9-24-2019

# Using Prior Knowledge for Verification and Elimination of Stationary and Variable Objects in Real-time Images

Foram Pravinkumar Patel
*University of Windsor*

# Using Prior Knowledge for Verification and Elimination of Stationary and Variable Objects in Real-time Images

by

FORAM PRAVINKUMAR PATEL

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

# Using Prior Knowledge for Verification and Elimination of Stationary and Variable Objects in Real-time Images

by

FORAM PRAVINKUMAR PATEL

APPROVED BY:

_____

M. Hlynka

Department of Mathematics and Statistics

_____

A. Mukhopadhyay

School of Computer Science

_____

X. Yuan, Advisor

School of Computer Science

September 24, 2019

# Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights. Any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office and that this thesis has not been submitted for a higher degree to any other University or Institution.

# Abstract

With the evolving technologies in the autonomous vehicle industry, now it has become possible for automobile passengers to sit relaxed instead of driving the car. Technologies like object detection, object identification, and image segmentation have enabled an autonomous car to identify and detect an object on the road in order to drive safely. While an autonomous car drives by itself on the road, the types of objects surrounding the car can be dynamic (e.g., cars and pedestrians), stationary (e.g., buildings and benches), and variable (e.g., trees) depending on if the location or shape of an object changes or not. Different from the existing image-based approaches to detect and recognize objects in the scene, in this research 3D virtual world is employed to verify and eliminate stationary and variable objects to allow the autonomous car to focus on dynamic objects that may cause danger to its driving. This methodology takes advantage of prior knowledge of stationary and variable objects presented in a virtual city and verifies their existence in a real-time scene by matching keypoints between the virtual and real objects. In case of a stationary or variable object that does not exist in the virtual world due to incomplete pre-existing information, this method uses machine learning for object detection. Verified objects are then removed from the real-time image with a combined algorithm using contour detection and class activation map (CAM), which helps to enhance the efficiency and accuracy when recognizing moving objects.

# Dedication

*It is my deepest gratefulness and sincere regard that I dedicate this thesis work to my adored parents Mr. Pravin Patel and Mrs. Mittal Patel and the rest of my family and friends.*

# Acknowledgements

Throughout this thesis, I have received excellent support and guidance. Firstly, I would like to express my sincere gratitude to my advisor, Dr. Xiaobu Yuan, for his continuous support, motivation and immense knowledge. As a mentor, his direction supported me all the time to learn something new and to complete this research. I would like to thank my thesis committee members, Dr. Asish Mukhopadhyay and Dr. Myron Hlynka for their meaningful remarks and recommendations.

I am grateful to my family, whose love, motivation and positive thoughts strengthened me to reach closer to my ambition. They are the role models and the source of my inspiration. Last but not the least, I would like to thank all the faculties and staff of the School of Computer Science and my friends who encouraged me at the University of Windsor.

# Table of Contents


Declaration of Originality ..... ii

Abstract ..... iv

Dedication ..... v

Acknowledgements ..... vi

List of Figures ..... x

List of Abbreviations/Symbols ..... xiii

List of Tables ..... xiv

**Chapter 1: Introduction ..... 1**

1.1 Overview ..... 1

1.2 Google's Self Driving Car ..... 2

1.3 Need to aware of surroundings ..... 3

1.4 Object Detection ..... 4

1.5 Feature Detection and Selection ..... 5

1.6 Object Elimination ..... 5

**Chapter 2: Literature Review ..... 7**

2.1 Object Detection ..... 7

2.1.1 Machine Learning Approach ..... 8

2.1.2 Deep Learning Approach ..... 11

2.2 Feature Detection and Selection ..... 15

2.3 Object Verification ..... 18

2.4 Transfer Learning ..... 20

2.5 Class Activation Map (CAM) Generation and Usage ..... 22

2.6 Object Elimination ..... 24

2.6.1 Instance Segmentation ..... 24

2.6.2 Diminished Reality ..... 27

2.7 Related Work ..... 28

2.8 Thesis Statement ..... 30

2.8.1 Problem Statement ..... 30

2.8.2 Thesis Contribution ..... 31

**Chapter 3: Proposed Approach** ..... **32**
3.1 Motivation ..... 32
3.2 Working of the Overall System ..... 32
3.2.1 Working of Individual Modules ..... 33
3.3 Proposed Methodology for Static and Variable Object Verification and Object Elimination ..... 36
3.3.1 Static and Variable Object Verification ..... 36
3.3.1.1 Object detection module ..... 37
3.3.1.2 Feature Detection module ..... 37
3.3.1.3 Feature Selection module ..... 38
3.3.1.4 Object Verification module ..... 39
3.3.1.5 Object Verification algorithm ..... 39
3.3.2 Static and Variable Object Elimination ..... 41
3.3.2.1 Contour Detection Module ..... 42
3.3.2.2 Generation of Class Activation Map (CAM) ..... 42
3.3.2.3 Object Masking ..... 43
3.3.2.4 Object Elimination algorithm ..... 43
3.4 Time Complexity of the Proposed Approach ..... 44
**Chapter 4: Implementation and Experiments** ..... **45**
4.1 Software Information ..... 45
4.2 Construction of 3D Virtual World ..... 45
4.3 Creation of Repository of Virtual World ..... 46
4.4 Experiments and Results of Object Detection ..... 47
4.4.1 Experiments and Results of Object Detection for Buildings ..... 48
4.4.2 Experiments and Results of Object Detection for Tree ..... 48
4.4.3 Experiments and Results of Object Detection for Street light ..... 49
4.4.4 Experiments and Results of Object Detection for Bench ..... 50
4.5 Experiments and Results of Object Verification ..... 51
4.5.1 Centroid Detection of Real-time Static Object ..... 51
4.5.2 Feature Detection Using FAST corner Detector of Real-time Object ..... 52
4.5.3 Calculation of the Distance of Real-time Object ..... 52
4.5.4 Feature Selection in Real-time Image ..... 52
4.5.5 Retrieving the Virtual Image and Keypoints ..... 53

4.5.6 Object Verification of Static Object Using Prior Knowledge ..... 53
4.5.7 Increasing the Confidence Score Using Neighbour Object ..... 54
4.6 Experiments and Results of Object Elimination ..... 55
4.6.1 Results of Generation of Class Activation Map (CAM) ..... 55
4.6.2 Results of Contour Detection ..... 57
4.6.3 Results of Object Elimination Using Combined Approach ..... 60
4.7 Results Comparison and Discussion ..... 62
4.7.1 Advantages of the Proposed Approach ..... 62
4.8 Limitations of the Proposed Approach ..... 64
**Chapter 5: Conclusion and Future Work** ..... **66**
5.1 Conclusion ..... 66
5.2 Future Work ..... 67
**References/Bibliography** ..... **68**
**Vita Auctoris** ..... **82**

# List of Figures

Figure 1: Google's Self-Driving Car prototype ..... 2
Figure 2: Google's Autonomous Car driving on the road ..... 4
Figure 3: Dynamic Object Detection ..... 4
Figure 4: Detected features on a chessboard ..... 5
Figure 5: Object Masking using Mask-R-CNN ..... 6
Figure 6: Object Detection to identify different objects ..... 7
Figure 7: Machine Learning algorithm workflow ..... 8
Figure 8: Example of a CSVM network (Image source: Yakoub et al., [19]) ..... 9
Figure 9: AdaBoost HOG detector applied on a test image (Image Source: Arunmozhi et al., [24]) ..... 10
Figure 10: Deep Learning Approach Workflow ..... 11
Figure 11: The architecture of R-CNN (Image source: Girshick et al., [49]) ..... 12
Figure 12: The Architecture of Fast R-CNN (Image source: Girshick et al., [48]) 12
Figure 13: The architecture of Faster R-CNN (Image source: Ren et al., [47]) ..... 13
Figure 14: Illustration of YOLO (Image source: Redmon et al., [40]) ..... 14
Figure 15: object detection results (Image source: Tian et al., [42]) ..... 15
Figure 16: Different types of detected features ..... 16
Figure 17: Feature Selection workflow ..... 16
Figure 18: Harris-Stephens Corner Detection (Image source: Haggui et al., [63]) 18
Figure 19: Image matching based on different corner detection methods. a Harris method, b the method in [75] ..... 18
Figure 20: Overall system overview of DCNN approach (Image source: Chen et al., [78]) ..... 19
Figure 21: Training the entire model Vs Transfer Learning ..... 20
Figure 22: Three strategies for fine-tuning ..... 21
Figure 23: Generated CAM for different breeds of dog using CNN ..... 22
Figure 24: CCAM applied to localize common objects (Image source: Li et al., [108]) ..... 23
Figure 25: Example of Instance segmentation ..... 25

Figure 26: Segmented results: a. initial contours and local region; b. Final evolved contours; c. segmented result (Image source: Lu et al. [109]) ..... 25
Figure 27: Mask R-CNN framework for instance segmentation (Image source: He et al., [115]) ..... 26
Figure 28: Concept of Diminished Reality ..... 27
Figure 29: Diminished reality techniques: a. diminish, b. Seeing through, c. replacing, d. inpainting (Image source: Mori et al., [12]) ..... 28
Figure 30: Overall system architecture ..... 33
Figure 31: Architecture of the proposed approach ..... 36
Figure 32: Flowchart of object verification component ..... 37
Figure 33: Pixel P selected as a corner point ..... 38
Figure 34: Flowchart of object elimination component ..... 41
Figure 35: Constructed 3D virtual World ..... 46
Figure 36: Constructed 3D Objects ..... 46
Figure 37: Rendered 3D object ..... 47
Figure 38: Extracted 3D features on rendered images ..... 47
Figure 39: Detected Skyscraper, Street light, Tree in the real-time image ..... 48
Figure 40: Detected Skyscraper, Tree in the real-time image ..... 48
Figure 41: Detection of Tree in the image ..... 49
Figure 42: Detected Trees, Skyscraper in the real-time image ..... 49
Figure 43: Street light, Trees detected in the test image ..... 50
Figure 44: Street light, Trees detected in the test image ..... 50
Figure 45: Bench detected in the test image ..... 51
Figure 46: Centroid Calculation ..... 51
Figure 47: Detected centroid in the real-time image with coordinates ..... 51
Figure 48: Detection of corner points using the FAST algorithm ..... 52
Figure 49: Detected corner points on real-time image ..... 52
Figure 50: Calculating the distance between keypoints and centroid ..... 52
Figure 51: Selected top 8 keypoints for top-right (1st quadrant) ..... 53
Figure 52: Plotted selected keypoints on the real-time image ..... 53
Figure 53: Plotted selected keypoints on the virtual image ..... 53

Figure 54: Object verification result of real-time object .....54
Figure 55: Selected keypoints of the virtual nearest static object (e.g. building) ...54
Figure 56: Selected keypoints of the real-time nearest static object (e.g. building) .....55
Figure 57: Verification result of neighbour stationary object (e.g. building) .....55
Figure 58: Increased confidence score of a verified static object .....55
Figure 59: Result of CAM Generation of Tree (left: input image, centre: generated heatmap, right: heatmap superimposed on input image) .....56
Figure 60: Result of CAM Generation of Street light (left: input image, centre: generated heatmap, right: heatmap superimposed on input image).....57
Figure 61: Result of CAM Generation of Street light (left: input image, centre: generated heatmap, right: heatmap superimposed on input image).....57
Figure 62: Result of Contour Detection of Trees.....58
Figure 63: Result of Contour Detection of Street light.....59
Figure 64: Result of Contour Detection of Bench .....59
Figure 65: Result of Contour Detection of Buildings.....60
Figure 66: Result of Object Elimination of Trees.....60
Figure 67: Result of Object Elimination of Street light.....61
Figure 68: Result of Object Elimination of Bench .....61
Figure 69: Result of Object Elimination of Buildings.....61
Figure 70: Result of the proposed algorithm .....62
Figure 71: Person Detected in the poster using RetinaNet model .....63
Figure 72: Reflection of car and person on the building detected as a dynamic object.....63
Figure 73: Detection of dynamic objects after object elimination.....64
Figure 74: Generated CAM for trees without leaves.....65

# List of Abbreviations/Symbols

| | |
|---|---|
| LiDAR | Light Detection and Ranging |
| RADAR | Radio Detection and Ranging |
| GPS | Global Positioning System |
| 3D | 3-dimensional |
| 2D | 2-dimensional |
| CNN | Convolutional Neural Network |
| ANN | Artificial Neural Network |
| R-CNN | Region Based Convolutional Neural Network |
| YOLO | You Only Load Once |
| ROI | Region of Interest |
| FCN | Fully Convolutional Network |
| SIFT | Scale Invariant Feature Transformation |
| ORB | Oriented FAST and Rotated BRIEF |
| BRIEF | Binary Robust Independent Elementary Features |
| SOA | Service Oriented Architecture |
| SURF | Speeded Up Robust Features |
| FAST | Features from Accelerated Segment Test |
| CAM | Class Activation Map |
| VGG | Visual Geometry Group |

# List of Tables

Table 1: Related Work ..... 30
Table 2: Time complexity of the proposed algorithm ..... 44
Table 3: List of software and tools used ..... 45

# Chapter 1: Introduction

## 1.1 Overview

With the aid of significant technologies and researches, now the vision of self-driving car on the road has become a possibility to run on the road. It is not a jaw-dropping concept as research in this direction has been carried out for years. Seemingly within just a few years, autonomous cars have gone from science fiction fantasy to road-bound reality [1]. Apart from many automobile giants like Tesla, GMC, Uber, and Mercedes-Benz, there are many other technology corporations like Google, Apple, IBM, and Intel that have infused billions of dollars in this research and development to turn this arduous vision of a fully autonomous car into an actuality. Nowadays, self-driving cars are being tested on the road, but they are far away from being feasible [2]. The design of such an advanced machine involves immense expertise to ensure smooth and safe driving.

While an autonomous car operates on the road, the knowledge of surroundings that consist of various objects mainly differentiated according to the orientations and movement should be taken into consideration. When a 3D virtual world is constructed with representations of static and variable objects in the real world, it helps the autonomous car to be familiar with the surroundings while driving. Darms et al. [3] and Hu et al. [4] have described *static objects* as those that do not move during the operation of the car. The list includes buildings and other roadside objects like benches, trees, and light pole. Fu, Kun, et al. have used multiple class activation map to extract discriminative parts of aircraft of different categories [5]. Verified objects are masked out in the real-time images.

The new method of this thesis makes use of a constructed virtual environment that works as prior knowledge to outperform various tasks such as object verification and elimination. The existence of the real objects is verified by matching feature points of physical objects with virtual objects. These verified objects are abolished

using the combined approach of contour detection and class activation map (CAM). This research aims to use pre-existing information to verify and eliminate stationary and variable objects from real-time scenes to allow an autonomous car to pay attention to moving objects like pedestrians and cars in order to drive safely.

### 1.2 Google's Self Driving Car

Many auto-giants companies like Tesla, Mercedes, Uber, Google, BMW, and Volvo have already developed their semi-autonomous cars on the market. Zhao et al. explained the key technology of a self-driving car [6]. Despite using exclusive hardware, autonomous vehicles are still far from being fully automatic. Google has started constructing a self-driving car – *waymo* almost ten years ago. In 2015, Google provided "the world's first fully driverless ride on public roads" to a legally blind friend of principal engineer Nathaniel Fairfield [7]. Figure 1 displays the Google's autonomous car. Much tech-savvy hardware is being used to improve visibility, to measure the distance to other objects, and to fetch geolocation information. Although some of the hardware are expensive, they are necessary as these sensors assist in detecting objects for safe driving. The rotating roof-top LiDAR is considered as the heart for object detection. The LiDAR is used to measure the distance to other objects to build the 3D map in order to see obstacles. The bumper-mounted radar is responsible for measuring the distance to vehicles in front and behind the car. Rear-mounted aerial receives geolocation information from GPS satellites, and ultrasonic sensors attached to one of the rear wheels monitors the car's movement. These devices are necessary for the safe operation of autonomous cars.

*Figure 1: Google's Self-Driving Car prototype*

### 1.3 Need to Aware of Surroundings

Situational awareness is the most essential key to safe driving. Drivers should be conscious of their location and the surroundings to navigate the car at the desired location [128]. The same objective has been applied for an autonomous vehicle to function on the road by recognizing objects and obstacles. When an autonomous car is running on the road, it must be responsive to the surroundings for safe moving. Sensor technologies including GPS provide information about the surrounding environment. These sensors gather data to narrate the change in the position and orientation of the car. They constantly pass on the information about surroundings like the position of pedestrians, and other objects near the car to the system in order to navigate smoothly.

There are mainly three categories of objects a car may come in the contact while driving:

- Stationary objects: Those objects that are static at the same location with the same pose. (e.g. Buildings, Bench, Street light)
- Variable objects: Those objects that stay stable at the same place, but a pose may vary (e.g. trees)
- Dynamic objects: Those object that may change the location and pose (e.g. human, animal, car)

Among all objects, dynamic objects create more danger to a self-driving car. These objects need to be detected. Darms et al. [3] and Hu et al. [4] have described *dynamic objects* as those potentially move during the observation periods. Figure 2 shows how an autonomous car observes the surroundings while driving with the aid of sensors to function safely and smoothly.

*Figure 2: Google's Autonomous Car driving on the road*

### 1.4 Object Detection

Sensors attached to the car are responsible to estimate distance and steer clear of collision with obstacles. The nearby objects are identified using various machine learning algorithms of object detection. Druzhkov et al. [8] published a survey of deep learning techniques for object detection and image classification. A domain shift framework based on image-style level and instance-level algorithm based on Faster- RCNN has been used for object detection [9]. Object classification has been performed using a fusion of CNN and light detection and ranging (LIDAR) for an autonomous vehicle [10]. Figure 3 depicts the object detection task where dynamic objects are located and displayed using bounding boxes with the respective class label and probability score. The proposed approach detects roadside stationary (e.g. buildings, street light) and variable (e.g. trees) objects using Faster-RCNN technique. The model has been trained to detect an instance of an object in the real-time scene.

