9-3-2019

# An Approach Of Automatic Reconstruction Of Building Models For Virtual Cities From Open Resources

Sumit Khairnar
*University of Windsor*

# AN APPROACH OF AUTOMATIC RECONSTRUCTION OF BUILDING MODELS FOR VIRTUAL CITIES FROM OPEN RESOURCES

By

**Sumit Khairnar**

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2019

# AN APPROACH OF AUTOMATIC RECONSTRUCTION OF BUILDING MODELS FOR VIRTUAL CITIES FROM OPEN RESOURCES

by

Sumit Khairnar

APPROVED BY:

Sep 3, 2019

## DECLARATION OF ORIGINALITY

## ABSTRACT

Along with the ever-increasing popularity of virtual reality technology in recent years, 3D city models have been used in different applications, such as urban planning, disaster management, tourism, entertainment, and video games. Currently, those models are mainly reconstructed from access-restricted data sources such as LiDAR point clouds, airborne images, satellite images, and UAV (uncrewed air vehicle) images with a focus on structural illustration of buildings' contours and layouts. To help make 3D models closer to their real-life counterparts, this thesis research proposes a new approach for the automatic reconstruction of building models from open resources. In this approach, first, building shapes are reconstructed by using the structural and geographic information retrievable from the open repository of OpenStreetMap (OSM). Later, images available from the street view of Google maps are used to extract information of the exterior appearance of buildings for texture mapping onto their boundaries. The constructed 3D environment is used as prior knowledge for the navigation purposes in a self-driving car. The static objects from the 3D model are compared with the real-time images of static objects to reduce the computation time by eliminating them from detection process.

## DEDICATION

*To my family and friends…*

## ACKNOWLEDGMENTS

I owe a debt of gratitude to Dr. Yuan, for the vision and foresight, which inspired me to conceive this thesis work. As my teacher and mentor, he has taught me more than I could ever give him credit for. I am also thankful to my thesis committee members, Dr. Ahmad, and Dr. Hlynka, for providing me extensive professional and personal guidance, which helped me learn a great deal about both scientific research and life in general.

Nobody has been more important to me in the pursuit of this thesis than my family members & colleagues. I would like to thank my parents; whose love and guidance are with me in whatever I pursue. They are the ultimate role models and the source of my inspiration. Most importantly, I wish to thank all the faculties and staff of the School of Computer Science and my friends who provided unending support and encouragement through my course work at the University of Windsor.

# TABLE OF CONTENTS.

DECLARATION OF ORIGINALITY .................................................................iii

ABSTRACT.....................................................................................................iv

DEDICATION..................................................................................................v

ACKNOWLEDGMENTS .................................................................................vi

LIST OF TABLES............................................................................................x

LIST OF ABBREVIATIONS/SYMBOLS.........................................................xi

LIST OF FIGURES .........................................................................................xii

**CHAPTER 1 INTRODUCTION**.........................................................1

1.1 Overview..........................................................................................1

1.2 Working of a self-driving car...........................................................2

1.3 Virtual 3D city model ......................................................................6

1.4 Texture mapping 3D city model ......................................................7

1.5 OpenStreetMap – VGI data ............................................................9

1.6 Thesis Contribution.........................................................................10

1.7 Structure of the thesis.....................................................................10

**CHAPTER 2 LITERATURE REVIEW**..............................................11

2.1 Construction of 3D city model........................................................11

2.1.1 Methods for 3D city modelling.......................................................11

2.1.2 Photogrammetry based 3D city models .........................................12

2.1.2.1 Aerial photogrammetry based model.............................................12

2.1.2.2 3D city model with aerial images and cadastral map .....................14

2.1.2.3 3D city model with computer vision techniques ............................15

2.1.2.4 3D city models with GIS ................................................................16

2.1.2.5 3D city modelling from satellite photogrammetry ..........................18

2.1.2.6 3D city modelling from single satellite image.................................18

2.1.2.7 3D city model from panorama images...........................................19

2.1.2.8 3D city model from video ....................19
2.1.2.9 3D city model from TLS data ....................20
2.1.3 Laser scanning based 3D models ....................21
2.1.3.1 Terrestrial laser-based approach ....................21
2.1.3.2 Aerial laser-based approach ....................22
2.1.3.3 3D city model with mobile mapping system ....................23
2.1.4 VGI data based 3D city models ....................24
2.1.5 Hybrid data-based 3D modeling ....................26
2.1.5.1 3D modelling with aerial laser and terrestrial laser ....................26
2.1.5.2 3D modelling with laser and photogrammetry ....................26
2.1.5.3 3D modelling with laser and videogrammetry ....................28
2.2 Texture mapping 3D city model ....................28
2.2.1 Photorealistic method ....................29
2.2.2 Pictometry based approach ....................30
2.2.3 Laser scanning based approach ....................31
2.2.4 Dynamic pulse function based method ....................32
2.3 Related works ....................36
2.4 Thesis statement ....................38
2.4.1 Problem statement ....................38

**CHAPTER 3 PROPOSED METHODOLOGY** ....................39

3.1 Motivation ....................39
3.2 Working of the overall system ....................39
3.2.1 Working of individual modules ....................41
3.3 Proposed methodology for construction of virtual environment ....................43
3.3.1 Construction of 3D building models ....................45
3.3.2 Geo-locating 3D environment ....................47
3.3.3 Texture image construction ....................48
3.3.4 Texture image mapping ....................53

**CHAPTER 4 IMPLEMENTATION AND EXPERIMENTS** ....................55

4.1 Software information ....................55
4.2 Data extraction from OSM server ....................55

4.3 Visualizing data in vector format ........ 56
4.4 Extruding footprints to 3D models ........ 58
4.5 Texture image extraction ........ 59
4.6 Reconstruction of texture image ........ 60
4.6.1 Perspective transform image ........ 60
4.6.2 Model-to-image comparison ........ 61
4.6.3 Occlusion removal ........ 62
4.7 Image stitching ........ 63
4.8 Texture mapping on the 3D model ........ 64
4.9 Size of the constructed model ........ 66
4.10 Use of 3D model in the overall system ........ 67
4.11 Result comparisons and discussions ........ 69
4.12 Drawbacks and limitations........ 71

**CHAPTER 5 CONCLUSION AND FUTURE WORK** ........ 72
5.1 Conclusion ........ 72
5.2 Future work ........ 73

REFERENCES/BIBLIOGRAPHY ........ 74

VITA AUCTORIS ........ 81

## LIST OF TABLES

Table 1: Tags used in OSM data structure……………………………………………9
Table 2: Summary of 3D reconstruction methods with texture information………33
Table 3: Review of 3D construction and texture mapping techniques……………36
Table 4: List of tools used for implementation and experiment……….…………..55

LIST OF ABBREVIATIONS/SYMBOLS

| | |
|---|---|
| LiDAR | Light Detection and Ranging |
| RADAR | Radio Detection and Ranging |
| IMU | Inertial Measurement Unit |
| GPS | Global Positioning System |
| UAV | Uncrewed Aerial Vehicle |
| 3D | 3-dimensional |
| 2D | 2-dimensional |
| OSM | OpenStreetMap |
| VGI | Volunteered Geographic Information |
| HD | High Definition |
| XML | Extensible Markup Language |
| GIS | Geographic Information System |
| VNG | Virtual Newcastle Gateshead |
| CC | Cyber City |
| TLS | Three-Line-Scanner |
| MMS | Mobile Mapping System |
| SOA | Service Oriented Architecture |
| CNN | Convolutional Neural Network |
| FCN | Fully Convolutional Network |
| DAE | Digital Asset Exchange |
| KML | Keyhole Markup Language |
| PID | Proportional Integral Derivative |
| MPC | Model Predictive Control |
| SAT-PP | Satellite Image Precision Processing |

LIST OF FIGURES

Figure 1.1 Working of a self-driving car ……………………………….……………………2

Figure 1.2 Computer vision through camera sensor ………………………………………3

Figure 1.3 Sensor fusion …………………………………………………………………...3

Figure 1.4 Path planning …………………………………………….…………………4

Figure 1.5 Control in a self-driving car ……………………………………………………5

Figure 1.6 Texture mapping importance ………………………………………….………8

Figure 1.7 A UAV for texture extraction ………...…………………………………...8

Figure 1.8 Node, way, relation ……………………………………………………………9

Figure 1.9 Representation of OSM data …………………………………………...…9

Figure 2.1 3D objects and buildings ……………………………………………………...13

Figure 2.2 3D model with aerial images and cadastral map ……………………………...14

Figure 2.3 3D model with stereo camera ………………………………………………....15

Figure 2.4 3D model with GIS data ……………………………………………………...17

Figure 2.5 VNG model in Autodesk LandXplorer ……………………………….…17

Figure 2.6 3D model from car images sequences …………………………………...19

Figure 2.7 3D environment from video with depth map …………………………………20

Figure 2.8 Data captured with TLS sensor ……………………………………………….21

Figure 2.9 3D city model with TLS data and CC Modeler ………………………………21

Figure 2.10 3D model based on laser scanning …………………………………………..22

Figure 2.11 3D model with LiDAR and image …………………………………………..22

Figure 2.12 3D model with point cloud and 3D city model with textures ……………….23

Figure 2.13 Setup for MMS and simulation of captured data …………………………....23

Figure 2.14 Level of detail ……………………………………………………………….24

Figure 2.15 OSM tags ……………………………………………………………………25

Figure 2.16 Extruded buildings from OSM ………………………………………………25

Figure 2.17 Combination of lasers; Virtual model; alignment …………………………...26

Figure 2.18 3D environment with laser and aerial images ……………………………….27

Figure 2.19 3D model of Berkley university ……………………………………….….27

Figure 2.20 Laser data and videogrammetry ………………………………………..28

Figure 2.21 Rectified image ………………………………………………………...30

Figure 2.22 Real-life object removal ………………………………………………..30

Figure 2.23 Flight path for pictometry ……………………………………………...30

Figure 2.24 Hidden angles in Pictomerty …………………………………………...31

Figure 2.25 Lase scanned data; texture image ……………………………………32

Figure 2.26 Dynamic pulse function …………………………………………...…...32

Figure 2.27 Final model of Karabuk university …………………………….…33

Figure 3.1 Overall system ………………………………………………………….40

Figure 3.2 Flowchart for proposed methodology …………………………………...44

Figure 3.3 Region to extract ………………………………………………………...45

Figure 3.4 Flowchart for 3D building models ……………………………………46

Figure 3.5 Pseudocode for building extrusion ……………………………………47

Figure 3.6 Georeferencing virtual 3D model ……………………………………48

Figure 3.7 Process of image extraction and reconstruction ………………………50

Figure 3.8 Texture mapping process ………………………………………………..53

Figure 4.1 Bounding box query ……………………………………………………..56

Figure 4.2 Vector format of XML data ……………………………………………..57

Figure 4.3 Footprints of OSM data ………………………………………………57

Figure 4.4 Extruded building models ……………………………………………….58

Figure 4.5 KML file for geo-locations ……………………………………………...58

Figure 4.6 Façade texture image …………………………………………………59

Figure 4.7 Perspective transform …………………………………………………60

Figure 4.8 Rendered image of 3D model …………………………………………...61

Figure 4.9 Model-to-image comparison ……………………………………………61

Figure 4.10 Background subtracted image …………………………………………62

Figure 4.11 Masked objects for removal ……………………………………………62

Figure 4.12 Occlusion removed façade texture …………………………………..………63

Figure 4.13 Three façade images to stitch ….……………………………..………...63

Figure 4.14 Stitched image ……………………………………………………………64

Figure 4.15 Textured 3D model (view 1) …………...……………………………………65

Figure 4.16 Textured 3D model (view 2) ………………………………………………...65

Figure 4.17 Textured 3D model (view 3) ………………………………………………...66

Figure 4.18 3D model geometry comparison ……………………………………………66

Figure 4.19 Building detection …………………………………………………………...67

Figure 4.20 3D features of building ……………...………………………………………68

Figure 4.21 3D features of building………………………….…………………………...68

Figure 4.22 3D features of building ………...……………………………………………68

Figure 4.23 Real-time image of 3D objects ………………………………………………70

Figure 4.24  Virtual image of 3D objects ………………………………………………...70

Figure 4.25 Drawbacks of texture image ………………………………………………...71

# CHAPTER 1
# INTRODUCTION

## 1.1 Overview

We have come to an era where self-driving cars are practically feasible. In recent years self-driving cars have gone from “maybe possible” to “definitely possible” to “inevitable” to “how did anyone ever think this wasn’t inevitable?” to “now commercially available” [53]. Besides Google, Uber, and Tesla, there are many other companies who have invested large sums of money into the research of this system. With such a huge amount of investment and interest in the development of self-driving systems, we can understand that self-driving automobiles are an inevitable future. But at their current stage, they are far from being viable [51]. With the levels of sophistication involved, a high cadre of expertise is required to ensure the smooth functioning of the conventional self-driving car.

Awareness of the nearby environment is necessary to ensure the perfect functioning of a self-driving car. To help the car familiarize itself with its surroundings, a virtual environment can be constructed. It would comprise of static objects that the car may interact with while driving. Several companies are working towards the construction of a 3D or HD map for self-driving cars. The sole purpose of these maps is to familiarize the car with ever-existing objects on the road, which are classified as static objects. Darms et al. [49] and Hu et al. [50] have defined *static objects* as those that do not move when the car is on the road. These include buildings and roadside amenities such as benches, poles, etc. We shall use the same definition for the extent of this thesis.

The main aim of our work is to construct a virtual environment consisting of most of the static objects that a car interacts with, when it is on the road.

## 1.2 Working of a self-driving car

A self-driving car is a vehicle which can sense the surroundings and can navigate without human input [53]. Self-driving cars can detect the environment with the help of a variety of sensors, such as LiDAR, RADAR, Camera, GPS, and IMU.

1) Camera: It functions as the eyes of a self-driving car. It helps detect objects found on the road.

2) RADAR: **RA**dio **D**etection **A**nd **R**anging uses radio waves to find the range, angle, and velocity of objects.

3) LiDAR: **Li**ght **D**etection **A**nd **R**anging measures the distance to the target by penetrating a laser beam towards the object. LiDAR is used to generate a 3D map of the surroundings.

4) IMU-enabled GPS: IMU stands for **I**nertial **M**easurement **U**nit, which provides the angular rate and orientation of a body. IMU-enabled GPS is used when GPS is unavailable.

5) Others: Other sensors used are infrared cameras, 360-degree cameras

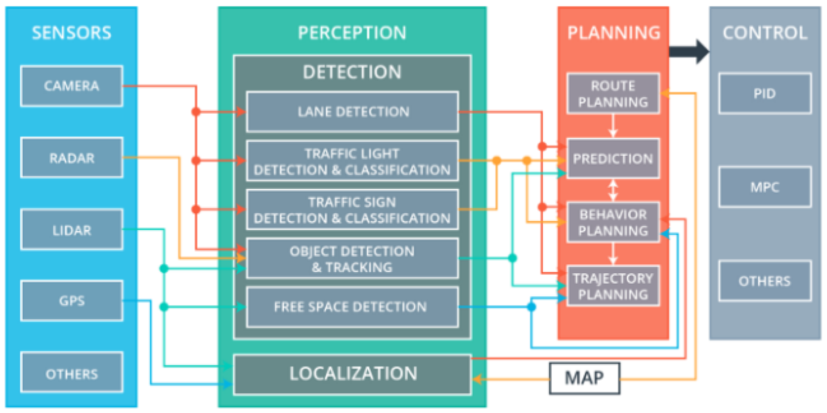


Figure 1.1. Working of self-driving car [53]

The work system of a self-driving car has been divided into five different components. These components use the information provided by the above-mentioned sensors to control the car.

## Computer Vision

Computer vision relates to how cameras are used to see the road and detect objects on the road. With the help of camera images, object identification is performed, and the pose of the detected object is estimated. Once identified, the objects are classified and tracked for better understating of the surroundings.

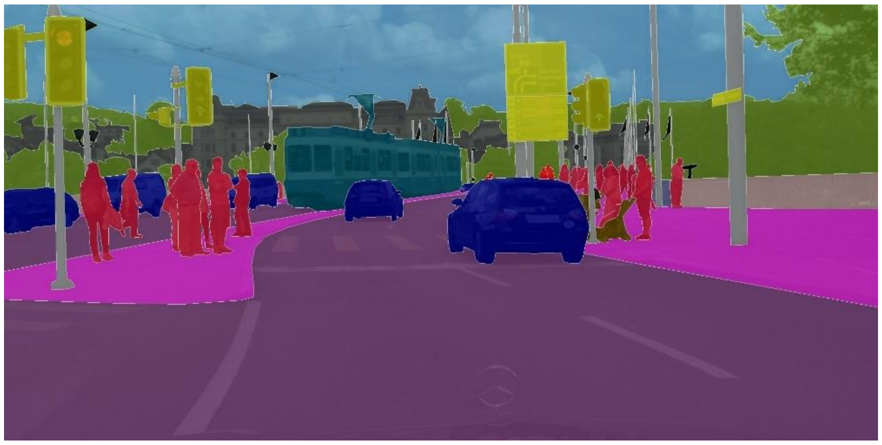


Figure 1.2. Computer vision through camera sensor

## Sensor Fusion

Sensor fusion is the process of integrating the data received from different sensors to build a detailed understanding of the car’s nearby environment [55].

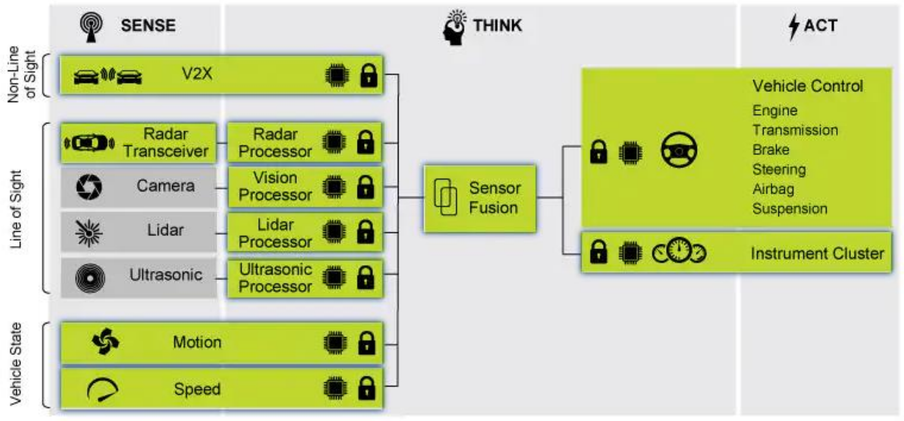


Figure 1.3. Sensor fusion [53]

Sensor fusion is required because data received from a single sensor is of little use when compared to the combined data [53]. Data received from cameras are better for object detection, but to measure the distance between the objects and the car, radar sensor gives better results than a camera. Radar provides better results due to the waves penetrating through the Radar sensor. This function helps it outperform the camera even in bad weather conditions.

All data received from different sensors is fused. It is then used to help the car understand better. As seen in Figure 1.3, the sensors mentioned in the line of sight module consist of LiDAR, RADAR, camera, and ultrasonic sensors, which are all fused together in the sensor fusion module.

## Localization

Localization is to know the position of the car in the real world. This information can be obtained once the environment is known. GPS sensor is used to acquire the real-world location of the car; also, mathematical algorithms such as Kalman Filter, Extended Kalman Filter, and Unscented Kalman Filter are used to make the GPS information more accurate.

## Path Planning

Path planning enables a self-driving car to find the fastest, the safest, and the most convenient route from the start point to the endpoint [54]. The car needs to detect all the static and dynamic objects (maneuverable) to bypass them. This makes path finding complicated.

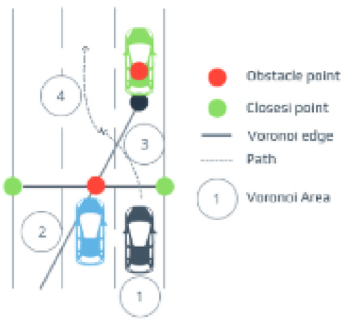


Figure 1.4. Path planning [54]

Major approaches for path planning are predictive control model, behavior-based model, and feasible model. Before crafting the trajectory plan, the path of the car, the path planning, the car maneuver, and the maneuver planning of the car is considered. Any dynamic object is referred to as a maneuver.

## Control

Control is the final step in the working system of the self-driving car. Once the trajectory for the start point to the end point is defined, vehicle control such as turning the steering wheel, hitting the throttle or brakes, is controlled.

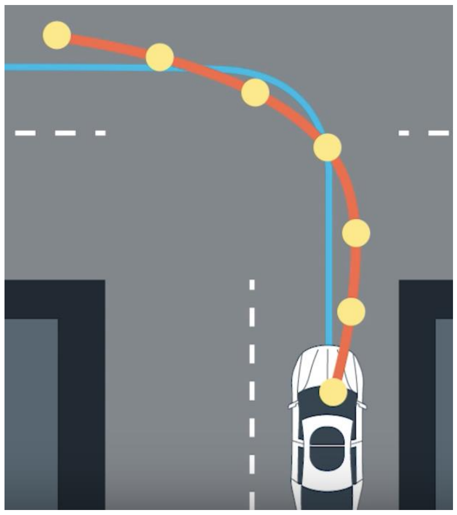


Figure 1.5. Control in self-driving car [56]

There are different types of controllers which are used in the system; some of them are, Proportional Integral Derivative (PID), Model Predictive Control (MPC), kinematic model, and dynamic model [56]. All these controllers take angle (yaw) and speed (v) into account while defining the trajectory.

As seen in Section 1.2, there are many different types of sensors used in a self-driving car. Also, with the addition of every sensor, the cost of the self-driving car increases. The total sensor cost of Google's self-driving car is around $150,000, of which the cost of the LiDAR sensor alone is around $70,000 [57]. The main function of the LiDAR sensor is to generate a 3D map of the surroundings. Later, this is combined with the information of other sensors, and the functioning is controlled.

This research mainly concentrates on the construction of a virtual 3D environment which can serve as prior information to the car, particularly, to use the VGI (Volunteered geographic information) data, which is open source. The construction and use of the 3D environment also reduce the overall cost of the self-driving car as it eliminates the use of the LiDAR sensor, which makes up half the cost of all the sensors combined.

## 1.3 Virtual 3D city model

Virtual 3D city models are used in an increasing number of applications, including landscape planning, disaster management, location-based services, tourism industry, and urban planning. Virtual 3D city models are an important visualization of urban geospatial and georeferenced information. The 3D model enables the visual representation of past and existing cities. It also provides a decision on whether a redevelopment of existing cities is required. Virtual 3D city models are not only restricted with visualization purposes but also provide data basis for spatial querying of thematic data, for computational models in urban security, and for noise propagation models [60].

In general, the term virtual 3D city models refers to a digital representation of different entities of the city and their geometric and topologic structures. A virtual 3D city model comprises of:

- Buildings,
- Vegetation objects such as trees and hedges,
- City furniture such as benches, night lamps,
- Water bodies such as rivers or lakes,
- Terrain surface

The usage of the information provided by the 3D model depends on the application domain.

