2019

A Voting Algorithm for Dynamic Object Identification and Pose Estimation

Chandini Ravindranathan Nair
University of Windsor

A Voting Algorithm for Dynamic Object Identification and Pose Estimation

by

CHANDINI RAVINDRANATHAN NAIR

A THESIS

Submitted to the Faculty of Graduate Studies
Through Computer Science
In Partial Fulfilment of the Requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2019

A Voting Algorithm for Dynamic Object Identification and Pose Estimation

by

Chandini Ravindranathan Nair

APPROVED BY:

_____

G.Lan

Odette School of Business

_____

A.Ngom

School of Computer Science

_____

X.Yuan, Advisor

School of Computer Science

May 14, 2019

Declaration of Originality

I confirm here that I am the sole creator of this thesis and no piece of this proposition has been distributed or submitted for production.

I affirm to the best of my insight, my proposition does not encroach upon anybody's copyright nor damage any restrictive rights and their thoughts, methods, citations, or some other material crafted by other individuals incorporated into my postulation, circulation or something else, and are entirely recognized as per the standard referencing hone. Moreover, I have included copyrighted material that outperforms the limits of reasonable managing inside the significance of the Canada Copyright Act. I affirm to acquire a composed consent from the copyright owner(s) to incorporate such material(s) in my postulation and have included duplicates of such copyright clearances in my reference section.

I pronounce this is a genuine copy of my thesis including any last updates, as affirmed by my thesis advisory group and the Graduate Studies office, and this theory has not been submitted for a higher degree to another University or Institution.

Abstract

While object identification enables autonomous vehicles to detect and recognize objects from real-time images, pose estimation further enhances their capability of navigating in a dynamically changing environment. This thesis proposes an approach which makes use of keypoint features from 3D object models for recognition and pose estimation of dynamic objects in the context of self-driving vehicles. A voting technique is developed to vote out a suitable model from the repository of 3D models that offers the best match with the dynamic objects in the input image. The matching is done based on the identified keypoints on the image and the keypoints corresponding to each template model stored in the repository. A confidence score value is then assigned to measure the confidence with which the system can confirm the presence of the matched object in the input image. Being dynamic objects with complex structure, human models in the "COCO-DensePose" dataset, along with the DensePose deep-learning model developed by the Facebook research team, have been adopted and integrated into the system for 3D pose estimation of pedestrians on the road. Additionally, object tracking is performed to find the speed and location details for each of the recognized dynamic objects from consecutive image frames of the input video. This research demonstrates with experimental results that the use of 3D object models enhances the confidence of recognition and pose estimation of dynamic objects in the real-time input image. The 3D pose information of the recognized dynamic objects along with their corresponding speed and location information would help the autonomous navigation system of the self-driving cars to take appropriate navigation decisions, thus ensuring smooth and safe driving.

Dedication

I dedicate this thesis work to God Almighty, my dear husband Mr. Navaneeth Narayanan, my beloved parents Mr. Ravindranathan Nair and Mrs. Hemalatha R Nair, my loving brother Mr. Vimal R Nair and the rest of my family and friends.

Acknowledgements

First and foremost, I would like to express profound thankfulness to my supervisor, Dr. Xiaobu Yuan, who has supported me throughout my thesis with his knowledge and expertise on this exciting field of research. His ideas and suggestions have helped me become more creative, without which I would not have been able to complete this research.

I would like to offer my sincere gratitude to the advisory group members, Dr. Alioune Ngom and Dr. George Lan for their significant remarks and recommendations for my research.

I want to thank, my friend, Mr. Jonathan Ketel, Full Stack Developer, UNI3T, Montreal, for proof reading this thesis book and providing his valuable suggestions.

I would like to thank all my friends, who have supported and helped me throughout my studies, here in Canada. I also thank my parents, husband and family for their blessings and financial support which enabled me to complete my studies successfully.

Table of Contents

Declaration of Originality iii

Abstract iv

Dedication v

Acknowledgements vi

List of Figures ix

List of Abbreviations/Symbols xii

List of Tables xiii

Chapter 1: Introduction 1

1.1 Google's Self Driving Car 3

1.2 Object Recognition – How do self-driving cars see? 4

1.3 Pose Estimation 5

Chapter 2: Literature Review 6

2.1 Object Recognition and Pose Estimation of Objects 6

2.2 Semantic Segmentation Techniques 8

2.2.1 Region Based Semantic Segmentation 10

2.2.2 Fully Convolutional Network Based Semantic Segmentation 13

2.2.3 Weakly Supervised Semantic Segmentation 13

2.3 Object Detection via Bounding Boxes 14

2.4 3D Object Recognition 15

2.5 Object Recognition and Pose Estimation of Cars using 3D Models 24

2.6 Object Recognition and Pose Estimation of Humans using 3D Models 31

2.7 Related Works 39

2.8 Thesis Statement 49

2.8.1 Problem Statement 49

2.8.2 Thesis Contribution 49

Chapter 3: Proposed System 52

3.1 Motivation 52

3.2 Relation of the thesis to other components of the overall system 53

3.2.1 Modules of the overall system directly related to the thesis research 55

3.3 Proposed Techniques for Dynamic Object Recognition and Pose Estimation 57

3.3.1 Object Recognition and Pose Estimation of Cars 58

3.3.2 Object Recognition and Pose Estimation of Humans 62
3.4 Proposed Technique for Dynamic Object Tracking 64
3.5 Design of the Flowchart and Algorithms 67
3.5.1 The Proposed Algorithm for Recognition, Pose Estimation and Tracking of Dynamic Objects in the Real-Time Input Image 70
3.5.2 The Proposed Voting Algorithm for Cars 71
3.5.3 The Proposed Object Tracking Algorithm 74
3.6 Time Complexity of the Proposed Algorithm 75
Chapter 4: Implementation and Experiments 77
4.1 Software Information 77
4.2 Experiments and Results of Object Recognition and Pose Estimation of Cars 78
4.2.1 Simulated input image with a single car 78
4.2.2 Real-time input image with a single car 84
4.2.3 Real-time input image with a car partially occluded by another car 88
4.2.4 Real-time input image with part of the vehicle missing 90
4.2.5 Real-time input image with objects other than cars or pedestrians 92
4.3 Experiments and Results of Object Recognition and Pose Estimation of Pedestrians 93
4.3.1 Input image with an adult on the road 93
4.3.2 Input image with an adult and child on the road 95
4.3.3 Real-time input image with pedestrians running across the road 97
4.3.4 Real-time input image with multiple people on road (Occluded humans) 98
4.4 Results of the Tracking of Dynamic Objects 100
4.5 Comparison and Discussions 102
4.5.1 Mean Average Precision 102
4.5.2 Estimation of Mean Average Precision using the proposed method 103
4.5.3 Results Comparison and Discussion: Cars 105
4.5.4 Results Comparison and Discussion: Humans 108
4.6 Observation and Limitations 110
Chapter 5: Conclusion and Future Work 112
References 114
Vita Auctoris 125

List of Figures

Figure 1: Google's Self-Driving Car prototype 4
Figure 2: Object recognition to identify different objects 6
Figure 3: Pose estimation in humans and cars 7
Figure 4: Object recognition in the context of autonomous cars 8
Figure 5: Semantic segmentation of objects on the road 9
Figure 6: The architecture of R-CNN (Image source: Girshick et al., 2014) 10
Figure 7: The architecture of Fast R-CNN. (Image source: Girshick, 2015) 11
Figure 8: An illustration of Faster R-CNN model. (Image source: Ren et al.,2016) 12
Figure 9: FCN Architecture 13
Figure 10: Weakly supervised semantic segmentation 14
Figure 11: Illustration of YOLO 15
Figure 12: Illustration of strategy suggested by Schmid and Mohr 19
Figure 13: A general 3D object recognition system 21
Figure 14: Graphical representation of SIFT algorithm 22
Figure 15: Process for Bag of Features Image Representation 23
Figure 16: Sample 3D car model (.obj) 24
Figure 17: Image Formation: Pinhole model (Perspective model) 25
Figure 18: Different views of 3D car model 26
Figure 19: Rendered images of 3D car models 27
Figure 20: Training and Inference stages of KeypointNet 28
Figure 21: SMPL model (orange) fit to ground truth 3D meshes (gray) 32
Figure 22: Representation of Standard Skinning 33
Figure 23: SMPL Model Pipeline 34
Figure 24: Part segmentation, Marking Correspondences and Surface Correspondence illustrative representation 35
Figure 25: Left: The image and the regressed correspondence by DensePose-RCNN, Middle: DensePose COCO Dataset annotations, Right: Partitioning and UV parametrization of the body surface 36
Figure 26: The user interface for collecting per-part correspondence annotations 36
Figure 27: The classification and regression steps of DensePose architecture 37
Figure 28: DensePose R-CNN architecture 38
Figure 29: Cross-cascading architecture 39
Figure 30: Flowchart of the overall system 53
Figure 31: Snippet of the repository 58
Figure 32: Object center candidates 60
Figure 33: Flowchart representing the proposed technique for object recognition and pose estimation of dynamic objects 68
Figure 34: Flowchart depicting the proposed technique for object tracking from input video 69
Figure 35: Input image 78
Figure 36: Keypoints detected using KeypointNet 79
Figure 37: Matched model from the repository 80
Figure 38: Image centre coordinates 80
Figure 39: Object center candidates plotted on the image 81

Figure 40: Representation of results in each step 81
Figure 41: Car detected with confidence score = 1.0 82
Figure 42: Step-wise results for the image of a car moving in the "Towards" direction 83
Figure 43: Step-wise results for the image of a car moving in the "Left" direction 83
Figure 44: A real time road scene with a single car 84
Figure 45: Cropping the detected car using ImageAI python library 85
Figure 46: Keypoint features identified using KeypointNet 85
Figure 47: Matched model from the repository 86
Figure 48: Object centre and object centre candidates 86
Figure 49: Step-wise representation of results for real time single car scenario 87
Figure 50: Car detected with confidence score = 1.0 in real time image 87
Figure 51: Real time road scene with multiple cars 88
Figure 52: Cars Detected and cropped using the ImageAI library and the Resnet model 89
Figure 53: Step-wise results for the image of a car partly occluded by the tyre of another car 89
Figure 54: Tractor unit without a trailer on the road 90
Figure 55: Step-wise results for vehicle with a missing part 91
Figure 56: Input images with no car or pedestrians 92
Figure 57: Keypoint features detected on images without cars or pedestrians 92
Figure 58: Input image with an adult on the road 94
Figure 59: Visualization of the isocontours of the UV fields in the image with an adult on the road 94
Figure 60: Textures mapped on the isocontours of the UV fields in an image with an adult on the road 95
Figure 61: Input image with an adult and a child crossing the road 95
Figure 62: Visualization of the isocontours of the UV fields in the image with an adult and a child crossing the road 96
Figure 63: Textures mapped on the isocontours of the UV fields for an image with an adult and a child crossing the road 96
Figure 64: Input image with two men running across the road 97
Figure 65: Visualization of the isocontours of the UV fields in an image with two men running across the road 97
Figure 66: Textures mapped on the isocontours of the UV fields for an image with two men running across the road 98
Figure 67: Input image with multiple pedestrians (few partly occluded) crossing the road 98
Figure 68:Visualization of the isocontours of the UV fields in the image with multiple pedestrians (few partly occluded) crossing the road 99
Figure 69: Textures mapped on the isocontours of the UV fields for an image with multiple pedestrians (few partly occluded) crossing the road 99
Figure 70: Different frames of the input video tracking the car with the estimated speeds displayed on it 100
Figure 71: Different frames of the input video tracking the humans with the estimated speeds displayed on it 101
Figure 72: Object recognition with Resnet model 106
Figure 73: Pictorial representation of how the proposed approach improves prediction confidence 107
Figure 74: Step-wise representation of results for input image of car with partial occlusion 107
Figure 75: Objects detected incorrectly instead of cars (Source: Tangruamsub et al.,2011) 108
Figure 76: Woman wearing a long skirt 109

Figure 77: Pose estimation results after applying DensePose model 110
Figure 78: Two cars in two opposite directions with keypoints at similar coordinate positions 111
Figure 79: Failure case with output keypoints flipped in the case of the KeypointNet model 111

List of Abbreviations/Symbols

HPE	Human Pose Estimation
CNN	Convolutional Neural Network
R-CNN	Region Based Convolutional Neural Network
SMPL	Skinned Multi-Person Linear Model
YOLO	You Only Load Once
SURREAL	Synthetic hUmans foR REAL tasks
LiDAR	Light Detection and Ranging
ROI	Region of Interest
FCN	Fully Convolutional Network
SIFT	Scale Invariant Feature Transformation
SOA	Service Oriented Architecture
SURF	Speeded Up Robust Features

List of Tables

Table 1: Review of object recognition and pose estimation methods 44
Table 2: Time complexity of the algorithm 76
Table 3: List of tools used for implementation of the proposed system 77
Table 4: Confusion Matrix 103
Table 5: Comparison of mAP 104

Chapter 1: Introduction

Human society is moving towards a future where cars will be able to drive by themselves and travelers can just sit back and relax. Apart from the major players like Tesla, Google and Uber, there are many other companies who have invested billions of dollars towards research in this area to make the dream of a fully autonomous car a reality. With so much investment and interest in driverless technology, it is easy to assume that self-operating cars are imminent, but they are still far from being viable. Creation of such a sophisticated machine requires great expertise to ensure smooth and safe driving. Object recognition and pose estimation of recognized objects are the two most important tasks in the application of vehicle surveillance [91]. An accurate solution to these problems will help with automatically analyzing a traffic scene. This includes the analysis of crucial information for vehicle navigation like determining the vehicle speed, traffic frequency, driver behaviour, or the shape and style of traffic participants. It is a known fact that there would be a variety of objects on the road at any given time. These objects can be either moving or stationary objects. In the context of self-driving vehicles, it is important to distinguish between the moving and stationary objects in order to take the correct navigation decisions. Kolski et al. [74] describes the objects like pedestrians and other vehicles which move on the road as dynamic obstacles. Darms et al. [75] clearly defines static obstacles as objects which are assumed not to move during the observation period, such as, buildings, traffic cones or parked cars which do not participate actively in traffic. In contrast, Darms et al. [75] define dynamic obstacles as objects which potentially move during the observation periods. Darms et al. [76] and Hu et al. [77] categorize the moving objects as "dynamic objects" and the objects which do not move as "static objects" in their works. Hu et al. [77] include blockages such as stones and construction signs into the category of static objects. This research solely concentrates on object recognition and pose

estimation of moving objects like cars and pedestrians, on the road. More formally, this thesis deals with object recognition and pose estimation of "dynamic objects" on the road.

A system with very high obstacle detection accuracy is required for a self-driving car, as lower accuracies can lead to errors which can be dangerous and cost human lives [78]. Most of the previous researches [1, 28, 84] use the mean average precision(mAP) (Section 4.5.1) as the measure of the accuracy in the object detection and pose estimation predictions. For example, the current state-of-the-art technique YOLOv2 has achieved a mAP of 78.6 [28]. Janai et al. [81] discuss the importance of reliable object detection in the context of self-driving vehicles, in order to avoid accidents that might be life-threatening. The work also discusses the difficulties in object detection due to the wide variety of object appearances and occlusions caused by other objects. Gao et al. [79] mention the bottlenecks in accuracy, efficiency, and timeliness of detection, recognition, tracking, and segmentation techniques based on traditional RGB images due to insufficient depth information. Gene Lewis [80] discusses the challenges in the object detection problem ported to the self-driving vehicles, such as the optimization problems and the speed issues of the object detection pipeline. Hence, even though 100% accuracy is almost impossible, there is a need for continuous improvement in the confidence level with which an object is identified, until it is safe to put self-driving vehicles on the road.

The aim of this research is to improve the confidence with which the system can confirm the presence of an object or obstacle at a location in the input image. An object is identified, and its 3D pose is determined by making use of keypoint information from the 3D object models. A voting algorithm is developed to vote out the best 3D model from the repository corresponding to the dynamic object present in the input image. The voting algorithm also estimates a confidence score, which signifies the confidence with which the system can confirm the presence of the identified

object in the input image. The term “voting” was adopted for this algorithm from a similar work done by Tangruamsub et al. [1], where the authors used keypoint features from 2D images (instead of 3D models used in this thesis) to vote out the best candidates for object recognition and pose estimation. The identified object is then tracked to get its speed and location information. The estimated pose information along with the speed and location information is then used to update the recognized dynamic object on to a virtual city which acts as a repository of both the static and dynamic objects on the road in real-time.

1.1 Google’s Self Driving Car

Many companies like Tesla, Mercedes, Uber, Google, BMW, and Volvo already have their semi-autonomous cars on the market. However, all of these require expensive hardware support and are still far from being fully autonomous vehicles. Among them is Google’s Self Driving Car. Google was the first company to make customer oriented driverless cars, but it is still far from viability. The car uses many highly advanced pieces of hardware for obstacle detection and therefore the overall cost of the vehicle is a major problem. However, this cannot be avoided as the vehicle needs to be able to detect and avoid obstacles like other cars, pedestrians, cyclists and animals. Google’s Self-Driving Car uses an array of detection technologies including sonar devices, stereo cameras, lasers, and radar. The LiDAR on top of the car is considered the heart of object detection. Google’s LiDAR system is great for generating an accurate map of the area surrounding the vehicle, but it is not ideal for monitoring the speed of other cars in real time. Therefore, the front and back bumper of the driverless car include radar. Hence, it is clear that many expensive hardware devices are currently being used for accurate object recognition.

Figure 1: Google's Self-Driving Car prototype

1.2 Object Recognition – How do self-driving cars see?

In addition to reducing human effort on the road, it is anticipated that self-driving cars will make driving safer and therefore reduce the number of traffic accidents and fatalities. This will require these vehicles to have highly accurate object recognition techniques. A large amount of machine learning and computer vision research is currently focused on this area. Basically, there are two categories of object recognition in the context of autonomous driving: - Semantic Segmentation and Object Detection using rectangular bounding boxes [11]. However, these methods are mostly done with 2D images. The novel idea used by the proposed system of this research makes use of the keypoint feature information from 3D object models to identify the dynamic objects from a real-time input image. The use of keypoint features extracted from 3D object models for object matching and identification is anticipated to improve the confidence of object recognition.

1.3 Pose Estimation

In the context of self-driving vehicles, the pose estimation of objects on the road is very significant in the autonomous navigation of the vehicle [12]. Pose estimation of recognized objects can be classified into two:

a. Pose estimation of objects which do not change its structure significantly.
b. Pose estimation of objects whose structure may change.

Examples of objects whose structure does not significantly change include vehicles such as cars and trucks. Whereas examples of objects whose structure does change significantly include people, animals, and birds. This research also focuses on the pose estimation of identified objects on the road, which will provide additional information for the autonomous navigation systems of the self-driving vehicles, to take good navigation decisions.

In this thesis, Chapter 2 contains a detailed review of the previous work done on object recognition and pose estimation in autonomous vehicles. It also includes the prerequisite techniques for the proposed approach. Chapter 3 explains the proposed system in depth. Chapter 4 discusses the different scenarios that were considered during the implementation of the proposed idea along with the results obtained. Chapter 4 also compares and discusses the results obtained using the proposed approach with the other existing approaches. Chapter 5 includes the conclusion of the anticipated impact in this area of research.

Chapter 2: Literature Review

This chapter discusses the relevant background of recent works in object recognition and pose estimation in the context of self-driving vehicles. This section also covers the technical background of object recognition and pose estimation using 2D and 3D images and how it can be extended with the proposed approach of using 3D object models to improve object recognition and pose estimation tasks.

2.1 Object Recognition and Pose Estimation of Objects

Object recognition is a computer vision technique that identifies objects in images or videos. Humans can easily spot people, objects, and other visual details when they view a photograph or watch a video. The goal of machine learning is to teach a computer to do what comes naturally to humans, specifically to gain a level of understanding of what objects an image contains. Figure 2 below shows an illustration of using an object recognition algorithm to identify if an object is a cat or a dog [11].

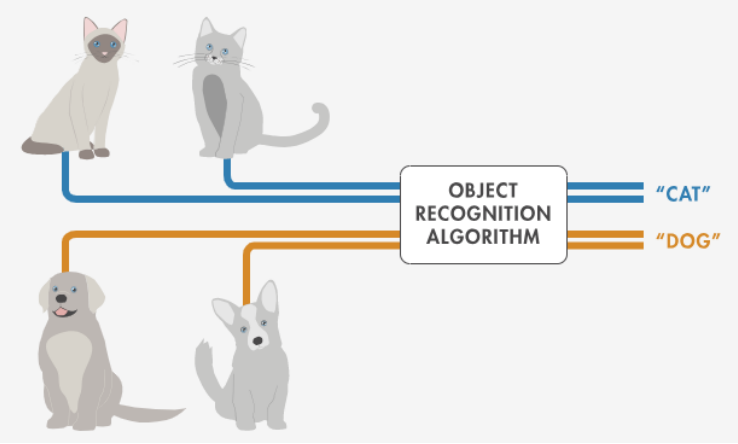

Figure 2: Object recognition to identify different objects

Object recognition is one of the most important technologies used in autonomous vehicle navigation. It allows the recognition of vehicles and pedestrians on the road or other objects such as a lamppost or a stop sign.

Pose estimation is a general problem in computer vision where the position and orientation of an object are detected [12]. It detects the keypoint locations that describe the object. Many prior works have been done on pose estimation of objects having skeletal structures such as humans, birds or animals, as well as on objects whose structure does not significantly change like that of cars. Figure 3 below depicts pose estimation in humans and cars [12].

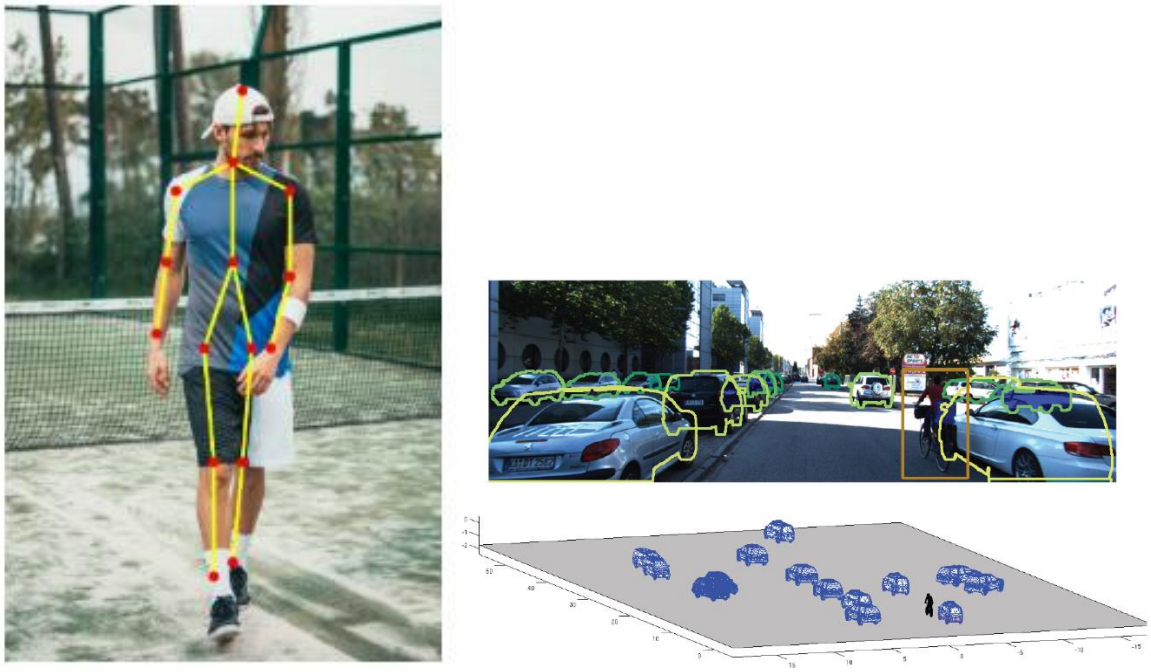

Figure 3: Pose estimation in humans and cars

Many researchers have explored different techniques for object recognition and subsequent pose estimation of objects on the road to improve the accuracy of object detection and ensure safe driving. Apart from the fact that driverless cars reduce human efforts on the road, it is anticipated

to make driving safer and reduce the number of road accidents. This motive requires that these cars need exemplary object recognition techniques. Plenty of different machine learning and computer vision researches are happening in this area every day.

Figure 4: Object recognition in the context of autonomous cars

There are two major categories of object recognition in the context of autonomous driving:

a. Semantic Segmentation Technique
b. Object Detection via bounding boxes

The different approaches adopted so far in both the categories are discussed in detail in the below sections.

2.2 Semantic Segmentation Techniques

The semantic segmentation technique is understanding an image at the pixel level. In semantic segmentation each of the pixels gets labeled to identify an object belonging to a specific class. In other words, a class is assigned to each pixel in the image.

Many of the existing advanced deep learning networks for self-driving vehicles are based on semantic segmentation and objects are detected based on edge detection. Also, many pieces of research like the works done by Teichmann et al. [14] and Ros et al. [15] deal with autonomous driving technology based on the semantic segmentation technique.

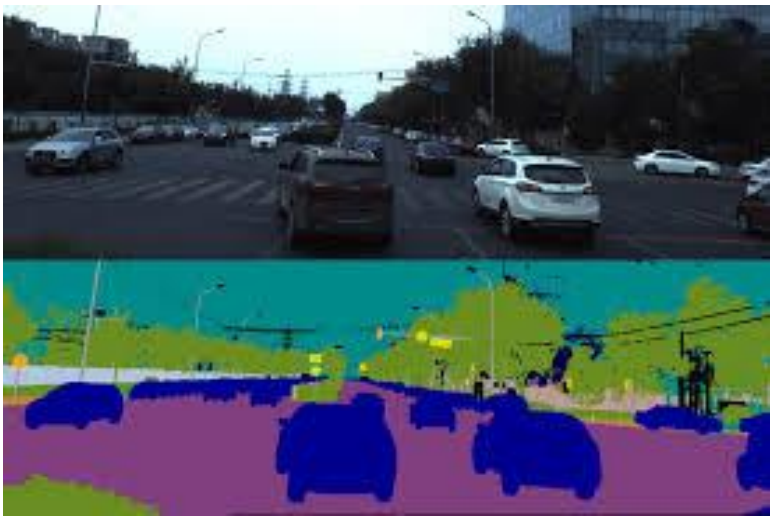

Figure 5: Semantic segmentation of objects on the road

Some standard deep networks which are used as the basis of semantic segmentation systems are AlexNet (Krizhevsky et al. [15]), VGG-16 (Simonyan et al. [16]), GooLeNet (Szegedy1 et al. [17]), ResNet (He et al. [18]).