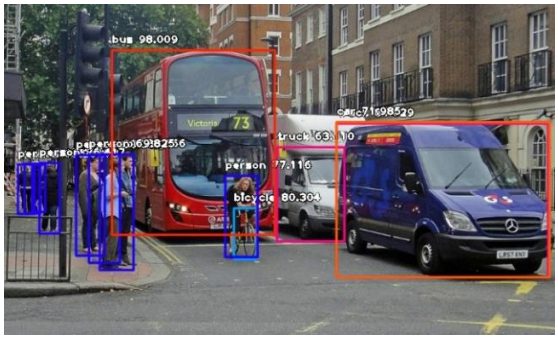



*Figure 3: Dynamic Object Detection*

### 1.5 Feature Detection and Selection

Feature plays a significant role in various image-based applications. It can be expounded as a vector to define an object or its accompaniment. Features are specific structure in the image such as edges, corners, blobs, and points. Different feature detection algorithms are Harris Corner detection, Shi-Tomasi Corner detector and Good Feature Track, SIFT, SURF, FAST algorithm for corner detection, BRIEF, and ORB. The survey of feature detection algorithm is presented by Li et al. [11]. The paper also presents mathematical models of algorithms. Performance comparison of various algorithms is presented based on accuracy, speed, scale invariance, and rotation invariance. Figure 4 shows the detected corner points using Harris corner detection algorithm. The proposed approach detects corner keypoints using feature detector and the most relevant feature points are selected using the mathematical approach.

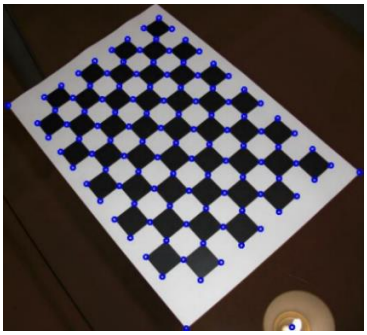


*Figure 4: Detected features on a chessboard*

### 1.6 Object Elimination

A part of a scene can be hidden as they were behind an invisible object using masking technologies. Object elimination can be implemented for various tasks like background removal, foreground subtraction, object detection, instance segmentation. Multiple approaches to perform elimination include contour detection, Mask-RCNN, diminished reality. A survey of different diminished reality techniques has been published by Mori et al. [12]. Yang et al. [13] introduced a contour detection framework using fully convolutional Encoder-Decoder Network for object detection and masking. Figure 5 shows the objects that are masked using

Mask-RCNN algorithm. The proposed method uses a combined approach of contour detection and class activation map for object elimination.

*Figure 5: Object Masking using Mask-R-CNN*

This research work is the primary approach for stationary (e.g. buildings, street light, benches) and variable (e.g. Trees) object verification and object elimination in real-time images giving the insights for an autonomous car to navigate smoothly on the road. The proposed approach performs verification task by matching interest points extracted from a real-world object with a virtual world object. Once the object is verified successfully, the removal of static and variable objects in the real-time scene is executed via the fusion method of contour detection and Class Activation Map (CAM) to achieve more accuracy.

In this thesis, Chapter 2 entails a review of the previous work done on object detection and object elimination for autonomous vehicles. It also caters to the criterion techniques for the proposed approach. Chapter 3 describes the proposed system of use of prior data for object verification and elimination in the real-time scene in depth. Chapter 4 demonstrates a detailed explanation of the proposed approach using experimental results and presents comparison with other related work. Chapter 5 contains the conclusion of the thesis together with possible future work.

# Chapter 2: Literature Review

This chapter provides a survey of the relevant background of recent works in object detection and object elimination using 2D and 3D images. It also covers use of prior knowledge to help improving the performance of object verification and object elimination.

## 2.1 Object Detection

Object detection is related to computer vision and image processing that locates instances of objects of a certain class in still images and videos. Object detection has many applications in computer vision area such as face detection, image retrieval, and pedestrian recognition. Figure 6 below displays an illustration of object detection algorithm to identify roadside objects (e.g. car, traffic light, truck) [14]. Zou et al. [15] discussed a survey of object detection in the last 20 years. This survey covers milestone detectors in history, detection datasets, metrics, fundamental building blocks of the recognition system, speed up techniques, and the recent state of the art detection methods. It also reviews some important identification applications, such as pedestrian detection, face detection, and text detection, etc., and makes an in-deep analysis of their challenges as well as technical improvements in recent years.

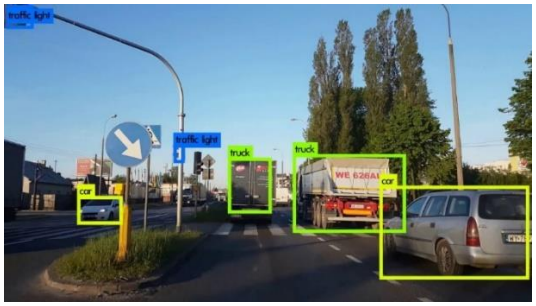



*Figure 6: Object Detection to identify different objects*

Methods of object detection mainly fall into either Machine Learning-based approaches or Deep Learning-based approaches.

### 2.1.1 Machine Learning Approach

In machine learning approaches, it is necessary to first define features using any one method listed below. Then classification is performed on the detected features using a technique such as Support Vector Machine (SVM), and Random Forest (RF) [69].

- Viola-Jones Object detection framework based on Haar features
- Scale-Invariant Feature Transform (SIFT)
- Histogram of Oriented Gradients (HOG) features

Erickson et al. [16] published a paper that uses machine learning for medical imaging. They have also reviewed different classification such as Naïve Bayes, Support Vector Machine, Neural Networks, k-Nearest Neighbours, Deep Learning, and Decision Tree to select suitable features. Figure 7 below portrays the workflow of general machine learning algorithms.

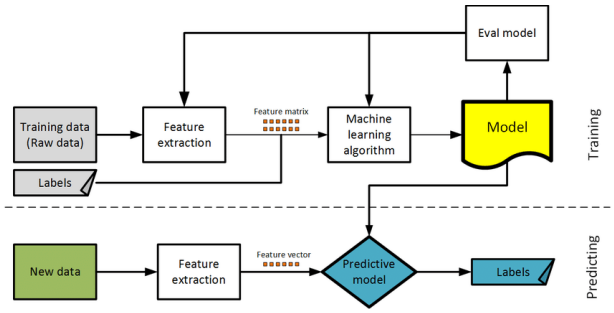



*Figure 7: Machine Learning algorithm workflow*

Lei et al. [17] analyzed the feature selection techniques for object-based classification of unmanned aerial vehicle imagery. Their work specifically emphases on assessing the effect of feature dimensionality and training the set size using SVM and RF classifiers to achieve different feature selection methods, including the filter method, wrappers, and embedded methods. Bakhshipour et al. [18] illustrated an algorithm for weed detection based on their pattern by applying support vector machine and artificial neural network. Their work also involves the use of shape features such as Fourier descriptors and moment invariant features. The classification has been carried out using SVM and ANN.

An approach of Convolutional SVM Network was applied for object detection in UAV Imagery [19]. The CSVM network is based on several alternating convolutional and reduction layers ended by a linear SVM classification layer. The convolutional layers in CSVM rely on a set of linear SVMs as filter banks for feature map generation. During the learning phase, the weights of the SVM filters are estimated through a forward supervised learning strategy unlike the backpropagation algorithm widely used in standard convolutional neural networks (CNNs) [19]. Figure 8 below illustrates the architecture of the Convolutional SVM algorithm.

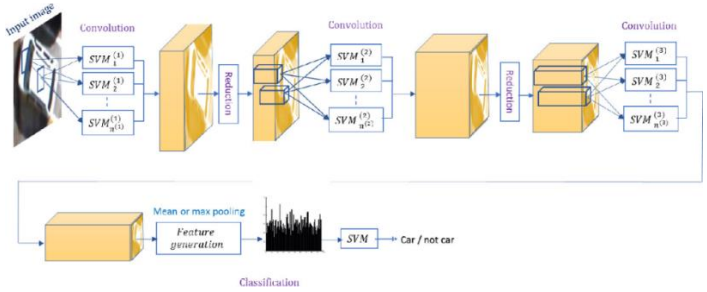



*Figure 8: Example of a CSVM network (Image source: Yakoub et al., [19])*

Wei et al. [20] designed an approach for multi-vehicle detection by combining Haar and HOG features. This algorithm makes full use of HOG characteristics and uses its good descriptive ability to describe target vehicles whereas Harr features are utilized to extract the prospect region of interest (ROI). Moreover, the obtained HOG features from the ROI target area can be selected by applying the cascade structured AdaBoost classifier features and target area classification. The precise target can be further detected by a Support Vector Machine (SVM) [20]. Chee et al. [21] proposed an algorithm for Pedestrian detection through the fusion of image gradient and magnitude properties extracted using Histogram of Oriented Gradient (HOG) and Histogram of Magnitude (HOM) features.

An automatic method for the recognition of individual oil palm trees using images from unmanned aerial vehicles (UAVs) was developed by Wang et al. [22]. First, using a support vector machine (SVM) classifier, UAVs images are categorized between vegetation and non-vegetation. Then, a feature descriptor based on the histogram of oriented gradient (HOG) has been designed for palm trees to extract

features for machine learning. Finally, SVM classifier has been trained and optimized using the HOG features from positive (i.e., oil palm trees) and negative samples (i.e., objects other than oil palm trees) [22]. An approach for object detection and classification was published by Rashid et al. [23] that uses a merged strategy of deep convolutional neural network and SIFT point features. Firstly, an improved saliency method is implemented, and the point features are obtained. Then, DCNN features are extracted from two deep CNN models like VGG and AlexNet. Thereafter, Reyni entropy-controlled method is executed on DCNN pooling and the SIFT point matrix for robust feature selection. Finally, the selected robust features are fused in a matrix by a serial approach, that is later fed to ensemble classifier for recognition [23]. An analysis of three commonly used strategies, Histogram of Oriented Gradients (HOG), Haar-like features and Local Binary Pattern (LBP) for object detection is investigated using a public dataset in [26]. Figure 9 below demonstrations the result of AdaBoost HOG detector on a test image.

A novel algorithm on a mobile system that can notify drivers about the possibility of collision with pedestrians was developed [24]. The partial Haar transform and HOG are fused for pedestrian detection.

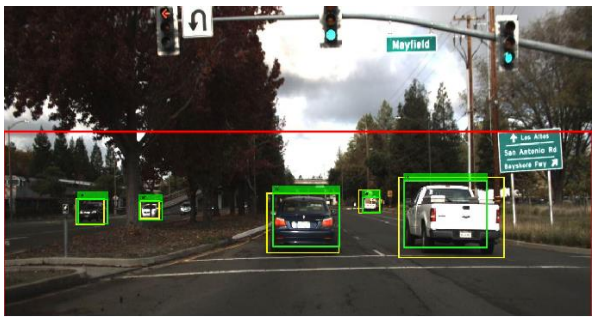



*Figure 9: AdaBoost HOG detector applied on a test image (Image Source: Arunmozhi et al., [24])*

Prasanna et al. [25] built an approach for human tracking system using joint Haar-like and HOG features where Haar characteristics are used for the object's structure and the HOG features for the edge. A set of mixed features is developed with these two features. Using Online Boosting, feature selection is performed to create a set

of robust features. Finally, with the help of SVM classifier, classification is executed.

### 2.1.2 Deep Learning Approach

In the Deep Learning Approaches, the algorithms are capable of performing end-to-end object localization task without defining any features and are based on Convolutional Neural Network (CNN) [69]. Deep Learning approaches are as follows:

- Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
- Single Shot MultiBox Detector (SSD)
- You Only Look Once (YOLO)

Brunetti et al. [27] published a survey on computer vision and object detection methodologies for pedestrian detection and tracking. Panchpor et al. [29] and Pouyanfar et al. [31] published a study on various object detection algorithms using deep learning. Arnold et al. [28] reviewed a survey on 3D object detection methods that utilize sensors and datasets for autonomous driving. 3D object identification technique introduces a third dimension that reveals the object's size and location information useful for path planning, collision avoidance, and so on [28]. Liu et al. [30] submitted a survey on advanced techniques for generic object detection. Sindagiet al. [32] published a survey of the latest approaches of CNN for single image-based crowd counting.

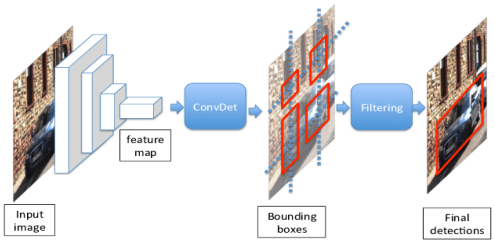



*Figure 10: Deep Learning Approach Workflow*

Girshick et al. [49] introduced an approach named *R-CNN: Regions with CNN features* for accurate object detection and semantic segmentation. Figure 10

illustrates the workflow of deep learning approach. The input to this approach is test image that locates the object using bounding boxes. Figure 11 shows the architecture of R-CNN.

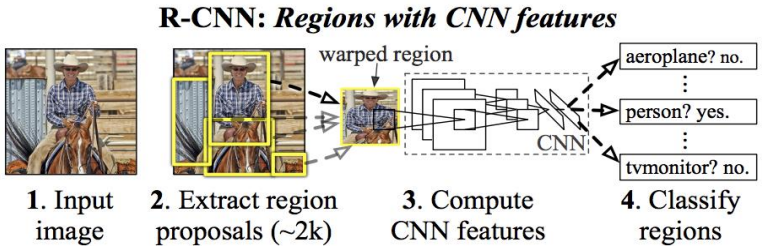



*Figure 11: The architecture of R-CNN (Image source: Girshick et al., [49])*

The algorithm uses a selective search to extract 2000 regions from the input image that are wrapped into a square and fed into CNN to produce 4096-dimensional feature vector. The CNN works as a feature extractor and the output dense layer consists of the extracted features that are forwarded to SVM classifier to identify the presence of an object within the bounding box. The problem with R-CNN is that it takes 47 seconds to generate an output. Also, it requires much time for training as 2000 regions for each image need to be classified.

Girshick et al. [48] developed an enhanced approach of R-CNN, i.e., Fast R-CNN algorithm. Fast R-CNN requires less time to generate an output than simple R-CNN. The architecture of Fast R-CNN is shown in Figure 12.

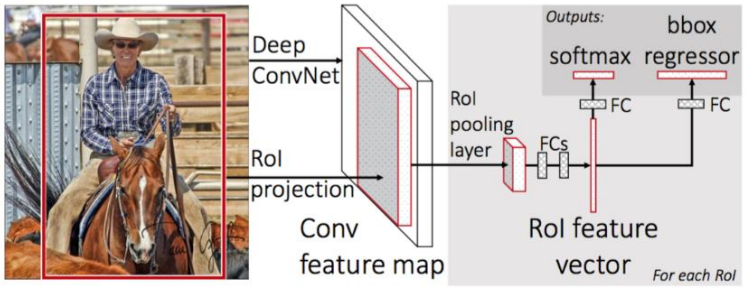



*Figure 12: The Architecture of Fast R-CNN (Image source: Girshick et al., [48])*

According to Fast R-CNN algorithm, the input image is directly fed into CNN to produce a convolutional feature map and by using them, the regions of proposals are

obtained. Then using ROI pooling layer, all the proposed regions are reshaped into a fixed size to be fed into a fully connected layer. The corresponding class label of the proposed region and offset value of the bounding box are predicted using soft-max layer. The Fast R-CNN is faster than R-CNN and the convolution operation is performed only once per image and feature map is generated from it.

Ren et al. [47] built an improved approach of Fast R-CNN called Faster R-CNN. Figure 13 depicts the architecture of Faster R-CNN.

In Faster R-CNN, the input image is supplied to CNN to create a convolutional feature map. Instead of using a selective search algorithm, a separate network is used to predict the region proposals. The expected region proposals are reshaped into a fixed size using ROI layer to classify the image within the proposed region and estimate the offset values for the bounding boxes using soft-max layer.

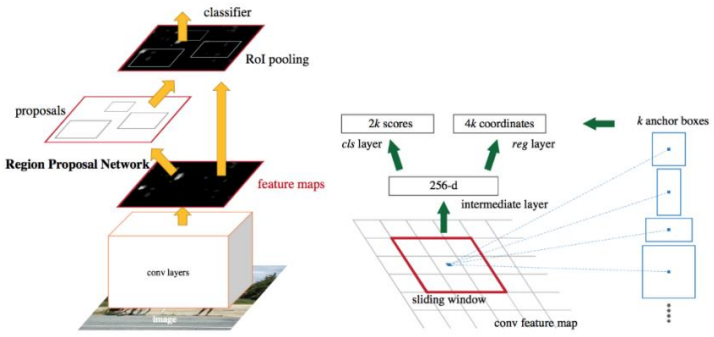



*Figure 13: The architecture of Faster R-CNN (Image source: Ren et al., [47])*

Neumann et al. [44] published a fully annotated dataset including tracking information for pedestrian detection and tracking at night. The scene is recorded taking advantage of industry-standard camera including different sensors and weather conditions. Sheng et al. [46] proposed an approach for vehicle area detection and vehicle brand classification using RCNN, Faster R-CNN, AlexNet, ResNet, VGGNet, and GoogLeNet.

Redmon et al. [40] designed YOLO (You Only Look Once) algorithm for object detection. Figure 14 shows the illustration of YOLO algorithm.

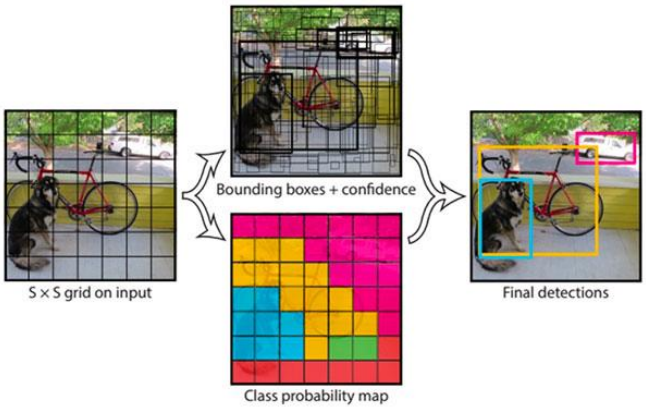



*Figure 14: Illustration of YOLO (Image source: Redmon et al., [40])*

In YOLO, a single neural network predicts the corresponding class probability and bounding box. First, the image is split into SxS grid where each grid cell predicts only single object, containing m bounding box and for each bounding box, the class probability and offset value are evaluated against a pre-set threshold value to locate the object.