> ‘*Every second producer has requests to provide other objects or information than he is presently producing, and three out of four users would like to have other city data than already available.’* [59]

For instance, in urban planning, realistic texture is required on 3D models as visual details are essential in urban redevelopment. Whereas, the tourism industry requires 3D models with thematic details such as the description of a historic monument, a popular landmark, or a restaurant. Hence, the question regarding the detailing of a 3D model remains not fully answered.

In general, a 3D city model is called ‘virtual environment’, ‘virtual 3D model’, ‘virtual 3D environment’, ‘3D model’, and ‘virtual city.’ All these names have been used in this thesis to refer to the 3D city model.

The construction of the 3D city model is carried out with the use of different types of data including sensory information, GIS data, CAD data, Building Information Model data, and VGI data. Sensors used in the construction of the 3D model are primarily LiDAR and cameras. LiDAR captures the 3D geometric information. Cameras capture the façade textures. The data captured with LiDAR sensor is visualized as 3D point cloud. Then, pre-processed to find the planar facades in point cloud. GIS data is a type of thematic data. It contains all the information in textual format. Building Information Model are pre-constructed models of existing buildings from the city. They are accurate with the complex construction. VGI data is crowdsourced data which is mapped by public. VGI data is mainly available in thematic format or visual format.

When integrated, there are two ways in which 3D models supports different applications. For some scenarios, 3D model provides the context for spatial information. For others, it works as a base model or prior information on which different applications perform.

## 1.4 Texture mapping 3D city model

‘Texture mapping is a powerful and flexible low-level graphics drawing primitive’ [58]. Texture provides means to store visual and thematic information for the 3D models of buildings and other components existing in the virtual environment. Particularly, textures allow the specification of the visual appearances of the facades of models.

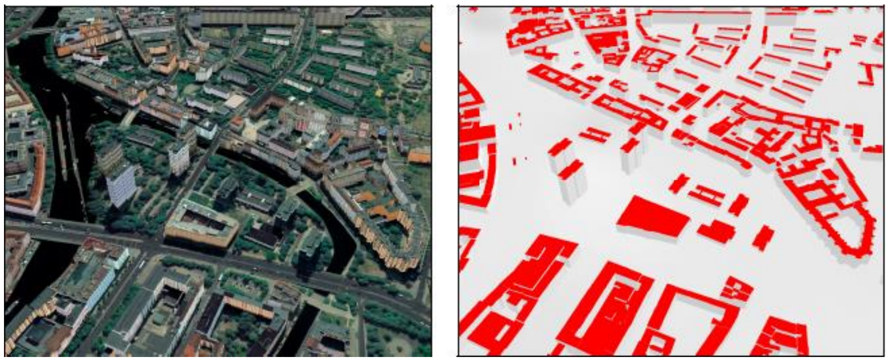


Figure 1.6. Texture mapping importance [58]

Mapping the texture is essential in the construction of the 3D city model that can be seen in Figure 1.6. The image on the left side contains actual textures, whereas the image on the right does not have any textures mapped.

Textures are constructed by either capturing from real sources or by digitally creating them. Textures captured from the real location are known as photorealistic textures. Photorealistic textures are captured with Airborne cameras and Airborne LiDAR sensors. Also, GPS sensor is combined with the whole setup, which helps geo-reference the captured images.

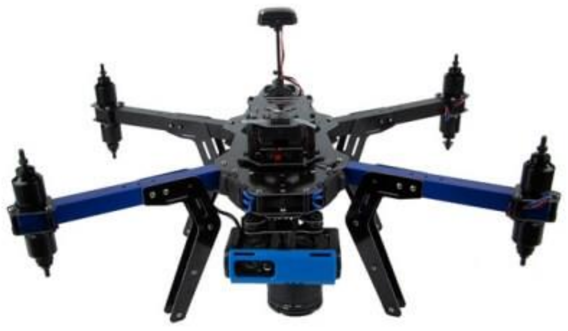


Figure 1.7: A UAV for texture extraction

As seen in Figure 1.7, a UAV is mounted with a camera and a LiDAR sensor with GPS, to capture the façade and roof textures. The other main approaches are Pictometry based texture mapping and Photorealistic texture approach, which are discussed in detail in Chapter 2.

## 1.5 OpenStreetMap – VGI data

OpenStreetMap is a crowdsourced mapping technology started in 2004 as a university project. The structure of OSM data is XML based, also, it uses tag information for mapping real-world locations. The structure of XML data is based on three primitives [61].

1. Node: a point in real-world with latitude and longitude
2. Way: an ordered list of nodes, such as a line; it represents linear features such as a stream or a railway
3. Relations: refers to an ordered sequence of way or nodes

In Figure 1.8, nodes, ways, and relations are shown, from left to right respectively.

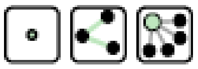


Figure 1.8. Node, Way, Relation (left to right order) [61]

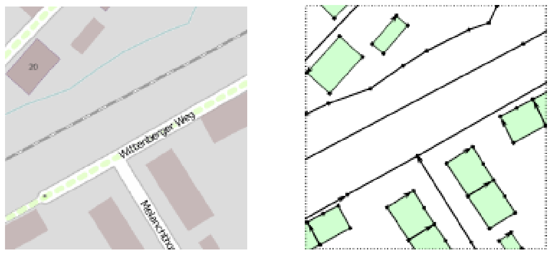


Figure 1.9: Representation of OSM data [61]

In Figure 1.9, in the left-side is the visual representation of OSM data on openstreetmap.org, and the right-side image is an actual mapping of the data [61].

Some of the important tags in OSM structure

| | |
|---|---|
| *height* | Describes the Height of a feature |
| *length* | Describes the Length of a feature |
| *width* | Width of a feature, not used in ways |
| *material* | Material of which the feature is made of |
| *surface* | Surface material of the feature |
| *building* | To mark the feature as a building; the value is set to yes or some specific type of building as hut or a garage |
| *building:part* | To model different areas of a building |

| | |
|---|---|
| *height* | To mention the height of the feature |
| *building:levels* | To mention the number of floors in the building |

Table 1.1: Tags used in OSM data structure

In Table 1.1, tags used to map an entity are shown. Similarly, *elevation, building:material, building:entrance, building:amenities, and sidewalk* are also used in some specific cases.

## 1.6 Thesis Contribution

Major contributions of this research work can be summarized as follows:

- 3D city model constructed with the proposed system helps reduce computation time of object detection by eliminating static objects (objects that do not move while the car is on the road)
- Self-driving car can be geo-localized in real environment by comparing real-time images with rendered images of the virtual world
- Proposed research work does not include the use of any sensors (mentioned in Section 1.2) to extract 3D structural information or to construct the environment.
- Once detected, all the dynamic objects can be updated into the virtual environment and can be used for better navigation of the self-driving car.

## 1.7 Structure of the thesis

The remainder of the thesis is structured as follows.

Chapter 2 provides technical details of construction of the 3D city model, texture extraction for the façade of the 3D model, and reconstruction of the texture image.

Chapter 3 discusses the proposed system for the construction of the 3D model, and texture extraction with mapping, details of the overall system and connection of this thesis work with the overall system.

Chapter 4 gives a detailed description of the implementation setup, experiments conducted, and the usage of the 3D model in the overall system.

Chapter 5 provides summary concluding the thesis with the direction of possible future work.

# CHAPTER 2
# LITERATURE REVIEW

This chapter discusses the relevant background of recent works in the construction of the 3D city model and texture mapping. This section also covers the technical background of 3D city models and texture extraction and mapping using different data sources.

## 2.1 Construction of 3D city model

Nowadays, 3D city modeling has become an important issue for researchers in the area of geomatics. Geomatical techniques play a key role in the construction of a 3D city model. For mapping technologies geomatics is an umbrella which consists of technologies like Photogrammetry, Geographical Information System, Remote sensing, Lasergrammetry, Global Positioning System, and Radargrammetry. Mainly in the construction of 3D environment, laser techniques and Photogrammetry are used. Singh S.P. et al. [1] conducted a review related to the usage of 3D city models in various applications. They presented the most representative geomatical technique for 3D city modeling and other related works of researchers.

### 2.1.1 Methods for 3D city modelling

Current methods are mainly categorized in the following approaches.

- Based on Automation
    1. Automatic
    2. Semi-automatic
    3. Manual
- Based on Data Input
    1. Photogrammetry based technique
    2. Laser scanning based technique
    3. VGI (Volunteered Geographic Information) based technique
    4. Hybrid data input-based technique

A further categorization of the methods based on data input is as follows.

- Photogrammetry based technique
    1. Aerial Photogrammetry based technique
    2. Aerial image and cadastral map based technique
    3. Computer vision based techniques
    4. GIS data based techniques
    5. Satellite Photogrammetry based technique
    6. Single satellite images based technique
    7. Panorama image based technique
    8. Video based technique
    9. TLS data based technique
- Laser scanning based technique
    1. Terrestrial Laser-based technique
    2. Aerial Laser-based technique
    3. Mobile mapping system based technique
- VGI data-based technique
    1. With the use of Government open data
    2. With the use of OpenStreetMap data
- Hybrid data input-based method
    1. Aerial Laser and Terrestrial Laser
    2. Laser and Photogrammetry
    3. Laser and Videogrammetry

### 2.1.2 Photogrammetry based 3D city models

#### 2.1.2.1 Aerial photogrammetry-based model

Nowadays, airborne data is mostly used to collect 3D structural information. Aerial photos are mostly used as raw data. With the help of stereo-pair images, a 3D point cloud is constructed. This type of semi-automatic method for the acquisition of 3D structural data from 2D aerial stereo images is presented in [2]. In the approach, digital photogrammetric workstation (Traster T10) was used alongside Microstation CAD package, and Consob (inhouse developed software); researchers used digital aerial images of Netherlands in the scale of 1:2200.

Data acquisition, superimposition, processing, updating, and visualization are the main processes in this work. Figure 2.1 demonstrates the reconstructed 3D objects and buildings developed with the approach in [2].

The relation between Photogrammetry and 3D city modeling was provided by Kobayashi [3].

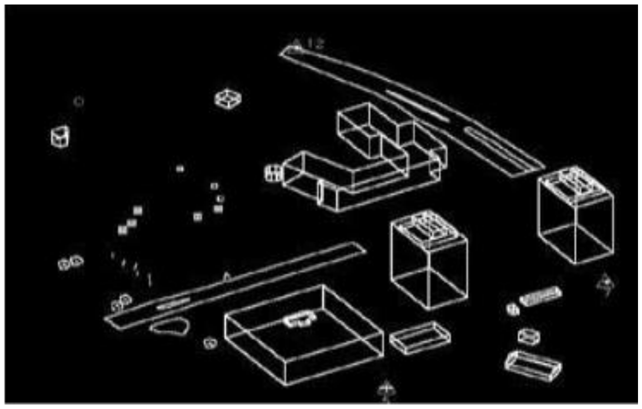


Figure 2.1. 3D objects and buildings [2]

The author also recommended a method to construct a 3D city model with the use of Photogrammetric processing. Along with Photogrammetry techniques, the author used aerial images for the construction of 3D city model and discussed the effectiveness and efficiency of the 3D model in terms of labor and reusability. The 3D construction model was based at Phoenix, USA.

Shashi and Jain [4] explored the use of Photogrammetry in scene visualization and in the construction of a 3D city model. A digital camera was used in the construction. In their approach, they perform close-range photogrammetric processing for better accuracy. The usage of digital camera instead of aerial or high-resolution cameras reduces the cost of the approach. This stands out as the chief benefit of their work. The authors also conclude that close-range photogrammetry gives the best solution for the construction of a 3D city model.

Leberl et al. [5] compared the point cloud generated from an image and point cloud generated with a laser system. They discuss the advantages and disadvantages of both the methods and conclude that the accuracy of the 3D point cloud is better when generated with photogrammetric methods.

Amat et al. [6] discussed a methodology to construct a 3D city model with the help of aerial images and close-range photography. In this method, the authors suggest that small 3D buildings, doors, windows are not visible from aerial photography, so close-range images are used to extract façade of the buildings. Certainly, with the combination of close-range photography and aerial photogrammetric techniques, 3D city model can be constructed.

Hammoudi and Dornaika [7] also provided a method for the construction of a 3D city model with aerial images. For this approach, geometric and photometric properties are used in perspective projection. The main advantage of the method is its use of direct raw images (without any pre-processing) and featurelessness. As a part of pre-processing, feature extraction and feature matching are avoided. An objective function is used to combine the dissimilarity between the captured images.

#### 2.1.2.2 3D city model with aerial images and cadastral map

Flamanc et al. [8] constructed a 3D city model by using the aerial images and cadastral maps. They also tested the approach with model-driven and test-driven systems. They used the cadastral map to extract information such as positions of existing structures of buildings, adjoining or adjacent streets, and dimensions.

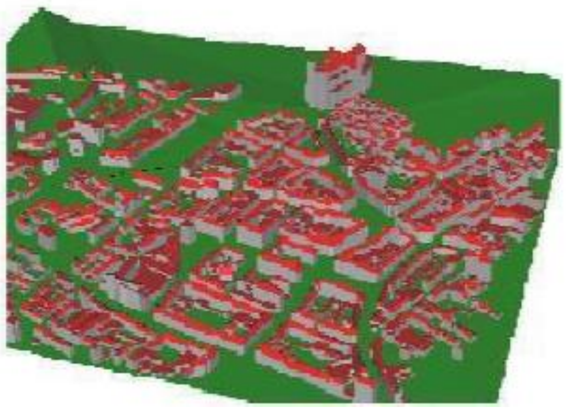


Figure 2.2. 3D model with aerial images and a cadastral map [8]

Figure 2.2 presents the 3D model constructed with the help of aerial images and a cadastral map.

### 2.1.2.3 3D city model with computer vision techniques

Lang and Forstner [9] suggested a semi-automatic technique for the acquisition of the 3D shape of buildings. Stereo cameras are used for extraction. The acquired information of 3D shapes is topographical. Figure 2.3 describes a 3D model constructed with a stereo camera.

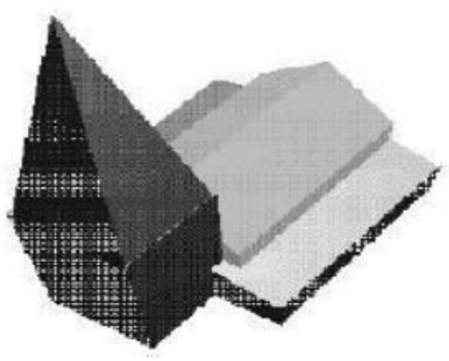


Figure 2.3. 3D model with a stereo camera [9]

Pollefeys et al. [10] provided an automated method to construct a textured 3D model from a sequence of images. Computer vision algorithms are used to construct the 3D city model. The accuracy of the model is not the best due to the use of computer vision algorithms. Hence, this approach is mainly used in archaeology and not in metrology. Authors tested this approach on the Roman site, Sagalassos, Turkey.

Jang and Jung [11] used ground images to construct a 3D city model. To capture ground images, they used a digital camera mounted with a compass and GPS sensor. All the captured images are referenced to the real-world coordinates with the help of Global Positioning System. To correct the pose of the images, they use Structure from Motion-based algorithm; they also register the 3D model with real coordinates with GPS.

Jurgen Dollner el al. [12] presented a pipeline to construct a virtual 3D city model handling tasks such as integration, managing, presenting, and distributing urban information. As input to the pipeline, Geo-referenced thematic data, Cadastral data, Digital Aerial photos, Digital Terrain Model, and 3D Geodata is passed. At the end of the process, the virtual 3D city model is constructed.

Cornelis et al. [13] discussed a method for constructing a 3D city model in real-time. Two cameras are used to record the calibrated videos, which work as an input to the system. They used Structure from Motion algorithm concept. They also use object detection algorithms for video recordings. The main feature of this method is that it constructs a virtual 3D city model in real-time.

Snavely Noah et al. [14] created a new method for the construction of a 3D city model by using the images available on the internet and introduced a new concept of photo-tourism. Authors use unordered images of the real-world site and construct the 3D model from the downloaded images. This method also uses Structure from Motion algorithms along with image-based rendering. This approach was tested on Google images of Notre Dame, Mount Rushmore, South Dakota, Sphinx (Giza), Colosseum located in Rome, and Great Wall of China.

Jianxiong Xiao et al. [15] provided an approach for automatic reconstruction of 3D models from street-side photos. Street-side photos are captured with a ground-level digital camera. To regularize the noisy and missing data, inverse-patch based composition method is used. Due to ground-level image capturing, skyscraper buildings are not modeled in this approach.

#### 2.1.2.4 3D city models with GIS

Gruen A. and Xinhua W. [16] developed a software named CyberCity Modeler, for automatic generation of a 3D point cloud. It has been developed in such a way that it generates structured data for city modeling from photogrammetrically measured points. This has been mainly designed to handle GIS data and to integrate raster images and vector data as hybrid GIS.

Nedal Al-Hanbali et al. [17] worked on constructing a 3D model of Artemis temple and Jerash City. Authors used photogrammetry principles and GIS data for the construction of the 3D model. The 3D GIS model was accurate in measurement, therefore used for visualization, preservation, and reconstruction of the temple in the city.

Nedal Al-Hanbali et al. [18] created a 3D model for Yarmouk University with the help of GIS Data and Photogrammetric techniques.

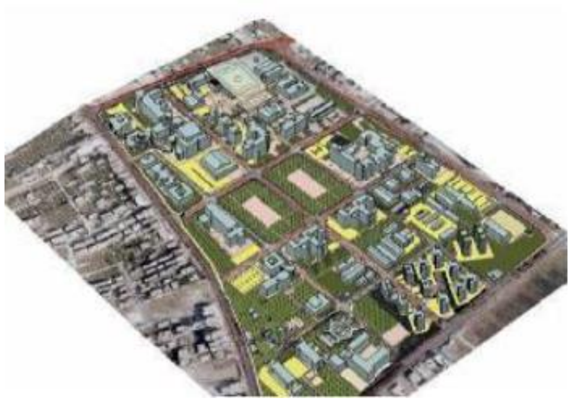


Figure 2.4. 3D model with GIS data [18]

Malumpong C. and Chen X. [19] used interoperable 3D GIS data with a 3D modeling software named Google Sketchup. The aim of the work was to integrate 3D GIS information with 3D modeling software to construct the virtual 3D city model and other objects of the city.

Razzak A. et al. [20] also proposed a technique for the construction of a 3D city model from 3D GIS data. Authors followed the GIS techniques for the virtual environment, and the final result contained all the objects visualized for users.

Thompson and Horne [21] worked on a VNG project, which focused on data exchange, CityGML, interoperability, and data accessibility issues in Autodesk LandXplorer software.

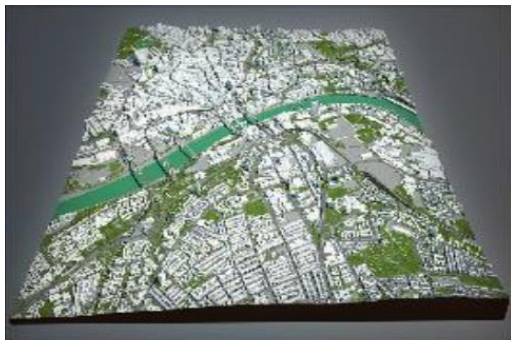


Figure 2.5. VNG model in Autodesk LandXplorer [21]

#### 2.1.2.5 3D City modelling from satellite photogrammetry

Tao and Young Hu [22] evaluated the concept of RFM in 3D reconstruction, due to which the generation of Digital Elevation Model is possible without physical sensor model. In the evaluation, authors studied two methods: forward RFM, and inverse RFM. They concluded that reconstruction accuracy is better with forward RFM. The approach was tested with real IKONOS stereo pairs to construct the 3D model.

Fraser et al. [23] discussed the use of IKONOS imagery for the extraction of buildings and positioning. They assessed the model with qualitative and quantitative approaches to construct the 3D model of the campus of the University of Melbourne.

Kocaman et al. [24] tested an approach for 3D city modeling with the use of high-resolution satellite images in SAT-PP software and CyberCity Modeler [16]. They extracted buildings and DSMs to construct 3D city model with IKONOS and stereo images.

Tack et al. [25] introduced and tested an approach for semi-automatic construction of 3D city model with tri-stereoscopic high-resolution satellite images. They also studied IKONOS triplet data with photogrammetry software named SAT-PP for Istanbul.

#### 2.1.2.6 3D city modelling from single satellite image

Huang et al. [26] developed a method for reconstruction of 3D objects from a single high-resolution satellite image. The geometry was constructed with the help of Rational Polynomial Coefficient method by using one high-resolution satellite image and Digital Elevation Model. The satellite ray determines the polynomials and shadows on the ground determines the azimuth of the object. Height of the objects is determined from the shadow size and its angle with the Sun. Authors tested the method on IKONOS image data and designed software for real-time reconstruction, extraction, and visualization.

Izadi and Saeedi [27] discussed a method for 3D building model with data input as a single satellite image. They designed a method for automatic building detection and height estimation with the shape of the roofs with the help of a single satellite image. The system detects multiple buildings without angular constraints, with the accuracy of 94% and a height error of 0.53 m on satellite image dataset.

### 2.1.2.7 3D city model from panorama images

Luhmann and Tecklenburg [28] suggested a method for 3D objects reconstruction by using multiple panorama images. In this work, they also discussed frame by frame panorama generation, image acquisition, calibration, and control point measurement. Each of the panorama images is oriented with the global coordinate system. By following an object reconstruction method such as space intersection and moving floating mark, the 3D model is constructed. Authors tested the approach by constructing the entrance hall of the University of Applied Sciences in Oldenburg.

Koseck and Micusik [29] designed an approach of 3D modeling by using street-view panoramic images with piecewise planar methodology. The images are captured with a camera calibrated in a car. The images are mapped one after one, and the 3D model is constructed.

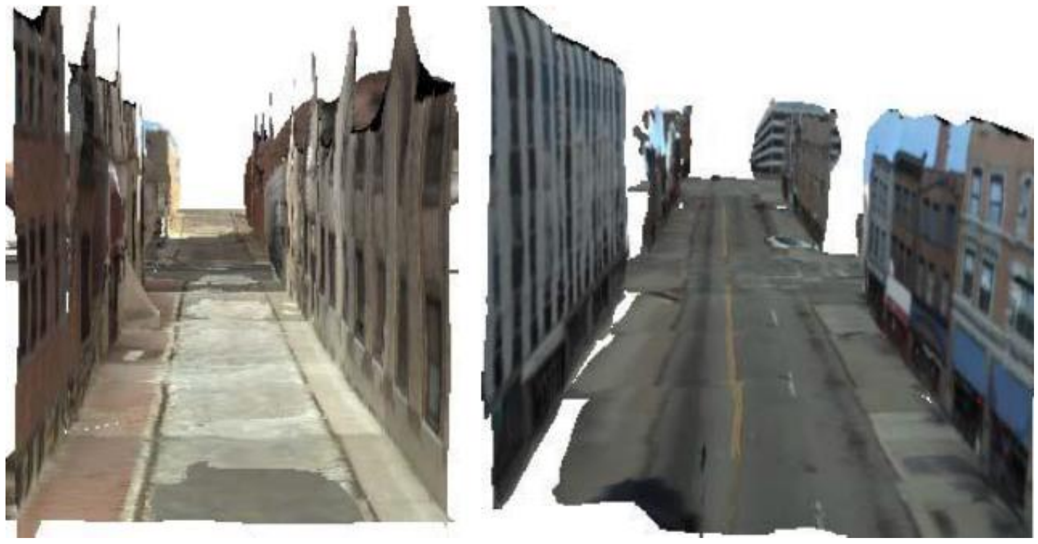


Figure 2.6. 3D model form car image sequences [29]

### 2.1.2.8 3D city model from video

The concept of constructing a 3D city model with videos is known as Videogrammetry. Videos are captured with a still digital camera or a camera recorder.