Basically, a semantic segmentation architecture can be broadly considered as an encoder followed by a decoder.

There are three main approaches which are currently adopted:

a. Region Based Semantic Segmentation
b. Fully Convolutional Network Based Semantic Segmentation
c. Weakly Supervised Semantic Segmentation

2.2.1 Region Based Semantic Segmentation

The region-based methods generally follow the "segmentation using recognition" pipeline, which first extracts free-form regions from an image and describes them, followed by region-based classification. There are three main region based deep learning segmentation approaches which were once state-of-the-art before the YOLO: R-CNN, Fast R-CNN and Faster R-CNN.

2014: R-CNN – An early application of CNN to object detection – The goal of R-CNN is to take an image, and then correctly identify where the specific objects are within that image. The main idea of R-CNN is composed of two steps. First, using selective search, it extracts about 2000 regions from the image. These are called the region proposals.

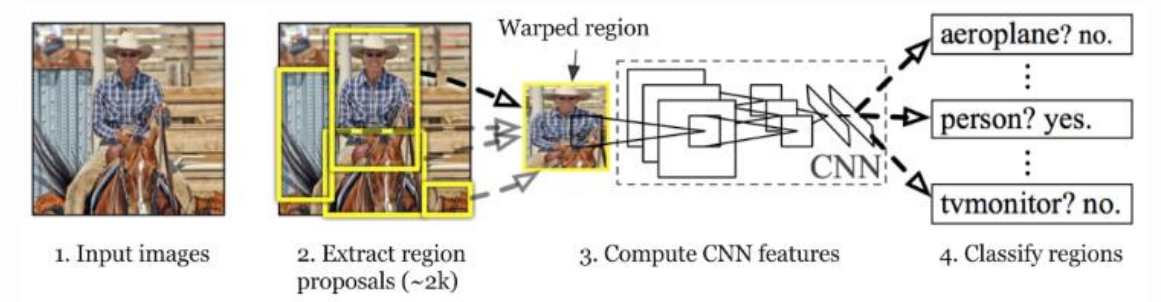

Figure 6: The architecture of R-CNN (Image source: Girshick et al., 2014)

These region proposals are wrapped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The convolutional neural network acts as a feature extractor and the output dense layer consists of features extracted from the image. These extracted features are then fed into a support vector machine, to classify the object present within that candidate region proposal. However, the R-CNN has some issues. The training time of the network is very high, as 2000 region proposals per image need to be classified. It is not easy to implement the R-CNN in real time as it takes around 47 seconds for a single test image. Also,

since the selective search algorithm is a fixed algorithm, no learning happens at that stage. This could lead to bad candidate region proposals.

2015: Fast R-CNN – Speeding up and Simplifying R-CNN – In Fast R-CNN instead of feeding the region proposals to the CNN, the input image is fed to the CNN to generate a convolutional feature map. The region of proposals is identified from the convolutional feature map and are then wrapped into squares.

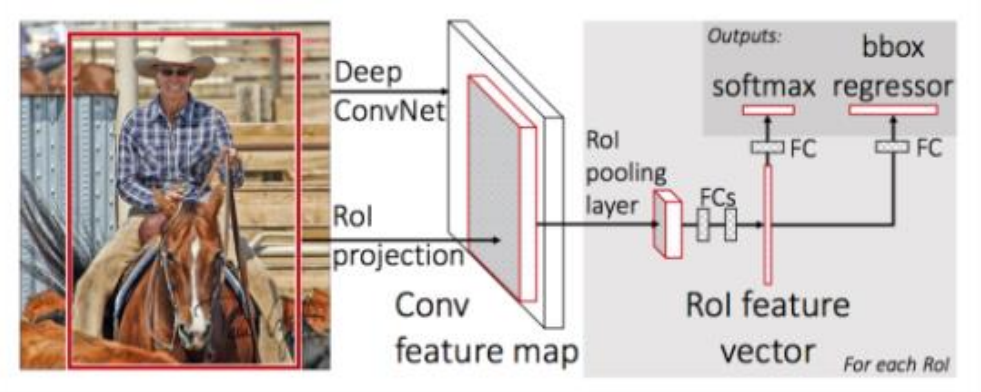

Figure 7: The architecture of Fast R-CNN. (Image source: Girshick, 2015)

By using a RoI pooling layer, it is reshaped into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, a softmax layer is used to predict the class of the proposed region and the offset values for the bounding box. Fast R-CNN is faster than R-CNN as there is no need to feed the 2000 region proposals to the convolutional neural network every time. Instead, as mentioned above, the convolution operation is done only once per image and the feature map is generated from it.

2016: Faster R-CNN – Speeding up Region Proposal

Both R-CNN and Fast R-CNN uses selective search to find out the region proposals, which is a slow and time-consuming process affecting the performance of the network. Faster R-CNN is an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals. Just like the approach in Fast R-CNN, the image is provided as an input to the convolutional network which provides a convolutional feature map.

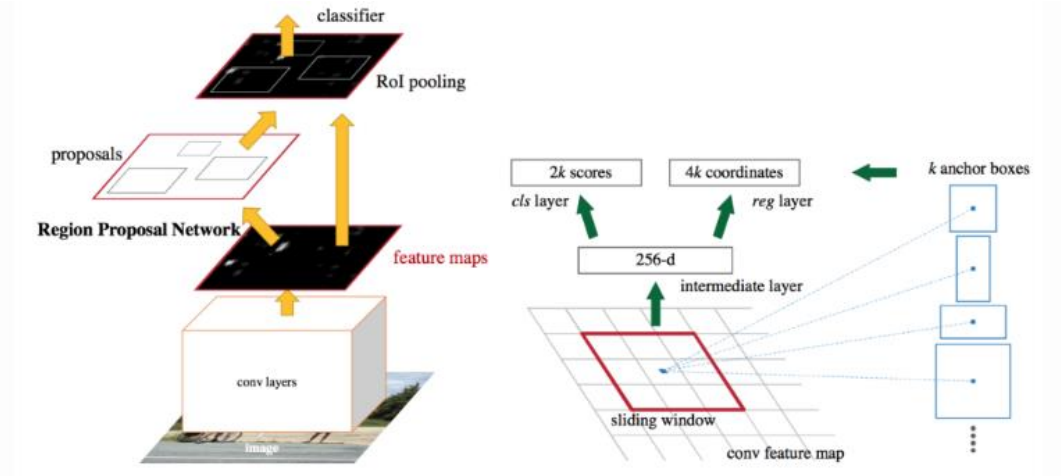

Figure 8: An illustration of Faster R-CNN model. (Image source: Ren et al.,2016)

Instead of using the selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The RoI pooling layer then reshapes the predicted region proposals and is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

2.2.2 Fully Convolutional Network Based Semantic Segmentation

The original Fully Convolutional Network (FCN) learns the mapping from pixels to pixels, without extracting the region proposals. The main idea of FCN is to make the classical CNN take arbitrary-sized images as input. CNNs have fixed fully connected layers which restricts them to accept and produce labels for specifically sized inputs only. However, the FCNs have only convolutional and pooling layers which give them the ability to make predictions on arbitrary-sized inputs. Figure 9 below shows the FCN architecture [46].

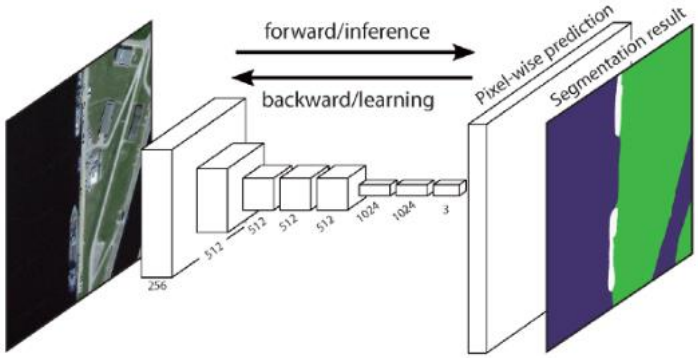

Figure 9: FCN Architecture

2.2.3 Weakly Supervised Semantic Segmentation

Most of the relevant methods in semantic segmentation rely on many images with pixel-wise segmentation masks. However, manually annotating these masks is quite time-consuming, frustrating and commercially expensive. The weakly supervised methods are dedicated to fulfilling the semantic segmentation by utilizing annotated bounding boxes.

Figure 10: Weakly supervised semantic segmentation

2.3 Object Detection via Bounding Boxes

This technique bounds the location of an object in the frame with a rectangular box. There are many high performing object detection models which detect multiple objects in a scene simultaneously in a very short time. The bounding boxes are a simpler way to describe the location of an object when compared to the segmentation technique. The CNNs could usually only classify images with a single object that take a sizable portion of it. This issue was solved using the sliding window approach. However, to identify the different object with various sizes, multiple window sizes are required which needs to be slide over the image. However, this is computationally very expensive and hence the YOLO was introduced. In case of the YOLO network, the image is split up into a grid and the entire image is run through a convolutional neural network.

YOLO – You Only Load Once

You only load once (YOLO) is a state-of-the-art, real-time object detection system. YOLO applies a single neural network to the full image. This network divides the image into regions and predicts the bounding boxes along with probabilities for each region. These bounding boxes are weighted

by the predicted probabilities. One of the advantages of YOLO is that it looks at the whole image during the test time. Unlike R-CNN, which requires thousands of networks for a single image, YOLO makes predictions with a single network. This makes the algorithm extremely fast, over 1000 times faster than R-CNN and 100 times faster than Fast R-CNN. Figure 11 shows the illustration of the YOLO [28].

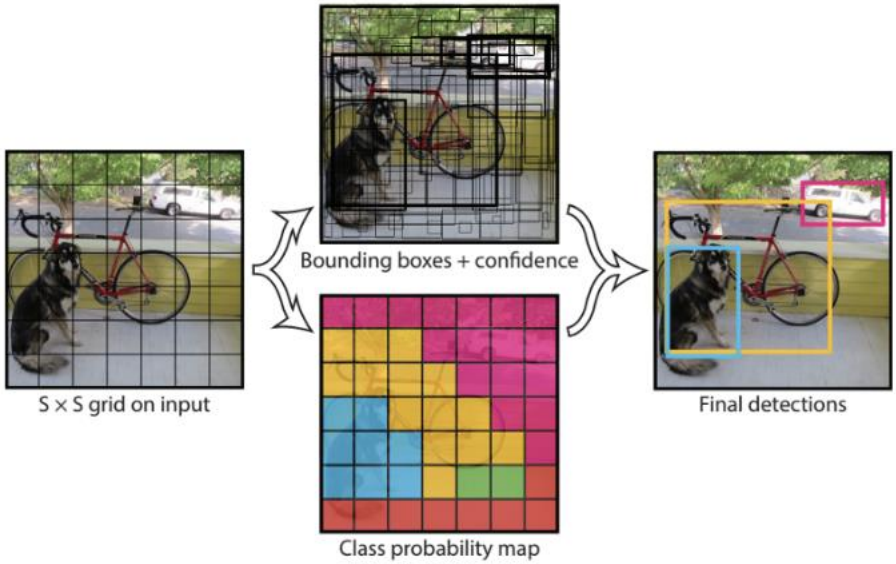

Figure 11: Illustration of YOLO

2.4 3D Object Recognition

With the increased availability of 3D data and the 3D sensors popularization, the 3D object classification and recognition area have had a growing boost in the last few years. There are various applications for the methods developed in this area which range from the field of robotics, directed to robot movement in environments and object manipulation by robotic arms, to the security domain, where the techniques of recognition and classification of 3D objects are used to detect possible danger objects. When considering a sophisticated technology like the autonomous

vehicles which deals with lives of people on the road, it seemed to be a good idea to make use of the 3D model information of the different objects on the road for recognizing them in the real-time images.

Object recognition is one of the basic application domains in computer vision. Extensive research has been and is still being done in this area, especially in 3D. 3D object recognition can be defined as the task of finding and identifying objects in the real world from an image or a video sequence. Object recognition is still a hot research topic in computer vision because it has many challenges such as viewpoint variations, scaling, illumination changes, partial occlusion, and background clutter. Many approaches and algorithms are proposed and implemented to overcome these challenges.

The object recognition research community can be split into two: Those who deal with 2D images and those who deal with 3D point clouds or meshes. By projecting the scene onto a plane by capturing the light intensity detected at each pixel, the 2D images can be created. Alternatively, 3D point clouds capture the 3D coordinates of points in the scene. The main difference between two dimensional and three-dimensional data is that 3D data includes depth information whereas 2D data does not. Cheaper sensors have been developed to acquire the 3D data from the real environments, such as RGB-D cameras. Mustafa et al. [47] and Krystof et al. [48] proposes works which make use of RGB-D cameras for object recognition. RGB-D cameras, such as the Microsoft Kinect, capture a regular color image (RGB) along with the depth (D) associated with each pixel in the image. Since the approach is being applied to the self-driving vehicle domain, it is important to first evaluate the different requirements and constraints for this application before deciding on the 3D object recognition technique. Various applications impose different requirements and constraints as discussed below:

a) Evaluation Time: In most of the industrial applications, the data need to be processed in real-time. In the case of self-driving vehicles, evaluation time is crucial to determine the correct navigation of the car in real time. The evaluation time depends strongly upon the number of pixels covered by the object as well as the size of the image area to be examined.
b) Accuracy: In the autonomous driving application, the object position needs to be determined very accurately. The error bounds cannot exceed more than a fraction of a pixel, else it might result in disastrous events.
c) Recognition ability: In the scenario of autonomous driving, it is required that the rate of false detection must be almost zero to ensure safe driving.
d) Invariance: In the autonomous driving scenario, it is worthwhile to achieve invariance with respect to illumination, scale, rotation, background clutter, occlusion, and viewpoint changes.

The 3D object recognition techniques can be categorized into four groups: geometry-based methods, appearance-based methods, three-dimensional object recognition schemes, and descriptor-based methods [29, 49,50,51,52]. In geometry- or model-based object recognition, the knowledge of an object's appearance is provided by the user as an explicit CAD-like model. Only the 3D shape is described and properties such as color and texture are not included. In contrast, the Appearance-based methods do not require an explicit user-provided model for object recognition. The object representations are usually acquired through an automatic learning phase, and the model typically relies on surface reflectance properties.

Some methods intend to locate the 3D position of an object in a single 2D image, essentially by searching for features which are invariant to viewpoint position. The overall system proposed in this research tries to make use of this approach, where the 3D models of the objects are rendered,

and the 3D keypoints are determined by selecting the feature points that are invariant to viewpoint position. This explained in detail in the next section.

Geometry – or – model-based object recognition techniques have many advantages like invariance to viewpoint and illumination. Also, they have a well-developed theory as many effective algorithms exist for analyzing and manipulating geometric structures. In contrast, most recent research efforts have been centered on appearance-based techniques such as advanced feature descriptors and pattern recognition algorithms. They typically include two phases. In the first phase, from the set of reference images, a model is constructed. The set includes the appearance of the object under different orientations, different illuminants and potentially multiple instances of a class of objects, for example, cars. In the second phase, parts of the input image (sub-images of the same size as the training images) are extracted, possibly by segmentation (by texture, color, motion) or by exhaustive enumeration of image windows over the whole image. The recognition system then compares an extracted part of the input image with the reference images. However, the major limitation of the appearance-based approaches is that they require isolation of the complete object of interest from the background and is hence sensitive to occlusion and require good segmentation.

The three-dimensional object recognition schemes are used by some applications that require a position estimate in 3D space and not just in the 2D image plane. Such systems make use of sensors which generates 3D data and perform matching in 3D space. Another way to determine the 3D pose of an object is to estimate the projection of the object location in 3D space onto a 2D camera image. Such a data representation is not “full” 3D yet and therefore is often called 2.5D.

Now, when it comes to object recognition in "real-world" scenes, characterization with geometric primitives like lines or circular arcs is not suitable and the algorithm must compensate for the

heavy background clutter and occlusion. Schmid and Mohr [53] suggested a two-way method for image-content description. In the first step, the keypoints are detected, i.e. points that exhibit salient characteristic like the corner of a building. Subsequently, for each interest point, a feature vector called region descriptor is calculated. Each region descriptor characterizes the image information available in a local neighborhood around one interest point. Figure 12 illustrates the strategy suggested by Schmid and Mohr [53].

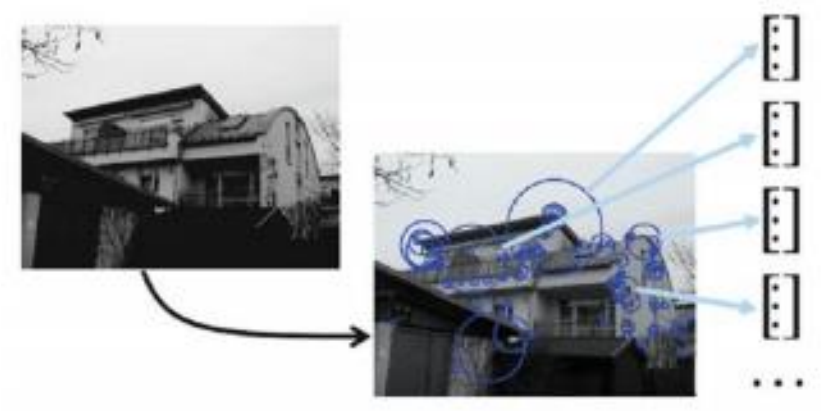

Figure 12: Illustration of strategy suggested by Schmid and Mohr

Object recognition can then be performed by comparing the information of region descriptors detected in a scene image to the information of region descriptors in a model database. A similar technique is being adopted in the proposed method. However, the proposed approach extracts keypoint features from the 3D object models and stores them in a repository.

The 3D data are obtained using different methods which makes use of extra hardware, for e.g. sensors. Most of the semi-autonomous cars now in the market, use a variety of expensive sensors for this purpose. Cost is hence one of the drawbacks of the autonomous cars today. Stereo photogrammetry or photogrammetry based on a block of overlapped images is the primary

approach for 3D mapping and object reconstruction using 2D images. Acquisition from acquired sensor data is done using a variety of sensors, including stereo cameras, time of fight laser scanners such as LiDARs, as well as infrared sensors such as the Microsoft Kinect or Panasonic DI-Imager. All these sensors can only capture a single view of the object with a single scan. This view is referred to as a 2½D scan of the object. Therefore, to capture the entire 3D shape of the object, the sensor captures multiple instances of the object from different viewpoints. The process of determining the similarity between the scene object and the model stored in a repository is one of the important tasks in 3D object recognition. This is done by computing the distance between feature vectors. In general, the recognition must search among the possible candidate features for identification of the best match and then assign the label to the matched object in the scene [54]. Based on the characteristics of the shape descriptor, the 3D object recognition methods can be divided into two main categories: global feature methods and local feature methods [54]. Figure 13 below shows a general 3D object recognition system [29]. Points from an image which gives the best definition for an object are called the keypoint features and they are very important and valuable in applications of image processing like object detection, object and shape recognition, and object tracking. Extracting these keypoint features helps to find the same objects in another image. Hence, it could be said that keypoints give the best information from an image.

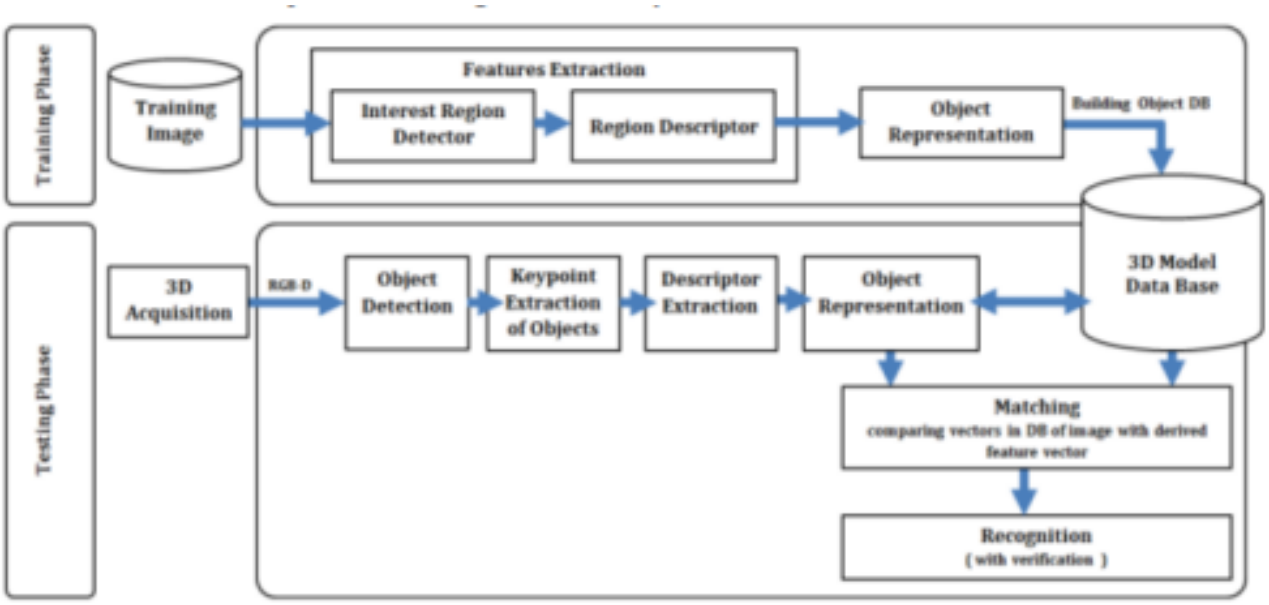

Figure 13: A general 3D object recognition system

Some of the important feature extraction techniques are discussed briefly below:

A. Harris Corner Detector: Harris and Stephens [55] developed an approach to extract corners and infer the contents of an image. In the context of the autonomous vehicles, it requires that the car recognizes static objects like buildings on the roadside. The detection of corner points of buildings will give more information regarding the pose and shape of the buildings. A corner is so special because, as it is the intersection of two edges and represents a point in which the directions of these two edges change. The Harris corner detector is popular because it is independent of rotation, scale, and illumination variations.

B. The SIFT Algorithm: Lowe [56] developed a feature detection and description technique called SIFT (Scale Invariant Feature Transformation). The keypoints are extracted and described as a vector. The resulting vectors can be used to find reliable matches between different images for object recognition, camera calibration, and 3D reconstruction. SIFT consists of three basic stages. First, the keypoints are extracted from the image. Then, these

keypoints are described as 128 vectors. Finally, the last step is the matching stage. This is depicted in Figure 14 below [29].

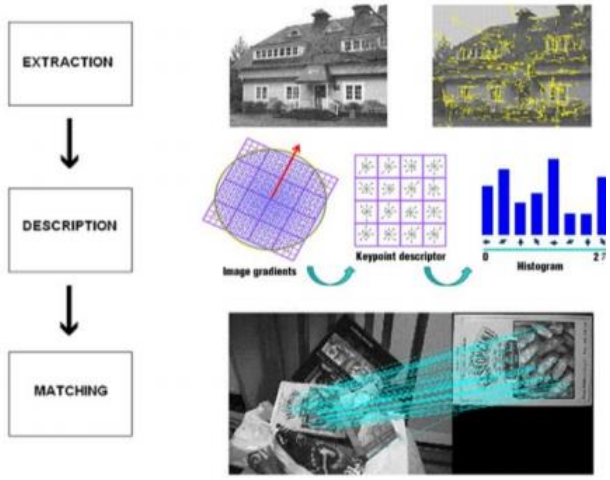

Figure 14: Graphical representation of SIFT algorithm

C. The SURF Algorithm: Bay et al. [57] developed the SURF Algorithm which is based on the SIFT algorithm [56]. It achieves higher speed than SIFT using integral images and approximations. These integral images are used for convolution. Like SIFT, SURF also works in three main stages: extraction, description, and matching. The difference between SIFT and SURF is that SURF extracts the features from an image using integral images and box filters.

D. The ORB Algorithm: The ORB algorithm has several advantages over the more established vector-based descriptors such as SIFT and SURF. ORB [58] is scale and rotation invariant, robust to noise and affine transformations. The algorithm is a combination of the FAST keypoint detection with oriented keypoints added to the algorithm.

E. The Bag-Of-Features Algorithm: A Bag of Features method proposed by Stephen et al. [59], represents images as order less collections of local features. At a high level, the

procedure for generating a Bag of Features image representation is illustrated in Fig 15 and summarized as follows [59]:

1) Build Vocabulary: Extract features from all images in a training set. Vector quantize, or cluster, these features into a "visual vocabulary," where each cluster represents a "visual word" or "term." In some works, the vocabulary is called the "visual codebook." Terms in the vocabulary are the codes in the codebook.

2) Assign Terms: Extract features from a novel image. Use Nearest Neighbors or a related strategy to assign the features to the closest terms in the vocabulary.

3) Generate Term Vector: Record the counts of each term that appears in the image to create a normalized histogram representing a "term vector." This term vector is the Bag of Features representation of the image. Figure 15 below represents the Process for Bag of Features Image Representation [29].

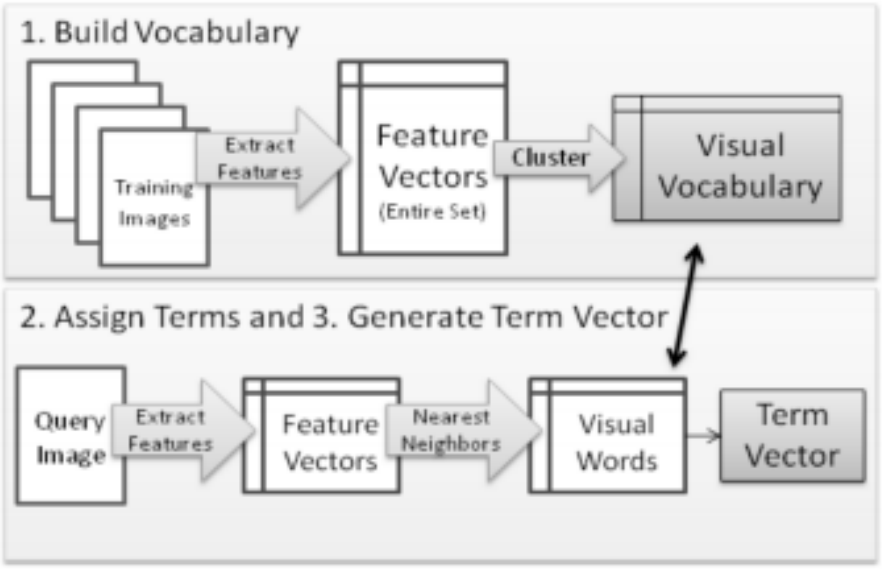

Figure 15: Process for Bag of Features Image Representation

All the previous works have done 3D object recognition from three different aspects: 3D object recognition from range images, 3D local descriptors, and 3D object recognition based on stereo vision. The method proposed in this research aims to use similar keypoint feature extraction technique followed by feature matching approach, using 3D object models. The KeypointNet [23] model is used for extracting the keypoint features from the 3D models after rendering it which serves as input to the proposed system which will be discussed in detail in Chapter 3.