Verma et al. [33] illustrated an algorithm with a fusion of monocular camera and a 2D Lidar for vehicle detection using YOLO. Possatti et al. [34] proposed to integrate the power of deep learning-based detection using YOLO with the prior maps utilized by car platform IARA (Acronym for Intelligent Autonomous Robotic Automobile) to recognize the relevant traffic lights of predefined routes. Mittal et al. [35] explained the object detection and classification tasks using YOLO algorithm. Putra et al. [36] developed an improved approach of YOLO for human and car recognition. Wang et al. [37], Zhang et al. [38], and Zhang et al. [39] used SSD approach for object detection. Chowdhury et al. [45] designed Faster R-CNN and SSD approach for pedestrian intention detection.

Židek et al. [41] built a method for object detection using deep learning technique trained by 3D virtual models. The CNN model is trained using 2D samples generated automatically from the 3D virtual models. Loing et al. [43] designed a methodology for localization without using the single real-time image by utilizing only 3D models of the robot and object for training the network. Tian et al. [42] built a virtual dataset

named ParallelEye. Faster R-CNN and DPM networks are trained via fusing ParallelEye virtual dataset with a real-time image dataset for object detection in real-time view. Figure 15 shows the results of object detection using two different models. Top row images are detected using a model purely trained on the real-time images whereas bottom row objects are detected using a network, trained using combined virtual and real-time scenes.

*Figure 15: object detection results (Image source: Tian et al., [42])*

**2.2 Feature Detection and Selection**

In computer vision and image processing, feature plays a vital role that can be described as a prominent characteristic of an image. They are benefited to execute certain tasks such as feature selection, feature matching, object recognition and so on. Features represent the specific structure in the image such as edges, centroid, blobs, and points. Berger et al. [72] explained the feature and the actual feature usage in the industry.

Features are identified using detectors from the image. Feature detection is the process of finding image features or keypoints of a given type at each pixel that are somehow special in the image. Several detectors for features detection are as follows:

- Harris corner detection
- Shi- Tomasi corner detection algorithm
- FAST (Features from Accelerated Segment Test)
- Laplacian of Gaussian

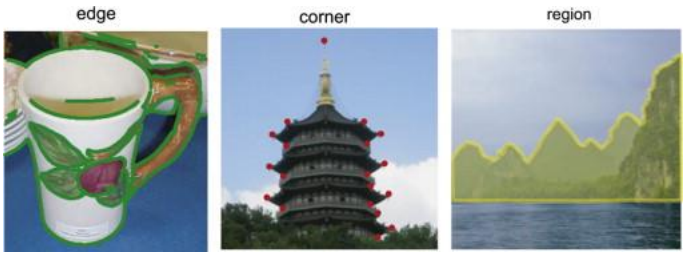



*Figure 16: Different types of detected features*

Feature extraction is the method of computing a descriptor from the pixel around each interest point using feature descriptors such as SURF, HOG, and FREAK. Feature starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction [76].

In computer vision, feature selection is expounded as a process for creating the new subset of essential features to enhance generalization by reducing overfitting, and for dimensionality reduction. Common names for feature selection are variable selection, attribute selection, or variable subset selection. Feature selection differs to feature extraction by returning the subset of selected features, whereas feature extraction defines new features from the function of the original features. Figure 17 portrays the workflow of the feature selection technique.

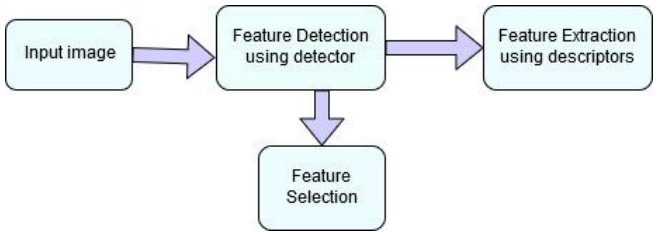



*Figure 17: Feature Selection workflow*

Feng et al. [73] designed a method for feature detection and matching using Harris corner detection algorithm. Making use of Harris corner detection algorithm, features are obtained. Rotation invariant Feature Descriptor (RFID) is used to represent the feature point information. Wang et al. [50] reviewed the corner

detection algorithms proposed in the last four decades. Corner detection algorithms can be divided into intensity-based, contour-based, and model-based methods. Intensity-based frameworks are based on measuring local intensity variation of the image. Contour-based methods identify corners by analyzing the shape of edge contour. Model-based algorithms extract corners by fitting the local image into a predefined model [50]. Karim et al. [51] presented a study on the comparison of feature extraction techniques by combining SURF with FAST and BRISK, followed by feature matching. Al-Rawabdeh et al. [53] submitted a review on the performance of FAST-9 and FAST-12 as well as the Harris detector in terms of the repeatability rate, completeness, and correctness under different threshold values for UAV object localization. Hore et al. [55] analyzed the performance of SIFT and SURF feature descriptors in different circumstances such as rotational effect, scaling effect, illumination effect, and blurring effect to achieve object recognition. Kabir et al. [64] presented a comparison of four feature detection methods for the modern and old buildings, including Canny edge detection, Hough line transform, Find Contours, and Harris Corner. A comparison of four feature detection approaches; Harris, SURF, FAST, and FREAK, is published by Ghosh et al. [67] for image mosaicing.

DeTone et al. [52] built a self-supervised approach for training feature detectors and descriptors to make them suitable for a large number of multi-view geometry problems. Gao et al. [56] developed a novel method utilizing shadows that automatically extracts building samples and verifies buildings accurately to enhance automation and accuracy. Liu et al. [59] presented an automatic methodology for building area extraction from optical high-resolution imagery using the newly developed morphological building index (MBI). The new FPGA (Field Programmable Gate Array) architecture for reuse of sub-image data was introduced in [54]. In the proposed architecture, a remainder-based approach is firstly designed for reading the sub-image and a fusion of FAST and BRIEF (Binary Robust Independent Elementary Features) descriptors is used for corner detection and matching. Karami et al. [65] analyzed the performance of the SIFT matching

algorithm against various image distortions such as rotation, scaling, fisheye, and motion distortion.

Zhao et al. [61] built a method to estimate the height of the building using both corner points and roofline. Haggui et al. [63] developed an approach using Harris corner detector for NUMA manycore recognition. Figure 18 shows the Harris corner identification on NUMA manycore.

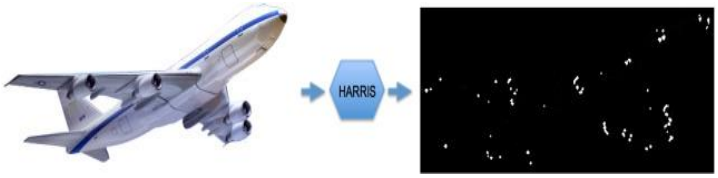



*Figure 18: Harris-Stephens Corner Detection (Image source: Haggui et al., [63])*

Wu et al. [57] proposed and implemented Deep Validation, a novel framework for real-world error-inducing corner detection using DNN-based system. A novel framework for corner detection and tracking for the real-time scene was presented in [58]. Ghandour et al. [60] designed Building Detection with Shadow Verification (BDSV) for building localization using the shadow, shape, and color features of buildings. Hu et al. [62] illustrated a non-interactive approach based on binary feature classification for building area recognition and building contours extraction from aerial images.

Infrared image matching using SUSAN corner detection was introduced in [75]. Figure 19 displays the comparative result of the proposed approach in [75] with Harris Corner Detection.

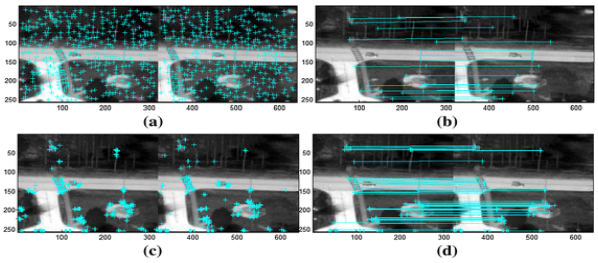



*Figure 19: Image matching based on different corner detection methods. a Harris method, b the method in [75]*

## 2.3 Object Verification

Verification has been implemented primarily for fingerprint, iris, and face matching. Chen et al. [78] proposed an approach for unconstrained face verification practicing Deep CNN features, trained using the CASIAWebFace dataset and the performance was evaluated on both IJB-A and LFW datasets. Figure 20 illustrates the overall system architecture of the proposed DNN framework for face verification in [78].

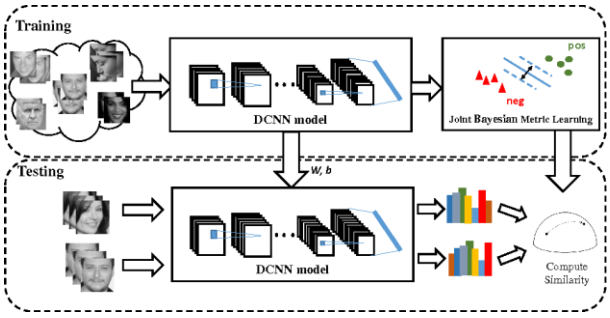



*Figure 20: Overall system overview of DCNN approach (Image source: Chen et al., [78])*

An approach for video-based unconstrained face verification and recognition was discussed in [79]. Crosswhite et al. [80] proposed a template adaptation method for face verification and identification. Their work also proved that it can be applied to existing state-of-art methods for enriched performance. Zhang et al. [82] evaluated a method for animal object detection and segmentation from wildlife monitoring videos captured by motion-triggered cameras, called camera-traps. First, using multilevel graph cut, animal object region proposals are generated in the spatiotemporal domain. Then using developed a cross-frame temporal patch verification method, these region proposals are determined if they are true animals or background patches.

Hsu et al. [83] developed an architecture for vehicle verification between two nonoverlapped views using sparse representation. Karami et al. [66] published a survey of image matching technologies- SIFT, SURF, BRIEF, and ORB. Taira et al. [84] built a system for indoor object localization that estimates the 6DoF camera pose and validates a query image with respect to a 3D indoor map. Yuan et al. [85]

proposed an approach for retina verification based on Structural Similarity (SSIM) to verify using similarity score. Kavitha et al. [86] designed an approach for a secured voting system using face, iris, and fingerprint verification. Qin et al. [87] suggested a deep learning-based segmentation methodology for finger-vein verification by training CNN to extract the vein patterns from any image regions and to estimate the probability of pixels to check if they belong to the vein or the background. They also made use of FCN to recover missing finger vein shapes for amended performance.

### 2.4 Transfer Learning

In machine learning, transfer learning is a method that makes use of knowledge gained while solving one problem to apply it for a different but related problem. For instance, knowledge obtained to recognize a cat can be applied for dog recognition. Some widely used pre-trained models are:

- VGG
- InceptionV3
- ResNet5

Figure 21 below illustrates the difference between training the entire model and using transfer learning.

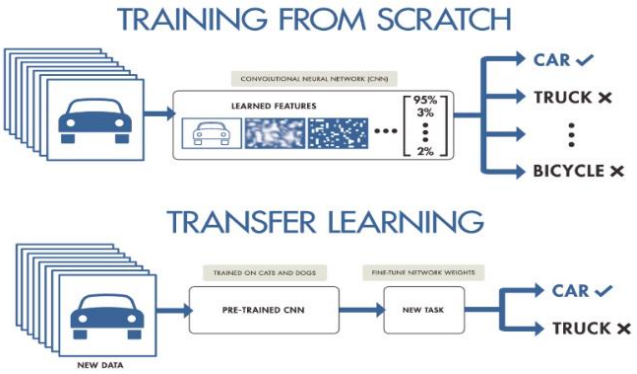



*Figure 21: Training the entire model Vs Transfer Learning*

The pre-trained models are primarily trained on the large dataset and, for a new similar dataset, the pre-trained model weights can be used for extracting the features. At the time of applying the pre-trained model on the new dataset, the original

classifier of the model has been removed and new classifier has been added to perform a defined task and later fine-tuned using any one of the following strategies:

1. Train the entire model: Use the architecture of the pre-trained model and train it according to the dataset from scratch.
2. Train some layers and leave the others frozen: For the small dataset, freeze more layers to prevent overfitting whereas for a large dataset, train more layers.
3. Freeze the convolutional base: The convolutional layer is set in its original form and its output is used for the classification task.

Figure 22 presents these 3 strategies of fine-tuning in a graphical way.

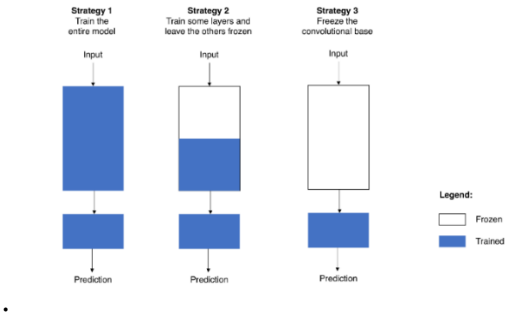



*Figure 22: Three strategies for fine-tuning*

Weiss et al. [97] reviewed the transfer learning methodology and its applications. Huh et al. [96] proposed an approach where they used pre-trained CNN features on various subsets of the ImageNet dataset and evaluated transfer performance on a variety of standard vision tasks. Yuan et al. [91] presented a learning-based framework for shadow removal using an online learning strategy and fine-tuned with the automatically identified examples in the new videos. Wang et al. [94] developed an architecture for ship detection via fusion of single shot multiBox detector (SSD) and transfer learning. A system for end-to-end airplane detection in remote sensing images using transfer learning approach is presented in [92]. Kapur et al. [90] used transfer learning for object detection in real-time video. A deep learning-based framework for detection and classification of breast cancer is proposed in [95].

Mohamed et al. [88] presented a work on applications of transfer learning for object detection. Yabuki et al. [89] and Singh et al. [93] designed a method for object detection using transfer learning based on CNN with feature extractor technique.

### 2.5 Class Activation Map (CAM) Generation and Usage

A Class Activation Map (CAM) is a technique for producing discriminative image regions used by CNN to identify defined class in the input image. CAM allows us to observe at which image regions CNN is looking and appropriate to a specific class. Zhou et al [98] proposed a framework of re-using the trained classifier for getting good localization of distinct class, without having bounding box coordinates information. Figure 23 presents the generated class activation map for different breeds of dog using CNN.

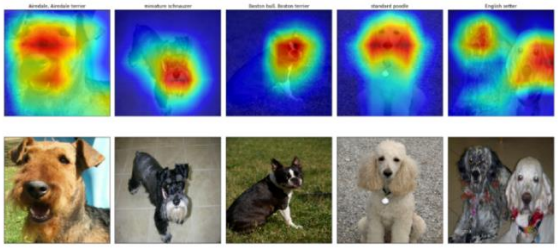


*Figure 23: Generated CAM for different breeds of dog using CNN*

CAM delivers certain assurance that the model has correctly learned the distinctive features between multiple categories in the form of visualization. Moreover, it conveys us to see what features are guiding the model's decision to classify various objects in the input image that are used by the model to make a prediction.

These means are pursued to generate class activation map:

1. The pre-trained network is used and most of its weights are freezed
2. The model is modified and fine-tuned to generate CAM output
3. Classifier is trained
4. The last convolutional layer is used to create CAM output
5. Generated CAM is displayed

To generate CAM, the network architecture is limited to have a global average pooling layer after the last convolutional layer, followed by the dense layer. To get this model structure, the network is modified and fine-tuned to get CAM.

In detail, the first building block for this layer is a convolutional layer that produces an output shape of in terms of batch size, number of filters, width, height. The output from the GAP layer is treated by the dense layer and softmax layer to assign a weight to each of the categories that set the importance of each the convolutional layer output. To generate CAM, output images from the convolutional layer are multiplied by their assigned weights and added. By superimposing the class activation map on input image allows us to identify the most essential image regions to the specific category.

Kwaśniewska et al. [99] demonstrate a method of face detection from low-resolution thermal images and the most relevant area is highlighted. Tang et al. [101] proposed a deep discriminative map network for visual tracking. The system utilizes two neural networks for positioning and size change estimation. Guo et al. [103] developed a methodology to recognize human attributes without the detection of a body part and the prior correspondence between body parts and attributes with the help of CAM network. Li et al. [105] showed a framework for remote sensing image scene classification using CAM. The attention map is generated using a pre-trained network as priors for the classification task that are used as an explicit input to end-to-end training for the first time, aiming to force the network to focus more on the most appropriate parts. Li et al. [108] used CAM to localize common objects in an input image. Figure 24 displays the result of produced CAM to locate common objects.

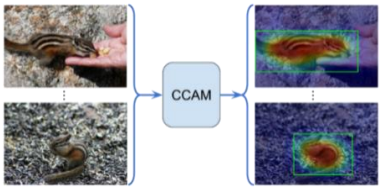



*Figure 24: CCAM applied to localize common objects (Image source: Li et al., [108])*

Charuchinda et al. [106] introduced a technique to build an image classification network using class activation map (CAM) to identify whether each sub-image contains the class of interest. The output of the CAM is the filter response where pixels with high probability are likely to belong to the class of interest. Vasu et al. [107] made use of class activation map to obtain a view of deep network's perception of aerial imagery and identify salient local regions. Moreover, the concept of transfer-learning is involved to train the model on a similar dataset. Fu et al. [13] applied a methodology of multi-class activation map for recognition of aircraft in the remote sensing images.

Pericherla et al. [100] designed an approach to reduce the L2 distance i.e., Euclidean distance between produced adversarial images and the original images using class activation map. Selvaraju et al. [102] introduced Grad-CAM model for class discriminative localization from any CNN-based network without modification and re-training and applied for image classification and captioning. Kumar et al. [104] developed a method for visualization and to understand the decisions made by deep neural networks (DNNs) for a given specific input.