Clip et al. [30] designed a system for 3D city modeling; the scene capturing is mobile in the system. The setup can be used with backpack or mounted on a car. The setup also consists of GPS and IMU to geo-reference the captured scenes. After capturing, computer vision techniques are used to construct the final 3D city model from the videos.

Zhang et al. [31] provided a concept for Depth Map Recovery from a sequence of videos. In the process, each frame is divided into images, and a Depth Map is constructed. To recover the camera parameters, bundle optimization, and disparity initialization Structure-from-motion is used; space-time fusion techniques are used to generate depth maps. Figure 2.7 shows the results of the approach in depth map format.

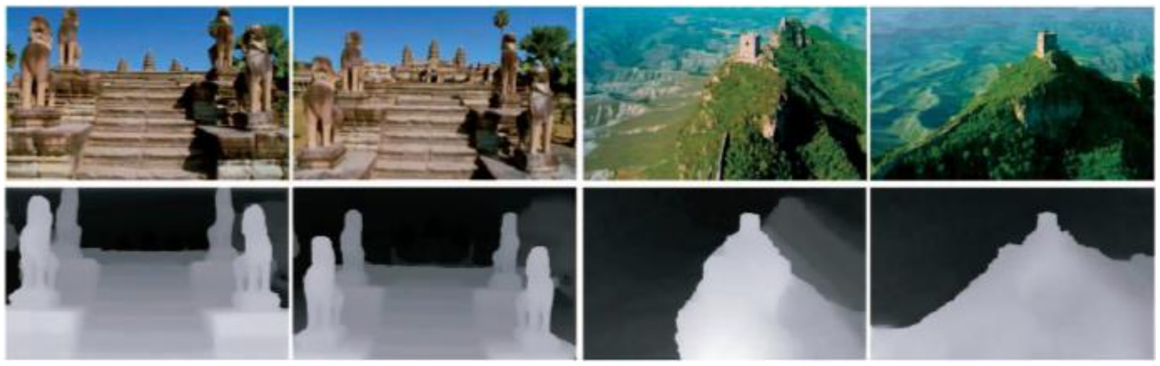


Figure 2.7. 3D Environment from video with depth map [31]

**2.1.2.9 3D city model from TLS data**

The TLS is an aerial multispectral digital sensor system by STARLABO, Tokyo. Three-Line-Scanner is a principle which is used to capture image triplets. It captures very high-resolution images of three points, forward, backward, and nadir.

Gruen et al. [32] used the TLS sensor to construct a 3D city model. Authors imported the TLS data in CC Modeler [16] and produced two datasets for Yokohama city. In Figure 2.8, the left image shows the area captured with the TLS sensor, and image in the right shows the detailed view of the same area. After importing the data in CC Modeler, the point cloud is generated for the TLS data. With the combination of CC Modeler and TLS stereoscopic measurement, 3D environment is constructed. In Figure 2.9, both views present the constructed 3D city with TLS sensor data and CC Modeler.

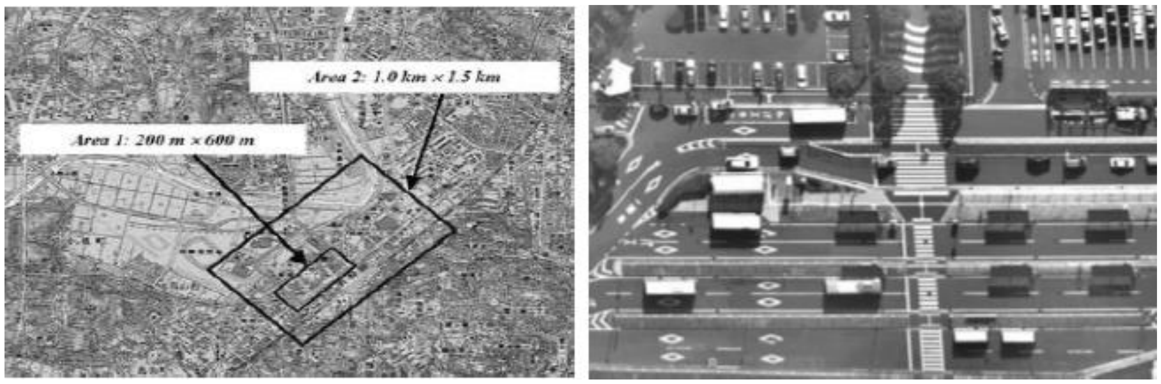


Figure 2.8. Data captured with the TLS sensor [32]

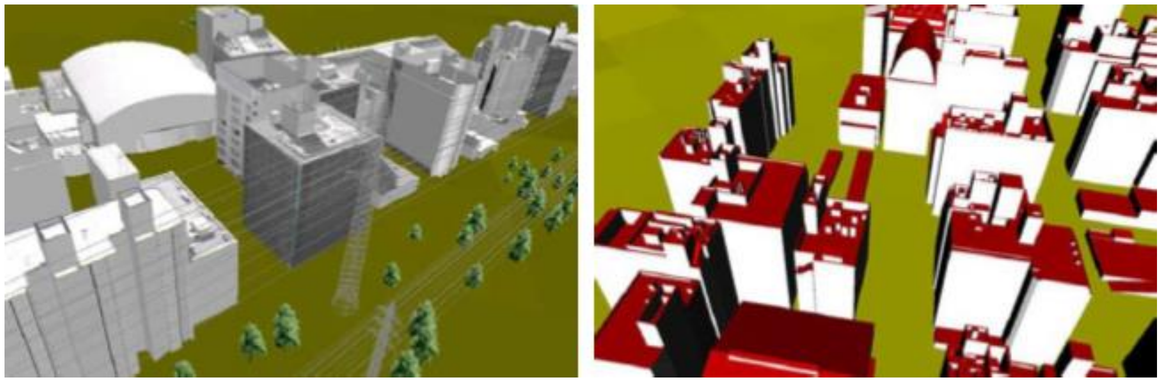


Figure 2.9. 3D city model with TLS data and CC Modeler [32]

### 2.1.3 Laser scanning based 3D models

#### 2.1.3.1 Terrestrial laser-based approach

Vosselman and Dijkman [33] proposed a method for construction of a 3D model with the use of 3d point cloud and ground plan. In this work, Hough transform method is used to extract planar surfaces from irregular point clouds. The algorithm has two faces; first, the intersection between lines and height jump edges is detected. In the second, models are refined by fitting the point cloud. Figure 2.10 shows the constructed model with this approach.

Ming et al. [34] proposed a method for automatic reconstruction of 3D city model by using LiDAR data and images from ground level.

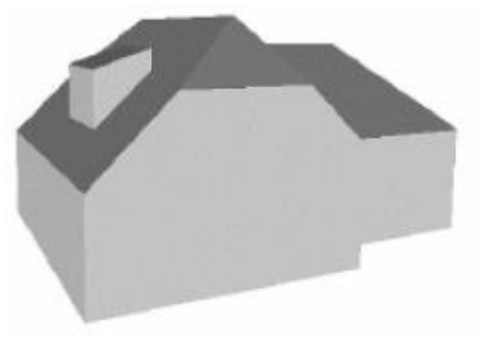


Figure 2.10. A 3D model based on laser scanning [33]

A 3D point cloud is generated with the help of LiDAR. Automatic plane detection is performed for the detection of a planar surface. Target recognition is performed for geo-referencing the model. This approach is used to construct the 3D city model with LiDAR data and digital imagery.

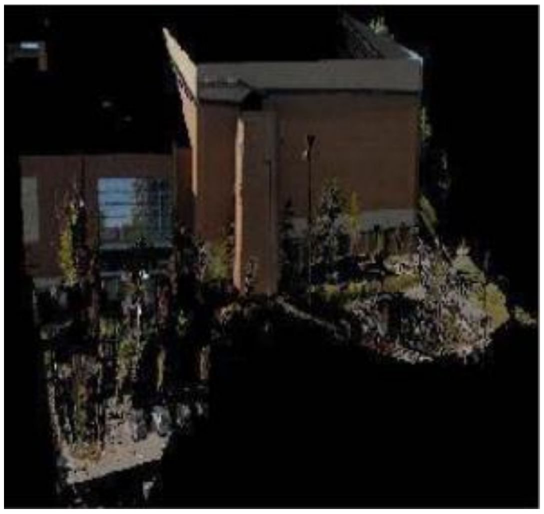


Figure 2.11. A 3D model with LiDAR and image [34]

### 2.1.3.2 Aerial laser-based approach

Dorninger and Pfeifer [35] designed an approach to determine 3D city models from the airborne laser-generated point cloud, from which they extracted building models and reconstructed them. An important aspect of this work is the detection of planar faces in the point cloud, which is used to detect the faces of 3D objects. Thus, primarily, this approach is based on 3D segmentation. In Figure 2.12, the left image shows the 3D model constructed with the point cloud, and image in the right shows the final construction with textures, extracted with an airborne camera.

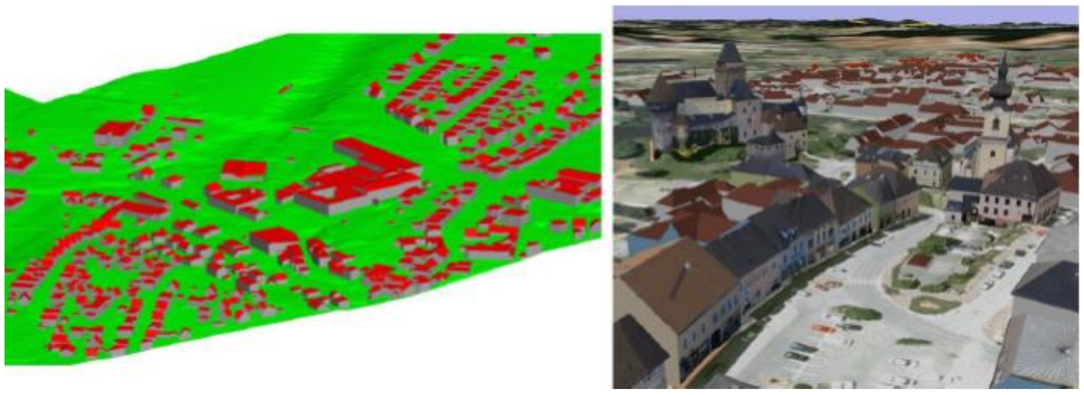


Figure 2.12. 3D model with point cloud (left); 3D city model with textures (right) [35]

### 2.1.3.3 3D city model with mobile mapping system

Mobile Mapping System (MMS) is the setup of a camera, LiDAR, RADAR, and GPS mounted on a vehicle. Once calibrated, street recordings are captured from the POV of the vehicle. After capturing the data, it is visualized, and a 3D city model is obtained. Google Street View uses the same concept to capture the street and visualize them from ground level.

Blaer and Allen [36] developed a system for the automatic reconstruction of the 3D city model by using a robot mounted with a laser scanner to capture the data. By visualizing the captured data, they create a 3D model.

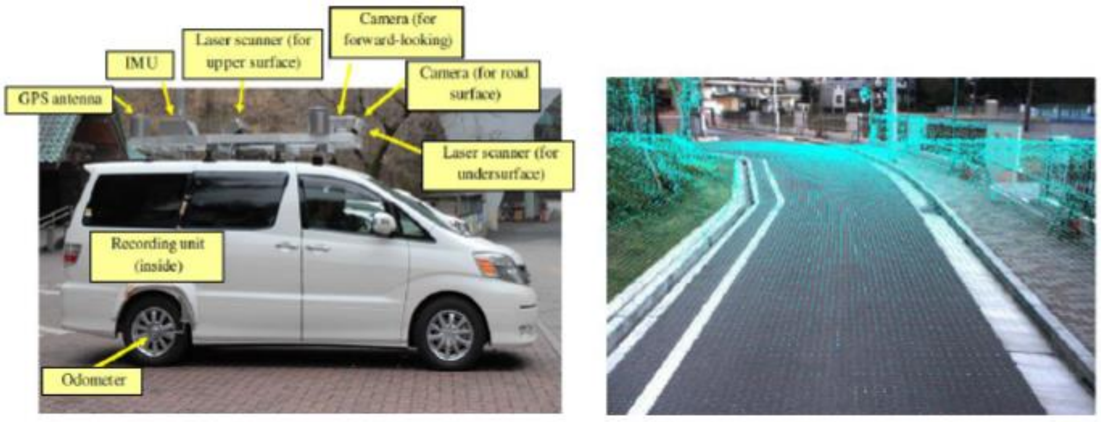

Camera (for forward-looking)
Laser scanner (for upper surface)
IMU
Camera (for road surface)
GPS antenna
Laser scanner (for undersurface)
Recording unit (inside)
Odometer

Figure 2.13. Setup for MMS (left); Simulation of captured data (right) [36]

In their work, they also developed a simulator to test the designed algorithm. With this approach, they created Uris hall at Columbia University and Fort Jay on Governors Island, New York.

### 2.1.4 VGI data-based 3D city models

In the last decade, research in the area of Volunteered Geographic Information has been at its best, resulting in an increase in the amount of 3D city development. One of the most used and popular projects established in 2004 is OpenStreetMap. Until now, there are more than 2 million active users and contributors with OpenStreetMap [45]. Research by Over et al. [46] introduced the possibility of constructing the 3D city model with the help of OpenStreetMap data, with extension to the research Groger et al. [47] conducted, which concluded that OpenStreetMap has the potential of creating LoD1 CityGML model.

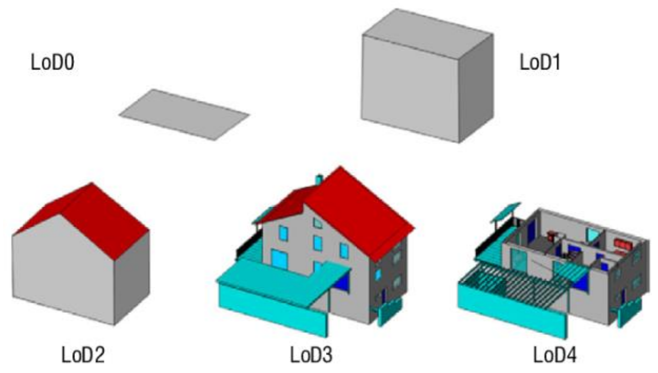


Figure 2.14. Level of Detail [48]

Figure 2.14 shows the information of Level of Detail. LoD is a 3D construction concept that checks the quantity of real-world details that have been acquired into the virtual environment. Moreover, as 3D models are constructed with different purposes that may not satisfy the universal criteria, the concept of LoD has been strongly criticized in the area of 3D reconstruction [48]. The main idea behind the construction of 3D environment with OSM data was to integrate OpenStreetMap and CityGML concepts. This concluded that it is possible to construct LoD1 and LoD2 3D models with OpenStreetMap but not with LoD3 and LoD4. Research work by Goetz M. et al. [44] provided a flowchart for highly detailed 3D city model with the use of OpenStreetMap data.

The earliest example of crowd generated 3D information is Google 3D Warehouse, started in 2006. The Warehouse contains geo-referenced real-world objects as well as non-referenced prototype models of different complex objects. In recent times, the development of Google SketchUp and ESRI Engine has increased the amount of production in 3D modeling. In 2007, Google and Microsoft included VGI data in their 3D modeling projects.

Goetz M. and Zipf A. [44] provided a method to construct a 3D city model by using OpenStreetMap data. OSM data is the tagged information of every place with geo-locations and height and area information. Authors developed a method to extrude the height parameter and convert the 2D mapped data into a 3D model.

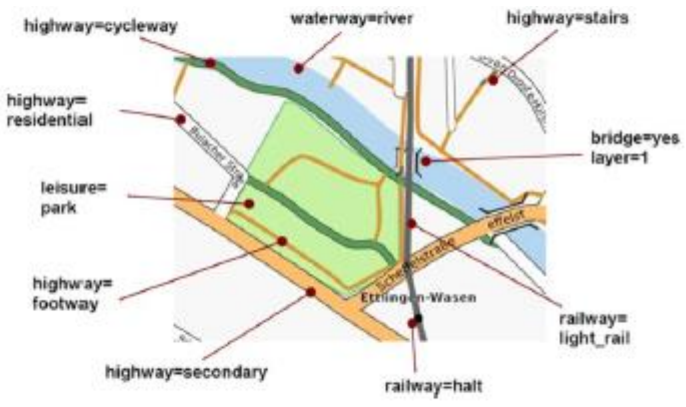


Figure 2.15. OSM tags [44]

Figure 2.15 shows the information about how the data is mapped in OpenStreetMap architecture.

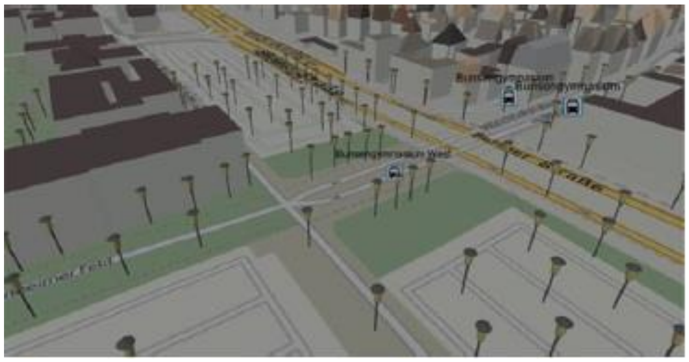


Figure 2.16. Extruded buildings from OSM [44]

Result of the work is shown in Figure 2.16 with extrusion of 2D features.

### 2.1.5 Hybrid data-based 3D modeling

In this method, multiple types of sensors are used to capture data for the construction of a 3D city model. Sensor combination such as Aerial Laser and Terrestrial Laser, Laser and Camera Images or VGI data and Laser, are used in 3D reconstruction.

#### 2.1.5.1 3D modelling with aerial laser and terrestrial laser

Bohm and Haala [37] designed a methodology for the construction of a 3D city model with a combination of Aerial Laser and Terrestrial Laser. The setup consisted of Leica HDS 3000 Terrestrial Laser and OPTECH ALTM 1225 Airborne Laser. Terrestrial Lasers collected façade information and geometry of the 3D model, and Aerial Laser collected information of roof shapes. A 3D point cloud is generated with a combination of both lasers. The combined data is visualized to construct a 3D environment.

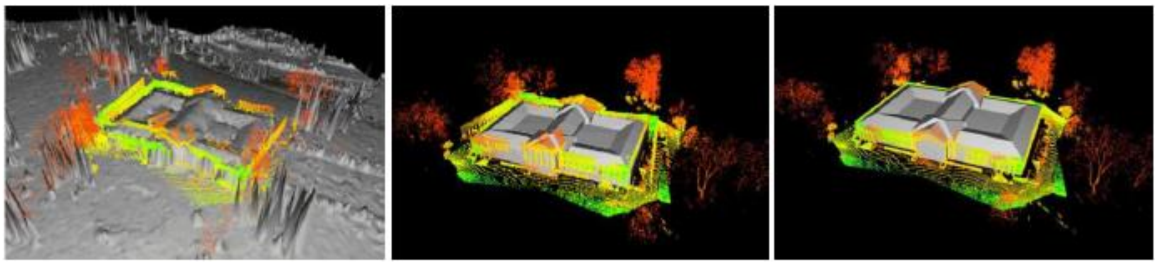


Figure 2.17. Combination of both lasers (left); Virtual model (center); Virtual model after alignment (right) [37]

In Figure 2.17, different phases of the methodology are shown; the image on the left is a combination of both lasers and image on the right is the final output of the approach.

#### 2.1.5.2 3D modelling with laser and photogrammetry

Habib et al. [38] designed a methodology to construct a 3D model with a combination of LiDAR data and aerial images. Point cloud generated with LiDAR is referenced with images to construct the final 3D environment. Conjugate features are used to geo-reference the images relative to LiDAR frames.

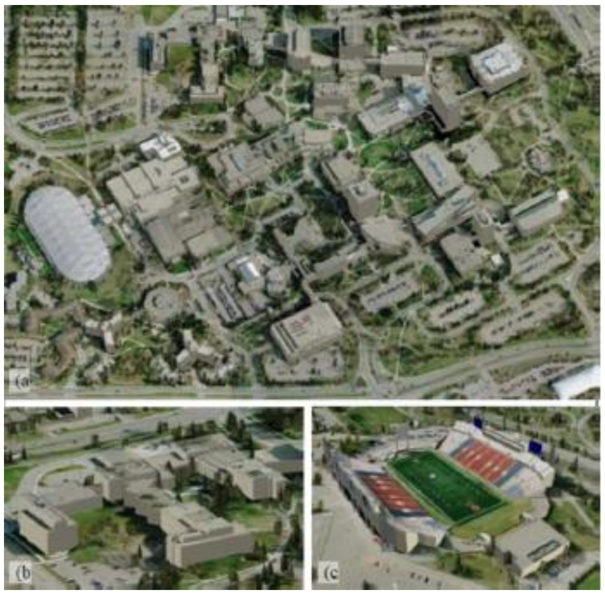


Figure 2.18. 3D environment with laser and aerial images [38]

The final output of the approach is shown in Figure 2.18, with textures mapped on 3D point cloud faces.

Frueh and Zakhor [39] developed a technique to construct a 3D textured model with the use of ground laser and camera. A truck was mounted with two lasers (one vertical and one horizontal) and one camera to capture recordings.

In this work, Markov Carlo Localization and correlation techniques are used for the final construction of the model. In 2003, the work was extended with the use of aerial laser and aerial imagery.

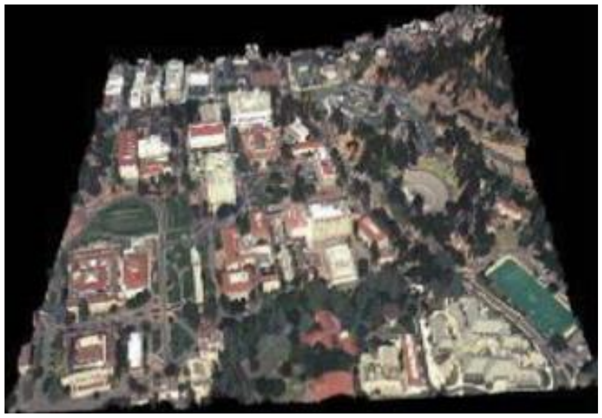


Figure 2.19. 3D model of Berkley University [39]

Data received from sensors is combined and visualized as a 3D model of Berkley University.