2.5 Object Recognition and Pose Estimation of Cars using 3D Models

The proposed system uses 3D models in the .obj file format for building up the input repository with the keypoint features extracted from these models. Figure 16 below shows a sample 3D car model (.obj). The OBJ file extension is known as Wavefront 3D Object File which was developed by Wavefront Technologies. The .obj is a file format used for a three-dimensional object containing 3D coordinates (polygon lines and points), texture maps, and another object information [61].

Figure 16: Sample 3D car model (.obj)

The .obj images need to be rendered to extract the optimal set of keypoints for recovering the relative pose between two views of an object.

Rendering of the 3D model

The image formation is the process by which a 3D representation of a scene is reduced to a 2D representation of that same scene, an image.

3D scene → transformation → 2D image

Figure 17 below illustrates the Image Formation: Pinhole model [62].

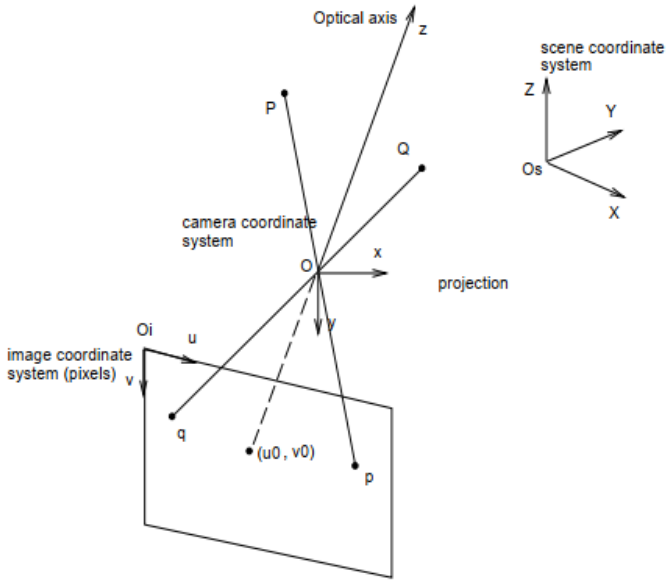

Figure 17: Image Formation: Pinhole model (Perspective model)

An image point (pixel), given by its image coordinates, is the result of a three-step transformation of a physical point defined in a scene reference frame.

The following three steps are applied in sequential order: 3D Euclidean Transformation, 3D-2D transformation, 2D-2D transformation.

a. A 3D Euclidean Transformation – 3D rigid displacement where a scene point, initially defined in the scene reference frame, is transformed so that they would be defined in the

camera reference frame. This transformation has 6 parameters corresponding to a 3D rotation and 3D translation.

b. A 3D-2D Transformation – 3D points defined in the camera reference frame are projected onto the image plane. These new points are called normalized coordinates.

c. A 2D-2D Transformation – The normalized coordinates expressed in the scene metrics, undergo a 2D affine transformation to become defined in pixels in the image plane reference frame.

After the three steps, a perspective projection of the 3D point, P = (X, Y, Z, 1) onto the pixel p = (u, v, 1) is done using the projection matrix information. u and v can be defined using below equations [62]:

$$u = \frac{m11X+m12Y+m13Z+m14}{m31X+m32Y+m33Z+m34}$$

$$v = \frac{m21X+m22Y+m23Z+m24}{m31X+m32Y+m33Z+m34}$$

It is clear from the above equation of u and v that, u and v coordinates make use of the X, Y, Z information of the image plane onto the pixel, thus saving the depth information.

Results of rendering a car model (.obj)

Consider the 3D car model from ShapeNet [60] dataset, shown in Figure 18 below.

Figure 18: Different views of 3D car model

Blender tool along with python scripting is used to render the 3D object model to 2D images with different possible orientations. The script also generates the rigid relative transformations of the different 3D object views onto the camera reference frame as discussed before.

For the 3D car model shown in Figure 18, below are some of the rendered images in different views.

Figure 19: Rendered images of 3D car models

The next task is to extract keypoints from these rendered images which would serve as input into the proposed algorithm of this research. The approach by Suwajanakorn et al. [23], presents "KeypointNet" which is an end-to-end geometric reasoning framework to learn an optimal set of category-specific 3D keypoints, along with their detectors. This architecture is being used by the proposed system to extract the keypoint features from the 3D models. The details of KeypointNet are discussed below.

KeypointNet

All the prior works used supervised task for finding keypoints from 3D objects, where a list of keypoints is given, and the goal is to proximate to those keypoints. However, selection and consistent annotation of keypoints in images of an object category are expensive and ill-defined. The KeypointNet model[23], finds the optimal set of 3D keypoints for a downstream task, without keypoint ground truth. Basically, when a rendered image is passed through KeypointNet, it predicts an ordered list of 3D keypoints, defined as pixel coordinates and associated depth values.

The network focuses on the task of relative pose estimation at training time, were given two views of the same object with a known rigid transformation T, the aim is to predict optimal lists of 3D keypoints, P1 and P2 in the two views that best match one view to the other as depicted in Fig 20 below [23].

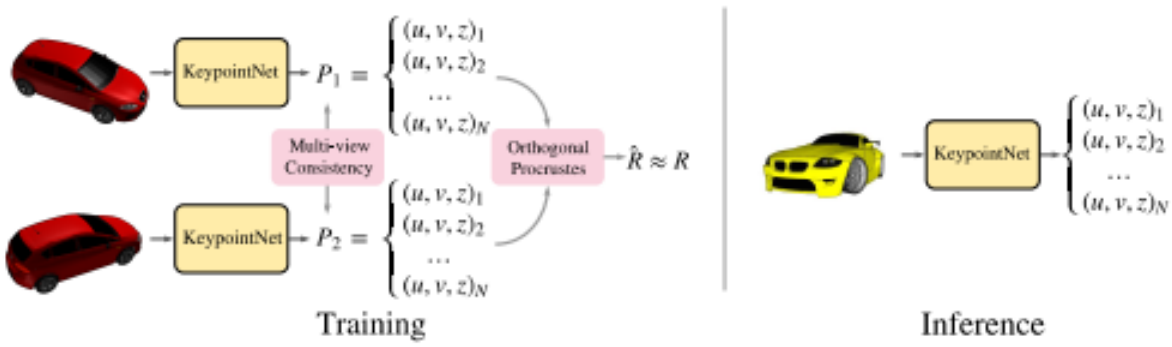

Figure 20: Training and Inference stages of KeypointNet

An objective function O (P1, P2) is formulated, based on which one can optimize a parametric mapping from an image to a list of keypoints. The objective consists of two primary components:

• A multi-view consistency loss that measures the discrepancy between the two sets of points under the ground truth transformation.

• A relative pose estimation loss, which penalizes the angular difference between the ground truth rotation R vs. the rotation $\hat{R}$ recovered from P1 and P2 using orthogonal procrustes.

There is a pair of images (I, I') of the same object from different viewpoints in each training tuple, along with their relative rigid transformation T ϵ SE (3) which transforms the underlying 3D shape from I to I'. T has the following matrix form:

$T = \begin{bmatrix} R^{3\,x\,3} & t^{3\,x\,1} \\ 0 & 1 \end{bmatrix}$, where R and t represents 3D rotation and translation respectively.

The goal of the multi-view consistency loss is to ensure that the keypoints track consistent parts across different views. To ensure that a 3D keypoint in one image projects onto the same pixel location as the corresponding keypoint in the second image, the approach assumes a perspective camera model with a known global focal length f. [x, y, z] denotes 3D coordinates, and [u, v] denotes pixel coordinates. The projection of a keypoint [u, v, z] from the image I into the image I' (and vice versa) is given by the projection operators:

$$[\hat{u}, \hat{v}, \hat{z}, 1]^T \sim \pi T \pi^{-1}([u, v, z, 1]^T)$$

$$[\widehat{u'}, \widehat{v'}, \widehat{z'}, 1]^T \sim \pi T^{-1} \pi^{-1}([u', v', z', 1]^T)$$ [23]

where, for instance, $\hat{u}$ denotes the projection of u to the second view, and $\hat{u}$' denotes the projection of u' to the first view. Here, π: R4 ⟶ R4 represents the perspective projection operation that maps an input homogeneous 3D coordinate [x, y, z, 1] ^{T} in camera coordinates to a pixel position plus depth:

$$\pi([x, y, z, 1]^T) = [\frac{fx}{z}, \frac{fy}{z}, z, 1]^T = [u, v, z, 1]^T$$

The symmetric multi-view consistency loss is defined as [23]:

$$L_{con} = \frac{1}{2N} \sum_{i=1}^{N} ||[ui, vi, ui', vi']^T - [\hat{u}i', \tilde{v}i', \hat{u}i, \tilde{v}i]^T||^2$$

The pose estimation objective is a differentiable objective that measures the misfit between the estimated relative rotation $\hat{R}$ (computed via Procrustes' alignment of the two sets of key points) and the ground truth R. The pose estimation objective is defined as [23]:

$$L_{pose} = 2 \arcsin(\frac{1}{2\sqrt{2}}) \|\check{R} - R\|_F)$$

Empirically, the pose estimation objective helps significantly in producing a reasonable and natural selection of latent keypoints, leading to the automatic discovery of interesting parts such as the wheels of a car. This is because these parts of the car are geometrically consistent within an object class (e.g., circular wheels appear in all cars), easy to track, and spatially varied, all of which improve the performance of the downstream task. Hence, this approach for keypoint detection seemed very appropriate for the proposed system.

KeypointNet Architecture

In order to ensure the translation equivariance for the mapping from images to keypoints, the network outputs a probability distribution map gi(u; v) that represents how likely keypoint i is to occur at pixel (u, v), with $\Sigma_{u,v}$ $g_i(u, v) = 1$. A spatial softmax layer is used to produce such a distribution over image pixels. Then the expected values of these spatial distributions are computed to recover a pixel coordinate as [23]:

$[u_i, v_i]^T = \sum_{u,v}[u \,.\, gi(u,v), v \,.\, gi(u,v)]^T$

For the z coordinates, a depth value is predicted at every pixel, denoted d_i (u, v), and computed as [23]:

$z_i = \sum_{u,v} di(u,v)gi(u,v)$

All kernels for all layers are 3×3, and 13 layers of dilated convolutions is stacked with dilation rates of 1, 1, 2, 4, 8, 16, 1, 2, 4, 8, 16, 1, 1, all with 64 output channels except for the last layer which has 2N output channels, split between g_i and d_i. LeakyRelu and Batch Normalization is used for all layers except the last layer. The output layers for di have no activation function, and the channels are passed through a spatial softmax to produce g_i. Finally, g_i and d_i are then converted to actual coordinates pi using the above equations.

2.6 Object Recognition and Pose Estimation of Humans using 3D Models

In the context of autonomous driving, there are a variety of objects to consider on the road. In the scope of this research, only dynamic objects on the road are considered. As discussed earlier, this can be mainly classified into two groups: 1) The objects whose structure does not change significantly, for e.g. cars, trucks, cycle and 2) Objects whose structure changes significantly, for e.g. humans, animals, and birds. Once the object recognition is done and the keypoints are detected and matched, it gives the coordinate information of the keypoints and the matched 3D model of the recognized object from the repository along with the pose information.

The pose estimation in case of objects whose structure doesn't change much is straight forward. As discussed in section 2.5, KeypointNet uses the pose estimation objective which helps significantly in producing a reasonable and natural selection of latent keypoints, leading to the automatic discovery of interesting parts of the object, such as the wheels of a car. Hence the keypoint coordinate information itself gives the pose of the car on the road. In the proposed system, the direction of car movement (Left, Right, Towards, Away) is retrieved from the annotation file of the matched model. This information is then used by our virtual city module (discussed in chapter 3) for updating the recognized object information in the virtual city.

The pose estimation for the objects whose structure changes, like that of humans is not this easy. A good literature survey was done in this area due to the complexity of determining the 3D pose and shape of the humans on road. Since most of the approaches only dealt with single humans in an image, it didn't seem convenient for use in the proposed system of autonomous driving as there would be always multiple people on the road. However, the approach by Guler et al. [40], called the "DensePose", establishes dense correspondences between an RGB image and a surface-based representation of the human body, referred to as dense human pose estimation. The approach

introduces the first manually-collected ground truth dataset for the task, by gathering dense correspondences between the SMPL model [41] and the person appearing in COCO dataset. The SMPL model is briefly explained below, before diving deep into the approach and implementation details of the DensePose.

SMPL: A Skinned Multi-Person Linear Model

Realistic human body models are important for many graphics, economics and computer vision applications. Many of the realistic models are from data but are not compatible with the existing rendering engines. In contrast, the SMPL model accurately represents a wide variety of body shapes in natural human poses. The parameters of the model are learned from data including the rest pose template, blend weights, pose-dependent blend shapes, identity-dependent blend shapes, and a regressor from vertices to joint locations. Figure 21 below shows the SMPL model(orange) fit to ground truth 3D meshes(gray) [41].

Figure 21: SMPL model (orange) fit to ground truth 3D meshes (gray)

SMPL is compatible with existing graphics pipeline because it is based on standard skinning methods. Figure 22 below represents the standard skinning [64].

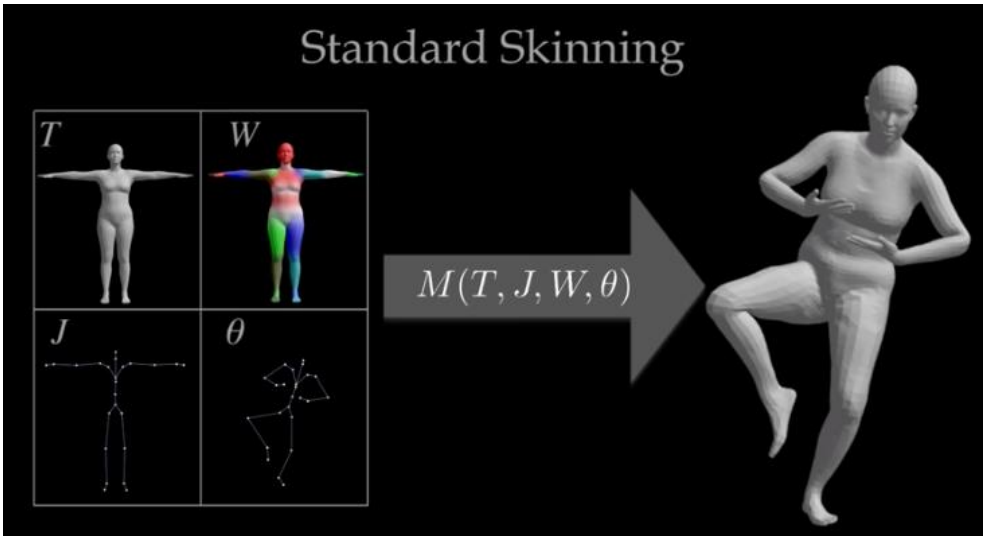

Figure 22: Representation of Standard Skinning

A skinned body model defines the vertices of a template T and a rest pose, joint positions J and blend weights W. Given the pose of the skeleton θ, skinning computes the vertex locations of the mesh using a linear blending of the vertices based on rotations of different parts. SMPL contains template mesh T in an initial pose and sets to the template to represent the new body shapes and pose-dependent shape changes. From training scans the shape blend shapes are learned, that capture the variation in human shape. Adding different combinations of shape blend shapes produces different body shapes. SMPL predicts the joint positions for a given body shape as the function of mesh vertices. These are shown in white dots in Fig 23 below. From training scans of people in many poses, the pose blend shapes are learned that capture how real bodies differ from blend skin bodies. Given a pose, SMPL computes the linear contribution of these blend shapes, the correct skinning errors and produce realistic pose-dependent deformations. Finally, SMPL uses standard blend skinning to transform the deformed template shape to the desired pose. The shape blend shapes are learned from approximately 4000 body scans from US and European CEASAR [42] datasets.

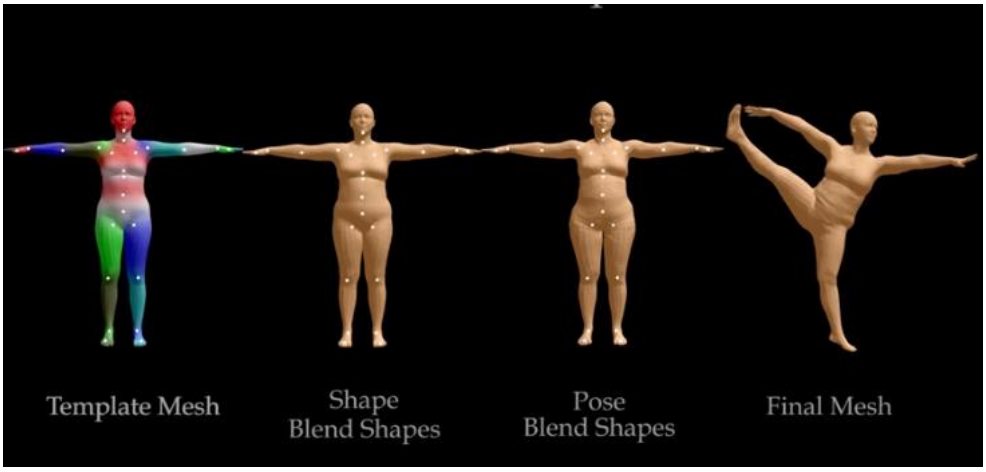

Figure 23: SMPL Model Pipeline

This SMPL models are used in the "DensePose" for 3D human pose estimation.

DensePose: Dense Human Pose Estimation in the Wild

The DensePose basically establishes dense correspondences between an RGB image and surface-based representation of the human body. The authors (Guler et al. [40]) mapped out every single pixel of a human body in the video/image. Any human in an image or video is a 2D grid of pixels, which humans can indeed tell is a 3D object represented by a 2D grid. The same must be achievable by the machine, i.e. transform the 2D human into a 3D model. DensePose tries to create a "correspondence", which is a computer vision term that is a measure of how well the pixels in one image correspond to pixels in another image. Here it is a 2D to 3D image correspondence. Since it requires that all the pixels be as close together as possible it is called dense correspondence. This method would require some object detection, object segmentation and pose estimation. As shown in Figure 24 below, in the first stage the annotators were asked to delineate regions corresponding to visible, semantically defined body parts like the Head, Torso, Lower/Upper Arms, Lower/Upper

Legs, Hands, and Feet. Figure 24 below shows Part segmentation, Marking Correspondences and Surface Correspondence [40].

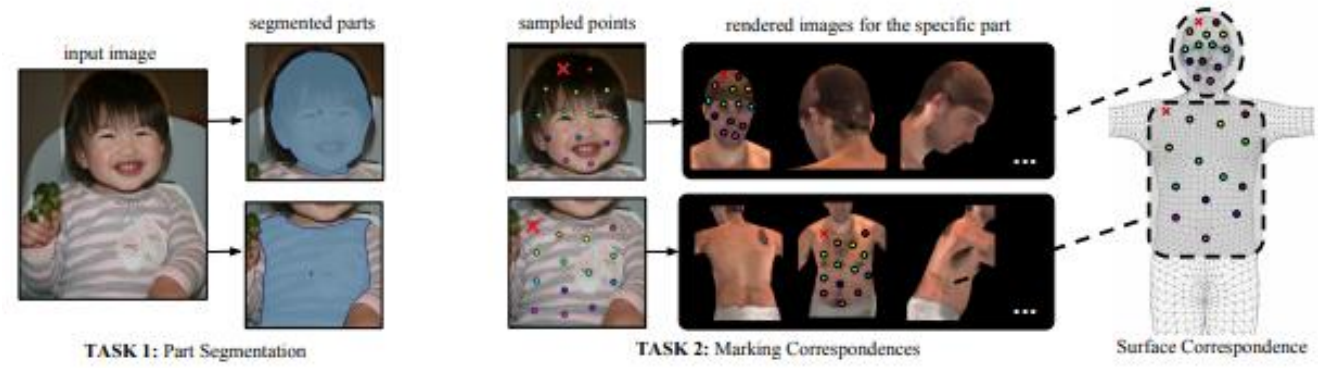

Figure 24: Part segmentation, Marking Correspondences and Surface Correspondence illustrative representation

In Figure 24, the red cross indicates the currently annotated point. The surface coordinates of the rendered views localize the collected 2D points on the 3D model.

For heads, hands, and feet, the annotators used the manually obtained UV fields provided in the SMPL model [41]. For rest of the parts, the annotators obtained the unwrapping via multi-dimensional scaling applied to pairwise geodesic distances. The UV fields for the resulting 24 parts are visualized in Figure 25 below. All these annotations where labelled with their corresponding 3D body part which acted as the label. This was done for 50K human images of the COCO dataset which summed up to be a total of 5 million manually annotated correspondences yielding the new DensePose-COCO dataset. The annotators estimated the body part behind the clothes so that for instance wearing a large skirt would not complicate the subsequent annotation of correspondences. In the second stage, every part region is sampled with a set of roughly equidistant points obtained via k-means. In order to simplify this task, the part surface is 'unfolded' by providing six pre-rendered views of the same body part and allows to place landmarks on any

of them (Figure 26). In Figure 25, the picture in left shows the image and the regressed correspondence by DensePose-RCNN. The middle image shows the DensePose COCO Dataset annotations and the picture in the right shows Partitioning and UV parametrization of the body surface [40].

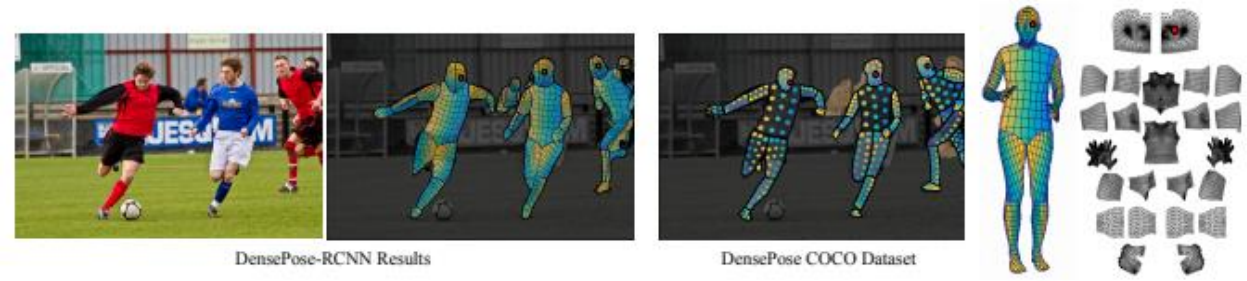

Figure 25: Left: The image and the regressed correspondence by DensePose-RCNN, Middle: DensePose COCO Dataset annotations, Right: Partitioning and UV parametrization of the body surface

This allows the annotator to choose the most convenient point of view by selecting one among six options instead of manually rotating the surface. As a point is indicated on any of the rendered part views, its surface coordinates are used to simultaneously show its position on the remaining views – this gives a global overview of the correspondence. Figure 26 below shows the user interface for collecting per-part correspondence annotations [40].

Figure 26: The user interface for collecting per-part correspondence annotations

The next task is to train a deep network that predicts dense correspondences between image pixels and surface points. The authors of DensePose [40], introduce improved architectures by combining the DenseReg [65] approach with the Mask-RCNN architecture [66], yielding the 'DensePose-R-CNN' system. Cascaded extensions of DensePose-RCNN are developed that further improve accuracy. In the first step, a network classifies a pixel as belonging to either the background or one of the several region parts that give a rough estimate of the surface coordinates. This is essentially a labeling task that can be trained using gradient descent. In the second step, a regression model would indicate the exact coordinates of the pixel within the region part. Formally, in the first stage it will assign position L to the body part C that has the highest likelihood as calculated by the classification branch and in the 2nd stage it would use the regressor to place the point L in the continuous coordinate pair (u, v) as shown in Figure 27 below [67].

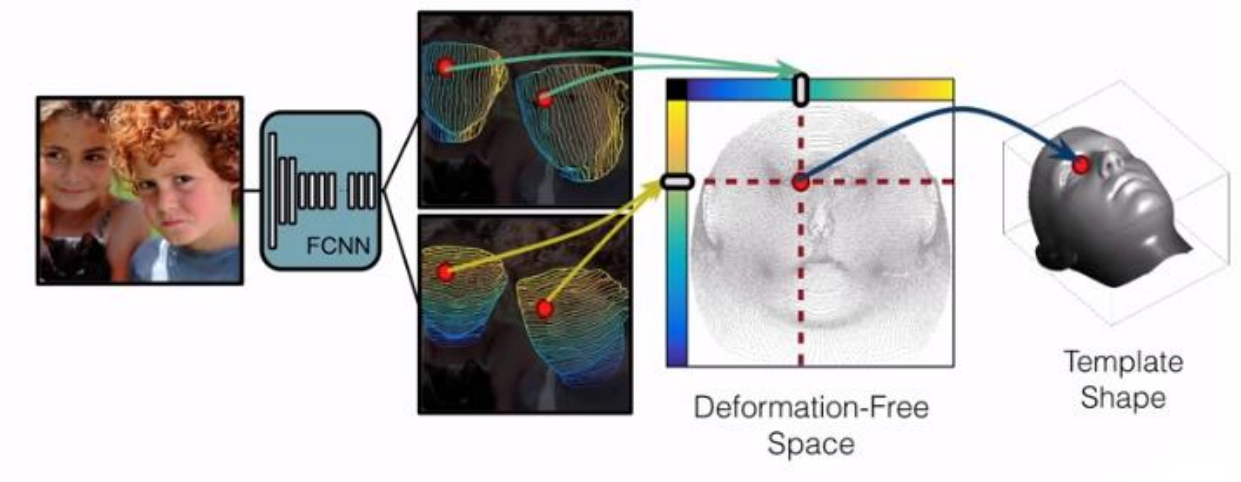

Figure 27: The classification and regression steps of DensePose architecture

Both the classification and regression tasks are trained by minimizing a respective loss function. But the regression loss is only considered for a part if the pixel is within a specific part. It requires a lot of task for a single network, like part segmentation and pixel localization. The technique of

Region of Interest pooling is used to create regions and feed the resulting features into region-specific branches. This decomposes the complexity of the task into controllable modules, all of which could be trained jointly in an end to end approach. Therefore, it is a fully convolutional network on top of ROI pooling, that is, entirely devoted to two tasks. That is generating a classification and regression head that provide part assignment and part coordinate predictions. DensePose-R-CNN architecture is shown in Figure 28.

