local perception has difficulty detecting and any false positives output from the network. These frames could then be used in training to improve local perception performance.

It may be that vehicle trust scores simply portray how reliable a vehicle's detections are, especially in the absence of malicious behaviour. It is likely a very small percentage of vehicles will be malicious; since malicious actions could cause death they could be severely punished. This should not be seen as a downside of the system as long as false or malicious information can be effectively detected, prevented from being used, and the agent which disseminated the false information can be identified.

For the TruPercept dataset, vehicles are within the information sharing range of the ego-vehicle only for a mean of 6.25 frames. Since the data was collected at approximately one to two Hertz this correlates to only 6.25 to 12.5 seconds in sharing proximity. This renders it practically impossible to obtain an accurate trust model of nearby vehicles that can be used for a moderate length of time. This information results in the recommendation that if vehicle trust levels are calculated, they should be done by a central server (which could be a distributed infrastructure, such as blockchain). This would increase the amount of data which could be used to calculate the trustworthiness of a vehicle and also minimize any warm-up period trust values may have when new vehicles enter into the vicinity of the ego-vehicle. Note that the behaviour of vehicles in GTA V may be not be completely representative of vehicle behaviours in the real-world but this information can be used as an initial estimate. Due to the technical and physical challenges of collecting data on all vehicles in the GTA V world simultaneously, a complete-world collection was not completed and vehicle trust cannot be fully experimented with using the TruPercept dataset.

Evaluating and comparing detections is a difficult task, especially if all entities are not synchronized. Synchronization could be done using GPS clocks since microsecond accuracy or better is available. If all systems are running at 10 Hertz, detections can be sent every 100 milliseconds, and synchronized at the zero of every second. However, some vehicles may run at different rates and restricting the running rates of algorithms might not be feasible. It may be more practical to integrate cooperative perception at the tracking level. Using speeds and headings for each object, the locations could be adjusted to differences in time. The confidence in a detection could then be adjusted over the passage of time instead of relying on a single detection instance. By including time as a dimension, it also makes objects easier to match. The TruPercept dataset is not suitable for use with tracking as the data rate is too low. The limitations of GTA V include no timestamps and the perspectives are captured with various synchronization errors and unknown times between. The times could be estimated but this was out of the scope of this thesis.

Another downside to this system model is increased latency. The perception algorithms on Waterloo's Autonomoose take less than 100 milliseconds. For this solution, first the 3D object detection network is run, then detections are exchanged, then evaluations are computed and shared, then all detections are aggregated to produce final detections. Having one or more layers of information transference could cause a delay in the shared detections and take up valuable resources. A possible solution is to run the shared detections at a lower rate. Another potential fix to this problem is to instead integrate cooperative perception at the tracking level since time steps are already part of the model.

The current solution takes into account perceptual confidence; however, it does not incorporate information on the amount of experience about a vehicle. This leaves the system susceptible to attacks from vehicles which enter the system, send a few detections to raise their trust value then begin attacking the system by sending detections in areas which no other vehicles have a viewpoint on but which may affect their driving pattern. This could be fixed by adding a factor by which a vehicle's trust is modified based on the quantity of experience.

Benchmark Proposal

To the best of our knowledge, there is no benchmark for comparing cooperative perception (with or without trust modelling) for autonomous driving. This is an important area where a benchmark challenge could be useful to help progress future work. Challenging scenarios and multiple behaviours could be added to the benchmark to increase difficulty and ensure proper coverage. The TruPercept data creation method is open-sourced, allowing easy creation of new scenes, and enables creation of new false detection scenarios simply by linking to a text file with the new detections. By providing this, along with all source code, it empowers researchers to compare their methods with ease and add new trustworthiness cases. Since the possibilities for malicious behaviours are vast, this will increase opportunities for in-depth testing of any new solutions which are suggested. Some additional scenarios which may highlight the benefits of distributed perception include:

- An emergency vehicle running a red light. (Which a vehicle does not have a line of sight to but another vehicle does.)
- Non-emergency vehicle running a red light. Coming from an obscured crossroads where a vehicle does not have a line of sight but other vehicles do.
- Simulate sensor failures on a vehicle with poor/faulty detections.