### 2.1.5.3 3D modelling with laser and videogrammetry

Zhao et al. [40] introduced a concept of alignment of continuous video on point cloud. LiDAR data and video data are captured for the same scene. A novel approach of registration is used to map the captured video on the 3D point cloud. Before the fusion of sensor data, the 3D point cloud is processed to find the planar surfaces.

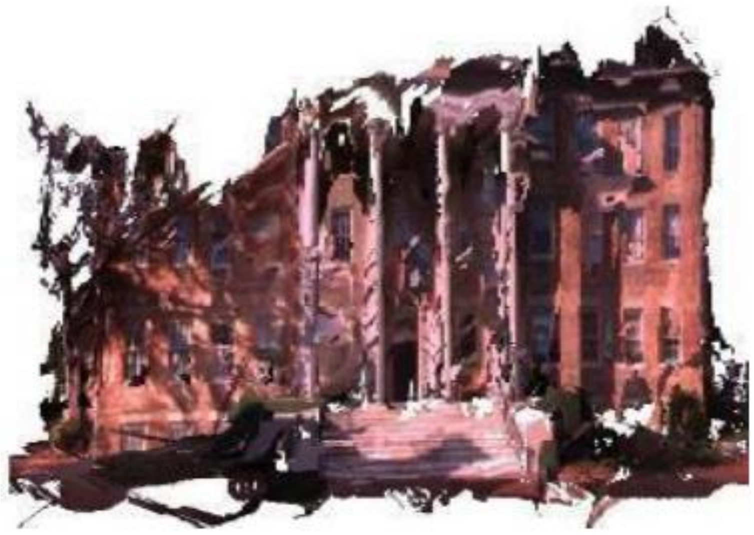


Figure 2.20. Laser data and videogrammetry [40]

## 2.2 Texture mapping 3D city model

3D city modeling mainly consists of three parts: geometric modeling, semantic modeling, and thematic modeling. One of the most important tasks is to texture map the geometric model. When texture-mapped, it increases the quality of the 3D model and gives a realistic touch to the virtual environment. Also, having realistic textures is necessary when the model is used for purposes like urban development, navigation system, virtual tour, or for this research work as prior information provided to the self-driving car.

Texture mapping the 3D model consists of following processes

- **Texture Extraction**

  Texture extraction is the process of capturing a real-world image from the building or constructing a computer-based graphical image to map onto the 3D building model. To capture images real-world scenes, a setup with the combination of camera and GPS is used. The camera captures the images from the façade and the GPS sensor geo-locates the image with global coordinates.
- **Texture Reconstruction**

  Texture reconstruction is the process of cleaning the objects which are in occlusion with the building façade. Objects which occlude with buildings are trees, vehicle, night lamps, benches, humans, and electricity wires.
- **Texture Mapping**

  Texture mapping is the process of applying an image onto the relative building façade.

Texture extraction, texture reconstruction, and texture mapping will be discussed in detail in upcoming chapters.

Methods for texture mapping 3D city model

1. Photorealistic method
2. Pictometry based method
3. Laser scanning based method
4. Dynamic pulse function-based method

### 2.2.1 Photorealistic method

Façade texture is required to give building models a realistic view. Texturing can be achieved with images, shaded polygons, or solid color. If the texture is an image, then the image must be rectified as a façade. In Figure 2.21, the rectification can be seen. The rectified image is mapped onto the relative façade side. Yang B. et al. [43] proposed a method to extract façade texture from the real buildings and processes the images to rectify them and to remove the occlusion.

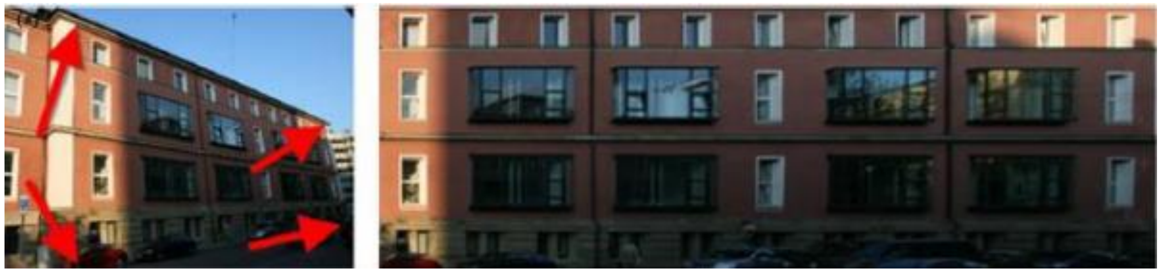


Figure 2.21. Rectified image [43]

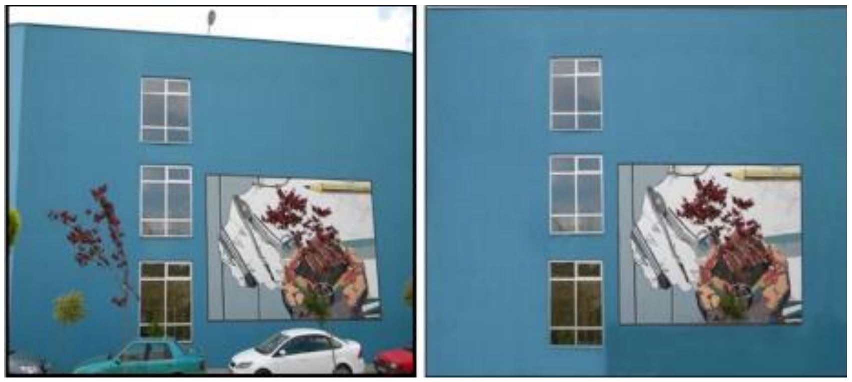


Figure 2.22. Real-life object removal [43]

As shown in Figure 2.22, the objects in occlusion are removed from the final image.

### 2.2.2 Pictometry based approach

In Pictometry, front and side elevated images are acquired along with the location of the buildings by using an Uncrewed Aerial Vehicle (UAV). Wang Y. et al. [41] proposed this system of using Pictometry to capture images for façade and roof. In this approach, five cameras capture five geo-referenced images for each side (one top and four sides).

Figure 2.23. Flight path for Pictometry [41]

In the low flying airplane, the camera angle is set 40-degree, which gives oblique angle images. Once the images are captured, the mesh is unfolded, and each image is mapped with the respective side. As shown in Figure 2.23, the low flying airplane captures five images of different sides.

One of the problems with this approach is that image capturing being at an oblique angle; some parts of the buildings are not captured in the cameras.

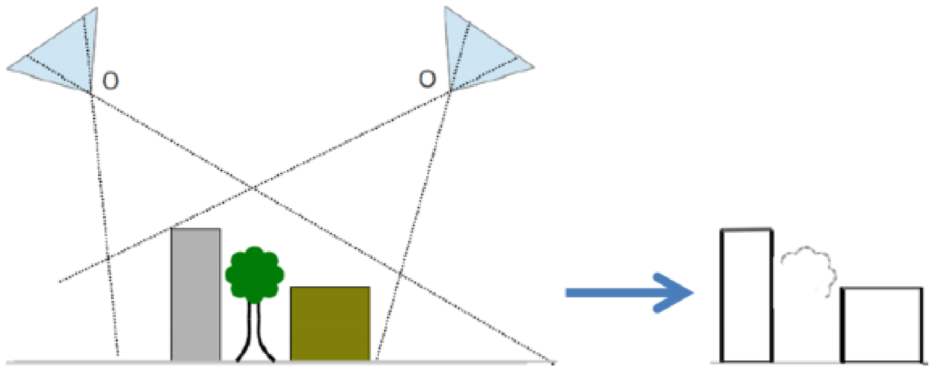


Figure 2.24. Hidden angles in Pictometry [41]

In Figure 2.24, it can be seen that while capturing from one side, the other building part is not visible.

### 2.2.3 Laser scanning based approach

In this method, a laser scanner is used at an oblique angle to generate 3D data of the buildings. To capture texture data, a camera is used at an oblique angle [39]. The data received from the camera is superimposed onto the data received from the laser scanner. To match the respective side with the image, 3D line segments from the laser scanner are matched with the 2D line segments of the aerial image.

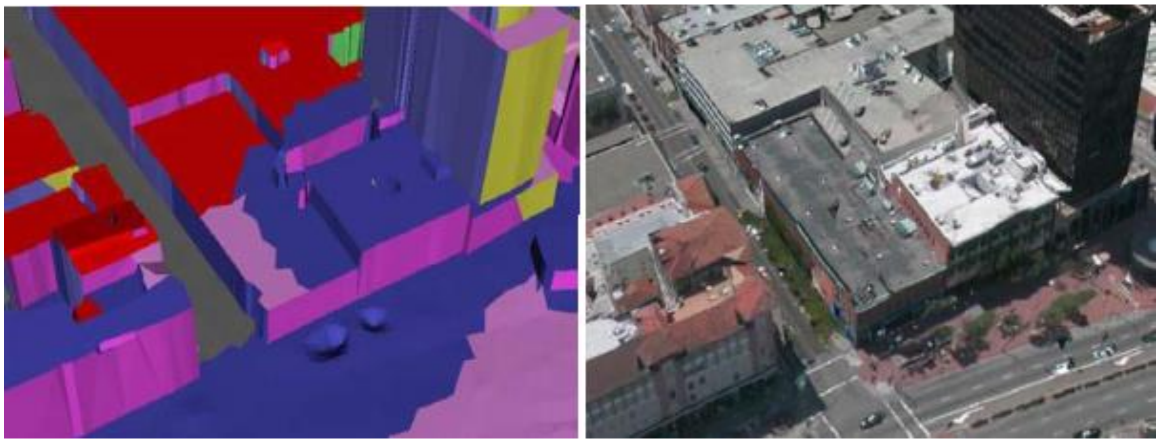


Figure 2.25. Laser scanned data (left); texture image (right) [39]

### 2.2.4 Dynamic pulse function-based method

Alizadehasharfi et al. [42] introduced a technique for texture construction named Dynamic pulse function. The system is a computer-based texture reconstruction. This method is applicable only to those facades which have repetitive objects such as air-conditioners, or windows. The output from this system is of high quality. As shown in Figure 2.26, the left side image contains all parts of the image. The image on the right is the final output constructed with the approach.

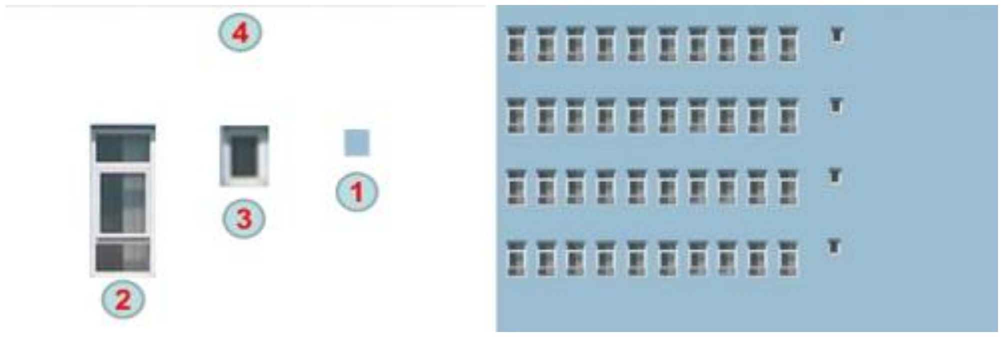


Figure 2.26. Dynamic pulse function [42]

For testing purposes, the 3D model of Karabuk University is modeled in SketchUp, and textures are created with Dynamic pulse function method.

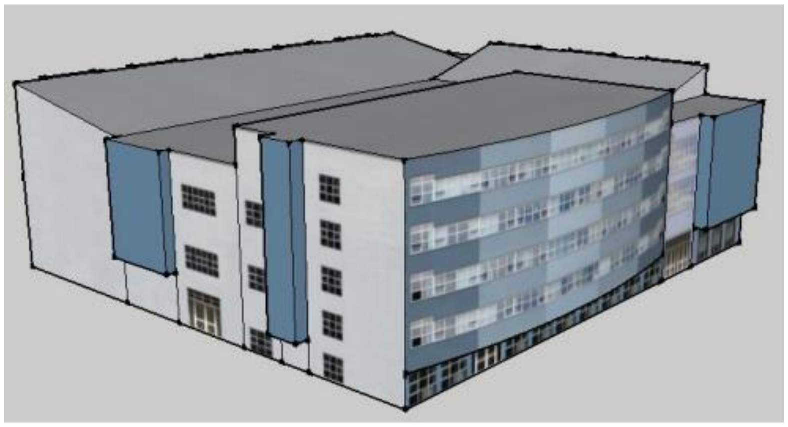


Figure 2.27. Final model of Karabuk University [42]

Summary of the methods discussed in Section 2.1 and 2.2 is provided in Table 2.1. Table consists information of method, input to the system, hardware sensors used, acquisition of building shapes, process involved in construction, texture extraction and output of the method.

| **Method** | **Input** | **Sensors** | **Building Shape** | **Process** | **Texture** | **Output** |
|---|---|---|---|---|---|---|
| Aerial Images + Close Range Images | Stereo paired Aerial Images | Airborne Sensor | Aerial images + CAD Package | Image processing based method | Close range images | 3D object model |
| Aerial Images + Cadastral maps | Aerial images | Airborne Sensor | Cadastral Information | Image processing based method | Not used | 3D object model + No texture |
| Computer vision | Image sequences | Stereo camera + GPS + Compass | Camera Images | Structure-from-motion | Camera images with GPS | 3D model with texture |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| GIS | Raster images + Vector data | N/A | N/A | Image processing, Point based method | N/A | 3D Point cloud |
| GIS + Digital Image | Raster images + Vector data + Digital image | Digital camera | Model constructed in Google SketchUp | Image processing based method | Close range digital images | 3D model with texture |
| Satellite Images | IKONOS satellite images | Not used | IKONOS images | Rational Functional model method | N/A | 3D object model + No texture |
| Single satellite image | IKONOS satellite image | Not used | Rational Polynomial coefficients and DEM model | Monoplotting technique and shadow of building | N/A | 3D object model + No texture |
| Panoram a Images | Digital panorama image | Camera + GPS | Digital images | Image calibration with space intersection | Digital panorama images | 3D object model with texture |
| Video | Video or Image sequences | Camera recorder + GPS | Recorded video | Computer vision with Video processing and SfM method | Video footage | 3D object model with texture |

| | | | | | | |
|---|---|---|---|---|---|---|
| Three-line-scanner | TLS data of forward, backward and nadir point | TLS sensor + GPS | Stereoscopic measurement with point cloud data | TLS data and CC Modeler software | Not used | 3D model with texture |
| UAV based model | Oblique angle aerial images and terrestrial images | UAV with GPS + camera | Different types of images | Image based methods and camera calibration with software | Close range terrestrial images | 3D model with texture |
| Laser based method | Point cloud and ground plan | LiDAR | From point cloud data | Segmentation of point cloud surfaces | Not used | 3D object model + no texture |
| Mobile mapping system | Point cloud and camera images | LiDAR + camera + GPS | From point cloud data and images | Mapping technique with point cloud and camera images | Camera mounted on MMS system | 3D model with texture |
| Aerial + ground laser | 3D point cloud of aerial and ground parts | Airborne LiDAR + Terrestrial LiDAR | By combining the 3D point cloud from both the lasers | Segmentation of point cloud surfaces | Not used | 3D object model + No texture |

| | | | | | | |
|---|---|---|---|---|---|---|
| Airborne Images + Ground LiDAR | Façade point cloud and aerial images | LiDAR + UAV camera + GPS | LiDAR 3D point cloud | Orthophoto generation technique to combine LiDAR and images | Aerial images | 3D model with texture |
| VGI data + Government open data | Vector information and building models | Not used | Visualizing vector data | Combination of building information models and extrusion | Not used | 3D object model + No texture |
| Our approach (VGI data) | Vector information from OSM | Not used | Vector data of OSM | Extrusion of 2D footprints and Model-to-image comparison with inpainting | Street-view imagery | 3D model with texture |

Table 2.1: Summary of 3D reconstruction methods with texture information

## 2.3 Related works

The table below gives the information about the work done so far by researchers in area closely related to this research work; also, mentioned are contributions and scope of improvements.

| **Research Paper** | **Contributions** | **Scope of Improvement** |
|---|---|---|
| Generating 3D city models without elevation data. Biljecki, F., Ledoux, H., & Stoter, J. 2017 | Uses OpenStreetMap data for the construction of 3D city model; also, open government data used for extra information of buildings | LiDAR is used for to extract the building height information, use of sensor increases cost; roof |

|  |  |  |
|---|---|---|
|  |  | information has not been used to from OSM |
| 3D city model construction based on a consumer-grade UAV. Zhongdi, Y. U., Hui, L. I., Fang, B. A., & Zhaoyang, W. A. N. G. 2018 | 3D city model is constructed with textures; same sensors are used to capture the geometry of buildings and texture facades | UAV is mounted with four different cameras so, use of UAV increases the cost of overall system; the textures are captured at oblique angles so need to be processed with rectification methods. |
| Improving accuracy of automated 3-D building models for smart cities. Yang, B., & Lee, J. 2019 | The construction is carried out with the use of hardware sensors; Airborne LiDAR and camera with GPS sensor is used Accuracy is better with the approach because of all the sensors; also, uses already existing building information | An airborne LiDAR is used in the approach which does not give high point density, so the geometry of the building is not much accurate |
| Automatic Texture Reconstruction of 3D City Model from Oblique Images. Kang, J., Deng, F., Li, X., & Wan, F. 2016 | Oblique images are used in the construction of 3D city model with the use of UAV; Texture information is also extracted with the same sensors | Only oblique images have been used in the construction method; use of façade images can help improve the accuracy which is also easy to capture |

| | | |
|---|---|---|
| Integration of aerial oblique imagery and terrestrial imagery for optimized 3D modeling in urban areas. Wu, B., Xie, L., Hu, H., Zhu, Q., & Yau, E. 2018 | A database has been used which consist aerial and terrestrial images of same objects; objects matching method has been used to combine the aerial and terrestrial images | The matching method uses point cloud data created from image but not LiDAR, that reduces the accuracy of the façade plane |

Table 2.2: Review of 3D construction and texture mapping techniques

## 2.4 Thesis statement

### 2.4.1 Problem statement

Literature survey suggests that there is a need for continued change in the methods with which the 3D city model is constructed. 3D city model is used in the self-driving car to know its surroundings; it cannot be used until it gives proper results. At the time of writing this thesis, most methods use the sensory information as input for the construction of 3D city model. The cost parameter is affected with the addition of sensors. Sensors used in the self-driving car are LiDAR, RADAR, camera, GPS enabled IMU, and infrared camera.

On the other hand, in recent years, the amount of research in the field of Volunteered Geographic Information has increased. Researchers have also proved that a 3D city model can be constructed with crowdsourced data [44] [46]. Also, a tremendous amount of effort is exerted in extracting the VGI data with increased accuracy. Also, when a 3D map is constructed with sensors (in real-time), it demands for more computation power. Object detection from real-time constructed 3D environment also consumes more time. Moreover, sensors mounted on the self-driving car need to be changed after two-three years. To solve these problems, the proposed method focuses on constructing a 3D environment beforehand which can be used to remove static objects resulting in reduced computation time. Furthermore, constructing a 3D environment from VGI (also known as crowdsourced or open-sourced) data reduces the cost of the overall system.

# CHAPTER 3
# PROPOSED METHODOLOGY

This chapter discusses about the proposed system to construct a virtual 3D city model and processes of texture mapping; including texture extraction, texture reconstruction, and texture mapping. This chapter contains the flowchart used for the construction of 3D model and detailed methods of texture mapping. Additionally, the chapter discusses about the working of the overall system and connection of virtual 3D city model with all other modules.

## 3.1 Motivation

In recent years, research in the area of self-driving cars has increased. Despite this, a perfectly functioning autonomous car is still not a reality. Also, semi-autonomous cars have been made available to the market in the last couple of years; and, they have been involved with some pedestrian fatalities [74]. Recently, two deaths involving Uber and Tesla self-driving systems have raised safety concerns. This has resulted in arriving at a conclusion that more computation time is required for dynamic object detection [71].

In our approach, a 3D environment of a real place is constructed, which helps in reducing the computation time for the self-driving system by eliminating static and variable objects (buildings, trees). Also, we are aware that the cost of 3D construction is a huge issue because of the usage of hardware sensors. Hence, this system proposes a method for the construction of a 3D environment with reduced cost and reduction in computation time for the self-driving system.

## 3.2 Working of the overall system

The overall system consists of six modules:

1. Construction on 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 3.1, all these modules are interconnected which each other

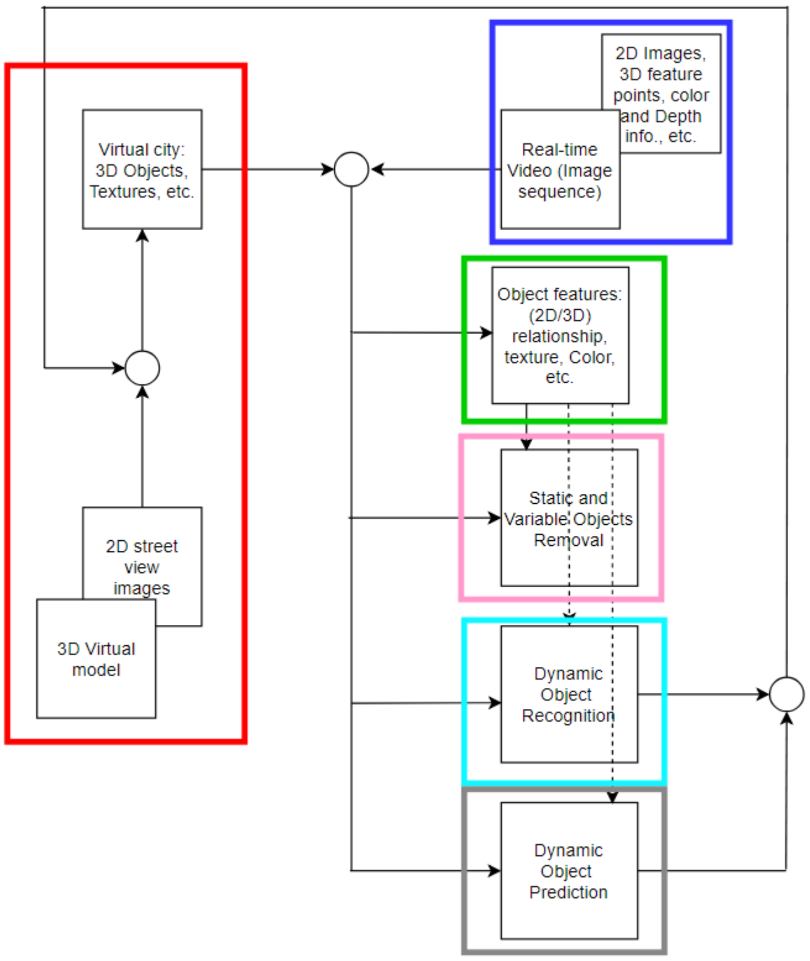

2D Images, 3D feature points, color and Depth info., etc.
Virtual city: 3D Objects, Textures, etc.
Real-time Video (Image sequence)
Object features: (2D/3D) relationship, texture, Color, etc.
Static and Variable Objects Removal
2D street view images
3D Virtual model
Dynamic Object Recognition
Dynamic Object Prediction

Figure 3.1. Overall system

In Figure 3.1, work shown in the red-colored box is the contribution of this thesis work. Its connection with all other modules, which are in different colored boxes, is also depicted using arrows.