## 2.6 Object Elimination

Object elimination is the process of masking or deleting an identified or verified object from a scene. Masking is a technique of hiding a part or a part of an object as if it were behind an invisible object. Object elimination can be implemented for various tasks such as foreground extraction, background removal, object detection, and instance segmentation. Approaches to implement object masking are as follows:

1. Contour detection-based masking
2. Mask-RCNN
3. Diminished Reality

The detail of each masking methodology is described in the following subsections.

### 2.6.1 Instance Segmentation

Instance segmentation can be defined as the identification of boundaries of the known objects at the detailed pixel. In computer vision, instance segmentation is the

problem of detecting and delineating each distinct object of interest appearing in the image. Instance segmentation can be achieved by various technologies like contour-based and Mask R-CNN. Figure 25 depicts the result of instance segmentation in the input image.

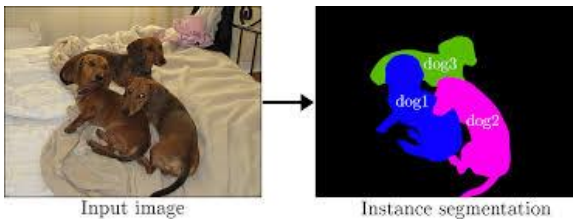



*Figure 25: Example of Instance segmentation*

Lu et al. [109] designed a method for lip segmentation. Lip segmentation is the initial step for the lip-reading system. They proposed an active contour model-based lip segmentation method that adopts local information. Figure 26 illustrates the result of lip segmentation using contour detection.

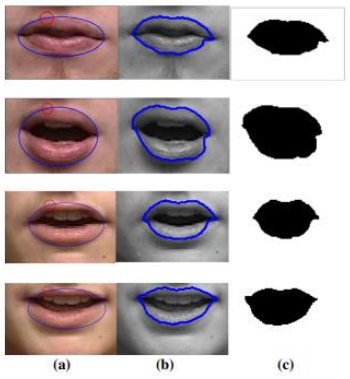



*Figure 26: Segmented results: a. initial contours and local region; b. Final evolved contours; c. segmented result (Image source: Lu et al. [109])*

Tesema et al. [110] introduced a methodology for human segmentation in still images using Deep Contour-Aware Network (DCAN) which is a unified multi-task deep learning framework combining the complementary object and contour information simultaneously for better segmentation performance. Griffiths et al. [111] presented an approach to improve public GIS building footprint labels using Morphological Geodesic Active Contours (MorphGACs). Van den Brand et al.

[112] discussed a method for vehicle detection and segmentation in the context of autonomous driving using fully convolutional network for semantic labeling and estimating the boundary of each vehicle. CNN provides the area around the contours that aids to separate the vehicle instance. Hayder et al. [113] introduced a distance transform-based mask representation that allows prediction of instance segmentations beyond the limits of initial bounding boxes. Chen et al. [81] developed a framework for more accurate detection and segmentation of histology images using deep contour-aware network. Yang et al. [13] suggested a deep learning approach for contour detection using fully convolutional encoder-decoder network. Li et al. [114] presented a novel method to borrow contour knowledge for salient object detection.

Mask R-CNN is widely used for instance segmentation. Mask R-CNN locates each pixel of the object in the image instead of the bounding boxes. He et al. [115] introduced a novel framework for instance segmentation: Mask R-CNN. Figure 27 shows the framework of Mask R-CNN.

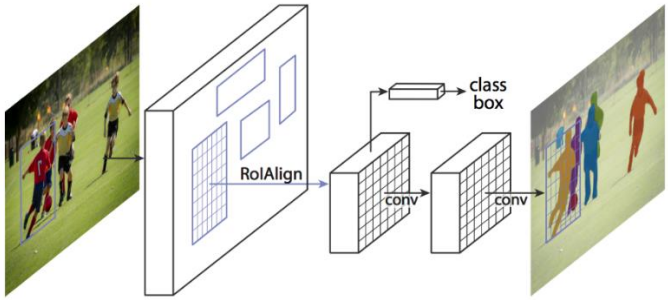



*Figure 27: Mask R-CNN framework for instance segmentation (Image source: He et al., [115])*

When an input image is passed to the network, it gives the object bounding box, classes, and masks on the detected object.

Mask R-CNN contains two stages: Firstly, it generates region proposals where there might be an object based on the input image. Secondly, it predicts the class of the object, refines the bounding box and applies a mask in the pixel level of the object based on the first stage proposal.

Novotny et al. [116] developed an approach for segmenting unknown 3D objects in real-time depth images using Mask R-CNN, trained on synthetic data. Liu et al. [117] suggested an improved version of Mask R-CNN for instance segmentation by applying the features from low levels. Yu et al. [118] built a model for fruit detection where Mask R-CNN adopts Resnet50 backbone network that is merged with the Feature Pyramid Network (FPN) architecture for feature extraction. For each feature map, RPN was trained to create region proposals. After generating mask images of ripe fruits using Mask R-CNN, a visual localization method for strawberry picking points was performed. Johnson at al. [119] demonstrated that Mask R-CNN allows highly effective and efficient segmentation of a wide range of microscopy images under various conditions and different cells.

### 2.6.2 Diminished Reality

Diminished Reality (DR) is a technique to virtually remove, hide, and see-through real objects from the real world. Diminished reality is the conceptual reverse of Augmented Reality. AR allows us to augment, add virtual world as desired whereas DR allows erasing physical content from the real-world scene. The real-time application of diminished reality includes furniture shopping, film studio, city planning, and interior designing. Figure 28 demonstrates the perception of DR by removing the glasses in an input image.

*Figure 28: Concept of Diminished Reality*

Mori et al. [12] published a survey on diminished reality techniques that is beneficial for virtually erase, hide an object from the real-time scene. They provided a concept of DR technologies and procedures for implementation. DR is performed by

executing diminish, seeing through, replace, and inpainting technologies. Figure 29 illustrates the concept of diminished reality methods.

Kawai et al. [120] performed diminished reality using inpainting method for background geometry removal with fewer constraints than the conventional ones. Mori et al. [121] designed an approach for capturing and reproducing the real world as desired using diminished reality.

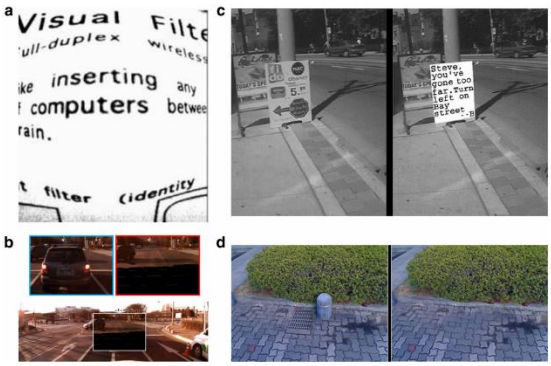



*Figure 29: Diminished reality techniques: a. diminish, b. Seeing through, c. replacing, d. inpainting (Image source: Mori et al., [12])*

Siltanen et al. [122] developed a novel framework for interior designing without using prior information of textures, via inpainting method. Nakajima et al. [123] introduced an approach for deleting an object using diminished reality.

### 2.7 Related Work

The below table 1 highlights correlated work done so far by researchers in the area closely related to this thesis, including their contributions and scope of improvements.

| **Research Paper** | **Contributions** | **Scope of Improvement** |
|---|---|---|
| Training and Testing Object Detectors with Virtual Images. Tian, Y., | Presents an artificial way to construct perceived image datasets automatically with precise annotation and | Generated virtual images can be used as prior data for object verification |

| | | |
|---|---|---|
| Li, X., Wang, K., & Wang, F. Y. (2018) | trained DPM and Faster R-CNN with real-time images and virtual images for object detection | without training any object detectors |
| A new FPGA architecture of FAST and BRIEF algorithm for on-board corner detection and matching. Huang, J., Zhou, G., Zhou, X., & Zhang, R. (2018) | FAST and BRISK descriptors are combined for corner detection and matching. | Corner matching between real-time image and virtual image using the proposed algorithm is not described. |
| Unconstrained face verification using deep cnn features. Chen, J. C., Patel, V. M., & Chellappa, R. (2016, March) | Face verification is performed using deep convolutional features that is trained using IARPA dataset. | Training of the model is required for face verification that increases the time. |
| Localizing Common Objects Using Common Component Activation Map. Li, W., Jafari, O. H., & Rother, C. (2019) | Designed CCAM model to localize common objects in an image by treating CAM as components to discover common elements. | This approach doesn't perform elimination by using the most relevant part generated by CAM. |
| Lip segmentation using localized active contour model with automatic initial contour. Lu, Y., & Zhou, T. (2018) | Lip segmentation is performed using an active contour-based model. | Requires deep learning model to train to get contours that is time-consuming. |
| Semantic object selection and detection for | Introduced a model for diminished reality that | Training of the model for target |

| | | |
|---|---|---|
| diminished reality based on SLAM with viewpoint class. Nakajima, Y., Mori, S., & Saito, H. (2017, October). | automatically recognizes the region to be removed, without generating the 3D model of the target object ad by utilizing SLAM, segmentation, and recognition framework. | object detection is mandatory. This consumes more computational power. |

*Table 1: Related Work*

## 2.8 Thesis Statement

### 2.8.1 Problem Statement

Literature survey clearly shows the need to detect dynamic objects accurately while an autonomous car is driving in order to avoid collisions. According to a literature survey, recent technologies use machine learning approaches for dynamic object detection and tracking in real-time that results in training the model and requires more computational sources. In some cases, the model detects an object that is not going to move (e.g. a person in the poster) as a dynamic object and gives bounding box as the trained models are image-based. This ends in consuming more power to process that exceptional object as a dynamic one. A similar concept applies to a car when a reflection of a car falls on the glass building, and object detection algorithm treats that reflection as a dynamic object. The methodologies for object elimination such as diminished reality, mask R-CNN uses machine learning and requires training of the model to detect the object that needs to be removed.

The proposed approach of object verification and object elimination uses constructed 3D virtual world as pre-existing knowledge. The proposed algorithm of object verification and removal makes use of virtual world to verify physical stationary (e.g. Buildings) and variable (e.g. Trees) in the real-time environment by matching keypoints of virtual objects with physical objects without any training. The proposed method of elimination uses a fusion of contour detection and class activation map (CAM) to remove verified objects in the real-time image. The

removal of stationary and variable objects will improve the accuracy and efficiency of the dynamic object algorithm.

### 2.8.2 Thesis Contribution

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

- Constructed 3D virtual model works as prior information for static (e.g. Building) and variable (e.g. Trees) object verification and elimination in the real-time scene in order to make dynamic objection algorithm efficient and accurate.
- The proposed approach of object verification doesn't require any machine learning algorithm for training.
- The verified object can be used to geo-locate the self-driving car in a real-time environment.
- For the objects having pre-existing data, the proposed fusion approach of contour detection and class activation map (CAM) for object elimination algorithm can be applied directly without any training.
- In case of objects without having prior knowledge, the model is trained using transfer learning concept to generate CAM to perform object elimination.
- After applying the object removal algorithm to the real-time image, the resulting image will be left with dynamic objects making an autonomous car focus only on those objects that cause more danger. This makes the detection method work faster.

# Chapter 3: Proposed Approach

This chapter highlights the proposed approach of stationary (e.g. Buildings, Bench, Street light) and variable (e.g. Trees) object verification and elimination with and without using prior information. It also includes pre-processing steps such as object detection and training the model. This chapter covers the architecture and flowcharts of the proposed system including detailed methods to perform them. Moreover, this chapter discusses the working of the overall system and linking of static and variable object verification and elimination model with other modules.

### 3.1 Motivation

In recent years, significant research has been made in the field of autonomous vehicle. Despite these advanced machine learning and computer vision technologies, a fully autonomous car is still far from reality. Moreover, semi-autonomous cars have been running on the road in the last couple of years for testing that involved with some pedestrian fatalities. Recently, Uber and Tesla self-driving car caused two deaths of level 3 and level 2 fatalities, that's why this raised the safety concern [77].

In the proposed approach, a constructed 3D environment of a real place is used as prior knowledge for static (e.g. building) and variable (e.g. trees) object verification and removal.

### 3.1 Working of the Overall System

The overall system consists of six modules:

1. Construction of virtual 3D environment
2. Rendered images of real-time video
3. 3D feature and keypoint extraction
4. Removal of static and variable objects
5. Dynamic object recognition
6. Dynamic object detection

As shown in Figure 30, all these modules are interconnected with each other.

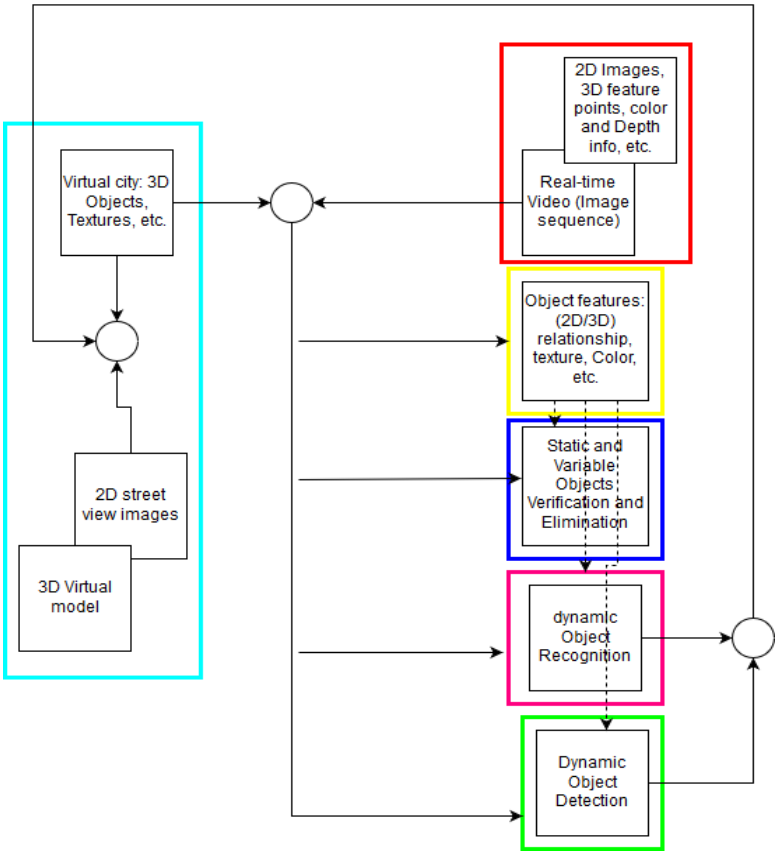



*Figure 30: Overall system architecture*

In Figure 30, the work shown in the blue-colored box is the contribution of this research work. Its connection with all other modules is justified in different colored boxes, represented using arrows.

The description of the overall system architecture is in depicted to Figure 30. The overall system primarily deals with the construction of a virtual 3D environment using OpenStreetMap data (VGI/crowdsourced) and the façade texture from Google street view images. The virtual 3D city model contains stationary objects, such as buildings, and some of the variable objects, such as trees. Apart from this, there is a separate repository that contains extracted 3D features of rendered images of stationary objects, such as buildings. The heatmap of real-time stationary objects is generated using prior understanding and stored in the repository. The module marked in the red-colored box in Figure 30 displays the real-time video that involves image frames are passed as input to the system. The module shown in the yellow-

colored box is responsible for creating the repository. The virtual models are rendered, features are extracted and stored in a repository The module marked in blue is the stationary and variable object verification and elimination, where the keypoints are first detected in the input image to verify the existence of that object in the real world by matching the extracted keypoints of input image (red-colored module) with the stored keypoints of the virtual model (yellow-colored box). Matching the features of the virtual environment and real-time image confirms the location of the car in the real-world that solves the problem of geo-localization of the self-driving car. After verifying the physical object, elimination of static and variable objects in the input image is carried out that provides more time for the identification and prediction of dynamic objects such as human beings or animals on the road, as those are the ones that create more danger to the navigation of the car. The module marked in pink deals with the object recognition and pose estimation of dynamic objects present in the real-time input image, such as cars, and pedestrians. Additionally, this module tracks the recognized objects from multiple frames of the video and calculates the speed of the dynamic object. This information including recognized object and its pose estimation, speed and location are used to update dynamic objects into the 3D virtual environment. The module marked in green color updates information of the dynamic objects of real-world into the virtual environment.

### 3.1.1 Working of Individual Modules

The modules that are directly associated with this thesis work are construction of 3D virtual world, 3D interest points extraction and repository creation, dynamic object detection. The virtual 3D city model and the real-time video are the input to the overall system.

1. **Construction of 3D Virtual World**

   Firstly, a virtual city is constructed using open source VGI data such as 2D street views and satellite images. 3D structural files are extracted with 3D structures of the buildings that are rendered, and the final 3D structure is obtained with the geolocation information that is externally mapped on to

the model. Textures are mapped onto buildings in the 3D model by extracting real-world images and georeferencing them. In this way, a virtual city with stationary (e.g. buildings) and variable objects (e.g. trees) is formed. Later this virtual city is updated with dynamic objects using real-time recognized dynamic object details. The virtual city with 3D static, variable, and dynamic object model information present in real-time road scenes is used by the self-driving car to navigate safely by knowing the surroundings. This module is marked using a light blue colored rectangle box in Figure 30.

2. **Keypoint Extraction and Dataset Creation**

   Using a constructed 3D virtual environment, the repository is created to perform static and variable object verification and elimination in the real-time input. Initialy, individual 3D models are rendered and interest points are extracted. In this research work, interest points refer to the corner points of the object. The extracted keypoints are stored in the file that is the input for object verification method. The heatmap is generated using structural information of the building that is stored in the repository to execute object elimination. After performing object elimination of real-time view, the resulting image will contain dynamic objects to improve the efficiency and accuracy of moving object detection algorithm. The module is presented in a yellow-colored box in Figure 30.

