SKOPE3D: A Synthetic Keypoint Perception 3D Dataset for Vehicle Pose Estimation

by

Himanshu Pahadia

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2023 by the
Graduate Supervisory Committee:

Yezhou Yang, Chair
Duo Lu
Mohammad Farhadi

ARIZONA STATE UNIVERSITY

May 2023

ABSTRACT

Intelligent transportation systems (ITS) are a boon to modern-day road infrastructure. It supports traffic monitoring, road safety improvement, congestion reduction, and other traffic management tasks. For an ITS, roadside perception capability with cameras, LIDAR, and RADAR sensors is the key. Among various roadside perception technologies, vehicle keypoint detection is a fundamental problem, which involves detecting and localizing specific points on a vehicle, such as the headlights, wheels, taillights, etc. These keypoints can be used to track the movement of the vehicles and their orientation.

However, there are several challenges in vehicle keypoint detection, such as the variation in vehicle models and shapes, the presence of occlusion in traffic scenarios, the influence of weather and changing lighting conditions, etc. More importantly, existing traffic perception datasets for keypoint detection are mainly limited to the frontal view with sensors mounted on the ego vehicles. These datasets are not designed for traffic monitoring cameras that are mounted on roadside poles. There's a huge advantage of capturing the data from roadside cameras as they can cover a much larger distance with a wider field of view in many different traffic scenes, but such a dataset is usually expensive to construct.

In this research, I present SKOPE3D: Synthetic Keypoint Perception 3D dataset, a one-of-its-kind synthetic perception dataset generated using a simulator from the roadside perspective. It comes with 2D bounding boxes, 3D bounding boxes, tracking IDs, and 33 keypoints for each vehicle in the scene. The dataset consists of 25K frames spanning over 28 scenes with over 150K vehicles and 4.9M keypoints. A baseline keypoint RCNN model is trained on the dataset and is thoroughly evaluated on the test set. The experiments show the capability of the synthetic dataset and knowledge transferability between synthetic and real-world data.

DEDICATION

Dedicated to my loving parents, family, and friends for their love, patience, and faith in me on this short journey and many more to come...

ACKNOWLEDGMENTS

Working on this thesis was a challenging yet fulfilling journey for me. It involved formulating a research problem, exploring numerous solutions, attempting to make the chosen solution work, and testing it.

I would like to express my deepest gratitude to my advisor, Dr. Yezhou Yang, for his invaluable guidance throughout my master's journey as this would not have been possible without his support. Special thanks to Dr. Duo Lu for conducting regular discussions with me, his insights and feedback have been instrumental in shaping my thesis work. I want to thank Dr. Mohammad Farhadi for helping me navigate through numerous challenges of graduate study and giving me the necessary exposure that helped me in my career.

I would also like to acknowledge the committee members, Dr. Yezhou Yang, Dr. Duo Lu, and Dr. Mohammad Farhadi, for their time, effort, and valuable feedback on my work.

I would like to thank my friends and family for their unwavering support and encouragement throughout my academic journey. I would like to express gratitude towards my parents and my brother, Abhishek Pahadia, for always believing in me. I would like to thank my friend, Sahasra Iyer, for supporting me throughout my graduate journey. Their belief in me has been a constant source of motivation and inspiration. This would not have been possible without them.

Finally, I would like to thank the University and its faculty for providing me with opportunities, resources, and support.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

As the number of vehicles on the road continues to increase, the capacity of current transportation networks and infrastructure is approaching saturation (Won *et al.*, 2016). This leads to traffic congestion, which is a major problem in many countries. In response, researchers are developing intelligent transportation systems that incorporate traffic monitoring that enables smart decision-making by the authorities and improves traffic scenarios. Traffic monitoring systems aim to improve the safety and efficiency of our roads by detecting and identifying vehicles, pedestrians, and other objects in real time using a variety of perception-based sensors like cameras, LIDAR, and RADAR. Advanced computer vision algorithms analyze traffic scenes and provide real-time information to drivers and transportation agencies. Traffic cameras are widely deployed today to monitor traffic conditions, particularly at intersections.

Deep learning has revolutionized the field of traffic monitoring, enabling advanced research in classification, segmentation, localization, and scene understanding. It allows the traffic monitoring systems to automate various tasks including vehicle and pedestrian detection, tracking (Fedorov *et al.*, 2019), and re-identification (Khan and Ullah, 2019). These tasks are mostly performed in the 2D image space. Among them, pose estimation and keypoint detection have emerged as active research areas in recent years. These techniques can be applied to a wide range of use cases, including human pose estimation (HPE) and vehicle pose estimation.

Human pose estimation is the process of detecting and localizing human body parts such as arms, legs, and joints, in an image or video. This technology finds applications in various fields, including motion capture (Desmarais *et al.*, 2021), gaming,

and human-robot interaction. With recent advancements in deep learning, 2D pose estimation has reached a detection rate above 90% for all human joints (Newell *et al.*, 2016). This is made possible by convolutional neural networks and access to large-scale datasets with annotated human keypoints. However, human pose estimation suffers from the significant challenge of handling keypoint occlusion.

Vehicle pose estimation, on the other hand, remains a complex and challenging task, due to the diverse range of vehicle types, colors, shapes, and sizes(Gupta *et al.*, 2021). Vehicle detection (Felzenszwalb *et al.*, 2009) (Girshick *et al.*, 2014) (He *et al.*, 2017) (Ren *et al.*, 2017), tracking (Choi, 2015) (Wang and Fowlkes, 2017) (Xiang *et al.*, 2015) (Zhang *et al.*, 2008), and reconstruction (Zia *et al.*, 2013) (Kar *et al.*, 2015) are heavily researched focus areas that have seen significant improvements due to the advent of deep learning (Reddy *et al.*, 2018). Particularly, the detection of vehicle parts such as wheels, headlights, doors, etc. across various views is being improved on a daily basis. However, precise regression of the vehicle's pose remains an outstanding challenge. One more reason for the inability of neural networks to regress vehicle pose accurately is the lack of publicly available 3D keypoint datasets. Existing datasets are mostly limited to frontal views captured from sensors mounted on top of ego vehicles. These frontal view datasets mainly target autonomous driving use cases. Another critical challenge is occlusion and partial occlusion, which limits the generalization of keypoint detection. The frontal-view datasets are vulnerable to occlusion and low perceptual range. On the contrary, datasets captured from roadside cameras provide a promising solution to these challenges. The strength of data captured from these roadside cameras comes from their robustness to occlusion and their ability to predict long-term events with a wider field of view (Ye *et al.*, 2022).

1.1 Motivation and Contribution

After an extensive exploration of numerous 3D datasets for vehicle perception, it became apparent that the majority of these datasets suffer from either a lack of keypoint annotations or an inadequate annotation scheme. Numerous vehicle perception 3D datasets were explored and found that most of the datasets have various issues related to keypoint detection. Notably, several datasets, such as KITTI (Geiger *et al.*, 2012), Lyft Level 5 (Kesten *et al.*, 2019), A2D2 (Geyer *et al.*, 2020), and ArgoVerse (Chang *et al.*, 2019), lack keypoint annotations altogether. Furthermore, datasets such as CarFusion (Reddy *et al.*, 2018), CityFlow (Tang *et al.*, 2019), Apollocar3d (Song *et al.*, 2019), and VeRi-776 (Wang *et al.*, 2017b), which do include keypoint annotations, fail to provide a good roadside perspective of intersections. The limitation results in challenges such as occlusion and a low field-of-view (FOV). Training on such datasets is unlikely to lead to a model that can accurately generalize 3D keypoint detection to a roadside traffic camera perspective. To address these shortcomings, 3D environment simulator CARLA (Dosovitskiy *et al.*, 2017) was leveraged to generate a large dataset that has various 3D/2D annotations, including 33 keypoints, 3D bounding box, 2D bounding box, and tracking ID for all the vehicles in the scene.

The main contributions of the thesis are below:

1. This thesis presents a one-of-its-kind synthetic perception dataset, named SKOPE3D (Synthetic keypoint perception 3D dataset), which contains 3D keypoint annotations of vehicles from a roadside camera perspective. The scenes in the dataset were carefully selected to include a range of weather conditions, lighting scenarios, road types, and camera viewpoints (as shown in Figure 1.1).

    - The dataset includes the following annotations - 33 3D keypoints, 3D and

2D bounding boxes, model type (sedan, SUV, hatchback, etc.) and tracking ID for each car within a 70 m radius.

2. Additionally, data extension modules for CARLA Simulator have been developed, enabling the dataset to be extended with new scenes, vehicles, and annotations.

3. KeyPoint RCNN, an extension of Mask R-CNN (He *et al.*, 2017), has been analyzed on the SKOPE3D dataset, exploring the transferability of knowledge between synthetic and real-world datasets.

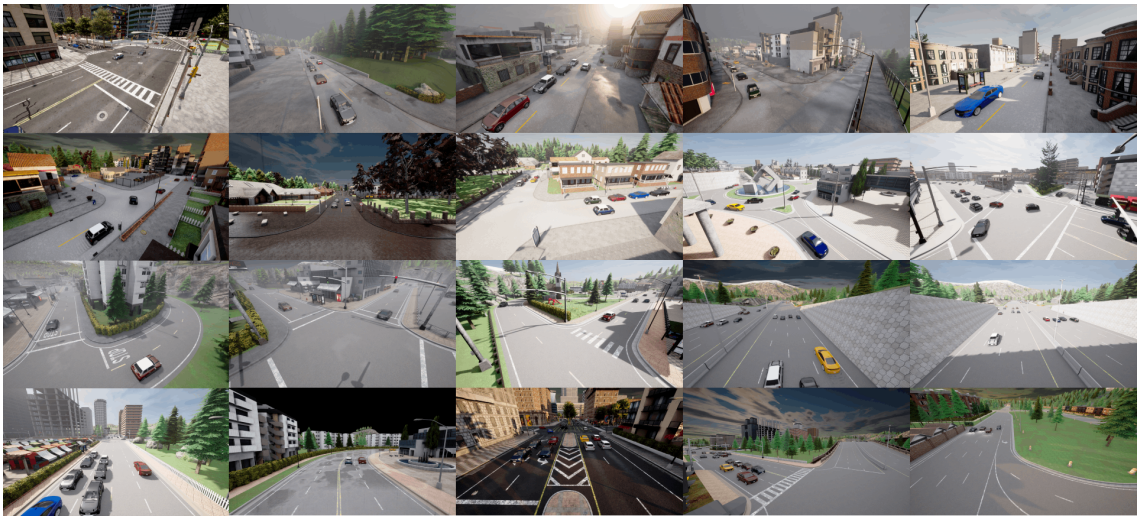

Figure 1.1: Snapshots of Scenes from the Skope3d Dataset

Chapter 2

BACKGROUND

The field of autonomous driving and traffic monitoring has seen significant advances with the advent of deep learning and large-scale datasets. While significant progress has been made in object detection, tracking, segmentation, and classification tasks, keypoint detection for vehicle pose estimation remains an understudied area. Unlike human pose estimation, keypoint detection for vehicle pose estimation is still in its nascent stage, primarily due to the scarcity of datasets with keypoint annotations for vehicles. Additionally, there is a lack of datasets from a roadside perspective, with most available datasets providing only a frontal-view perspective. This gap in the availability of datasets for keypoint detection from different perspectives has hindered progress in this area.