3.7 Future and Related Work

The TruPercept algorithms were developed for integrating reports from peers into the decision making of the vehicles and include a trust modelling component. We examine precision of the ultimate perception, in varied contexts of trust. Contrasting with models such as [50], [49], and [22] in that the TruPercept models have a critical focus on issues of perception, mapped out to operate alongside the trust modelling. The TruPercept cycle of acquiring detections from peers is similar to the computation models of Minhas et al. [50] and Finnson et al. [22] in that actions are taken based on the most trustworthy reports received.

These other papers have additional suggestions of value for future work, namely distinguishing the roles of vehicles as a way of modelling trustworthiness and experimenting with different calculations of majority consensus. Other interesting directions for expanding the TruPercept study of trust are suggested by Zhang [73]: coping with data sparsity and anticipating certain specific malicious attacks. These are other possible directions for the future. We have, however, offered some valuable new methods for running experiments to see the contribution of trust modelling (beyond the simulation methods of Finnson et al. [22] which focus on measuring average speed of vehicles).

Bayesian Deep Learning is a technique which aims to predict different types of uncertainties produced from the deep learning model [38]. Recently, Harakeh et al. presented BayesOD [28] which applies Bayesian deep learning to 2D object detectors for autonomous driving. The network score which they provide gives a more accurate estimate on the probability the detection is accurate. By using a Bayesian network score, it would allow for future cooperative perception methods to utilize these probabilities with Bayesian trust modelling techniques.

Other future work could include:

- Using a perceptual simulator such as CARLA [17] to collect synchronized cooperative perception data with timestamps.
- Experiments to examine how cooperative perception performs in high and low traffic density environments.
- Cooperative perception with 360 degree LiDAR.
- A study on the effects of untrustworthy vehicles on message evaluation. How can malicious vehicles manipulate evaluations to cause problems?
- How can vehicles manipulate the position and orientation of objects for malicious reasons? Should trust models have a separate module watching for this or are IoU matching evaluations enough?
- Mechanism Design to encourage vehicles to provide truthful information instead of disseminating information which benefits themselves.
- Model visibility function differently such as using a tanh function or creating a statistical model to show real probability of detection based on parameters such as point count, object class, and distance.

- Eliminating need for transfer of evaluations and visibility values, for example by calculating probability of visibility based on distance-metrics and evaluations with only positive matches. Having two stages of sending and receiving increases system latency. Visibility could, to some extent, be computed by each vehicle for each other vehicle (like ego reasoning using a map to deduce which vehicle might have the best view of a blind spot).
- Alternative heuristic methods:
  - Additive methods for vehicle trust
  - Penalize very low rated detections higher
  - Request assistance for truncated/occluded objects
  - Eliminate detections with low local confidence scores
  - Set minimum threshold to believe unmatched received detections
  - Reduce scores of unmatched received detections due to uncertainty in trustworthiness
  - Retroactively giving feedback when objects which were occluded or had poor visibility come into sight.
- Model each potential object as a POMDP such as with [13].
- Probabilistic models for aggregating all received messages into a final list.
- Deep Learning methods for aggregating all received messages into a final list.
- Exchanging appearance features beyond boxes to allow better matching, an example of which is presented by Baser et al. [6].

3.8 Conclusion

Multiple trust modelling methods for ad hoc networks were presented, with a majority being used for VANETs. The topic was evaluated from a new perspective, integration with 3D object detection. A multi-agent solution attempting to increase 3D object detection range and accuracy for AVs was presented. A new dataset for comparing cooperative perception with local perception methods is presented and is publicly available at `https://tinyurl.com/y2nwy52o`.

From this research there are a few points of interest:

- Cooperative perception creates a gargantuan security flaw for AVs since the correctness of incoming information cannot be guaranteed.
- Cooperative perception techniques need to incorporate trust modelling or incoming information evaluation. It is fairly easy to concoct situations which could cause an AV to act erratically and create dangerous situations.
- Cooperative perception methods should be integrated into perceptual pipelines and evaluated on large datasets to be considered relevant. Creating a module to prevent a contrived scenario from occurring is not necessarily hard. Integrating modules into complex perceptual systems to improve overall performance and prevent a plethora of prospectively hazardous scenarios is a significantly more onerous task.

Even though cooperative perception can introduce security flaws, it still has promise in extending perceptual range, reducing blind spots, and improving localization and orientation estimates. Hopefully future work can find greater improvements to perception through cooperative techniques. The takeaway from this research should not be reduced expectations of cooperative perception due to apparent security flaws. Alternatively, realize the dangers of introducing information from other agents and conduct a thorough assessment of possible risk factors and mitigation techniques so that cooperative perception can be introduced to AV systems without allowing malicious agents to expose security flaws.