3. **Dynamic Object Recognition:**

   This module matches features of the dynamic objects in the input image with the feature information of 3D object models stored in the repository to find a suitable match of 3D model for each of the dynamic objects present in the input image. After finding the corresponding 3D model from the repository, a voting algorithm is used for the matching purpose, and to estimate the confidence score that signifies the assurance of object identification. This process improves the confidence of recognition and pose estimation of dynamic objects in the input image. This module is shown in the pink-colored box in Figure 30.

## 3.2 Proposed Methodology for Stationary and Variable Object Verification and Object Elimination

The proposed system applies the use of constructed 3D virtual environment as prior information for stationary (e.g. buildings, bench, street light) and variable (e.g. trees) objects verification and elimination in the real-time scene. Figure 31 below illustrates the architecture of the proposed methodology.

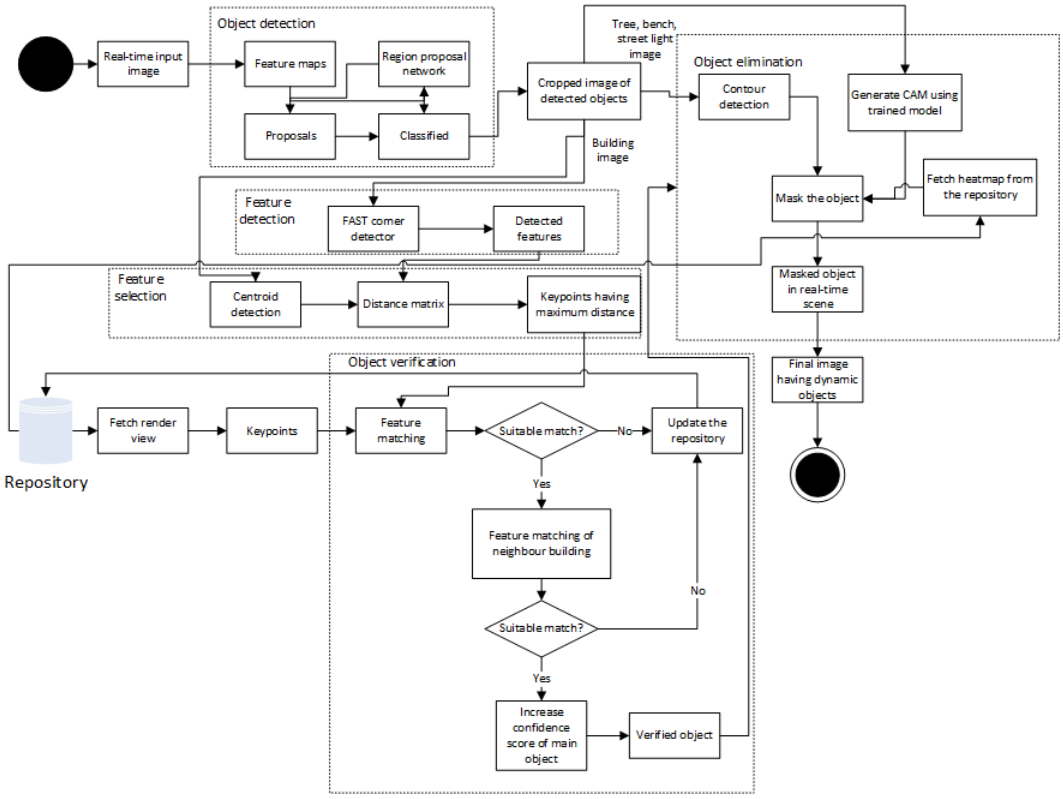



*Figure 31: Architecture of the proposed approach*

The proposed system includes two sub-modules that are as follows:

1. Object Verification
2. Object Elimination

### 3.2.1 Stationary and Variable Object Verification

Object verification is achieved by matching the interest points of the real-time world with the virtual environment. For this research work, stationary objects are buildings, bench, street light whereas variable object is tree, those are verified in the

real-time scene taken by the camera mounted on the top of an autonomous car. The stationary object (building) having prior knowledge is passed to object verification module, whereas in case of objects those are not present in the virtual world (bench, trees, street light) are sent to elimination module after performing object detection task. Figure 32 depicts the architecture of object verification module.

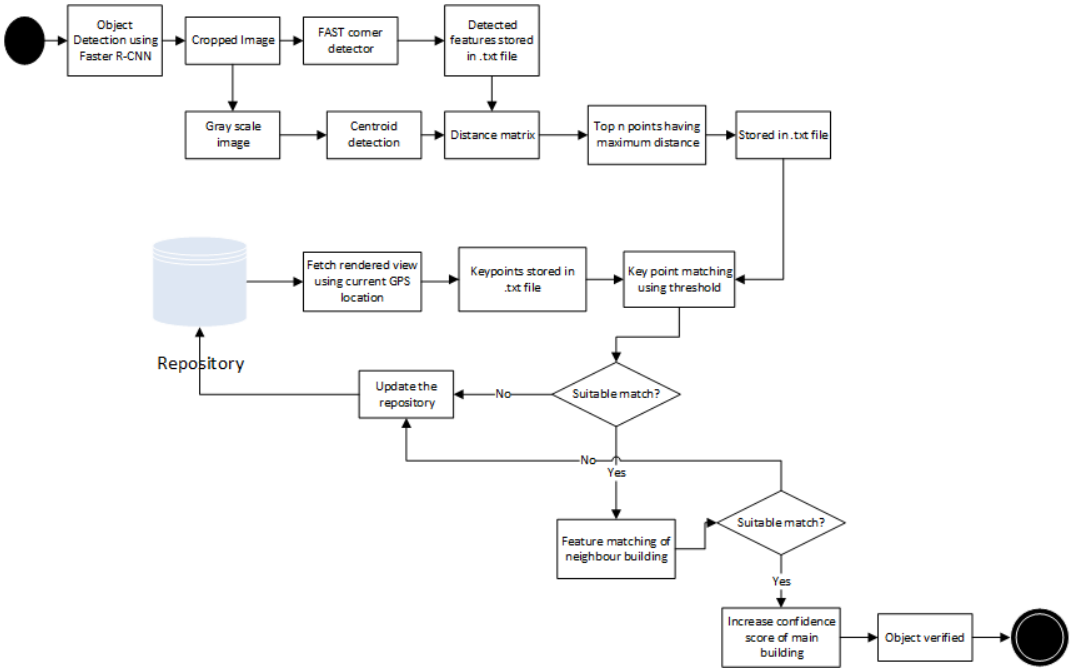



*Figure 32: Flowchart of object verification component*

### 3.2.1.1 Object Detection Module

Primarily, an object detection model is trained using Faster R-CNN algorithm (discussed in section 2.1.2) on the custom dataset to detect buildings, trees and street light in an input image. A real-time input image is fed into this trained network that gives an output image containing a corresponding class label with its probability of belonging to that class as well as bounding box around the detected instance with its coordinates. Then, the information of bounding box coordinates is utilised to crop that object from the input image. This cropped image is used by other module.

### 3.2.1.2 Feature Detection Module

For detected stationary objects (buildings), corner points define the shape of that object. Extreme points are the most relevant interest points that describe the

building's structure more efficiently. As discussed in section 2.2, many corner detection algorithms are FAST (Feature from Accelerated Segment Test) detector, Harris corner detector, and Shi-Tomasi corner detection. Among them, FAST is commonly used as a corner detector because of its speed than the other methods that is feasible for real-time scenarios [124].

In the FAST algorithm, from an input image, pixel P having an intensity $I_p$ is selected to be identified as a corner point or not. The appropriate threshold value is t. 16 pixels circle is selected around that selected pixel. The pixel P is a corner if there exists a set of n (chosen to be 12) contiguous pixels in the circle that are brighter than $I_p + t$ or darker than $I_p - t$. A high-speed test examines only the four pixels at 1, 9, 5 and 13 (First 1 and 9 are tested if they are too brighter or darker. If so, then checks 5 and 13). If P is a corner, then at least three of these must be brighter than $I_p + t$ or darker than $I_p - t$. If neither of these is the case, then P cannot be a corner. Figure 33 shows pixel P is chosen as a corner.

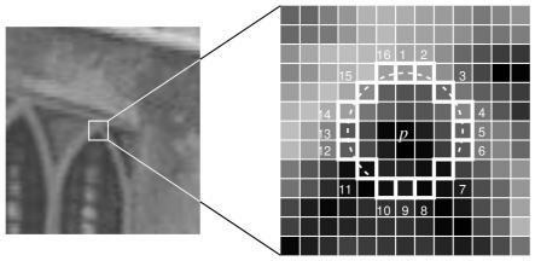



*Figure 33: Pixel P selected as a corner point*

After applying the FAST algorithm to an input image, the corner points are detected that are somehow special to define the structure of the building.

### 3.2.1.3 Feature Selection Module

Detected features in the real-time image are not efficient for the verification task using virtual image interest points. Feature selection creates a subset of interest points from the parent set; comprising of more significant, non-redundant points.

Extreme points are effective in order to portray the shape of the building. Normally, corner points are the points that are far from the centroid of the shape. The same

perception has been used here to find the most meaningful corner points of arbitrary shape.

The centroid of a shape is the arithmetic mean (i.e. the weighted average) of all the points in a shape [125]. If a shape consists of n distinct points then the centre is,

$C = \frac{1}{n}\sum_{i=1}^{n} x_i$

In computer vision and image processing, image moment can be expressed as a weighted average of image pixel intensities that is beneficial for finding specific properties such as centroid, area, radius, etc.

Centroid $(C_x, C_y) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right)$, where M denotes the Moment

Once the object centroid is computed, the distance from the centroid to the detected keypoints using FAST is calculated using Euclidean Distance. Many distance functions are Euclidean, Manhattan, Cosine, Jaccard, etc. are mainly used for distance calculation.

The Euclidean distance between two points x and y is the length of line segment connecting them. Euclidean distance can be calculated by,

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

The interest points having maximum distance from centre to detected feature points are selected using the four-quadrant method. For each quadrant, a stated number (i.e., top 8 points) of top extreme points are selected to pass to the other module. In this way, the most significant key points are selected for use.

### 3.2.1.4 Object Verification Algorithm

**Algorithm: Object Verification of Static and Variable Objects**

**INPUT**: Real-time image

**OUTPUT**: Verified object

Step 1: Real-time input image is passed to an object detection module for detection of stationary and variable objects

Step 2: For stationary object having prior information go to step 3, otherwise go to step 11

Step 3: Detect feature points using FAST corner detector

Step 4: Find the centroid of a detected object

Step 5: Calculate the Euclidean distance between centroid to all the detected features of step 3

Step 6: Find the top 8 far points from the centroid for each quadrant

Step 7: Fetch the corresponding virtual rendered image from the repository. If a virtual rendered image is not available, go to step 12.

Step 8: Verify the existence of an object in the real-world by matching the interest point between real-time objects and virtual objects

Step 9: Calculate the confidence score

Step 10: If verified successfully, perform step 3 to step 9 for the nearest object in real-time. For a successful match of neighbouring object, increase the confidence score of the primary object and go to step 11. For an unsuccessful match, go to step 11.

Step 11: Pass the verified object to object elimination module

Step 12: Exit. Check for the next object

---

### 3.2.1.5 Object Verification Module

Object verification is a technique to verify the existence of an object using prior knowledge. The stationary object's presence is verified by matching the selected interest points of the physical environment with the feature points of the virtual rendered image using some manually decided threshold value.

The confidence score is calculated on matched feature points, that is,

$$\textit{confidence score} = \frac{number\ of\ matched\ keypoints}{total\ number\ of\ keypoints}$$

For a valid confidence score, the object is considered as successfully verified. Following this, the neighbouring object is matched with the appropriate virtual

image to improve the confidence score of the verified object. If the neighbour object matches positively, the confidence score of the primary verified object is increased.

The verified object is then passed to the next module, Object elimination for further process.

### 3.2.2 Stationary and Variable Object Elimination

Once the static and variable objects are verified successfully, they are fed into an object removal module. The object masking is achieved by the fusion approach of contour detection and Class Activation Map (CAM). In the proposed system, contour detection is performed without using any deep neural network. The attention map is generated using either CNN or available prior data of the virtual city model. After performing stationary and variable object elimination in the real-time input image, the resulting image contains dynamic objects. Figure 34 displays the flowchart of the object elimination component.

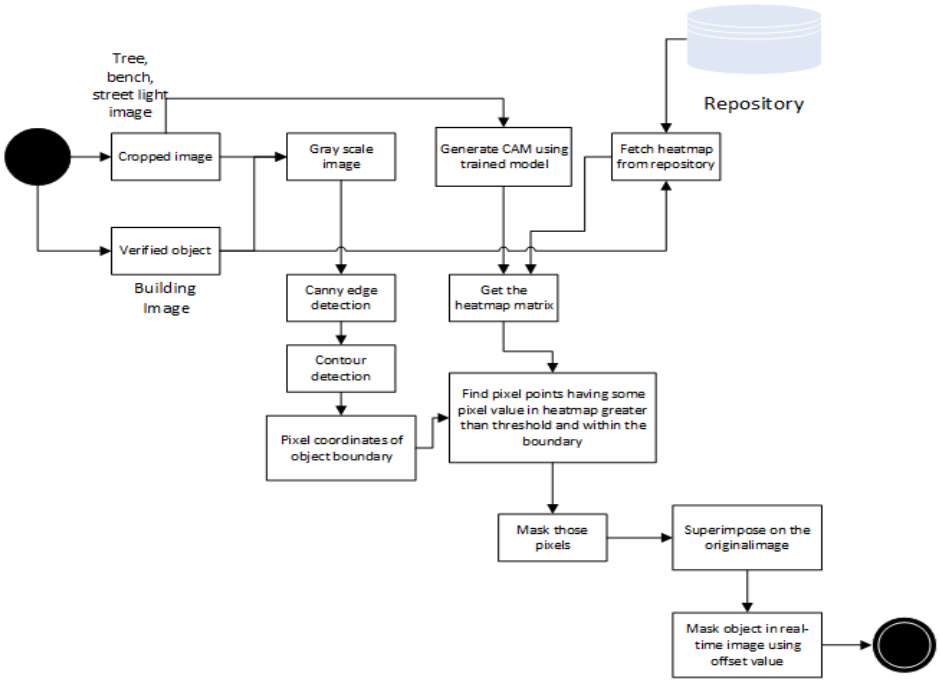



*Figure 34: Flowchart of object elimination component*

#### 3.2.2.1 Contour Detection Module

Contours are simple curves joining all the continuous points, having the same color or intensity. Contours are useful for various tasks such as shape analysis, object detection and recognition [126].

For this research, contours are computed from the detected edges. Edges are obtained as points those are extrema of the image gradient. Once the edges are found using a canny edge detection algorithm [127], contours are formed. Contours define the boundary of an object in the image. The pixels within the boundary are considered as an object segment. Only contours are not sufficient for an assurance of the actual object as it detects some outliers of other objects near to the actual object. To address this problem class activation map (CAM) is used.

#### 3.2.2.2 Generation of Class Activation Map (CAM)

As discussed in section 2.5, CAM allows us to see at which image regions CNN is looking and how relevant to a specific class in the form of visualization. The most significant region of an object is highlighted, indicating the higher chances of having the classified object's part in the image. The generated CAM comprises pixel valued probability value that can be utilized to set the threshold. For this thesis, stationary objects are buildings, bench and street light whereas variable objects are trees.

There are three different scenarios for generating CAM:

1. With using prior information of the 3D model
2. Without using prior knowledge and using a pre-trained model
3. Without using 3D models of virtual city and by training the model

In the first scenario, pre-existing data of a static object (i.e., building) is available in the virtual city to use. The information of structure is used to generate heatmap without training the convolutional neural network. The created heatmap is stored in the repository that is used for the masking process.

In case of a stationary object whose 3D model is not available (i.e., bench), a pre-trained model is responsible for generating CAM. VGG16 model is trained on the

ImageNet dataset to classify 1000 different classes is used to create a visualization map for benches.

In the third scenario, there is neither prior information in the constructed 3D world nor a pre-trained model is present to generate an attention map. The convolutional neural network is trained to produce an attention map. In machine learning, transfer learning is the concept of using the gained knowledge for solving a similar problem (as discussed in section 2.4). In transfer learning, there is no need to train the convolutional model from scratch. For this research work, the concept of transfer learning is used for training the network for producing the visualization map of detected objects in the real-time scene. The model is trained using the ImageNet dataset that produces attention map of an input image.

The created Class Activation Map (CAM) shows the most significant pixels of an object as highlighted regions. Those pixels are having high chances of belonging to the instance.

#### 3.2.2.3 Object Masking

After detecting the contour points and class activation map of an object to be eliminated, the fusion of both the approaches is applied to mask the object. The pixel points that are within the object boundary and having pixel value in the heatmap equal or greater the defined threshold are selected and eliminated.

After removing the stationary and variable objects from the real-time scene, the resulting image contains dynamic objects that increase the efficiency of detection algorithm.

#### 3.2.2.4 Object Elimination Algorithm

**Algorithm: Object Elimination of Static and Variable Objects**

**INPUT**: an input image having an object to be masked

**OUTPUT**: an output image with the masked object

Step 1 Perform Contour detection of an input image

Step 2: Find the pixels that are within the boundary of an object

Step 3: An input image is passed to the trained CNN network to generate CAM. For an object having prior knowledge, fetch the heatmap from the created repository. If not, go to step 6

Step 4: Obtain the pixels from step 2 and having pixel values in heatmap greater than the defined threshold

Step 5: Eliminate the selected pixels in the input image obtained from step 4

Step 6: Exit

---

### 3.3 Time Complexity of the Proposed Approach

The time complexity of an algorithm is calculated based on the programmatical execution. The proposed algorithm verifies the existence of an object in the real-time scene by matching interest points of the input images with the virtual rendered images of 3D object models stored in the repository. As each keypoints (i) of the input image are matched with the keypoints(j) of the virtual model's rendered image as well as the same matching is performed for the nearest object, therefore, the time complexity for object verification task is calculated as O (i * j * 2).