The description of the overall system is in reference to Figure 3.1. The overall system primarily deals with the construction of a virtual 3D environment with the use of OpenStreetMap data (VGI/crowdsourced) and the façade texture from Google street view images. The virtual 3D city model consists of static objects, such as buildings, and some of the variable objects, such as trees. Apart from this, there is a separate repository which contains 3D models of dynamic objects, such as cars. The module marked in the blue-colored box in Figure 3.1 shows the real-time video (image sequences) passed as input to the system. Real-time image is an image received by the self-driving system through the camera mounted on car. The virtual environment is rendered, and keypoint features and 3D features are stored in a repository; this work is performed in the module colored in green. The module marked in pink is the static and variable object elimination module. In this module, the keypoint features of the input image (blue module) and keypoint features of the virtual environment (pink module) are matched. Matching the keypoint features of the virtual environment and real-time image confirms the location of the car in the real-world; this solves the problem of geo-localization of the self-driving car. With the matching, location of the static objects is also confirmed, and they are eliminated from the object identification process; which 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 which have impact on the navigation of the self-driving system. The module marked in cyan deals with the object recognition and pose estimation of dynamic objects present in the real-time input image, such as cars. Additionally, this module tracks the recognized objects from multiple frames of the video and calculates the speed of the dynamic object. The recognized object with the pose information along with the object speed and location is used to update dynamic objects into the 3D virtual environment. The module marked in grey color updates the dynamic objects' information into the virtual environment.

### 3.2.1 Working of individual modules

The modules which are directly associated with this research work are extraction of objects features, and dynamic object prediction. The virtual 3D city model and the real-time video are the input to the overall system.

1. **Keypoint extraction and dataset creation:**

   In the scope of this module, only cars and humans are considered as dynamic objects. In the case of cars, the 3D object models stored in the repository are rendered. KeypointNet [62] is used to extract the keypoint features from different rendered views of car models. The coordinate information of the identified keypoints in the rendered image, orientation details of the keypoint, and the direction of the car (left, right, towards, away), are stored in an annotation file. In case of humans, the DensePose [63] model is adopted and integrated into the overall system for human pose estimation. The DensePose model has its own manually collected ground truth dataset, which annotates dense correspondence between the image and a 3D surface model by asking the annotators to segment the image into semantic regions and to then localize the corresponding surface point for each of the sampled points on any of the rendered part images [64]. The surface coordinates of the rendered views localize the collected 2D points on the 3D model. The dynamic object recognition module uses this repository for matching the keypoint features of the dynamic objects in the input image with the keypoint feature information of the 3D object models stored in the repository and a suitable 3D model corresponding to the object in the input image is retrieved.
2. **Static and variable object removal:**

   In this module, static and variable objects in the real-time image are detected with the Fast R-CNN approach. Once detected, the keypoint features of a virtual image are matched with the keypoint features of a real-time image to verify static and variable objects. Once verified, a heat map is generated for the verified images, and with the help of the heat map and the contour detection, verified objects are eliminated from the image. In this way, static and variable objects are eliminated. This helps reduce the object detection time while the car is running.
3. **Dynamic object recognition:**

   This module matches keypoint features of the dynamic objects in the input image with the keypoint feature information of 3D object models stored in the repository to find a suitable matching 3D model for each of the dynamic objects present in the input image. A voting algorithm is used for the matching purpose, which also estimates a confidence score that signifies the confidence of object identification. This process improves the confidence of recognition and pose estimation of dynamic objects in the input image. This module is marked in cyan color in Figure 3.1.

4. **Dynamic object prediction:**

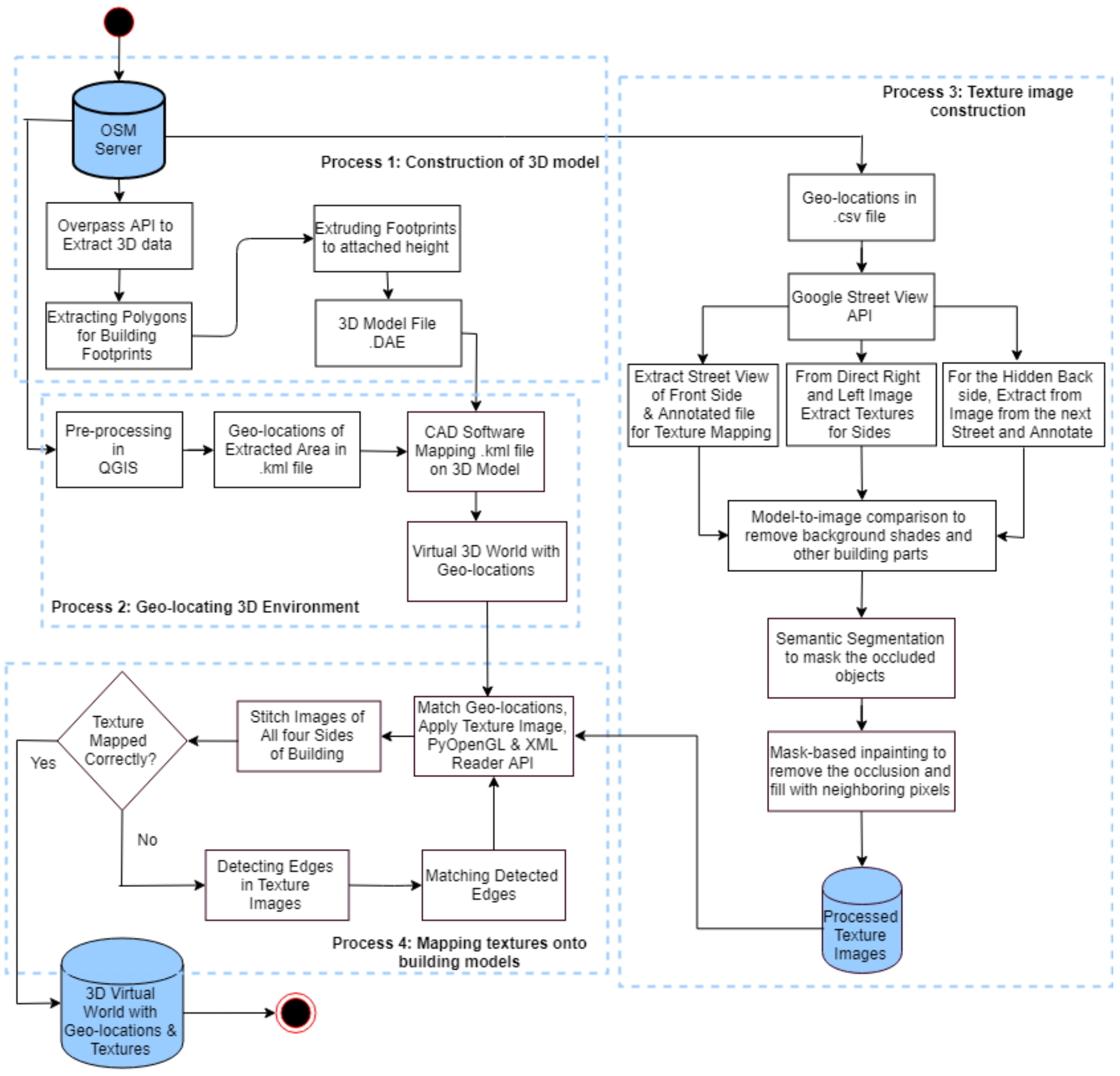
   This module uses the information from the dynamic object recognition module to update the virtual city with dynamic objects on the road in real-time. The recognized object, with its pose information, speed and location, is used to update the virtual city with the identified dynamic objects in real-time. Prior knowledge about the dynamic, static and variable objects from the virtual city and IoT is then used to determine the appropriate navigation decision of the self-driving car. This module is marked in grey color in Figure 3.1. The output of different modules is visualized in this module with the use of AirSim Simulator. Also, future prediction about the navigation of dynamic objects is calculated in this module. Moreover, this module predicts the distance from dynamic objects and visualizes them with different indicators.

## 3.3 Proposed methodology for construction of virtual environment

The proposed methodology consists of four processes followed by the sub-processes,

1) **Construction of 3D building models**
   - I. Extracting 3D structural data
   - II. Converting the 2D footprints into 3D models
2) **Geo-locating the 3D environment**
   - I. Setting the offset between local coordinates and world coordinates
3) **Texture image construction**
   - I. Extracting Textures from street-level imagery
   - II. Reconstructing the texture images
4) **Mapping texture onto building models**
   - I. Comparing the façade plane with the image

With the usage of a virtual 3D environment, the car becomes aware of its surroundings so that it can function accordingly. Currently, most methods use hardware sensors to know the surroundings. Whereas, this research work aims to use opensource VGI data for the construction of the 3D virtual environment. As seen in Chapter 2, research in the area of VGI has increased in recent years; latest research proves [44] that VGI can contain the 3D structural data of different types in the form of a database. This idea has been exploited in our research work to gather 3D structural information. OpenStreetMap data is used in this approach as opensource data. The virtual 3D environment constructed with this approach reduces the cost of the overall system by eliminating the usage of hardware sensors. Flowchart of the proposed system is shown in Figure 3.2.



OSM Server
Process 1: Construction of 3D model
Overpass API to Extract 3D data
Extracting Polygons for Building Footprints
Extruding Footprints to attached height
3D Model File .DAE
Pre-processing in QGIS
Geo-locations of Extracted Area in .kml file
CAD Software Mapping .kml file on 3D Model
Virtual 3D World with Geo-locations
Process 2: Geo-locating 3D Environment
Process 3: Texture image construction
Geo-locations in .csv file
Google Street View API
Extract Street View of Front Side & Annotated file for Texture Mapping
From Direct Right and Left Image Extract Textures for Sides
For the Hidden Back side, Extract from Image from the next Street and Annotate
Model-to-image comparison to remove background shades and other building parts
Semantic Segmentation to mask the occluded objects
Mask-based inpainting to remove the occlusion and fill with neighboring pixels
Processed Texture Images
Match Geo-locations, Apply Texture Image, PyOpenGL & XML Reader API
Stitch Images of All four Sides of Building
Texture Mapped Correctly?
Yes
No
Detecting Edges in Texture Images
Matching Detected Edges
Process 4: Mapping textures onto building models
3D Virtual World with Geo-locations & Textures

Figure 3.2. Flowchart proposed methodology

### 3.3.1 Construction of 3D building models

As discussed, OpenStreetMap data is used as 3D structural information. This works as a base for the building models. Overpass API is used to extract OpenStreetMap data from its server; extracted data is a read-only copy of the main OpenStreetMap database, which delivers an arbitrary amount of data. To extract data with Overpass API, coordinates of four real-world points are supplied; with the use of four locations, it creates a bounding box of the region, and that region is extracted in XML data format.

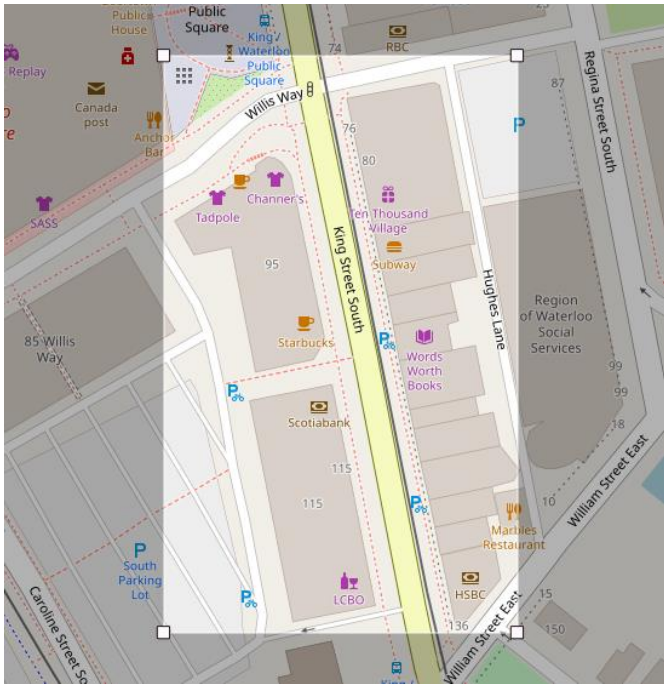


Figure 3.3. Region to extract

An example depicting the use of Overpass API is shown in Figure 3.3; a region to extract is created with the use of bounding boxes. Once the XML file of 3D structures is extracted, it is visualized as 2D footprints by using OSM library. 2D footprints is a digital drawing of building in 2-dimensions with the height information attached to it. In Figure 3.4, the flowchart for the construction of 3D building models is shown.

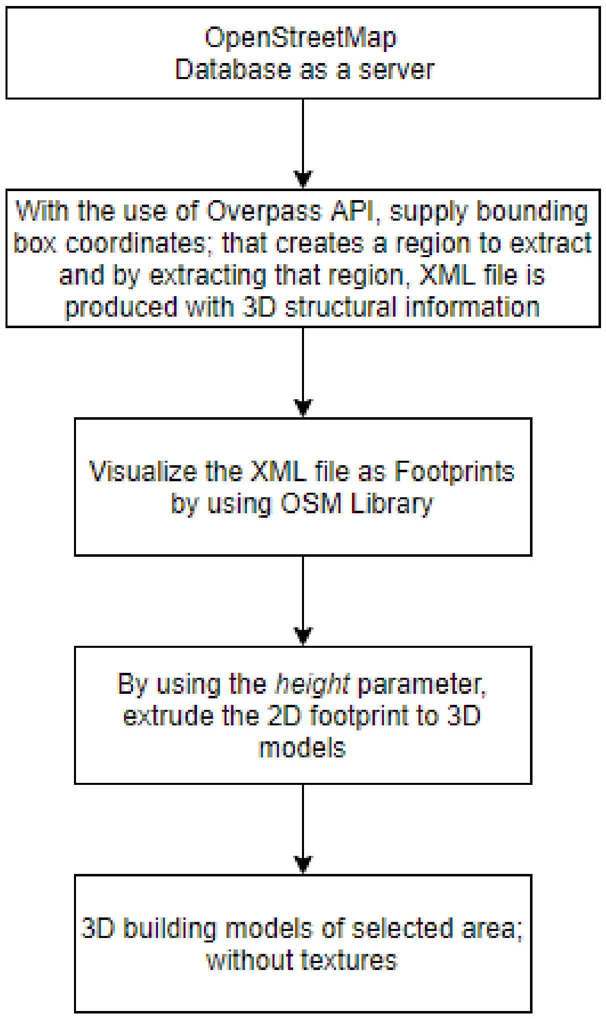
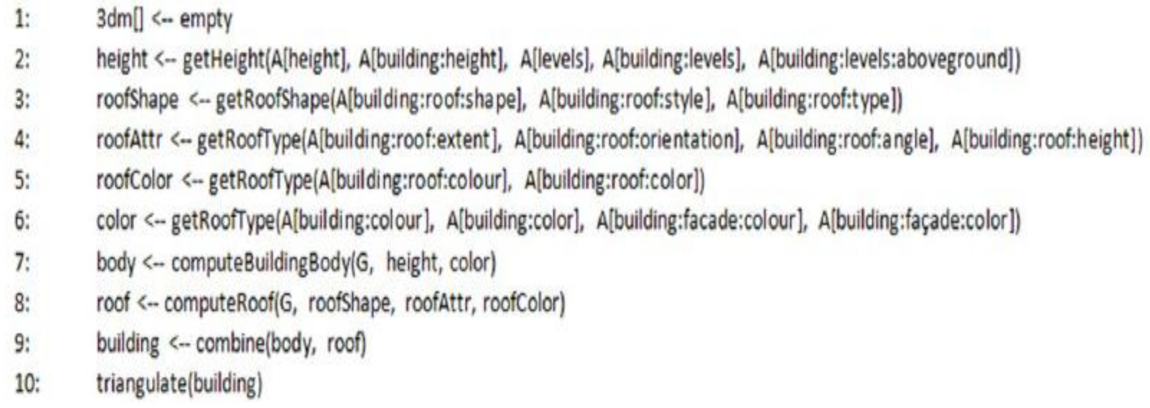

OpenStreetMap Database as a server
With the use of Overpass API, supply bounding box coordinates; that creates a region to extract and by extracting that region, XML file is produced with 3D structural information
Visualize the XML file as Footprints by using OSM Library
By using the *height* parameter, extrude the 2D footprint to 3D models
3D building models of selected area; without textures

Figure 3.4 Flowchart for 3D building models

Once the 2D footprints are visualized, the height information is used to extrude the 2D footprints into 3D building models. The most commonly used method for converting 2D footprints to 3D buildings is extrusion. Starting with the basic planar surface, each edge and vertex is selected. They are extended up to their attached height. Pseudocode for the extrusion process is described in Figure 3.5. The extrusion process for the OpenStreetMap data starts by looking for the tags that are relevant to the extrusion process. Some of these tags are ***building: height, building:levels, building:levels:aboveground, building:color, building:colour, building:façade:color, building:façade:colour, building:roof, building:roof:shape***, and ***building:roof:type***. After checking for the height and roof information, separate parts are created for body and roof, which are combined to construct one 3D model. This thesis does not include the roof portion in the process. This is due to the limited availability of databases containing roof information. The pseudocode for the extraction of OSM:tags and combining the building parts is described in Figure 3.5.



```
1:    3dm[] <-- empty
2:    height <-- getHeight(A[height], A[building:height], A[levels], A[building:levels], A[building:levels:aboveground])
3:    roofShape <-- getRoofShape(A[building:roof:shape], A[building:roof:style], A[building:roof:type])
4:    roofAttr <-- getRoofType(A[building:roof:extent], A[building:roof:orientation], A[building:roof:angle], A[building:roof:height])
5:    roofColor <-- getRoofType(A[building:roof:colour], A[building:roof:color])
6:    color <-- getRoofType(A[building:colour], A[building:color], A[building:facade:colour], A[building:façade:color])
7:    body <-- computeBuildingBody(G, height, color)
8:    roof <-- computeRoof(G, roofShape, roofAttr, roofColor)
9:    building <-- combine(body, roof)
10:   triangulate(building)
```

Figure 3.5. Pseudo code for building extrusion [44]

In this way, 3D building models are constructed from the OSM data with the GIS extrusion process. The constructed 3D building model is without textures and geo-locations.

### 3.3.2 Geo-locating 3D environment

Geo-locating the virtual 3D environment is necessary for the virtual environment to work as a GIS system. The XML file extracted from OpenStreetMap server contains geo-locations of all buildings, but when the data is visualized and extruded to be converted into a 3D environment, it loses the geo-locations of all buildings. Hence, the virtual environment is geo-located. The process of geo-locating the virtual environment starts with pre-processing of the XML database file. First, when the 2D footprints are visualized in QGIS, only the geo-location layer is extracted and saved in a separate file with .kml (Keyhole Markup Language) extension. QGIS is a geographic information system software, mainly designed for the analysis of spatial data. As the construction of 3D models and extraction of the kml file (geographic location) has been carried out from the same database, the kml file contains the geographic location information of each building. Once the kml file is extracted, the virtual 3d model and the geographic location (kml) file is imported in CAD software and matched with each other. Thus, the virtual 3D model is geo-located. This process of georeferencing is described in Figure 3.6.

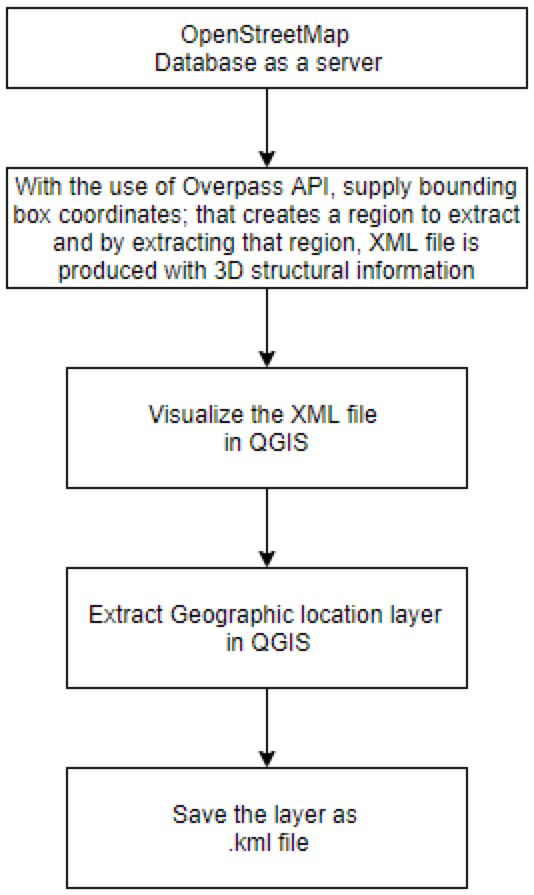

OpenStreetMap Database as a server
With the use of Overpass API, supply bounding box coordinates; that creates a region to extract and by extracting that region, XML file is produced with 3D structural information
Visualize the XML file in QGIS
Extract Geographic location layer in QGIS
Save the layer as .kml file

Figure 3.6. Georeferencing virtual 3D model

### 3.3.3 Texture image construction

To make the virtual 3D environment look realistic, texture mapping is important. In Chapter 2, most of the current methods use photorealistic texture mapping, which is to capture texture from real locations. Usually, photorealistic texture mapping uses the combination of a camera and a GPS. The camera captures the façade texture information and the GPS references the image to geocoordinates. In our research work, a photorealistic approach is used for texture mapping; the texture is extracted from the Google street view imagery. The texture image construction is a two-part process: first, the extraction of texture images and second, the reconstruction of texture images.

1) Texture image extraction

   Texture images are extracted with the Google street-view static API. By supplying the geo-location as an input, the texture image file is fetched from google street view database. The geo-locations file used for geolocating the virtual 3D model, is reused as an input. Here, instead of a keyhole Markup Language file, a comma separated value (csv) file is used. While querying with the street view static API, these parameters are used to fetch the texture image [72].

- *Location*: takes either a string value (such as Niagara Falls, ON) or latitude and longitude value. When the address string or coordinates are provided, the API sometimes uses different camera location to provide a better picture
- *Pano*: is a specific panorama ID
- *Size*: specifies the output image size in pixels; specified as {width x height} – for example, 400 x 600 provides an image of 400 pixels wide, and 600 pixels high

Some of the optional parameters are

- *Heading*: indicates the compass heading of the camera; input values are from 0 to 360 (90 indicates East, and 180 indicates South). If the *heading* is not provided, then a value is calculated which directs at the location
- *Fov*: stands for Field of View of an image. This represents the zoom level for an image, with the default being 90 and maximum 120.
- *Pitch*: specifies the up or down angle of the camera relative to the street view vehicle. Positive value angles the camera upwards, and negative value angles the camera downwards.
- *Radius*: sets a radius to search an image, centered on given latitude and longitude, input values in meters

With the combination of default and optional parameters, a query is generated and supplied with street view static API to extract an image of a specific location. With the extraction of every image, one annotated file gets generated with the pano id and the geolocation of that image, which helps to match every image with relative façade plane.