Chapter 4

Conclusion

This thesis presented two novel synthetic datasets for AV perception. The first is an unlimited (generation code available) dataset for various perception tasks such as 3D object detection and point cloud segmentation. The dataset was validated by increasing the performance of AVOD-FPN, a state-of-the-art 3D object detection network. The second dataset is, to the best of our knowledge, the first cooperative perception dataset for AVs to present full image and point cloud data. This dataset enables testing of cooperative perception techniques which fuse information sources at any stage of the pipeline. Cooperative perception for AVs is an open area of research that could greatly improve perceptual accuracy when objects of interest are truncated or occluded. Cooperative perception could greatly increase safety in densely populated urban environments where people, cyclists, and vehicles can enter into the ego-vehicle's field of view at close proximity. In general, synthetic data is useful as huge datasets can be gathered at relatively low cost. Additionally, it is easier to create supplementary labels with refined information and in situations (cooperative perception) that are challenging to replicate in the real-world.

From the initial investigation into cooperative perception, the most beneficial augmentation was reached from using nearby vehicles to improve position and orientation estimates. Large numbers of false positives made integrating detections from multiple vehicles a difficult task. Methods which can more accurately identify which detections are false, malicious, and true appear to be the next step to increasing the benefits of cooperative perception.

References

[1] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber, and N. Kaempchen. Experience, results and lessons learned from automated driving on germany's highways. *IEEE Intelligent Transportation Systems Magazine*, 7(1):42–57, Spring 2015.

[2] Christoph Allig and Gerd Wanielik. Alignment of perception information for cooperative perception. In *IEEE Intelligent Vehicles Symposium*, 2019.

[3] Matt Angus, Mohamed ElBalkini, Samin Khan, Ali Harakeh, Oles Andrienko, Cody Reading, Steven Lake Waslander, and Krzysztof Czarnecki. Unlimited road-scene synthetic annotation (URSA) dataset. *CoRR*, abs/1807.06056, 2018.

[4] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. Cooperative object classification for driving applications. In *IEEE Intelligent Vehicles Symposium*, 2019.

[5] Venkat Balakrishnan, Vijay Varadharajan, and Uday Tupakula. Subjective logic based trust model for mobile ad hoc networks. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, SecureComm '08, pages 30:1–30:11, New York, NY, USA, 2008. ACM.

[6] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki. Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1426–1433, June 2019.

[7] Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, and Daniel Krajzewicz. Sumo – simulation of urban mobility: An overview. *Proceedings of SIMUL*, 2011, 10 2011.

[8] National Transportation Safety Board. Preliminary report, highway, hwy18mh010. https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf, May 2018. [Online; accessed 02-July-2019].

[9] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *CoRR*, abs/1903.11027, 2019.

[10] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] C. Chen, J. Zhang, R. Cohen, and P. Ho. A trust modeling framework for message propagation and evaluation in vanets. In *2010 2nd International Conference on Information Technology Convergence and Services*, pages 1–8, Aug 2010.

[12] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. *CoRR*, abs/1905.05265, 2019.

[13] Shuo Chen, Athirai Aravazhi Irissappane, and Jie Zhang. Pomdp-based decision making for fast event handling in vanets. In *AAAI*, 2018.

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[15] Alejandro Correa, Robert Alms, Javier Gozalvez, Miguel Sepulcre, Robbin Blokpoel Michele Rondinone, Leonhard Lücken, and Gokulnath Thandavarayan. Infrastructure support for cooperative maneuvers in connected and automated driving. In *IEEE Intelligent Vehicles Symposium*, 2019.

[16] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[17] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938, 2017.

[18] Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. Technical Report EN 302 637-2 V1.3.2, ETSI, Nov 2014.

[19] Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS) . Technical Report TR 103 562 0.0.15 Draft, ETSI, Jan 2019.

[20] Jin Fang, Feilong Yan, Tongtong Zhao, Feihu Zhang, Dingfu Zhou, Ruigang Yang, Yu Ma, and Liang Wang. Simulating LIDAR point cloud for autonomous driving using real-world scenes and traffic flows. *CoRR*, abs/1811.07112, 2018.