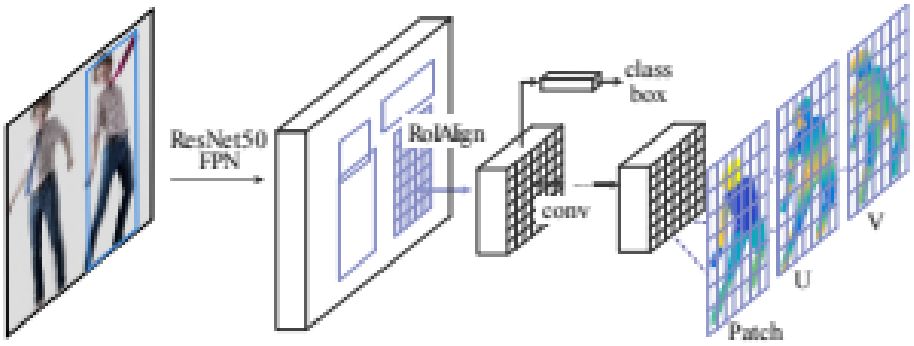

Figure 28: DensePose R-CNN architecture

To improve the accuracy of the model a technique called cascading is used. Cascading means using a collection of models with all the collected information from the output of one model as additional information for the next classifier in the cascade.

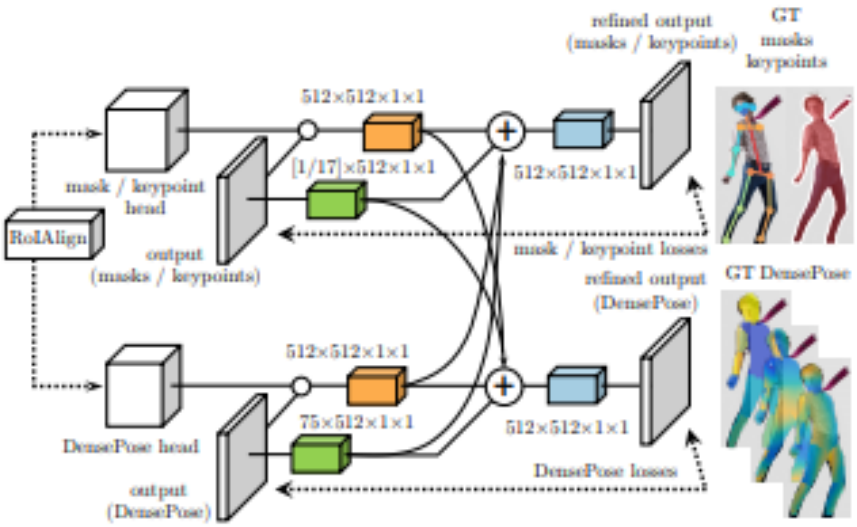

Figure 29: Cross-cascading architecture

The output of the ROI aligns module feeds into the dense network as well as network for masking and keypoint tasks. The first stage predictions from all tasks are then combined and fed into a second stage refinement unit of each branch.

'DensePose' predicts the pose of multiple humans and humans occluded by other humans or objects even in situations with a lot of distractions. It also predicts body parts behind the clothing. This makes it the best approach to be adopted into the overall proposed system.

2.7 Related Works

Table 1 below depicts the prominent works done so far by the researchers in this area along with their accomplishments and scope of improvements.

Title	Author	Year	Accomplishments	Scope of Improvement
Discovery of latent 3D keypoints via end-to-end geometric reasoning	S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi.	2018	An end-to-end geometric reasoning framework based on deep learning architecture, to learn an optimal set of category-specific 3D keypoints from rendered 3D models of cars, that are optimized for the downstream pose estimation task.	No confidence measures of the detected keypoints on the real-time input images are estimated to evaluate the performance of pose estimation.
YOLO9000: Better, Faster, Stronger	Joseph Redmon, Ali Farhadi	2016	A state-of-the-art, real-time object detection system using deep learning architecture, that can detect over 9000 object categories with high accuracies.	Uses the normal 2D image COCO dataset for the deep learning network and still has scope to improve the accuracy when used for sophisticated systems like autonomous

				vehicles which deals with human lives on the road.
3D YOLO: end-to-end 3D object detection using point clouds	Ezeddin Al Hakim	2018	A LiDAR based 3D object detection model that operates in real-time, with emphasis on autonomous driving scenarios. The proposed model takes point cloud data as input and outputs 3D bounding boxes with class scores in real-time.	Less accurate, when compared to YOLOv2, which is currently the state-of-the-art.
3D object recognition using a voting algorithm in a real-world environment	S. Tangruamsub, K. Takada, and O. Hasegawa	2011	An object detection and pose estimation method based on a voting technique, using keypoint features extracted from images giving improved	Uses normal 2D images and an existing feature extraction technique like SIFT or SURF and does not exploit the 3D object models as is done our approach.

			average precision and detection time.	
Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach	Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, Yichen Wei	2017	A weakly-supervised transfer learning method that uses mixed 2D and 3D labels in a unified deep neural network which gives the 3D pose of the human in the image with good results.	Even though the 3D poses with (x, y, z) coordinates are determined a complete reconstruction of the human 3D model for the pose in the real-time input image is not done.
Deeply Learned Compositional Models for Human Pose Estimation	Wei Tang, Pei Yu and Ying Wu	2018	Exploits deep neural networks with a hierarchical compositional architecture and bottom-up/top-down inference stages to learn the compositionality of human bodies. In addition, the approach proposes a novel bone-	Uses 2D images and manual annotations for all images which is labour intensive. Does not necessarily give the 3D information of the human in the image.

			based part representation.	
Unite the People: Closing the Loop Between 3D and 2D Human Representations	Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, Peter V. Gehler	2017	An approach to create high quality 3D body model fits for multiple human pose datasets called the UP-3D dataset with rich annotations. This dataset is then used to train models which predicted 31 segments and 91 landmark locations on the human body giving state-of-the-art results for 3D human pose and shape estimation.	The 3D model configuration does not always match the image evidence and it only recovers a rough pose. It also does not handle occluded or multi-person scenarios.
Neural Body Fitting: Unifying Deep Learning and Model-Based	Mohamed Omran Christoph Lassner Gerard Pons-Moll Peter	2018	The approach integrates a statistical body model within a CNN, leveraging reliable bottom-up	It does not support challenging settings involving multiple, possibly occluded, people.

Human Pose and Shape Estimation	V. Gehler Bernt Schiele		semantic body part segmentation and robust top-down body model constraints. It presents a robust, efficiently trainable framework for 3D human pose estimation from 2D images	
DensePose: Dense Human Pose Estimation In The Wild	Rıza Alp Güler, Natalia Neverova, Iasonas Kokkinos	2018	An approach to establish dense correspondence between human pixels of an RGB image to the 3D surface of human body. This seems to be very suitable in the context of autonomous vehicles as it handles large amounts of occlusion, scale, and pose variation.	This approach has space for improvement when applied to a different domain like the self-driving vehicles. For example, integrating our voting algorithm to this approach would help improve the confidence of pose estimation.

Table 1: Review of object recognition and pose estimation methods

The previous researches in object recognition used different approaches with the state-of-the-art being deep learning approaches, like YOLO. In most of the previous approaches either semantic segmentation or object detection using rectangle bounding boxes was applied. For example, AlexNet (Krizhevsky et al. [16]), VGG-16 (Simonyan et al. [17]), GooLeNet (Szegedy1 et al. [18]) and ResNet (He et al. [19]) all delivered promising deep learning architectures with high accurate results, for use in semantic segmentation tasks of objects. A region based semantic segmentation approach was used by R-CNN (Girshick et al. [25]) in 2014. This was first replaced by a version with improved performance, the Fast-R-CNN (Girshick et al. [26]) in 2015 and later by the Faster-RCNN (Girshick et al. [27]) in 2016. These approaches were once state-of-the-art before YOLO (Redmon et al. [28]) was introduced. YOLO tremendously decreased the computation time of training the model by loading the image into the network only once, unlike the other methods which use the selective search algorithm which requires greater computational time. All these approaches use 2D images to train their models and to perform object recognition.

Object recognition is still a hot research topic in computer vision because it has many challenges such as viewpoint variations, scaling, illumination changes, partial occlusion, and background clutter. The major difference between two- and three-dimensional data is that 3D data include depth information which is not included in 2D data. There are many previous works which were implemented to overcome these challenges. The method proposed in this research aims to make use of the keypoint features extracted from 3D models to improve the confidence with which the system can confirm the presence of a recognized object at a location. Hence, a deep review was done on the previous works in different domains which did object recognition by extraction of the keypoint features. The survey paper by Carvalho et al. [8] comprises works of around 446 different approaches for object recognition using features of 3D models.

Many previous works make use of keypoint feature extraction from 2D images, RGB-D images, 3D models and point cloud data for solving the object recognition problem of 3D objects in different domains. For example, Tangruamsub et al. [1], adopted an approach of learning and matching between model and image objects using a voting technique for object recognition and posterior pose estimation in the context of the car domain. Shimamura et al. [3] adopted a method of local feature extraction, followed by a candidate's verification on the database and posterior matching evaluation through a geometric verification based on consistency constraints to handle 3D viewpoint changes under cluttered scenes for robust object recognition. Wohlkinger et al. [4] used RGB-D images with CAD to address the problem of real-time 3D shape-based object class recognition. The work presents its scaling to many categories and the reliable perception of categories using a novel shape descriptor for partial point clouds based on shape functions which are capable of training on synthetic data and classifying objects from a depth sensor in a single partial view in a fast and robust manner. The approach by Socher et al. [5] uses a deep learning architecture based on a combination of convolutional and recursive neural networks (CNN and RNN) for learning features and classifying RGB-D images. Chi Li et al. [6] uses RGB-D images for semantic scene segmentation, partitioning the scene in different object regions, and object pose estimation through a model registration method.

As mentioned earlier, the approach used in this research is different from the above methods as the proposed idea aims to extract keypoint features from 3D models of objects on the road and apply a voting algorithm to match it with the real-time input image features and determine the presence of the object with high confidence value along with the pose of the recognized object.

Shah et al. [7] uses 3D object models to present a local surface description technique for automatic three-dimensional (3D) object recognition using detection of highly repeatable keypoints by

computing the divergence of the vector field at each point of the surface. Suwajanakorn et al. [23] uses the approach of an end-to-end geometric reasoning framework based on deep learning architecture, to learn an optimal set of category-specific 3D keypoints from rendered 3D car models which are optimized for the downstream pose estimation task. The KeypointNet model [23] is used in this research for extracting the keypoint features from the 3D models and the real-time input images, which serve as input to the proposed algorithm. This is explained in detail in Chapter 3.

Unlike objects which do not significantly change their structure, such as cars or trucks, object recognition and 3D pose estimation in case of the humans is complicated due to the complex structure and rich variation in poses, clothing, hairstyle, body shapes, occlusions, viewpoints, motion blur, and other factors. Various works have been done in this area recently and hence it is currently one of the hot topics in computer vision. Tang et al. [2] and Zhou et al. [34] use 2D image datasets as proposed by Sapp et al. [30], Johnson et al. [31], Andriluka et al. [32] and Ionescu et al. [33], for the human pose estimation task. Zhou et al. [34] uses a weakly-supervised transfer learning method that uses a 2D pose estimation sub-network with a 3D depth regression sub-network in a unified deep neural network which gives the 3D pose of humans in the image with good results. Tang et al. [2] exploit deep neural networks with hierarchical compositional architecture and bottom-up/top-down inference stages to learn the compositionality of human bodies along with a novel bone-based part representation. The 2D keypoint prediction in humans has seen considerable progress in recent years and could be considered nearly solved in the works by Tang et al. [2], Insafutdinov et al. [9], Newell et al. [10] and Ramakrishna et al. [44].

The above discussed approaches do not make use of 3D human body models. Since the overall approach of the proposed system tries to utilize 3D models; works using 3D models to determine

human pose and shape were reviewed further. However, 3D pose estimation from single images remains a challenge. An approach to creating high-quality 3D human body model fits for multiple human pose datasets called the UP-3D dataset with rich annotations is used in the work by Lassner et al. [38]. This dataset is then used to train models which predicted 31 segments and 91 landmark locations on the human body giving state-of-the-art results for 3D human pose and shape estimation. Varol et al. [35] and Saint et al. [37] present 3D human datasets. In the work by Saint et al. [37], the authors present a dataset named, 3DBodyTex, which consists of 3D body scans with high-quality texture information along with a fully automatic method for body model fitting to a 3D scan. The approach used in Varol et al. [35] presents the SURREAL (Synthetic hUmans foR REAL tasks) dataset with synthetically generated but realistic images of people rendered from 3D sequences of human motion capture data. The approach by Loper et al. [41] is used by most of the human pose estimation research to generate the 3D human body shapes. SMPL (A skinned Multi-Person Linear Model) [41] is a realistic 3D model of the human body that is based on skinning and blend shapes and is learned from thousands of 3D body scans. Based on the parameters provided it generates body images from the learned thousands of 3D body shapes. Omran et al. [39] use an approach that integrates a statistical body model within a CNN, leveraging reliable bottom-up semantic body part segmentation and robust top-down body model constraints. It presents a robust, efficiently trainable framework for 3D human pose estimation from 2D images. However, in Lassner et al. [38], the 3D model configuration does not always match the image evidence and only recovers a rough pose. The approach used in both the works of Lassner et al. [38] and Omran et al. [39] cannot handle occluded or multi-person scenarios.

Guler et al. [40], introduced an approach to establish dense correspondences between human pixels of an RGB image to the 3D surface of the human body. This seems to be very suitable in the

context of autonomous vehicles as it handles large amounts of occlusion, scale, and pose variation and hence this approach is adopted by the proposed system for human pose estimation discussed later in Chapter 3.

2.8 Thesis Statement

2.8.1 Problem Statement

The literature suggests that there is need for continuous improvement in the confidence level with which an object is identified, until it is safe to put self-driving vehicles on the road. The input into the proposed system is the image frames of the video of a real-time road scene captured by the camera attached to a self-driving vehicle. The main goal of this thesis is to perform object recognition and subsequent pose estimation of the dynamic objects from these image frames with the help of 3D object model information stored in the repository. The object tracking module provides additional information like the speed and location of the recognized object. The combination of the 3D information of the recognized dynamic object along with its speed and location is the output from the system and is anticipated to allow the autonomous navigation system of the self-driving vehicles to recognize the presence of a dynamic object at a given location with improved confidence which helps the system to take appropriate navigation decisions, thus ensuring smooth and safe driving.

2.8.2 Thesis Contribution

The major contributions of this thesis can be summarized as follows:

- The proposed system matches keypoint features of the dynamic objects in the input image with the keypoint feature information of 3D object models stored in the repository to find a suitable matching 3D model for each of the dynamic objects present in the input image. A voting algorithm has been developed for this purpose which also estimates a confidence score that signifies the confidence of the object identification. The results of the proposed approach show that this 3D model information improves the confidence of recognition and pose estimation of the dynamic objects in the input image.
- Once the cars and pedestrians in the input image are recognized, the proposed system tracks these objects in order to get their location and speed information.
- The 3D dynamic object model information along with their speed and location details are then used to update a virtual city with these objects in real-time.
- The information of the dynamic objects from the virtual city can then be used by the autonomous navigation system of the self-driving vehicles to take appropriate navigation decisions.

The previous works on recognition and pose estimation of dynamic objects, in the context of autonomous driving show the need for continuous improvement in the confidence level of obstacle detection before it is safe to put these self-driving vehicles on the road. The proposed system improves the confidence level of recognition and subsequent pose estimation of dynamic objects on the road. This thesis concentrates solely on cars and pedestrians under the dynamic object category. In the case of cars, the keypoint features of the cars in the input image are matched with the keypoint feature information of 3D car models stored in the repository. This research makes use of the KeypointNet [23] model to extract keypoints corresponding to very specific parts of a car (for example wheels and headlights). This helps in finding a suitable matching 3D car model

from the repository for the car identified in the input image. In the case of pedestrians, the complex structure of the human body makes the pose estimation task cumbersome. Building a 3D human model repository is a labour-intensive task which is beyond the scope of this thesis work. Hence, the DensePose [40] deep learning network developed by the Facebook research team is adopted and integrated into the proposed system of this research. DensePose [40] establishes dense correspondences between the pedestrians in the input image and a 3D surface-based representation of the human body. Once the cars and pedestrians in the input image are recognized, these objects are tracked to get their location and speed information. The objects are tracked from different frames of the input video and their location and speed information are determined. With the help of the results obtained, this research claims that the use of keypoint feature information extracted from 3D object models helps estimate the pose of objects with improved confidence, when compared to methods like Tangruamsub et al. [1] which use keypoint feature information extracted from 2D images or when compared to other deep learning techniques [28,84] for object recognition in self-driving vehicles.

Chapter 3: Proposed System

This chapter discusses the proposed system developed for the recognition and pose estimation of dynamic objects on the road, followed by the tracking of recognized objects, in the context of self-driving vehicles. The chapter contains the algorithm used to identify the dynamic objects with their pose and confidence scores. Once the dynamic objects are recognized, the proposed system tracks these objects to get their location and speed information. Additionally, the chapter discusses the relationship and contributions of this thesis to the other components of an overall system which was developed to tackle the various problems that are prevalent in self-driving vehicles today.

3.1 Motivation

There has been a lot of research taking place in the field of autonomous driving these days and yet a fully automated vehicle is still a dream. Additionally, the semi-autonomous cars that are available on the market, have recently been involved in pedestrian fatalities. For example, the two recent deaths involving Uber and Tesla vehicles using driverless systems have raised the debate on safety to such a level that it threatens to significantly delay or derail the adoption of the technology. It is clear that still there is a lot of room for improvement in the technologies used for autonomous vehicles. This system proposes a new approach to improve the object recognition and pose estimation techniques, which is discussed in detail in the following sections.

3.2 Relation of the thesis to other components of the overall system

This research work is related to the overall system shown in Figure 30. The relation and contributions of this thesis to the overall system is discussed below.

There are basically six different modules for this system which are interconnected with each other.

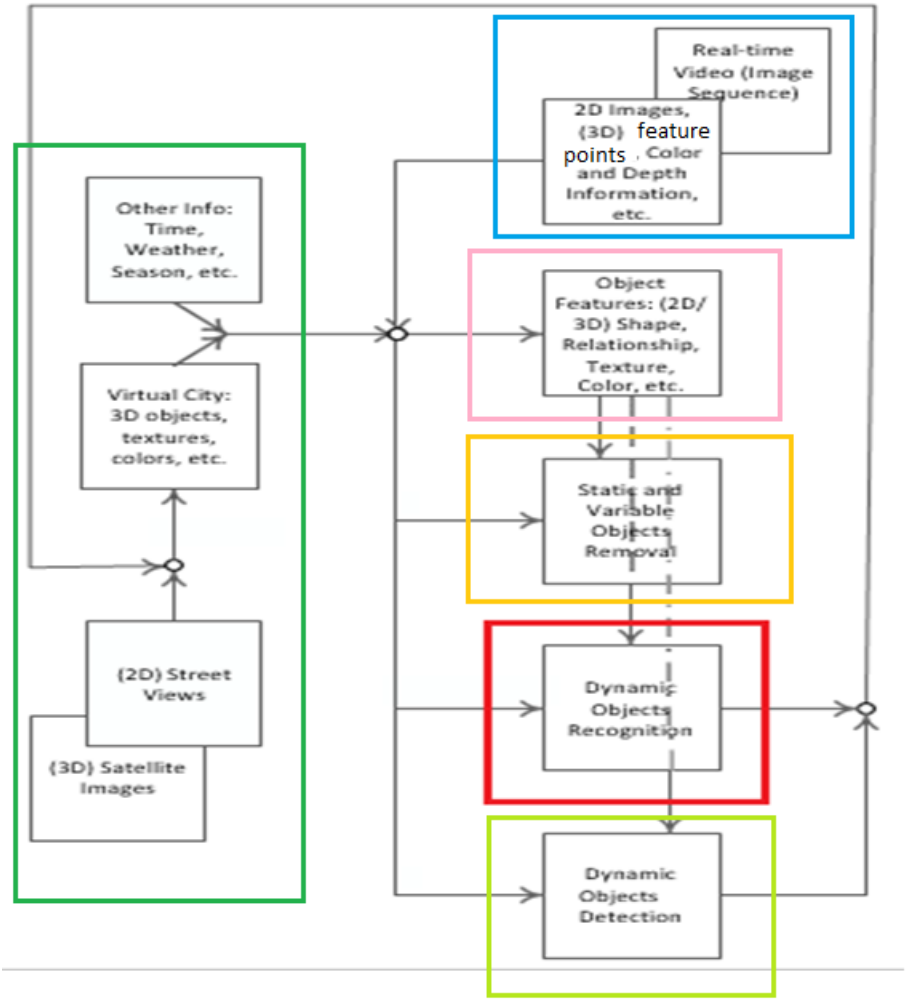

Figure 30: Flowchart of the overall system

The overall system primarily deals with the creation of a virtual city from open source/cloud VGI data such as 2D street views and satellite images. This virtual city contains 3D models of static objects like buildings and variable objects like trees. Apart from this, a repository containing the 3D models of dynamic objects (such as cars) is created separately. The module marked in blue in Figure 30 shows the real-time video (image sequence) passed as input into the overall system. The keypoint features of the input image are identified along with the orientation details. The module marked in yellow in Figure 30 is the static and variable object elimination module. In this module, the identified keypoint features of the static objects like buildings or variable objects like trees in the input image are matched with the 3D models of these static objects present in the virtual city. The matching of the keypoint features of static objects in the input image with the keypoints of the 3D models of the static objects in the virtual city verifies the presence of the static or variable object at a location. Once verification is done, the static and variable objects are removed and only the dynamic objects are retained. This reduces the processing time as only the recognition of the objects which would impact the navigation of the car need to be taken care of. The module marked in red in Figure 30 is the area dealt with by this thesis. This module handles the object recognition and pose estimation of the dynamic objects present in the input image. This module additionally tracks the recognized object from multiple frames of the input video. The recognized object with its pose information along with the objects speed and location details are used to update the virtual city with the identified dynamic objects in real-time. The module marked in light green in Figure 30 updates the simulation of the dynamic objects in to the virtual city and uses this information along with IoT (internet of things) to determine the navigation of the self-driving vehicle.

3.2.1 Modules of the overall system directly related to the thesis research

The modules of the overall system which are directly related to this thesis are mainly the virtual city module (marked in green), the extraction of object features module (marked in pink) and the dynamic object detection module (marked in light green). The virtual city module and the extraction of object features module act as input into the proposed system. The dynamic object detection module then uses the output from the proposed system for determining the navigation decisions of the self-driving vehicle. The three modules which signify the contribution of this thesis to the overall system are explained in detailed below.

1. **Creation of the Virtual City:** A virtual city is first created from open source VGI data such as 2D street views and satellite images. 3D structural files are extracted with 3D structures of the buildings. These are rendered, and the final 3D structure is obtained, with the geolocation information externally mapped on to the model. By extracting real-world images and georeferencing them, the textures are mapped onto buildings in the 3D model. In this way, a virtual city with static objects like buildings and variable objects like trees is created. Later this virtual city is updated with dynamic objects in real-time. The virtual city with 3D static, variable and dynamic object model information present in real-time road scenes is used by the autonomous navigation system to determine the correct navigation decisions for the self-driving car. This module is marked using a dark green rectangle box in Figure 30.
2. **Key point Extraction and Dataset Creation:** As discussed earlier, in the scope of this thesis only cars and humans are considered in the category of dynamic objects. In the case of cars, the 3D object models stored in the repository are rendered as mentioned in Section 2.4.1. KeypointNet [23] is used to extract the keypoint features from different rendered

views of 3D car models. The coordinate information of the identified keypoints of the rendered image, orientation information details of each keypoint and the direction that the car is moving in (Left, Right, Towards, Away) are stored in annotation files. In the case of humans, the DensePose [40] model is adopted and integrated into the overall system for human pose estimation. The DensePose [40] model has its own manually-collected ground truth dataset called the COCO-DensePose dataset which annotates dense correspondence between the image and a 3D surface model by asking the annotators to segment the image into semantic regions and to then localize the corresponding surface point for each of the sampled points on any of the rendered part images. The surface coordinates of the rendered views localize the collected 2D points on the 3D model.

The dynamic object recognition module uses this repository for matching the keypoint features of the dynamic objects in the input image with the keypoint feature information of the 3D object models stored in the repository, and a suitable 3D model corresponding to the object in the input image is retrieved. This module is marked using a pink rectangle box in Figure 30.

3. **Dynamic Object detection using IoT:** This module uses the information from the dynamic object recognition module to update the virtual city with dynamic objects on the road in real-time. The recognized object with its pose information along with the speed and location of the objects are used to update the virtual city with the identified dynamic objects in real-time. Prior knowledge about the dynamic, static and variable objects from the virtual city and IoT is then used to determine the appropriate navigation decision of the self-driving car. This module is marked using a light green rectangle box in Fig 30.

3.3 Proposed Techniques for Dynamic Object Recognition and Pose Estimation

The dynamic object recognition and pose estimation module is the most crucial module in determining the navigation of an autonomous car. With deep learning techniques being the state-of-the-art in object recognition and pose estimation, the aim of this thesis is to create an algorithm which will further improve the confidence with which the system can confirm the presence of an object at a given location in an input frame. As discussed in chapter 2, the 3D object models contain additional depth information which helps the system recognize the 3D structure of objects in an input frame. This idea is exploited in this research, and a repository of rendered images of 3D object models is used to match the objects in the input image and find the best suitable 3D model from the repository for the each of the dynamic objects present in the input image. The use of 3D object models is anticipated to improve the confidence of object recognition and the information from the 3D models will provide the overall system with crucial information to help determine the navigation decisions of the self-driving vehicle in real-time. This thesis deals only with cars and pedestrians in the category of dynamic objects. However, in future this work can be extended to other objects like bicycles, buses, animals, and birds.

The proposed technique developed for both cars and pedestrians, makes use of the datasets created by identifying the keypoints on 3D object models. The most suitable matching 3D model for each of the dynamic objects in the input image is then retrieved using the keypoint information stored in the repository or dataset.

For cars, a repository is created with keypoint information for different rendered views of 3D car models (from the ShapeNet dataset [60]). However, due to the complex structure of humans, the labour-intensive task of manually creating such a repository is beyond the scope of this thesis work. Hence the DensePose [40] model which has its own manually-collected ground truth dataset

called the COCO-DensePose dataset has been adopted and integrated into our system for 3D pose estimation of pedestrians on the road. The sections below discuss the approach adopted in the case of cars and humans in detail.