2) Reconstruction of texture image

In the reconstruction of texture images, the occluded objects are removed to make textures look realistic. In occlusion removal, different objects such as cars, trees, humans, and night lamps are removed from texture image [43]. To remove occlusion and not lose the clarity of the image, an inpainting based method is used. Inpainting is a technique used to restore missing parts of an image or fill patches in an image.

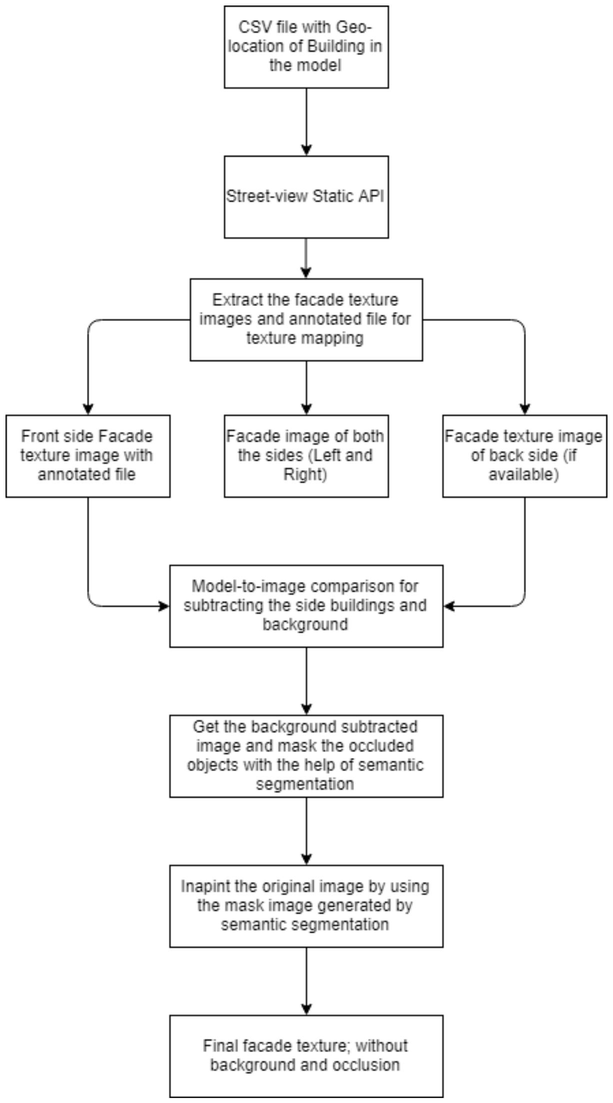

CSV file with Geo-location of Building in the model
Street-view Static API
Extract the facade texture images and annotated file for texture mapping
Front side Facade texture image with annotated file
Facade image of both the sides (Left and Right)
Facade texture image of back side (if available)
Model-to-image comparison for subtracting the side buildings and background
Get the background subtracted image and mask the occluded objects with the help of semantic segmentation
Inapint the original image by using the mask image generated by semantic segmentation
Final facade texture; without background and occlusion

Figure 3.7 Process of image extraction and reconstruction

In this approach, a mask-based inpainting approach is used to remove occlusion in the image. Figure 3.7 describes the process of image extraction and reconstruction.

As shown in Figure 3.7, all the four sides of textures are extracted for a building and annotated with the pano id and geo-locations. After extracting images for all sides, the model-to-image comparison is performed to remove the background of the image. Therein, the existing virtual 3d model (without textures) is rendered, and images are obtained for all different orientations. Then, the image from the virtual 3D model is projected onto the extracted texture image to match the edges of model-image and texture image. Once the edge-match is performed the remaining part of the texture image is masked and removed to construct the façade texture image of model size. This texture image still contains the occluded objects. To remove them, a mask based inpainting approach is used; which works based on two images, the original image and an image with the mask on objects to be removed. The masked images are obtained with the help of semantic segmentation. A pre-trained model of FCN with ResNet 101 is used to detect and mask the real-life objects from the image. The pre-trained model is trained to detect 16 different classes, such as vegetation, humans, animals, cars, etc. With the help of the masked image and the original image, the occluded objects are removed with inpainting method. Therein, the removed region is filled with the help of neighboring pixels.

Algorithm for texture extraction and texture reconstruction, with the use of model-to-image comparison and mask-based inpainting method is discussed.

**Algorithm: Texture image extraction and reconstruction**

**INPUT**: CSV file of Geo-locations

**OUTPUT**: Texture Images with model-image comparison and occlusion removal

```
Step 1: Extract the façade texture images with the use of street-view static API
Step 2: store the annotation file with pano id and geo-locations for the images
Step 3: If texture for any of the four sides is not extracted, increase the radius for the image
        in step 1 and repeat step 2.
Step 4: get the rendered image for the model-to-image comparison
Step 5: compare the model image with texture image; if it doesn't provide significant
        results, then detect edges and match for both images and repeat step 5
Step 6: perform FCN based ResNet 101 semantic segmentation to mask the image for
        inpainting purpose
Step 7: mask based inpainting to remove the object and neighboring pixels to fill the
        removed region
Step 8: texture images for building facades, with background subtracted and occlusion
        removed
Step 9: End
```

### 3.3.4 Texture image mapping

For texture mapping, GPS-assisted texture mapping approach is used. As discussed in Chapter 2, most methods use a hardware setup of a GPS and a camera to extract façade textures. Images captured with this setup are georeferenced by GPS sensor. In this approach, the façade texture is extracted from street-view imagery, which is not geo-referenced. To solve this problem, we use an annotated file to keep track of geolocations of each façade image in our approach. To select the façade texture image from the database, the geo-location of façade plane in the model, and the geo-location of façade texture image from the annotated file is matched. After matching the facade texture with the help of geolocations, the texture image is mapped onto the façade plane. If this does not provide a significant result, the edges of the 3D model plane and the edges of texture image are matched. Thereafter, the façade texture is mapped.

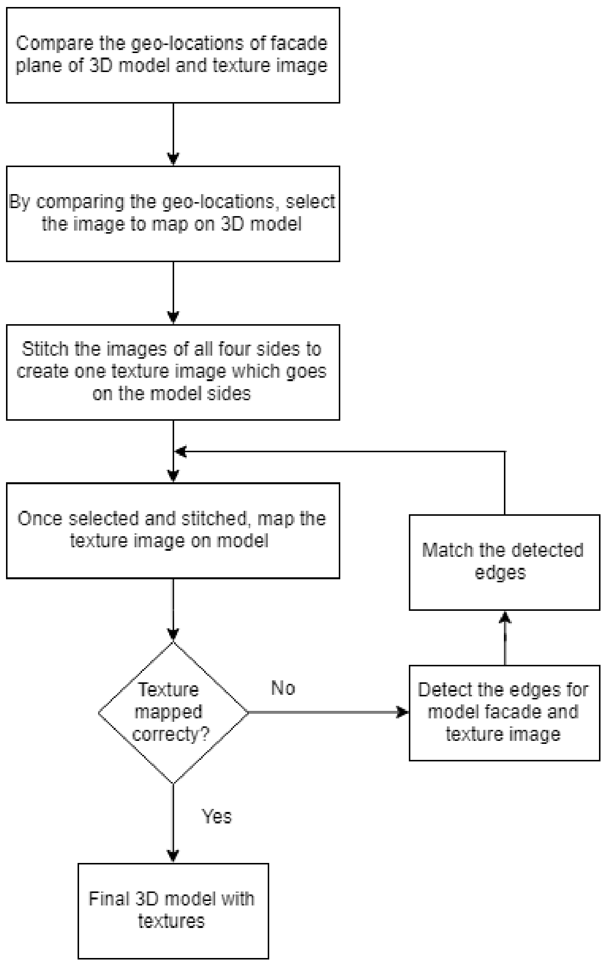

Compare the geo-locations of facade plane of 3D model and texture image
By comparing the geo-locations, select the image to map on 3D model
Stitch the images of all four sides to create one texture image which goes on the model sides
Once selected and stitched, map the texture image on model
Match the detected edges
Texture mapped correcty?
No
Detect the edges for model facade and texture image
Yes
Final 3D model with textures

Figure 3.8. Texture mapping process

The flowchart of texture image selection for specific façade and texture mapping with edge matching algorithm is shown in Figure 3.8.

With the algorithms used in the proposed system, the final output is a virtual 3D city model with texture mapped on the façade planes. The detailed description is of algorithms is provided in Section 3.3.1, 3.3.2, 3.3.3, and 3.3.4

The output of this proposed system is used by the overall system to geo-localize the self-driving car in the real-world. 3D features and keypoints are extracted from static objects in the virtual world. The extracted features are compared with the real-time image to eliminate the static objects from the object detection part which reduces the time for object detection. This provides more time for the detection of dynamic objects. The results of these processes are discussed in Chapter 4 in detail.

# CHAPTER 4

# IMPLEMENTATION AND EXPERIMENTS

The proposed approach is implemented on Windows OS using Python Programming Language, in the implementation, different Python, OpenCV and OpenGL libraries are used. The list of software and tools used is given in Table 4.1.

## 4.1 Software information

The implementation of proposed methodology was performed on Alienware 1.5.0 x64-based Desktop, with NVIDIA 8.1.940.0 and Intel 64 ~ 3192 MHz GPU.

| **ITEM** | **DETAILS** |
|---|---|
| Operating System | Windows |
| Languages | Python 3.7.1 |
| IDE | Spyder, Anaconda Navigator, JOSM |
| Python Libraries | OpenCV, Scikit, PyOpenGL |
| Tools | Maya 2019, SketchUp, 3D Viewer, QGIS, CAD Software |

Table 4.1: List of tools used for implementation and experiments

## 4.2 Data extraction from OSM server

To perform the experiment, a downtown area from Waterloo Region is selected; all the extraction and texture mapping is performed for the specific area. Location of the selected region is **King St S at Wills Way to King St S at William St E, Waterloo, ON**. In the proposed system, data extraction is performed with Overpass API by querying into OpenStreetMap server.

Query:

```
<osm-script>
  <union into="_">
    <query into="_" type="Node">
      <bbox-query s="43.46456" w="-80.51661" n="43.461" e="-80.52503"/>
    </query>
  </union>
  <print e="" from="_" geometry="skeleton" ids="yes" limit=""
mode="meta" n="" order="id" s="" w=""/>
</osm-script>
```

By running this query on OSM server, the OSM data for the supplied coordinates is returned in XML format; which contains the 3D structural information of the selected data.

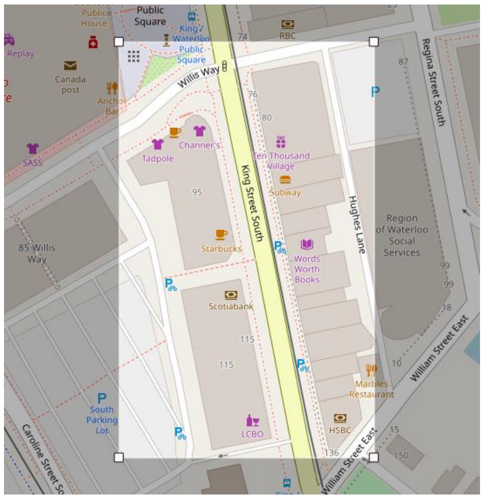


Fig 4.1. Bounding box query

The query mentioned above creates a bounding box for the supplied coordinates as shown in Figure 4.1. The XML file contains all the information of the buildings in the bounding box, such as geo-location, building height, address, type of building, and available facilities in the building. Moreover, in some places, the roof information is also available.

## 4.3 Visualizing data in vector format

In Figure 4.2, the XML format of OpenStreetMap data is visualized in vector format in Java OpenStreetMap editor to check the available information in the extracted area. JOSM is the map editor for OpenStreetMap data. Almost all the tagging of the data is also performed with this editor. Hereon, to visualize the footprints of the building, the OSM library is used in Python. The library is able to understand the tags of OSM architecture in python.

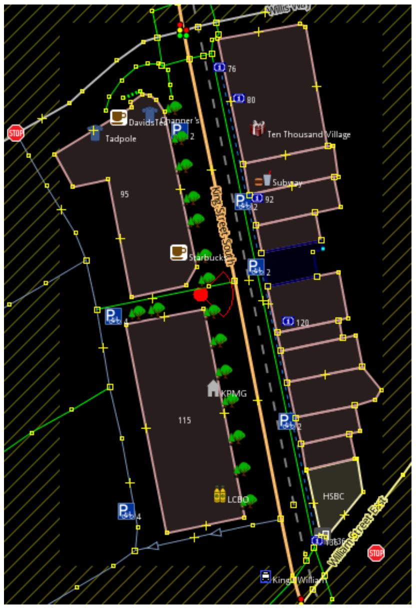


Figure 4.2 Vector format of XML data

As shown in Figure 4.3, the buildings and roads are represented as footprints. To visualize buildings, flat polygons are used; for roads and crossroads, strings are used; to visualize trees and crossing signals, a point has been used.

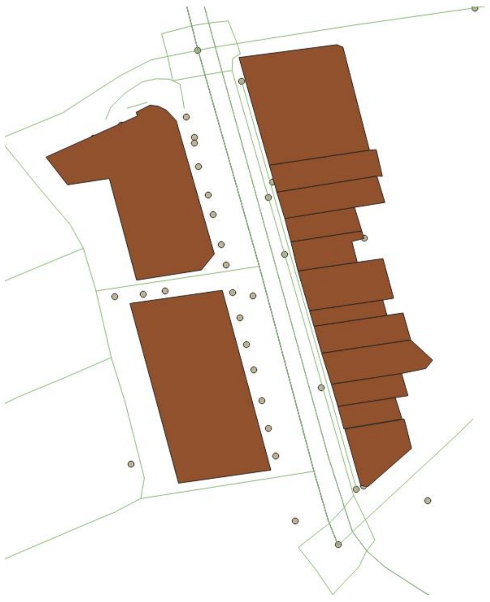


Figure 4.3. Footprints of OSM data

## 4.4 Extruding footprints to 3D models

Once the footprints are acquired, the buildings are extruded by using the attached height information with pseudo code shown in Figure 3.5. In this process, traffic signals and other night poles or benches are not extruded; only the buildings, roads, and some of the trees are extruded. The result of building extrusion process is shown in Figure 4.4.

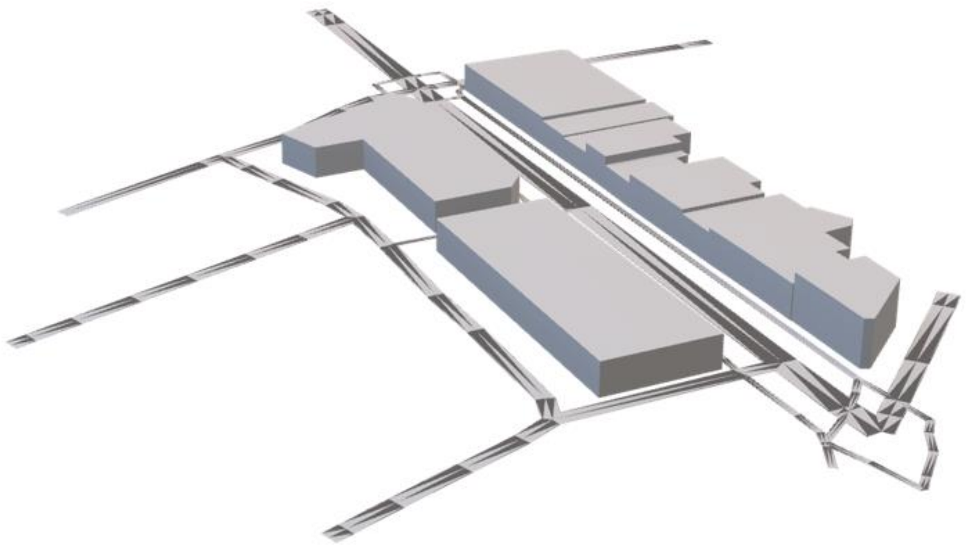
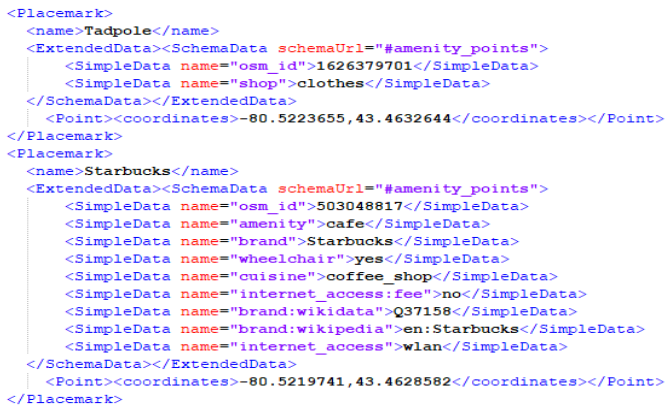


Figure 4.4. Extruded building models

3D model shown in Figure 4.4 is geo-located with the help of Keyhole Markup Language file which contains the geo-locations of all the objects in 3D model.



```xml
<Placemark>
  <name>Tadpole</name>
  <ExtendedData><SchemaData schemaUrl="#amenity_points">
      <SimpleData name="osm_id">1626379701</SimpleData>
      <SimpleData name="shop">clothes</SimpleData>
  </SchemaData></ExtendedData>
    <Point><coordinates>-80.5223655,43.4632644</coordinates></Point>
</Placemark>
<Placemark>
  <name>Starbucks</name>
  <ExtendedData><SchemaData schemaUrl="#amenity_points">
      <SimpleData name="osm_id">503048817</SimpleData>
      <SimpleData name="amenity">cafe</SimpleData>
      <SimpleData name="brand">Starbucks</SimpleData>
      <SimpleData name="wheelchair">yes</SimpleData>
      <SimpleData name="cuisine">coffee_shop</SimpleData>
      <SimpleData name="internet_access:fee">no</SimpleData>
      <SimpleData name="brand:wikidata">Q37158</SimpleData>
      <SimpleData name="brand:wikipedia">en:Starbucks</SimpleData>
      <SimpleData name="internet_access">wlan</SimpleData>
  </SchemaData></ExtendedData>
    <Point><coordinates>-80.5219741,43.4628582</coordinates></Point>
</Placemark>
```

Figure 4.5. KML file for geo-locations

To extract the geo-locations file, the XML file of OpenStreetMap data is visualized in QGIS software, from which the geo-location layer is exported in kml file format. Also, with the help of CAD software, the kml file is mapped onto the 3D model. Figure 4.5 shows the details of two building models from the selected area.

## 4.5 Texture image extraction

After the construction of 3D virtual model, photorealistic textures are extracted from street-view imagery with street view static API.

```
function initialize() {
  var fenway = {lat: 43.4634198, lng: - 80.5219242};
  var map = new google.maps.Map(document.getElementById('map'), {
    center: fenway,
    zoom: 14
  });
  var panorama = new google.maps.StreetViewPanorama(
      document.getElementById('pano'), {
        position: fenway,
        pov: {
          heading: 34,
          pitch: 10
        }
      });
  map.setStreetView(panorama);
}
```

A script with the details of Geo-locations and optional parameters such as zoom, heading, pitch, and field of view, extracts the images for texture façade.

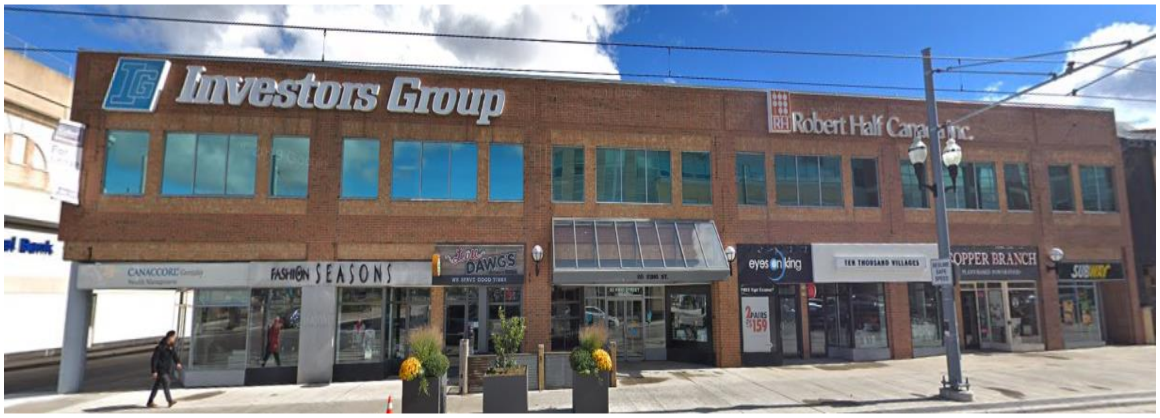


Fig 4.6. Façade texture image [72]

With the geo-locations {lat: 43.4634198, lng: -80.5219242}, the street view static API returns the image shown in Figure 4.6. The image is extracted with the use of optional parameters: zoom, heading, and pitch. With the extraction of the texture image, one annotated file is created which holds the geo-location records of the image and pano id, which helps in mapping the textures onto the model. In the proposed method, texture is a specific visual image which is extracted from street-view imagery and mapped onto each side of the building.

## 4.6 Reconstruction of texture image

### 4.6.1 Perspective transform image

When the texture images are extracted from the street-view, they are captured as a 360-degree image because when Google MMS captures the image, it uses 360-degree camera. To straighten the image, getPerspectiveTransform is used from OpenCV. Therein, the corner points of the image are detected, and the image is straightened with respect to those corner points. To detect the corner points, contours are detected in the image and the intersection of contours is stored as a corner point. Google street-view images are not straight as they are captured with a 360-degree camera that creates a panorama or a fisheye image.

In corner detection, if there is no corner detected in the process, then the biggest contour is selected as a building. In that, with no corner points in the image, there is no intersection of horizontal and vertical axis. With no intersection, there is only one contour detected in the contour detection process. The biggest contour area is selected as a building in an image. The originally extracted image is shown in Figure 4.6. The perspective transform image is shown in Figure 4.7.

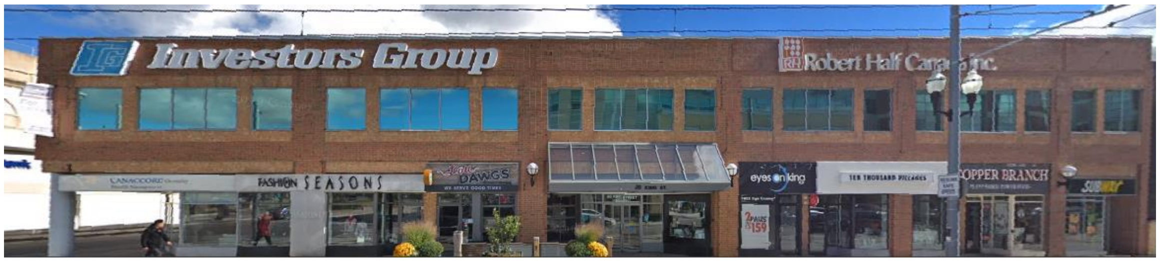


Figure 4.7. Perspective transform

### 4.6.2 Model-to-image comparison

After extracting the façade texture image from street view imagery, the rendered image of the model and the façade texture image are compared by projecting them on each other to detect the edges, With the projection, the background shades and other connected buildings are subtracted. To reconstruct, the image shown in Figure 4.7 is used as the façade texture. Also, the same view is fetched from 3D model with the help of geo-locations, which shown in Figure 4.8.

