Towards HD Map Updates With Crosswalk Change Detection From Vehicle-mounted Cameras

Tom Bu

CMU-RI-TR-22-34

August 2, 2022

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
John Dolan, Advisor
Christoph Mertz, Advisor
Deva Ramanan
Dinesh Reddy

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Many autonomous vehicles rely on high-definition maps that contain road layout and road semantics as priors for perception, planning and prediction. However, these maps can become stale over time as the road environment changes. This thesis develops a road monitoring framework that allows for automatic change detection of crosswalks with a cost-effective sensor suite of vehicle-mounted cameras and GPS data. Furthermore, this thesis explores using an edge computer on a commercial bus to receive and analyze live data captured in Pittsburgh.

Contributions of this thesis include evaluating object detectors trained from different types of datasets, representing crosswalks in the bird's-eye-view for more robust change detection, and finally incorporating the system on an actively running bus. The first contribution of this thesis is an evaluation of the CARLA simulator as an effective tool to provide automatic annotations for custom street-view objects on a simulated vehicle-mounted camera. Despite the sim-to-real domain gap, models trained on CARLA-generated annotations for two custom objects, fire hydrants and crosswalks, are shown to perform as well as those trained on 200 real-world images and can be used to augment existing datasets. The second contribution of this thesis is a method that maps detections from 2D images onto a ground plane by using multi-view geometry and 3D reconstructions of the scene. With this method, detections from multiple frames can be accumulated in the bird's-eye-view to better represent an intersection, and consistency checks can be performed to remove false detections. Lastly, this thesis explores using the crosswalk change detector in an edge-computing enabled commuter bus that has active cameras. With GPS locations of seventeen existing crosswalk intersections, the bus can send relevant images for the crosswalk change detector to analyze. Change detection results show robustness in high-traffic scenes where vehicles often occlude the road and robustness to pose differences between current and reference images.

Acknowledgments

I would like to thank my family, friends, classmates, and professors at CMU without whom this thesis would not be possible. Particularly, I would like to thank my advisors, John Dolan and Christoph Mertz, for their patience and guidance and for taking me on as a student, working with me, and imparting their knowledge and experience; my thesis committee members, Deva Ramanan and Dinesh Reddy, for checking in on my thesis and providing feedback along the way; collaborators, Xinhe Zhang, with whom I published my first conference paper, and, William Pridgen, Canbo Ye, Anurag Ghosh, and Khiem Vuong for taking the time to debug issues with me about the bus system; and labmates and classmates, Chen Fu and Zherui Zhang, Shun Iwase, Gaurav Pathak, Xindi Wu, Jingxiang Lin, and Silvia Gu for the many fruitful course project, homework, and research discussions.

Lastly, this research was supported in part by the CMU Argo AI Center for Autonomous Vehicle Research, by NSF under Award No 2038612, and by Carnegie Mellon University's Mobility21 National University Transportation Center, which is sponsored by the US Department of Transportation.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Developing fully autonomous driving systems is one of the grand challenges in artificial intelligence and one carrying significant value to modern society. Achieving such a system is expected to have an impact across many aspects of modern living. For one, with 40,000 traffic deaths occurring annually in the USA [44], autonomous driving systems can reduce motor fatalities with their 360° perception and crash avoidance capability. Second, it would release humans from the relatively monotonous task of driving, either for commuting or for package delivery. An example is the trucking industry, which relies on human drivers to transport goods over long distances, a physically taxing and isolating task, causing labor shortages of truck drivers and putting supply chains at risk. An autonomous vehicle could work continuously day and night and improve delivery. Lastly, some 20% of the land in cities is occupied by parking infrastructure. Autonomous vehicles, through robotaxis, can transport people to their destination and be stationed at a more remote location to allow for more green space or affordable housing in urban areas.

Enabling many self-driving vehicles are high-definition (HD) maps. These maps encode much of the driving environment, such as geometric and semantic information about crosswalks, lanes, and driveable areas. These priors aid autonomous vehicles in performing cm-level lane localization [40], planning [27], prediction [5], and perception [69]. Currently, most AVs are geofenced inside areas that are mapped to ensure that safety is held to the highest standard. However, a challenge with HD maps is finding efficient ways to update the map. Lambert et al.[33] analyze the frequency

Figure 1.1: Examples of the different types of changes in the dash cam dataset collected. In (a) and (b) a crosswalk is added where none existed. In (c) and (d) a plain crosswalk is replaced by a zebra crosswalk.

of map changes over the period of 5 months. Subdividing a city map into 30 meter by 30 meter tiles, they estimate that there is a probability of 0.005517% of a vehicle encountering a changed lane geometry or crosswalk and an upper-bound of 0.7% of map tiles changing in a 5-month span. It is estimated that 3.225 trillion miles are driven per year and that there are 276 million vehicles registered in the US[45, 46]. In line with [33]'s derivation, 9.5 billion encounters with map changes occur per year and a single driver is estimated to see a change every 79.5 days, assuming that the average road width is four meters wide:

$$\frac{365 \text{ days}}{3.225 \cdot 10^{12} \text{ miles}} \cdot \frac{1 \text{ mile}}{1609 \text{ m} \cdot 4 \text{ m}} \cdot \frac{900 \text{ m}^2}{1 \text{ tile}} \cdot \frac{1 \text{ tile}}{5.517 \cdot 10^{-5} \text{ changes}} \cdot 276 \cdot 10^6 \text{ cars} \approx 79.5 \text{ days} \tag{1.1}$$

Subject to how many miles are driven per day and how often are on the highway, this estimate is not a trivial frequency. With the statistics applied to Pittsburgh which has an area of 140 km^2, Pittsburgh experiences 7.4 tile changes per day:

$$1.4 \cdot 10^8 \text{ m}^2 \text{ Pittsburgh} \cdot \frac{1 \text{ tile}}{900 \text{ m}^2} \cdot \frac{7 \text{ tile changes}}{1000 \text{ tiles}} \cdot \frac{1}{5 \text{ months}} \cdot \frac{1}{30 \text{ days}} \approx 7.4 \text{ tile changes per day} \tag{1.2}$$

Typically, building an HD map is a laborious process requiring fleets of specialized vehicles and hand annotations for road semantics. Rather than periodically rebuilding

the map in its entirety, it is more cost-effective to find locations where changes have occurred. While it is possible to generate maps on the fly using onboard sensors and deep-learning detection algorithms, it is less accurate and unable to look far ahead and around corners or through vehicles. Moreover, maps are relatively static, so to reduce onboard compute and be informed of the road beyond what is immediately visible, an offline map is often preferred.

The challenge with change detection is that the location and time of changes are unknown. Because road changes can occur everyday, deploying specialized vehicles to perform daily mapping would be costly. Crowd-sourced photographs have recently shown success in mapping and displaying changes in the environment [3, 41, 42]. In this thesis, we leverage a commercial bus that travels daily around the city with camera and GPS sensors. In contrast to [3, 41], which use photos scraped from the internet, images from the bus provide regular crowd-sourced data. Though a commuter bus travels a limited route and has a narrow coverage of the city, the framework requirements are minimal and can work for other service vehicles like garbage trucks and postal cars to expand the map coverage. Given the limited compute available on these vehicles and the need for regular monitoring of the environment, this research aims at efficiently detecting map-relevant changes in noisy and high-traffic environments using vehicle-mounted cameras.

1.1 Scope

HD map change detection is a large problem that this thesis makes a significant step in solving. The scope of this thesis is listed as follows:

- **Degree of automation:** This thesis detects changes that would be relevant to HD maps and reduces the search space in which map labelers have to look for change. Precise change detection is desired, but ultimately, this thesis expects a human to be the final arbiter of change due to the inability of computer vision algorithms to completely understand the subtleties of road markings and change, such as removed crosswalks due to road repaving or degraded crosswalks.
- **Objects of interest:** This thesis primarily detects changes in crosswalks as shown in Figure 1.1 and no changes as shown in Figure 1.2. Crosswalks are

Figure 1.2: Instances of no change in the crosswalk between the reference on the left column and the current image on the right. However, due to vehicle occlusions and subtle viewpoint differences, determining that there is no change from single images alone is challenging.

important for autonomous vehicles because they inform vehicles of areas where pedestrians legally have the right-of-way and are more likely to be present. Crosswalk detection is not an easy task for reasons such as deteriorated paint, different shapes and styles, and occlusions. Crosswalks also often extend across the entire image and are frequently out of the image frame, as seen in Figure 5.6, which poses an additional challenge for computer vision tasks [49]. This thesis focuses on zebra crosswalks even though another type of crosswalk exists, a "plain crosswalk," that is designated by two white lines. However, plain crosswalks can be easily confused with regular lane markings and can lead to many false positives. Therefore, plain crosswalks are left to be studied in the

future.

- **Sensors:** In contrast to past works [33, 35], which use LiDAR sensors and cameras, this thesis proposes to use only camera and GPS data. Cameras and GPS sensors are commonly seen in commercial vehicles and are cost-effective. Relying only on camera data, the framework can more easily be crowd-sourced to more vehicles through inexpensive dashboard cameras and obtain better coverage. Though it is more challenging to estimate depth in the image with an RGB camera, it can be achieved through multi-view geometry. In order to more easily deal with any type of camera, calibrated cameras are assumed, which can improve downstream detection and structure-from-motion.
- **Reference data:** For change detection, past data must exist to compare against. In this thesis, reference data consist of images, in contrast to a map or a 3D model. Except when occlusions occur, images capture 3D scenes quite well. Additionally, this thesis assumes all images are GPS tagged.

1.2 Challenges

This thesis focuses on building computer vision tools to automatically detect crosswalk changes in 3D. The following lists several of the challenges that this thesis addresses:

- **Object Detection:** In order to identify changes in the environment, images must be captured and understood semantically. Abstraction, encoding, and post-processing of raw pixels are thus needed, so that objects with different shapes and illumination can be assigned to the same class.
- **Regular feedback:** Autonomous vehicles are expected to be deployed continuously, and stale maps can lead to invalid path planning and collisions. Changes in the road markings need to be detected as soon as possible, within a day or a week.
- **Noisy and high-traffic environments:** Autonomous vehicles operate in high-traffic areas. Vehicles and pedestrians constantly occlude road labels, as seen in Figure 1.2. A 3D-aware algorithm is needed to distinguish between actual and nuisance changes in the environment map.

1.3 Contribution

The contributions of this thesis are:

- **Object detector:** This thesis uses standard object detectors in 2D that are general to different cameras and mounting setups, for crowd-sourced mapping. Furthermore, different sources of 2D data are evaluated, i.e. public, simulated, and locally collected data. Specifically, an autonomous vehicle simulator is used to generate labelled data of custom street-view objects for real world deployment, and a bus is used to passively collect diverse data and boostrap an object detector.
- **Change detector:** This thesis maps 2D detections onto a ground plane without LiDAR, aligns detections, and makes bird's-eye-view change detection. Given multiple image frames, detections are accumulated to better represent the scene, and consistency checks are made. Additionally, the change detector provides interpretable results as each crosswalk in the scene is marked as verified, removed, or added.
- **Bus platform:** This thesis utilizes a live bus with an edge computer, safety cameras, and cellular internet, which can be used to find scenes of interest and send live images to a server for change detection analysis. The safety system of a bus operates independent of the driver, and thus, the edge computer is accessible to third party groups. Buses provide regular, crowd-sourced data collection.

1.4 Outline

The thesis is organized as follows:

- **Chapter 2** discusses the background knowledge that this thesis builds upon. Some of the established or state of the art methods are used in conducting experiments and are applied in order to perform change detection. The original papers are cited for reference if further details are needed.
- **Chapter 3** discusses related works in using synthetic images for training object detectors and in applying neural networks in detecting changes for different

applications. This chapter shows the variety of changes the community have explored such as detecting structural and tree changes.

- **Chapter 4** discusses the applicability for simulators to provide automatic annotations for custom street-view objects of interest. This chapter uses a simulated vehicle-mounted camera and shows that despite the sim-to-real domain gap, models trained on CARLA-generated annotations perform as well as those trained on 200 real-world images. Training an initial model with simulated data allows for the filtering of data collected on real vehicle-mounted images for real-world data collection and dataset creation.
- **Chapter 5** discusses an extensible platform that enables ordinary vehicles to perform change detection and reduce reliance on manual monitoring and expensive mapping vehicles. It describes a passive solution to collect data and monitor crosswalks with same-day feedback. Specifically, a two-part change detector accepts a stream of data, filters irrelevant images, and performs 3D crosswalk localization and comparisons across time.

Chapter 2

Background

This chapter describes existing tools used in the rest of the thesis. This chapter begins with discussing a hierarchy of scene recognition that can be learned by computers. Then, it discusses simulated ways annotations can be generated to train scene-recognition models as well as the challenge of sim-to-real domain adaptation for real-world deployment. Next, the chapter discusses how maps are commonly represented through birds-eye-view maps and how 3D scenes can be reconstructed from RGB images. Lastly, this chapter discusses the latest successes in using crowd-sourced images to understand the environment and edge computing to generate low latency results.

2.1 Scene Recognition

Many methods exist for scene recognition in 2D images and come in different forms. The general methods include bounding box detection [19, 20, 38, 51, 52], semantic segmentation [10, 55], instance segmentation [26], and panoptic segmentation [30, 32], as seen in Figure 2.1. All methods infer on a single frame. Some have extended the work to perform tracking across multiple frames by detecting the same object in each frame, such as [65, 66]. There are others that combine detections across multiple timestamps using a bayesian occupancy grid [54]. This thesis uses both bounding box detection and instance segmentation.

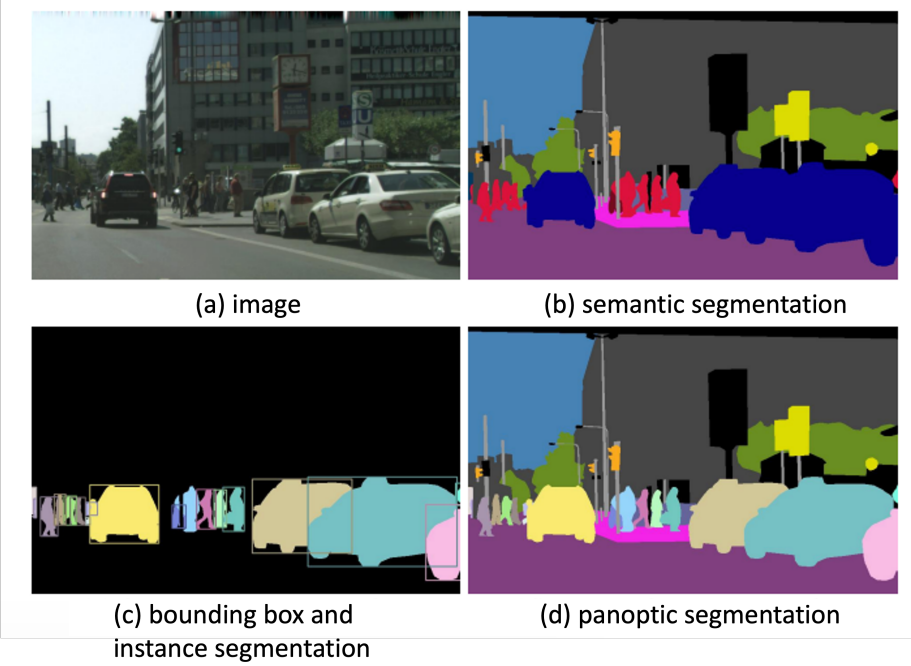


Figure 2.1: An image (a) with its respective annotations for semantic segmentation (b), bounding boxes and instance segmentation (c), and panoptic segmentation (d).[31]

2.1.1 Synthetic Data

The aforementioned models are grounded in deep learning methods, which are data-hungry, needing large amounts of accurately labeled and diverse data. This makes it especially difficult for models to learn objects or events when only a few labeled examples are available. Others have used synthetic data for training neural networks, as seen in the growing number of synthetic datasets, generated from simulators such as SYNTHIA [56], GTA5 [29, 53], and CARLA [14]. Examples of annotated synthetic images are shown in Figure 2.2. The simulators are built upon a game engine like Unreal Engine. Another approach to generate synthetic data is Cut-Paste [17], which pastes object instance cut-outs on random background images. This method is relatively straightforward and easy to implement. However, it is not geometrically aware and is not able to simulate objects in context like a backpack carried by a person. A third type of synthetic data is neural renderings [68], which make use of Generative Adversarial Networks (GANs). However, GANs have an issue of unstable convergence [21], require a pre-existing dataset, and cannot be used when no real

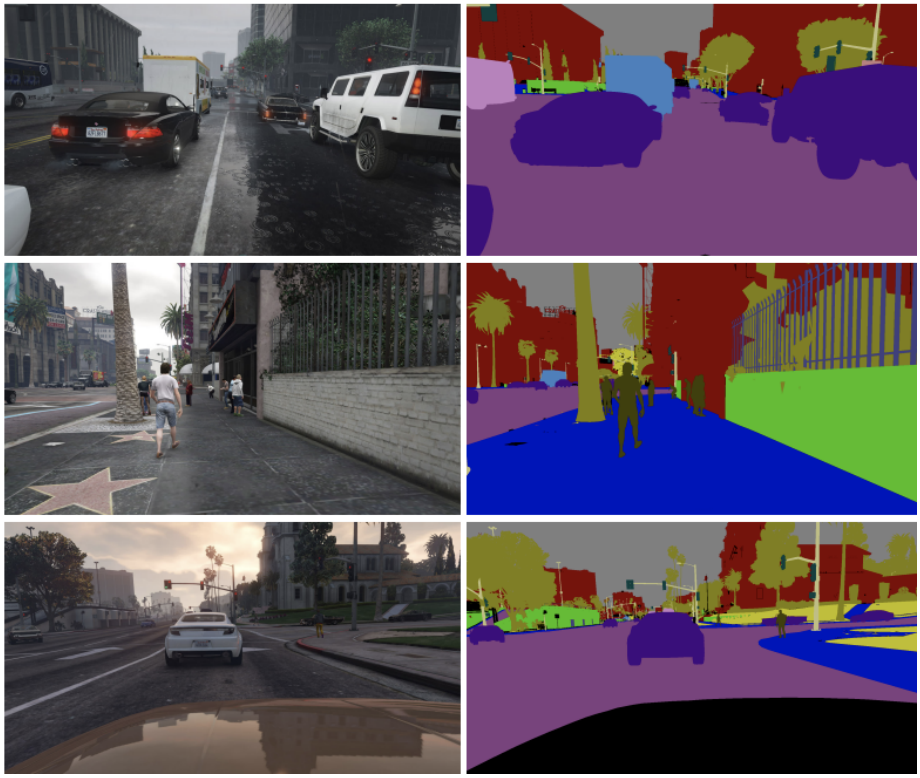


Figure 2.2: Synthetic image data and semantic segmentation annotations created in GTA5 [53].

images are available.

2.1.2 Domain Adaptation

Since there are subtle but important differences between synthetic and real-world data in color style, texture, and appearance, training with synthetic images requires a domain adaptation strategy to overcome the domain gap. Past works have analyzed different methods to maximize the performance of domain adaptation from synthetic-to-real object detection. For example, [63] uses a domain randomization strategy, where they perturb the environment in non-photorealistic ways, adding "flying distractors," random background images, lighting, camera positioning, and textures, while Cut-Paste does something similar by pasting object instances on as many backgrounds as possible to provide enough variety for the model to learn from.

2.2 Birds-Eye-View Maps

For autonomous vehicles, a popular representation of the world is through a bird's-eye-view (BEV) map. Because much of the information for navigation is confined to the ground, BEV maps compactly capture the spatial configuration of the scene. Such is the case for road markings that lie on the ground. This is in contrast to the front-view image of the scene captured by a vehicle-mounted camera, where the scale of objects is proportional to the distance from the camera. Several works have achieved this using inverse perspective mapping [18, 50, 71] or by ray-casting image pixels to a ground surface triangle mesh [33]. This thesis uses an inverse perspective mapping approach and adapts it to generate BEV detections.