2.1 Deep Neural Networks

Deep learning has experienced a significant boost in recent years due to its ability to recognize patterns and make predictions from vast amounts of data. Deep learning models are built using artificial neural networks, which are designed to mimic the structure and function of the human brain. Artificial neural networks (ANN) (Goodfellow *et al.*, 2016) consist of connected perceptrons, also called neurons, which enable a series of mathematical operations such as summation, product, and activation functions. By feeding data into an ANN model, it can recognize the patterns and adjust its hyperparameters accordingly. ANNs can be arranged in various configurations to optimize performance.

2.1.1 Convolutional Neural Networks

The advances in computer vision tasks such as object detection and semantic segmentation have been driven by the development of powerful neural networks like Convolutional Neural Networks (CNN) (LeCun *et al.*, 1998). CNNs are specialized neural networks that are designed to automatically learn hierarchical representations of image data by applying convolutional filters to the input image. These filters are used to extract features, such as edges, corners, and textures, from the image at different scales, which are then used to make predictions.

CNNs use a set of convolutional filters to scan the input image and extract meaningful features, such as edges, corners, and textures. These filters are typically small in size and are applied to the entire image using a sliding window approach. Multiple convolutional filters are used to learn hierarchical representations, starting from low-level features such as edges and gradually building up to higher-level features such as object shape.

After the convolutional layers, the network uses pooling layers to reduce the dimensionality of the output. These layers reduce the size of the feature maps and make the network more efficient. Common pooling operators include max pooling and average pooling, which downsample the data by taking the maximum or average of a local neighborhood patch, respectively.

CNNs have shown state-of-the-art performance on several computer vision tasks, including image classification, object detection, and semantic segmentation. They can be applied to various domains, including traffic monitoring, age/gender classification, medical imaging, robotics, self-driving cars, and object re-identification, among others. This is made possible due to the availability of large datasets, such as ImageNet (Deng *et al.*, 2009), NIH Chest X-ray (Wang *et al.*, 2017a), and the KITTI dataset

(Geiger *et al.*, 2012), among others.

2.1.2 Region-CNN

Object detection is a computer vision task that involves identifying and localizing objects of interest in an image or a video. The objective is to output a set of bounding boxes, each corresponding to the detected object in the scene along with its class. There are various approaches to object detection, a popular one being Region-CNN (R-CNN) (Girshick *et al.*, 2014). RCNN combines a region proposal network (RPN) with a CNN to generate candidate object proposals and classify them. Another approach is Single Shot Detector (SSD), which can directly predict the class probabilities and bounding boxes for each object in the image.

R-CNN uses a selective search (Uijlings *et al.*, 2013) to generate approximately 2000 region proposals or bounding boxes for image classification. Each of these proposed regions is passed to the convolutional neural network for image classification. Once the CNN classifies each patch, each bounding box can be refined using regression.

Several updates to this R-CNN model, such as Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren *et al.*, 2015), and Mask R-CNN (He *et al.*, 2017), have improved upon R-CNN. These new approaches are conceptually intuitive, flexible, and robust, with faster training and inference times.

2.1.3 Mask R-CNN

Mask R-CNN (He *et al.*, 2017) is a variant of the Faster R-CNN model that enables object detection and pixel-level mask generation for each detected object. This allows for better object segmentation, making it a valuable task in instance segmentation and object tracking. Mask R-CNN has a two-stage architecture (shown in Figure 2.1

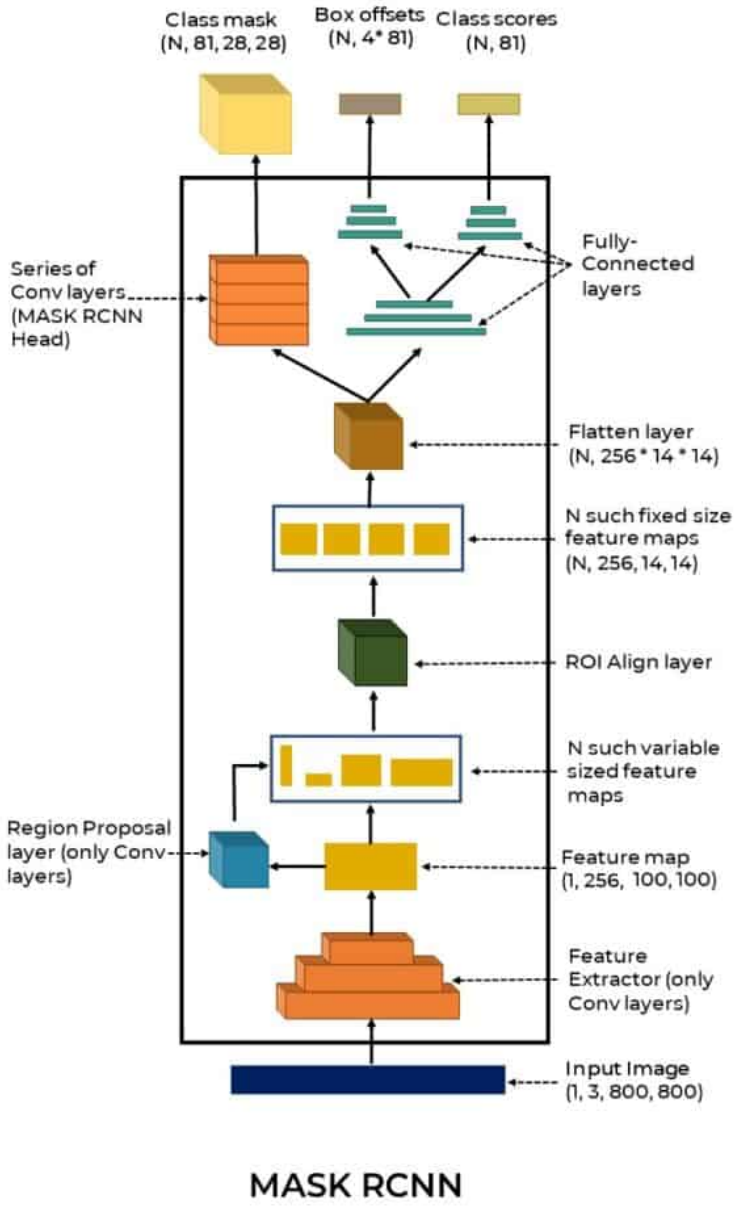

Figure 2.1: Mask R-CNN Architecture (Image Courtesy: Patil (2021))

taken from Patil (2021)) -

1. Region proposals - A Region Proposal Network (RPN) generates the region proposals or candidates of possible objects in the image.

2. Convolutional Neural Network - For each region proposal, the feature maps proposed are RoI pooled according to the region and go through the remainder of the network, extracting the class, bounding box, and the binary mask.

The key contribution of Mask R-CNN is the introduction of a new RoIAlign layer, which addresses the misalignment problem between the feature map and the object proposal caused by the RoIPool layer in Faster R-CNN. RoIAlign uses bilinear interpolation to extract features from the feature map at pixel-level accuracy.

Mask R-CNN outperformed most state-of-the-art models in instance segmentation tasks on COCO, Cityscapes, and Pascal VOC when it was introduced in 2017.

The authors also showed how Mask R-CNN can be extended to the Keypoint R-CNN - a model that enables keypoint detection.

2.1.4 Keypoint R-CNN

Keypoint R-CNN is a classic keypoint detector introduced alongside Mask R-CNN (He *et al.*, 2017). The Mask R-CNN can be easily extended to the pose estimation network using the keypoint detection module. The architecture (shown in Figure 2.2) is exactly the same as Mask R-CNN, differing only in the output size and the way the keypoints are encoded in the keypoint mask. It is mainly used for and excels at human keypoint detection. MS-COCO (Lin *et al.*, 2014) dataset offers various annotations of everyday objects like image recognition, segmentation, and captioning. It also comes with human keypoint annotations consisting of 17 keypoints for people, including the head, neck, shoulders, elbows, wrists, hips, knees, and ankles (Figure 2.3 taken from Patil (2021))). Keypoint R-CNN is extensively trained and studied on this dataset.

Keypoint R-CNN modifies the Mask R-CNN, by one-hot encoding a keypoint instead of the whole mask of the detected object. The input of Keypoint R-CNN differs from the Mask R-CNN as shown in figure 2.4.