The object elimination module starts with finding the pixels that are within the contour boundary (i) and having pixel value greater than a defined threshold (j). The time complexity for object removing task is calculated as O (i * j). Table 2 below shows the time complexity of the proposed algorithm.

| **Module** | **Time Complexity** | **Details** |
|---|---|---|
| Object Verification | O (i * j * 2) | i = number of keypoints in the real-time input image<br>j = number of keypoints in the rendered images |
| Object Elimination | O (i * j) | i = number of keypoints in the real-time input image<br>j = number of interest points in the rendered images |

*Table 2: Time complexity of the proposed 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, various Python and OpenCV libraries were used. The list of software and tools used is given in the below Table 3.

### 4.1 Software Information

In order to implement the proposed approach, execution was carried out on the Alienware 1.5.0 x64-based Desktop, with NVIDIA 8.1.940.0 and Intel 64 ~ 3192 MHz GPU. Some results are obtained using the Dell laptop with x64-based system and Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz processor.

| **ITEM** | **DETAILS** |
|---|---|
| Operating System | Windows |
| Language | Python 3.6.5 |
| IDE | Jupyter Notebook, Anaconda Navigator |
| Python Libraries | OpenCV, Scikit, Keras, Tensorflow, Numpy, Pandas, Matplotlib, SciPy |
| Tools | 3D Viewer |

*Table 3: List of software and tools used*

### 4.2 Construction of 3D Virtual World

To employ the proposed approach of object verification and object elimination using prior data, the 3D virtual world has been produced using open source/cloud VGI data (2D street views and satellite images). This 3D environment comprises stationary (e.g. Buildings), and variable (e.g. Trees) objects. Figure 35 and 36 below displays an example of a constructed 3D virtual world.

The execution of the proposed framework is demonstrated using the real-time location: **King St S at Wills Way to King St S at William St E, Waterloo, ON.**

The virtual world has been composed for the respective real-time region to be utilized as prior knowledge and exploited to perform for various tasks like object verification and object elimination of stationary and variable objects in the real-time scenes.

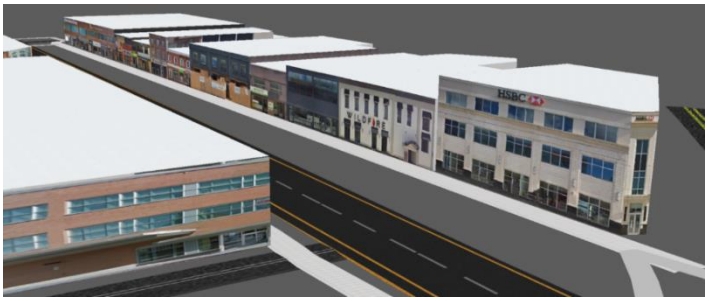



*Figure 35: Constructed 3D virtual World*

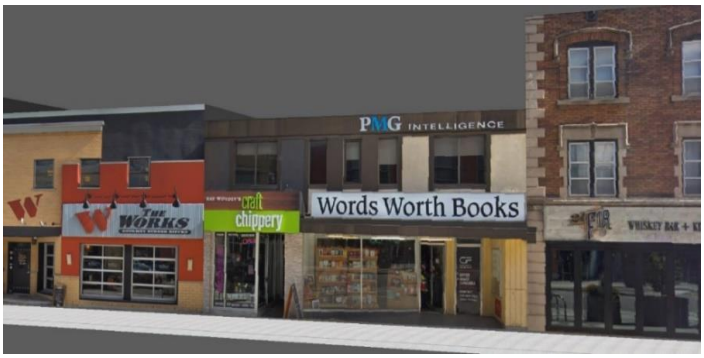



*Figure 36: Constructed 3D Objects*

### 4.3 Creation of Repository of Virtual World

Initially, from the constructed virtual world, 3D objects are rendered. The 3D interest points are extracted from rendered images. The rendered images and extracted 3D features are stored in the repository that can be further practiced for verification and to confirm the existence of that object in the real-time vision. Below is the Figure 37 which shows the rendered image of the 3D virtual object and Figure 38 that illustrates the method of 3D keypoints extraction of the rendered image.


*Figure 37: Rendered 3D object*


*Figure 38: Extracted 3D features on rendered images*

### 4.4 Experiments and Results of Object Detection

As mentioned in section 3.2.1.1, the model is trained by applying Faster-RCNN algorithm to recognize static (e.g. buildings, street light) and variable (e.g. tree) objects in the real-time view.

The model has been trained using the Google OpenImage dataset for buildings, trees, and street light detection. From 1000 images of each class, 800 was used for training and 200 images for testing the network. The model is trained for 50 epochs with a batch size of 1000. The resulting output of the detection algorithm is bounding box with coordinates around the object, class label, and probability score.

These identified objects i.e., bounding boxes are cropped and individually treated to execute object verification and object elimination methodologies.

### 4.4.1 Experiments and Results of Object Detection for Buildings

Figure 39 and Figure 40 depict the implementation result of object detection of buildings with bounding box, class label and probability score.

```
hsbc_corner_1.JPG
Elapsed time = 20.260802030563354
[('Skyscraper',      90.63831567764282),      (Street      light,
80.7957112789154), ('Tree', 88.04613947868347)] [(50, 152, 101,
356), (101, 50, 763, 381), (814, 254, 966, 356)]
```


*Figure 39: Detected Skyscraper, Street light, Tree in the real-time image*

```
scotiabank_corner_2.JPG
Elapsed time = 18.652015686035156
[('Skyscraper', 86.19678020477295), ('Tree', 94.59443688392639)]
[(107, 26, 803, 401), (0, 0, 214, 535)]
```


*Figure 40: Detected Skyscraper, Tree in the real-time image*

### 4.4.2 Experiments and Results of Object Detection for Tree

Figure 41 and Figure 42 display the experimental outcome of object detection of trees with bounding box, class label and probability value.

```
red_oak.jpg
Elapsed time = 16.516764640808105
```

```
[('Tree',   95.89992165565491),   ('Tree',   88.04725408554077),
('Tree', 76.72315835952759)] [(0, 0, 265, 321), (227, 170, 340,
302), (0, 208, 37, 302)]
```

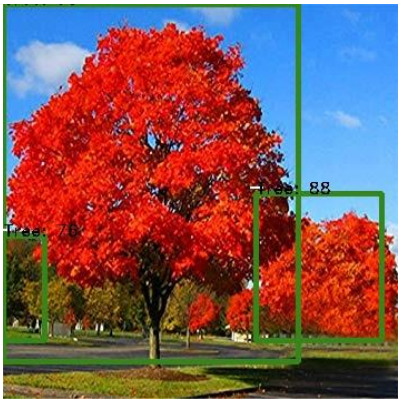


*Figure 41: Detection of Tree in the image*

```
real_canada_street_tree.jpg
Elapsed time = 17.315558671951294
[('Tree',   97.35758304595947),   ('Tree',   87.55118250846863),
('Skyscraper', 96.16764783859253)] [(288, 0, 672, 512), (0, 224,
64, 288), (32, 0, 352, 352)]
```


*Figure 42: Detected Trees, Skyscraper in the real-time image*

### 4.4.3 Experiments and Results of Object Detection for Street light

Implementation result of static object (e.g. Street light) detection are shown in Figure 43 and Figure 44.

```
street_light_real.jpg
Elapsed time = 16.90429949760437
[('Street light', 97.33138084411621), ('Tree', 68.5300409793853
8), ('Tree', 62.40453124046326)] [(138, 0, 249, 608), (0, 581, 5
5, 636), (0, 581, 359, 636)]
```

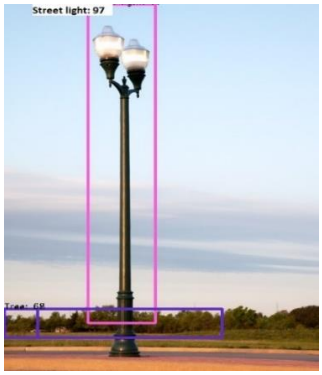



*Figure 43: Street light, Trees detected in the test image*

```
street_light_trees.jpg
Elapsed time = 19.07902717590332
[('Tree', 85.15207767486572), ('Tree', 76.06753706932068), ('Tr
ee', 62.04541325569153), ('Tree', 57.062774896621704), ('Street
light', 66.56368374824524)] [(0, 358, 921, 921), (0, 0, 307, 40
9), (0, 1024, 102, 1126), (51, 1024, 153, 1075), (460, 102, 614
, 614)]
```

*Figure 44: Street light, Trees detected in the test image*

### 4.4.4 Experiments and Results of Object Detection for Bench

For Bench detection, Python library ImageAI [68] and pre-trained model RetinaNet [68] is used. Figure 45 shows the result of bench detection in the test images.

*Figure 45: Bench detected in the test image*

### 4.5 Experiments and Results of Object Verification

The implementation results illustrate the outcome of proposed approach of static (e.g. Buildings) and variable (e.g. Trees) object verification using prior information (described in section 3.2.1).

#### 4.5.1 Centroid Detection of Real-time Stationary Object

According to the mentioned algorithm in section 3.2.1, primarily, the centroid of the stationary object (e.g. building) is estimated in the real-time cropped image.

```
# calculate moments for each contour
M = cv2.moments(contours[0])

# calculate x,y coordinate of center
cX = int(M["m10"] / M["m00"])
cY = int(M["m01"] / M["m00"])
img2 = cv2.circle(gray_image, (cX, cY), 5, (255, 255, 255), -1)
img2 = cv2.putText(gray_image, "centroid", (cX - 25, cY - 25),cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
```

*Figure 46: Centroid Calculation*

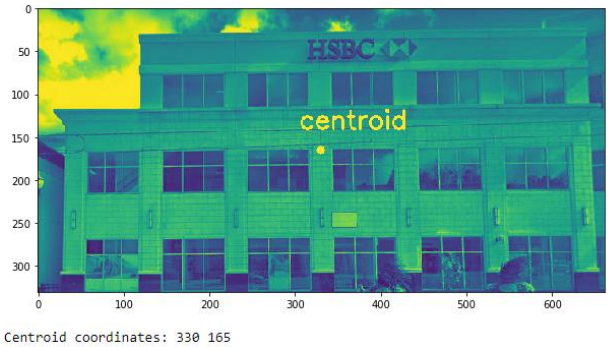



*Figure 47: Detected centroid in the real-time image with coordinates*

### 4.5.2 Feature Detection Using FAST Corner Detector of Real-Time Object

Next step is the use of OpenCV FAST feature detection algorithm that detects corner points.

```
# Initiate FAST object with default values
fast = cv2.FastFeatureDetector_create(30)
```

*Figure 48: Detection of corner points using the FAST algorithm*


*Figure 49: Detected corner points on real-time image*

### 4.5.3 Calculation of the Distance of Real-Time Object

Nextly, the detected corner points' coordinates are stored in the file. The distance is calculated for each interest points from the centroid which is stored in another file for further use.

```
for p in range(count):
    dist = math.hypot(xcord[p]-cX,ycord[p]-cY)
    #dist = math.sqrt(((xcord[p]-cX)**2)+((ycord[p]-cY)**2))
    d[p] = dist
```

*Figure 50: Calculating the distance between keypoints and centroid*

### 4.5.4 Feature Selection in Real-Time Image

Using a four-quadrant approach, the top 8 feature points are selected those are far from the centroid. This method is applied to each quadrant.

```
trdist = []
for i in trght:
    trdist.append(d[i])
print("Distance of top-right points", trdist)

#top 8 points from trdist
tr = np.argpartition(trdist,-8)[-8:] #index value of top 8 distance in trdist
print("Index value of top 8 top-right keypoints: ",tr)

top_right = np.array(trght)[tr.astype(int)]
print(top_right)
```

*Figure 51: Selected top 8 keypoints for top-right (1st quadrant)*

After selecting the top 8 keypoints for each quadrant, they are plotted on the real-time cropped image as well as stored in the file for further use.


*Figure 52: Plotted selected keypoints on the real-time image*

### 4.5.5 Retrieving the Virtual Image and Keypoints

The proposed approach of keypoint selection is applied to virtual object feature selection. Figure 53 shows the feature selection of the corresponding virtual rendered image using the proposed approach.


*Figure 53: Plotted selected keypoints on the virtual image*

### 4.5.6 Object Verification of Stationary Object Using Prior Knowledge

As per the proposed approach, verification is carried out with the aid of the virtual world and physical objects. Verification is performed by matching selected interest points of real-time (Figure 52) with virtual (Figure 53) objects using a threshold value. Figure 53 shows the results of the verification. If the confidence score of the match is equal or greater than 70%, the object is considered as verified positively.

```
Number of matched keypoints out of 32 points:  29
Matched keypoints:  [[3.0, 323.0], [4.0, 208.0], [5.0, 321.0], [19.0, 120.0], [21.0, 124.0], [22.0, 126.0], [27.0, 309.0], [28.
0, 303.0], [28.0, 305.0], [30.0, 303.0], [31.0, 137.0], [31.0, 161.0], [31.0, 306.0], [34.0, 149.0], [640.0, 31.0], [642.0, 10
8.0], [643.0, 303.0], [644.0, 80.0], [644.0, 144.0], [645.0, 109.0], [652.0, 303.0], [653.0, 281.0], [653.0, 300.0], [654.0, 28
6.0], [654.0, 306.0], [655.0, 304.0], [656.0, 78.0], [657.0, 326.0], [658.0, 115.0]]
Number of unmatched keypoints:  3
Confidence score:  0.90625
Confidence probability:  90.625
```

*Figure 54: Object verification result of real-time object*

### 4.5.7 Increasing the Confidence Score Using Neighbour Object

After verifying the real-time object successfully with the help of prior knowledge and using some threshold value, the neighbouring object is used to increase the confidence score of the verified object. The same procedure is applied for the verification of neighbouring building. Figure 55 - 58 illustrates the result of increasing the probability of verified object using neighbouring building. If the neighbouring object is verified successfully, then 0.5 confidence score is added to the main verified stationary object's confidence score.


*Figure 55: Selected keypoints of the virtual nearest static object (e.g. building)*

*Figure 56: Selected keypoints of the real-time nearest static object (e.g. building)*

```
Number of matched keypoints out of 32 points of neighbour building:  26
Matched keypoints of neighbour building:  [[3.0, 198.0], [3.0, 202.0], [4.0, 141.0], [7.0, 127.0], [8.0, 398.0], [9.0, 382.0],
[9.0, 385.0], [9.0, 396.0], [11.0, 126.0], [12.0, 122.0], [14.0, 121.0], [14.0, 390.0], [16.0, 128.0], [18.0, 390.0], [407.0, 1
99.0], [419.0, 184.0], [420.0, 189.0], [424.0, 175.0], [446.0, 187.0], [469.0, 179.0], [471.0, 400.0], [472.0, 411.0], [473.0,
180.0], [474.0, 411.0], [482.0, 178.0], [487.0, 390.0]]
Number of unmatched keypoints of neighbour building:  6
Confidence score:  0.8125
Confidence probability:  81.25
```

*Figure 57: Verification result of neighbour stationary object (e.g. building)*

```
Confidence score of main building after matching with neighbour building:  0.95625
Confidence probability of building:  95.625
```

*Figure 58: Increased confidence score of a verified static object*

### 4.6 Experiments and Results of Object Elimination

This section includes the implementation result of the proposed algorithm of static (e.g. Buildings, Bench, Street light) and variable (e.g. Tree) object removal explained in section 3.2.2.

#### 4.6.1 Results of Generation of Class Activation Map (CAM)

According to the proposed technique as described in section 3.2.2, the model is trained using Transfer Learning to generate Class Activation Map (CAM) for the objects without having prior understanding, i.e., those are not present in the virtual world. The model is trained using pre-trained top layers weights of VGG16 that is trained on the ImageNet dataset for 1000 different classes. The proposed model is trained using 1000 training images of each class (i.e., Tree, Street light) with 32 batch size. Figure 59 displays the execution output of CAM generation of Trees. The result of CAM generation of Street light is shown in Figure 60.

Generation of heatmap:

```
heatmap = np.mean(conv_layer_output_value, axis=-1)
heatmap = np.maximum(heatmap, 0)
heatmap = heatmap/heatmap.max()
```

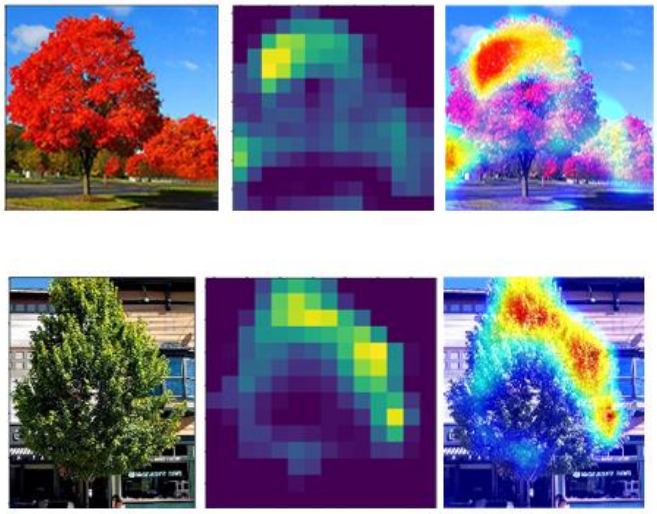
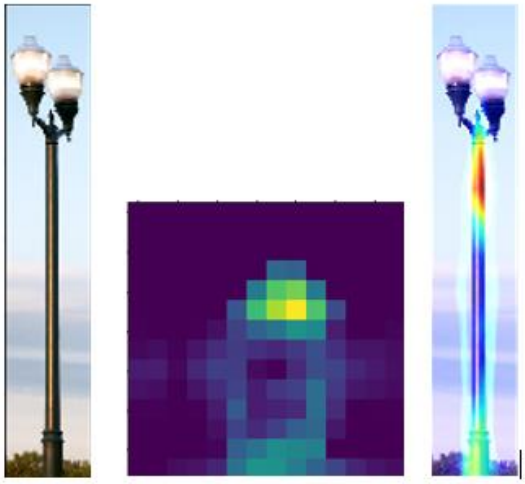


*Figure 59: Result of CAM Generation of Tree (left: input image, centre: generated heatmap, right: heatmap superimposed on input image)*



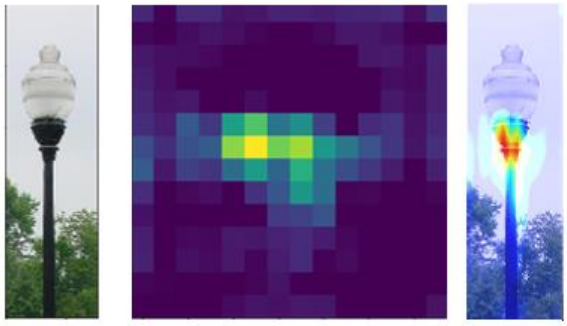


*Figure 60: Result of CAM Generation of Street light (left: input image, centre: generated heatmap, right: heatmap superimposed on input image)*

For CAM generation of Bench, pre-trained model VGG16 is used that is trained on the ImageNet dataset for 1000 various classes. Figure 61 illustrates the result of CAM generation of Bench.