3.3.1 Object Recognition and Pose Estimation of Cars

As mentioned earlier, a repository is initially created with rendered images of 3D car models in different orientations and views. For each of these rendered images, the keypoint coordinate information and the direction in which the vehicle is moving is stored in separate annotation files. Since the work of the overall system is still in progress, in the efforts to test the proposed algorithm, a small repository with the above-mentioned information was created as shown in Figure 31 below.

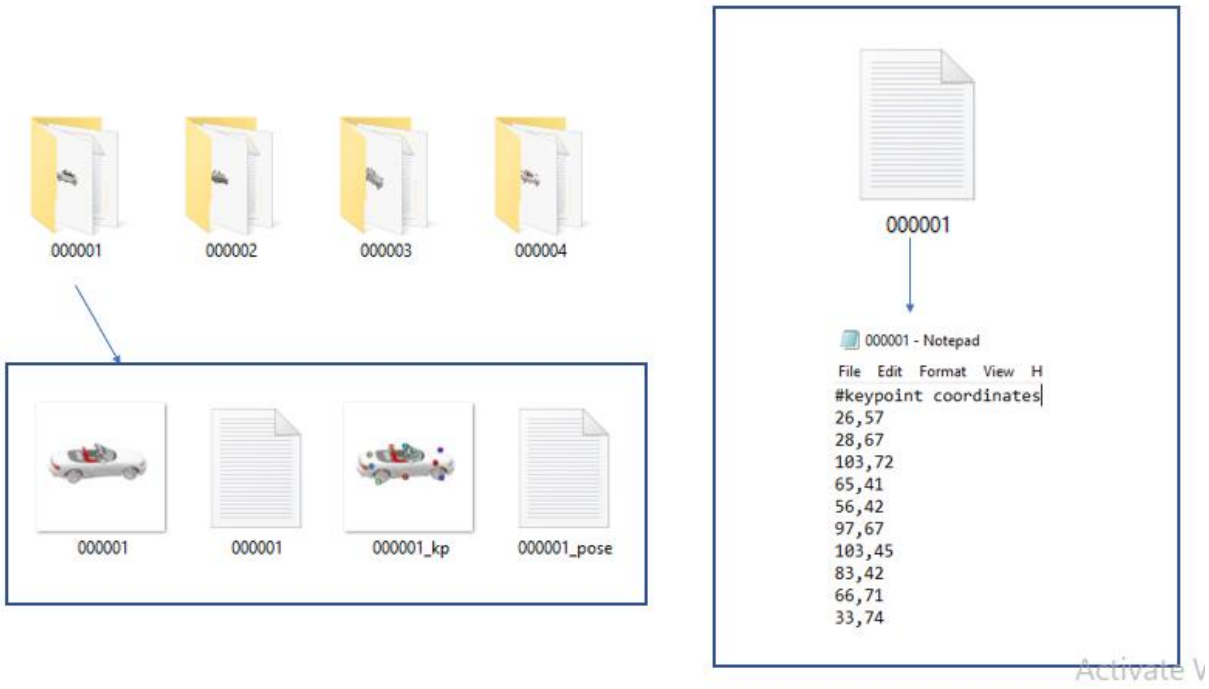

Figure 31: Snippet of the repository

The repository contains one folder for each of the rendered images. Rendered images in different orientations and views were considered to make the repository rich enough to test the prototype. Each folder contains four files as shown in Figure 31.

The folder contains a file which is the rendered image of the 3D car model in a particular orientation and view. Another file containing the rendered car image with the keypoint features plotted on it is also present in the folder. Apart from these, the folder contains two annotation files. One of the annotation files contains details of the (u,v) coordinates of the keypoint features. The other annotation file contains the direction in which the car is moving in the rendered image. Currently, four directions are being considered by the system: Left, Right, Towards, Away. This labeling is used by the algorithm to predict the direction in which the car is moving in the real-time image.

The repository with the keypoint information of the rendered images of 3D car models in different possible views serves as the dataset for finding a suitable matching 3D model along with the pose information for each of the cars identified in the real-time input image. The new approach developed for this purpose is discussed below.

3.3.1.1 The New Approach

A voting algorithm has been developed which is inspired by the approach proposed by Tangruamsub et al. [1]. Consider a real-time input image with a single car on the road. The single car scenario is used for explaining the algorithm and the special scenarios (occlusion and a car partly invisible due to some obstacle) which are discussed in the later sections. The keypoints are identified on the real-time input image using the KeypointNet model [23]. First, the algorithm

compares the keypoints of the input image with the set of keypoints belonging to each rendered image in the repository and find the best match among them. Once the matched model is found, the rest of the unmatched feature points are back-projected on to the input image. For each of these keypoints the object centre candidates are found and the ones at a distance greater than the pre-defined threshold from the center of the image are removed. To make sure that all keypoints are part of a single object in the image, a clustering technique has been developed to group the object centre candidates. The idea is that, if all the object centre candidates are close to each other then there is a high chance that the object exists. This idea is again inspired by the work of Tangruamsub et al. [1], where they used a similar approach to cluster the feature points of 2D images. This is well depicted in Figure 32 below [1].

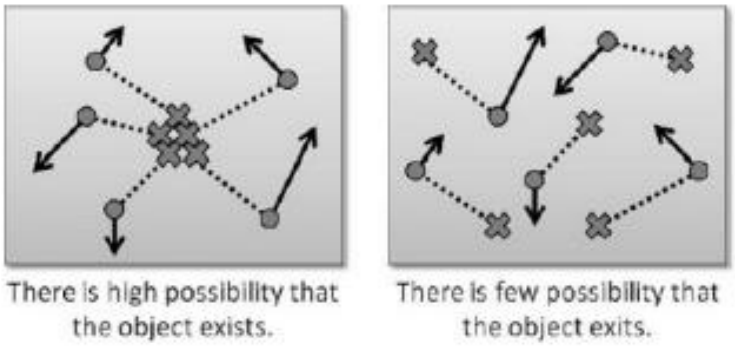

Figure 32: Object center candidates

The number of clusters and the keypoints within each cluster is then used to assign a confidence score value which gives information about the presence of the number and pose of objects at that location in the frame. The definition of the term confidence score and the method used in this research to calculate it is discussed in detail below.

3.3.1.2 Confidence Score

In the context of machine learning, the term "confidence" is simply the probability of some event. If an event has high probability, it means it has high confidence [82,83]. In the same way, the term "confidence score", is used in the scope of this research, as a measure of the probability of the object being at a given location in the real-time input image.

In the work by Tangruamsub et al. [1], in order to determine the presence and position of an object in an input image, the authors clustered the object centre candidates computed from matching keypoints and used the number of cluster elements as a confidence measure. This technique of using the number of cluster elements to determine the confidence score was adopted by this research. However, in this work, the confidence score is calculated as a probability value between 0.0 and 1.0, with 1.0 representing the highest confidence value and 0.0 representing the lowest confidence value. The proposed algorithm first checks if all keypoints which matched with the template model fall into a single cluster. This signifies that all keypoints are close to each other and the input image contains only single object. If all keypoints match, then the confidence of the object being present at the location is high and the confidence score is calculated as:

$$\text{confidence score} = \frac{\text{Number of matched keypoints}}{\text{Number of matched keypoints in the cluster}} = 1.0$$

If the cluster does not contain all keypoints that matched with the template model, then the confidence score is calculated as:

$$\text{confidence score} = \frac{\text{Number of matched keypoint features}}{\text{Total number of keypoint features in the cluster}}$$

This gives the confidence or the probability of the object being at that location.

In special cases like occluded objects in a single image, more than one cluster will be formed as some of the object centre candidates corresponding to the different keypoints would be far from each other. In such cases, if any of the clusters have very few object centre candidates corresponding to the keypoints (<=4 keypoints), then the confidence of the object being present at that location must be very low. Hence, in such cases, the formula below is used to calculate the confidence score:

$$\text{confidence score} = \frac{\text{Number of matched keypoint features in the cluster}}{\text{Total number of matched keypoint features}}$$

This is necessary because, the number of features in the cluster is very low, even for a small number of matched features the probability would be high, which would not give an accurate measure of the presence of the object at that location. However, considering the entire set of identified keypoints, would give the confidence score of the object being at that location.

The proposed voting algorithm explained in section 3.5.2 below, clearly defines how the confidence score is calculated from the number of matched keypoints in the different cases.

The experiments and results of the different scenarios discussed in Chapter 4, will help with understanding the details of the algorithm.

3.3.2 Object Recognition and Pose Estimation of Humans

In the case of pedestrians, the pose estimation is challenging due to the higher complexity and flexibility of the human body as well as the larger variation in poses. Building the 3D human model repository is a labour-intensive task which is beyond the scope of this thesis work. Therefore, the DensePose [40] deep learning network developed by the Facebook research team is adopted and

integrated into the proposed system of this research for pedestrian pose estimation. DensePose [40] establishes dense correspondences between pedestrians in the input image and a 3D surface-based representation of the human body. The DensePose [40] system, primarily created a manually-collected ground truth dataset called the COCO-DensePose dataset by gathering dense correspondences between the SMPL model [41] and the persons appearing in the COCO dataset [63]. The DensePose [40] system accomplished the task of creating the COCO-DensePose dataset through a novel annotation pipeline which exploits 3D surface information during annotation. In the first stage of the annotation pipeline, the annotators were asked to segment the image into semantic regions of the human body like the head, torso, lower/upper arms, lower/upper legs, hands, and feet, and to then localize the corresponding surface point for each of the sampled points on any of the rendered part images. In the second stage, the surface coordinates of the rendered views localize the collected 2D points on the 3D model. The DensePose [40] system gathered annotations for 50K humans, collecting more than 5 million manually annotated correspondences. The COCO-DensePose dataset is then used to train a deep network that predicts dense correspondences between the input image pixels and surface points. The architectures of the DenseReg [65] approach and the Mask-R CNN system [66] were combined to develop the DensePose-RCNN system. The DensePose-RCNN is a fully convolutional network that combines classification and regression tasks. In the first step, the pixel is classified as belonging to either the background or one of several region parts which provide a coarse estimate of the surface coordinates. In the second step, a regression system indicates the exact coordinates of the pixel within the part. Cascaded extensions of DensePose-RCNN were developed to further improve the accuracy of the system. The DensePose [40] system successfully estimates human body pose while handling a large variability of scales, poses, and occlusion. The DensePose [40] model is integrated

into the proposed system which delivers dense correspondence by regressing body surface coordinates at any image pixel of pedestrians in the input image.

3.4 Proposed Technique for Dynamic Object Tracking

After the task of object recognition and pose estimation, the dynamic objects are tracked using the real-time input video. The speed of the moving vehicles and pedestrians is determined and passed to the "Dynamic object detection using IoT" module (Refer Section 3.2). This additional information along with the updated 3D model from the virtual city can be used to ensure safe driving. Object tracking in video is a classic computer vision problem. It consists not only of detecting the object in a scene, but also of recognizing the object in each frame. There are various computer vision algorithms which perform object tracking. Traditionally, speed detection was achieved by directly measuring the distance from the camera to the vehicle in order to calibrate the speed of the vehicle across the frames. For speed cameras specifically, Doppler radar technology is used to measure the change of microwaves reflected from a vehicle in order to obtain the speed at which it is moving. Here, the input video stream is used to detect the speed of the vehicles and pedestrians across different frames of the video. In video surveillance, normally some features of the moving objects are extracted using which the object is tracked. The approach developed by Kagita et al. [69] uses the Haar Cascade Classifier to identify the vehicles/vehicle boundaries. The Haar cascade classifier can be created for any object with the OpenCV python library [68]. To build the Haar cascade positive (images containing the object to be tracked) and negative (images without the object to be tracked) images are needed. The OpenCV library has the function opencv_createsamples which can be used to create a set of positive samples. Once the negative and positive sample images are collected, two description files need to be created. The

file named bg.txt contains the information for the negative samples. The function opencv_createsamples, create the positive samples along with the description file named info.lst which contains the following information: image filename, the number of objects in the image, and their locations. The location information contains the x and y coordinates, the width and the height of the rectangular bounding box for the object within the image. The function opencv_createsamples uses the file named info.lst to create the vector file, by passing the location of the info.lst, the number of images to be present in the file, and the dimensions of the image to be kept in this file. At this point, the Haar classifier is trained using the files mentioned previously: bg.txt, info.lst and vector file. The command “opencv_traincascade” is used to train the classifier which uses the above files and the number of stages required to train as input parameters. After training the file cascade.xml is created, which can now be used to track object of interest in any new video. For most common objects like cars, buses, bikes or pedestrians the trained .xml files are easily available. Therefore, object tracking can be easily done with the help of the OpenCV python library.

Once a vehicle and its location information are detected in a video, the next task is to find the speed at which the vehicle is moving by considering the different frames of the video sequence. The midpoints of the bounding boxes surrounding the vehicle are obtained and used as the "position" of the vehicle which will be compared to the position in future frames. The distance a vehicle has travelled in the image plane would be the relative difference, measured in pixels, at various vehicle positions in subsequent frames. The time is calculated as [69]:

Time = Number of pixels progressed by vehicle/frames per second

Distance is calculated as [69]:

Distance = sqrt(mid1*mid1+mid2*mid2), where mid1 and mid2 are the positions of the vehicle in frame one and two respectively.

If the difference is less than 10 pixels, then the car is assumed to be stationary and is neglected when updating values. To generalize the direction in which the vehicle is progressing subjecting to change in pixel values, Manhattan distance is being calculated between the centroids of the vehicles in subsequent frames and the resulting pixel value is converted from the image plane to the object plane. In this research the same approach has been implemented for pedestrians, by using the file cascade.xml corresponding to pedestrians for the detection of pedestrians followed by their speed estimation. Once the speed of the vehicle in the image plane is calculated, the pixel values obtained can then be converted into measures representative of object plane, using the distance of the object from the camera attached to the vehicle using the formula [69]: exact distance to object (mm) = focal length (mm) * real height of the object (mm) * image height (pixels)/object height (pixels) * sensor height (mm). This would help in determining the relative distance between the moving car and the moving object on the road and hence calculate the relative speed. Also in autonomous driving scenarios, precise 3D localization and trajectory estimation is of fundamental importance. In order to prevent collisions, it is crucial to be aware of the extent and the orientation of objects in world space, especially for objects close to the camera. Osep et al., [92] proposes a tracking framework which exploits both 2D and 3D measurements for object tracking in self-driving vehicles. Use of 3D bounding boxes assist in locating the objects in world coordinates. Osep et al., [92] introduced a novel 2D-3D Kalman filter, which is keeping both an image- and a world-space (position and size) estimate. These estimates are loosely coupled to ensure the consistency of a track. This coupling enables us to track distant objects and continue these tracks with more precise information in the close range, while smoothly transitioning between the

modalities. The use of 3D bounding boxes for object tracking would be useful to get the measurements in world coordinates and can be considered as a future scope to this thesis work.

The results for object tracking using the proposed method are shown in Chapter 4.

3.5 Design of the Flowchart and Algorithms

The flowchart representing the complete proposed technique is depicted in Figure 33 below. The real-time image obtained from the cameras of the self-driving vehicle is the input to the system. The 3D models corresponding to the dynamic objects in the input image along with the pose information obtained from the u, v coordinates of the keypoints of the objects, is the output of the system. Additionally, the recognized objects are tracked using multiple frames of the input video (sequence of images) to obtain the speed and location of the objects. The object tracking technique is depicted in a separate flowchart in Figure 34.

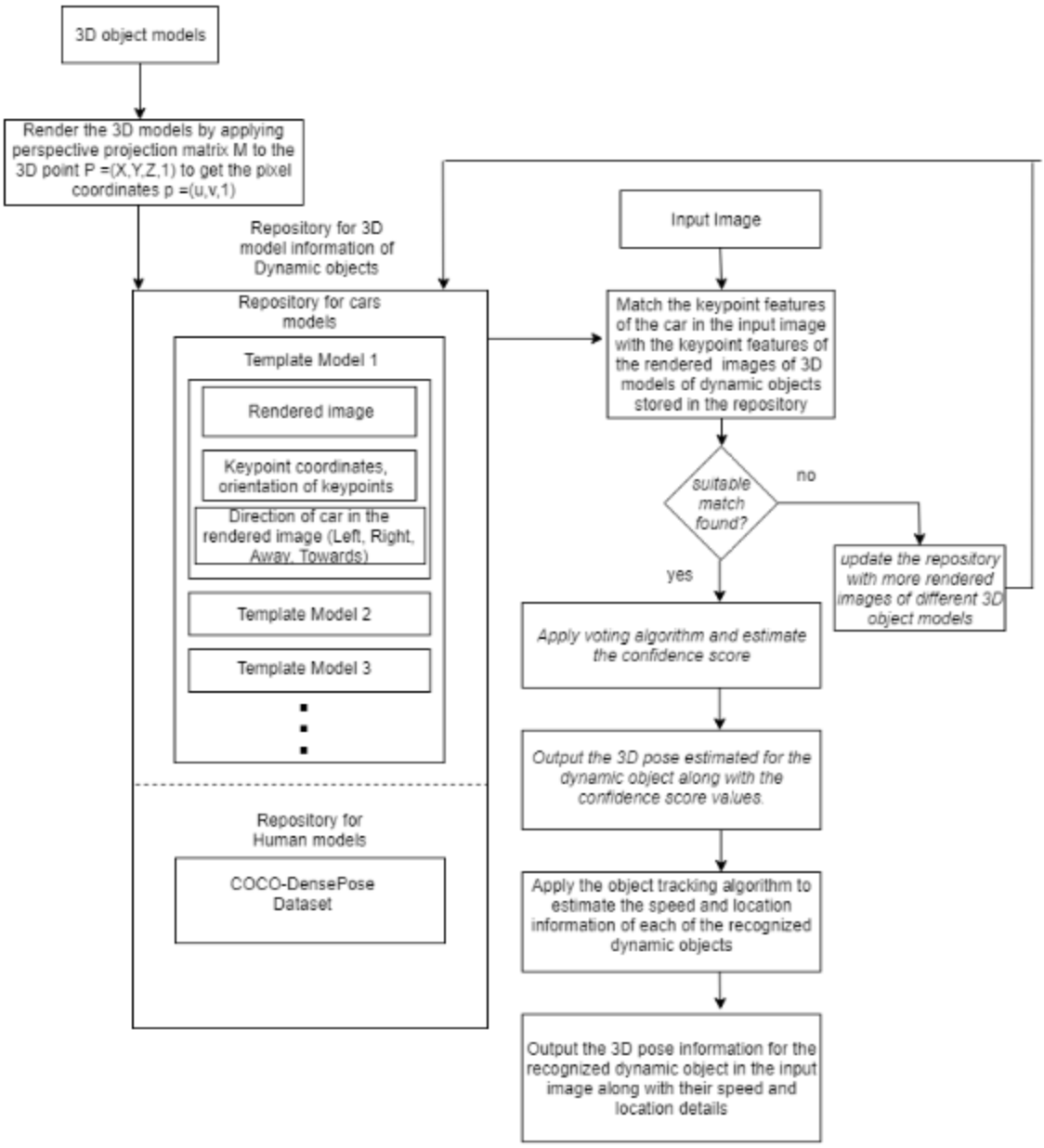

Figure 33: Flowchart representing the proposed technique for object recognition and pose estimation of dynamic objects

The proposed system has an additional object tracking module to track the speed and location of the dynamic objects in the input video sequence. The flowchart in Figure 35, depicts the proposed technique for tracking cars and pedestrians from the input video (sequence of image frames).

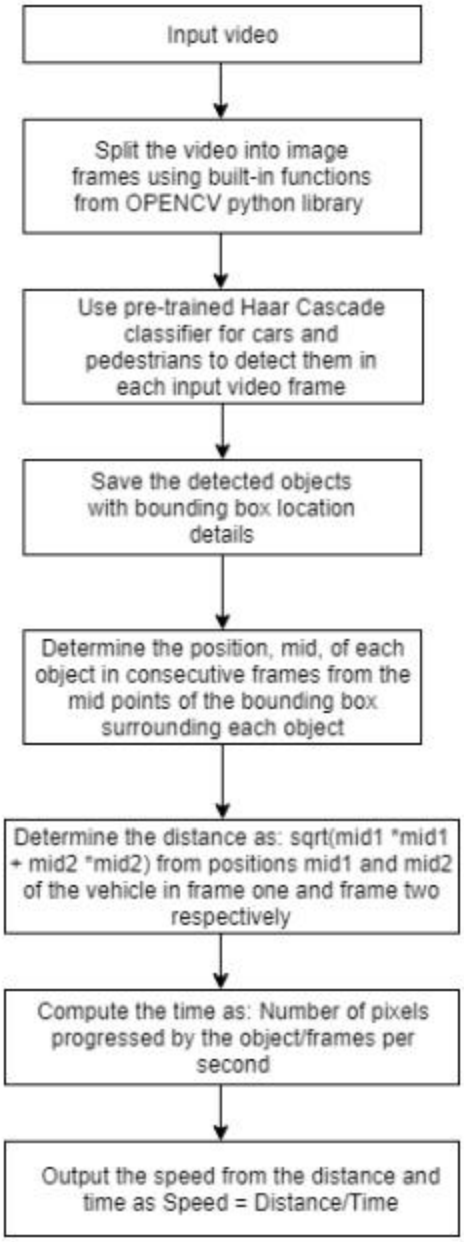

Figure 34: Flowchart depicting the proposed technique for object tracking from input video

3.5.1 The Proposed Algorithm for Recognition, Pose Estimation and Tracking of Dynamic Objects in the Real-Time Input Image

The step-wise algorithm of the proposed technique for the recognition and pose estimation of dynamic objects followed by tracking of the recognized objects from the real-time input image is discussed below.

Algorithm 1: The proposed technique for recognition and pose estimation of dynamic objects followed by tracking of the recognized objects from the real-time input image

INPUT: The real-time input image captured by the camera sensor of the self-driving car
OUTPUT: The 3D pose information along with the speed and location details of all dynamic objects present in the input image.

Step 1: Identify the keypoint features from the real-time input image.

Step 2: If the object is a car, do steps 3 to 7, else if the object is a human, do steps 8 to 10

Step 3: Match the keypoint features of the car in the input image with the keypoint features of the rendered images of 3D car models stored in the repository.

Step 4: If a suitable matching car model is found, do step 6 and step 7, else do step 5.

Step 5: Update the repository with more rendered images of different 3D car models.

Step 6: Apply the proposed voting algorithm and estimate the confidence score value.

Step 7: Output the confidence score along with the direction of car movement obtained from the annotation file of the matched template model for each of the cars recognized in the input image.

Step 8: Apply the DensePose-RCNN model pre-trained with the COCO-DensePose dataset and retrieve the UV coordinates of the pixels within each body part of the pedestrians in the input image.

Step 9: Visualize the isocontours of the UV fields on the pedestrians in the input image.

Step 10: Map the textures on to the isocontours of the UV fields and visualize the 3D human pose estimated from these UV coordinates.

Step 11: Track each of the recognized dynamic objects (cars and pedestrians) from the input video to determine their location and speed.

Step 12: End

The proposed voting algorithm to estimate the confidence score value of the recognized cars in the input image is discussed in detail in section 3.5.2 and the proposed algorithm for the tracking of the dynamic objects to determine their location and speed is discussed in detail in section 3.5.3.

3.5.2 The Proposed Voting Algorithm for Cars

The Voting and Clustering techniques are explained in detail below. In the work by Tangruamsub et al. [1], the authors developed a similar voting algorithm, using 2D images. This approach by Tangruamsub et al. [1] is compared with the proposed method and in order to prove that the use of 3D models gives better results.

Algorithm 2: The voting technique for determining the confidence scores of objects in the real time image

INPUT: The repository with rendered images and their corresponding keypoint coordinate information, orientation details and the direction of the car.
OUTPUT: The confidence scores for the object in the image along with its direction of movement.

Step 1: Each keypoint of the input image is matched with the keypoints of the models stored in the repository by calculating the distance between the j^{th} keypoint feature of the input image to the i^{th} key point feature of the template model as:

$j1NN = \arg\min j\ dij$ [1]

dij - is the distance between *qj and pi*

qj - the *j*-th keypoint feature of the input image

pi - the *i*-th keypoint feature of a template model.

This distance is compared to all keypoint features of the template models to find the best template model match. If the distance exceeds the predefined threshold in all matches, it is concluded that *qj* does not match any template model. Repeat for all keypoints in the input image and the template model, T, with maximum matched keypoints is selected as the best match. If none of the keypoints of the input image match with any of the template models, go to Step 11.

Step 2: Save the matched key points and unmatched keypoints of T in separate lists.

Step 4: Back project the unmatched keypoints of the T on to the real time car image.

Step 5: Find the centre of the input image as (C_x, C_y).

Step 6: Find the distance of all the keypoints to the center of the real time image and remove all the points which are farther than a specific threshold.

Step 7: Find the object centre candidates from the matched features using the formula below:

$$\begin{cases} X = x_{in} + \dfrac{\sigma_{in}}{\sigma_{temp}} \times \sqrt{\Delta x^2 + \Delta y^2} \times \cos\left(\theta + \theta_{temp} - \theta_{in}\right) \\ Y = y_{in} - \dfrac{\sigma_{in}}{\sigma_{temp}} \times \sqrt{\Delta x^2 + \Delta y^2} \times \sin\left(\theta + \theta_{temp} - \theta_{in}\right) \end{cases} \quad [1]$$

σ_{temp} – scale of a keypoint

θ_{temp} - orientation of a keypoint of a matched template model

σ_{in} – scale of a keypoint of an input image

θ_{in} – orientation of a keypoint of an input image

Δx, Δy – distance vector from the keypoint to the object centre

Step 8: Apply the clustering technique on these object center candidates to determine the confidence score of the car being at the location in the specified direction.

Step 9: If the clusters are determined and confidence scores predicted. Go to Step 10.

Step 10: Insert a bounding box over the image and display the confidence score, along with the direction of the car as determined from the annotation file of the matched template model.

Step 11: Add more models to the repository until the repository is rich enough to handle any input car image in any orientation.

Step 12: End

The Clustering Technique

The voting algorithm explained above uses the following clustering technique in order to find the cluster into which an object center candidate belongs. This algorithm helps to determine if there is more than one prominent cluster in an image. This is useful for determining the presence of more than one car in the real-time input image and for other occlusion scenarios.

Algorithm 3: Clustering technique for grouping the object center candidates

INPUT: The set of object centre candidates O and cluster list C.
OUTPUT: The confidence scores for different cluster determined by the algorithm

Step 1: Let O be the set of all object centre candidates and n be the total number of elements in O. Assign any random element from set O (here the first element in the list is chosen) to the first cluster C_1.

Step 2: Set an intra-cluster distance d.

Step 3: For all each of the remaining elements of O, repeat step 4 to step 6.

Step 4: For all the clusters currently in list C, do step 5.