[21] Di Feng, Christian Haase-Schuetz, Lars Rosenbaum, Heinz Hertlein, Fabian Duffhauss, Claudius Glaser, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *CoRR*, abs/1902.07830, 2019.

[22] John Finnson, Jie Zhang, Thomas Tran, Umar Farooq Minhas, and Robin Cohen. A framework for modeling trustworthiness of users in mobile vehicular ad-hoc networks and its validation through simulated traffic flow. In Judith Masthoff, Bamshad Mobasher, Michel C. Desmarais, and Roger Nkambou, editors, *User Modeling, Adaptation, and Personalization*, pages 76–87, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[23] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[24] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[25] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[26] Craig Glennie and Derek D. Lichti. Static Calibration and Analysis of the Velodyne HDL-64e S2 for High Accuracy Mobile Scanning. *Remote Sensing*, 2(6):1610–1624, June 2010.

[27] Cosmo 3D Programmer's Guide. Chapter 9 viewing the scene. https://techpubs.jurassic.nl/manuals/nt/developer/Cos3C_PG/sgi_html/ch09.html. [Online; accessed 11-September-2019].

[28] Ali Harakeh, Michael Smart, and Steven L. Waslander. Bayesod: A bayesian approach for uncertainty estimation in deep object detectors. *CoRR*, abs/1903.03838, 2019.

[29] Khondokar Fida Hasan. Gnss time synchronisation in co-operative vehicular networks. Master's thesis, Queensland University of Technology, 2018.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[31] Laurens Hobert, Andreas Festag, Ignacio Llatser, Luciano Altomare, Filippo Visintainer, and Andras Kovacs. Enhancements of v2x communication in support of cooperative autonomous driving. *IEEE Communications Magazine*, 53:64–70, 12 2015.

[32] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. *CoRR*, abs/1803.06184, 2018.

[33] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *IEEE Intelligent Vehicles Symposium*, 2019.

[34] Athirai A. Irissappane, Jie Zhang, Frans A. Oliehoek, and Partha S. Dutta. Secure routing in wireless sensor networks via pomdps. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2617–2623. AAAI Press, 2015.

[35] Dominik Jargot. Deep end-to-end network for 3 d object detection in the context of autonomous driving. Master's thesis, Delft University of Technology, 2019.

[36] Adrián Jiménez-González, José Ramiro Martínez-de Dios, and Aníbal Ollero. An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. *Sensors*, 11(12):11516–11543, Dec 2011.

[37] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *CoRR*, abs/1610.01983, 2016.

[38] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[39] S. Kim, Z. J. Chong, B. Qin, X. Shen, Z. Cheng, W. Liu, and M. H. Ang. Cooperative perception for autonomous vehicle control on the road: Motivation and experimental results. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5059–5066, Nov 2013.

[40] S. Kim, B. Qin, Z. J. Chong, X. Shen, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus. Multivehicle cooperative driving using cooperative perception: Design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):663–680, April 2015.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[42] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *IROS*, 2018.

[43] Jungwook Lee, Sean Walsh, Ali Harakeh, and Steven L. Waslander. Leveraging pre-trained 3d object detection models for fast ground truth generation. *CoRR*, abs/1807.06072, 2018.

[44] Hao Li. *Cooperative perception : Application in the context of outdoor intelligent vehicle systems*. PhD thesis, l'École Nationale Supérieure des Mines de Paris, Sept 2012.

[45] Peilun Li, Xiaodan Liang, Daoyuan Jia, and Eric P. Xing. Semantic-aware grad-gan for virtual-to-real urban scene adaption. *CoRR*, abs/1801.01726, 2018.

[46] Q. Li, A. Malip, K. M. Martin, S. Ng, and J. Zhang. A reputation-based announcement scheme for vanets. *IEEE Transactions on Vehicular Technology*, 61(9):4095–4108, Nov 2012.

[47] Mark Martinez, Chawin Sitawarin, Kevin Finch, Lennart Meincke, Alex Yablonski, and Alain L. Kornhauser. Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars. *CoRR*, abs/1712.01397, 2017.

[48] Yosi Mass and Onn Shehory. Distributed trust in open multi-agent systems. In *Proceedings of the Workshop on Deception, Fraud, and Trust in Agent Societies Held During the Autonomous Agents Conference: Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, pages 159–174, London, UK, UK, 2001. Springer-Verlag.