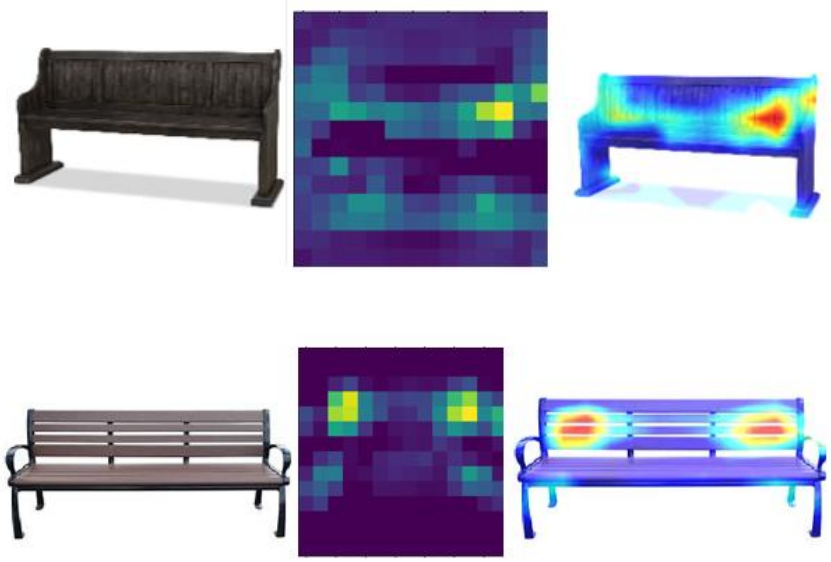


*Figure 61: Result of CAM Generation of Street light (left: input image, centre: generated heatmap, right: heatmap superimposed on input image)*

### 4.6.2 Results of Contour Detection

As per the described methodology in section 3.2.2.1, contours are detected on real-time objects using canny edge detection algorithm. Figure 62 depicts the result of the contour detection of Trees.

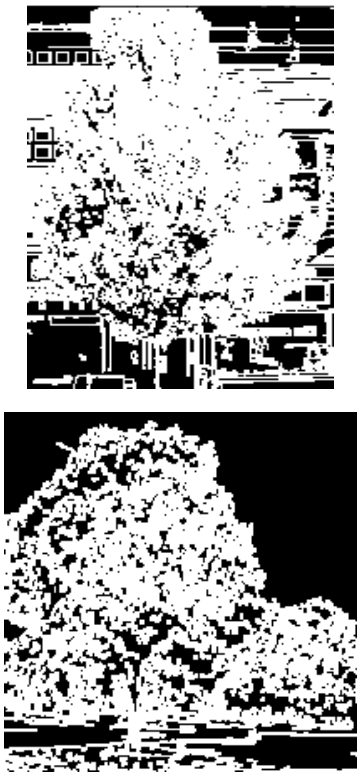


*Figure 62: Result of Contour Detection of Trees*

The results of contour detection of Street light is shown in Figure 63.

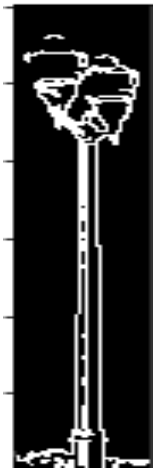


*Figure 63: Result of Contour Detection of Street light*

Figure 64 displays the result of the contour detection of Bench.

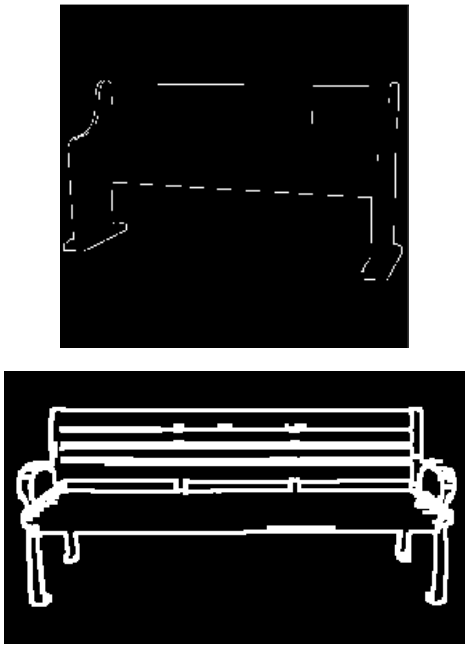


*Figure 64: Result of Contour Detection of Bench*

The output of contour detection of the building is displayed in the below Figure 65.

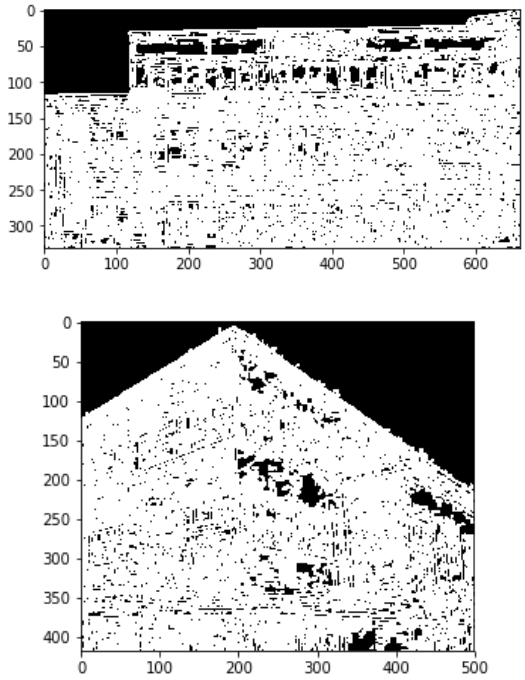



*Figure 65: Result of Contour Detection of Buildings*

### 4.6.3 Results of Object Elimination Using Combined Approach

As per the mentioned technique in section 3.3.2.3 for static and variable object elimination, combined approach of Class Activation Map (CAM) (Figure 59-61) and Contour Detection (Figure 62-65) is used. The points having equal or greater value than defined threshold value and within the object boundary are selected and masked out in the input image. Figure 66, 67, 68, 69 depict the result of object elimination of Tree, Street light, Bench and Buildings, respectively.

*Figure 66: Result of Object Elimination of Trees*

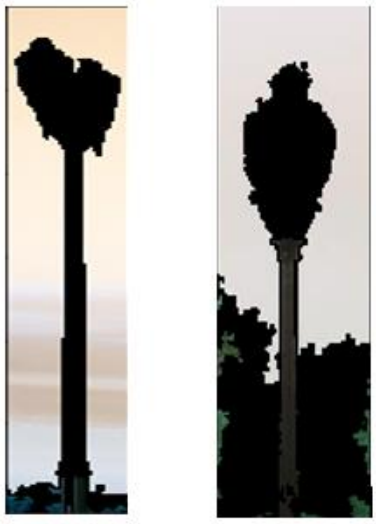


*Figure 67: Result of Object Elimination of Street light*

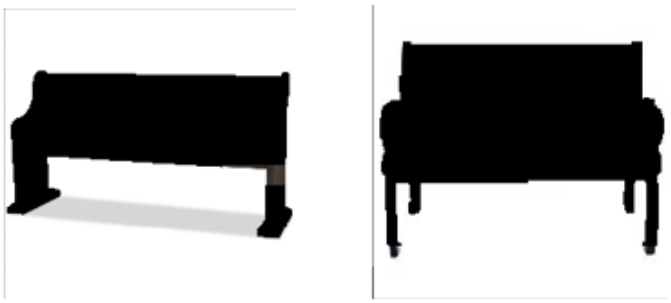


*Figure 68: Result of Object Elimination of Bench*

*Figure 69: Result of Object Elimination of Buildings*

Figure 70 illustrates the final output of the proposed algorithm where static and variable objects are masked out and dynamic objects are left in the input image.


*Figure 70: Result of the proposed algorithm*

## 4.7 Results Comparison and Discussion

### 4.7.1 Advantages of the Proposed Approach

Nair et al. [71] proposed an approach for moving object detection and human pose estimation. The proposed method uses trained model RestinaNet [68] for dynamic object detection in real-time images. This model detects a human in the poster as well as the reflection of a person on the building that is time-consuming as it processes that exceptional object as a dynamic object that is not going to move. Figure 71 illustrates the person detected in the poster using RetinaNet model [68] in the real-time image.

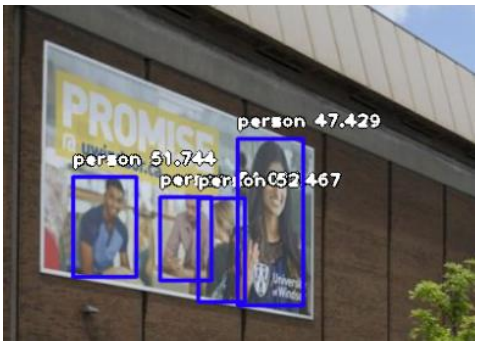



*Figure 71: Person Detected in the poster using RetinaNet model*

The RetinaNet model [68] also detects the reflection of the car and person as a dynamic object that ends in utilizing more time for an object that will remain stationary. Figure 72 depicts the outcome of object detection using RetinaNet model [68]. The left image shows the detected object class with its probability, bounding box coordinates, and execution time of the object detection algorithm on the test image without masked objects and the right image depicts the outcome of object detection algorithm. The object detection algorithm took 18.6 seconds to run on the original real-time image without applying the proposed approach of object elimination.

```
person  :  33.36006700992584  :  [400 300 422 357]
--------------------------------
car  :  44.00858283042908  :  [ 23 296 104 332]
--------------------------------
car  :  55.09122014045715  :  [191 298 284 350]
--------------------------------
potted plant  :  55.63378930091858  :  [475 333 574 434]
--------------------------------
car  :  94.80350017547607  :  [  8 371 640 567]
--------------------------------
car  :  93.2698905467987  :  [615 391 810 569]
--------------------------------
Wall time: 18.6 s
```

*Figure 72: Reflection of car and person on the building detected as a dynamic object*

The proposed method in this research uses prior knowledge to verify and remove the stationary (e.g. buildings) and variable (e.g. trees) objects in the real-time image. After applying the proposed approach for masking the building as illustrated in section 4.6.3, the unusual objects are eliminated in the image that allows moving

object detection algorithm works accurately and efficiently. Figure 73 illustrates the result of moving object detection method after masking. The building is eliminated in the image therefore, the car is not detected on the side of the buildings that makes the dynamic object detection algorithm to perform efficiently. The execution time of the object detection algorithm on the masked image is 15.8 seconds. The implementation time of the proposed object elimination method is 548 milliseconds.


*Figure 73: Detection of dynamic objects after object elimination*

The proposed approach of object elimination gives a significant result for the dynamic object detection algorithm. The removal of stationary and variable objects in the real-time image allows the moving object detection method to perform more efficiently and accurately.

### 4.8 Limitations of the Proposed Approach

The proposed technique uses a constructed 3D virtual world as prior knowledge to verify the existence of static and variable objects in the real-time environment. This method of verification and elimination is dependent on the virtual world that can not be applied to the objects without having prior information.

For the objects that are not present in the virtual world, the machine learning technique is required for training to generate CAM of that stationary and variable object in the physical world. The description of that technique is explained in section 3.2.2.2 and illustrated in section 4.6.

As mentioned in section 4.6.1, the model trained using transfer learning is used to produce a Class Activation Map (CAM) for elimination. The trained model is not accurate for all types of trees as trees may change their shape according to the weather. The proposed trained model to generate Class Activation Map (CAM) is not accurate for trees without leaves. Figure 74 shows the result of the generated Class Activation Map (CAM) for the tree with no leaves.

*Figure 74: Generated CAM for trees without leaves*

# Chapter 5: Conclusion and Future Work

## 5.1 Conclusion

With the help of recent machine learning and computer vision techniques, many leading automobile companies are stepping towards to build an autonomous car, i.e., a car drives by itself without any human inputs. In other words, software on the wheels. The several benefits of a driverless car include less traffic, human comfort, increased safety, time and space-saving. Despite using modern approaches, it is still far away to become fully robotic. However, for the driverless-car to function with negligence of accidents, it needs to be aware of surroundings including stationary (e.g. buildings, street light, benches), variable (e.g. trees) and dynamic (e.g. pedestrians, car) objects. The main objective of the proposed approach is to verify and mask the stationary and variable objects in the real-time image using the virtual world as prior data.

The constructed virtual 3D world assists an autonomous car to understand the surroundings while moving on the street. Using this information, static and variable objects are verified and removed by matching the feature points of the physical world with a virtual world. For object removal, the fusion technique of contour detection and Class Activation Map (CAM) is used. This allows an autonomous car to focus on moving objects that adds a significant danger to drive.

Section 4.5 and 4.6 illustrates the results of the proposed method of stationary and variable object verification and elimination, respectively. The results prove that the proposed technique is capable for verifying the existing object in the real-time environment by matching the extracted keypoints between the real-time and the virtual object using some threshold. The method for object removal uses a combined way of contour detection and Class Activation Map (CAM) for accurate results. The elimination method of stationary and variable objects reduces the execution time of dynamic object detection algorithm as well as improves the efficiency of the

algorithm. The saved time can be invested in moving object detection and prediction algorithm to work faster and accurately in the real world.

### 5.2 Future Work

However, the proposed approach in this research of stationary and variable object verification and removal illustrates reasonable results, there is a room for improvement.

1. The proposed approach of training a model to generate Class Activation Map (CAM) for trees can be enhanced by adding various types and shapes of the trees according to weather to make it work more efficiently.
2. The proposed approach for verification and elimination of static and variable objects can be applied to verify and remove street light and trees in the real-time scene if prior information is available in the virtual world to use.

# References/Bibliography


1. Milestones of self-driving car: https://www.digitaltrends.com/cars/history-of-self-driving-cars-milestones/
2. https://www.wired.com/story/guide-self-driving-cars/
3. Darms, M., Rybski, P., & Urmson, C. (2008, June). Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In *2008 IEEE Intelligent Vehicles Symposium* (pp. 1197-1202). IEEE.
4. Hu, X., Chen, L., Tang, B., Cao, D., & He, H. (2018). Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. *Mechanical Systems and Signal Processing*, *100*, 482-500.
5. Fu, K., Dai, W., Zhang, Y., Wang, Z., Yan, M., & Sun, X. (2019). Multicam: Multiple class activation mapping for aircraft recognition in remote sensing images. *Remote Sensing*, *11*(5), 544.
6. Zhao, J., Liang, B., & Chen, Q. (2018). The key technology toward the self-driving car. *International Journal of Intelligent Unmanned Systems*, *6*(1), 2-20.
7. https://medium.com/waymo/scenes-from-the-street-5bb77046d7ce#.tq11yyoqw
8. Druzhkov, P. N., & Kustikova, V. D. (2016). A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, *26*(1), 9-15.
9. Chen, Y., Li, W., Sakaridis, C., Dai, D., & Van Gool, L. (2018). Domain adaptive faster r-cnn for object detection in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3339-3348).

10. Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. (2018). Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, *14*(9), 4224-4231.
11. Li, S. (2017, September). A review of feature detection and match algorithms for localization and mapping. In *IOP Conference Series: Materials Science and Engineering* (Vol. 231, No. 1, p. 012003). IOP Publishing.
12. Mori, S., Ikeda, S., & Saito, H. (2017). A survey of diminished reality: Techniques for visually concealing, eliminating, and seeing through real objects. *IPSJ Transactions on Computer Vision and Applications*, *9*(1), 1-14.
13. Yang, J., Price, B., Cohen, S., Lee, H., & Yang, M. H. (2016). Object contour detection with a fully convolutional encoder-decoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 193-202).
14. Object Detection : https://towardsdatascience.com/object-detection-using-deep-learning-approaches-an-end-to-end-theoretical-perspective-4ca27eee8a9a
15. Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object Detection in 20 Years: A Survey. *arXiv preprint arXiv:1905.05055*.
16. Erickson, B. J., Korfiatis, P., Akkus, Z., & Kline, T. L. (2017). Machine learning for medical imaging. *Radiographics*, *37*(2), 505-515.
17. Ma, L., Fu, T., Blaschke, T., Li, M., Tiede, D., Zhou, Z., ... & Chen, D. (2017). Evaluation of feature selection methods for object-based land cover mapping of unmanned aerial vehicle imagery using random forest and support vector machine classifiers. *ISPRS International Journal of Geo-Information*, *6*(2), 51.
18. Bakhshipour, A., & Jafari, A. (2018). Evaluation of support vector machine and artificial neural networks in weed detection using shape features. *Computers and Electronics in Agriculture*, *145*, 153-160.