Step 5: Compute the distance of O_i to every element present in the cluster C_k. The distance formula used:

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

where Oi = (x1, y1) and Oj = (x2, y2).

If the distance of between Oi and any other element of C_k exceeds the threshold d, do Step 6, otherwise assign O_i to the same cluster, and go back to step 3.

Step 6: If O_i, doesn't fit into any cluster, assign it to a new cluster and add it to end of the list C.

Step 7: Once all the elements O_i of O are assigned to different clusters, find the number of clusters formed.

Step 8: If the number of clusters = 1, go to step 9 else go to step 12.

Step 9: Apart from the object center candidates corresponding to the back projected keypoints, check if, the cluster contains object center candidates which correspond to the keypoints in the matched set of feature points, M, that was saved in Step 2 of the voting algorithm. If yes, do Step 10, else do Step 11.

Step 10: Assign confidence_score = 1.0

Step 11: Assign confidence_score = $\frac{\text{Number of matched keypoint features}}{\text{Total number of keypoint features in the cluster}}$

Step 12: If number of clusters > 1, for each cluster, repeat step 13 and 14

Step 13: Check the number of object centre candidates corresponding to the matched set M in the cluster. Let this count be denoted as 'm_i'.

Step 14: Check the number of points p within the cluster. If p > 1 and < =4, go to step 15, else go to step 16.

Step 15: Compare the cluster set with the matched set, and if none of the points belong to it, assign confidence_score = 0.0 and assign label = 'WARNING' instead of 'CAR', else set confidence_score as: confidence_score = $\frac{\text{Number of matched keypoint features(mi) in the cluster}}{\text{Total number of matched keypoint features}}$

Step 16: Compare the cluster set with the matched set, and if number of matched keypoints = 0, do step 17, else if number of matched keypoints > 4, do step 9 else do step 18.

Step 17: Assign confidence_score = 0.0 and assign label = 'WARNING' instead of 'CAR'.

Step 18: Assign confidence_score as:

confidence score = $\frac{\text{Number of matched keypoint features(mi) in the cluster}}{\text{Total number of matched keypoint features}}$

Step 19: End

Note that confidence scores are set as a value between 0.0 and 1.0, with 1.0 showing the highest confidence.

Once the different clusters have confidence scores assigned, the keypoint coordinates of each cluster is used to determine the bounding box for each object. The output displays the bounding box around the object with the confidence score and the direction that the car is moving in retrieved from the annotation file of the matched model.

3.5.3 The Proposed Object Tracking Algorithm

Algorithm 4: The proposed technique for tracking dynamic objects from the input video

---

INPUT: The real-time input video captured by the camera of the self-driving car
OUTPUT: The speed and location details of all the dynamic objects present in the input video.

Step 1: Split the input video into image frames using built-in functions from the OPENCV Python libraries.

Step 2: Use the pre-trained Haar Cascade classifier for cars and humans to detect the vehicles and pedestrians in each image frame.

Step 3: Save the detected objects with bounding box location details.

Step 4: Determine the position, mid, of each object in consecutive frames from the midpoints of the bounding box surrounding each object.

Step 5: Determine the distance as: sqrt (mid1 * mid1 + mid2 * mid2) from the positions mid1 and mid2 of the objects in frame one and frame two respectively.

Step 6: Determine the time as: Number of pixels progressed by the object/ frames per second.

Step 7: Output the speed computed from the distance and time as: Speed = Distance / Time.

Step 8: End

The main algorithm of the proposed technique in Section 3.5.1 and the sub-algorithm in Section 3.5.2 and Section 3.5.3, help in the 3D pose estimation of the dynamic objects in the input image and provides additional information of their location and speed by tracking these dynamic objects from the input video in real-time. The combination of this information from the proposed system is used by the overall system to update the dynamic objects on-to the virtual city which is later used by the autonomous navigation system of the self-driving vehicle to take appropriate navigation decisions.

3.6 Time Complexity of the Proposed Algorithm

The time complexity is calculated based on the programmatical implementation. The proposed technique starts with keypoint feature matching between the input images and the rendered images of 3D object models stored in the repository. Since each keypoint feature(i) of the input image is matched with the keypoint features (j) of all the models (n) stored in the repository, the time complexity for this task is calculated as O (i * j * n). The voting algorithm determines the object center candidates for all matched keypoint features (x) of the input image plus the unmatched keypoint features back-projected from the matched template model from the repository (y). Thus,

the time complexity for this task is calculated as O(n), where n = x + y. After applying the clustering technique, confidence scores are estimated for each cluster formed based on the number of keypoint features in each cluster. The time complexity for this task is calculated as O(m), where m is the number of clusters formed. Once the object is identified and the pose estimated, the recognized object is tracked to determine the location and speed of the object, using multiple frames of the input video. The time complexity for object tracking thus depends on the number of dynamic objects present in the video. The time complexity determining both the location and speed of the objects is therefore calculated as, O(n), where n is the number of dynamic objects present in the input video.

Module	***Time Complexity***	***Details***
Keypoint feature matching between the input object and the objects stored in the repository.	O (i * j * n)	i = number of keypoints identified on the input image j = number of keypoints identified on the rendered images of 3D models stored in the repository n = number of rendered images of 3D models stored in the repository
Voting Technique & Confidence Score estimation	Detection of the object centre candidates – O (n)	n = number of matched keypoints plus the back-projected keypoints from the matched template model from repository on the input object.
	Confidence score assignment – O(m)	m = number of clusters(objects) formed.
Object Tracking	Determination of location using bounding box – O(n)	n = number of dynamic objects in the input video
	Determination of speed – O(n)	n = number of dynamic objects in the input video

Table 2: Time complexity of the algorithm

Chapter 4: Implementation and Experiments

The proposed approach was implemented on Windows using the Python programming language. During the implementation phase of this research work, different Python and OpenCV libraries were used. The list of software and tools used are given below.

4.1 Software Information

The entire implementation of the thesis was done on an Alienware 1.5.0 x64-based Desktop, with NVIDIA version 8.1.940.0 and one Intel64 ~ 3192 Mhz GPU.

Item	***Details***
OS	Windows/ Linux
Languages	Python v3.7.1
IDE	Jupyter Notebook, Anaconda Prompt
Python libraries	OpenCV, Tensorflow, Pytorch, Scikit, CUDA 10.0, ImageAI
Tools	Blender, 3D viewer

Table 3: List of tools used for implementation of the proposed system

4.2 Experiments and Results of Object Recognition and Pose Estimation of Cars

The proposed algorithm has been implemented by creating a small repository of 3D car models from the ShapeNet dataset [60]. In the scope of the overall system, the repository of car models must be updated with 3D models of a variety of real time cars. However, since this module is still a work in progress, the proposed algorithm has been tested with a limited number of 3D car models from the ShapeNet dataset [60] which resembles real-time cars. Initially before attempting with real-time car images, some rendered images of the cars currently available in the repository were placed onto road backgrounds to create a real time feel and effect. This was done in order to check how well the algorithm would perform if the 3D car models in repository are very close to the real-time car images.

The algorithm was tested on different scenarios and it gave decent results. Each of these scenarios are discussed below with the results obtained.

4.2.1 Simulated input image with a single car

Consider the below input image with a single car on the road.

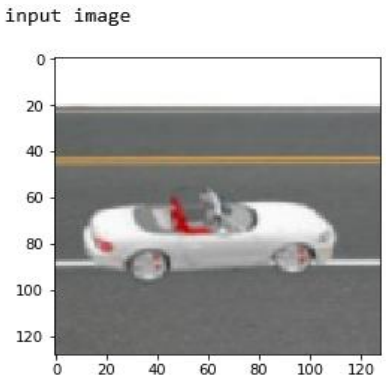

Figure 35: Input image

The KeypointNet model [23] identifies the keypoint features of the input image (Figure 35). It can be observed that few feature points are detected outside of the object on the image. The proposed voting algorithm removes the points which do not lie on the object by determining the object centre candidates and eliminating the points which lie outside a pre-defined threshold distance from the centre.

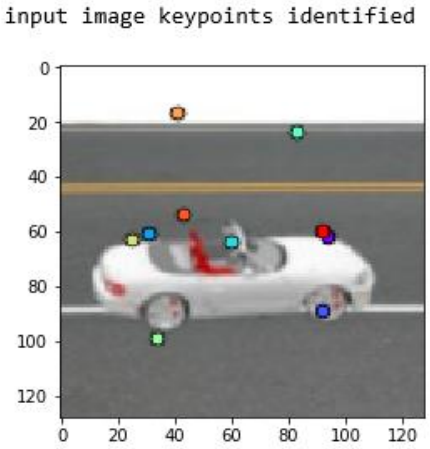

Figure 36: Keypoints detected using KeypointNet

When implementing the voting algorithm, as discussed above, each of the feature points are matched with the models in the repository in order to find the matching template model. Figure 37 shows the matching model from the repository after applying the algorithm.

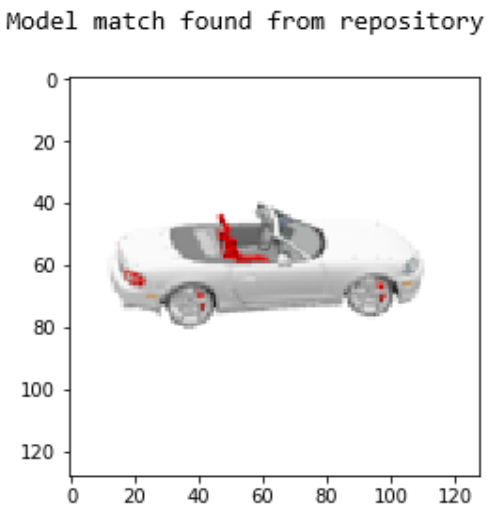

Figure 37: Matched model from the repository

Next, the image centre is determined which is then used to find the object center candidates, as discussed in the algorithm above.

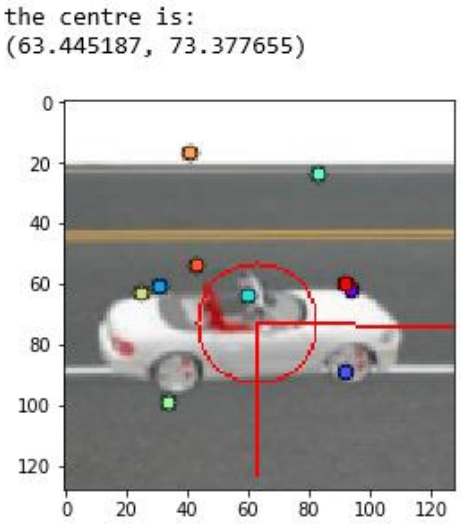

Figure 38: Image centre coordinates

As discussed in Step 6 and Step 7 of the voting algorithm, all the keypoints lying outside of the object are eliminated and the object center candidates are determined as shown in Figure 39 below.

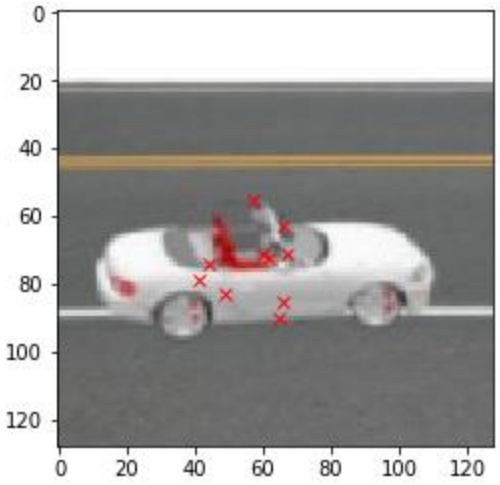

Figure 39: Object center candidates plotted on the image

The results in each step are shown in Figure 40 below.

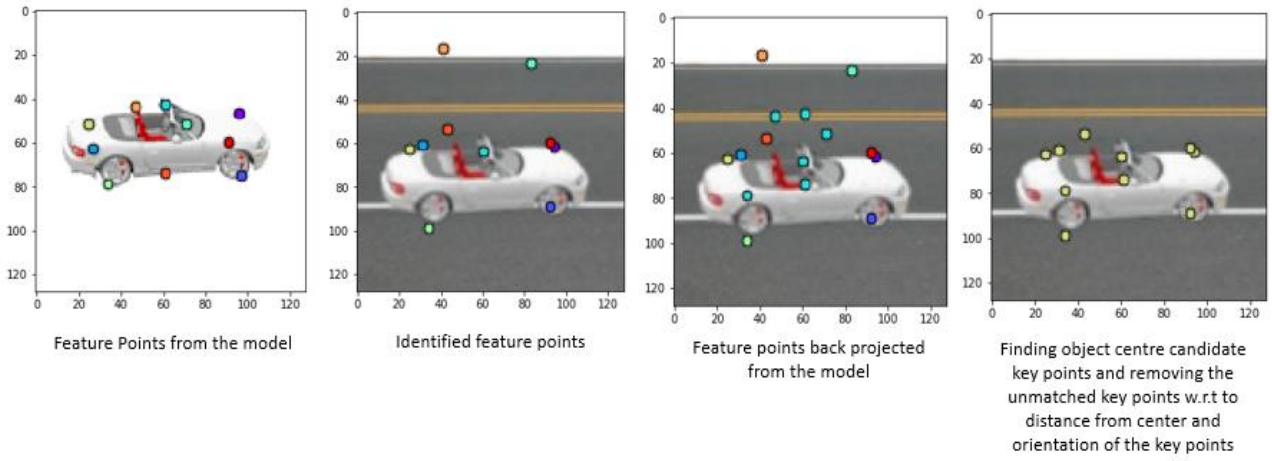

Figure 40: Representation of results in each step

Finally, the confidence score value is determined as discussed in the algorithm. In this example all keypoint features came under a single cluster and matched with the points on template model from repository. Therefore, a confidence score of 1.0 is assigned as explained in the algorithm. This signifies that the object is present at that location with a very high confidence level. This is

programmatically represented with a bounding box and the confidence score value printed on it as shown in Figure 41 below. The direction of the car is also determined from the annotation file of the matched template model from the repository.

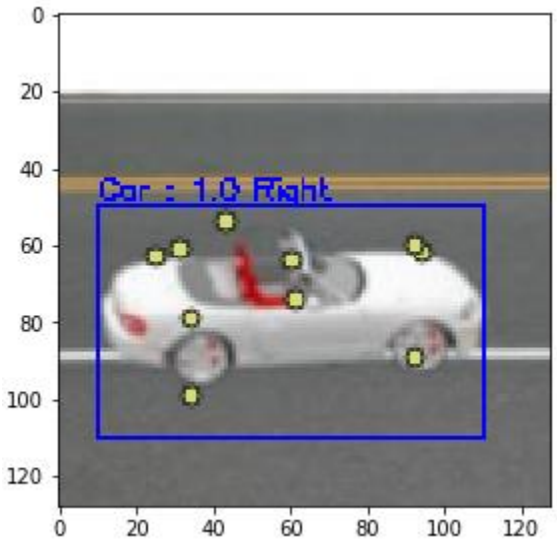

Figure 41: Car detected with confidence score = 1.0

Two more examples are shown below. The first one, shows the step-wise results of a single car moving in the “Left” direction and the second shows the step-wise results of a single car moving “Towards” the autonomous car.

Car moving Towards

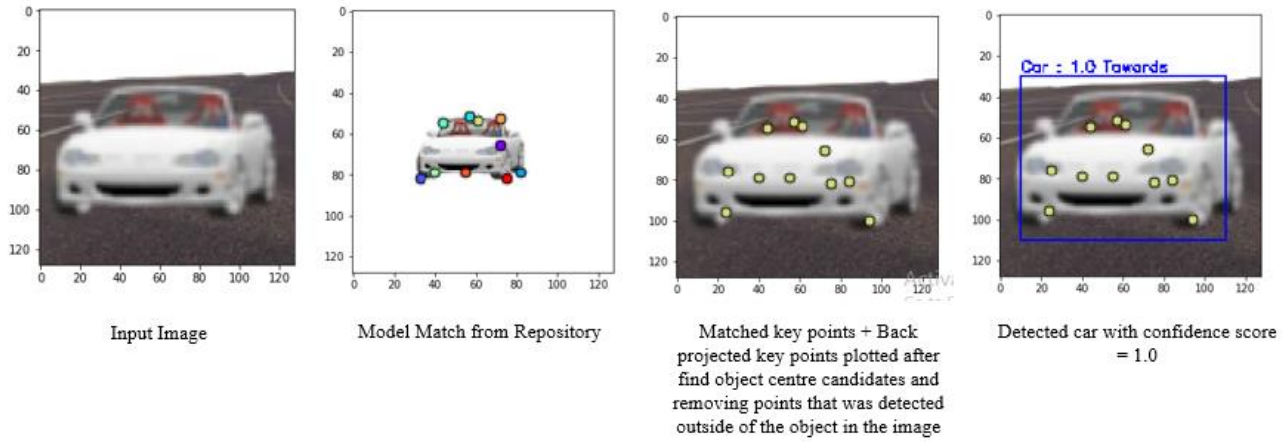

Figure 42: Step-wise results for the image of a car moving in the "Towards" direction

Car moving Left

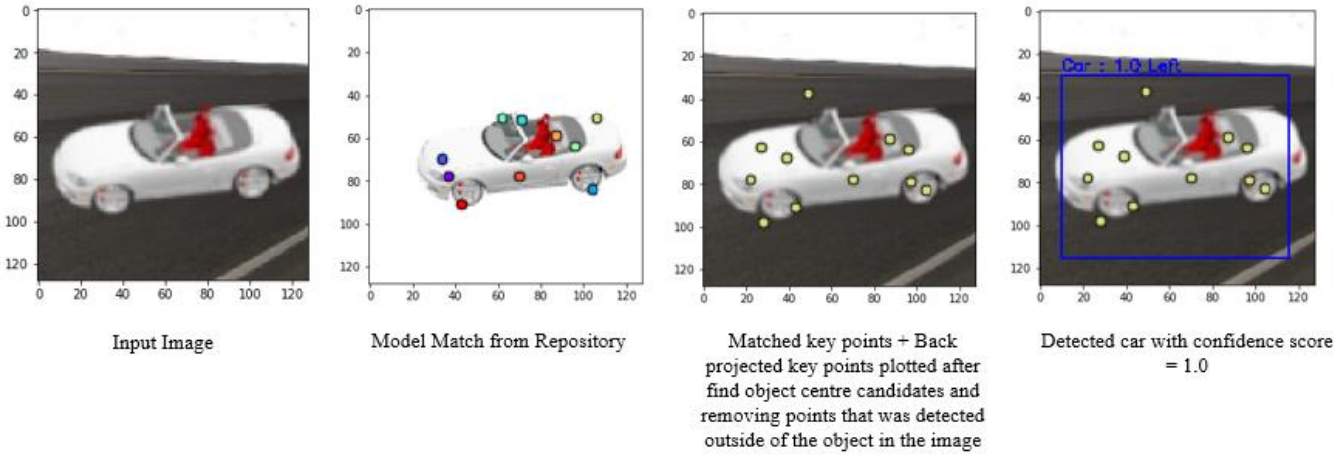

Figure 43: Step-wise results for the image of a car moving in the "Left" direction

4.2.2 Real-time input image with a single car

In case of the real-world road scenes, there would be multiple cars on the road. Therefore, it is required to initially identify the individual cars on the road and then crop them separately, as the pre-requisite for the KeypointNet model [23] is that the images be in 128 x 128 size for detecting the keypoints. The voting algorithm is then applied to determine the confidence score value with which the presence of the car can be confirmed at that location.

For this purpose, the python library, ImageAI (Olafenwa et al. [72]), is used that helps software developers easily integrate state-of-the-art computer vision technologies into new applications. The RetinaNet [72] model is used for object detection in real time. The comparison section discusses how the method improves the confidence of the presence of object at a given location.

While considering the real-time road scenes, the camera attached to the autonomous vehicle captures different frames. The Figure 44 below shows one such scene.

Figure 44: A real time road scene with a single car

The ImageAI python library along with the RetinaNet model [72], detects the object and crops it using the functions within the ImageAI python library as shown in Fig 45 below.

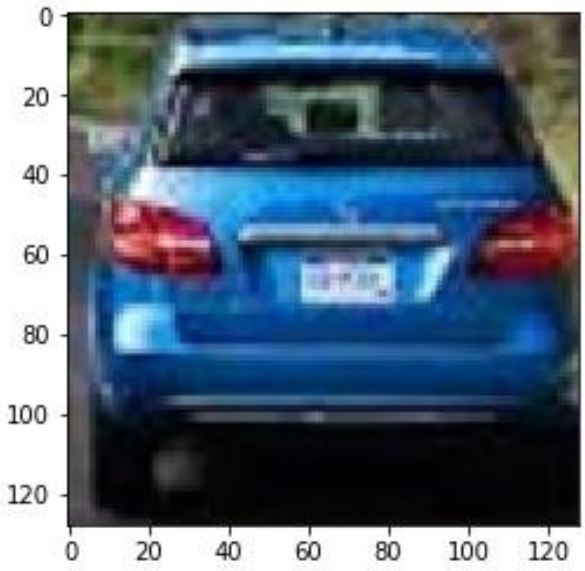

Figure 45: Cropping the detected car using ImageAI python library

The keypoints detected using KeypointNet [23] as shown in Figure 46 below.

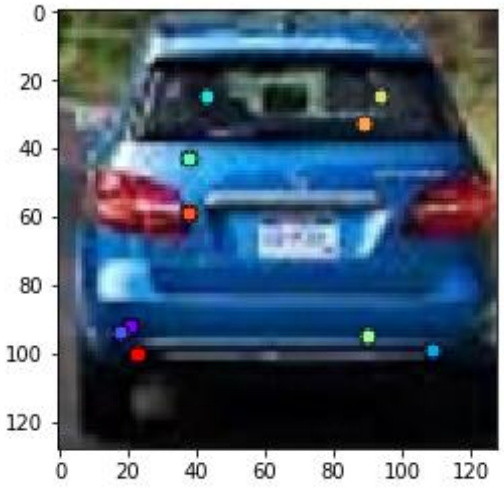

Figure 46: Keypoint features identified using KeypointNet

The matched model from the repository is shown in Figure 47 below.

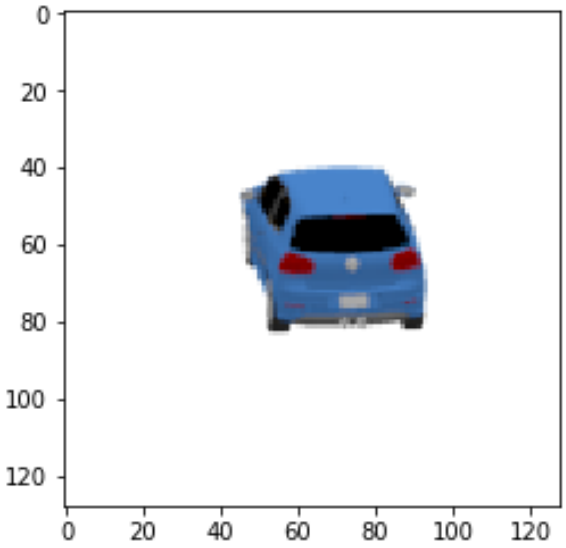

Figure 47: Matched model from the repository

Note that it was challenging to find a model matching the cars in the real time input image with the same orientation and view. The repository developed by the proposed overall system needs to be updated with real-time 3D car models as part of future work. The image centre and object centre candidates are determined using the algorithm. See Figure 48 below.

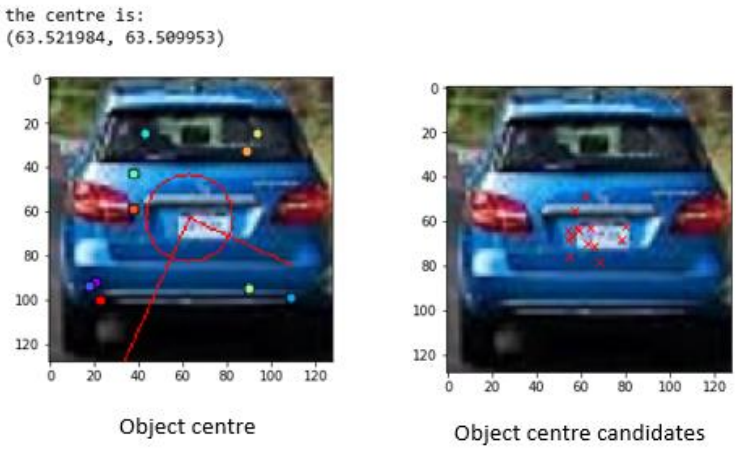

Figure 48: Object centre and object centre candidates

The clustering algorithm is then applied and the confidence score value of the object at the location is determined. The step wise flow is shown in Figure 49 below.

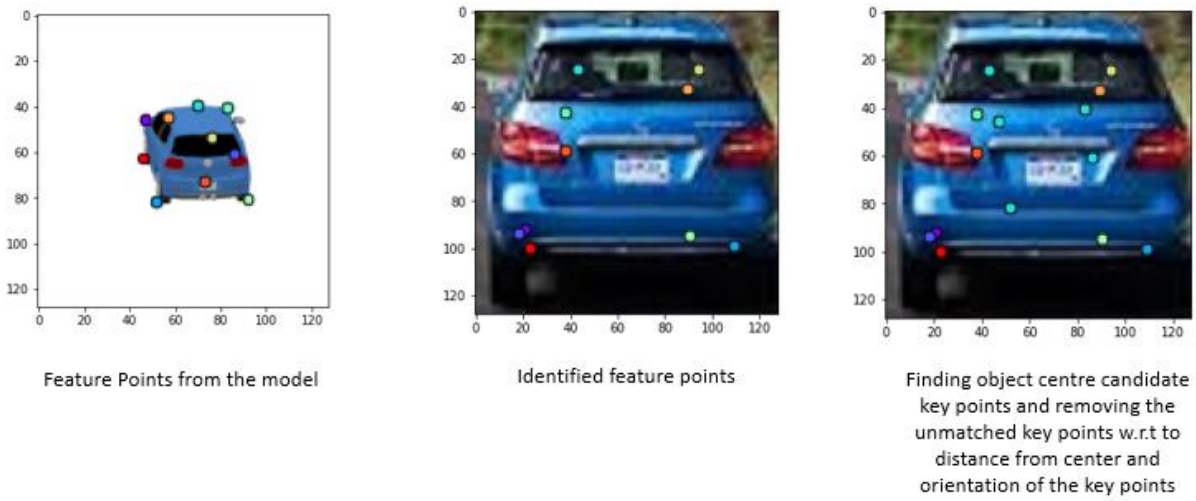

Figure 49: Step-wise representation of results for real time single car scenario

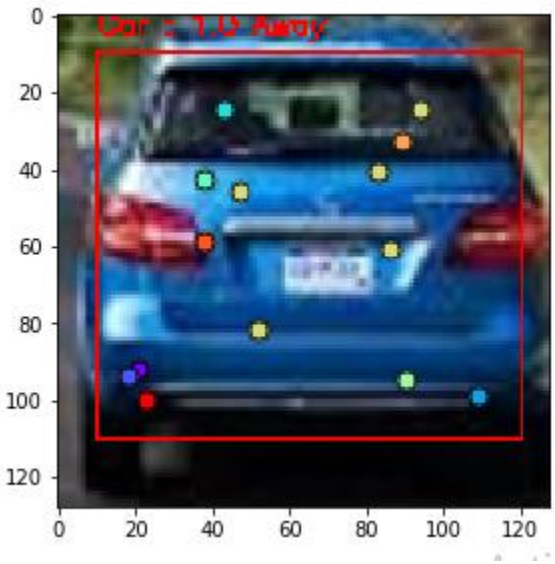

Figure 50: Car detected with confidence score = 1.0 in real time image

The proposed algorithm works quite well for a single car in the real-time input image. The algorithm was also tested on some special cases such as images with cars partly occluded by another car and images in which only a part of the vehicle is present (like the head of a trailer). The algorithm gave decent results in these scenarios. Each scenario is discussed in the sections below.