Figure 4.8. Rendered image of 3D model

In the model-to-image comparison, the façade texture image extracted from street-view imagery is projected onto the 3D model image. Once projected, the edges of the rendered image the façade texture image are compared to match the images. The projection of both images is shown in Figure 4.9.

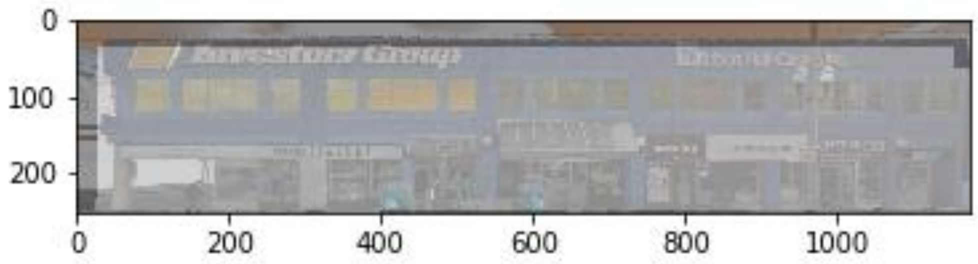


Figure 4.9. Model-to-image comparison

In Figure 4.9, there is a difference in the 3D model image and the texture image, which is removed by matching the edges of both the images. The final reconstructed image is shown in Figure 4.10.

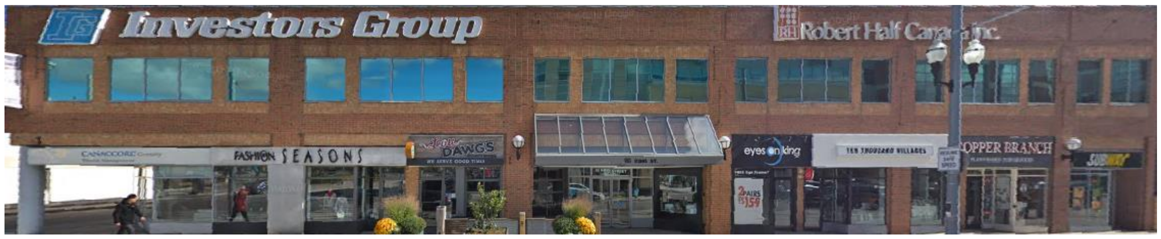


Figure 4.10. Background subtracted image

### 4.6.3 Occlusion removal

In the process of occlusion removal, a mask-based inpainting method is used. For the masking purpose, the texture image is semantically segmented, wherein different objects are detected in an image and masked with red. For semantic segmentation, a pre-trained model of FCN (Fully Convolutional Network) with ResNet 101 architecture is used.

Semantic segmentation is the process of defining specific pixel in an image to a class label. Generally, these labels include a person, furniture, flower, car, etc., just to mention a few. Semantic segmentation is image classification at pixel level. For example, there is an image that has many trees, segmentation labels that as a tree or describes it with a different color mask. For semantic segmentation, Figure 4.10 works as an input. Objects such as vegetation, light poles, electric poles and dynamic objects such as humans, cars, and animals are masked to be removed from the original image. Figure 4.11 shows the detected and masked objects.

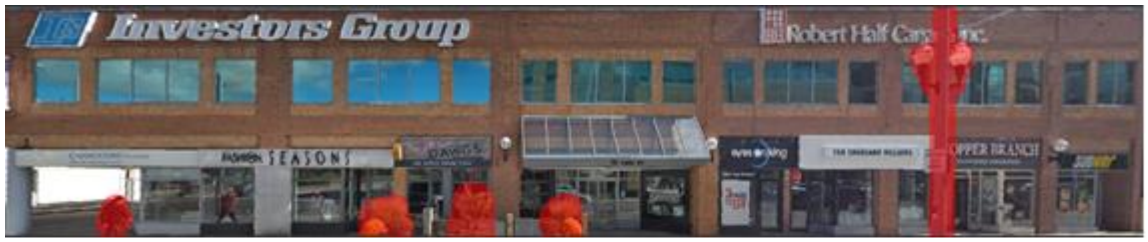


Figure 4.11. Masked objects for removal

Once the semantically segmented image is masked with the objects to be removed, mask-based inpainting is performed. Therein, the masked region is removed and tried to be filled with the neighboring pixels. The mask-based inpainting is useful when the removed region is small because of the use of neighboring pixels to fill the region. Figure 4.12 shows the occlusion removed façade texture image.

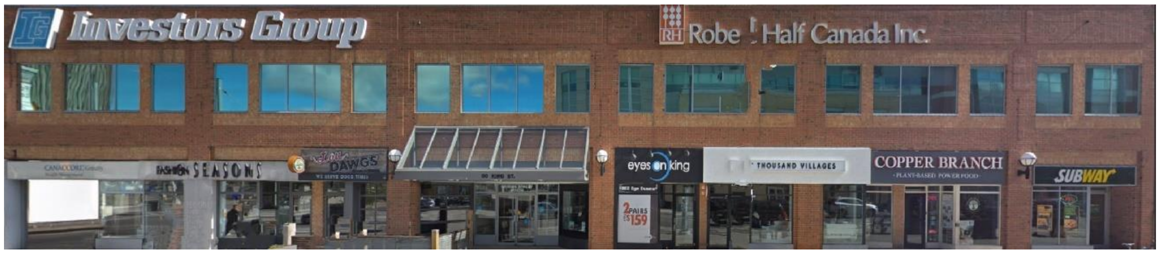


Figure 4.12. Occlusion removed façade texture

## 4.7 Image stitching

Herein, the OpenCV library is used with *createstitcher* class. This takes different images as input and combines them as one.

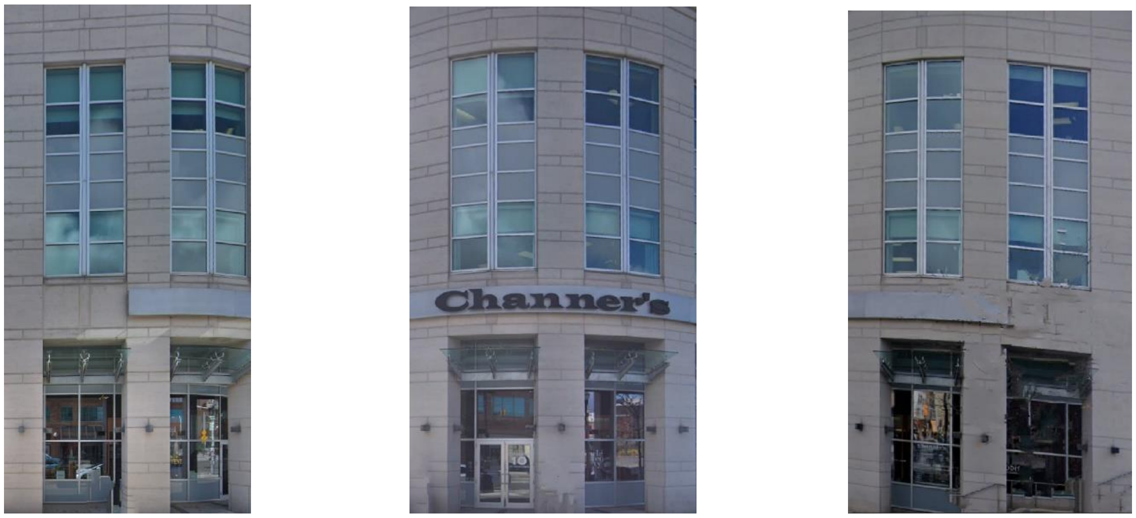


Figure 4.13. Three façade images to stitch

In Figure 4.13, three different images are shown, which are of different facades. All three images are stitched as one to wrap onto the 3D model, which is shown in Figure 4.14.

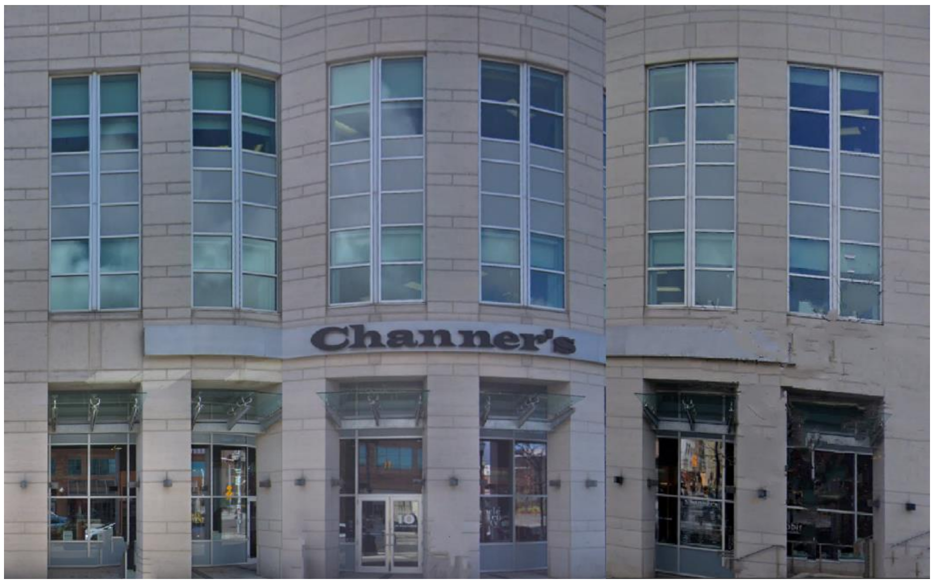


Figure 4.14. Stitched images

## 4.8 Texture mapping on the 3D model

In this process, the reconstructed texture image is mapped onto the relative 3D model by comparing the geo-locations. For mapping, relative to the geo-location of the façade plane, a texture image is acquired with the help of the annotated file. The annotated file has the details such as pano id and geo-location of that image. Thus, by comparing the geo-location of a specific image and façade plane of a 3D model, the texture is mapped onto the 3D model. If this doesn't provide significant results, then the edges of the façade plane and the edges of the texture image are matched, and the texturing process is repeated.

Roads are mapped as a string on the OSM server. For the coordinates, four different points of different locations are tagged as a road. To map the texture on them specific file is selected from the reconstructed texture database. Once the file is selected, geo-locations are matched for the texture image and string of road. After matching, the texture image is mapped onto the façade plane of road structure.

In the proposed methodology, texture mapping of the 3D building is necessary because of the specific usage of the 3D model. The 3D city model is compared with the real-time image of buildings. In the comparison process, 3D features are extracted for the virtual environment and real-time image and stored in a repository. With the matching of 3D features, location of the building is confirmed, and it is eliminated from the objects detection process. In the 3D feature matching process, if 3D features of the virtual environment do not match with the 3D features of real-time image, then the building location is not confirmed, and that creates a mismatch for the static object elimination process. For this specific use of the 3D model, the textures are necessary on the building façade.

In Figure 4.15, 4.16, and 4.17 different views of the 3D model are shown with textures mapped on buildings.

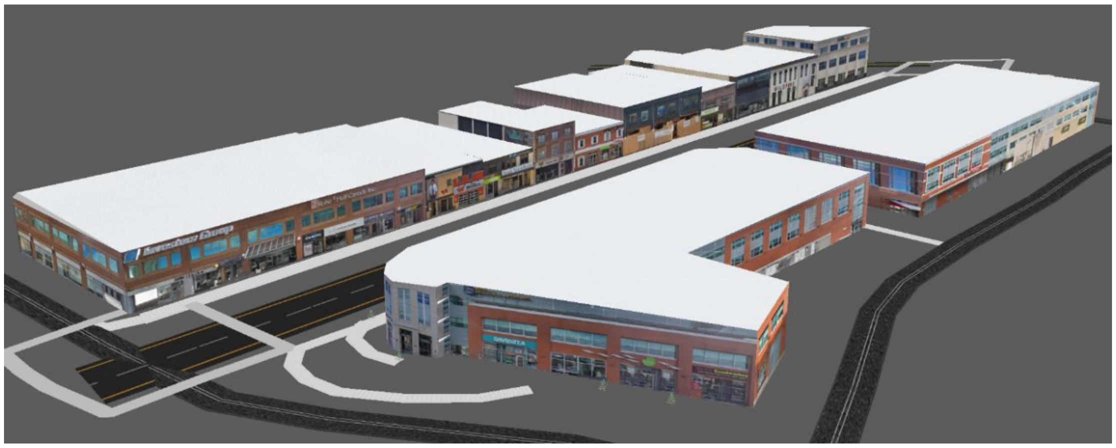


Figure 4.15. Textured 3D model (view 1)

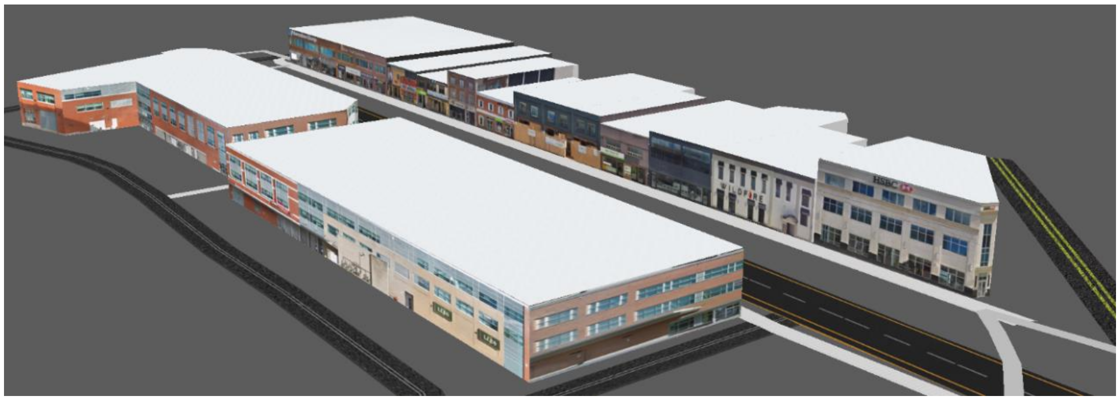


Figure 4.16. Textured 3D model (view 2)

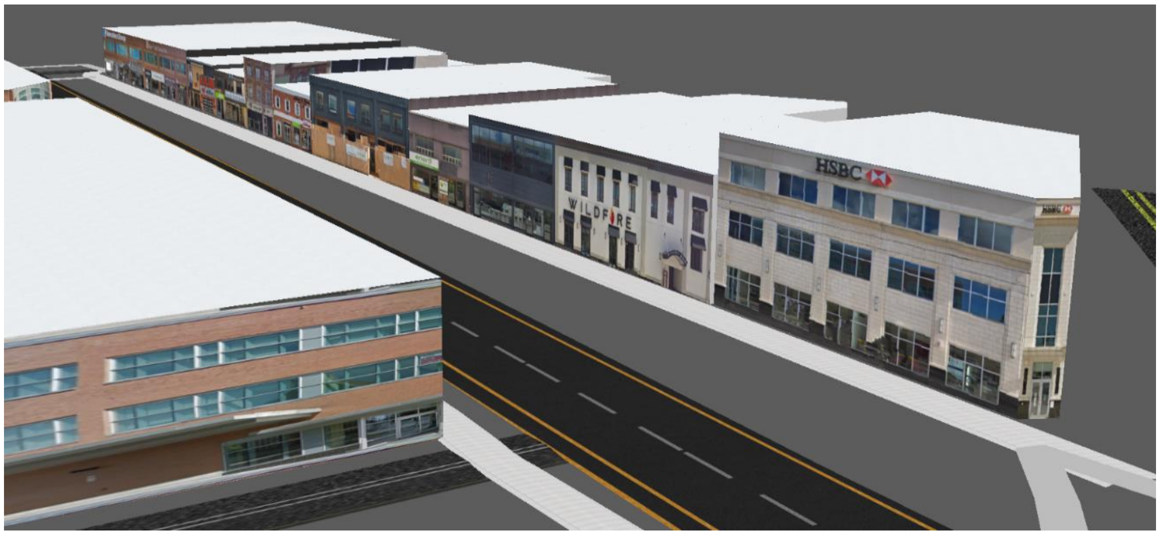


Figure 4.17. Textured 3D model (view 3)

## 4.9 Constructed 3D model in real-world

To check the geometry size of constructed model, the final model is imported into Google SketchUp and mapped onto a digital map. As seen in Figure 4.18, the edges of the 3D model match with the digital map layer.

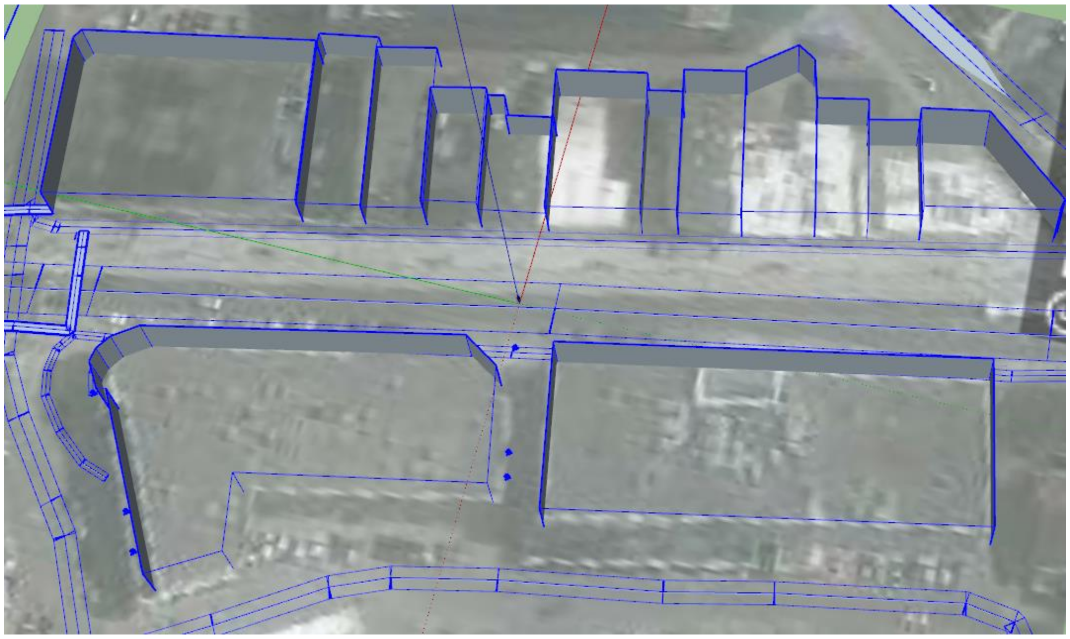
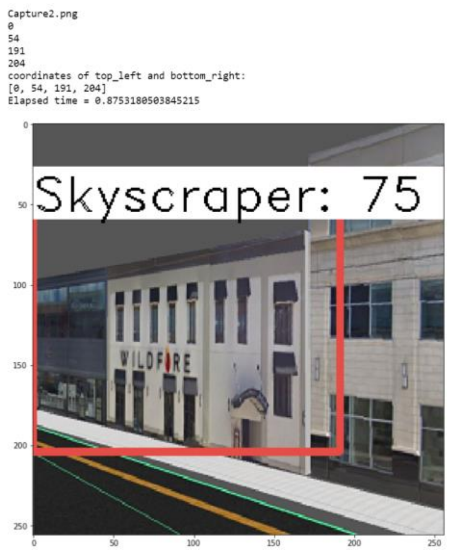


Figure 4.18. 3D model geometry comparison [73]

## 4.10 Use of 3D model in the overall system

The constructed 3D model is used as prior information in a self-driving car system. From the 3D environment buildings are detected. From the detected buildings, 3D features and keypoints are extracted. The detected building is shown in Figure 4.19.



```
Capture2.png
0
54
191
204
coordinates of top_left and bottom_right:
[0, 54, 191, 204]
Elapsed time = 0.8753180503845215
```

Figure 4.19. Building detection

Once the building is detected, heat map is generated to remove the buildings connected with it. After the heat map is generated, as shown in Figure 4.20 (left side), 3D features are detected and saved in a repository.

Once the 3D features and keypoints are extracted and stored in a repository, 3D features of the real-time image are extracted. The real-time image is of same building as building from virtual world. In this way, the 3D features of a virtual image and the 3D features of a real-time image are matched to confirm the location of building. 3D features detected on real-time image are shown in Figure 4.20 (right side).