2.3 Scene Reconstruction

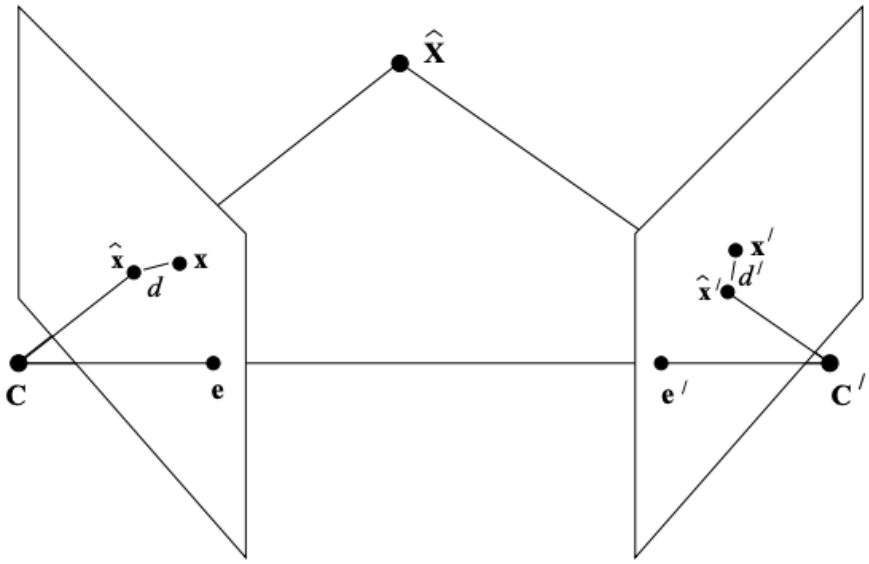


Figure 2.3: Minimization of the reprojection error between the 2D correspondences and the projected 3D point [24]

Images are projective transformations of 3D scenes into 2D image planes. Using multiple images and multiple views of the scene, the 3D scene can be reconstructed using structure-from-motion algorithms. COLMAP [60, 61] is one such software to

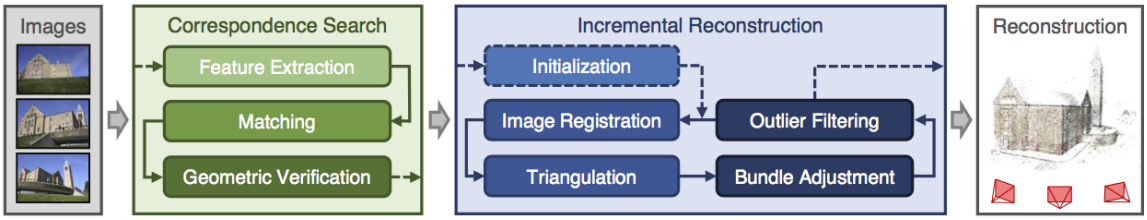


Figure 2.4: COLMAP's incremental Structure-from-Motion pipeline [60, 61]

reconstruct and estimate poses of the images taken. The camera poses are estimated by matching 2D key points between images and calculating the rigid-body transformation between the images. COLMAP uses SIFT features [39] by default, which are hand-crafted features. Recent research has used learned features and learned matching between images [13, 58, 59]. The scene is reconstructed in the form of 3D point clouds by triangulating points between different images, and the reconstruction error is the distance between the 2D key point and the 2D projection as shown in Figure 2.3. This error is further optimized using bundle adjustment. The complete process is shown in Figure 2.4.

2.4 Crowd-sourcing

Crowd-sourced photographs have recently shown success in mapping and displaying changes in the environment [3, 41, 42]. [41] illustrates changes across billboards in Times Square and reconstructs the scene from noisy images taken from internet collection of photos. Both [42] and [3] use images taken by the public in order to reconstruct 3D point clouds of the scene at large scales. These advances are necessary in mapping large areas like road networks and changes across time. The benefit is that many of the photographs are being captured for other purposes such as phototourism or providing visual evidence for accidents, in the case of dash-cams. Groups like Karta, GoogleStreetView, and Mapillary already have been using streetview images to build a database to improve mapping. This thesis extends the work and seeks to extract changes across time with footage from vehicle-mounted cameras and bus-mounted security cameras.

2.5 Edge-Cloud Computing

Edge computing is the deployment of computing resources at the location where data is produced. Because of bandwidth, time, and storage constraints, computing cannot be done on the cloud with low-latency results. However, since smart devices deployed in the world lack the necessary computing resources to often run complicated vision algorithms, the heaviest computation is often offloaded to the cloud. One such division of labor is such that the on-device computing acts as a filter for live data and sends only relevant information back to the server for fine-grained analysis and can devote more time and resources to the filtered data. [22] implements real-time processing of image capturing and sensing on wearable devices. The system integrates a token-based flow control mechanism that gracefully maintains low latency even during large compute volumes. [11, 70] apply this framework to map traffic signs, traffic cones, and potholes. This thesis uses an edge-computer located on a bus to filter out irrelevant images that have low chances of having a crosswalk and uses the [22]'s system for sending the data to a server to analyze images with crosswalks.

Chapter 3

Related Work

3.1 Online Map Generation

Some recent works attempt to use deep learning networks to generate maps or road semantics on the fly [34, 54]. With online map generation, change detection would simply become a map comparison between two different time points.

[34, 54] both use surround view cameras and predict a bird's-eye view (BEV) of the road semantics directly. This is in contrast to the indirect approach by first learning features in the image space and then transforming them into BEV space through an inverse perspective mapping [2]. Although directly learning the map is ideal, training data for this are particularly limited. At the moment, NuScenes [8] and Argoverse [9] are the only datasets that contain 3D annotations of drivable areas, crosswalks, and lanes. Because these datasets are only labelled for a few cities, the map generation results are difficult to generalize to other locations and environments.

3.2 Change Detection

3.2.1 2D Change Detection

Various groups tackle change detection through a learned approach. [57] uses aligned omnidirectional images of a scene to perform change detection before and after a natural disaster. It uses superpixel segmentation and uses the features encoded

by a convolutional neural network for change comparison. [4] uses deconvolutional networks to perform pixel-wise change detection, where they train a model that detects relevant structural changes such as construction, building demolition and traffic signs between coarsely registered image pairs, while ignoring irrelevant changes such as seasonal and lighting variations and dynamic object changes like vehicles and pedestrians. They rely on monocular cameras, GPS, and inertial odometry sensors to perform SLAM and densely reconstruct the scene. Aligned image pairs are generated by projecting the dense reconstruction into another camera. One drawback of [57] and [4] is that they only perform change detection at the image level. The methods would perform poorly if a relevant scene change is obstructed from the viewpoint of the image. Furthermore, if multiple images are taken of the same scene, there are no consistency guarantees between images if changes are detected differently.

[6] detect and catalogue tree changes in a city and use both street view and aerial images from 2006 to 2016. While the approach of taking RGB images from aerial or satellite cameras provides a bird's-eye view of the scene with a larger field of view of the ground, the precision is coarser and scans of the scene are at a lower frequency, which can lead to a significant lag between a change and a detection. Additionally, the farther the aerial camera is from the object, the higher the likelihood that vehicles or buildings will occlude the map environment, which can further delay a detection.

3.2.2 3D Change Detection

[41] illustrates changes across billboards in Times Square and performs accurate structure-from-motion and grouping of 3D points in time, based on temporal and spatial clustering. They reconstruct the scene from noisy images taken from a large, crowd-sourced internet collection of photos.

[33] uses sensor inputs as well as current map information to predict if change has occurred. They use LiDAR and RGB data and experiment with ego-view and BEV viewpoints and make a binary classification for a given drive. The drawback of this approach is that it relies on an expensive autonomous vehicle's sensor suite for change detection, which would be difficult to crowd-source, and assumes precise localization of the vehicle. Another drawback is that the method does not provide localization of the changes, except for activation heat maps from the neural network.

Change detection using 3D models has also been explored by [47, 48]. Such 3D methods typically require a model as a prior, which is difficult to generate and error-prone. Any error in 3D model building will affect downstream performance of the change detection algorithm.

3.2.3 Reporting Change Detections on Maps

Crack formation on roads is an important change that often needs to be reported. In [64], the authors propose a super-pixel-based over-segmentation method to detect cracks on the road from the single image obtained from a windshield-mounted camera. Descriptors are later extracted for classification. While detections are made at the image level, severity of crack formations is reported along the road network of a general map.

Similarly, [11] addresses change detection in terms of detecting hazards like traffic cones and potholes on the road. The system reports each detection on a "LiveMap" where detections are reported in real time as a vehicle sees the objects. However, one issue is that each image corresponds to a detection and is plotted on the map. There is not an intuitive way to cluster them without tuning distance parameters for each category and adjusting for different frame rates.

3.3 Summary

In summary, this thesis adopts aspects of each of the above sections to perform online change detection of crosswalks in 3D. With respect to generating online maps, this thesis is similar in that it generates a top down map of crosswalks from a single pass of a vehicle through an area. The difference is that generating online maps is not learned directly because labelled data at the image level is far easier to access, which means a better ability to adapt to different vehicles, cameras, and environments. With respect to 2D change detection, this thesis similarly focuses only on camera images and extracts features from images, except this thesis produces features as instance segmentations as opposed to only pixel-wise classifications. This approach adds more meaning and interpretability of detected changes. Furthermore, this thesis hopes to translate these features into 3D in order to aggregate and summarize results

across frames for a more holistic output when changes are reported to an end user. With respect to works in 3D change detection, this thesis aligns closely with [41] where streets are represented as planes. However, the application is different and this thesis's approach is more proactive in collecting images and identifying changes of interest as opposed to general scene changes. With regards to [33], this thesis does not use LiDAR and estimates image poses using multi-view geometry, which makes the task more challenging but enables crowdsourcing.

Chapter 4

CARLA Simulated Data for Object Detection

This section of the thesis builds upon past work on synthetic image generation and domain randomization, by using the open-source CARLA simulator, with its active community of developers, to create a diverse dataset of custom objects[7]. Simulators like CARLA have the advantage of preserving geometric consistency and providing a large variety of domains. Developed object detection models in this chapter can be used in Chapter 5, by filtering for relevant image queries from the bus and using them to bootstrap models for further fine-tuning.

4.1 Approach

Following the strategy of other domain randomization methods, we try to create a variety of random renderings with the CARLA simulator, with the idea that the real world will be interpreted as part of the synthetic data distribution. Randomization properties include lighting, precipitation, and actor parameters that can be changed to allow for different appearances of our objects. We evaluate our crosswalk model on the Mapillary Vistas Dataset (MVD) [43] and our fire hydrant model on the MVD dataset, MS COCO dataset [37], and a locally collected Pittsburgh dataset. MVD [43] is a large-scale street-level dataset of 25K images with instance-level annotations of 100 object categories. MS COCO [37] has 200K images and 80 object categories.

The Pittsburgh dataset has 1,358 images of Pittsburgh fire hydrants. The reason we evaluate the fire hydrant model on three different datasets is that we sought to evaluate the generality of our trained models on different images collected from different environments and with different methodologies, since models trained on one dataset generally will perform the best when evaluated on the same dataset. However, to explore the effectiveness in detecting local fire hydrants, some of our 3D models of fire hydrants are generated from images of Pittsburgh fire hydrants as described in Section 4.1.1, which biases the CARLA synthetic dataset but also makes the 3D models more realistic. To compare our method with another synthetic data approach, we use Cut-Paste synthetic images as a baseline.

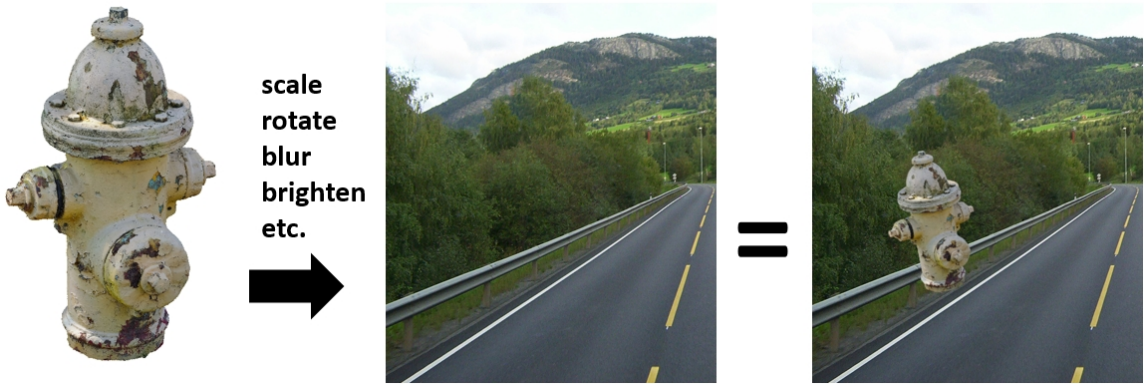


Figure 4.1: Example of a fire hydrant pasted on a background image using the Cut-Paste method.

4.1.1 Baseline: Cut-Paste

Our Cut-Paste method is based on [17]. We started with their code and made it compatible with Python 3.x, modified it to avoid some edge artifacts, and added a 3D reconstruction method to speed up the creation of cutout examples. Usually one takes several pictures from different vantage points of the object and then cuts the object out from each image. This can be tedious, and it does not scale well. Instead, we took many images of a fire hydrant and used COLMAP [60] to create a 3D model of it. After cleaning the model we took virtual snapshots of the fire hydrant on a white background to produce the cutouts. Fig. 4.1 shows such a snapshot pasted onto

some background. Before pasting, the snapshot was manipulated by scaling, rotating, blurring, and changing contrast and intensity. We produced additional varieties of fire hydrants by changing texture and shading. Our code and detailed instructions are publicly available[1].

4.1.2 CARLA

The simulator we use is CARLA [14], an open source simulator for urban driving built on the Unreal Engine rendering platform. CARLA was developed to support training, experimenting, and validation of autonomous driving models, including perception and control, and includes 8 urban layouts and a flexible setup of sensor suites that can be used to collect RGB images and ground-truth semantic segmentations. A wide range of environmental conditions can be specified, including 9 independent weather parameters and 2 sun angles.

We constructed large-scale and diverse synthetic datasets using publicly available 3D CAD models of fire hydrants and crosswalks and our own set of 3D reconstructed models of fire hydrants. We specifically chose objects that are not in CARLA's default object semantic segmentation labels to show the ease of creating annotated datasets of new objects. We manually placed each 3D model into the CARLA map and duplicated the same model in different locations. We then used Unreal Engine's ray tracing to generate semantic segmentations of the captured image. Photos were taken from the perspective of a virtual vehicle-mounted camera and were saved whenever the area of the object met a certain area threshold in the frame. Because we make the assumption that the objects have minimal occlusion and do not overlap, bounding box annotations were then created by performing a closing morphological operation on the segmentation image and by taking the minimum and maximum positions of each disconnected object segmentation. Detailed instructions, code, and data are made publicly available[2][3][4].

[1] https://github.com/chrmertz/synth_train_data
[2] https://www.github.com/xinhez/simulation-for-detection
[3] https://www.kaggle.com/xinhez/synthetic-fire-hydrants
[4] https://www.kaggle.com/buvision/synthetic-crosswalks

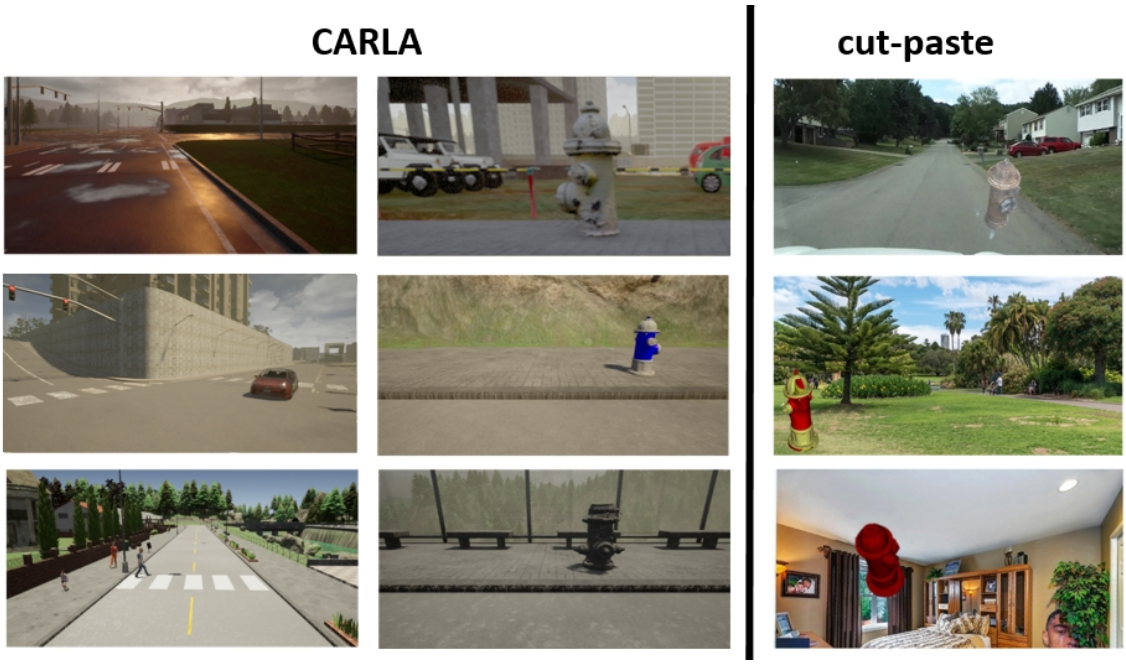


Figure 4.2: Synthetic image examples. CARLA-generated images are shown in the first two columns, where we illustrate domain randomization of fire hydrants and crosswalks from camera positioning, weather, vehicle-pedestrian obstacles, and styles. In the third column, we show Cut-Paste images for fire hydrants with different poses, styles, and background images.

4.1.3 Domain Randomization

CARLA provides a variety of parameters for generating diverse images, as shown in Fig. 4.2.

- Content Variation: CARLA provides eight pre-built maps with different buildings and environments. Not only is each map unique, but different locations within each map provide interesting and valuable variation to allow for at least 20 object placements. Furthermore, random actors can be placed in the maps using some of the default blueprints. As of this writing, 30 vehicle and 26 pedestrian blueprints exist. These actors can add more uniqueness to each snapshot of a particular object and can also provide occlusion of the object, forcing the model to learn how to ignore these distractors. Also, CARLA makes it easy to select all target objects and modify the material style in bulk, such as going from white to yellow crosswalks.

- Viewpoint Variation: CARLA allows users to change the positioning and orientation (6 DoF) of the virtual camera within the vehicle. These parameters are initialized at the start of each simulation. Driving varies x, y, and heading of the vehicle and thereby of the virtual camera.
- Weather Variation: For each map, the weather parameters are continuous random variables with ranges as follows: cloudiness $\in$ [0, 100], precipitation deposits (i.e., rain puddles) $\in$ [0, 100], sun altitude angle $\in$ [-90, 90], sun azimuth angle $\in$ [0, 360], precipitation (i.e., falling rain) $\in$ [0, 100], wind intensity $\in$ [0, 100], fog density $\in$ [0, 180], fog distance $\in$ [0, 180], and road wetness $\in$ [0, 100]. For every frame of the simulation, we increment one parameter with a step size of 25, and if the value exceeds the parameter's upper bound, we set the parameter to the modulus after dividing by the parameter's upper bound.