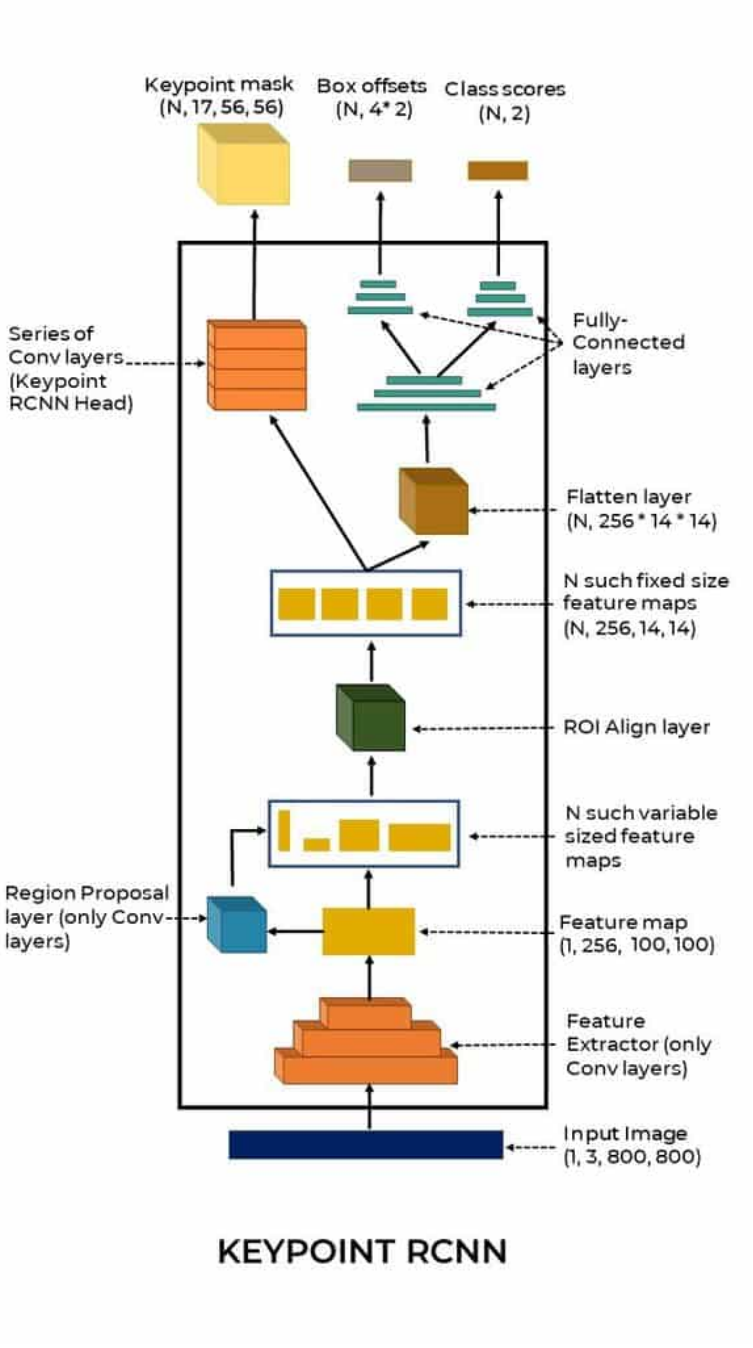

Figure 2.2: Keypoint R-CNN Architecture (Image Courtesy: Patil (2021))

Keypoint R-CNN incorporates an additional branch in the network that predicts the keypoint locations for objects in addition to their bounding boxes and class predictions. This branch is a fully convolutional network that outputs a heatmap for

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle

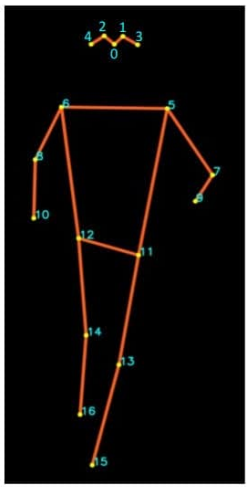

Figure 2.3: Human Keypoint Annotation from MS-COCO Dataset (Image Courtesy: Patil (2021))

each keypoint, which indicates the likelihood of the keypoint being present at each location in the image.

2.2 Keypoint detection

Keypoints, also known as key features, are specific points or regions of an object that are of interest and can be used to detect objects, recognize objects, or estimate their pose. Keypoint detection is an essential building block for various computer vision tasks, such as pose estimation, simultaneous localization and mapping (SLAM), structure from motion, and camera calibration. Keypoint detection has been used even before deep learning became mainstream. There are various industry-wide applications that use keypoint detection algorithms, such as FAST, SIFT, and ORB, among others. Additionally, keypoint detection has been researched using various deep-learning techniques.

Huge advances have been made in human pose benchmarks. Human keypoint detection can be used for a wide variety of applications, including Pose estimation

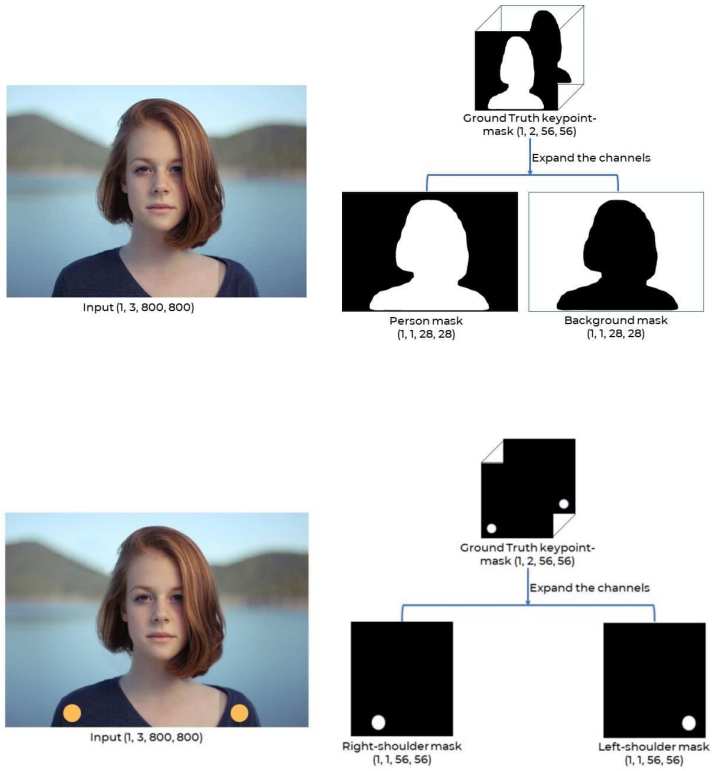

Figure 2.4: Mask R-CNN Class-wise Output Feature Map (Top) and Keypoint Encoding in Output Mask (Bottom) (Image courtesy: Patil (2021))

(Xiao *et al.*, 2018), gender classification (Barra *et al.*, 2019), violence recognition (Soliman *et al.*, 2019), hand gesture recognition (Simon *et al.*, 2017), face pose estimation (Barra *et al.*, 2020), and more.

Recently, vehicle pose estimation has also drawn some attention. Vehicle pose estimation can be used in a variety of applications, such as pose detection (Sánchez *et al.*, 2020), enhancing instance segmentation (Murthy *et al.*, 2017), vehicle tracking (Feng *et al.*, 2021), accurate speed estimation (Llorca *et al.*, 2016), vehicle re-identification (Wang *et al.*, 2017b), 3D scene reconstruction, traffic surveillance (Zhang *et al.*, 2020) etc.

Keypoint prediction methods can be divided into two different categories:

- **Top-down approach:** In this approach, first, all the instances are detected in the given image with the help of an external object detector like Faster R-CNN (Ren *et al.*, 2015), Feature Pyramid Networks (Lin *et al.*, 2017) or YOLO (Redmon and Farhadi, 2018). Then keypoints locations are regressed on each of the instances. There are various methods that follow this approach, including, Mask R-CNN (He *et al.*, 2017), Simple baseline for human pose estimation (Xiao *et al.*, 2018), CNN with deep supervision of hidden layers (Li *et al.*, 2018), Stacked hourglass (Newell *et al.*, 2016), Occlusion-net (Reddy *et al.*, 2019), etc.

- **Bottom-up approach:** In this approach, first, all the keypoints are detected in the given image and then reconstruct each instance associates the detected keypoints. There are various methods that use this approach for human pose detection like PifPaf (Kreiss *et al.*, 2019), Pose estimation using part affinity fields (Cao *et al.*, 2017), Associative embedding (Newell *et al.*, 2017), Deepcut (Pishchulin *et al.*, 2016), Deepercut (Insafutdinov *et al.*, 2016), etc.

2.3 Dataset overview

Due to active research in autonomous driving and traffic monitoring, there's an increase in the number of large-scale traffic scene datasets (Table 2.1). These datasets are divided into two types -

1. Frontal-view autonomous driving datasets

2. Roadside perspective monitoring datasets

Frontal-view datasets are primarily used in autonomous driving use cases. The KITTI Vision Benchmark Suite, released in 2012, was one of the first datasets for

Table 2.1: Comparison of 3d Autonomous Driving and Traffic Monitoring Datasets. The Datasets Are Divided Based on Their View Type: Frontal and Roadside. 2d Boxes Indicate Datasets That Only Have 2d Annotations. 'Y' Denotes Available, '/' Denotes an Unknown Value, While '-' Denotes That the Information Is Unavailable.

View	Dataset	RGB Frames	Scenes	LIDAR	Keypoints	3D Boxes	2D Boxes	RGB Resolution	Year	Diversity
Frontal	KITTI (Geiger *et al.*, 2012)	15K	22	Yes	-	80K	80K	1392x512	2013	
	ApolloScape (Huang *et al.*, 2019)	144K	/	Yes	-	70K		3384x2710	2019	Night
	Lyft Level 5 (Kesten *et al.*, 2019)	46K	366	Yes	-	1.3M		1920x1080	2019	
	A2D2 (Geyer *et al.*, 2020)	12K	/	Yes	-	9K		1928x1208	2019	-
	H3D (Patil *et al.*, 2019)	27.7K	160	Yes	-	1M		1920x1200	2019	-
	Argoverse (Chang *et al.*, 2019)	22K	113	Yes	-	993K		1920x1200	2019	Rain, Night
	CityScapes (Gählert *et al.*, 2020)	5K	1150	Yes	-	27K		2048x1024	2019	-
	nuScenes (Caesar *et al.*, 2020)	1.4M	1000	Yes	-	1.4M		1600x900	2019	Rain, Night
	Waymo Open (Sun *et al.*, 2020)	230K	1150	Yes	-	12M	9.9M	1920x1080	2020	Rain, Night, Dawn
	Apollocar3D (Song *et al.*, 2019)	5.5K	22	Yes	Y	60K		3384x2710	2018	-
Roadside	BoxCars116K (Sochor *et al.*, 2018)	116K	137	No	-	116K		128x128	2020	-
	Rope3D (Ye *et al.*, 2022)	50K	26	Yes	-	1.5M		1920x1080	2022	Rain, Night, Dawn
	CarFusion (Reddy *et al.*, 2018)	54K	13	No	Y	100K	100K	1920x1080	2018	-
	CityFlow-ReID (Tang *et al.*, 2019)	50K	40	No	Y	229K		/	2019	-
	VeRi-776 (Wang *et al.*, 2017b)	50K	20	No	Y	40K		133x152	2020	-
Roadside	SKOPE3D (current)	25K	28	No	4.9M	151K	151K	1920x1080	2023	Rain, Night, Dawn, Evening

self-driving use cases. It provided multimodal data and opened up various challenges in the field of autonomous driving. There are other large-scale datasets that took inspiration from KITTI and improved the quality of autonomous driving datasets like H3D, ApolloScape and ApolloCar3D, Waymo Open Dataset, Argoverse, Lyft Level 5, A2D2, and many more. These datasets lead to advanced research in the field of autonomous driving, but they were from an ego-vehicle view perspective.