4.2.3 Real-time input image with a car partially occluded by another car

Most often multiple cars are found on the road and there are situations where one car is partially covered by another car or a small part of another car.

Consider the image of a real-time road scene in Figure 51 below.

Figure 51: Real time road scene with multiple cars

After applying the ImageAI python library and the Resnet model [72] for getting the cropped images of each of the identified cars as discussed in section 4.2.2, the different car images are obtained as shown in Figure 52 below.

(a) (b) (c)

Figure 52: Cars Detected and cropped using the ImageAI library and the Resnet model

As discussed in the above scenarios, when using the proposed algorithm, the image centre and object centre candidates are found to determine the keypoints that lie on the object. Figure 53 shows the results of the different steps of the algorithm.

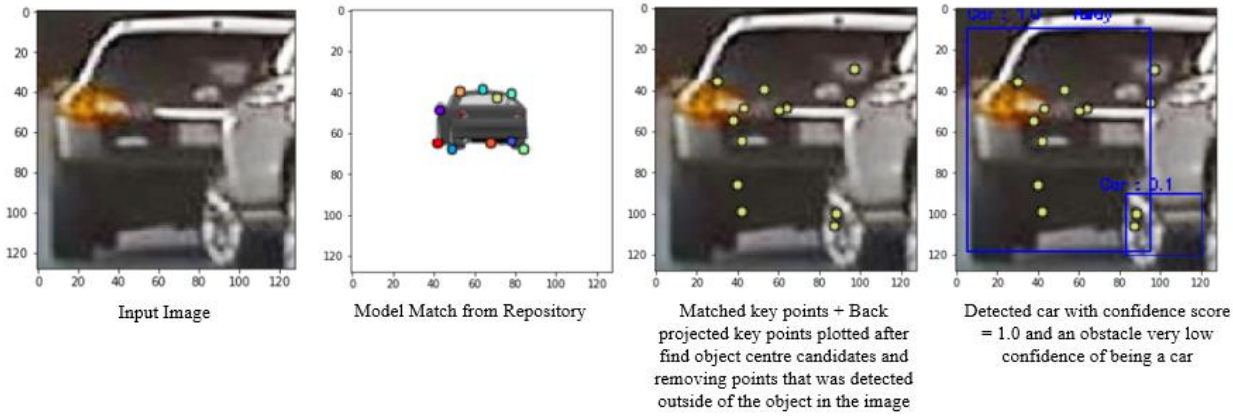

Figure 53: Step-wise results for the image of a car partly occluded by the tyre of another car

In this case, its evident that the main car is partly covered by wheels of the car behind it. The proposed algorithm correctly separates the keypoints identified on the wheels of the car behind into a different cluster. In this case, two clusters are formed. After identifying keypoints, model matching, back projecting model keypoints and removing key points far from centre, a total of 13 keypoints are obtained. Out of these 13 keypoints, 9 keypoints are the matched keypoints plus the back projected keypoints. The second cluster contains one of the matched keypoint. Hence if the

first cluster contains the rest of the nine keypoints, a high confidence score value is assigned to it. The first cluster containing all object centre candidates of the prominent car in the image, has 11 keypoints and the second cluster has just 2 keypoints. In the first cluster, 8 points match the template model (which includes all the matched and back projected keypoints), hence as mentioned in our algorithm, a high confidence score = 8/8 = 1.0 is assigned to that cluster. However only 1 point in 2nd cluster matches the template model, hence a very low confidence score of 1/9 = 0.1 is assigned to it as depicted in Figure 53.

4.2.4 Real-time input image with part of the vehicle missing

Consider the input image shown in Figure 54 below, where only the head of the trailer is moving on the road.

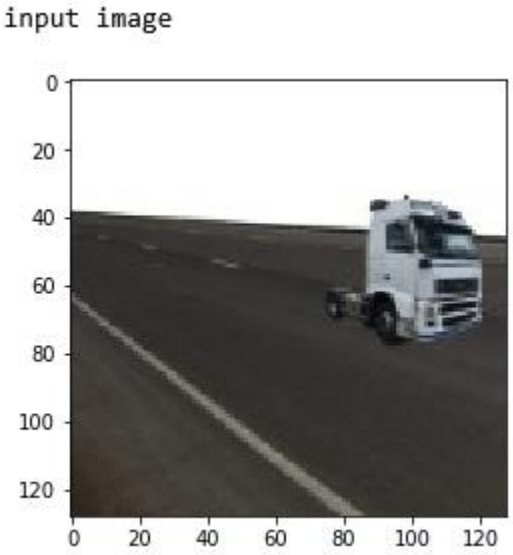

Figure 54: Tractor unit without a trailer on the road

When the repository only has the 3D model of the entire truck, i.e. the tractor unit along with its trailer (Figure 55), then the algorithm identifies the missing part of the vehicle. Initially, the keypoints of the input image match with the keypoints of the 3D model of the truck. Then the unmatched keypoints are back projected, and the object centre candidates are determined. Once the points lying far from the object centre are removed, the clustering technique is applied. Two clusters were formed. The first cluster had four keypoints matching the "matched" keypoint set (obtained in Step 2 of the proposed algorithm) and the back projected set of keypoints. Hence, a confidence score of, 4/10 = 0.4 was assigned to it. However, the second cluster didn't have any keypoints which matched with the 'matched' keypoint set (obtained in Step 2 of the proposed algorithm) and only had keypoints matching with the back projected set. Therefore, as discussed in the algorithm, a confidence score of, 0/10 = 0.0 was assigned to the second cluster. The step-wise results are displayed in Figure 55 below.

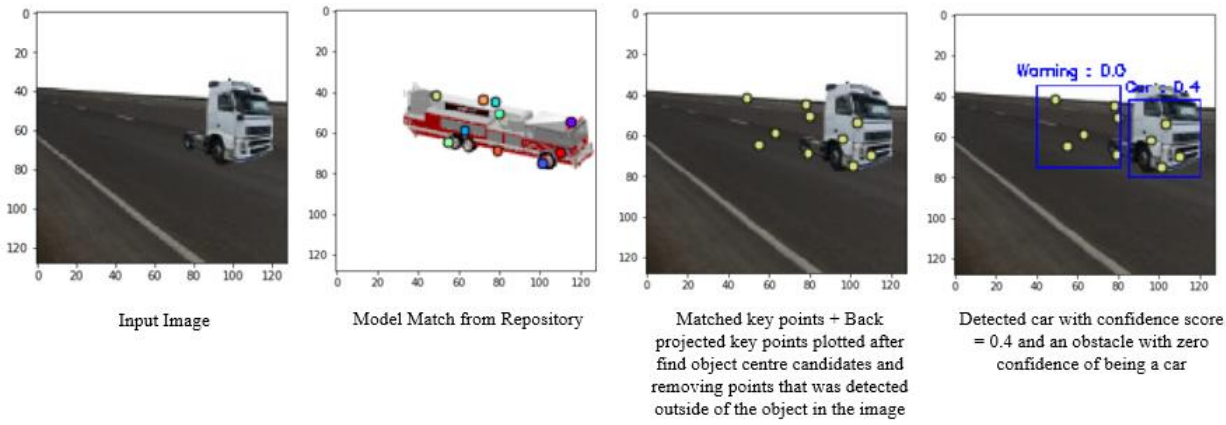

Figure 55: Step-wise results for vehicle with a missing part

The results of the proposed voting technique applied on cars shows how the keypoint features extracted from 3D models help in estimating the confidence of object recognition and determining

the occluded and missing parts of a vehicle. The next section discusses the experiments and results of object recognition and pose estimation of pedestrians in the real-time input.

4.2.5 Real-time input image with objects other than cars or pedestrians

Consider the images in Figure 56 below with no car or pedestrians in it.

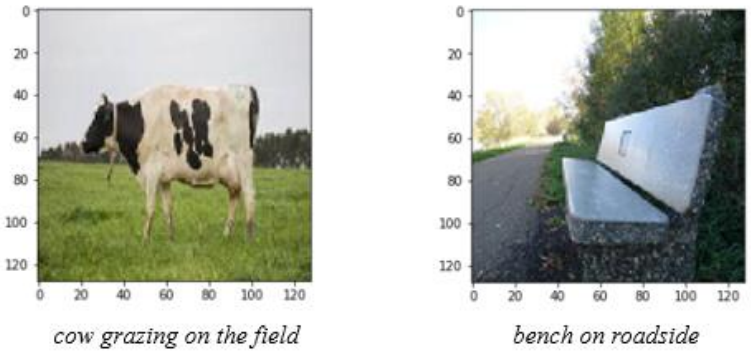

Figure 56: Input images with no car or pedestrians

The 3D keypoint features detected on these images does not match with any of the keypoint features of the rendered images of 3D models stored in the repository and hence goes unrecognized by our system, as expected.

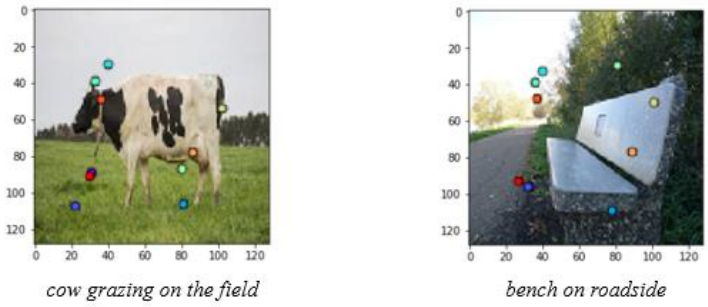

Figure 57: Keypoint features detected on images without cars or pedestrians

Figure 57 depicts the keypoint features detected on images without the objects of interest. This again signifies the advantage of using latent 3D keypoints for object recognition, rather than 2D feature points, as the 3D keypoints detects the interesting parts of different objects. Also, the orientation information of the keypoints of different rendered 3D models stored in the repository used for matching the dynamic objects in the input image helps in identifying the exact object, rather than giving false predictions.

4.3 Experiments and Results of Object Recognition and Pose Estimation of Pedestrians

As discussed in section 3.3.2 of chapter 3, due to the challenges in mapping the 2D human images to 3D models of the human body and the tedious effort and time required to create the repository of human 3D models in different pose and type along with labelling of their keypoint features, "DensePose"[40] model has been adopted for 3D human pose estimation in the scope of the overall system. The source code of “DensePose” [40], publicly available in their GitHub repository [73] was used to test the functioning of the overall system in different road scenarios. The different scenarios and their corresponding results are given below.

4.3.1 Input image with an adult on the road

Consider the input image of an adult on road as in Figure 58 below.

Figure 58: Input image with an adult on the road

The “DensePose” determines the UV coordinates from the human 3D models as explained in section 3.3.2. The visualization of the isocontours of the UV fields are shown in Figure 59 below.

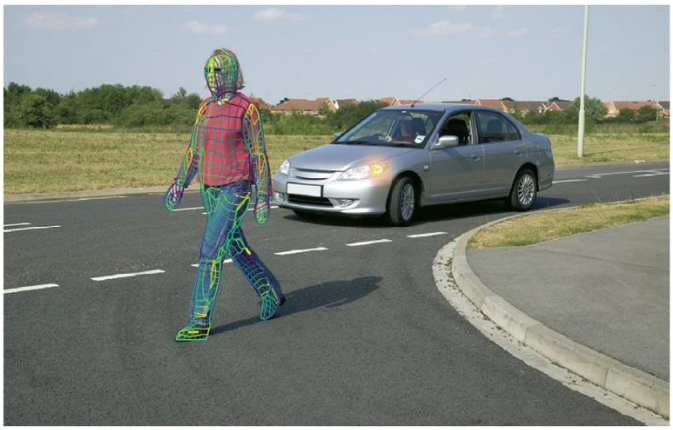

Figure 59: Visualization of the isocontours of the UV fields in the image with an adult on the road

The textures are then mapped on to the isocontours of the UV fields to get the complete look and feel of the 3D models. This is depicted in Figure 60 below.

Figure 60: Textures mapped on the isocontours of the UV fields in an image with an adult on the road

4.3.2 Input image with an adult and child on the road

Consider the input image of an adult and a child on road as in Figure 61 below.

Figure 61: Input image with an adult and a child crossing the road

The visualization of the isocontours of the UV fields are shown in Figure 62 below.

Figure 62: Visualization of the isocontours of the UV fields in the image with an adult and a child crossing the road

The textures are then mapped on-to the isocontours of the UV fields to get the complete look and feel of the 3D models. This is depicted in Figure 63 below.

Figure 63: Textures mapped on the isocontours of the UV fields for an image with an adult and a child crossing the road

4.3.3 Real-time input image with pedestrians running across the road

Consider an input image with two men running across the road as shown in Figure 64 below.

Figure 64: Input image with two men running across the road

The visualization of the isocontours of the UV fields are shown in Figure 65 below.

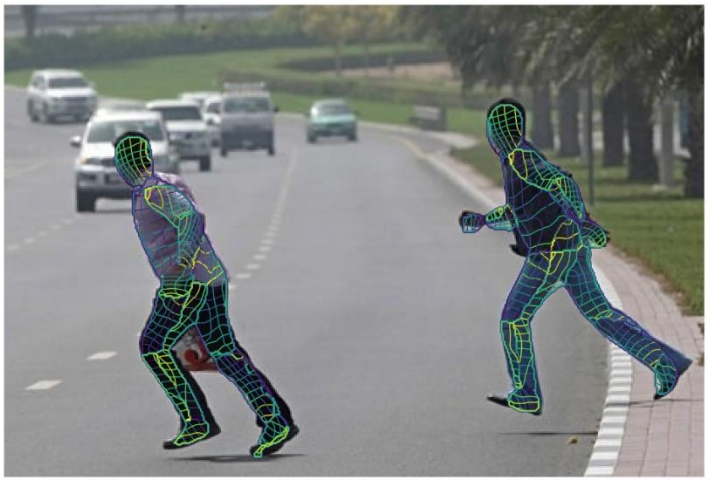

Figure 65: Visualization of the isocontours of the UV fields in an image with two men running across the road

The textures are then mapped on-to the isocontours of the UV fields to get the complete look and feel of the 3D models. This is depicted in Figure 66 below.

Figure 66: Textures mapped on the isocontours of the UV fields for an image with two men running across the road

4.3.4 Real-time input image with multiple people on road (Occluded humans)

Consider an input image with multiple pedestrians (few partly occluded) on the road as shown in Figure 67 below.

Figure 67: Input image with multiple pedestrians (few partly occluded) crossing the road

The visualization of the isocontours of the UV fields are shown in Figure 68 below.

Figure 68:Visualization of the isocontours of the UV fields in the image with multiple pedestrians (few partly occluded) crossing the road

The textures are then mapped on to the isocontours of the UV fields to get the complete look and feel of the 3D models. This is depicted in Figure 69 below.

Figure 69: Textures mapped on the isocontours of the UV fields for an image with multiple pedestrians (few partly occluded) crossing the road

The above results show that the "DensePose" model [40], can handle a variety of human poses on the road, multiple humans on the road, humans which are partly occluded by other humans or

objects, and short or tall humans. The scope of improvements to this model is discussed in section 4.6.

4.4 Results of the Tracking of Dynamic Objects

As explained in section 3.4, object tracking for cars and pedestrians on the road is also done as part of this research. The objects along with their speed in km/hr is depicted with rectangular bounding boxes. Figure 70 below, shows the different frames of the input video with the object tracking details in cars.

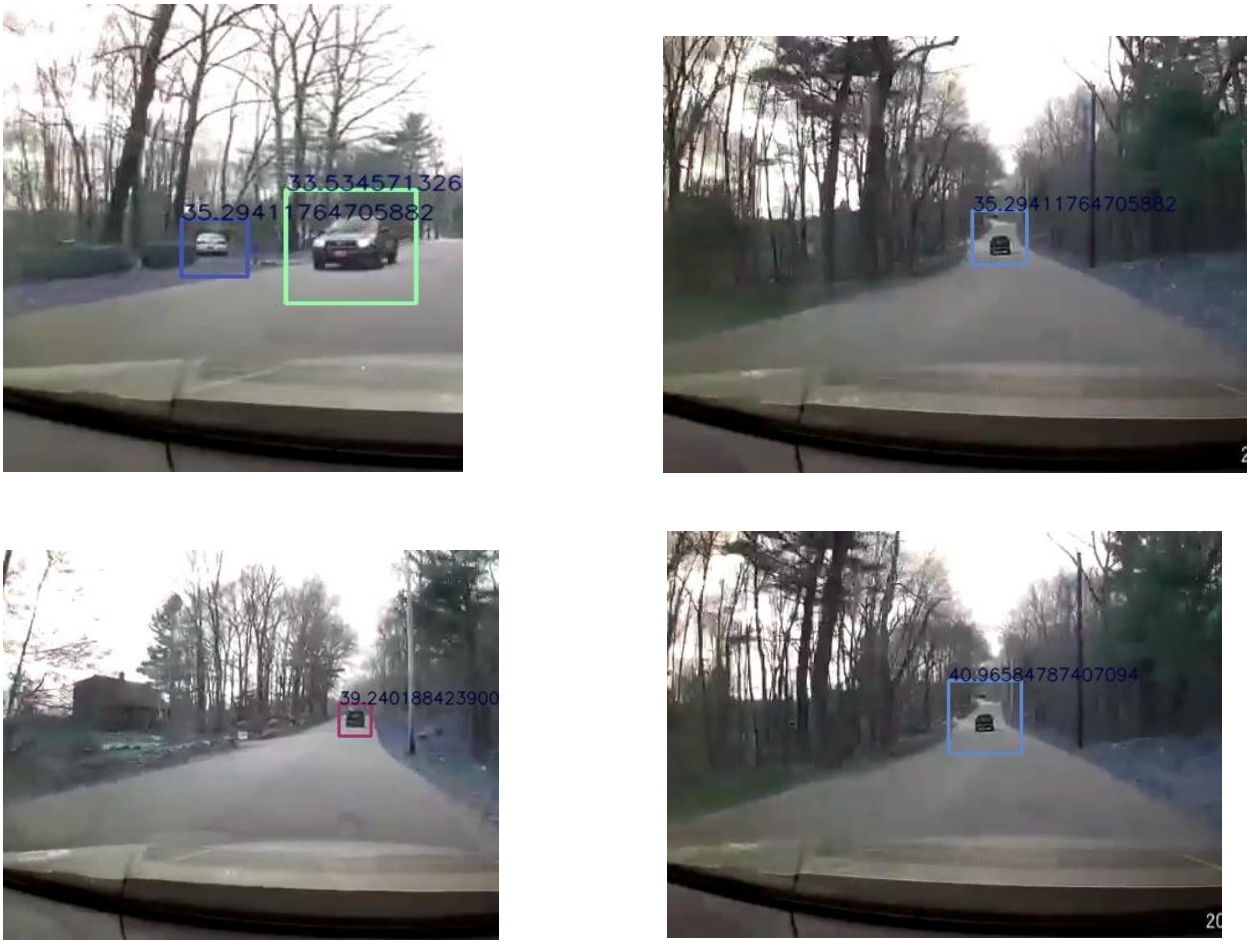

Figure 70: Different frames of the input video tracking the car with the estimated speeds displayed on it

Similarly, the Figure 71 below, shows the different frames of the input video with the object tracking details in pedestrians.

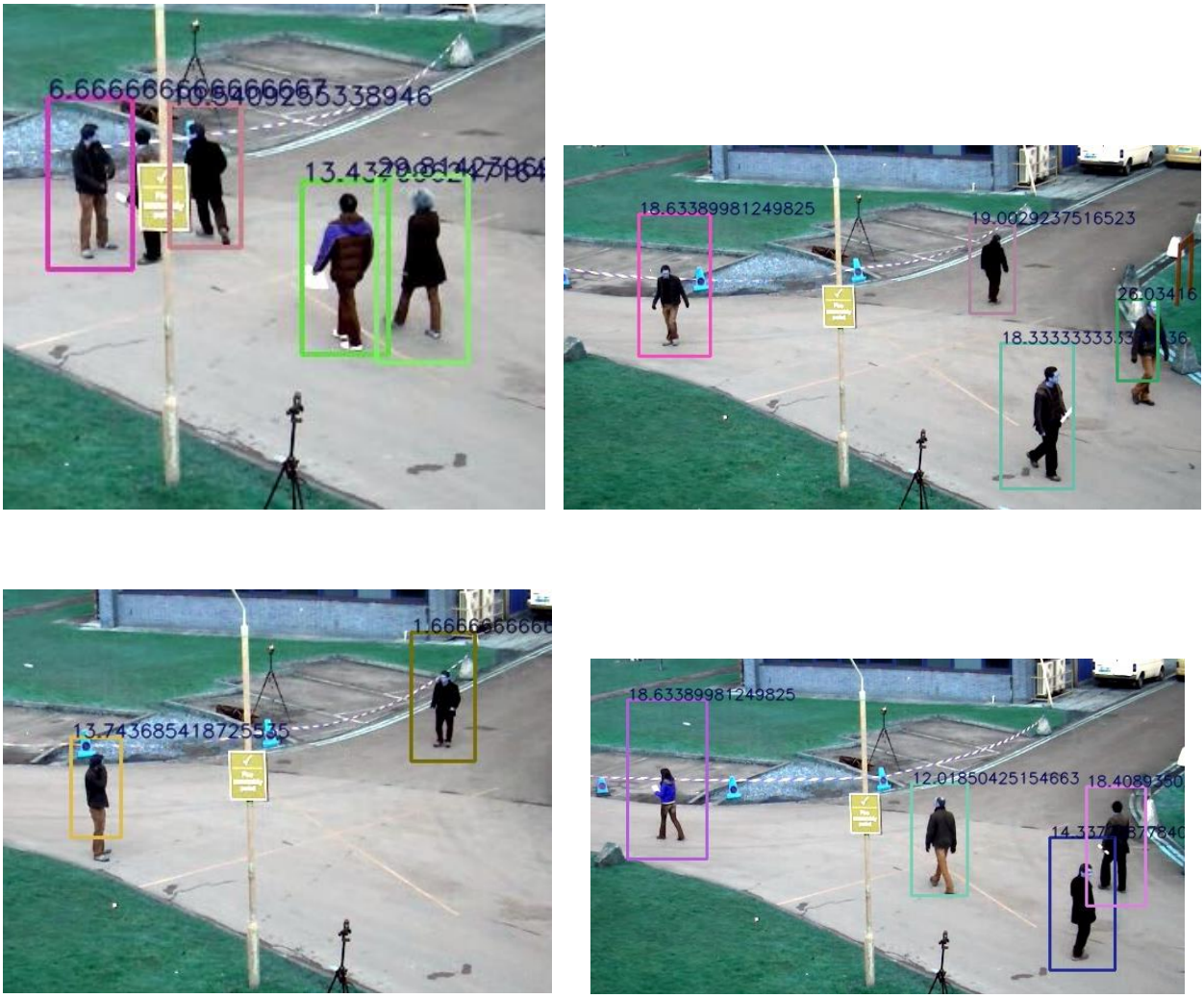

Figure 71: Different frames of the input video tracking the humans with the estimated speeds displayed on it

As discussed in section 3.4, the distance a vehicle has travelled in the image plane would be the relative differences in pixels at various vehicle positions in the subsequent frames. To get the exact speed of an object, it is required to traverse a few frames in order to have an idea of the distance it covers over the frame. The speed and location details are also passed on to the virtual city for updating the recognized dynamic objects.

4.5 Comparison and Discussions

As mentioned in the introduction, most of the previous researches [1, 28, 84] use the mean average precision(mAP) as the measure of accuracy in the object detection and pose estimation predictions. Therefore, it is important to first define the term mean average precision before using it to compare the results of the proposed approach with that of the previous works.

4.5.1 Mean Average Precision

AP (Average precision) is a popular metric in measuring the accuracy of object detectors like Faster R-CNN or SSD, etc. Average precision computes the average precision value for the recall value over 0 to 1[85]. Precision and Recall may be defined as below [89]:

Recall: is the fraction of relevant instances that are retrieved.

$R = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truth\ instances}$, where, TP is the True Positive and FN is the False Negative.

Precision: is the fraction of retrieved instances that are relevant.

$P = \frac{TP}{TP+FP} = \frac{TP}{all\ predicted}$, where, the TP is the True Positive and FP is the False Positive.

mAP (mean average precision) is the average of AP. In some context, the AP for each class is computed and averaged. But in some contexts, they mean the same thing. In the state-of-the-art YOLOv3 approach [88] there is no difference between AP and mAP [85].

4.5.2 Estimation of Mean Average Precision using the proposed method

Due to limited open source 3D model car datasets closely matching real-time cars and other constraints like the time taken to render each model and to create corresponding annotation files, so far, the proposed algorithm was tested with 100 models in the repository and 40 real-time images. The 4 different pose labels, Left, Right, Towards and Away are used to determine the direction in which the car is moving on the road.

The confusion matrix table was used to calculate the mean average precision. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known [90]. The table below shows the confusion matrix obtained using the proposed approach for the test set of 40 input images.

Predicted Pose of the Cars					
Actual Pose of the Cars		Left	Right	Towards	Away
	Left	8	1	0	0
	Right	3	12	0	0
	Towards	0	0	6	0
	Away	0	0	0	10

Table 4: Confusion Matrix

The actual poses are the direction of the car in the input image. And the predicted poses are the direction of the car specified in the annotation file of the matched model found from the repository by the proposed approach.