4.1.4 Detection Network

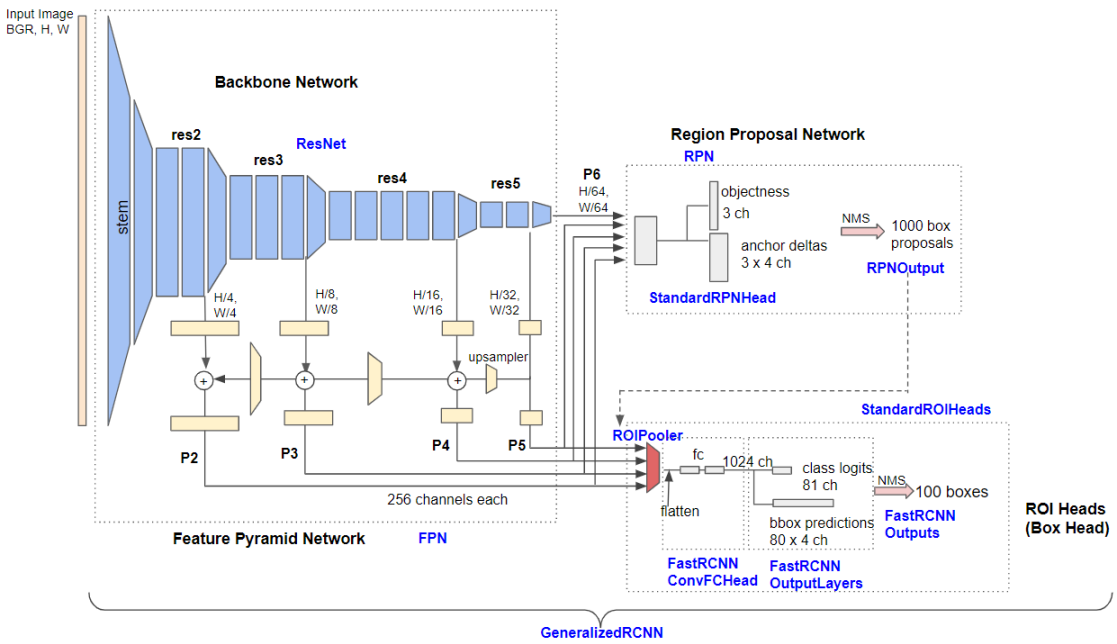


Figure 4.3: Detectron2 implementation of Faster RCNN with a feature pyramid network as the backbone [28]

We use the Detectron2 [67] library, which provides high-quality implementations

of state-of-the-art object detection algorithms, including Faster R-CNN [52], as seen in Figure 4.3, which is used here. In this work, we used the ResNet network with a depth of 101 layers [25] combined with a Feature Pyramid Network (FPN) [36], which extracts features of the input image at different scales. The decoder of the network consists of output heads for bounding-box recognition, i.e., classification and regression, with a loss function of $L = L_{class} + L_{bbox}$. L_{bbox} uses an L_1 loss, and L_{class} uses a cross-entropy loss. This model is chosen because Faster R-CNN is one of hte most used detectors, and this architecture showed the highest performance among Detectron2's models. The crosswalk detector is trained for 20 epochs on the synthetic data and has a learning rate of 0.001, while the fire hydrant detector is trained for 200 epochs with a learning rate of 0.005.

4.2 Results and Analysis

We benchmark our methodology on the MVD dataset [43] for the tasks of fire hydrant and crosswalk detection. And for the fire hydrant, we additionally evaluate on the MS COCO dataset[37] and a locally collected Pittsburgh dataset.

We filter the MVD and COCO datasets to search for our target objects. Because the MVD dataset images range from eight to forty megapixels in size, we resize and crop the images for memory purposes so that the shortest side is 600 pixels. Because some objects seen at high resolution become unrecognizable after resizing to a low resolution, for crosswalks, we remove bounding boxes that are below 10 pixels in height. For fire hydrants, we cropped the original images to 800 by 600 resolution. In Tables 4.1 and 4.2, we specify the instance counts from each dataset used to train and test models for crosswalk and fire hydrant detectors, respectively.

4.2.1 Synthetic and Real World Training Image Comparison

We evaluate our detectors on real world images. For crosswalks, because very few small instances exist, we remove all small instances in the training and test sets, and balance the number of medium and large instances in CARLA and low-shot experiments as indicated in Table 4.1. The different crosswalk models we train are

Table 4.1: Instance distribution of **crosswalks** for the CARLA, MVD low shot, and MVD full and test datasets, where the column headings, S (pixel area$\leq 32^2$), M (32^2 <pixel area $\leq 96^2$) and L(96^2 <pixel area), refer to the size of the object instances, according to COCO standards, respectively and low shot refers to anything other than the full dataset. The superscript *st* indicates that the number of instances at each size either uses all available images, except for small, which is excluded due to the limited real-world annotations.

Dataset	S	M	L	total
MVD^{st}_{test}	0	348	282	630
$\text{CARLA}^{st}_{train}$	0	10,000	10,000	20,000
$\text{MVD20}^{st}_{train}$	0	10	10	20
$\text{MVD120}^{st}_{train}$	0	60	60	120
$\text{MVD600}^{st}_{train}$	0	300	300	600
$\text{MVD2000}^{st}_{train}$	0	1000	1000	2000
$\text{MVDFull}^{st}_{train}$	0	2817	2305	5122

indicated below:

- CARLA: trained with domain-randomized synthetic data.
- N-Shot: trained with N randomly sampled, real-world images.
- CARLA + N: trained with domain-randomized synthetic data and N real-world images.
- Full: trained with all available real-world images.
- CARLA + Full: trained with CARLA and all available real-world images.

For fire hydrants, we noticed that the MVD and MS COCO datasets are unbalanced in different ways (see Table 4.2). MVD has more small and medium examples, whereas MS COCO has more large examples in their standard dataset. We will later see that this has a significant effect on the evaluation. We therefore created additional "medium" training and testing sets. For medium training, we balance the set so that the numbers of small, medium, and large instances are in a 1:2:1 ratio. For medium testing, there are only medium-sized instances. We perform these balances to accurately assess the performance of these models across the different datasets. In addition, we also trained the same model on synthetic data generated by the Cut-Paste method and CARLA without domain randomization. The different models we train are indicated below:

Table 4.2: Instance distribution of **fire hydrants** for the MVD, MS COCO, Pittsburgh, CARLA, and Cut-Paste datasets for standard (a) and medium-sized (b) object training and testing, denoted by the superscript *st* and M, respectively. Standard training uses all available instances of the object for training, while medium-sized training uses twice as more medium-sized instances than other sizes. The column headings S, M, and L refer to small, medium, and large object instances, respectively. The / in (b) is to indicate separate datasets.

(a) Standard training and testing

	S	M	L	total
MVD^{st}_{test}	66	85	28	179
COCO^{st}_{test}	126	172	352	650
Pitt^{st}_{test}	0	337	1021	1358
MVD^{st}_{train}	657	738	242	1637
COCO^{st}_{train}	268	329	719	1316
$\text{CARLA}^{st}_{train}$	10000	10000	10000	30000
$\text{Cut-Paste}^{st}_{train}$	10000	10000	10000	30000

(b) Medium-sized object training and testing

	S	M	L	total
COCO^{M}_{test} / MVD^{M}_{test} / Pitt^{M}_{test}	0	200	0	200
CARLA^{M}_{train} / $\text{Cut-Paste}^{M}_{train}$	5000	10000	5000	20000
$\text{COCO20}^{M}_{train}$ / MVD20^{M}_{train}	5	10	5	20
$\text{COCO120}^{M}_{train}$ / $\text{MVD120}^{M}_{train}$	30	60	30	120
$\text{COCO600}^{M}_{train}$ / $\text{MVD600}^{M}_{train}$	150	300	150	600

- Cut-Paste: trained with Cut-Paste method-generated synthetic data.
- NON-DR-CARLA: trained with synthetic data without domain randomization.
- CARLA: trained with domain-randomized synthetic data.
- N-Shot: trained with N randomly sampled, real-world images.
- CARLA + N: trained with domain-randomized synthetic data and N real-world images.
- Full: trained with all available real-world images.

The crosswalk model was pretrained on the MS COCO dataset for 37 epochs. For the fire hydrant model, because the MS COCO dataset contained fire hydrants, we used a pretrained model with ResNet 101 weights trained on ImageNet data [12]. Pretrained

models were provided by the Detectron2 library [67]. We used the standard MS COCO evaluation metrics as described in [1] and recorded the mean average precision (AP), averaged for intersection over union (IoU) $\in [0.5 : 0.05 : 0.95]$, AP_{50} (IoU = 0.5), AP_{75} (IoU = 0.75), AP_{small} for small objects (pixel area $\leq 32^2$), AP_{medium} for medium objects (32^2 < pixel area $\leq 96^2$), AP_{large} for large objects (96^2 < pixel area). For crosswalks, the quantitative results are shown in Table 4.3 with qualitative results shown in Fig. 4.5. We show the precision-recall curve in Fig. 4.4. For fire hydrants, the quantitative and precision-recall curve results for standard test sets are shown in Table 4.4 and Fig. 4.6, respectively. The same for medium test sets are shown in Table 4.5 and Fig. 4.7, respectively, where the precision-recall curve is for detections of the models on the Pittsburgh dataset. We show the qualitative results of the CARLA-trained model in Fig. 4.8. In the discussions below we will consider a few % (absolute) differences in AP as not significant.

4.2.2 Crosswalks

Crosswalks are more challenging objects than fire hydrants because they are flat objects mostly viewed at an oblique angle, which results in large perspective distortions. Table 4.3 shows that the various APs of the MVD (full) trained model are more than twice the value of those of the CARLA-trained model. One trend that is similar to the fire hydrants is that the CARLA-trained model performs relatively better for large-sized objects.

The comparison with various levels of low- to medium-shot models shows tendencies similar to those seen with fire hydrants. The CARLA model performs better than a 20-shot model and roughly similarly to a 120-shot model. Adding the CARLA images to those of the low- to medium-shot generally increases the performance but has little effect when added to those models with many real images.

Correct, missed, and false detections of crosswalks by the CARLA-trained model can be seen in Fig. 4.5. The model is able to find crosswalks in different states of repair, with various illuminations, and from different perspectives. Many of the missed crosswalks are from the perspective of a pedestrian standing on the sidewalk. This perspective is underrepresented in the CARLA dataset because we took the images from a virtual camera mounted on a vehicle. A frequently occurring false detection is

Table 4.3: Average Precision (%) table for crosswalk detection on the MVD test set.

Model	AP	AP_{50}	AP_{75}	AP_M	AP_L
$CARLA^{st}_{train}$	11.7	25.4	10.6	4.8	21.4
$MVD20^{st}_{train}$	1.2	3.8	0.3	0.2	2.3
$CARLA^{st}_{train}$ + $MVD20^{st}_{train}$	16.2	33.8	16.3	7.3	27.7
$MVD120^{st}_{train}$	14.4	35.5	8.4	5.9	25.0
$CARLA^{st}_{train}$ + $MVD120^{st}_{train}$	19.1	39.5	18.2	9.0	32.2
$MVD600^{st}_{train}$	25.9	53.6	21.6	13.2	42.1
$CARLA^{st}_{train}$ + $MVD600^{st}_{train}$	26.3	52.6	23.6	15.1	40.4
$MVD2000^{st}_{train}$	34.4	63.6	32.2	19.6	52.0
$CARLA^{st}_{train}$ + $MVD2000^{st}_{train}$	35.5	63.9	34.2	21.5	53.1
$MVDFull^{st}_{train}$	41.7	71.3	43.4	27.6	58.1
$CARLA^{st}_{train}$ + $MVDFull^{st}_{train}$	40.1	67.9	39.7	24.1	58.7

that of a car. Cars, like people in the case of fire hydrants, might be underrepresented and not well simulated in CARLA. One category that is not simulated at all in CARLA is snow. A mixture of snow and slush is prone to cause a false detection; one example is shown in the bottom left corner under the "false" column in Fig. 4.5.

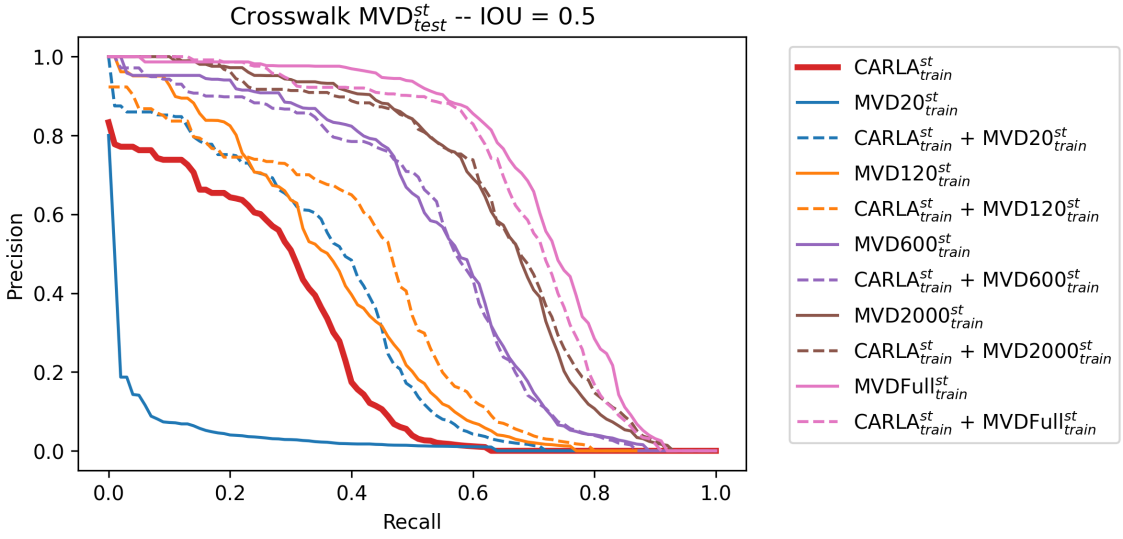


Figure 4.4: Precision recall curves for crosswalk models evaluated on the MVD dataset.

Figure 4.5: Qualitative results from crosswalk predictions: CARLA-trained model tested on MVD dataset. Shown are correct, missed and false detections.

4.2.3 Fire Hydrants

In the following discussion, we notice two intermixed tendencies. One is that the models trained on synthetic data perform best for large-sized objects. The second is that any model performs worse when tested outside its primary domain.

The fire hydrant 3D models we used for the synthetic training data mimicked Pittsburgh fire hydrants. It is therefore expected that the CARLA and Cut-Paste-trained models perform well on the Pittsburgh dataset. As shown in Table 4.4c, they are indeed competitive with the standard models trained on MVD or COCO for large objects (AP_L). They are worse for medium-sized objects (AP_M). In contrast, when the synthetic-trained models are tested in different domains (MVD or COCO standard test sets), they generally perform worse than the standard models trained on the corresponding MVD or COCO. The CARLA performance is especially bad for small-sized objects. This can be seen in Table 4.4a and 4.4b. A similar performance drop when changing domains can be observed when comparing MVD and COCO datasets with each other. COCO-trained models perform best when tested on the COCO test set, but worse when tested on MVD, and vice versa. This emphasizes the importance of the domain that is tested. MVD are all images taken on streets, whereas COCO are more general. Another significant difference is that COCO has many more large objects and MVD has more medium- and small-sized objects. This explains why the COCO-trained model is good at detecting large objects and the MVD-trained model is good at detecting small objects.

To remove the bias in object size, we created the "medium" test and training sets as described above. In Table 4.5, the differences between the MVD and COCO results are less pronounced than before. Such curated datasets are more appropriate for a detailed study. The first question we wanted to investigate was: at what point does the CARLA model perform similarly to the MVD or COCO models? The numbers in Table 4.5 indicate that MVD or COCO models trained with 120 images perform similarly to the CARLA model. The second thing we wanted to find out is how much a low-shot model improves when adding CARLA images. In Table 4.5, the COCO models get mostly better and sometimes stay the same when adding CARLA data. For MVD models, adding CARLA images gives inconsistent results.

Fig. 4.8 shows correct, missed, and false detections. The correctly detected fire

Table 4.4: Average Precision (%) table for fire hydrant detection on the COCO, MVD, and Pittsburgh standard test sets.

(a) COCO standard test set

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Cut-Paste$^{st}_{train}$	30.5	47.8	33.4	6.3	20.5	44.8
CARLA$^{st}_{train}$	34.9	55.6	39.4	2.8	26.3	50.7
MVD$^{st}_{train}$	40.2	71.0	43.9	19.9	47.4	43.9
COCO$^{st}_{train}$	60.2	83.0	69.1	25.9	56.3	74.1

(b) MVD standard test set

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Cut-Paste$^{st}_{train}$	15.0	28.2	12.1	4.8	19.1	35.4
CARLA$^{st}_{train}$	25.2	44.8	26.4	6.9	32.5	49.2
MVD$^{st}_{train}$	46.4	81.9	49.7	29.7	55.6	58.0
COCO$^{st}_{train}$	39.5	66.9	44.9	16.8	49.3	62.8

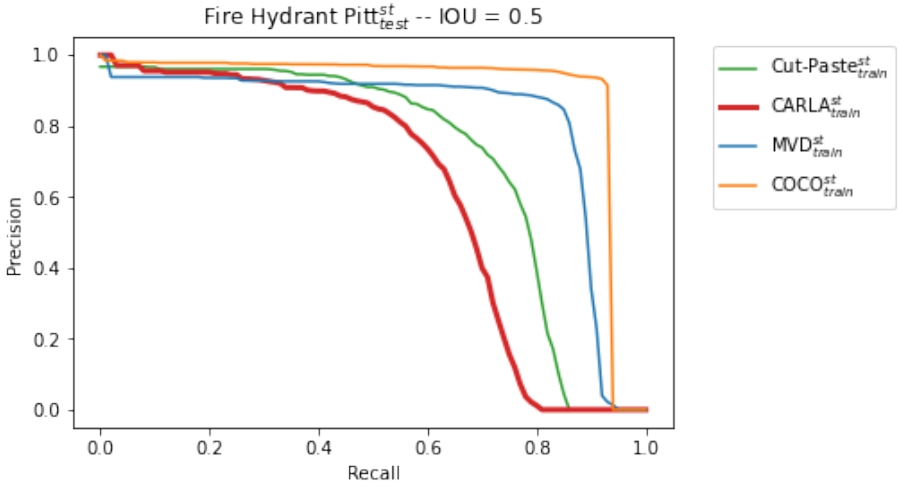
(c) Pittsburgh standard test set

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Cut-Paste$^{st}_{train}$	24.8	64.4	13.4	-	14.0	32.5
CARLA$^{st}_{train}$	21.5	61.4	7.6	-	11.3	32.5
MVD$^{st}_{train}$	28.3	75.7	13.2	-	27.8	33.4
COCO$^{st}_{train}$	34.1	87.9	16.4	-	23.1	39.3

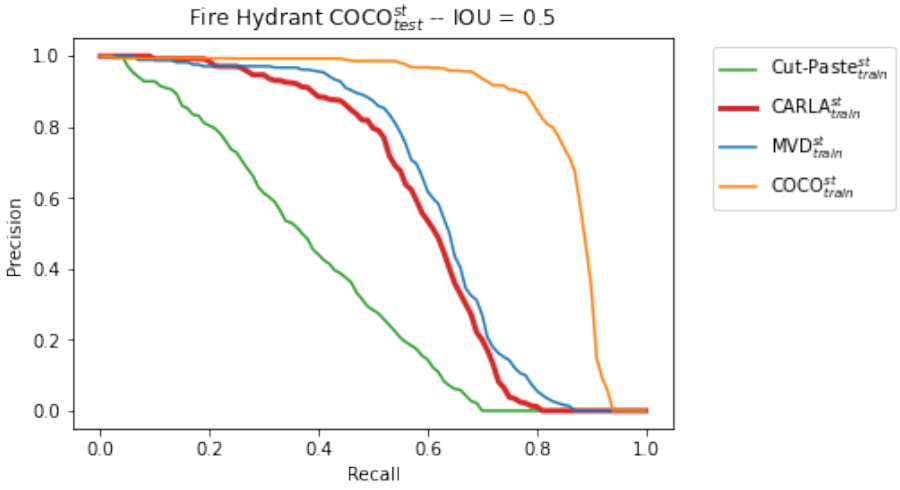
hydrants are in a variety of environments, including snow and poor illumination. Some of the missed fire hydrants have unusual shapes, are surrounded by clutter, or are blurred. Among the false detections, people are the most common objects. This points to the problem that in CARLA people are not very well simulated.

Table 4.5: Average Precision (%) table for fire hydrant detection on the MVD, MS COCO, and Pittsburgh medium test set.