Roadside perspective datasets, on the other hand, can help with 3D localization from a roadside surveillance camera perspective (Altekar *et al.* (2021), Lu *et al.* (2021)) . However, the availability of such datasets is limited. A few datasets in this category are CAROM Air, BoxCars (Sochor *et al.*, 2018), Rope3D (Ye *et al.*, 2022), and PASCAL3D+ (Xiang *et al.*, 2014) dataset. Most of these datasets lack keypoint annotations for the vehicles in the scene. The ApolloCar3D dataset has keypoint an-

notations but it's from a frontal-view perspective as shown in the right-side image in Figure 2.5. The CarFusion (Reddy *et al.*, 2018) dataset has keypoint annotations but it was recorded by people holding their smartphones while walking on the sidewalk as visible from the Top-Left side image in Figure 2.5. Therefore, it cannot be considered a traffic monitoring dataset. The veri-776 dataset (Wang and Fowlkes, 2017) has 133x152 cropped image patches of vehicles from the CityFlow dataset (Tang *et al.*, 2019) with keypoint annotations (two still frames shown on Bottom-Left corner of Figure 2.5). It doesn't have annotations on the entire scene similar to ApolloCar3D.

Figure 2.5: Still Frames from Datasets with Keypoint Annotations

2.4 Unreal Engine 4

Unreal Engine 4 is a cutting-edge game engine developed by Epic Games that is widely used in the video game industry, as well as in other fields such as architecture, film and television, and virtual reality. It's renowned for its real-time, ultra-realistic, and high-fidelity synthetic simulations.

Unreal Engine supports C++ and its own visual scripting library called Blueprint, which enables game developers to create complex game logic with ease.

Due to its high-quality immersive experiences, advanced graphics rendering, and physics simulation, UE4 is increasingly being used in the autonomous industry for

simulating scenarios, including designing and evaluating autonomous agents. A sample snapshot of UE4 is shown in Figure 2.6.

Figure 2.6: Unreal Engine 4 Showcasing a Sample Carla Map

2.5 CARLA Simulator

CARLA (Car Learning to Act) is an open-source simulator designed for research in autonomous driving. It was developed at the Computer Vision Center (CVC) and Intel Intelligent Systems Lab. CARLA supports the designing, studying, evaluation, and validation of autonomous agents, which includes both perception and control. CARLA has been built as an open-source layer over Unreal Engine 4, which provides state-of-the-art rendering quality, realistic physics, NPC logic, and other plugins.

While commercial game simulation urban environments like Grand Theft Auto V (Richter *et al.*, 2016) (Richter *et al.*, 2017) offer high-quality graphics rendering, unlike CARLA, they lack high customization, control over the environment, and sensor support. CARLA is used for generating the SKOPE3D dataset.

The CARLA simulation platform includes a wide range of maps with urban layouts, a multitude of vehicle models, buildings, pedestrians, street signs, etc. which were specially designed for the simulator. It also supports a wide range of sensors

such as RGB camera, depth camera, RADAR, LIDAR, etc. It provides signals that can be used to train driving strategies, such as GPS, speed, acceleration, and collision details. These sensors can be used to collect data from the simulations and be used for machine learning solutions.

The environmental conditions can be changed as needed, including weather and the time of the day, providing a highly customizable simulation environment as shown in Figure 2.7.

Figure 2.7: CARLA Simulator Screen Captures Showcasing Dynamic Weather and Lighting Conditions (Row 1: Sunny, Rainy; Row 2: Evening, Noon)

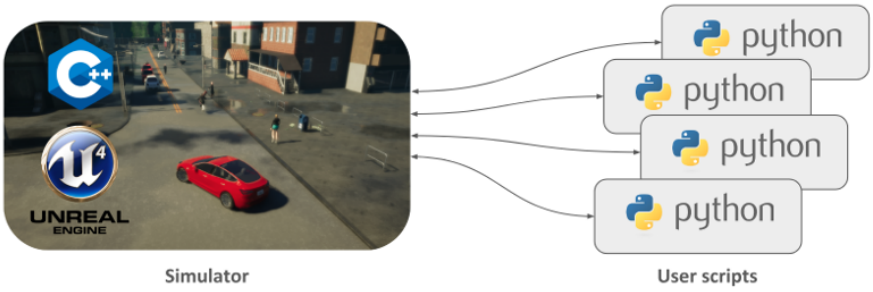

Figure 2.8: Interaction Between Carla Simulator and User Scripts That Control the Environment

There are various elements that build up a scene in CARLA -

- **Environment**: The environment is composed of 3D models of static objects such as vegetation, road infrastructure, buildings, and traffic signs. The environment also includes dynamic objects such as vehicles and pedestrians. The CARLA simulator comes with over 10 precompiled maps that follow the OpenDRIVE standard, which is a well-established standard for defining a road network's logic.

- **Traffic Manager**: CARLA has a built-in Traffic Manager system that takes control of the non-playing character (NPC) vehicles in the scene. These are additional vehicles in the scene other than the one that is learning. It basically acts as a conductor and tries to recreate realistic behaviors of NPC cars and pedestrians. In this project, all the vehicles are controlled by the traffic manager, since, there's no need for an ego vehicle.

- **Sensors**: There are numerous sensors like RGB cameras, LIDAR, RADAR, depth sensors, segmentation cameras, etc. that CARLA supports. In the case of SKOPE3D, the RGB camera and depth sensor are the ones being used.

- **Recorder**: This module can record the entire simulation and play it step by step for every actor in the world.

CARLA engine is controlled via Python scripts that can control the vehicles, environment, and other low-level features of the maps as shown in 2.8.

The CARLA Simulator doesn't support keypoint annotations as part of its feature suite.

Chapter 3

METHODOLOGY

This section presents an overview of the data generation pipeline that leverages an exclusive feature in CARLA. The CARLA simulator does not come with keypoint annotation out-of-the-box; however, it provides a feature called "wheel center of mass location" from its physics engine. The data generation pipeline utilizes this feature and extracts all 33 keypoints of a vehicle in 3D. These keypoints are then projected onto a 2D image and utilized for training a keypoint detector.

3.1 Keypoint definition

For vehicle keypoints, CAROM Air's (Lu *et al.* (2023)) keypoint definition was utilized, which defines keypoints in groups of two (left or right) or four (front-right, front-left, rear-right, rear-left) symmetrical key features. A total of 33 keypoints were defined for SKOPE3D as illustrated in Table 3.1 and illustrated in the Figure 3.1. These keypoints are notable key features of a vehicle such as corners of the windshield, corners of the bumper, headlights, taillights, wheel centers, and wheel bottoms, among others. These keypoints enable the ability to provide accurate and reliable information about the positioning and orientation of a vehicle.

At first glance, vehicle keypoint estimation might seem easier compared to humans due to the rigid structure of vehicles that limits their pose and less complex occlusion, but there are still various difficulties. Camera perspective plays a larger role in vehicles than in humans, and intra-class variability is huge due to a large number of cars being released every year with variable models, sizes, and types (Sánchez *et al.*, 2020). A fitting example of this would be the Tesla Cybertruck, which is unlike any other car

Table 3.1: Definition of Vehicle Keypoints (Courtesy: Lu *et al.* (2023))

ID	keypoint definition
0-3	corners of rooftop
4-7	corners of the front and rear windshield
8-11	centers of the front and read lights
12-15	corners of front and rear bumpers
16-19	centers of wheels
20-23	corners of chassis bottom surface
24-25	outermost corners of side mirrors
26-27	corners of the front door windows
28-31	wheel-ground contact point
32	center of the brand logo in the front

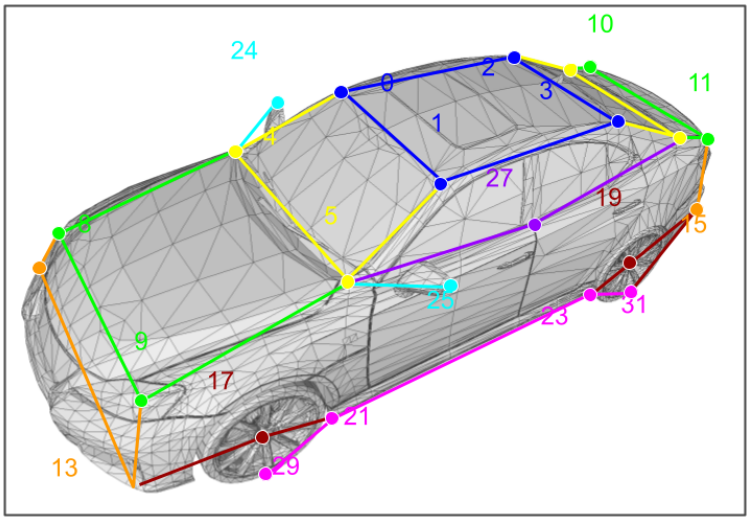

Figure 3.1: Vehicle Keypoints definition (Image Courtesy: Lu *et al.* (2023))

ever designed.

3.2 Data generation pipeline

CARLA simulator does not support keypoint annotations as part of its feature suite. However, after exploring the CARLA simulator and its Python API, it was

found that the simulator has a feature of getting the wheel's center of mass, as shown in Figure 3.2. This feature was utilized and extended to act as a reference point in 3D to extract keypoint annotations for all the vehicles in the scene.

A module was developed that allows the extraction of 33 keypoints for any vehicle in the scene, which only requires annotating the 3D models of vehicles once.

Figure 3.2: Annotated 3d Model of Carla's Mini Cooper Vehicle, Red Arrows Pointing to the Wheel Centers.

Figure 3.3 shows the entire data generation pipeline.

There are three major steps in the data generation pipeline:

1. 3D labeling - The platform requires annotated 3D models of vehicles available in CARLA. These annotated keypoints of each model are parsed and new keypoint separation data is generated that will be the input to the Trilateration Algorithm.

2. Scene configuration - The scene is initialized with all necessary sensors and annotated vehicles.

3. Image and annotation generation - Then during simulation, the wheel centers are extracted which are then passed as reference points to the Trilateration

algorithm that calculates the 3d location of all 33 keypoints in 3D. After checking for occluded vehicles, data for a single frame is generated. This process is repeated for each frame of an entire scene.