From the formula of precision defined in Section 4.5.1, the precision values for each of the poses of the car is calculated as below [86,87]:

P(Left) = $\frac{8}{(8+3+0+0)}$ = 0.727

P(Right) = $\frac{12}{(1+12+0+0)}$ = 0.923

P(Away) = $\frac{6}{(0+0+6+0)}$ = 1.00

P(Towards) = $\frac{10}{(0+0+0+10)}$ = 1.00

The mean Average Precision is then calculated as: $\frac{(0.727 + 0.923 + 1.00 + 1.00\ )}{4}$ = 0.9107 = 91.07 %

Table 5 contains the mAP values estimated for object recognition in state-of-the-art deep learning approaches, approach by Tangruamsub et al. [1] and the approach proposed in this thesis.

Existing deep learning approaches			
Approach	**mAP**		
YOLOv2 on VOC2007	78.6		
3D YOLO based on LiDAR data (Hakim,2018 [84])	65.10		
***[1]Comparison with Tangruamsub et al., 2011**			
Approach	**Number of Models/Images in Repository**	**Number of Test Images**	**mAP**
Tangruamsub et al.	553	544	86
Our method	100	40	91.07

Table 5: mAP values for different approaches

*Comparison with the approach using voting algorithm for object recognition and pose estimation by extracting feature points from 2D data.

The repository needs to be updated with more car models and their corresponding annotation files in order to test the proposed approach with additional input images.

4.5.3 Results Comparison and Discussion: Cars

Having discussed the results of the proposed approach on different scenarios of cars, the way in which this approach would improve object recognition in the context of autonomous driving is discussed below.

4.5.3.1 Advantages of the proposed system

The major advantages that were observed as a result of using 3D models for object recognition and subsequent pose estimation of the dynamic objects in the proposed system are discussed below.

a. Improvement in confidence of object recognition and pose estimation

The state-of-the-art deep learning models (ResNet, YOLO) detects objects with high accuracy. However, applications like that of autonomous vehicles which involve the lives of humans on the road require even more precise and accurate predictions. This is one of the reasons that even with a lot of advancements in deep learning or other machine learning techniques, a fully autonomous vehicle is still far from viability. Figure 72 below shows the output of object detection with the accuracy of the predicted object using the Resnet model [72]. The prediction gives an accuracy of 92.847% that the detected object is a car.

Figure 72: Object recognition with Resnet model

The proposed approach aims to improve the accuracy level by using 3D car models and compare the real-time image keypoints with that of the 3D model keypoint features. The use of 3D object models helps us to make use of the depth and orientation information which help detect very specific keypoints of the car (like the wheels and headlights). This keypoint features help us to assign a confidence score to the detected object as explained in the voting algorithm.

Therefore, it can be claimed that the algorithm would help to improve the confidence that the detected object is a car at that location. The proposed algorithm gave a confidence score = 1.0 (highest confidence value) that the object is present at the location if all keypoints identified match exactly with the template model keypoints. Apart from just detecting the object with a confidence value the algorithm also gives the direction of the moving car (Left, Right, Away, Towards). This is pictorially represented in Figure 73 below.

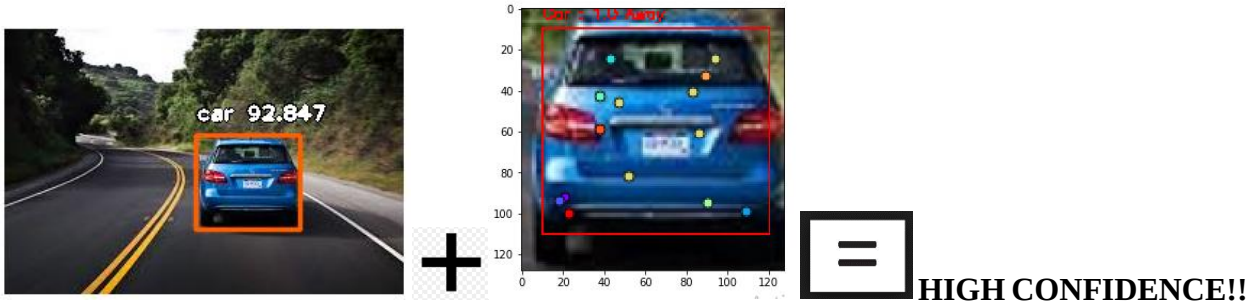

Figure 73: Pictorial representation of how the proposed approach improves prediction confidence

Figure 74 below depicts another example in which the back tyre of the car is occluded. The 3D model of car in the same orientation is retrieved from the repository by the system. The system detects the car in the image with high confidence after back projecting the keypoint features of the matched template model onto it.

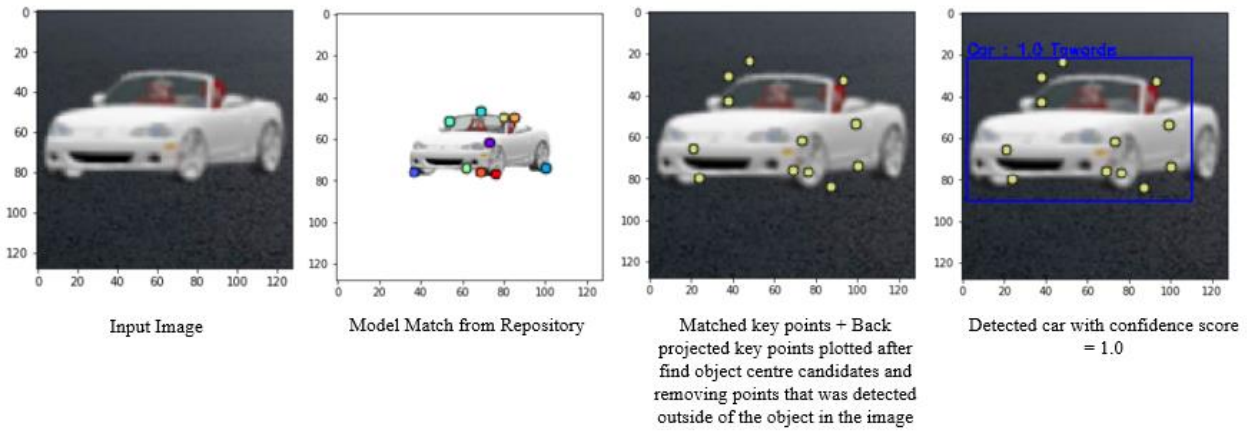

Figure 74: Step-wise representation of results for input image of car with partial occlusion

Hence it is clear than the use of keypoint features extracted from 3D models not only yield better confidence for object recognition but also can be used for back-projection for the purpose of object verification.

b. No incorrect detection of objects by the system

Tangruamsub et al.,2011 [1] discusses the cases of incorrect detections in their official paper. The system proposed in [1] recognizes cow, cycle and cat instead of cars as shown in Figure 75.

Figure 75: Objects detected incorrectly instead of cars (Source: Tangruamsub et al.,2011)

As discussed in section 4.2.5, the proposed system can filter the images having no car or pedestrian, thus avoiding incorrect detections. This signifies the advantage of using latent 3D keypoints for object recognition (Figure 57), rather than feature points extracted from 2D images. Also, the orientation information of the keypoints of different rendered 3D models stored in the repository used for matching the dynamic objects in the input image helps in identifying the exact object, rather than giving incorrect detections.

4.5.4 Results Comparison and Discussion: Humans

In the context of autonomous driving, the human pose estimation is very crucial, as even a slight error in estimation can cost human lives. The “DensePose” model [40], has given good results in

a variety of scenarios, like scenes with multiple persons, pedestrians occluded by other pedestrians or objects, and short and tall pedestrians. The official paper of DensePose claims an AP (Average Precision) of 87.5 in case of pose estimation obtained through cascading. "DensePose", instructs the annotators to estimate the body part behind the clothes, so that for instance wearing a long skirt would not complicate the subsequent annotation of correspondences. Though this is very useful in many applications, in the context of autonomous driving, it is important to know the exact location of the legs behind the long skirt. For instance, consider the woman in Figure 76 below. The results after applying the "DensePose" model [40] are shown in Figure 77 below. As expected it predicts the limbs behind the long skirt. Now, suppose that the women in the figure is handicapped and does not have legs. In the case of autonomous driving, this information is very crucial to make suitable navigation decisions. The proposed voting technique can be integrated into this approach to determine a confidence score value of the body part being there at that location in the image. This additional information would improve the confidence with which the autonomous car could take navigation decisions.

Figure 76: Woman wearing a long skirt

Figure 77: Pose estimation results after applying DensePose model

However, with limited resources and manpower, creating the repository with the different human models with their corresponding keypoint annotations is cumbersome. Therefore, it is kept aside as future scope of the overall proposed system.

4.6 Observation and Limitations

The proposed approach finds the suitable matching car model from the repository in most cases. Therefore, the correct direction of the car is also retrieved from the annotation files corresponding to the matched template model in most cases.

However, it was observed that in some scenarios where the “Left” pose and “Right” pose of the cars look very similar (for cars whose front and back look somewhat similar), like the two cars shown in Figure 78, the pattern of keypoints detected look very similar, and hence the template models match with each other even though they are in a different pose. The confusion matrix (Table 3) also shows that the “Left” and “Right” poses were wrongly predicted in some cases. The

KeypointNet model [23] also reports in their official paper that the orientation network fails to predict correct orientation and the output keypoints are flipped in the case of cars whose front and back look similar. This is shown in Figure 79 which is taken from the official paper of the KeypointNet model [23]. This is another area of the system which has scope for improvement in the future.

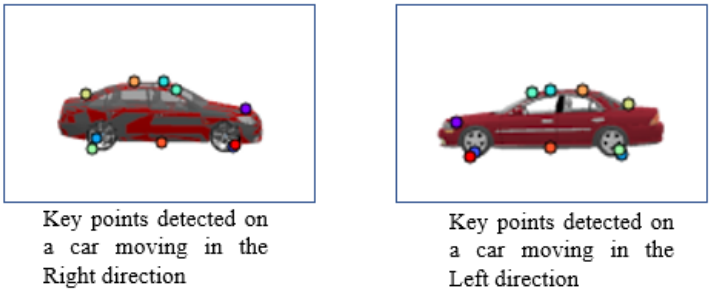

Figure 78: Two cars in two opposite directions with keypoints at similar coordinate positions

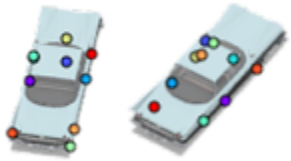

Figure 79: Failure case with output keypoints flipped in the case of the KeypointNet model

Chapter 5: Conclusion and Future Work

Most of the leading car companies today are looking forward to making the dream of a fully autonomous car a reality, with an eye on the huge impact or revolution it would create. There are several advantages that an autonomous car offers. These include less traffic, increased safety, and less wastage of time on driving. However, for the autonomous cars to perform with zero faults or accidents, it is required that the machine does object recognition and pose estimation of the different objects on the road just as the human brain does. This requires the machine to understand the 3D structure of any object it sees on the road. The aim of the proposed approach is to make the system recognize the 3D structure of the objects on the road with the help of 3D object models. The virtual city, proposed in this research, contains 3D models of the buildings and other static objects on the road. This information is used by the car to determine the presence of a building (or tree) and get its 3D views which helps it to take the correct navigation decisions. The dynamic objects are identified, and their corresponding 3D models are retrieved from the repository. In addition to that, the object is tracked to determine its speed and location information using bounding boxes. All this information is sent to the virtual city where the simulated dynamic objects are updated in real-time.

Even though the 3D models of cars or other objects are expensive, it is worth the investment when it comes to the safety of humans on the road. This approach of using 3D models and extracting the keypoints of the significant parts of the cars, human body, and buildings have proved to improve the object recognition confidence. As discussed in Chapter 4, the experiments show that the proposed algorithm was able to handle the special cases of cars occluded by other cars or identify obstacles when a part of the car is missing. It is anticipated that once all the modules of the overall system are completed, much better results can be produced. This means the completion of a fully

updated dynamic object repository with rendered images of real-time car models, and completion of other modules like the static and variable object elimination which would improve the overall performance of the system.

Though the proposed approach has given decent results, there is a lot of room for improvement. As discussed, in section 4.5.3, the human pose estimation needs to be improved to be able to handle all scenarios in the context of the autonomous driving domain. Also, currently the proposed algorithm uses a set of pre-defined threshold values (Refer to the algorithm in section 3.5.2), which have been determined by running a variety of different experiments. Automating the task of finding the threshold values for the voting and clustering algorithm can be taken up as future work. Also, the KeypointNet model [23], can be modified to extract a different number of keypoints and then update the repository accordingly. The proposed system can then use this new set of keypoints. The time complexity of the system can be reduced considerably if deep learning techniques are integrated into the system to recognize the dynamic objects belonging to different classes (cars or pedestrians) initially and then apply the proposed algorithm for pose and confidence score estimation. This is another area for future scope. Finally, the proposed system can also be extended to other dynamic objects like animals and birds or any other objects which would impact driving on the road.

References

[1] S. Tangruamsub, K. Takada, and O. Hasegawa, “3d object recognition using a voting algorithm in a real-world environment,” in Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision (WACV), ser. WACV ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 153–158. [Online]. Available: http://dx.doi.org/10.1109/WACV.2011.5711497

[2] Wei Tang, Pei Yu and Ying Wu, "Deeply Learned Compositional Models for Human Pose Estimation", ECCV 2018, SpringerLink: https://link.springer.com/conference/eccv

[3] Jun SHIMAMURA, Taiga YOSHIDA, Yukinobu TANIGUCHI, Hiroko YABUSHITA, Kyoko SUDO, and Kazuhiko MURASAKI, "The method based on view-directional consistency constraints for robust 3D object recognition", MVA2015 IAPR International Conference on Machine Vision Applications, May 18-22, 2015, Tokyo, JAPAN

[4] W. Wohlkinger and M. Vincze, “Ensemble of shape functions for 3d object classification,” in 2011 IEEE International Conference on Robotics and Biomimetics, Dec 2011, pp. 2987–2992

[5] R. Socher, B. Huval, B. Bhat, C. D. Manning, and A. Y. Ng, “Convolutional-recursive deep learning for 3d object classification,” in Proceedings of the 25th International Conference on Neural Information Processing Systems, ser. NIPS’12. USA: Curran Associates Inc., 2012, pp. 656–664. [Online]. Available:http://dl.acm.org/citation.cfm?id=2999134.2999208

[6] C. Li, J. Bohren, E. Carlson, and G. D. Hager, “Hierarchical semantic parsing for object pose estimation in densely cluttered scenes,” in 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 5068–5075

[7] S. A. A. Shah, M. Bennamoun, and F. Boussaid, “A novel feature representation for automatic 3d object recognition in cluttered scenes,” Neurocomputing, vol. 205, pp. 1 – 15, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215017385

[8] Luís Eduardo Ramos de Carvalho, Aldo Von Wangenheim, “Literature review for 3D object classification/recognition”, 2017.

[9] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepercut: A deeper, stronger, and faster multiperson pose estimation model. In Proc. of the European Conference on Computer Vision (ECCV), 2016

[10] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In Proc. of the European Conference on Computer Vision (ECCV), pages 483–499.Springer, 2016

[11] Object recognition: https://www.mathworks.com/solutions/deep-learning/object-recognition.html

[12] Human pose estimation: https://www.learnopencv.com/deep-learning-based-human-pose-estimation-using-opencv-cpp-python/

[13] Semantic segmentation blog: https://medium.com/intro-to-artificial-intelligence/semantic-segmentation-udaitys-self-driving-car-engineer-nanodegree-c01eb6eaf9d

[14] Marvin Teichmann, Michael Weber, Marius Zollner, Roberto Cipolla and Raquel Urtasun, "MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving",2018 IEEE Intelligent Vehicles Symposium (IV) Changshu, Suzhou, China, June 26-30, 2018

[15] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, Antonio M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban

Scenes", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3234-3243

[16] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, 2012 ImageNet Competition

[17] Karen Simonyan , Andrew Zisserman, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”, Published as a conference paper at ICLR 2015

[18] Christian Szegedy1 , Wei Liu2 , Yangqing Jia1 , Pierre Sermanet1 , Scott Reed3 , Dragomir Anguelov1 , Dumitru Erhan1 , Vincent Vanhoucke1 , Andrew Rabinovich4, “Going Deeper with Convolutions”, published at CVPR2015

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian SunDeep, "Residual Learning for Image Recognition", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

[20] https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html

[21] R-CNN tutorial: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[22] Semantic segmentation using deep learning: https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef

[23] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi. Discovery of latent 3D keypoints via end-to-end geometric reasoning. In NIPS, 2018.

[24] Beginner’s guide to object detection for self driving cars: https://skymind.ai/wiki/autonomous-vehicle

[25] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation",arXiv:1311.2524v5 [cs.CV] 22 Oct 2014

[26] R. Girshick. Fast R-CNN. In ICCV, 2015.

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[28] Joseph Redmon, Ali Farhadi, "YOLO9000: Better, Faster, Stronger", http://pjreddie.com/yolo9000/, 2016

[29] Khaled Alhamzi, Mohammed Elmogy, Sherif Barakat, "3D Object Recognition Based on Image Features: A Survey", International Journal of Computer and Information Technology (ISSN: 2279 – 0764) , Volume 03 – Issue 03, May 2014

[30] Sapp, B., Taskar, B.: Modec: Multimodal decomposable models for human pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition.(2013) 3674–3681

[31] Johnson, S., Everingham, M.: Clustered pose and nonlinear appearance models for human pose estimation. In: British Machine Vision Conference. (2010)

[32] Andriluka, M., Pishchulin, L., Gehler, P., Schiele, B.: 2d human pose estimation: New benchmark and state of the art analysis. In: IEEE Conference on computer Vision and Pattern Recognition. (2014) 3686–3693

[33] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(7):1325– 1339, Jul 2014

[34] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, Yichen Wei, "Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach", arXiv:1704.02447v2 [cs.CV] 30 Jul 2017

[35] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, Cordelia Schmid,"Learning From Synthetic Humans",The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 109-117

[36] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler1, Javier Romero, Michael J. Black, "Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image", Springer International Publishing,2016

[37] Alexandre Saint, Eman Ahmed, Abd El Rahman Shabayek, Kseniya Cherenkova, Gleb Gusev, Djamila Aouada1 and Bjorn Ottersten, "3DBodyTex: Textured 3D Body Dataset", https://www.researchgate.net/publication/327263103, 2018

[38] Christoph Lassner, Javier Romero, Martin Kiefe, Federica Bogo, Michael J. Black, Peter V. Gehler, "Unite the People: Closing the Loop Between 3D and 2D Human Representations", arXiv:1701.02468v3 [cs.CV] 25 Jul 2017

[39] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter V. Gehler, Bernt Schiele, "Neural Body Fitting: Unifying Deep Learning and Model-Based Human Pose and Shape Estimation", arXiv:1808.05942v1 [cs.CV] 17 Aug 2018

[40] Rıza Alp Guler, Natalia Neverova, Iasonas Kokkinos, "DensePose: Dense Human Pose Estimation in The Wild", arXiv:1807.03146v2 [cs.CV] 23 Nov 2018

[41] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia), 34(6):248:1–248:16, Oct. 2015.

[42] ROBINETTE, K., BLACKWELL, S., DAANEN, H., BOEHMER, M., FLEMING, S., BRILL, T., HOEFERLIN, D., AND BURNSIDES, D. 2002. Civilian American and European Surface Anthropometry Resource (CAESAR) final report. Tech. Rep. AFRL-HEWP-TR-2002-0169, US Air Force Research Laboratory.

[43] ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: Shape Completion and Animation of PEople. ACM Trans. Graph. (Proc. SIGGRAPH 24, 3, 408–416

[44] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[45] V. Ramakrishna, T. Kanade, and Y. Sheikh. Reconstructing 3D human pose from 2D image landmarks. Proc. of the European Conference on Computer Vision (ECCV), pages 573– 586, 2012.

[46] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", In CVPR, 2015.

[47] Mohamad, Mustafa. "3D Object Recognition using Local Shape Descriptors." Tech. rep. School of Computing Queen's University Kingston, Ontario, Canada, 2013.

[48] Litomisky, Krystof. "Consumer RGB-D Cameras and their Applications." Tech. rep. University of California, 2012.

[49] Treiber, Marco. An Introduction to Object Recognition Selected Algorithms for a Wide Variety of Applications. Springer-Verlag London Limited, 2010.

[50] Yang, Ming-Hsuan. "Object Recognition". University of California at Merced. n.d.

[51] Matas, Jir and Stepan Obdrzalek. "Object recognition methods based on transformation covariant features." Proc of European signal processing conf on EUSIPCO (2004).

[52] Outline of object recognition. http://en.wikipedia.org/wiki/Outline_of_object_recognition. 3 2014.

[53] Schmid, C. and R. Mohr. "Local Grey Value Invariants for Image Retrieval." IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:530–535 (1997).

[54] Liang-Chia Chen, Hoang Hong Hai, Xuan-Loc Nguyen and Hsiao-Wen Wu. "Novel 3-D Object Recognition Methodology Employing a Curvature-Based Histogram" International Journal of Advanced Robotic Systems 10 (2013).

[55] Harris, C. and M. Stephens. "A Combined Corner and Edge Detector." Alvey Vision Conference, 147–151 (1988).

[56] Lowe, D.G. "Distinctive Image Features from Scale-Invariant Viewpoints." International Journal of Computer Vision, 60:91–111(2004).

[57] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. "SURF: Speeded Up Robust Features." Computer Vision–ECCV 2006. Springer Berlin Heidelberg (2006).

[58] Rublee, Ethan, et al. "ORB: an efficient alternative to SIFT or SURF." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[59] O'hara, Stephen and draper, bruce a. "Introduction to the Bag of Features Paradigm for Image Classification and Retrieval." arXiv preprint arXiv:1101.3354 (2011).

[60] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, Fisher Yu, "ShapeNet: An Information-Rich 3D Model Repository",arXiv:1512.03012v1 [cs.GR] 9 Dec 2015.

[61] Details about 3D object files, https://www.reviversoft.com/file-extensions/obj

[62] B.Boufama, "Image Formation and Camera Model", Lecture Notes, University of Windsor.

[63] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollar, "Microsoft COCO: Common Objects in Context", arXiv:1405.0312v3 [cs.CV] 21 Feb 2015

[64] SMPL: A Skinned Multi-Person Linear Model (SIGGRAPH Asia 2015): Presentation video: https://www.youtube.com/watch?v=kuBlUyHeV5U.

[65] R. A. Guler, G. Trigeorgis, E. Antonakos, P. Snape, S. Zafeiriou, and I. Kokkinos. Densereg: Fully convolutional dense shape regression in-the-wild. In CVPR, 2017. 2, 5

[66] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. CVPR, 2017. 2, 5, 6, 10

[67] DensePose tutorial video: https://www.youtube.com/watch?v=EMjPqgLX14A

[68] Haar Cascade OpenCV python tutorial : https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/

[69] Mohan Kagita, "Real time Vehicle Speed and Lane Detection", UNSW,Sydney,Australia, https://github.com/kmr0877/Realtime-Object-Tracking-using-Opencv

[70] Aditya Pai, "Real Time Detection and Classification of Vehicles and Pedestrians Using Haar Cascade Classifier with Background Subtraction", https://github.com/AdityaPai2398/Vehicle-And-Pedestrian-Detection-Using-Haar-Cascades

[71] Sultan Daud Khan, "Estimating Speeds and Directions of Pedestrians in Real-Time Videos: A solution to Road- Safety Problem", Conference: Ageing AI 2013 The Challenge of Ageing Society: Technological Roles and Opportunities for Artificial Intelligence.

[72] ImageAI Python: https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606

[73] "DensePose" GitHub repository: https://github.com/facebookresearch/Densepose

[74] S. Kolski, D. Ferguson, C. Stachniss, and R. Siegwart, "Autonomous driving in dynamic environments", in Proc. Workshop on Safe Navigation in Open and Dynamic Environments at the 2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2006), Oct. 9-15, 2006, Beijing, China. Piscataway, NJ: IEEE, 2006.

[75] Darms, M., Rybski, P., Urmson, C.," Vehicle Detection and Tracking for the Urban Grand Challenge", AAET - Automatisierungs-, Assistenz- und eingebettete Systeme fr Transportmittel, Symposium 13./14. Feb 2008, Braunschweig, 2008.

[76] Darms, M., Rybski, P., & Urmson, C. (2008b). Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, the Netherlands (pp. 1192–1202). IEEE

[77] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, ‘‘Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles,’’ Mech. Syst. Signal Process., vol. 100, pp. 482–500, Feb. 2018.

[78] How to Self-Driving Cars See, Computer Vision for Object Detection: https://towardsdatascience.com/how-do-self-driving-cars-see-13054aee2503

[79] Gao HB, Cheng B, Wang JQ, Li KQ, Zhao JH, Li DY. Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment.IEEE Trans Ind Inform 2018;99:1.

[80] Gene Lewis,"Object Detection for Autonomous Vehicles", Stanford University, Stanford, CA

[81] J. Janai, F. Guney, A. Behl, and A. Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-ofthe-art. arXiv preprint arXiv:1704.05519, 2017.

[82] Meaning of confidence, Machine learning theory: http://hunch.net/?p=317

[83] Difference between confidence and accuracy in machine learning: https://stats.stackexchange.com/questions/310559/difference-between-confidence-and-accuracy

[84] Ezeddin Al Hakim, “3D YOLO: end-to-end 3D object detection using point clouds”, Stockholm, Sweden 2018

[85] Mean Average Precision for Object Detection: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[86] Computing mean average precision/recall for multiclass/multi-label classification: https://stats.stackexchange.com/questions/21551/how-to-compute-precision-recall-for-multiclass-multilabel-classification

[87] Performance measure on multiclass classification: https://www.youtube.com/watch?v=HBi-P5j0Kec

[88] Joseph Redmon, Ali Farhadi, "YOLOv3: An Incremental Improvement", University of Washington

[89] Dr. Robin Gras, Lecture Slides: Artificial Intelligence, University of Windsor.

[90] Confusion matrix terminology: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[91] Zhong-Hua Hao, Shi-Wei Ma, "Object recognition and pose estimation using appearance manifolds",Hao, ZH. & Ma, SW. Adv. Manuf. (2013) 1: 258. https://doi.org/10.1007/s40436-013-0022-5

[92] A. Osep, W. Mehner, M. Mathias, and B. Leibe, “Combined image-and world-space tracking in traffic scenes,” in Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017.

Vita Auctoris

NAME: Chandini Ravindranathan Nair

PLACE OF BIRTH: Ernakulam, Kerala, India

YEAR OF BIRTH: 1991

EDUCATION: Bachelor of Technology, 2008-2012
Cochin University of Science and
Technology, Ernakulam, Kerala, India
Master of Science in Computer Science,
2017- 2019
University of Windsor, Windsor, ON