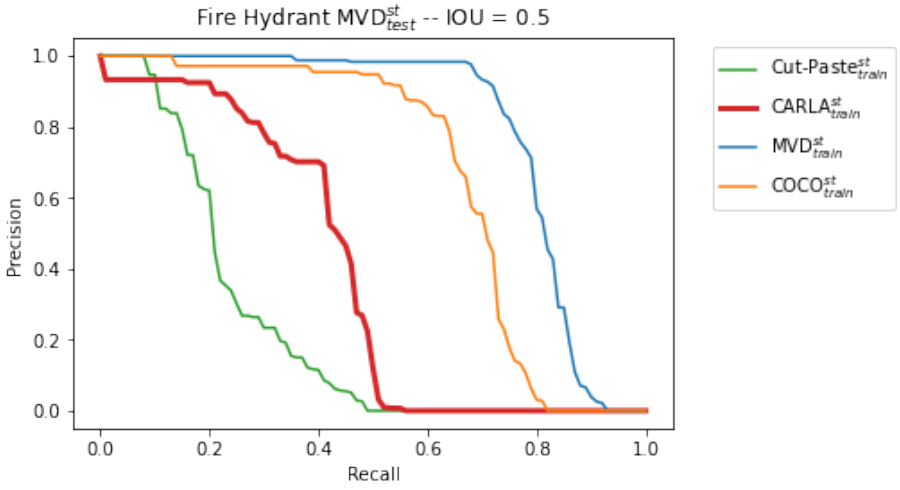
	AP_M^{COCO}	AP_M^{MVD}	AP_M^{PITT}
Cut-Paste$^M_{train}$	18.7	14.8	10.7
MVD20$^M_{train}$	32.6	32.3	20.5
MVD120$^M_{train}$	39.2	39.6	23.5
MVD600$^M_{train}$	46.2	52.5	26.2
COCO20$^M_{train}$	22.9	14.3	18.8
COCO120$^M_{train}$	40.4	34.1	19.8
COCO600$^M_{train}$	51.4	47.6	22.7
CARLA$^M_{train}$	28.2	37.5	16.5
CARLA$^M_{train}$ + MVD20$^M_{train}$	29.5	39.9	20.2
CARLA$^M_{train}$ + MVD120$^M_{train}$	39.7	49.2	30.3
CARLA$^M_{train}$ + MVD600$^M_{train}$	40.8	54.1	28.3
CARLA$^M_{train}$ + COCO20$^M_{train}$	36.9	36.1	20.9
CARLA$^M_{train}$ + COCO120$^M_{train}$	41.8	44.8	24.0
CARLA$^M_{train}$ + COCO600$^M_{train}$	49.8	48.7	27.1



(a) Pittsburgh

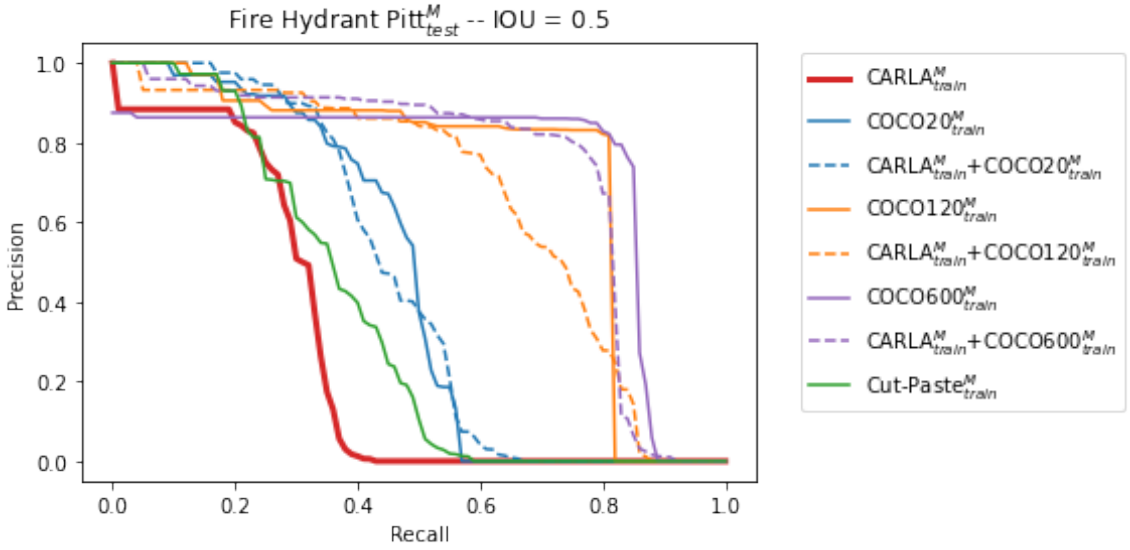


(b) COCO

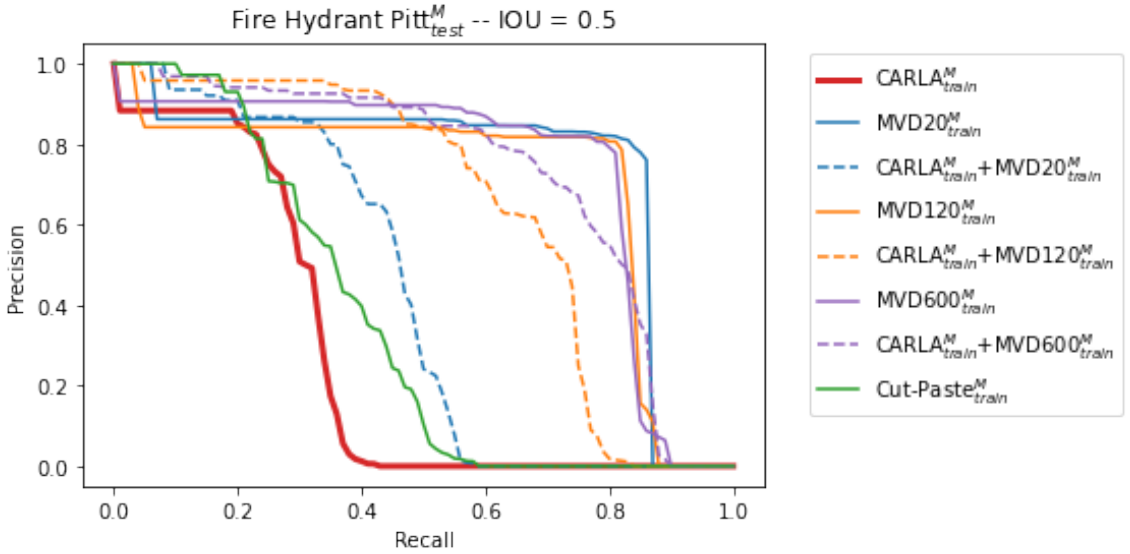


(c) MVD

Figure 4.6: Precision recall curves for fire hydrant standard models evaluated on the COCO, MVD, and Pittsburgh standard test sets.



(a) Fire hydrant medium COCO-trained models



(b) Fire hydrant medium MVD-trained models

Figure 4.7: Precision recall curves for fire hydrant medium models evaluated on the medium Pittsburgh dataset.

Figure 4.8: Qualitative results from fire hydrant predictions: CARLA-trained model tested on all three data sets. Shown are correct, missed and false detections.

4.2.4 Other Observations

We compared the CARLA-generated synthetic dataset with the Cut-Paste [17] strategy. In the Cut-Paste strategy, we take virtual snapshots of 7 3D fire hydrant meshes, while rotating them 360° and randomly placing them on 596 background images for training. As shown in Tables 4.4 and 4.5, CARLA performs better than Cut-Paste in the medium test case and the standard test case in COCO and MVD. However, Cut-Paste performs better than CARLA on the Pittsburgh dataset, likely because the 3D fire hydrant meshes used in Cut-Paste were from Pittsburgh fire hydrants.

To test the importance of domain randomization, we tested one model that was trained on CARLA images without domain randomization. The AP was only a third of the AP from a CARLA model with domain randomization. This clearly shows how important domain randomization is to decrease the sim-to-real gap.

On a standard desktop PC with one Nvidia graphics card it took CARLA about 4 hours to produce 1000 training images. The Cut-Paste method is much faster; it does not require a GPU and takes about 20 minutes to create 1,000 training images.

4.3 Summary

In this chapter, we show that synthetic data generated in CARLA can be an effective way to generate data for rare, novel, or ignored objects. It works best if the model is well adapted to the tested domain while at the same time domain randomization is applied. We further observe that the best performance is on large-sized objects. We apply our method to two object classes in MVD to show the ease of generating novel classes in CARLA and our method's wide applicability for augmenting existing datasets and training detectors with zero to few-shot real image labels. Our method of using CARLA is a promising tool for the expanding need of data collection and is especially helpful for improving the performance of models trained on small datasets.

There are several avenues for future steps. One is to create more realistic simulations in CARLA by improving the simulations of vehicles and people and the incorporation of snow. Better renderings can decrease the domain gap between synthetic and real world images and lead to fewer false detections. Another avenue is to improve the network architecture and training method to better adapt to real

world images. For example, an adversarial loss could be incorporated in training to penalize significant differences between predicting on real and synthetic images. Another example is to take advantage of the abundant real-world images and use self-supervision like constrastive learning to train a better backbone.

Applications of this work include monitoring of city-specific objects that are poorly represented in public datasets, such as a new crosswalk style or novel traffic equipment. In the case of autonomous driving, it would be important to be aware of any additions or removals of these rare objects that could change the meaning of traffic rules, and synthetic data can allow engineers to train detection models preemptively.

Chapter 5

Bus Platform for Crosswalk Detection and Monitoring

This chapter uses the experience in training 2D object detectors in Chapter 4 to detect crosswalk changes across time with vehicle-mounted cameras. In order to use crosswalk detections for change prediction, the crosswalk detections need to be spatially aligned across different times. Because of challenges in performing change detection from the perspective view, this chapter explores methods to map crosswalk detections out of the image plane and onto a global 3D ground plane to perform crosswalk change detection from the bird's-eye view. Not only does working in a ground plane avoid the need to have aligned images, it allows for detections in sequential dash cam images to be composited into a single coordinate frame to better represent the scene. An example is when one image captures the front crosswalk of an intersection well while the image after moving forward ten meters captures the back crosswalk well. In this chapter, two datasets are introduced: one that uses safety cameras from a single bus that regularly passes the same locations and another that consists of images from dash cams from multiple vehicles. Our proposed method shows robustness in high-traffic areas and is able to localize changes to a specific crosswalk at an intersection. Detailed instructions, code, and data are made publicly available[5][6].

[5] https://github.com/tom-bu/crosswalk_change_detector
[6] https://www.kaggle.com/datasets/buvision/crosswalkchange

5.1 Approach

Figure 5.1: The left four images show the commuter bus, an image of the computer, the cabinet that contains the computer and electronics, and one of the cameras. The right diagram shows the field of view of each installed camera, where one faces forward and four side cameras face opposite directions from each other.

5.1.1 Vehicle-mounted Camera Data

This thesis evaluates change detection on two datasets. The primary dataset is derived from a metro commuter running between downtown Pittsburgh and Washington, Pennsylvania. A photo of the transit bus is shown in Figure 5.1. As of this writing, the bus has collected more than a year's worth of data and continues to collect data daily, and is therefore a valuable means to deploy the proposed method for live change detection. The bus makes at least two round-trips every weekday and contains a computing and storage device to acquire data from multiple sensors and to carry out the preliminary data processing. Four waterproof cameras are installed on the exterior four corners of the bus and one on the interior behind the windshield. On the exterior, two front cameras look backwards while two rear cameras look forwards, as seen in Figure 5.1. Data are transmitted from the bus through two cellular antennas and two dual-band WiFi antennas. There are also a GPS and an IMU on the bus. To manage data acquisition, communication, and storage, the Robot Operating System

(ROS) [62] is used. The advantages of using ROS are that it provides convenient data preprocessing packages and uses a modular publisher-subscriber-based messaging protocol. There are seventeen scenes with zebra crosswalks, as seen in Figures 5.2 and 5.3. There are six instances of change where five scenes have removed crosswalks and one scene where one crosswalk is transformed from a plain crosswalk into a zebra crosswalk, as shown in Figure 5.4. The remaining eleven scenes have no changes.

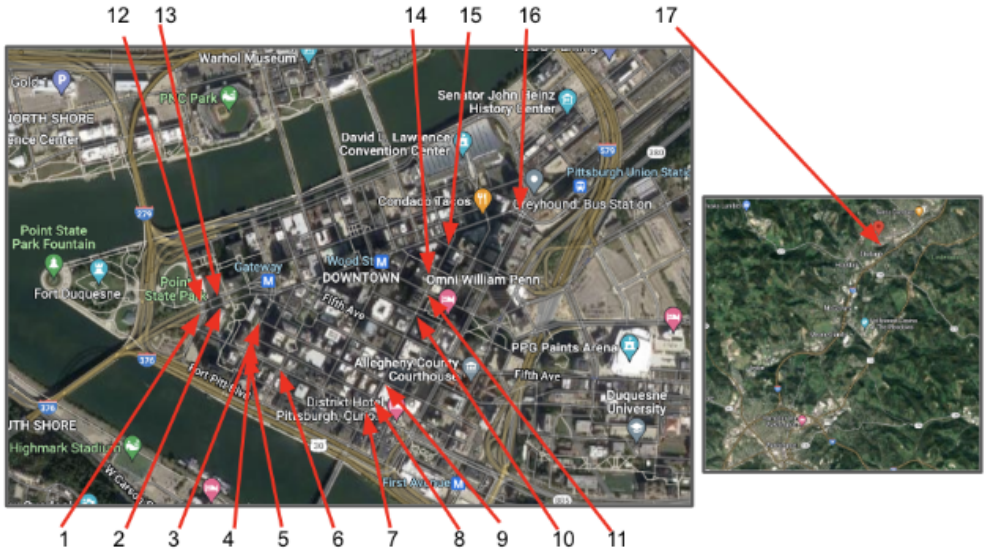


Figure 5.2: Locations of 17 zebra crosswalk scenes regularly observed by the bus. Most occur in downtown Pittsburgh with one occurring in Washington County.

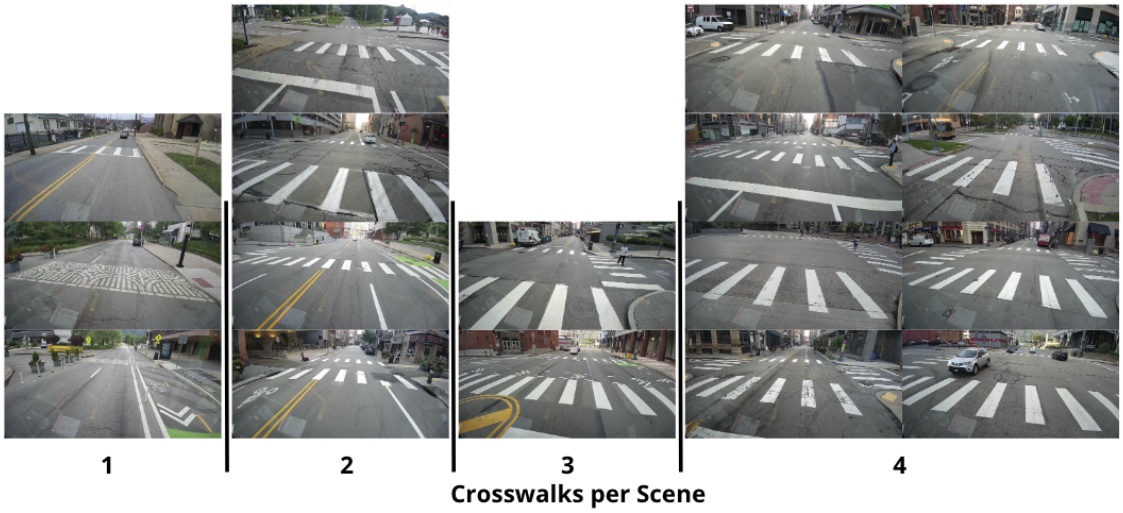


Figure 5.3: Images of the 17 crosswalk scenes and the diversity of each. They are grouped by the number of crosswalks at each scene.

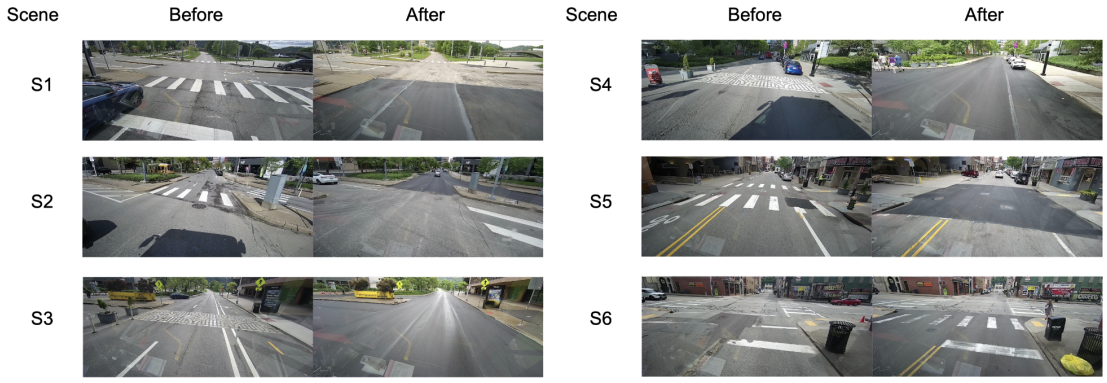


Figure 5.4: Here are six examples of changes that are observed on the bus. There exist many crosswalk removals due to a repaving of the road but one instance of a crosswalk transformation from a plain-to-zebra crosswalk in scene S6.

Another dataset exists that contains a collection of dash cam footage from different cameras and vehicles, as shown in Figure 5.5. The data sources are not actively collecting data, but this dataset is valuable in evaluating the ability to use the proposed methods to crowd-source data from multiple vehicles, since a single bus often takes a fixed route and has limited coverage of the road network. Crosswalks in this dataset mostly lie beyond the bus route, and the scenes add variety in terms of scenery and pedestrian and vehicle traffic. Another note is that most of the scenes in this dataset occur outside of downtown Pittsburgh, which is where many of the bus scenes are recorded. In this dataset, there are eight scenes where change occurred. But because recordings exist before the change occurred, each scene can be evaluated for no change as well, making a dataset of sixteen examples.

Figure 5.5: Here are four examples of changes that are observed. The left two scenes show added crosswalks, while the right shows plain-to-zebra crosswalk transformations.

5.1.2 2D Crosswalk Detector

The thesis uses the Detectron2 [67] library, which provides high-quality implementations of state-of-the-art object detection algorithms, to train a Mask R-CNN [26] model for instance segmentation of crosswalks. Mask R-CNN was chosen because it is a popular model architecture for instance segmentation and well recognized. In this work, we used the ResNet network with a depth of 50 layers [25] combined with a Feature Pyramid Network (FPN) [36], which extracts features of the input image at different scales. The decoder of the network consists of output heads for instance segmentation, i.e., classification, bounding box regression, and segmentation, with a loss function of $L = L_{RPNclass} + L_{RPNbbox} + L_{class} + L_{bbox} + L_{mask}$. $L_{RPNclass}$ and

$L_{RPNbbox}$ are the losses for the region proposal network. L_{bbox} and $L_{RPNbbox}$ use an L_1 loss, while $L_{RPNclass}$, L_{class} and L_{mask} use a cross-entropy loss. The crosswalk detector is pre-trained on the MS COCO dataset and then fine-tuned on the Mapillary Vistas Dataset [43], and the model with the highest validation accuracy is kept.