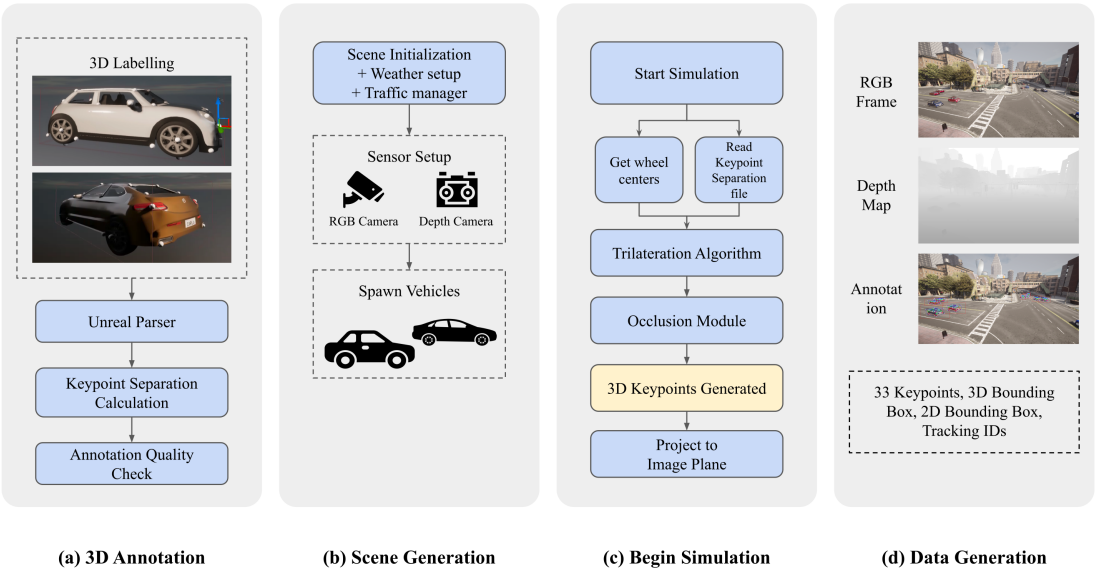

Figure 3.3: Skope3d's Data Generation Pipeline

3.3 3D keypoint Annotation

For annotating the 3D keypoints, the game engine Unreal Engine 4 (UE4) was used. While any 3D modeling software could have been used for this task, UE4 was chosen because there would be no discrepancies in the scaling factor of the 3D models. The models will be exactly the same scale as the models in the simulator environment.

A vehicle skeleton with 33 3D spheres, each numbered according to the keypoint definition illustrated in table 3.1. Annotating a vehicle requires the following steps -

- Take the spheres marked for wheel center and place them at the center of each wheel

- Place all the other keypoints according to the vehicle's shape

An example of the 3D annotations can be seen in figure 3.4

Figure 3.4: Mini Cooper Annotations. White Spheres on the Vehicle Are 3d Annotation Points.

3.3.1 Unreal Parser

For this use case, a module called the Unreal Parser was developed. This parser takes into the all 3D annotations aka the sphere's metadata from Unreal Engine's object panel and parses the keypoint's 3D location in the world coordinate frame. It then encodes this information into a JSON file for use in downstream tasks.

3.3.2 Keypoint Separation Algorithm

After the 3D annotation is successful, the annotation data is extracted from Unreal Engine and stored in JSON format. A simple Unreal Parser script is designed that can easily parse the 3D objects and extract each keypoint's 3D position from its transform.

Once the keypoints have been parsed, they can be passed through the keypoint separation algorithm, which calculates the Euclidean distance of all the keypoints

from four reference points - the center of mass locations of the front right wheel, front left wheel, rear right wheel, rear left wheel.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Each keypoint will have four distances - d_1, d_2, d_3, d_4, each with respect to the reference points. The 3D world coordinate of each reference point is also stored in the keypoint separation file.

3.3.3 Quality of annotation

To ensure the quality of the annotations, it was necessary to ensure that the annotated wheel center of mass locations coincided with CARLA's wheel center obtained from the physics engine. This was achieved using simple distance formula between the annotated wheel center and actual wheel center in a scene.

A demo scene was designed for this purpose, calculating the distance between wheel centers in 3D using the following formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

where (x1, y1, z1) and (x2, y2, z2) are the coordinates of any two wheel centers.

After calculating the distances between the wheels, the scale_factor is calculated, which determines the quality of the annotations, using the following formula:

$$annotated\_wheel\_distance = d\_1$$

$$actual\_wheel\_distance = d\_2$$

$$scale\_factor = d\_2/d\_1$$

where annotated_wheel_distance is the Euclidean distance between manually annotated points and actual_wheel_distance is the distance between the 3D points obtained from CARLA's API during simulation.

If scale_factor equals 1, our annotations were perfect, if it's less than 1, the distance between the annotated points needs to be reduced, and if it's more than 1, the distance between the annotated points needs to be increased.

3.4 Scene configuration

After annotating the keypoints on the 3D models of vehicles, the next step in the data generation pipeline is to configure the scene. A series of manually selected scenes with unique viewpoints of the roads were chosen for data collection.

A scene is selected from these predefined configurations for recording, and environmental conditions are set by randomly choosing a weather configuration and time of the day or by using one of the presets that have been defined.

The scene configuration presets comes with predefined locations for camera sensors that are used to place the camera and depth sensors in the scene with default settings (Size: 1920x1080; FOV: 110).

The next element in the scene configuration is the traffic manager which allows for the control of traffic and vehicle spawning with auto-pilot configuration. Vehicles are randomly spawned in close proximity to the camera with varying color configurations to diversify the dataset.

A sample viewpoint is shown in figure 3.5

3.5 Annotation Generation

Once the scene has been set up, a Python client communicates with the simulator's server to start the simulation step-by-step. At each step, the data collection client

Figure 3.5: Sample Snapshot of a Scene after Setting up the Scene, Camera Sensor, and Spawning Vehicles

queries the Physics Engine to get the wheel's center of mass for all the vehicles. This provides the (x, y, z) coordinates for the front left wheel, front right wheel, rear left wheel, and rear right wheel for the vehicles in the scene in the world coordinate frame. For each vehicle that has been annotated in 3D, the keypoint separation file containing the distance of all the keypoint's with respect to the center of mass of all the wheels of the vehicle is read. Using these reference points and the distance of all the keypoints, the Trilateration algorithm is used to calculate the position of all the keypoints.

3.5.1 Trilatertion

Trilateration is a method used to determine the location of an object by measuring its distance from three or more reference points. This algorithm is mainly used in the Global Positioning System (GPS), it helps in determining someone's location on the planet.

It centers around finding someone's position on Earth by knowing the location of the orbiting GPS satellites and their distance from these satellites. At least three

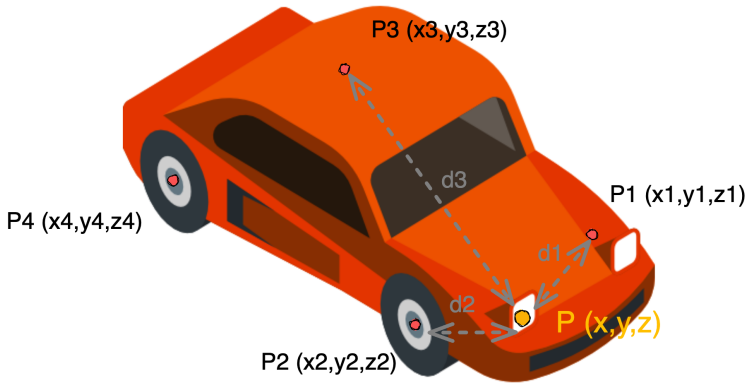

Figure 3.6: Trilateration Example That Uses Three Wheel Centers to Estimate the Location of Left Headlight (Key Point 9, According to Our Definition)

known points are required for trilateration to work. In the case of SKOPE3D dataset, four reference points are known for the vehicles.

Figure 3.6 shows how a single keypoint can be estimated using this algorithm. Point P is the right headlight keypoint from the keypoint definition. The known reference points are P_1, P_2, P_3, and P_4 which are the wheel's center of mass locations of the vehicle. The distance of point P from these reference points is also known - d_1, d_2, d_3, and d_4.

The basic principle of trilateration is that the location of an object can be determined by finding the intersection of three or more circles, where each circle represents the distance from a reference point.

Mathematically, this can be expressed as a system of equations:

$$(x - x1)^2 + (y - y1)^2 + (z - z1)^2 = d1^2$$

$$(x - x2)^2 + (y - y2)^2 + (z - z2)^2 = d2^2$$

$$(x - x3)^2 + (y - y3)^2 + (z - z3)^2 = d3^2$$

where (x, y, z) is the location of the object, $(x1, y1, z1)$, $(x2, y2, z2)$, and $(x3, y3, z3)$ are the locations of the reference points, and d1, d2, and d3 are the distances from the object to each reference point. After using the trilateration algorithm, the location of the keypoint is extracted in the format (x, y, z) in the world reference frame. All 33 keypoints for every vehicle in the scene are localized in the world-coordinate frame using the Trilateration algorithm.

3.5.2 Occlusion Module

CARLA does not have a built-in occlusion detection engine, so vehicles that are behind the buildings and are not visible by the camera sensors are still provided by the simulator when querying all the vehicles, leading to wrong annotations. To address this issue, an occlusion module was developed that is designed to find large object occlusions like buildings and big trucks.

To remove occluded vehicles from the annotations, sample scenario is shown in Figure 3.7:

- The Euclidean distance is calculated between the camera's center $(x1, y1, z1)$ and the vehicle's center $(x2, y2, z2)$ using CARLA's Python API. Suppose this distance is $d1$.
- If an object in the scene, such as a building or large truck, is occluding the target vehicle, its depth can be obtained using the depth camera. This depth camera captures the scene depth at the time of capture, and thus provides the depth of the object between the camera and the target vehicle. Suppose this depth is $d2$.
- The difference between the depth $d2$ and the Euclidean distance $d1$ is then

calculated, and compared against the *occlusion_threshold*. If this difference exceeds the threshold, the target vehicle is considered occluded and removed from the annotations.

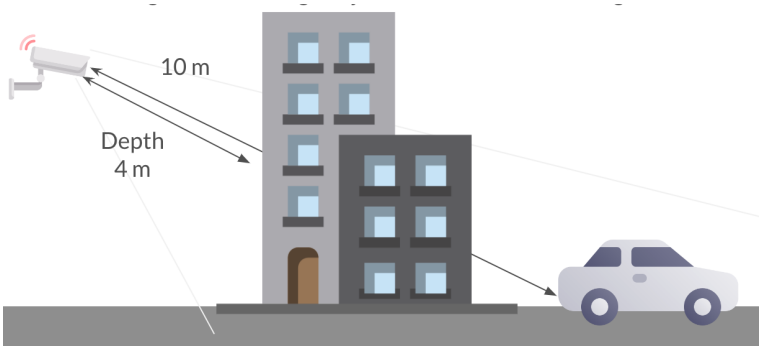

Figure 3.7: Occlusion Scenario, Depth Is 4 M, Euclidean Distance Is 10 M, the Grey Car Is Occluded in Annotaitons.

3.6 Dataset format

The dataset structure is shown in figure 3.8. The dataset's official split is heterologous, where the training and testing sets are divided based on the scene. Out of 25 scenes, 23 scenes will be used for training, while the remaining 2 unseen scenes from Town 4 are used for validation. This enables us to test the generalization ability of the keypoint detection approaches.