[49] U. F. Minhas, J. Zhang, T. Tran, and R. Cohen. Intelligent agents in mobile vehicular ad-hoc networks: Leveraging trust modeling based on direct experience with incentives

for honesty. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 243–247, Aug 2010.

[50] U. F. Minhas, J. Zhang, T. Tran, and R. Cohen. A multifaceted approach to modeling agent trust for effective communication in the application of mobile ad hoc vehicular networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(3):407–420, May 2011.

[51] Atanu Mondal and Sulata Mitra. Identification, authentication and tracking algorithm for vehicles using vin in centralized vanet. In Vinu V. Das and Janahanlal Stephen, editors, *Advances in Communication, Network, and Computing*, pages 115–120, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[52] M. Obst, L. Hobert, and P. Reisdorf. Multi-sensor data fusion for checking plausibility of v2v communications by vision-based multiple-object tracking. In *2014 IEEE Vehicular Networking Conference (VNC)*, pages 143–150, Dec 2014.

[53] G. Pavlin, P. D. Oude, M. Maris, and T. Hood. Distributed perception networks: An architecture for information fusion systems based on causal probabilistic models. In *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 303–310, Sept 2006.

[54] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, abs/1711.08488, 2017.

[55] Matej Racinsky. 3d map estimation from a single rgb image. Master's thesis, Czech Technical University in Prague, Czech Republic, 2018.

[56] M. Raya, P. Papadimitratos, V. D. Gligor, and J. . Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, pages 1238–1246, April 2008.

[57] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. *CoRR*, abs/1709.07322, 2017.

[58] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. *CoRR*, abs/1608.02192, 2016.

[59] Konstantin Riedl, Sebastian Kurscheid, Andreas Noll, Johannes Betz, and Markus Lienkamp. Road network coverage models for cloud-based automotive applications: A case study in the city of munich. In *IEEE Intelligent Vehicles Symposium*, 2019.

[60] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[61] Artem Savkin, Monika Kasperek, and Federico Tombari. Sampling/importance resampling for semantically consistent synthetic to real image domain adaptation in urban traffic scenes. In *IEEE Intelligent Vehicles Symposium*, 2019.

[62] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-a^2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *CoRR*, abs/1907.03670, 2019.

[63] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 567–576, June 2015.

[64] Ilhem Souissi, Nadia Ben Azzouna, and Tahar Berradia. Towards a self-adaptive trust management model for vanets. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 6: SECRYPT, (ICETE 2017)*, pages 513–518. INSTICC, SciTePress, 2017.

[65] András Varga. The omnet++ discrete event simulation system. *Proc. ESM'2001*, 9, Jan 2001.

[66] Haris Volos, Takashi Bando, and Kenji Konishi. Reladec: Reliable latency decision algorithm for connected vehicle applications. In *IEEE Intelligent Vehicles Symposium*, 2019.

[67] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *CoRR*, abs/1903.01864, 2019.

[68] Waymo. Waymo safety report: On the road to fully self-driving. https://waymo.com/safety, 2017. [Online; accessed 02-July-2019].

[69] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3d lidar point cloud. *CoRR*, abs/1710.07368, 2017.

[70] Qing Xu, Tony Mak, Jeff Ko, and Raja Sengupta. Vehicle-to-vehicle safety messaging in dsrc. In *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, VANET '04, pages 19–28, New York, NY, USA, 2004. ACM.

[71] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: sparse-to-dense 3d object detector for point cloud. *CoRR*, abs/1907.10471, 2019.

[72] Xiangyu Yue, Bichen Wu, Sanjit A. Seshia, Kurt Keutzer, and Alberto L. Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. *CoRR*, abs/1804.00103, 2018.

[73] Jie Zhang. A survey on trust management for vanets. In *Proceedings of the 2011 IEEE International Conference on Advanced Information Networking and Applications*, AINA '11, pages 105–112, Washington, DC, USA, 2011. IEEE Computer Society.

[74] Jie Zhang, Chen Chen, and Robin Cohen. Trust modeling for message relay control and local action decision making in vanets. *Security and Communication Networks*, 6(1):1–14, 2012.

[75] Li Zhang. Cs 559: Computer graphics lecture 9: Projection. https://slideplayer.com/slide/7633069/, Spring 2008. [Online; accessed 11-September-2019].