Fine-tuning the Detector on Pittsburgh Data

The model trained on the Mapillary Vistas Dataset is a general purpose detector. This detector is used for the second dataset with multiple camera sources. However, for the bus dataset, because the bus runs along the same route multiple times, many examples of the same crosswalk instance can be acquired in a variety of lighting and scene conditions, as shown in Figure 5.6. This can be used to improve the performance of the detector through fine-tuning. It is observed that the Mapillary Vistas Dataset-trained detector performs worse than the fine-tuned model because the Mapillary Vistas Dataset is collected from around the world and thus has some domain differences. In an iterative fashion, the model can inference on the latest data and poor detections can be annotated using the VGG Image Annotator [15, 16] and added to the dataset to retrain the model. The process is shown in Figure 5.7.

Figure 5.6: Crosswalks captured in different conditions. From left to right on the top row are winter, rainy, and overcast weather. On the bottom are different times of a sunny day.

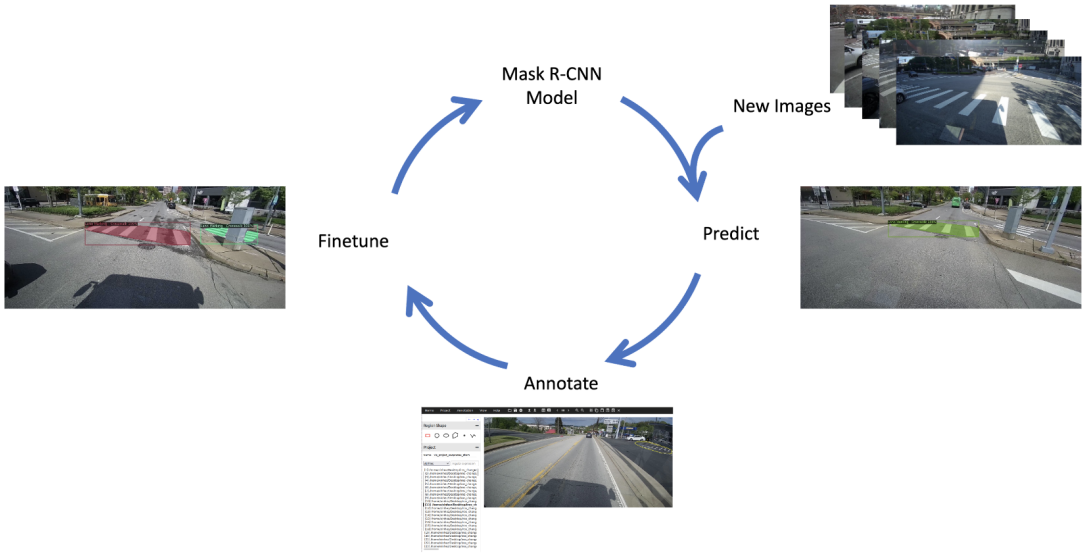


Figure 5.7: This demonstrates the fine-tuning process of Pittsburgh crosswalk detection where the cycle begins with an off-the-shelf Mask R-CNN model that predicts on new images collected in the wild. Incorrectly confident predictions are annotated while leaving accurate predictions. The process of efficiently fine-tuning the model on these labels led to improved results, as shown by the detection on the left.

Crosswalk Type: Zebra vs. Plain

Crosswalks can be split into two main categories: plain and zebra. In this thesis, zebra crosswalks are the primary focus for the following reasons: plain crosswalks have lower visibility compared to zebra crosswalks; plain crosswalks can easily be confused with lane markings depending on the viewpoint angle; plain crosswalks deteriorate more quickly due to wheel tracks; and plain crosswalks are more easily occluded by cars and road debris like snow. Thus, plain crosswalks are left for future analysis.

Instance Segmentation vs. Bounding Box

One design choice is the use of an instance segmentation model over a pure bounding box model, i.e., Mask R-CNN[26] over Faster R-CNN[52]. Mainly, an instance segmentation model provides a tighter bound on an object compared to a bounding box. This is the case for crosswalks that are parallel to the bus and appear diagonal in the front camera. In this configuration, corners of the bounding box would be non-crosswalk areas. For change detection, where intersection over union is used for comparison, a loose bounding box would produce frequent false positives.

5.1.3 BEV Crosswalk Change Detector

Previously, crosswalk detections existed in the 2D image plane. This section describes how to map 2D crosswalk detections onto a ground plane. To do so, COLMAP [60, 61] software is required to perform structure-from-motion, to reconstruct the 3D scene and estimate poses of the images taken. The steps are as follows.

1. Camera Calibration

All cameras are expected to be calibrated or have low distortion. If cameras are not calibrated, a checkerboard grid can be used to estimate radial and tangential parameters or COLMAP can be used to estimate the parameters automatically. For successful COLMAP calibration, many images have to be taken from each uncalibrated camera and the camera ID for each image must be specified.

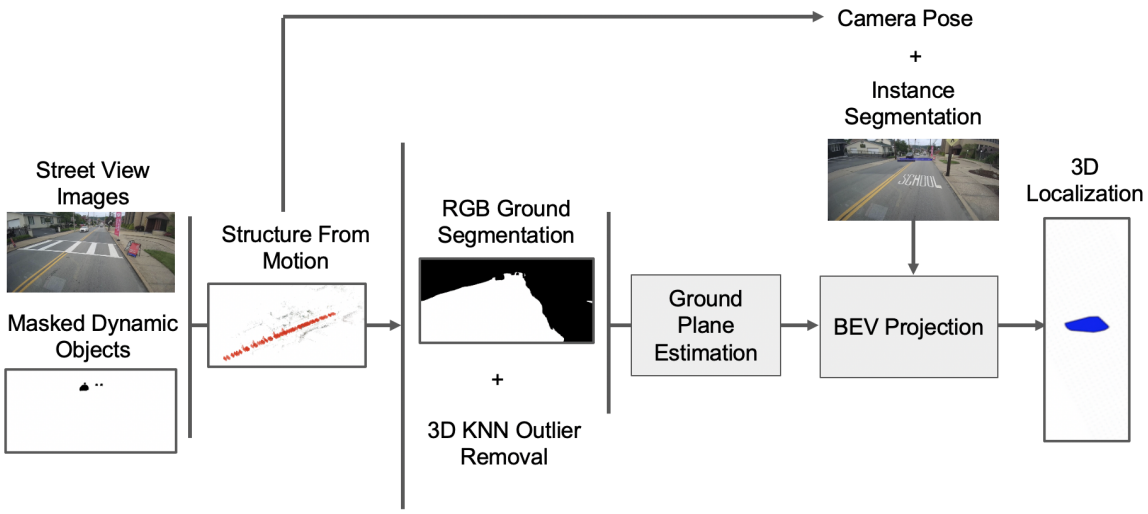


Figure 5.8: Pipeline of the image plane to ground plane mapping of the crosswalk detector.

For example, the raw images from the bus contain large radial distortion, with distortion being most severe at the corners of the image. This can lead to poorly estimated camera poses because many key points exist on the perimeter of the image where static objects like buildings are seen and provide localization. Using a checkerboard calibration method resolves large parts of the distortion in the center of the image. However, the top corners are poorly calibrated, as seen in Figure 5.9. By performing structure-from-motion at a single intersection where the bus has recorded images driving in both directions, as seen in Figure 5.10, COLMAP can provide a better estimate of the camera intrinsic and distortion parameters.

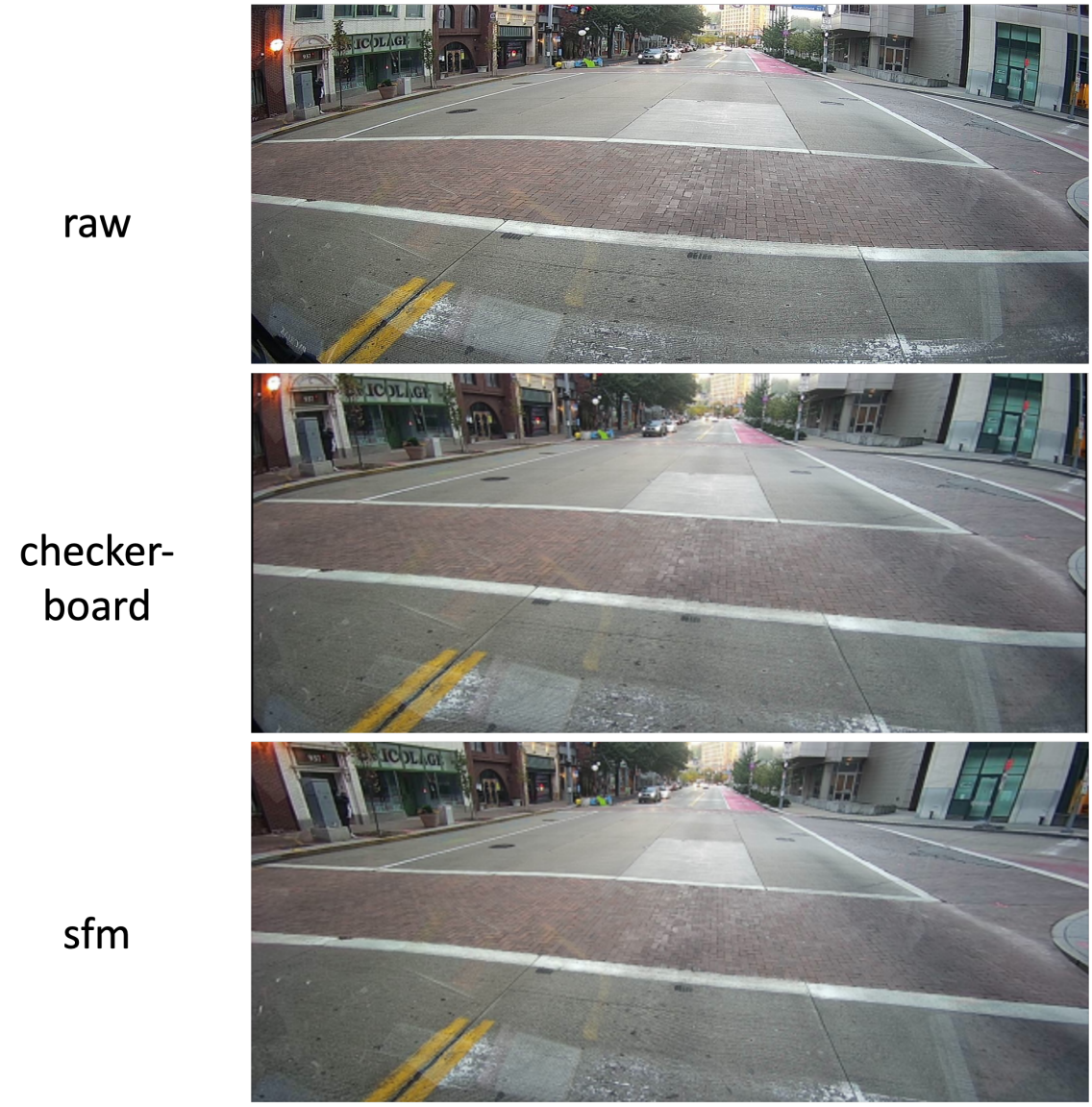


Figure 5.9: Camera calibration from the raw image to the checkerboard to the sfm estimated.

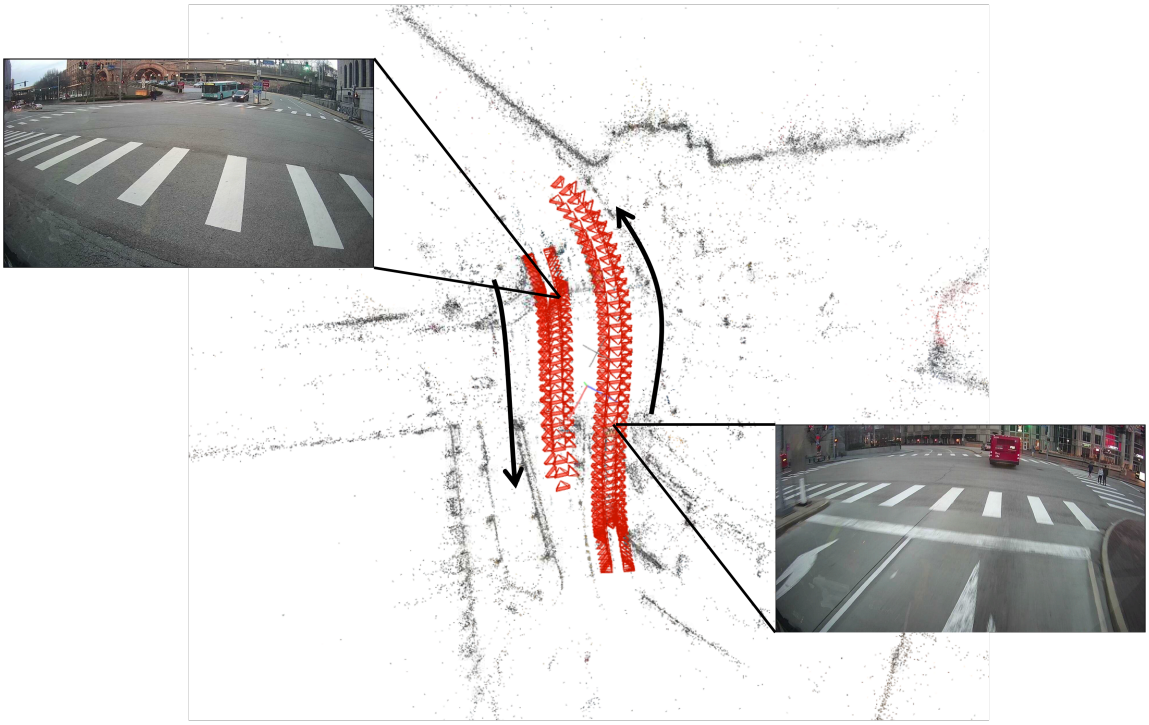


Figure 5.10: Sparse reconstruction and image pose calibration of the bus at an intersection driving in both directions. Each red triangle represents one image. All five cameras of the bus are used. The left track is the bus driving down while the right track is the bus driving up, as indicated by the arrows. Images of the center camera from each drive are shown.

2. Dynamic Objects

Because street view scenes often contain dynamic objects such as people, vehicles, and the sky, it is necessary to exclude their key points in the structure-from-motion process. The reasons are as follows:

- Oftentimes dynamic objects will not be consistent between scenes taken on different days. An example is one set of cars parked on the side of the road one day and another set of cars parked on another day. In this case, the structure-from-motion will create separate scene graphs and models of the scene.
- Another situation occurs when vehicles drive at the same speed as the ego-vehicle. If a vehicle in front of the ego-vehicle maintains the same distance from the ego-vehicle as the ego-vehicle drives forward, the estimated distance of the front vehicle will approach infinity.

In order to address these issues, an off-the-shelf panoptic segmentation model [30] trained on MS COCO [37] is used to output masks for dynamic objects to determine whether a key point lies within the mask and should be excluded as seen in Figure 5.11. One thing to note is that if much of the image is masked, it can also lead to poor localization of that image.

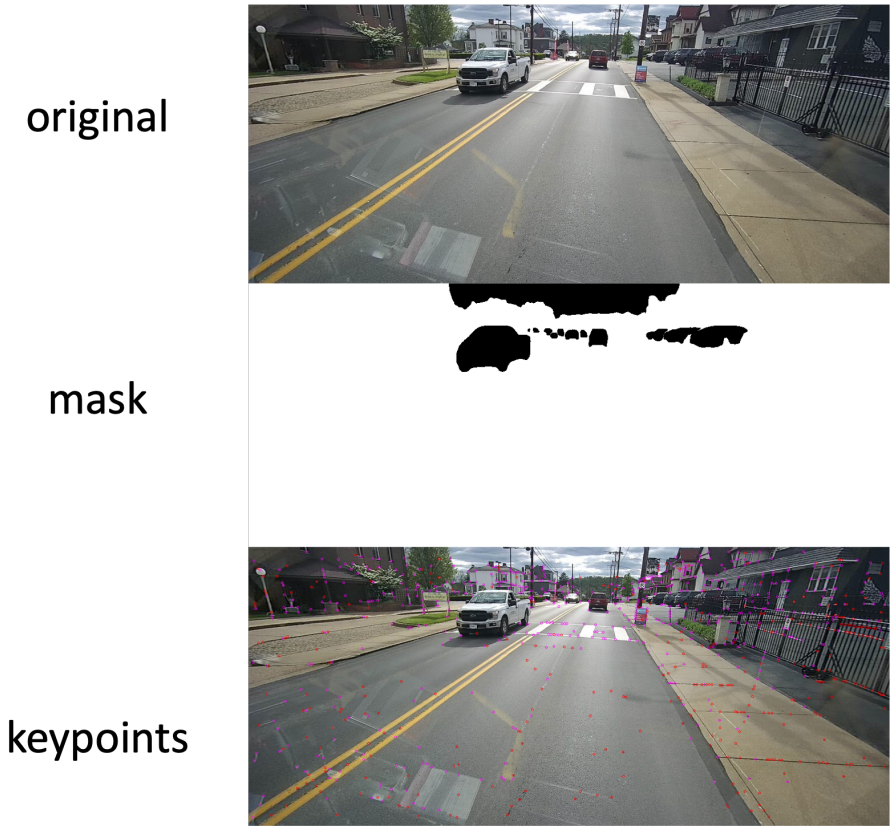


Figure 5.11: Masking key points for dynamic objects. The top image shows the original image from the scene. The middle image shows a binary mask of dynamic objects in the scene such as vehicles and the sky. Key points that lie in these areas are not used in the structure-from-motion process. The bottom image shows detected key points in the image, but none exist where the vehicles and sky are.

3. Ground Plane Estimation

While working at small scales, the road can be considered a flat surface. Even though roads are curved downwards on the sides, this planar road assumption is a practical and effective assumption in helping map detections between the 2D image plane and the 3D ground plane. The ground plane is thus estimated in the 3D reconstruction. Structure-from-motion provides a list of correspondences between 2D pixels and their 3D counterpart. Using a panoptic segmentation model [30] trained on MS COCO [37], parts that belong to the road are segmented. With the 2D-to-3D point pairing, a 3D point cloud segmentation of the road can be obtained. Because the road has repeated textures, the key point matching can be imprecise and because images are taken from a camera with only forward motion, the depth for some ground points can be poorly estimated. These outliers are removed using k-Nearest Neighbors, where the distance to 5 nearest neighbors is calculated. Points with distances below the mean of distances are kept. From then, three points are randomly selected to estimate a plane and RANSAC is performed to find the plane that fits the most number of ground points. The ground plane is shown in Figure 5.12, where the green points are points on the ground plane.

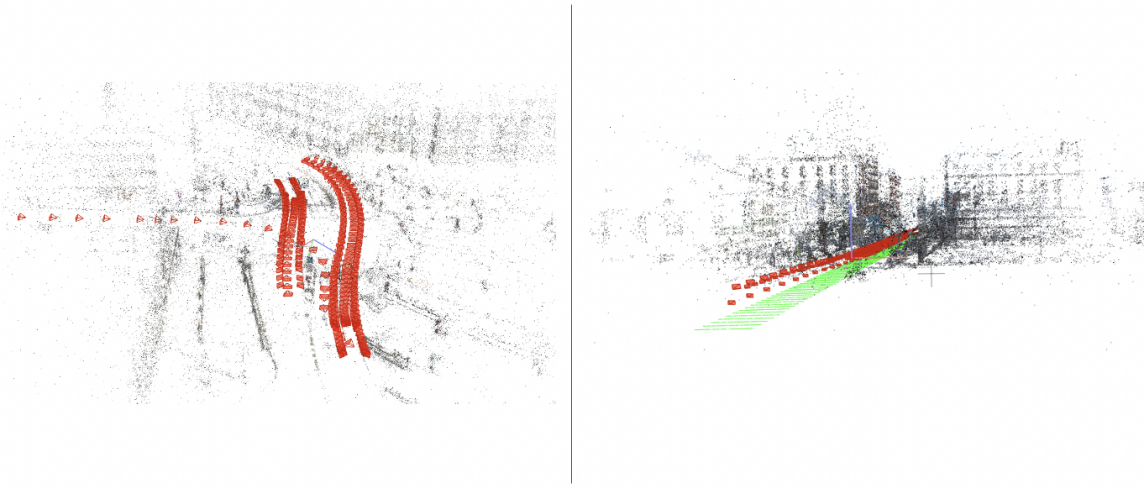


Figure 5.12: Structure-from-motion of images aligned across time. The left shows a reconstruction with images from a bus and dash cam photos. The right shows multiple dash cam photos aligned, where the green points indicate the ground plane estimation.