Each frame in SKOPE3D has two separate images as shown in figure 4.3 -

- RGB Frame 1920x1080
- Depth map 1920x1080

SKOPE3D has two types of annotations -

- Keypoint data
- 3D bounding box and tracking data

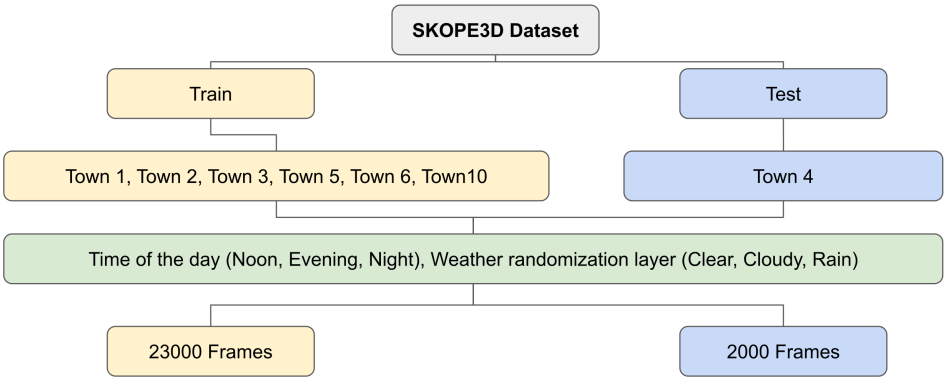

Figure 3.8: Skope3d Dataset Structure

3.6.1 Keypoint data

Each detected vehicle in a frame will have its own keypoint CSV file, whose structure is shown in table 3.2. Additionally, each vehicle will have its metadata CSV file that contains the tracking ID of the vehicle, the model of the car, and 3D bounding box corners (in 2D). The 3D coordinates of the 3D bounding box can be calculated using camera intrinsics and extrinsics. The structure of the file is in table 3.3. The dataset's earlier version includes two vehicle classes: Sedan and SUV. However, more classes like pickup trucks, vans, and trucks will be added in later versions of the dataset.

Table 3.2: Keypoint Data File Format

Row	Column	Meaning
0	0	Keypoint coordinate offset x (origin of the patch in the image)
0	1	Keypoint coordinate offset y (origin of the patch in the image)
0	2	Vehicle type ID
1	0	bounding box x0
1	1	bounding box y0
1	2	rotation flag (not used)
2	0	bounding box x1
2	1	bounding box y1
2	2	mirror flag (not used)
3-36	0	Keypoint coordinate x
3-36	1	Keypoint coordinate y
3-36	2	Keypoint visibility flag

Table 3.3: 3d Bounding Box and Tracking Data

Row	Column	Description
0	0	Tracking ID for that scene
0	1	Name of the vehicle's 3D model
1-9	0	Bounding box corner x coordinate in world coordinate frame
1-9	1	Bounding box corner y coordinate in world coordinate frame

Figure 3.9: Sample Rgb Frame (Top) and Depth Map (Bottom) from a Scene

Chapter 4

EXPERIMENTS

4.1 Problem Statement

To localize the 3D keypoints of vehicles given ambiguous images captured under various settings and scenes, including different viewpoints and road configurations.

4.2 Evaluation Metrics

There are multiple evaluation metrics for keypoint detections. The chosen metrics are proposed by Sanchez et al. (Yang and Ramanan, 2012) and include:

- **Percentage of Correct Keypoints (PCK)**: The metric measures the number of labeled keypoints that are correctly predicted. It's a keypoint similarity metric that says that a predicted keypoint is correct if its distance to the given ground-truth keypoint is equal to or less than $\alpha * L$ where $L = max(height, width)$ (basically L is the bigger dimension of the instance's bounding box) and $0 < \alpha < 1$. $\alpha = 0.1$ is used (Sánchez *et al.*, 2020).

- **Precision Recall of the bounding boxes**: Precision measures the proportion of predicted bounding boxes that are correct. Recall measures the proportion of actual bounding boxes that are correctly detected. We analyze the value on different values of IoU thresholds.

- **Precision Recall of the keypoints**: Similarly, precision measures the proportion of predicted keypoints that are correct, while recall measures the proportion of actual keypoints that are correctly detected. This work extends the

keypoint similarity metric with $\alpha = 0.1$ to calculate precision-recall. Recall in the current case is not a feasible metric for keypoint evaluation because the visibility parameter is set to 1 and there will be rarely any false negatives in the prediction.

4.3 Experimental Setup

The SKOPE3D dataset contains 25K images with the training and validation ratio set to 9:1. To analyze the capability of the dataset, a subset of the dataset with 13.5K frames was extracted, with the training and validation ratio set to 8:2. This split is heterologous in nature. The training set has 10.4K images from various different scenes and the validation set has the remaining 2.9K images from three unseen scenes of the dataset. This helps in validating the generalization ability of the keypoint detector as well as the synthetic dataset.

The test set consists of three distinct scenes with varying difficulty levels, namely easy, medium, and hard. The difficulty level of each test set was determined based on factors such as adverse weather conditions, low vehicle visibility, and partial occlusion.

The Easy scene has good visibility, sunny weather, and fewer partially occluded vehicles, and the medium scene has average visibility, rainy weather, and a few partially occluded vehicles as depicted in Figure 4.1. The test set also includes a challenging scene (hard) with adverse weather conditions and low visibility. A sample frame from this scene is depicted in Figure 4.2, demonstrating the difficulty of the task at hand.

4.3.1 Baseline Implementation Detail

The architecture used is the one proposed by (He *et al.*, 2017): Keypoint R-CNN. The implementation is based on PyTorch. This architecture consists of an ImageNet pre-trained ResNet50-FPN as its backbone network. The feature Pyramid Network

Figure 4.1: Samples Frames from the Test Set: Top - Easy Scene, Bottom - Medium Scene

fuses feature maps at multiple scales to preserve information at multiple levels.

The input to the model is a list of tensors, each of shape $[4, 1080, 1920]$, one for each image and in the 0-1 range. It expects the annotations to be in targets (list of dictionaries), containing:

- Boxes ($FloatTensor[N, 4]$) the ground-truth 2D boxes in $[x1, y1, x2, y2]$ format, with $0 \leq x1 < x2 \leq w$ and $0 \leq y1 < y2 \leq h$
- Labels ($Int64Tensor[N]$): the predicted labels for each instance
- Keypoints ($FloatTensor[N, K, 3]$): the K keypoints location for each of the N instances, in the format $[x, y, visibility]$, where visibility=0 means that the

Figure 4.2: Sample Frame from the Hard Scene (Low Visibility, Muddy Road, Rainy Weather, Vehicles Partially Occluded Due to Viewpoint)

keypoint is not visible.

In inference mode, the model requires only the input tensor and returns the predictions as a List[Dict[Tensor]], for each image. The content of Dict is as follows, where N is the number of detected instances:

- Boxes ($FloatTensor[N, 4]$): the predicted boxes in $[x1, y1, x2, y2]$ format, with $0 <= x1 < x2 <= W$ and $0 <= y1 < y2 <= H$.
- Labels ($Int64Tensor[N]$): the predicted labels for each instance
- Scores ($Tensor[N]$): the scores or each instance

  item keypoints ($FloatTensor[N, K, 3]$): the locations of the predicted keypoints, in $[x, y, v]$ format. $K = 33$ in this case

As for the learning rate parameters, the following parameters were tested and achieved good results on the test set:

- Learning rate: 0.001

- Step size: 3
- Gamma: 0.2

The model was trained for 10 epochs, which took around 11.5 hours on an NVIDIA TITAN X GPU.

4.4 Main Results and Analysis

In order to evaluate how well synthetic data generalizes to new, unseen scenes, several experiments were conducted. The evaluation was performed on three distinct scenes with varying difficulty levels, namely easy, medium, and hard. The difficulty level of each test set was determined based on factors such as adverse weather conditions, low vehicle visibility, and occlusion. After updating the learning rate parameters, the loss converged to a reasonable extent. While the classification loss converged to 0.0138 in 10 epochs, the keypoint loss did not go below 0.42. All experiments used the keypoint predicted score to be 0.9 and above.

Continuing with the experiments, the model was first evaluated with scene 4's 1000 frames test set (easy difficulty). A sample prediction can be seen in Figure 4.4. The use of a deeper backbone model ResNet50 has helped in keypoint detection. All backbone layers were set to be trainable, and pre-trained weights from ImageNet were used.

As shown in Table 4.1, the model performed well on the easy scene. On PCK, a value of 98.65% was obtained (prior to hyperparameter tuning, it was 96.4%) on a 0.9 IoU threshold. The bounding box precision was 0.408 and the recall was 0.345. The keypoint precision and recall do not vary much with the IoU threshold. The bounding box precision improved drastically when the IoU threshold was reduced, as expected.

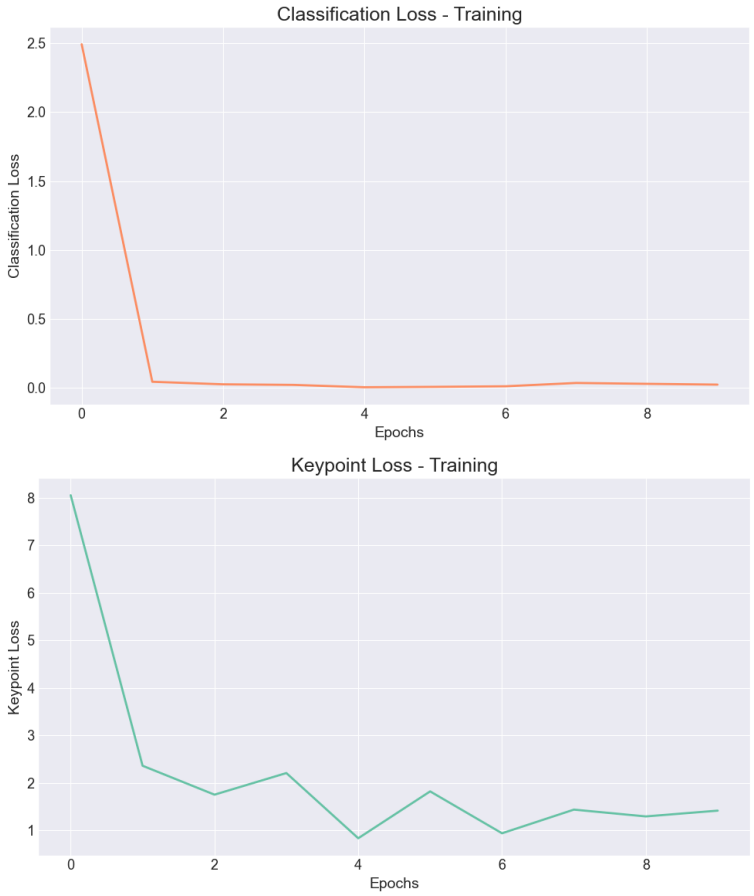

Figure 4.3: Classification Loss Converged Well and Keypoint Loss Struggled to Go Below 0.42

As shown in table 4.2, The model performed well on the medium scene, which featured rain but comparatively better visibility conditions than the hard scene. On a 0.9 IoU threshold, PCK was 94.56%, with fairly good precision and recall for both bounding boxes and keypoints.