19. Bazi, Y., & Melgani, F. (2018). Convolutional SVM networks for object detection in UAV imagery. *Ieee transactions on geoscience and remote sensing*, *56*(6), 3107-3118.
20. Wei, Y., Tian, Q., Guo, J., Huang, W., & Cao, J. (2019). Multi-vehicle detection algorithm through combining Harr and HOG features. *Mathematics and Computers in Simulation*, *155*, 130-145.
21. Chee, K. W., & Teoh, S. S. (2019). Pedestrian Detection in Visual Images Using Combination of HOG and HOM Features. In *10th International Conference on Robotics, Vision, Signal Processing and Power Applications* (pp. 591-597). Springer, Singapore.
22. Wang, Y., Zhu, X., & Wu, B. (2019). Automatic detection of individual oil palm trees from UAV images using HOG features and an SVM classifier. *International Journal of Remote Sensing*, *40*(19), 7356-7370.
23. Rashid, M., Khan, M. A., Sharif, M., Raza, M., Sarfraz, M. M., & Afza, F. (2019). Object detection and classification: a joint selection and fusion strategy of deep convolutional neural network and SIFT point features. *Multimedia Tools and Applications*, *78*(12), 15751-15777.
24. Mihçioğlu, M. E., & Alkar, A. Z. (2019). Improving pedestrian safety using combined HOG and Haar partial detection in mobile systems. *Traffic Injury Prevention*, 1-5.
25. Prasanna, D., & Prabhakar, M. (2018). An efficient human tracking system using Haar-like and hog feature extraction. *Cluster Computing*, 1-8.
26. Arunmozhi, A., & Park, J. (2018, May). Comparison of HOG, LBP and Haar-Like features for on-road vehicle detection. In *2018 IEEE International Conference on Electro/Information Technology (EIT)* (pp. 0362-0367). IEEE.
27. Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, *300*, 17-33.
28. Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., & Mouzakitis, A. (2019). A survey on 3d object detection methods for

autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*.

29. Panchpor, A. A., Shue, S., & Conrad, J. M. (2018, January). A survey of methods for mobile robot localization and mapping in dynamic indoor environments. In *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)* (pp. 138-144). IEEE.
30. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2018). Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*.
31. Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., ... & Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, *51*(5), 92.
32. Sindagi, V. A., & Patel, V. M. (2018). A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, *107*, 3-16.
33. Verma, S., Eng, Y. H., Kong, H. X., Andersen, H., Meghjani, M., Leong, W. K., ... & Rus, D. (2018, May). Vehicle Detection, Tracking and Behavior Analysis in Urban Driving Environments Using Road Context. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1413-1420). IEEE.
34. Possatti, L. C., Guidolini, R., Cardoso, V. B., Berriel, R. F., Paixão, T. M., Badue, C., ... & Oliveira-Santos, T. (2019). Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars. *arXiv preprint arXiv:1906.11886*.
35. Mittal, N., & Kapoor, A. V. A. P. S. (2019). Object Detection and Classification Using Yolo.
36. Putra, M. H., Yussof, Z. M., Lim, K. C., & Salim, S. I. (2018). Convolutional neural network for person and car detection using yolo framework. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *10*(1-7), 67-71.

37. Wang, X., Hua, X., Xiao, F., Li, Y., Hu, X., & Sun, P. (2018). Multi-Object Detection in Traffic Scenes Based on Improved SSD. *Electronics*, *7*(11), 302.
38. Zhang, S., Wen, L., Bian, X., Lei, Z., & Li, S. Z. (2018). Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4203-4212).
39. Zhang, Z., Qiao, S., Xie, C., Shen, W., Wang, B., & Yuille, A. L. (2018). Single-shot object detection with enriched semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5813-5821).
40. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
41. Židek, K., Lazorík, P., Piteľ, J., & Hošovský, A. (2019). An Automated Training of Deep Learning Networks by 3D Virtual Models for Object Recognition. *Symmetry*, *11*(4), 496.
42. Tian, Y., Li, X., Wang, K., & Wang, F. Y. (2018). Training and testing object detectors with virtual images. *IEEE/CAA Journal of Automatica Sinica*, *5*(2), 539-546.
43. Loing, V., Marlet, R., & Aubry, M. (2018). Virtual training for a real application: Accurate object-robot relative localization without calibration. *International Journal of Computer Vision*, *126*(9), 1045-1060.
44. Neumann, L., Karg, M., Zhang, S., Scharfenberger, C., Piegert, E., Mistr, S., ... & Schiele, B. (2018, December). NightOwls: A pedestrians at night dataset. In *Asian Conference on Computer Vision* (pp. 691-705). Springer, Cham.
45. Chowdhury, D. R., Garg, P., & More, V. N. (2019, April). Pedestrian Intention Detection Using Faster RCNN and SSD. In *International Conference on Advances in Computing and Data Sciences* (pp. 431-439). Springer, Singapore.

46. Sheng, M., Liu, C., Zhang, Q., Lou, L., & Zheng, Y. (2018, May). Vehicle Detection and Classification Using Convolutional Neural Networks. In *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)* (pp. 581-587). IEEE.
47. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).
48. Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).
49. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).
50. Wang, J., & Zhang, W. (2018, March). A Survey of Corner Detection Methods. In *2018 2nd International Conference on Electrical Engineering and Automation (ICEEA 2018)*. Atlantis Press.
51. Karim, S., Zhang, Y., Asif, M. R., & Ali, S. (2017). Comparative analysis of feature extraction methods in satellite imagery. *Journal of Applied Remote Sensing*, *11*(4), 042618.
52. DeTone, D., Malisiewicz, T., & Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 224-236).
53. Al-Rawabdeh, A., Almagbile, A., Aldayafleh, O., Zeitoun, M., & Hazaymeh, K. (2019). Evaluating the Performance of Corner Detection Approaches for Features Extraction from UAV Images.
54. Huang, J., Zhou, G., Zhou, X., & Zhang, R. (2018). A new FPGA architecture of fast and BRIEF algorithm for on-board corner detection and matching. *Sensors*, *18*(4), 1014.
55. Hore, S., Chatterjee, S., Chakraborty, S., & Shaw, R. K. (2018). Analysis of different feature description algorithm in object recognition. In *Computer*

*Vision: Concepts, Methodologies, Tools, and Applications* (pp. 601-635). IGI Global.

56. Gao, X., Wang, M., Yang, Y., & Li, G. (2018). Building extraction from RGB VHR images using shifted shadow algorithm. *IEEE Access*, *6*, 22034-22045.
57. Wu, W., Xu, H., Zhong, S., Lyu, M. R., & King, I. (2019, June). Deep validation: Toward detecting real-world corner cases for deep neural networks. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 125-137). IEEE.
58. Alzugaray, I., & Chli, M. (2018). Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robotics and Automation Letters*, *3*(4), 3177-3184.
59. Liu, C., Huang, X., Chen, H., Yang, J., & Gong, J. (2018, July). Building Area Extraction from High-Resoluton Satellite Imagery Based on Morphological Building Index. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium* (pp. 8201-8204). IEEE.
60. Ghandour, A., & Jezzini, A. (2018). Autonomous building detection using edge properties and image color invariants. *Buildings*, *8*(5), 65.
61. Zhao, Y., Qi, J., & Zhang, R. (2019, May). Cbhe: Corner-based building height estimation for complex street scene images. In *The World Wide Web Conference* (pp. 2436-2447). ACM.
62. Hu, Y., Hu, X., Li, P., & Ding, Y. (2018). Building detection from orthophotos using binary feature classification. *Multimedia Tools and Applications*, *77*(3), 3339-3351.
63. Haggui, O., Tadonki, C., Lacassagne, L., Sayadi, F., & Ouni, B. (2018). Harris corner detection on a NUMA manycore. *Future Generation Computer Systems*, *88*, 442-452.
64. Kabir, S. R., Akhtaruzzaman, M., & Haque, R. (2018). Performance Analysis of Different Feature Detection Techniques for Modern and Old Buildings. In *RTA-CSIT* (pp. 120-127).

65. Karami, E., Shehata, M., & Smith, A. (2017). Image Identification Using SIFT Algorithm: Performance Analysis Against Different Image Deformations. *arXiv preprint arXiv:1710.02728*.
66. Karami, E., Prasad, S., & Shehata, M. (2017). Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*.
67. Ghosh, P., Pandey, A., & Pati, U. C. (2015). Comparison of different feature detection techniques for image mosaicing. *ACCENTS Transactions on Image Processing and Computer Vision*, *1*(1), 1-7.
68. ImageAI Python: https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606
69. Object Detection: https://en.wikipedia.org/wiki/Object_detection
70. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.
71. Nair, C. R. (2019). A Voting Algorithm for Dynamic Object Identification and Pose Estimation.
72. Berger, T., Lettner, D., Rubin, J., Grünbacher, P., Silva, A., Becker, M., ... & Czarnecki, K. (2015, July). What is a feature?: a qualitative study of features in industrial software product lines. In *Proceedings of the 19th International Conference on Software Product Line* (pp. 16-25). ACM.
73. Feng, J., Ai, C., An, Z., Zhou, Z., & Shi, Y. (2019, July). A Feature Detection and Matching Algorithm Based on Harris Algorithm. In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)* (pp. 616-621). IEEE.
74. Liu, Y., Zhang, H., Guo, H., & Xiong, N. (2018). A FAST-BRISK Feature Detector with Depth Information. *Sensors*, *18*(11), 3908.
75. Chen, X., Liu, L., Song, J., Li, Y., & Zhang, Z. (2018). Corner detection and matching for infrared image based on double ring mask and adaptive SUSAN algorithm. *Optical and Quantum Electronics*, *50*(4), 194.
76. Feature Extraction: https://en.wikipedia.org/wiki/Feature_extraction

77. List of self-driving car fatalities: https://en.wikipedia.org/wiki/List_of_self-driving_car_fatalities
78. Chen, J. C., Patel, V. M., & Chellappa, R. (2016, March). Unconstrained face verification using deep cnn features. In *2016 IEEE winter conference on applications of computer vision (WACV)* (pp. 1-9). IEEE.
79. Ranjan, R., Bansal, A., Xu, H., Sankaranarayanan, S., Chen, J. C., Castillo, C. D., & Chellappa, R. (2018). Crystal loss and quality pooling for unconstrained face verification and recognition. *arXiv preprint arXiv:1804.01159*.
80. Crosswhite, N., Byrne, J., Stauffer, C., Parkhi, O., Cao, Q., & Zisserman, A. (2018). Template adaptation for face verification and identification. *Image and Vision Computing*, *79*, 35-48.
81. Chen, H., Qi, X., Yu, L., Dou, Q., Qin, J., & Heng, P. A. (2017). DCAN: Deep contour-aware networks for object instance segmentation from histology images. *Medical image analysis*, *36*, 135-146.
82. Zhang, Z., He, Z., Cao, G., & Cao, W. (2016). Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification. *IEEE Transactions on Multimedia*, *18*(10), 2079-2092.
83. Hsu, S. C., Chang, I. C., & Huang, C. L. (2018). Vehicle verification between two nonoverlapped views using sparse representation. *Pattern Recognition*, *81*, 131-146.
84. Taira, H., Okutomi, M., Sattler, T., Cimpoi, M., Pollefeys, M., Sivic, J., ... & Torii, A. (2018). InLoc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7199-7209).
85. Yuan, X., Gu, L., Chen, T., Elhoseny, M., & Wang, W. (2018, March). A fast and accurate retina image verification method based on structure similarity. In *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)* (pp. 181-185). IEEE.
86. Kavitha, S. N., Shahila, K., & Kumar, S. P. (2018, February). Biometrics Secured Voting System with Finger Print, Face and Iris Verification. In *2018*

*Second International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 743-746). IEEE.

87. Qin, H., & El-Yacoubi, M. A. (2017). Deep representation-based feature extraction and recovering for finger-vein verification. *IEEE Transactions on Information Forensics and Security*, *12*(8), 1816-1829.
88. Mohamed, E., Sirlantzis, K., & Howells, G. (2019). Application of Transfer Learning for Object Detection on Manually Collected Data.
89. Yabuki, N., Nishimura, N., & Fukuda, T. (2018, June). Automatic object detection from digital images by deep learning with transfer learning. In *Workshop of the European Group for Intelligent Computing in Engineering* (pp. 3-15). Springer, Cham.
90. Kapur, P. (2018). *Object Detection in Video Based on Transfer Learning Using Convolution Neural Network* (Doctoral dissertation).
91. Yuan, X., Li, D., Mohapatra, D., & Elhoseny, M. (2018). Automatic removal of complex shadows from indoor videos using transfer learning and dynamic thresholding. *Computers & Electrical Engineering*, *70*, 813-825.
92. Chen, Z., Zhang, T., & Ouyang, C. (2018). End-to-end airplane detection using transfer learning in remote sensing images. *Remote Sensing*, *10*(1), 139.
93. Singh, H. (2019). *Efficient Object Detection using Transfer Learning* (Doctoral dissertation).
94. Wang, Y., Wang, C., & Zhang, H. (2018). Combining a single shot multibox detector with transfer learning for ship detection using sentinel-1 SAR images. *Remote sensing letters*, *9*(8), 780-788.
95. Khan, S., Islam, N., Jan, Z., Din, I. U., & Rodrigues, J. J. C. (2019). A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, *125*, 1-6.
96. Huh, M., Agrawal, P., & Efros, A. A. (2016). What makes ImageNet good for transfer learning?. *arXiv preprint arXiv:1608.08614*.
97. Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 9.

98. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2921-2929).
99. Kwaśniewska, A., Rumiński, J., & Rad, P. (2017, July). Deep features class activation map for thermal face detection and tracking. In *2017 10th International Conference on Human System Interactions (HSI)* (pp. 41-47). IEEE.
100. Pericherla, S. R., Duvvuru, N., & Jayagopi, D. B. (2019, May). Improving Adversarial Images Using Activation Maps. In *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)* (pp. 843-847). IEEE.
101. Tang, W., Liu, B., & Yu, N. (2017, September). Visual Tracking by Deep Discriminative Map. In *Pacific Rim Conference on Multimedia* (pp. 733-742). Springer, Cham.
102. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 618-626).
103. Guo, H., Fan, X., & Wang, S. (2017). Human attribute recognition by refining attention heat map. *Pattern Recognition Letters*, *94*, 38-45.
104. Kumar, D., Wong, A., & Taylor, G. W. (2017). Explaining the unexplained: A class-enhanced attentive response (clear) approach to understanding deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 36-44).
105. Li, J., Lin, D., Wang, Y., Xu, G., & Ding, C. (2019). Deep Discriminative Representation Learning with Attention Map for Scene Classification. *arXiv preprint arXiv:1902.07967*.
106. Charuchinda, P., Kasetkasem, T., Kumazawa, I., & Chanwimaluang, T. (2019, March). On Building Detection Using the Class Activation Map: Case Study on a Landsat8 Image. In *2019 10th International Conference of*

*Information and Communication Technology for Embedded Systems (IC-ICTES)* (pp. 1-4). IEEE.

107. Vasu, B., Rahman, F. U., & Savakis, A. (2018, June). Aerial-cam: Salient structures and textures in network class activation maps of aerial imagery. In *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)* (pp. 1-5). IEEE.
108. Li, W., Jafari, O. H., & Rother, C. (2019). Localizing Common Objects Using Common Component Activation Map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 28-31).
109. Lu, Y., & Zhou, T. (2018). Lip segmentation using localized active contour model with automatic initial contour. *Neural Computing and Applications*, *29*(5), 1417-1424.
110. Tesema, F. B., Wu, H., & Zhu, W. (2018, March). Human Segmentation with Deep Contour-Aware Network. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence* (pp. 98-103). ACM.
111. Griffiths, D., & Boehm, J. (2019). Improving public data for building segmentation from Convolutional Neural Networks (CNNs) for fused airborne lidar and image data using active contours. *ISPRS Journal of Photogrammetry and Remote Sensing*, *154*, 70-83.
112. van den Brand, J., Ochs, M., & Mester, R. (2016, November). Instance-level segmentation of vehicles by deep contours. In *Asian Conference on Computer Vision* (pp. 477-492). Springer, Cham.
113. Hayder, Z., He, X., & Salzmann, M. (2017). Boundary-aware instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5696-5704).
114. Li, X., Yang, F., Cheng, H., Liu, W., & Shen, D. (2018). Contour knowledge transfer for salient object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 355-370).

115. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).
116. Novotny, D., Albanie, S., Larlus, D., & Vedaldi, A. (2018). Semi-convolutional operators for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 86-102).
117. Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8759-8768).
118. Yu, Y., Zhang, K., Yang, L., & Zhang, D. (2019). Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Computers and Electronics in Agriculture*, *163*, 104846.
119. Johnson, J. W. (2018). Adapting mask-rcnn for automatic nucleus segmentation. *arXiv preprint arXiv:1805.00500*.
120. Kawai, N., Sato, T., & Yokoya, N. (2015). Diminished reality based on image inpainting considering background geometry. *IEEE transactions on visualization and computer graphics*, *22*(3), 1236-1247.
121. Mori, S., & Saito, H. (2018). An overview of augmented visualization: observing the real world as desired. *APSIPA Transactions on Signal and Information Processing*, *7*.
122. Siltanen, S. (2017). Diminished reality for augmented reality interior design. *The Visual Computer*, *33*(2), 193-208.
123. Nakajima, Y., Mori, S., & Saito, H. (2017, October). Semantic object selection and detection for diminished reality based on SLAM with viewpoint class. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)* (pp. 338-343). IEEE.
124. FAST corner detection: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html
125. Centroid of blob: https://www.learnopencv.com/find-center-of-blob-centroid-using-opencv-cpp-python/

126. Contours: https://docs.opencv.org/3.4.0/d4/d73/tutorial_py_contours_begin.html
127. Canny Edge detection: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html
128. How Autonomous Vehicles Perceive and Navigate Their Surroundings: https://velodynelidar.com/newsroom/title-how-autonomous-vehicles-perceive-and-navigate-their-surroundings/

# Vita Auctoris

| | |
|---|---|
| NAME: | Foram Pravinkumar Patel |
| PLACE OF BIRTH: | Ahmedabad, India |
| YEAR OF BIRTH: | 1996 |
| EDUCATION: | Bachelor of Engineering, 2013-2017<br>Kadi Sarva Vishwavidyalaya,<br>Gandhinagar, Gujarat, India<br>Master of Science in Computer Science, 2018-2019<br>University of Windsor, Windsor, ON |