4. Coordinate Frame Adjustment

The reconstructed scene from COLMAP's structure-from-motion treats the coordinate frame of the first registered image as the global coordinate frame. To generate an accurate bird's-eye-view, the 3D reconstruction and the camera poses are translated and rotated such that the z-axis of the global coordinate frame aligns with the normal of the ground plane. The translation is given by the location of the ground plane and the rotation is calculated by using Rodrigues's formula for finding the rotation between the ground plane normal vector and the current z-axis vector. Lastly, the y-axis is aligned with the direction of one of the camera sequences, by applying a 2D rotation of the reconstruction.

5. Homography Transformation

The crosswalk Mask R-CNN model, described in Section 5.1.2, is used to detect each instance of a crosswalk in each image, while a homography transformation between each image plane and the 3D ground plane is estimated. This homography transformation will be used to map each 2D detection out of the image plane and onto the 3D ground plane. Because the z-axis of the world frame is aligned with the ground plane, the x and y coordinates provide the positions in the BEV representation. To estimate the homography transformation, the ground plane is represented by a uniform grid of 3D points. When these points are projected into each image, they can be used as correspondences between the two planes. The homography transformation can be calculated with at least 4 correspondences using the Direct Linear Transformation algorithm.

6. Postprocessing

Now that crosswalk detections are compiled into one global coordinate system, some useful postprocessing steps are used.

- **Non-maximum Suppresion:** First, non-maximum suppression is performed such that for detections that overlap by an IOU of 0.10 or more, the higher-confidence detection is kept.
- **Multi-frame Consistency:** To avoid false positive crosswalk detections,

detection consistencies are checked across multiple frames. The intuition is that the same crosswalk should appear in multiple frames and be detected each time as the vehicle moves towards it. Therefore, if the same crosswalk detection on the ground plane is not seen in more than one frame, it is likely a false positive and should be removed. Cases of this occur when certain angles or shadows give the false impression of a crosswalk to the detector, but at another frame the detector predicts correctly. Therefore, for each crosswalk in BEV, a minimum of three detections, depending on the camera frame rate, are needed with an IOU of 0.10 or greater to be used for downstream change comparison. An IOU of 0.10 is used because experiments shown in Table 5.3.

7. BEV Comparison

Once all detections are compiled into a single global frame and postprocessed, comparisons can be made between detections from the reference images and the query images. Detections that have an IOU greater than 0.10 are matched. Those that are not matched are classified as either added or removed crosswalks as seen in Figure 5.13. Though an IOU of 0.10 is relatively low, it allows for flexibility in the detection since small inaccuracies in the 2D to 3D mapping can exist and since some crosswalks may be occluded by vehicles and the detection only partially covers the crosswalk.

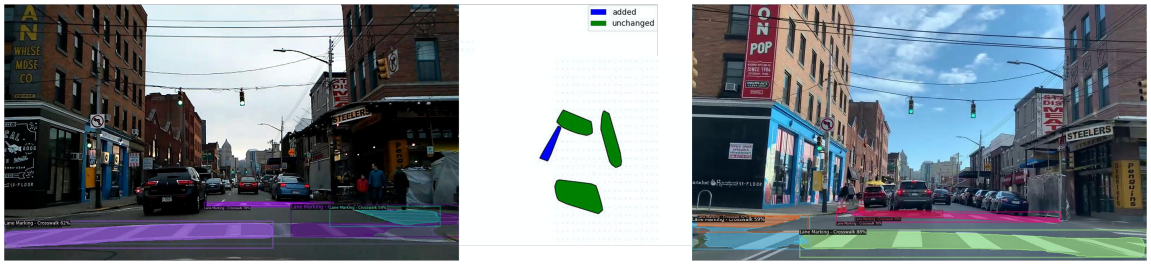


Figure 5.13: BEV output of the fine change detector. The left image is taken in 2017 and the right image is taken in 2021. Both come from dash cam recordings.

5.1.4 Live Data Pipeline with Gabriel BusEdge

The previous section described the change detection method when before and after images are provided at a given scene. This section describes how the change detector

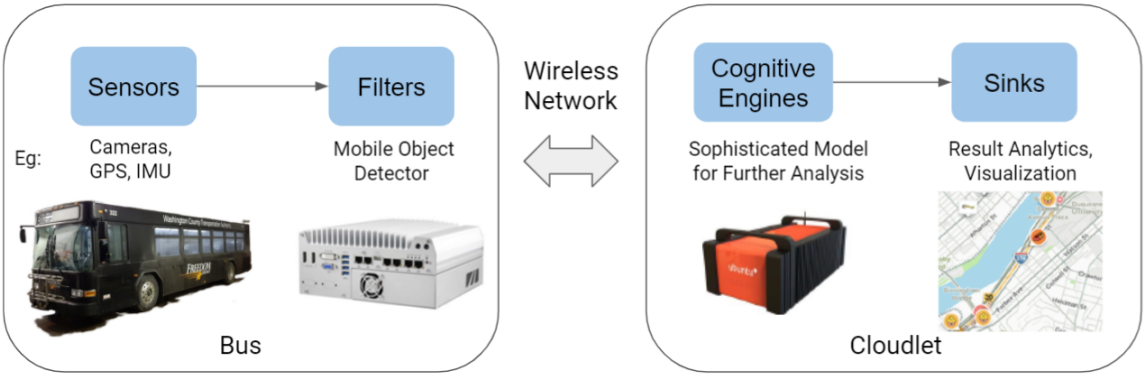


Figure 5.14: Major Components of the BusEdge Platform [22, 70]

Brand	Safety Vision
Model	RoadRecorder 9000
CPU	Intel Core i7-8700t @ 2.40GHz
RAM	16 GB
Storage	5 TB
Power Input	9-48V DC
Dimensions	289 × 118 × 250 mm

Table 5.1: Technical specifications of the bus computer.

can be integrated with a bus using Gabriel [22] and why edge computing is needed. The main idea is that the previous section's method works on small spatial scales such as an intersection, and a method is needed to extract from city-scale image streams to images from local scenes. Otherwise, not only would performing structure-from-motion on the city scale take significant time and computational resources, but also many of the data are irrelevant and can be discarded, such as highways, where crosswalks do not exist. Sections 5.1.5 to 5.1.8 describe how the Gabriel-BusEdge system developed by [22, 70] is adapted to create a low-latency system for near-

CPU	Intel Core i7-7700 CPU @ 3.60GHz
OS	Ubuntu 18.04
GPU	2 GeForce GTX 1070
RAM	32 GB
Storage	10 TB

Table 5.2: Technical specifications of the cloudlet server.

daily change prediction results. The Gabriel-BusEdge system abstracts the data pipeline into four parts: *Sensors* and *Filters* that exist on the "client" bus and the *Cognitive Engines* and *Sinks* that live on the "cloudlet" server. Figure 5.14 shows the abstraction. The client computer has the specifications listed in Table 5.1 and lives on the bus reading the live sensor data. Meanwhile, the cloudlet server has specifications listed in Table 5.2.

Sensors are the data sources on the bus, mainly the video streams from multiple cameras and GPS data. The data often need preprocessing and engineering before being sent to the filters next.

Filters represent the data reduction components running on the in-vehicle computer. Multiple applications can exist for filtering the data, and each application can send its outputs to the specific cognitive engine on the cloudlet server for further analysis.

Cognitive Engines are modules on the server that analyze data received from the filters in greater detail, and the processes often carry heavier compute requirements than those of the filters.

Sinks represent the final components on the cloudlet server to collect all of the results from different cognitive engines and summarize the results through analytics or visualizations. This can take the form of an OpenStreetMap [23] server or an email notification.

5.1.5 Client Sensor

On the client side, as the sensor data are read, the following preprocessing and feature engineering steps are taken before sending the data to the client filter.

Data Sync

Because the camera sensors are read at a different frame rate from the GPS sensor, data need to be matched. Binary search is used to find the GPS data with the closest time stamp as the camera data.

Heading Calculation

With some engineering, another piece of information that is recorded is the bus heading. Using the history of the bus GPS, the geographic heading of the bus is

calculated. Formally, given the GPS location $x \in \mathbb{R}^2$ at time t, the heading of the bus is given by $dir^t = x^t - x^{t-1s}$, where $dir \in \mathbb{R}^2$.

Redundant Images

As the bus picks up passengers or waits at a traffic light, the bus will generate redundant information where there is little change between the current image and the last. These redundant ones are filtered out by measuring if the distance in the corresponding GPS coordinate exceed a given threshold from the previously recorded GPS coordinate.

Buffer

A buffer of data is created of the sensor data. This ensures that no data are dropped while the data are being preprocessed.

5.1.6 Client Filter

The client filter reads the preprocessed data stored in the buffer and releases the buffer memory as it is read. The data undergo the following steps and then useful data are sent to the cloudlet server's cognitive engine.

Filter Type

Multiple use-cases exist for what is relevant information and what should be sent to the server. Two examples are as follows:

- **Proximity to known crosswalk GPS locations:** If the GPS locations of some crosswalk scenes are known, images from the bus that are close to the known GPS locations can be extracted to verify if the known crosswalks still exist. Because some noise is present in the GPS data, a wide radius is used to capture multiple frames close to each scene's GPS coordinate. Note that because the GPS is the location of the bus, when the GPS is closest to the known GPS, the crosswalk scene is no longer viewable by the bus, but rather, the crosswalk may lie underneath the bus. To address this issue, the current GPS location is offset by a multiple of the bus' heading vector: $\hat{x} = x + 0.001 * \frac{dir}{||dir||}$, where $\hat{x}$ is

the viewed GPS. If this viewed GPS is within the proximity of an intersection, the image is assigned to that known crosswalk.

- **Detected objects of interest:** Besides verifying known crosswalks, new crosswalks outside of the known crosswalk locations can also found. In this case, an instance segmentation model should exist on the edge computer of the bus to be used as a classifier for the presence of a crosswalk in each image. Though an instance segmentation model is a larger model than a typical classification model, the instance segmentation predictions can be used to explain classification results if needed. Note that a detection that only occurs in one image but not in the neighboring images is a sign of a false positive detection because the scene is continuous as the bus moves forward. Positive classifications that persist over multiple sequential images indicate a high chance of a crosswalk detected and indicate these images are worth sending to the cloudlet server.

Image Cluster vs. Single Frame

In the original formulation of the Gabriel-BusEdge system presented in [22, 70], a single image frame is sent to the cloudlet server. This formulation emphasizes low latency and detecting objects on the latest images. This would be necessary if the detection result is time-critical, but because the bus is driven manually, the latest image requirement can be relaxed and images can be sent to the cloudlet server as clusters. Clusters are beneficial in this situation because the structure-from-motion on the server side performs better when more images are provided. It also means that more consistency checks across frames can be made to reason whether crosswalk detections are real or false positives. Figure 5.15 illustrates the data pipeline on the client side.

5.1.7 Server Cognitive Engine

The cognitive engine on the cloudlet server then receives the data packet. The packet contains images for each scene, the crosswalk scene assignment, and the corresponding GPS and bus heading of each image. Multiple cognitive engines can exist, but in this thesis the primary cognitive engine is the BEV crosswalk change detection pipeline described in section 5.1.3. The following changes are made to the BEV change

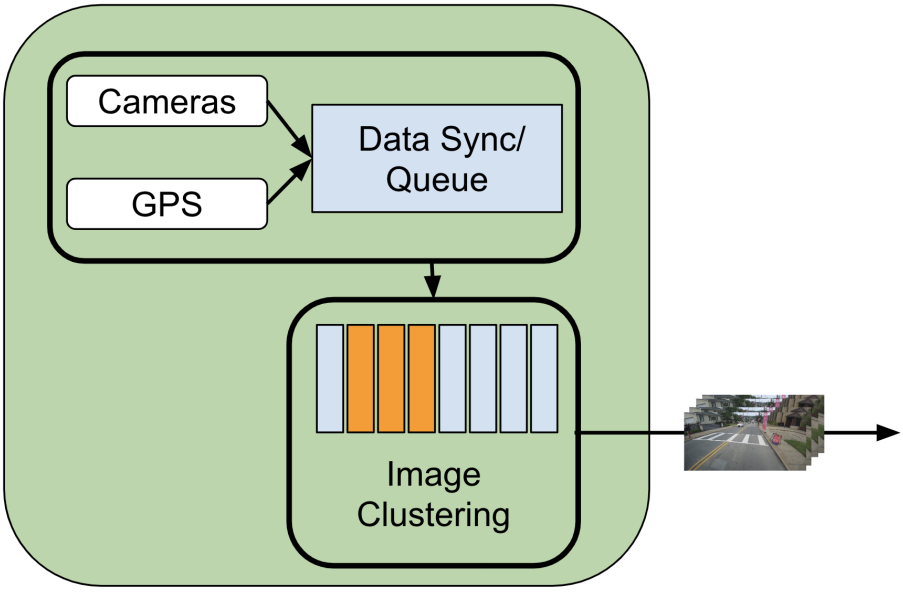


Figure 5.15: Diagram illustrating the preprocessing and filtering steps that occur on the client side before sending data to the cloudlet server.

detection pipeline to work with incoming bus data. The output of the pipeline will be the type of change that has occurred. However, in cases where the data are unsuitable for structure-from-motion or have heavy occlusion, the scene can be discarded until the next time the bus passes it. After a decision is made, the result can be sent to the cloudlet sink.

Reference Image Bank

In the BEV change detection pipeline, there are current "query" images and past "reference" images. The search for "reference" images is automated by searching by the crosswalk scene assignment, the GPS location, and the heading. The heading is necessary because images facing in the same direction as the query images are needed in structure-from-motion. Feature matching between images uses image features such as SIFT are only robust to 30-40° rotations. Therefore, images taken at a greater angle will create broken structure-from-motion models. Formally, the direction dir between the query images $query$ and the reference images ref should have a cosine

similarity greater than some threshold, i.e. $\frac{dir_{query} \cdot dir_{ref}}{||dir_{query}|| \cdot ||dir_{ref}||} > \cos\theta$. $\cos\theta = 0.5$ is a good threshold because it means $\theta = 60°$ and provides some flexibility in case the GPS direction is not accurate. If the images are not assigned a crosswalk scene, then the GPS is needed, so that relevant images with a GPS x^{ref} are within some threshold to the query image's GPS, x^{query}, $||x^{ref} - x^{query}||_2 < a$
where a can be tuned. For known crosswalks, the reference images are pregrouped and labelled with their heading. Pregrouping the images ensures that the images capture the scene well for structure-from-motion, i.e., there is ambient lighting and there are few obstructions for crosswalk detection.

Start and End Image Alignment

Even though the BEV change detector does not need images to be taken from the exact same location, some alignment is needed to make sure that the reference images and the query images start and end close to each other. If not, then if the query images go beyond the reference images and see the next intersection's crosswalks, the BEV change detector will incorrectly predict added crosswalks, only because the reference images are unable to see as far as the query images. The reverse could also happen. Thus, after the image poses are estimated from structure-from-motion, the start and end images from both the reference and query images are aligned so that both have intersecting trajectories. This alignment is performed by finding the nearest neighbor between reference and query image poses in $\mathbb{R}^3$. Both reference and images are ordered with respect to the time they are taken. Thus, for the start and end reference images, the closest query images are found. This will determine the start and end images for valid query images. With new start and end of query images, the opposite is performed, where the closest reference images are found. This produces valid start and end images for reference images. This nearest neighbor search produces an intersection of query and reference images that start and end close to each other.

5.1.8 Server Sink

After the data packet is processed by the BEV change detector, the results can be sent to a sink described as follows.

Data for Training

One type of sink can be the images and predictions of each image from the cognitive engine. Poor detections can indicate where the detector is weak, and the images can be used for further fine-tuning, while building a more diversified dataset. Similarly, images can be queried by GPS for a specific crosswalk in different lighting and weather conditions to make the model more robust.

Summary for the Day and Week

Change results can be summarized in an email on a daily or weekly basis. If one scene is evaluated multiple times because the bus has passed by the scene more than once, a probability of change can be given, where the number of times change is detected is summed and divided by the total number of bus passes at the scene. A probability threshold of 0.5 can be used for an overall decision.

Live Map

Lastly, the results can be visualized on a map display, as seen in Figure 5.16. It shows the qualitative result from the live bus data, where the blue line on the map illustrates the entire route of the bus. The icons of crosswalks are marked along the bus trajectory according to the GPS information of the images. Users can click on the icon to view the image to verify detection results. Figure 5.17 illustrates the data pipeline on the server side.

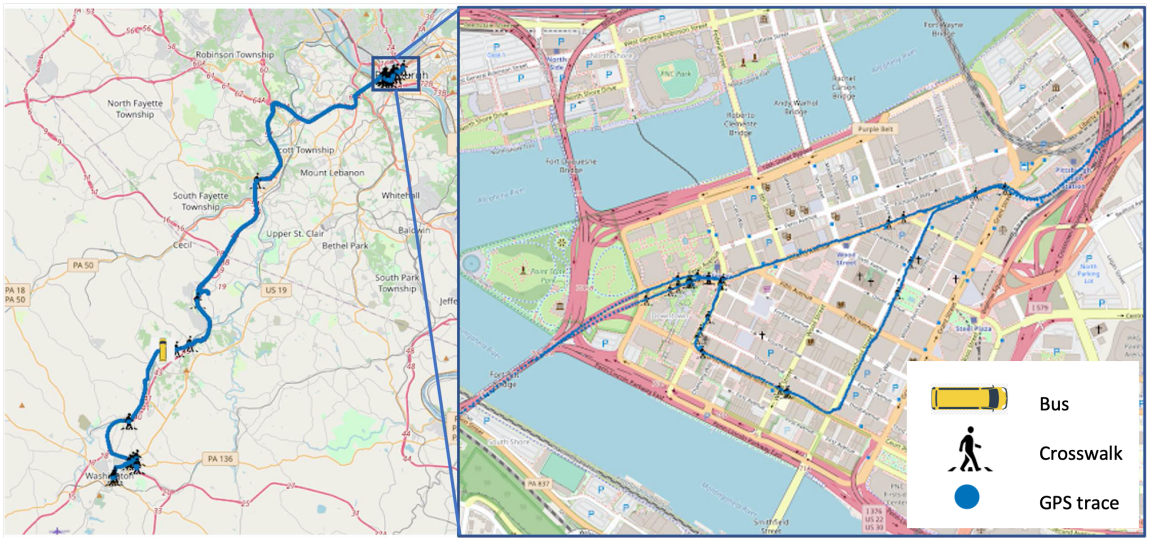


Figure 5.16: Display of the bus trajectory and crosswalk detections on a Live Map.

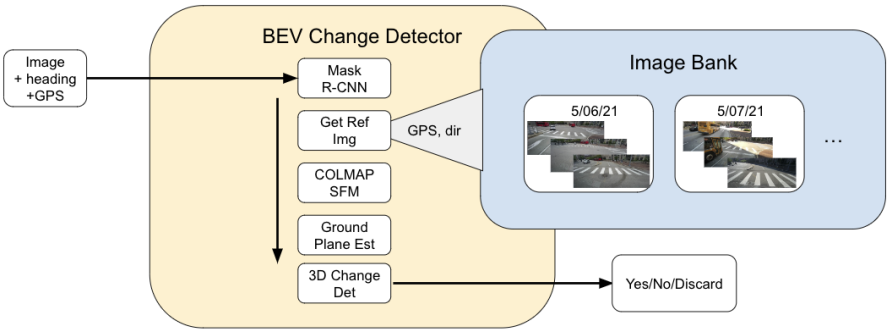


Figure 5.17: Diagram illustrating where the BEV change detector is inserted into the cloudlet server and how to query for reference images.