The hard scene was the most challenging from the test set, and the model's performance suffered. As shown in table 4.3, the percentage of correct keypoints did not surpass the 64.74% mark, and precision and recall also suffered due to visibility conditions. Figure 4.5 shows a sample prediction for this scene. The model Failed to Predict the Keypoints Accurately on a Hard Scene, as Demonstrated by the Two Ve-

hicles on the Left. The Poor Visibility and Occlusion Caused the Model to Consider the Two Vehicles as a Single Object, Resulting in Inaccurate Predictions.

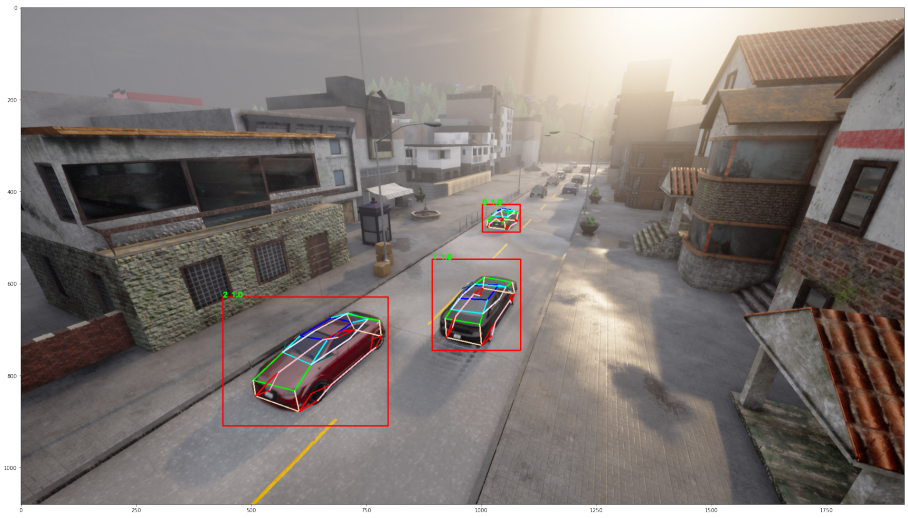

Figure 4.4: Sample Good Prediction on Easy Scene, Although It Missed to Detect a Few Vehicles at a Distance from the Camera

Figure 4.5: Model Failed on the Hard Scene

4.5 Model Generalizability

In addition to testing on the SKOPE3D dataset, the model was also evaluated on real-life images. While quantitative results could not be obtained due to a lack of a similar dataset for testing that has 33 keypoints, qualitative results were shown in

Table 4.1: Easy scene evaluation: PCK with $\alpha = 0.1$, BB Precision-recall and KP Precision-recall with Varying IoU Threshold. All Runs with Resnet-50 Backbone and Input Size 1920x1080

IoU Threshold	**PCK %**	**Bounding box**		**Keypoint**	
		Precision	Recall	Precision	Recall
0.9	98.65	0.408	0.345	0.462	0.931
0.8	95.59	0.793	0.641	0.488	1.0
0.7	95.01	0.951	0.775	0.486	1.0
0.6	95.51	0.952	0.776	0.488	1.0

Table 4.2: Medium scene evaluation: PCK with $\alpha = 0.1$, BB Precision-recall and KP Precision-recall with Varying IoU Threshold.

IoU Threshold	**PCK %**	**Bounding box**		**Keypoint**	
		Precision	Recall	Precision	Recall
0.9	94.56	0.635	0.537	0.433	0.894
0.8	93.77	0.941	0.776	0.475	0.983
0.7	95.75	0.951	0.785	0.477	0.987
0.6	93.67	0.962	0.795	0.479	0.992

Table 4.3: Hard scene evaluation: PCK with $\alpha = 0.1$, BB Precision-recall and KP Precision-recall with Varying IoU Threshold.

IoU Threshold	**PCK %**	**Bounding box**		**Keypoint**	
		Precision	Recall	Precision	Recall
0.9	45.183	0.123	0.19	0.146	0.553
0.8	63.046	0.382	0.615	0.366	0.99
0.7	64.743	0.513	0.809	0.385	1.0
0.6	64.575	0.568	0.882	0.386	1.0

Figure 4.6. The top-left frame displays an accurate keypoint regression on a sedan, along with a false positive. The top-right frame demonstrates the ability to detect multiple types of vehicles. The bottom-left frame showcases keypoint detection on an SUV, while the bottom-right frame demonstrates keypoint detection on a pickup truck, despite the latter not being part of the training dataset.

This shows the extraordinary capabilities of keypoint R-CNN and the power of synthetic data. A model trained on the synthetic dataset is able to predict keypoints on the real dataset, showing the knowledge transferability between synthetic and real-life images.

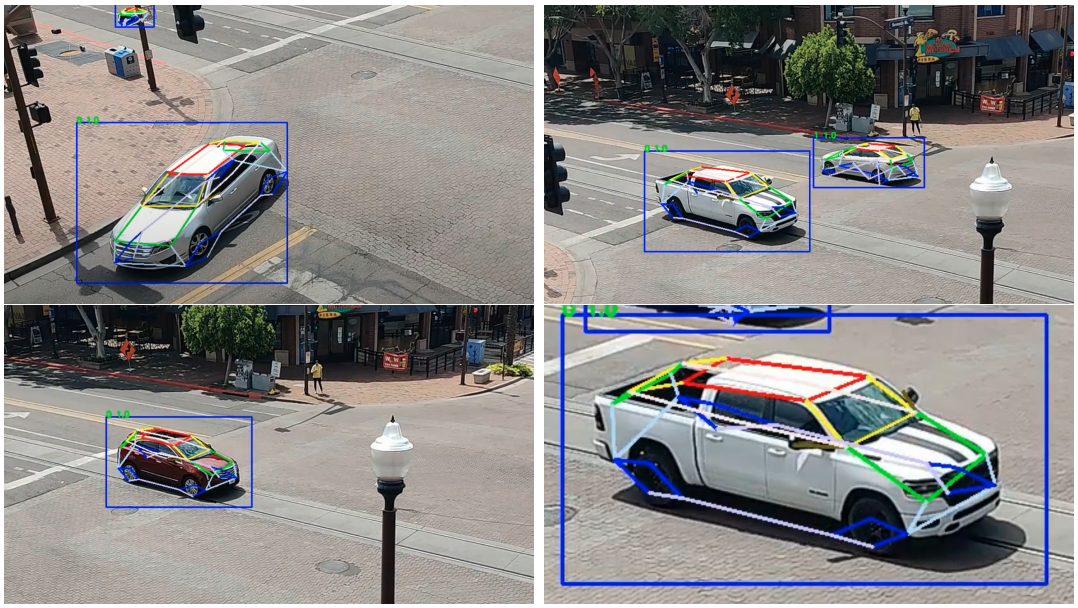

Figure 4.6: Sample Frames Were Extracted from a Smartphone Video of the Mill Avenue Intersection, Tempe, Az.

Chapter 5

CONCLUSION AND FUTURE WORK

This chapter summarizes the thesis, the limitations of the study and shows the potential future research directions.

Figure 5.1: Skope3d Dataset: Different Scenes in the Dataset

5.1 Summary

The thesis proposes a unique high-diversity synthetic keypoint perception 3D dataset (Figure 5.1), SKOPE3D, which includes 33 keypoint annotations for each vehicle and is collected from a roadside view using the CARLA simulator. This feature makes SKOPE3D distinct from previously released traffic monitoring datasets. Additionally, as the dataset is simulated, it can be extended using publicly released data extension scripts. Furthermore, SKOPE3D cannot be utilized for illegal surveil-

lance as it does not include any sensitive information such as license plates, human faces, roads, and buildings.

Traffic monitoring datasets with keypoint annotations and 3D/2D annotations are valuable for traffic analysis, autonomous driving, and 3D scene reconstruction, and have the potential for use in a wide variety of studies.

Moreover, the thesis presents an evaluation of a classic keypoint detector on the dataset, Keypoint R-CNN. Performance was measured using widely accepted metrics, such as PCK and precision-recall, and the experiments demonstrated the real value of such a synthetic dataset as it achieved good prediction accuracies.

The study represents a step towards bridging the gap between synthetic and real-world data, as training a keypoint detector on a synthetic dataset allows the model to generalize on real-world images to some extent. This demonstrates the knowledge-transferability between synthetic and real-world data and highlights the value of SKOPE3D.

5.2 Delimitations

As the dataset is synthetically generated using a simulator, certain delimitations were imposed in order to fulfill the objectives. Every constraint was established to enable the work presented in this thesis to serve as a foundation for future extensions, with fewer imposed limitations. The following are the delimitations in the work:

1. The keypoint visibility parameter is set to 1 as the occlusion module was not producing accurate results for keypoint depth. The occlusion module identifies occlusions caused by buildings and other large structures but not partial occlusions caused by other vehicles. Future versions of the dataset could address this limitation.

2. All vehicles are assumed to move on a perfectly flat horizontal plane, and no vertical movement is considered.

3. The position of the camera and depth sensor is fixed in a particular scene with no motion or rotation during recording.

4. The Unreal Engine is used to simulate weather, but it may not perfectly replicate actual weather scenarios such as rain.

5.3 Future work

The study has a lot of scope for future work that can improve the value and capabilities of the dataset. Several future works may involve:

1. Occlusion module can further be extended to keypoints in future versions.

2. Adding real-world annotated images to the dataset and fine-tuning the keypoint detector using a combination of synthetic and real datasets may result in improved real-world performance.

3. The data extension scripts will be released to the research community, enabling others to extend the dataset by adding new scenes, vehicles, and more annotations such as LIDAR and instance segmentation.

BIBLIOGRAPHY

Altekar, N., S. Como, D. Lu, J. Wishart, D. Bruyere, F. Saleem and K. L. Head, "Infrastructure-based sensor data capture systems for measurement of operational safety assessment (osa) metrics", SAE International Journal of Advances and Current Practices in Mobility **3**, 2021-01-0175, 1933–1944 (2021).

Barra, P., S. Barra, C. Bisogni, M. De Marsico and M. Nappi, "Web-shaped model for head pose estimation: An approach for best exemplar selection", IEEE Transactions on Image Processing **29**, 5457–5468 (2020).

Barra, P., C. Bisogni, M. Nappi, D. Freire-Obregón and M. Castrillon-Santana, "Gender classification on 2d human skeleton", in "2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART)", pp. 1–4 (IEEE, 2019).

Caesar, H., V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 11621–11631 (2020).

Cao, Z., T. Simon, S.-E. Wei and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 7291–7299 (2017).

Chang, M.-F., J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 8748–8757 (2019).

Choi, W., "Near-online multi-target tracking with aggregated local flow descriptor", in "Proceedings of the IEEE international conference on computer vision", pp. 3029–3037 (2015).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in "2009 IEEE conference on computer vision and pattern recognition", pp. 248–255 (Ieee, 2009).

Desmarais, Y., D. Mottet, P. Slangen and P. Montesinos, "A review of 3d human pose estimation algorithms for markerless motion capture", Computer Vision and Image Understanding **212**, 103275 (2021).

Dosovitskiy, A., G. Ros, F. Codevilla, A. Lopez and V. Koltun, "Carla: An open urban driving simulator", in "Conference on robot learning", pp. 1–16 (PMLR, 2017).

Fedorov, A., K. Nikolskaia, S. Ivanov, V. Shepelev and A. Minbaleev, "Traffic flow estimation with data from a video surveillance camera", Journal of Big Data **6**, 1–15 (2019).

Felzenszwalb, P. F., R. B. Girshick, D. McAllester and D. Ramanan, "Object detection with discriminatively trained part-based models", IEEE transactions on pattern analysis and machine intelligence **32**, 9, 1627–1645 (2009).

Feng, J., D. Zeng, X. Jia, X. Zhang, J. Li, Y. Liang and L. Jiao, "Cross-frame keypoint-based and spatial motion information-guided networks for moving vehicle detection and tracking in satellite videos", ISPRS Journal of Photogrammetry and Remote Sensing **177**, 116–130 (2021).

Gählert, N., N. Jourdan, M. Cordts, U. Franke and J. Denzler, "Cityscapes 3d: Dataset and benchmark for 9 dof vehicle detection", arXiv preprint arXiv:2006.07864 (2020).

Geiger, A., P. Lenz and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite", in "2012 IEEE conference on computer vision and pattern recognition", pp. 3354–3361 (IEEE, 2012).

Geyer, J., Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn *et al.*, "A2d2: Audi autonomous driving dataset", arXiv preprint arXiv:2004.06320 (2020).

Girshick, R., "Fast r-cnn", in "Proceedings of the IEEE international conference on computer vision", pp. 1440–1448 (2015).

Girshick, R., J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 580–587 (2014).

Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning* (MIT Press, 2016), `http://www.deeplearningbook.org`.

Gupta, D., B. Artacho and A. Savakis, "Vehipose: a multi-scale framework for vehicle pose estimation", in "Applications of Digital Image Processing XLIV", vol. 11842, pp. 517–523 (SPIE, 2021).

He, K., G. Gkioxari, P. Dollár and R. Girshick, "Mask r-cnn", in "Proceedings of the IEEE international conference on computer vision", pp. 2961–2969 (2017).

Huang, X., P. Wang, X. Cheng, D. Zhou, Q. Geng and R. Yang, "The apolloscape open dataset for autonomous driving and its application", IEEE transactions on pattern analysis and machine intelligence **42**, 10, 2702–2719 (2019).

Insafutdinov, E., L. Pishchulin, B. Andres, M. Andriluka and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model", in "Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14", pp. 34–50 (Springer, 2016).

Kar, A., S. Tulsiani, J. Carreira and J. Malik, "Category-specific object reconstruction from a single image", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 1966–1974 (2015).

Kesten, R., M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang and V. Shet, "Lyft level 5 av dataset 2019", `https://level5.lyft.com/dataset/` (2019).

Khan, S. D. and H. Ullah, "A survey of advances in vision-based vehicle re-identification", Computer Vision and Image Understanding **182**, 50–63 (2019).

Kreiss, S., L. Bertoni and A. Alahi, "Pifpaf: Composite fields for human pose estimation", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 11977–11986 (2019).

LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE **86**, 11, 2278–2324 (1998).

Li, C., M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager and M. Chandraker, "Deep supervision with intermediate concepts", IEEE transactions on pattern analysis and machine intelligence **41**, 8, 1828–1843 (2018).

Lin, T.-Y., P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature pyramid networks for object detection", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 2117–2125 (2017).

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft coco: Common objects in context", in "Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13", pp. 740–755 (Springer, 2014).

Llorca, D. F., C. Salinas, M. Jimenez, I. Parra, A. Morcillo, R. Izquierdo, J. Lorenzo and M. Sotelo, "Two-camera based accurate vehicle speed measurement using average speed at a fixed point", in "2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)", pp. 2533–2538 (IEEE, 2016).

Lu, D., E. Eaton, M. Weg, W. Wang, S. Como, J. Wishart, H. Yu and Y. Yang, "Carom air-vehicle localization and traffic scene reconstruction from aerial video", (2023).

Lu, D., V. C. Jammula, S. Como, J. Wishart, Y. Chen and Y. Yang, "Carom-vehicle localization and traffic scene reconstruction from monocular cameras on road infrastructures", in "2021 IEEE International Conference on Robotics and Automation (ICRA)", pp. 11725–11731 (IEEE, 2021).

Murthy, J. K., G. S. Krishna, F. Chhaya and K. M. Krishna, "Reconstructing vehicles from a single image: Shape priors for road scene understanding", in "2017 IEEE International Conference on Robotics and Automation (ICRA)", pp. 724–731 (IEEE, 2017).

Newell, A., Z. Huang and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping", Advances in neural information processing systems **30** (2017).

Newell, A., K. Yang and J. Deng, "Stacked hourglass networks for human pose estimation", in "Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14", pp. 483–499 (Springer, 2016).

Patil, A., S. Malla, H. Gang and Y.-T. Chen, "The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes", in "2019 International Conference on Robotics and Automation (ICRA)", pp. 9552–9557 (IEEE, 2019).

Patil, C., "Human pose estimation using keypoint RCNN in PyTorch", `https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/`, accessed: 2023-4-21 (2021).

Pishchulin, L., E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 4929–4937 (2016).

Reddy, N. D., M. Vo and S. G. Narasimhan, "Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 1906–1915 (2018).

Reddy, N. D., M. Vo and S. G. Narasimhan, "Occlusion-net: 2d/3d occluded keypoint localization using graph networks", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 7326–7335 (2019).

Redmon, J. and A. Farhadi, "Yolov3: An incremental improvement", arXiv preprint arXiv:1804.02767 (2018).

Ren, J., X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai and L. Xu, "Accurate single stage detector using recurrent rolling convolution", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 5420–5428 (2017).

Ren, S., K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", Advances in neural information processing systems **28** (2015).

Richter, S. R., Z. Hayder and V. Koltun, "Playing for benchmarks", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 2213–2222 (2017).

Richter, S. R., V. Vineet, S. Roth and V. Koltun, "Playing for data: Ground truth from computer games", in "Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14", pp. 102–118 (Springer, 2016).

Sánchez, H. C., A. H. Martínez, R. I. Gonzalo, N. H. Parra, I. P. Alonso and D. Fernandez-Llorca, "Simple baseline for vehicle pose estimation: Experimental validation", IEEE Access **8**, 132539–132550 (2020).

Simon, T., H. Joo, I. Matthews and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping", in "Proceedings of the IEEE conference on Computer Vision and Pattern Recognition", pp. 1145–1153 (2017).

Sochor, J., J. Špaňhel and A. Herout, "Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance", IEEE transactions on intelligent transportation systems **20**, 1, 97–108 (2018).

Soliman, M. M., M. H. Kamal, M. A. E.-M. Nashed, Y. M. Mostafa, B. S. Chawky and D. Khattab, "Violence recognition from videos using deep learning techniques", in "2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)", pp. 80–85 (IEEE, 2019).

Song, X., P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li and R. Yang, "Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 5452–5462 (2019).

Sun, P., H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 2446–2454 (2020).

Tang, Z., M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 8797–8806 (2019).

Uijlings, J. R., K. E. Van De Sande, T. Gevers and A. W. Smeulders, "Selective search for object recognition", International journal of computer vision **104**, 154–171 (2013).

Wang, S. and C. C. Fowlkes, "Learning optimal parameters for multi-target tracking with contextual interactions", International journal of computer vision **122**, 484–501 (2017).

Wang, X., Y. Peng, L. Lu, Z. Lu, M. Bagheri and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases", CoRR **abs/1705.02315**, URL `http://arxiv.org/abs/1705.02315` (2017a).

Wang, Z., L. Tang, X. Liu, Z. Yao, S. Yi, J. Shao, J. Yan, S. Wang, H. Li and X. Wang, "Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification", in "Proceedings of the IEEE international conference on computer vision", pp. 379–387 (2017b).

Won, M., T. Park and S. H. Son, "Toward mitigating phantom jam using vehicle-to-vehicle communication", IEEE transactions on intelligent transportation systems **18**, 5, 1313–1324 (2016).

Xiang, Y., A. Alahi and S. Savarese, "Learning to track: Online multi-object tracking by decision making", in "Proceedings of the IEEE international conference on computer vision", pp. 4705–4713 (2015).

Xiang, Y., R. Mottaghi and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild", in "IEEE winter conference on applications of computer vision", pp. 75–82 (IEEE, 2014).

Xiao, B., H. Wu and Y. Wei, "Simple baselines for human pose estimation and tracking", (2018).

Yang, Y. and D. Ramanan, "Articulated human detection with flexible mixtures of parts", IEEE transactions on pattern analysis and machine intelligence **35**, 12, 2878–2890 (2012).

Ye, X., M. Shu, H. Li, Y. Shi, Y. Li, G. Wang, X. Tan and E. Ding, "Rope3d: the roadside perception dataset for autonomous driving and monocular 3d object detection task", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 21341–21350 (2022).

Zhang, L., Y. Li and R. Nevatia, "Global data association for multi-object tracking using network flows", in "2008 IEEE conference on computer vision and pattern recognition", pp. 1–8 (IEEE, 2008).

Zhang, S., C. Wang, Z. He, Q. Li, X. Lin, X. Li, J. Zhang, C. Yang and J. Li, "Vehicle global 6-dof pose estimation under traffic surveillance camera", ISPRS Journal of Photogrammetry and Remote Sensing **159**, 114–128 (2020).

Zia, M. Z., M. Stark, B. Schiele and K. Schindler, "Detailed 3d representations for object recognition and modeling", IEEE transactions on pattern analysis and machine intelligence **35**, 11, 2608–2623 (2013).