5.2 Results

5.2.1 Evaluating the Crosswalk Detector

Because a change detector is only as good as its object detector, the trained instance segmentation crosswalk detector is evaluated in three different ways: binary classification of each image, binary classification of image clusters, and instance segmentation. A round trip recording of the bus with 8622 images taken at one image per second was used. The first metric is with binary classification of whether any crosswalk is detected

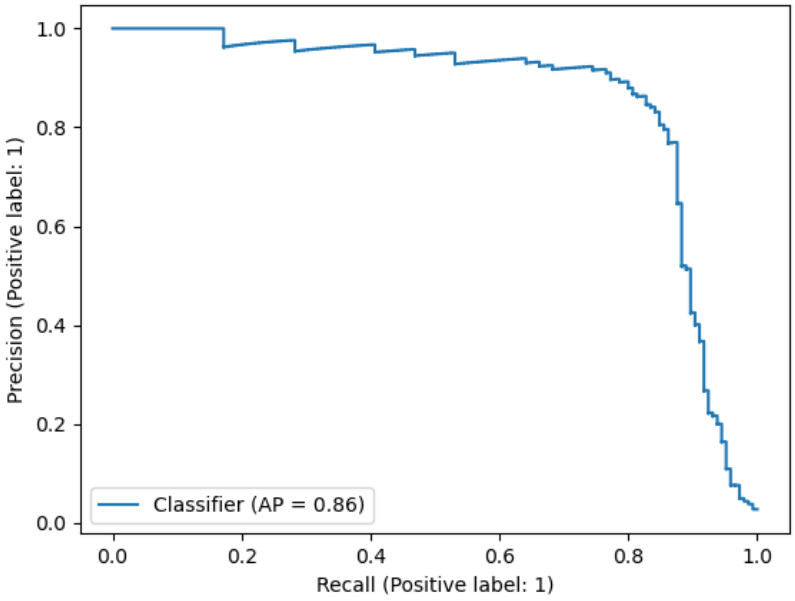


Figure 5.18: Precision-recall curve when evaluating the crosswalk detector with binary classification per image.

in an image. The classification accuracy is useful because images sent by the bus client are only those with crosswalks detected. Even though the crosswalk detector is an instance segmentation model and provides more information than is needed for the task, the results are more interpretable than a classification alone in the case of false classifications. In other words, the instance segmentation model's accuracy is evaluated at an IOU of 0. The average precision is 0.86, and the precision-recall

curve is shown in Figure 5.18.

Because images are sent to the bus in image clusters to represent intersections or scenes with crosswalks, it is also useful to measure the classification accuracy of whether these clusters contain crosswalks. In this case, clusters of 5 images are used taken at 1 fps. Clusters that have crosswalks in all images are given a positive ground truth label, and clusters with no crosswalks in all images are given a negative label. The score given by the model to each cluster is the number of frames with crosswalks detected with a confidence > 0.5 divided by the number of frames in the cluster (5). The average precision is 0.91, and the precision-recall curve is shown in Figure 5.19.

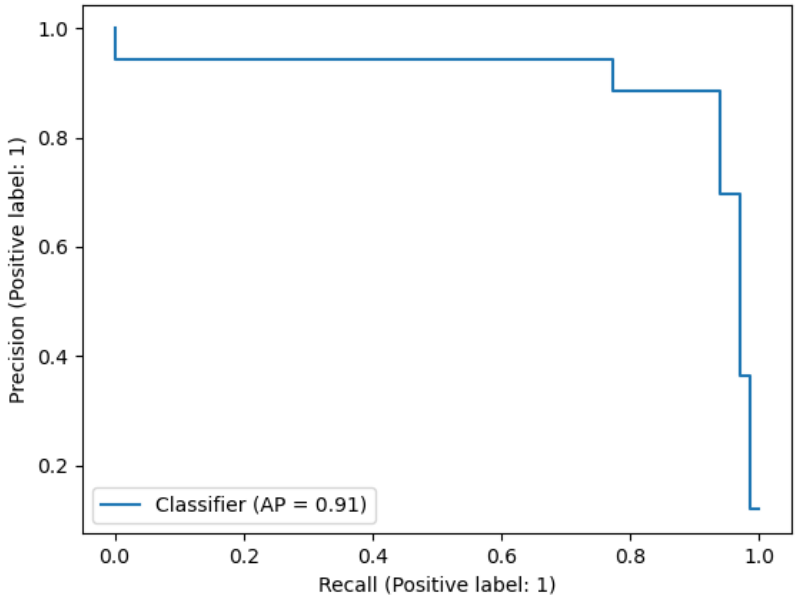


Figure 5.19: Precision-recall curve when evaluating the crosswalk detector with binary classification on image clusters.

Lastly, for the instance segmentation model, the standard MS COCO evaluation metrics [1] are provided, where the mean average precision is recorded by average intersection over union (IoU) $\in [0.5 : 0.05 : 0.95]$, AP_{50} (IoU $= 0.5$), AP_{75} (IoU $= 0.75$), AP_{small} for small objects (pixel area $\leq 32^2$), AP_{medium} for medium objects ($32^2 <$ pixel area $\leq 96^2$), AP_{large} for large objects ($96^2 <$ pixel area). Because the downstream task is to compare detections from query images against detection from

reference images, the IOU = 0.1 is a useful metric since the matching criteria for crosswalk verification is at IOU = 0.1. Another consideration is that because image sequences are used, crosswalks get larger as the vehicle approaches them. A low AP_{small} can be compensated with a high AP_{large} because it means that the crosswalk will eventually be detected. The evaluation results are shown in Table 5.3. Even though by the MS COCO metrics, the accuracy is average, having standards as strict as MS COCO is not representative in demonstrating the detector's usefulness, especially in this application where an IOU = 0.1 and using predominantly large instances is fine.

Table 5.3: Detection accuracy for instance segmentation. The first six lines are metrics used in the MS COCO evaluation. Additional metrics are recorded to show that high average precision (AP) is achieved at smaller IOU and for larger objects, which is more indicative of the detector's utility in this application. AP_{MVD} represents the AP achieved from training on the Mapillary Vistas Dataset, and AP_{tuned} represents the AP achieved from fine-tuning the model on locally collected data. As shown, fine-tuning shows significant improvements to the model.

	IOU	Area	AP_{MVD}	AP_{tuned}
COCO Metric	0.50:0.95	All	9.6	22.8
	0.50	All	33.3	59.8
	0.75	All	0	0
	0.50:0.95	Small	3.4	6.5
	0.50:0.95	Medium	11.1	18.0
	0.50:0.95	Large	10.6	32.9
	0.10	All	48.5	72.6
	0.10	Small	28.6	27.4
	0.10	Medium	51.2	68.1
	0.10	Large	54.9	83.8
	0.50	Small	16.7	26.8
	0.50	Medium	38.3	51.3
	0.50	Large	33.6	75.3

5.2.2 Evaluating the BEV Change Detector

As mentioned in Section 5.1.1, a change dataset exists for both images recorded by a bus and images from multiple sources. The bus dataset contains 17 different intersection scenes, six of which contain changes. Since intersections may have multiple crosswalks, the total number of crosswalks across the 17 intersections is 50. The images from the bus dataset are prefiltered by weather, lighting, and start and end frames to represent the ideal input for both detection and structure-from-motion. Under these optimal conditions, the BEV change detector is able to show strong results in classifying whether each scene and crosswalk has a change or not. However, deployment of the methods live on the bus have not shown consistent results like these, and the reasons are addressed in Section 5.3. Moreover, while 17 scenes is not representative of all crosswalks in a city, these are the crosswalk intersections that exist on the bus route that the bus passes on a regular basis. Qualitative results of six of the 17 changes are shown in Figure 5.20, where many crosswalks have been removed due to road repaving. Each crosswalk is represented by a polygon, and the color indicates the type of change. If any crosswalk experiences a change, the change prediction for the scene is positive. For the change dataset collected from multiple sources, eight changes and eight no changes exist. The accuracy is 0.88 for change detection and 0.63 for no change detection. Because the images are taken from multiple sources, image quality, size, distortion, and view points are different. This leads to different data distributions of how crosswalks appear in the image. As a result, the detection accuracy suffers and causes poor change detection results.

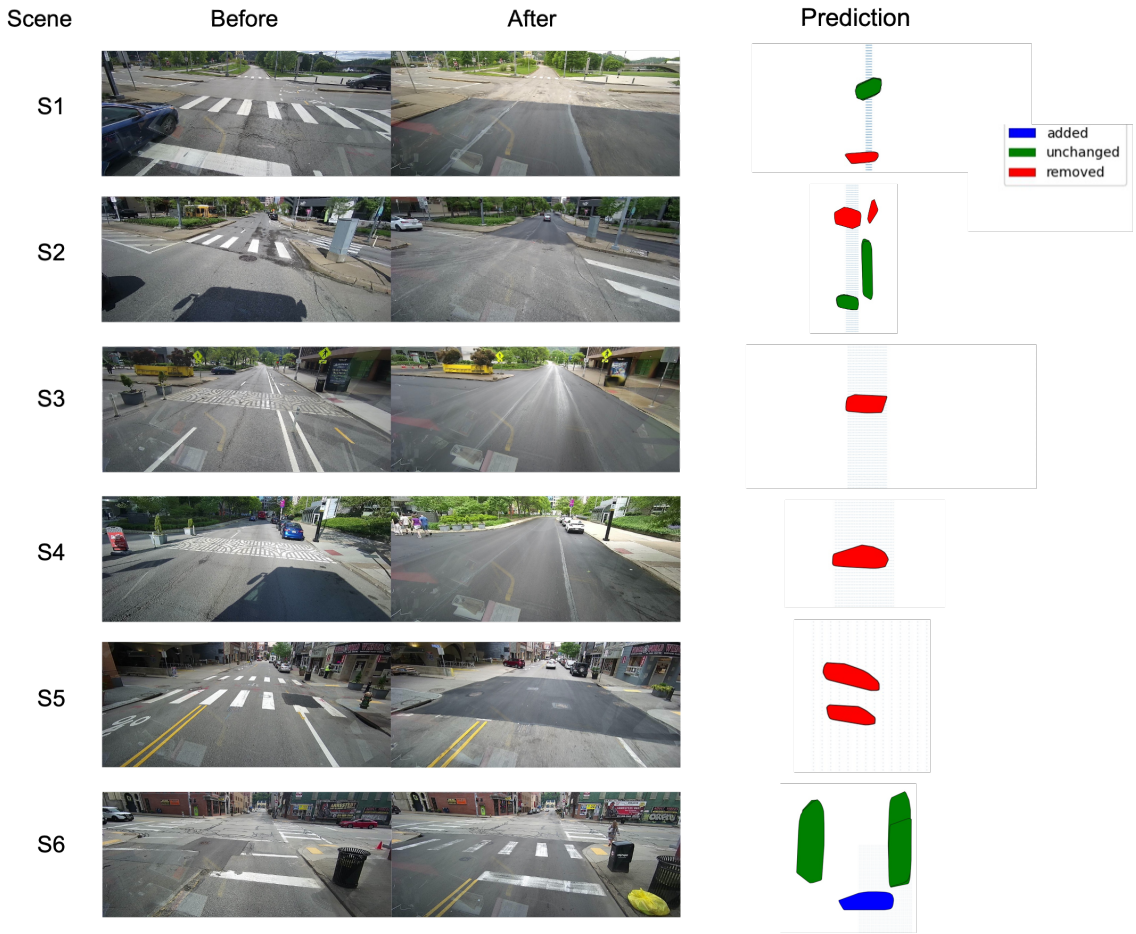


Figure 5.20: For six different scenes, one image each from the query and reference sequences is shown along with the change prediction. In each of the six scenes, one or more crosswalks have changed. Scenes 1-5 have crosswalks removed, while scene 6 has one plain crosswalk transformed into a zebra crosswalk.

5.2.3 Data Filtering

A round trip of the bus takes approximately four hours, and the bus will often make two round trips per day. Using the trip from June 28, 2022 as an example, the trip generated 133,357 images from a single camera alone at 5 frames per second. Because the bus will be stationary at parking lots, bus stops, and traffic lights, it generates many redundant images. After filtering for movement using the GPS, 17,067 images are left, which is a 90% data reduction. There are two additional filters that can be independently applied on the bus. The first is the GPS filter that uses known crosswalk intersection locations to filter for images. This filter reduces the non-stationary images to 984 images, which presents another 94% data reduction. The second filter is a crosswalk filter. Using a Mask R-CNN model with a confidence threshold of 0.50 for detections, the non-stationary images can also be reduced to 609 images, which represents a 96% reduction. The crosswalk filter detects both new and old crosswalks, but the reason why the crosswalk filter extracts fewer images than the GPS filter is that the crosswalk filter can only detect crosswalks when there is good visibility, such as crosswalks 5-10 meters in front of the bus or when there are no occlusions, unlike the GPS filter. This data reduction shows the necessity for an edge computer to reduce the amount of data that need to be sent to the cloud.

5.2.4 Self-Correction

If the BEV change detector is performed on the same location across multiple days, the majority vote is potentially able to eliminate false positives. Because Pittsburgh weather is mostly overcast, days with strong lighting that can cause these situations can be overruled by evaluating the scene at other time points, as seen in Figure 5.21. An example of false detection is at t_3 where shadows and reflections on the road confuse the detector.

Figure 5.21: We examine how the change detector performs over four time steps, where t_0 is the reference. Each row represents a different location. In these three locations, the detector determines there is change at t_3 due to false detections due to lighting, but because of the correct detections in the other three time steps, the final determination is that there is no change.

5.2.5 Live Deployment

The methods were deployed live over 7 workdays across two weeks, where two intersections were monitored. One human intervention was required to reset the wireless connection between the bus and the server and some image clusters were dropped by the Gabriel system, but each intersection was able to be evaluated each day. Because the bus performed multiple trips per day, the intersections were evaluated multiple times per day. The results are shown in Figure 5.22 for the number of correct and incorrect evaluations. The reasons for errors are further discussed in Section 5.3.

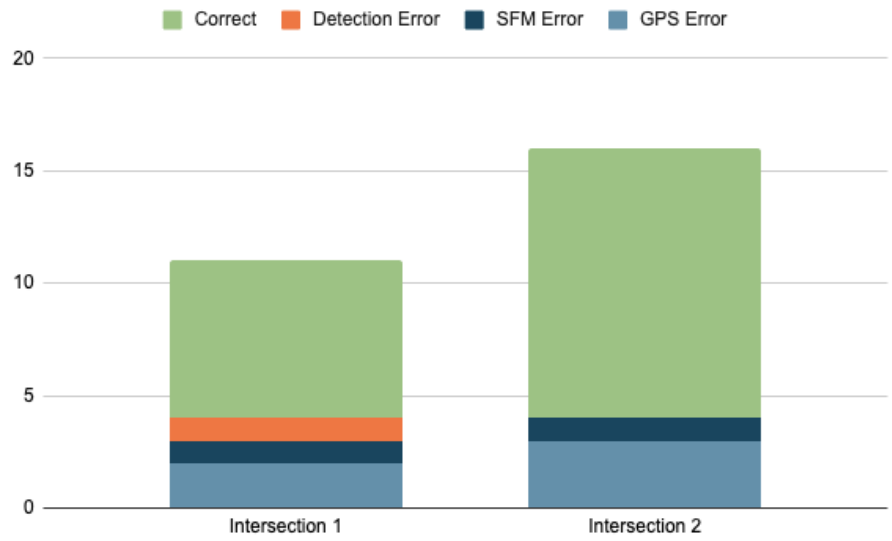


Figure 5.22: Two intersections were monitored with the live change detection system over two weeks. These two intersections exhibited changed crosswalks, and the system was able to provide correct predictions the majority of the time. The number of incorrect predictions and the corresponding reasons are shown, with GPS errors contributing the greatest number of errors, followed by structure-from-motion (SFM) errors and detection errors. The difference in the number of evaluations for each intersection is due to dropped image clusters by the Gabriel system.

5.3 Analysis and Limitations

As the performance shows, the BEV change method provides accurate and interpretable results on both single and multiple sourced images. We have been able to deploy the system in the real world. Here are the issues that need to be addressed to make it better. From a study of over 66 change determinations across the 17 different scenes received from the bus, there are three main sources of error preventing 100

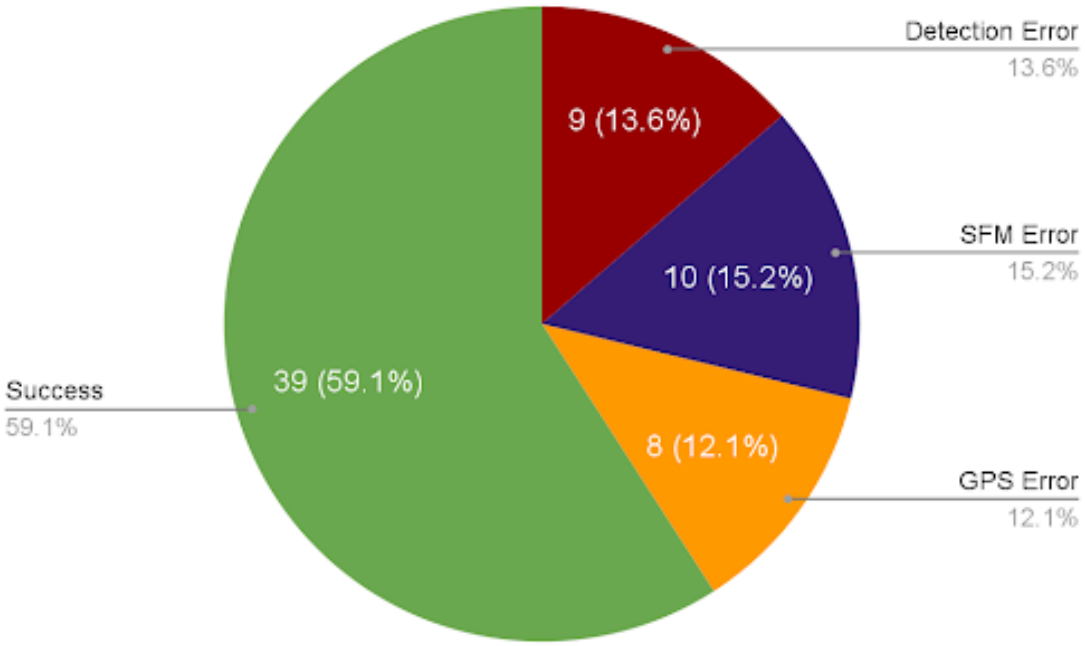


Figure 5.23: Sources of error from applying the change detector on replayed bus data at 66 scenes across four days. Sources of error can be separated into structure-from-motion (SFM), GPS, and detection errors. Success means a correct determination of change or no change at the scene.

5.3.1 GPS Failure

For known crosswalk scenes, the bus client relies on accurate GPS information in order to cluster images and send them to the cloudlet server. There is one particular street in downtown where both sides of the street are lined with skyscrapers. GPS

is already known to only be accurate up to 10 meters, but the GPS signal when surrounded by skyscrapers can bounce off of buildings, causing greater noise and an error up to 20 meters. Because image clusters are assigned to each crosswalk scene based on the closest known GPS, if the GPS for one scene is off, the error can propagate to succeeding crosswalk scenes. This can cause images with crosswalks to appear at the beginning or at the end of the cluster, cutting off individual crosswalks that need to be verified and leading to false positive detections of crosswalk changes, as shown in Figure 5.24. This can be improved by using a Kalman Filter and visual odometry.

Figure 5.24: An example of GPS error. Two image sequences that are supposed to describe the same scene. The query images actually start after the crosswalk scene due to GPS errors and do not have any images of the crosswalks. Therefore, when comparing query and reference detections in BEV, the prediction shows a crosswalk removal prediction.

5.3.2 Structure-from-motion Failure

The BEV change detection method relies on accurate reconstruction of the scene and localization of images through structure-from-motion. Fundamental challenges

exist in the problem formulation of performing structure-from motion with dash cam images for change detection. First, because roads are uniformly textured surfaces and make up large portions of the image, images often must rely on key points found on the top of the image where static buildings exist for key point matching. This can be problematic when few structures exist besides the road. Second, because the task uses dash cam footage, there is a risk of motion blur because the vehicle can travel at high speeds. Blurring can affect the ability to obtain good feature descriptors for key points. Third, the forward motion of the vehicle can be problematic for structure-from-motion because accurate triangulation can only occur along the line of motion. Triangulation perpendicular to the line of motion can have high uncertainty. Lastly, because the goal is to detect change, convergence problems can occur if the reference and query images are too spread apart in time and significant environment changes occur. If small parts of the environment change, such as crosswalks, the changed areas ideally are excluded from the structure-from-motion as outlier points. Otherwise, the process will have difficulty reconciling points that look different but exist in the same location and cause either camera poses or the scene reconstruction to be inaccurate. The downstream effect is that with inaccurate scene reconstruction, the ground plane will not be estimated correctly, and with inaccurate image localization, 2D detections will be mapped onto the 3D ground plane incorrectly, as shown in Figure 5.25. One possible solution is to use several runs for the reference images or use images from third party sources.

5.3.3 Detection Failure

The third main source of error is failed detection. The following examples are the most common reasons for failed detections.

- False detections: other nearby road markings can be confused for crosswalks because of the elongated font. Examples are “stop” or “school”. If these are falsely detected as crosswalks in the scene, then the change detector will output these as added crosswalks.
- Limited view: there are several cases where limited view of the scene inhibits accurate BEV change detection results. For example, the camera may not capture the entire sidewalk at large intersections, like when crosswalks lie on the

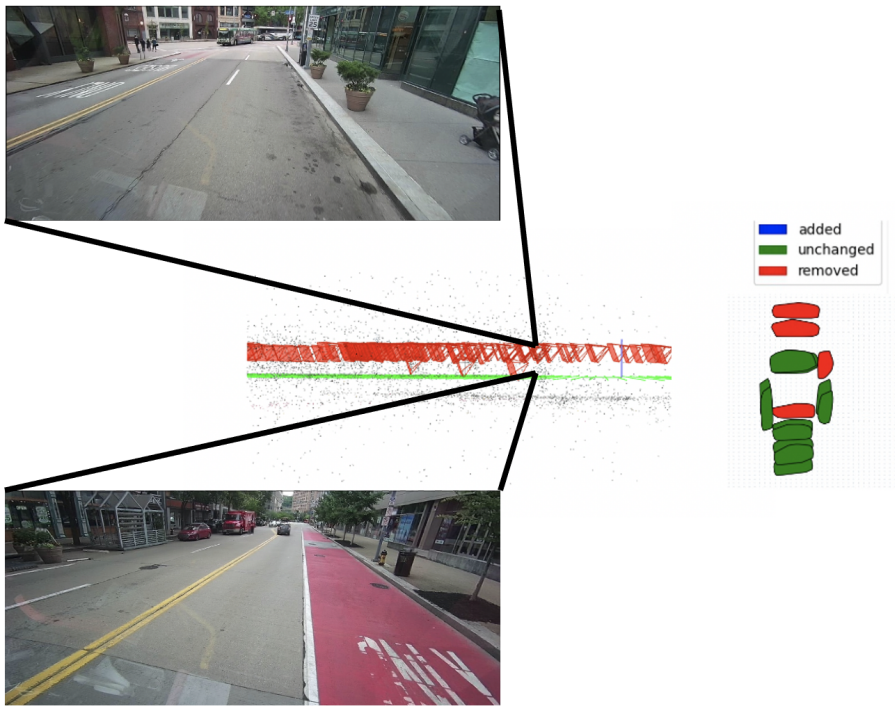


Figure 5.25: An example of structure-from-motion error. Two images from different locations are estimated to be close to each other in the reconstruction. This causes two planes of the ground to form and the plane estimator aligns only to one of them. An elevated ground plane causes misaligned detections on the ground plane and therefore false predictions of change.

left or right side of the intersection but the vehicle is driving on the other side. Another case is when crosswalks are occluded by vehicles. The BEV change detector is able to deal with temporary occlusions like when a car drives across a crosswalk, blocks the view, but then drives away, but when vehicles stop at a traffic light on top of a crosswalk and occlude the crosswalk at every frame, the detection of the crosswalk is not guaranteed and a crosswalk removal may be reported. This error may be self-corrected as mentioned in Section 5.2.4, but it will depend on the traffic amount on average.

- Difficult lighting conditions: object detectors work best when there is ambient light. Special optimization is necessary in difficult lighting conditions, which this

work leaves for future work. This includes night time vision and rain weather. Wet roads can cause vehicle reflections and bright specular reflections of vehicle headlights that can simulate the whiteness of markings. On the other hand, even on dry days when it is bright, strong lighting can cause oversaturation in parts of the image, making detection of that region impossible, as shown in Figure 5.26. This is not a big problem for our application because runs with bad weather or lighting can be skipped.

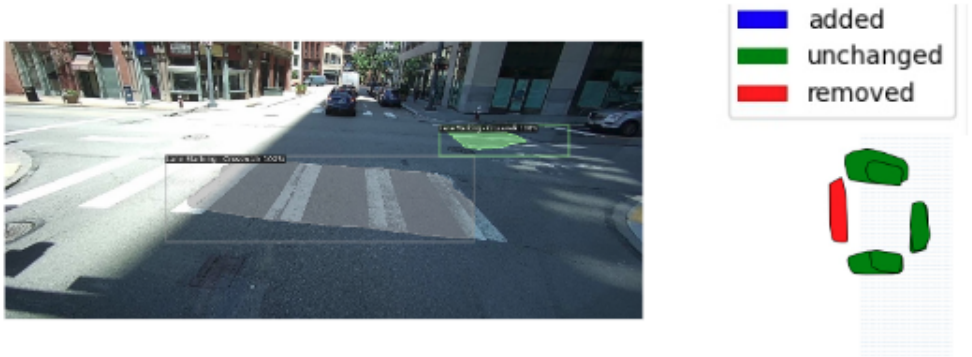


Figure 5.26: An example of a detection error due to difficult lighting conditions. The crosswalk on the left hand side is not captured well due to oversaturation of the image pixels and is therefore not detected by the model. This triggers a false crosswalk removal in the change detector.

5.3.4 Other Considerations

Ground Plane Assumption

The BEV change detector uses a mapping of detections from 2D images onto a 3D ground plane with a homography transformation. The assumption is that the ground is flat. To be more faithful to the 3D environment, more rigorous work can be done. For example, instead of using a ground plane, a ground surface could be estimated with a 3D mesh. Such a task can be more difficult because the structure-from-motion output can be noisy, but it would represent the environment more accurately. The 3D mesh could account for the slight curvature that roads have on the edges and could better represent curves of the road on hills. The road curves in three of the seventeen crosswalk scenes observed by the bus, where the angle between the true ground and the estimated plane are off by four degrees as shown in Figure 5.27. Furthermore, instead of a homography transformation of the 2D detections, ray tracing can be done to project each detection onto the ground surface mesh. Depending on how reliably the 3D ground mesh can be created, this approach could lead to more stable results.

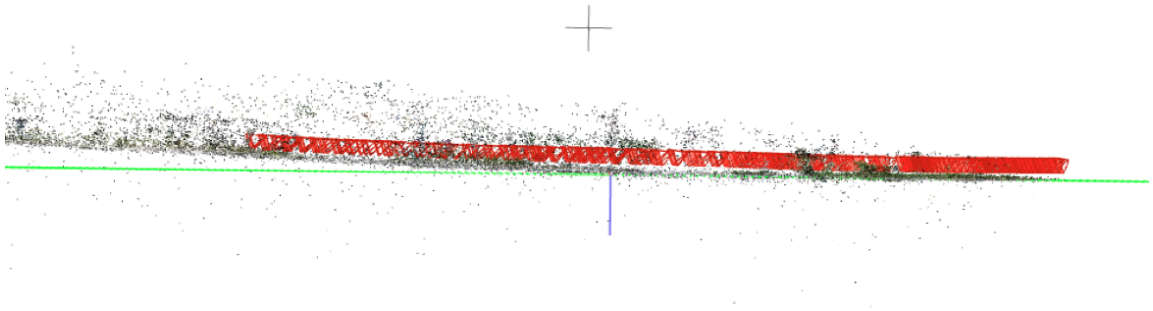


Figure 5.27: An example where the estimated plane (green) deviates from the true ground points due to changing slopes of the road.

Deteriorated Crosswalks

Many road markings experience heavy deterioration and may remain so for a significant portion of the time. Humans can often deduce upon closer inspection that a crosswalk exists by looking at the yellow mats set in place by the Americans with Disabilities Act as seen in Figure 5.28, but an object detector usually relies on the existence

of white markings for visual cues. This problem presents a challenge for the BEV change detector, since meaning exists at a location but the visual signs are not present. Currently, the detection model is not expected to accurately detect such deteriorated crosswalks, but in the future, for situations like these, it may be recommended to only perform semantic segmentations of road markings rather than classifying if certain road markings are crosswalks. This problem of detecting deteriorated crosswalks also suggests further collaboration with local municipalities to maintain road infrastructure.

Figure 5.28: An example of where paint from a plain crosswalk has eroded over time, presenting a challenge for object detectors.

Unknown Categorization

There are instances where even an experienced driver might not know how to interpret the road. One instance is where the road is repaved and there has not been enough time to draw crosswalks. The situation is ambiguous since there is no indication that the crosswalk will be redrawn or not, as shown in Figure 5.29.

Figure 5.29: An example of an ambiguous case where no crosswalk is painted but there are other indications of a crosswalk from traffic signs. This occurs after a recent road repaving, and more time is needed to determine if the traffic signs will be removed or if a crosswalk will be repainted.

5.4 Summary

In this chapter, we show that BEV crosswalk change detection can be performed with vehicle-mounted cameras. The methods work best when there is ambient light, few occlusions, and a flat road surface. We apply our method to two different scenarios, one where images are collected from different vehicles and cameras and another where images are collected from a single bus camera. Furthermore, the method was deployed live on a bus for daily feedback. Instruments that were necessary in obtaining low-latency feedback were access to an edge computer that removes more than 95% of the irrelevant images and a cellular connection between the edge computer and the server. Our methods are a significant step in scaling detections of map changes using crowd-sourced data and a cost-effective sensor suite.

Chapter 6

Conclusions

This thesis develops an approach to produce scalable methods for crosswalk detection and crowd-sourced and camera-only methods for crosswalk change detection.

6.1 Summary of Contributions

Chapter 4: CARLA Simulated Data for Object Detection The CARLA simulator is used to provide automatic annotations for street-view objects, crosswalks and fire hydrants. Images captured by a simulated vehicle-mounted camera can be used to train and deploy models for real world application such as object detection and data filtering, especially if objects have limited publicly available data. Experimental results also show that models trained on CARLA-generated annotations perform as well as those trained on 200 real-world images and CARLA-generated data can be used to supplement existing datasets for improved performance.

Chapter 5: Bus Platform for Crosswalk Detection and Monitoring A cost-effective sensor platform that enables ordinary vehicles to perform sophisticated scene analysis and change detection and reduce reliance on manual monitoring and expensive mapping vehicles is presented. The approach accumulates the same object appearing in multiple frames and compactly represents the scene in the bird's-eye-view. It also accounts for 3D information by taking advantage of multi-view geometry rather than relying on LiDAR sensors. The approach is evaluated on data collected at multiple locations in Pittsburgh with images collected from multiple vehicles and

a single bus vehicle that actively collects data. The method is customized to live bus data and automates search for reference images based on GPS, direction, and crosswalk detection and provides same-day feedback.

6.2 Future Work

Future work includes strengthening the robustness of the BEV change detector and expanding its capabilities. For example, in the open world, the bus will observe intersections at non-optimal times. Situations include when crosswalks are heavily occluded by vehicles or are poorly imaged by the camera on rainy or sunny days. A classifier is needed to discard these situations to prevent false positive reports of change. Furthermore, more analysis needs to be done on how the methods can perform on other objects. The most logical step is to apply the method to text markings on the road and then to lane markings, though lane markings can be more challenging because they are markings with no bounds. More longitudinal analysis is also needed of this system's change prediction at more crosswalk intersections on the bus route. Lastly, the goal is to expand the system to multiple buses to expand the coverage of map change detection.

Bibliography

[1] https://cocodataset.org/#detection-eval. 4.2.1, 5.2.1

[2] Syed Ammar Abbas and Andrew Zisserman. A geometric approach to obtain a bird's eye view from an image. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4095–4104. IEEE, 2019. 3.1

[3] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79, 2009. doi: 10.1109/ICCV.2009.5459148. 1, 2.4

[4] Pablo F Alcantarilla and Simon Stent. Street-View Change Detection with Deconvolutional Networks. page 10. 3.2.1

[5] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018. URL http://arxiv.org/abs/1812.03079. 1

[6] Steve Branson, Jan Dirk Wegner, David Hall, Nico Lang, Konrad Schindler, and Pietro Perona. From google maps to a fine-grained catalog of street trees. *ISPRS Journal of Photogrammetry and Remote Sensing*, 135:13–30, 2018. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2017.11.008. URL https://www.sciencedirect.com/science/article/pii/S0924271617303453. 3.2.1

[7] Tom Bu, Xinhe Zhang, Christoph Mertz, and John M. Dolan. Carla simulated data for rare road object detection. In *Proceedings of IEEE International Intelligent Transportation Systems Conference (ITSC '21)*, pages 2794 – 2801, September 2021. 4

[8] 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. In *CVPR*, 2020. 3.1

[9] 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. 3.1

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. URL http://arxiv.org/abs/1606.00915. 2.1

[11] Kevin Christensen. Computer vision for live map updates. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, June 2019. 2.5, 3.2.3

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 4.2.1

[13] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 337–33712, 2018. doi: 10.1109/CVPRW.2018.00060. 2.3

[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, November 2017. 2.1.1, 4.1.2

[15] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/, 2016. 5.1.2

[16] Abhishek Dutta and Andrew Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6889-6/19/10. doi: 10.1145/3343031.3350535. URL https://doi.org/10.1145/3343031.3350535. 5.1.2

[17] D. Dwibedi, I. Misra, and M. Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1310–1319, 2017. doi: 10.1109/ICCV.2017.146. 2.1.1, 4.1.1, 4.2.4

[18] Andreas Geiger. Monocular road mosaicing for urban environments. In *2009 IEEE Intelligent Vehicles Symposium*, pages 140–145, 2009. doi: 10.1109/IVS.2009.5164267. 2.2

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014. doi: 10.1109/CVPR.2014.81. 2.1

[20] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015. 2.1

[21] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2.1.1

[22] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, page 68–81, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327930. doi: 10.1145/2594368.2594383. URL https://doi.org/10.1145/2594368.2594383. (document), 2.5, 5.14, 5.1.4, 5.1.6

[23] Mordechai (Muki) Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7:12–18, 2008. 5.1.4

[24] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition, 2003. ISBN 0521540518. (document), 2.3

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. 4.1.4, 5.1.2

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322. 2.1, 5.1.2, 5.1.2

[27] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 449–468, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01234-2. 1

[28] Hiroto Honda. Digging into detectron 2, Jan 2022. URL https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd. (document), 4.3

[29] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 746–753. IEEE, 2017. 2.1.1

[30] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *CoRR*, abs/1801.00868, 2018. URL http://arxiv.org/abs/1801.00868. 2.1, 5.1.3, 5.1.3

[31] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic Feature Pyramid Networks. In *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR), June 2019. (document), 2.1

[32] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2.1

[33] John Lambert and James Hays. Trust, but verify: Cross-modality fusion for HD map change detection. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=cXCZnLjDm4s. 1, 1.1, 2.2, 3.2.2, 3.3

[34] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework, 2021. 3.1

[35] Justin Liang and Raquel Urtasun. End-to-end deep structured models for drawing crosswalks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1.1

[36] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. doi: 10.1109/CVPR.2017.106. 4.1.4, 5.1.2

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and Larry Zitnick. Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2014. 4.1, 4.2, 5.1.3, 5.1.3

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2.1

[39] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. 2.3

[40] Julieta Martinez, Sasha Doubov, Jack Fan, loan Andrei Bârsan, Shenlong Wang, Gellért Máttyus, and Raquel Urtasun. Pit30M: A Benchmark for Global Localization in the Age of Self-Driving Cars. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4477–4484, October 2020. doi: 10.1109/IROS45743.2020.9340924. ISSN: 2153-0866. 1

[41] Kevin Matzen and Noah Snavely. Scene chronology. In *Proc. European Conf. on Computer Vision*, 2014. 1, 2.4, 3.2.2, 3.3

[42] Suraj M.S., Hugo Grimmett, Lukáš Platinský, and Peter Ondrúška. Visual vehicle tracking through noise and occlusions using crowd-sourced maps. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*,

pages 4531–4538, 2018. doi: 10.1109/IROS.2018.8593378. 1, 2.4

[43] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017. URL https://www.mapillary.com/dataset/vistas. 4.1, 4.2, 5.1.2

[44] Department of Transportation. Overview of motor vehicle crashes in 2020. URL https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812580. 1

[45] U.S. Department of Transportation. Strong economy has americans driving more than ever before, . URL https://www.fhwa.dot.gov/pressroom/fhwa1905.cfm. 1

[46] U.S. Department of Transportation. Highway statistics series, . URL https://www.fhwa.dot.gov/policyinformation/statistics/2020/mv1.cfm. 1

[47] Emanuele Palazzolo and Cyrill Stachniss. Fast image-based geometric change detection given a 3d model. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6308–6315, 2018. doi: 10.1109/ICRA.2018.8461019. 3.2.2

[48] Thomas Pollard and Joseph L. Mundy. Change detection in a 3-d world. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, 2007. doi: 10.1109/CVPR.2007.383073. 3.2.2

[49] Lorenzo Porzi, Samuel Rota Bulò, and Peter Kontschieder. Improving panoptic segmentation at all scales. In *Computer Vision and Pattern Recognition (CVPR)*, June 2021. 1.1

[50] Lauri Rapo. Generating road orthoimagery using a smartphone, 2018. 2.2

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91. 2.1

[52] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 2.1, 4.1.4, 5.1.2

[53] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 102–118, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46475-6. (document), 2.1.1, 2.2

[54] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations

from images using pyramid occupancy networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2.1, 3.1

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597. 2.1

[56] 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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, Las Vegas, NV, USA, June 2016. IEEE. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.352. 2.1.1

[57] Ken Sakurada and T. Okatani. Change detection from a street image pair using cnn features and superpixel segmentation. In *BMVC*, 2015. 3.2.1

[58] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 2.3

[59] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2.3

[60] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (document), 2.3, 2.4, 4.1.1, 5.1.3

[61] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. (document), 2.3, 2.4, 5.1.3

[62] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. URL https://www.ros.org. 5.1.1

[63] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 2.1.2

[64] Srivatsan Varadharajan, Sobhagya Jose, Karan Sharma, Lars Wander, and Christoph Mertz. Vision for road inspection. In *IEEE Winter Conference on Applications of Computer Vision*, pages 115–122, 2014. doi: 10.1109/WACV.2014.6836111. 3.2.3

[65] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE Winter Conference on Applications of Computer*

Vision (WACV), pages 748–756. IEEE, 2018. doi: 10.1109/WACV.2018.00087. 2.1

[66] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017. doi: 10.1109/ICIP.2017.8296962. 2.1

[67] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 4.1.4, 4.2.1, 5.1.2

[68] Weihuang Xu, Nasim Souly, and Pratik Prabhanjan Brahma. Reliability of gan generated data to train and validate perception systems for autonomous vehicles. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pages 171–180, January 2021. 2.1.1

[69] Binh Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *CoRL*, volume abs/2012.11704, 2018. 1

[70] Canbo Ye. Busedge: Efficient live video analytics for transit buses via edge computing. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, July 2021. (document), 2.5, 5.14, 5.1.4, 5.1.6

[71] Hengrun Zhang, Ming Yang, Chunxiang Wang, Xinhua Weng, and Lei Ye. Lane-level orthophoto map generation using multiple onboard cameras. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pages 855–860, 2014. doi: 10.1109/ROBIO.2014.7090439. 2.2