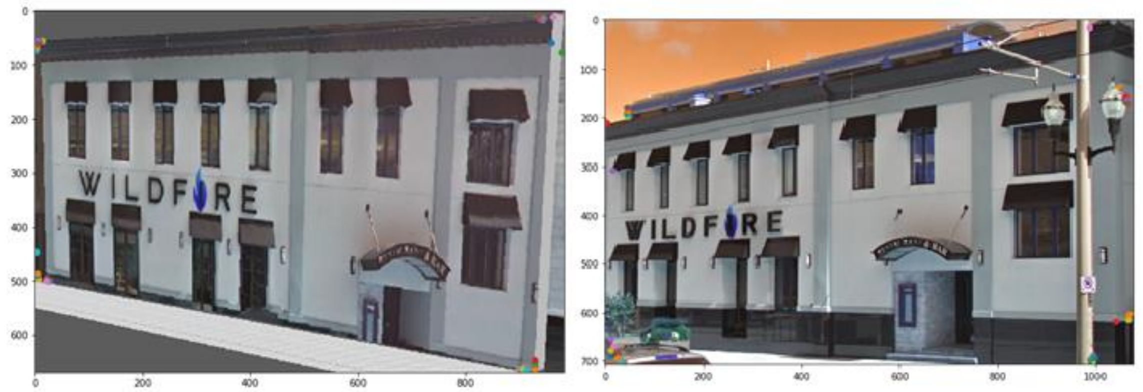


Figure 4.20. 3D features of buildings (virtual image in left; real-time image in right)

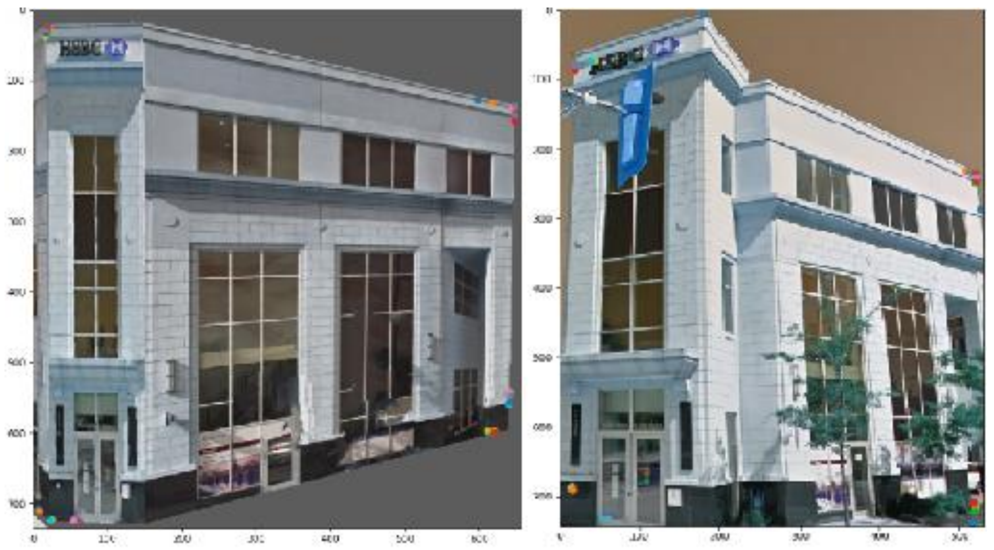


Figure 4.21. 3D features of buildings (virtual image in left; real-time image in right)

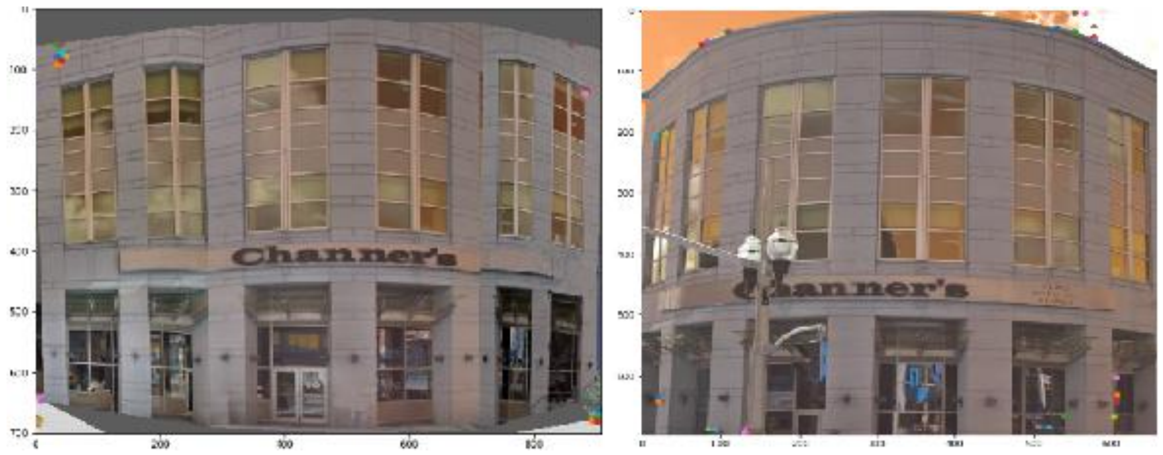


Figure 4.22. 3D features of buildings (virtual image in left; real-time image in right)

Once confirmed, all the static objects from the virtual environment are eliminated from the process of object detection to reduce the computation time for object detection.

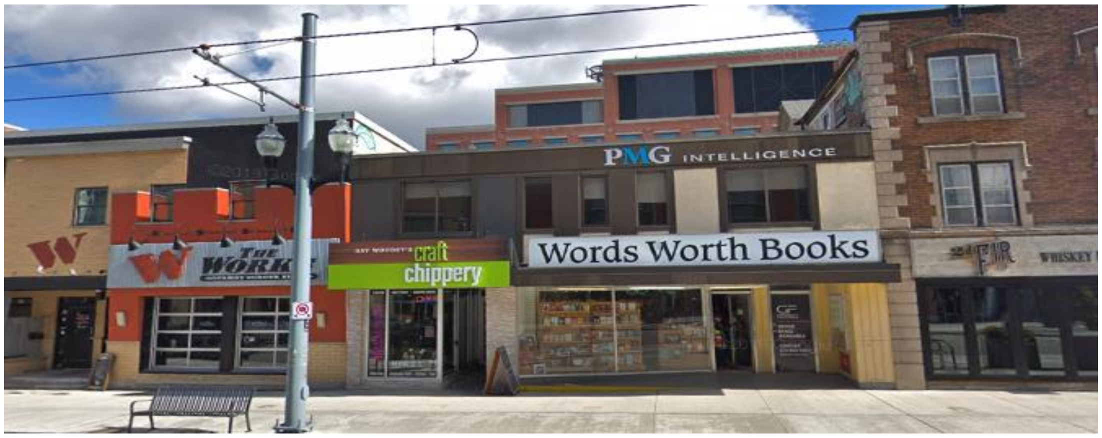
## 4.11 Result comparisons and discussions

3D city model constructed with the proposed methodology is primarily used to provide information of the static objects present in the city. Aim of the overall system is to use the 3D city model and eliminate the static objects from the object detection process. The elimination reduces the computation time for static object detection and provides extra time for the detection of dynamic objects. The methodologies referred in Section 2.3 aim to construct a 3D city model from different resources. As [65] uses OpenStreetMap data for 3D structural information with the combination of LiDAR data for the height of building object. With the use of the sensor, the overall cost of the system increases. They also use Government provided open data in the construction process. The availability of open-source Government data is uncertain as that depends on individual city. The approach presented by [68] does not use the opensource data for the construction of 3D model. As an input, aerial and terrestrial images are provided to the proposed system. The approach is based on image matching algorithms. The approach provides significant results, but it is mainly based on the usage of the sensors. Approaches suggested in [45] and [46] provide good results by using OpenStreetMap data for the 3D construction, but they do not provide any information about the texture mapping of building facades.

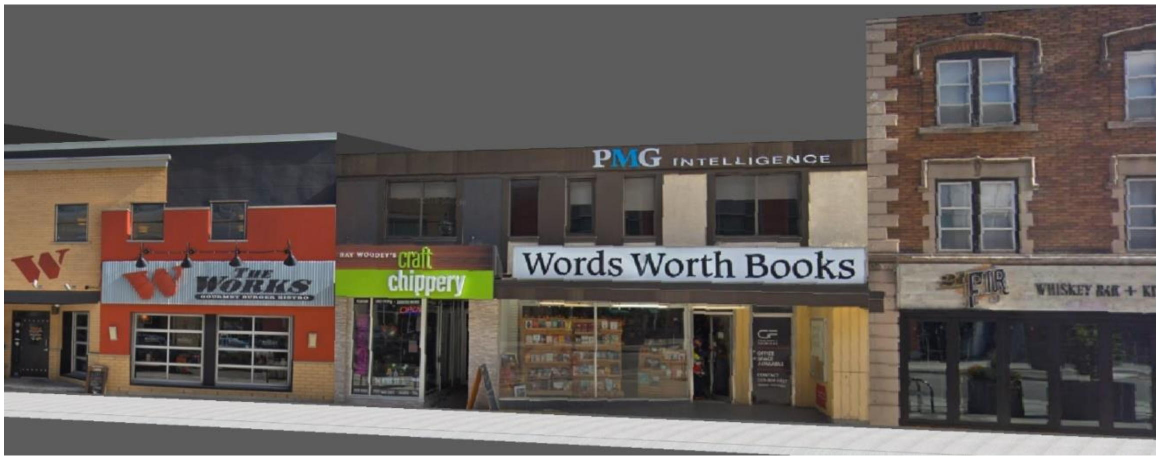
To texture map the 3D model, existing approaches use a mobile mapping system or computer generated textures to map the building façades. The mobile mapping system consists of a camera and GPS sensor. Computer generated textures are mainly used in virtual tour or gaming industry. Here, the 3D model is used as prior information for the navigation of a self-driving car.

The proposed method in this research work only uses OpenStreetMap data to extract 3D structural information. For the texture purposes, street-view imagery is used to reconstruct the building façade. The results from Section 4.4 and 4.10 provides information about the working of this approach. In Section 4.4, the 3D model is generated from the OSM data. In Section 4.10, provides different views of the constructed 3D model.

Moreover, to compare the quality of the 3D model, street-level image of a building and same building from virtual 3D environment is shown in Figures 4.23 and 4.24 respectively.



4.23. Street-level image of buildings



4.24. Virtual image of 3D objects

Aim with the construction of 3D model has been to extract 3D features from the virtual environment and then compare them with the features of a real-time image. With this, real-time location of the car is known. Also, by matching the 3D features, static objects are detected and removed from detection process.

Approaches discussed in Section 2.3 provide significant results. Here, the 3D environment is used for the specific situation of providing prior information to the self-driving car. For that, the 3D model constructed with the proposed approach provides significant results. Section 4.10 provides different situation with for the usage of 3D model.

## 4.12 Drawbacks and limitations

In the proposed methodology, the construction of the 3D virtual environment is carried out with opensource data; which is crowdsourced VGI data from OpenStreetMap. One of the limitations of the OSM data is its availability; it is available at almost all the places, but the quality of the data is not the same. At some places, objects such as benches, night lamps, and some of the trees are not mapped into the database.

In the database, roof types and height is not mentioned, that has been a major limitation in the proposed method. Without the roof portion, accuracy for the matching of 3D features and key points between the virtual environment and real-time image reduces; because the main difference between the virtual image and real-time image is roof portion of the building .

In the texture extraction process, the textures are extracted from Google street-view imagery. Herein, only the images captured by Google's mobile mapping system are extracted. If there is any street, which has not been captured by MMS, those texture images cannot be acquired by the proposed methodology. Also, sometimes in the occlusion removal process, if the image quality is low, then the quality of the final texture images reduces because the neighboring pixels are used to fill the removed region of the image. The drawback of unavailability of texture image is shown in Figure 4.25. Texture for the back side of the buildings has not been mapped due to lack of availability of texture images.

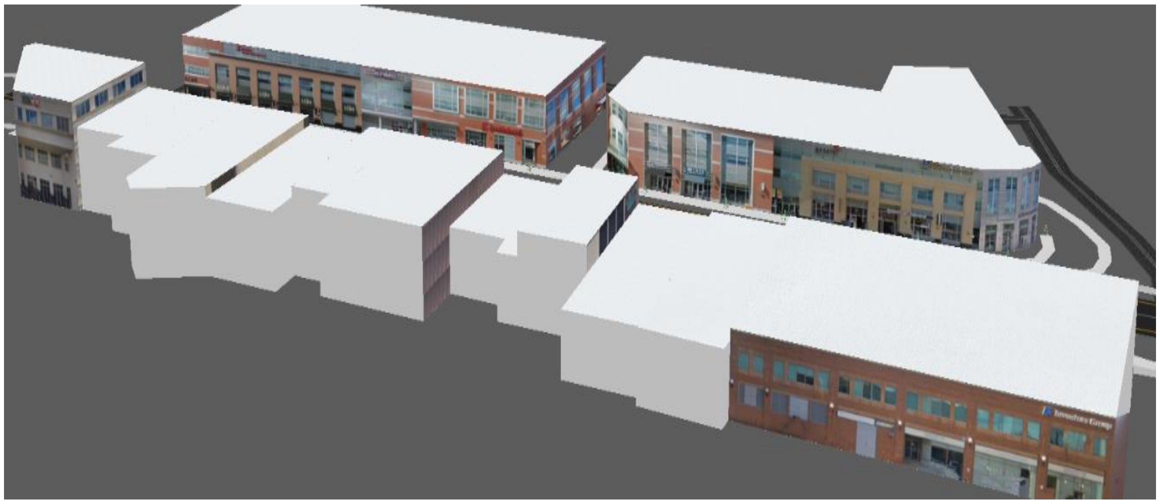


Figure 4.25. Drawback texture image

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

Most of the leading car companies today are looking forward to making the dream of a fully autonomous car a reality, with an intention to create a huge impact or a revolution [52]. There are several advantages that an autonomous car offers. These include less traffic, increased safety, and less wastage of time on driving. However, for the self-driving system to function with zero faults or accidents, it is required that the system knows the surroundings of the car. This requires the car to understand the already existing static objects on the road. The aim of the proposed system is to construct a 3D environment, which can be passed to the self-driving system as prior information; this helps the system understand the surroundings.

On the other hand, with the increasing number accidents by the self-driving systems, the virtual 3D model can be used to eliminate the static objects from the object detection process. This saves time. The saved time can also be very useful in a critical decision making situation. The constructed virtual environment uses the OpenStreetMap data, which does not use any of the sensors to map the information. The 3D features are extracted from the virtual environment, which are matched with the 3d features of real-time image; this helps the self-driving system geo-localize the car into the real-world.

Even though the virtual 3D environment has been constructed with the use of crowdsourced data, results from Sections 4.8, 4.9, and 4.10 prove that constructed 3D environment provides significant outputs for the extraction of keypoint features and 3D features. Also, the extracted keypoints and 3D features are matched with the features of real-time images to eliminate the static objects during the run of the car. This reduces the time for object detection. This time can be invested in the detection of dynamic objects and their prediction.

## 5.2 Future work

1) As the virtual 3D environment does not have a roof structure on top of buildings, one of the future works could be to construct roof structures either with the availability of 3D structural data or with 3D construction with multiple 2D images.
2) Currently, textures are unavailable for specific areas. In the future, an alternate data source can be merged with the system for the extraction of textures from specific areas; which would result in increased accuracy of the 3D environment for the detection of static objects.

## REFERENCES/BIBLIOGRAPHY

1. Singh, S. P., Jain, K., & Mandla, V. R. (2013). Virtual 3D city modeling: techniques and applications. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* (2), 73-91.
2. Zlatanova, S., Painsil, J., & Tempfli, K. (1998). 3D object reconstruction from aerial stereo images.
3. Kobayashi, Y. (2006). Photogrammetry and 3D city modeling. *Digital Architecture and Construction*, *90*, 209.
4. Shashi, M., & Jain, K. (2007). Use of photogrammetry in 3D modeling and visualization of buildings. *ARPN Journal of Engineering and Applied Sciences, 2*(2), 37-40.
5. Leberl, F., Irschara, A., Pock, T., Meixner, P., Gruber, M., Scholz, S., & Wiechert, A. (2010). Point clouds. *Photogrammetric Engineering & Remote Sensing*, *76*(10), 1123-1134.
6. Ainah, A. N., & Halim, S. (2010). Integration of Aerial and Close-Range Photogrammetric Methods for 3D City Modeling Generation. *Geoinformation Science Journal*, *10*(1), 49-60.
7. Hammoudi, K., & Dornaika, F. (2011). A featureless approach to 3D polyhedral building modeling from aerial images. *Sensors*, *11*(1), 228-259.
8. Flamanc, D., Maillet, G., & Jibrini, H. (2003). 3d city models: an operational approach using aerial images and cadastral maps. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, *34*(3/W8), 53-58.
9. Lang, F., & Forstner, W. (1996). 3D-city modeling with a digital one-eye stereo system. In *In Proceedings of the XVIII ISPRS-Congress*.
10. Pollefeys, M., Koch, R., Vergauwen, M., & Van Gool, L. (2000). Automated reconstruction of 3D scenes from sequences of images. *ISPRS Journal of Photogrammetry and Remote Sensing*, *55*(4), 251-267.
11. Jang, K. H., & Jung, S. K. (2006, June). 3D city model generation from ground images. In *Computer Graphics International Conference* (pp. 630-638). Springer, Berlin, Heidelberg.

12. Döllner, J., Baumann, K., & Buchholz, H. (2006). *Virtual 3D city models as foundation of complex urban information spaces* (pp. 107-112). na.
13. Cornelis, N., Leibe, B., Cornelis, K., & Van Gool, L. (2008). 3d urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision*, *78*(2-3), 121-141.
14. Snavely, N., Seitz, S. M., & Szeliski, R. (2008). Modeling the world from internet photo collections. *International journal of computer vision*, *80*(2), 189-210.
15. Xiao, J., Fang, T., Zhao, P., Lhuillier, M., & Quan, L. (2009, December). Image-based street-side city modeling. In *ACM transactions on Graphics (TOG)* (Vol. 28, No. 5, p. 114). ACM.
16. Gruen, A., & Wang, X. (1998). CC-Modeler: a topology generator for 3-D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, *53*(5), 286-295.
17. Al-Hanbali, N., Al Bayari, O., Saleh, B., Almasri, H., & Baltsavias, E. (2006). Macro to micro archaeological documentation: Building a 3D GIS model for Jerash city and the Artemis Temple. In *Innovations in 3D Geo Information Systems* (pp. 447-468). Springer, Berlin, Heidelberg.
18. Al-hanbali, N., Fedda, I., Awamleh, B., & Dergham, M. (2006). Building 3D GIS Model of a University Campus for Planning Purposes: Methodology and Implementation Aspects.
19. Malumpong, C., Chen, X., & FoS, G. I. S. (2008). Interoperable three-dimensional GIS city modeling with geo-informatics techniques and 3D modeling software.
20. Ziboon, A. R. T., & Mohsin, A. N. (2009). 3-D Virtual Maps Production for Mosul City by USING GIS Techniques. *Engineering and Technology Journal*, *27*(9), 1775-1789.
21. Thompson, E. M., & Horne, M. (2010). 3D-GIS integration for virtual NewcastleGateshead.
22. Vincent Tao C., Hu Yong. (2002). 3D Reconstruction methods based on Rational Function Model, Photogrammetric Engineering and Remote Sensing.
23. Fraser, C. S., Baltsavias, E., & Gruen, A. (2002). Processing of Ikonos imagery for submetre 3D positioning and building extraction.

24. Kocaman, S., Zhang, L., Gruen, A., & Poli, D. (2006, February). 3D city modeling from high-resolution satellite images. In *Proceedings of ISPRS Workshop on Topographic Mapping from Space, Ankara, Turkey* (pp. 14-16).
25. Tack, F., Goossens, R., & Büyüksalih, G. (2009). Semi-automatic city model extraction from tri-stereoscopic VHR satellite imagery. In *ISPRS Workshop on Object Extraction for 3D City models* (Vol. 38, No. 3/W4, pp. 89-96).
26. Huang, X., & Kwoh, L. K. (2008). Monoplotting–A semiautomated approach for 3D reconstruction from single satellite images. *Int. Arch. Photogramm., Rem. Sens. & Spatial Inf. Sc*, *37*(B3b-2), 735-740.
27. Izadi, M., & Saeedi, P. (2011). Three-dimensional polygonal building model estimation from single satellite images. *IEEE Transactions on Geoscience and Remote Sensing*, *50*(6), 2254-2272.
28. Luhmann, T., & Tecklenburg, W. (2004). 3-D object reconstruction from multiple-station panorama imagery. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *34*(5/W16), 8.
29. Micusik, B., & Kosecka, J. (2009, June). Piecewise planar city 3D modeling from street view panoramic sequences. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2906-2912). IEEE.
30. Clipp, B., Raguram, R., Frahm, J. M., Welch, G., & Pollefeys, M. (2008, March). A mobile 3d city reconstruction system. In *Workshop on Virtual Cityscapes, IEEE Virtual Reality*.
31. Zhang, G., Jia, J., Wong, T. T., & Bao, H. (2009). Consistent depth maps recovery from a video sequence. *IEEE Transactions on pattern analysis and machine intelligence*, *31*(6), 974-988.
32. Gruen, A., Zhang, L., & Wang, X. (2003). 3D city modeling with TLS (Three Line Scanner) data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *34*, 24-27.
33. Vosselman, G., & Dijkman, S. (2001). 3D building model reconstruction from point clouds and ground plans. *International archives of photogrammetry remote sensing and spatial information sciences*, *34*(3/W4), 37-44.

34. Li-Chee-Ming, J., Gumerov, D., Ciobanu, T., & Armenakis, C. (2009, September). Generation of three dimensional photo-realistic models from LiDAR and image data. In *2009 IEEE toronto international conference science and technology for humanity (TIC-STH)* (pp. 445-450). IEEE.
35. Dorninger, P., & Pfeifer, N. (2008). A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors*, *8*(11), 7323-7343.
36. Blaer, P. S., & Allen, P. K. (2009). View planning and automated data acquisition for three-dimensional modeling of complex sites. *Journal of Field Robotics*, *26*(11-12), 865-891.
37. Böhm, J., & Haala, N. (2005). Efficient integration of aerial and terrestrial laser data for virtual city modeling uusing lasermaps.
38. Habib, A. F., Kersting, J., McCaffrey, T. M., & Jarvis, A. M. Y. (2008, July). Integration of lidar and airborne imagery for realistic visualization of 3d urban environments.
39. Früh, C., & Zakhor, A. (2003). Constructing 3d city models by merging aerial and ground views. *IEEE Computer Graphics and Applications*, *23*(6), 52-61.
40. Zhao, W., Nister, D., & Hsu, S. (2005). Alignment of continuous video onto 3D point clouds. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8), 1305-1306.
41. Wang, Y.; Schultz, S.; Giuffrida, F. Pictometry's proprietary airborne digital image system and its application in 3D city modeling. Int. Arch. Photogram. Remote Sens., 37, 1065-1070, 2008.
42. Alizadehashrafi B, Towards Enhancing Geometry Textures Of Three Dimensional City Elements. PhD Thesis, Faculty of Geoinformation and Real Estate Universiti Teknologi, Malaysia, 17-32, 59-74, 2012.
43. Lee, J., & Yang, B. (2019). Developing an optimized texture mapping for photorealistic 3D buildings. *Transactions in GIS*, *23*(1), 1-21.
44. Goetz, M.; Zipf, A. (2012). Towards defining a framework for the automatic derivation of 3D CityGML models from volunteered geographic information.

45. Fan, H. & Zipf, A. (2016). Modelling the world in 3D from VGI/Crowdsourced data. In: Capineri, C, Haklay, M, Huang, H, Antoniou, V, Kettunen, J, Ostermann, F and Purves, R. (eds.) European Handbook of Crowdsourced Geographic Information, pp. 435–446, London: Ubiquity Press.
46. Over, M., Schilling, A., Neubauer, S. & Zipf, A. (2010). Generating web-based 3D City Models from OpenStreetMap: The current situation in Germany. Computer Environment and Urban System (CEUS), vol. 34(6), pp. 496–507.
47. Gröger, G., Kolbe, T. H., Czerwinski, A. & Nagel, C. (2008). OpenGIS® City Geography Markup Language (CityGML) Implementation Specification. Available at: http://www.opengeospatial.org/legal/.
48. Biljecki, F., Stoter, J. E. (2017). Level of detail in 3D city models Filip Biljecki. Nederland: Delft University of Technology
49. Darms, M., Rybski, P., & Urmson, C. (2008b). Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, the Netherlands (pp. 1192–1202). IEEE
50. X. Hu, L. Chen, B. Tang, D. Cao, and H. He. (2018). ‘‘Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles,’’ Mech. Syst. Signal Process.
51. Self-driving Cars Are Still Years Away. That's Probably A Good Thing. Michael Hobbes - https://www.huffingtonpost.ca/entry/autonomous-vehicles-uncertain-future_n_5d4c71f4e4b09e7297435cd4
52. What Is a Self-driving Car? The Complete Wired Guide Alex Davies - https://www.wired.com/story/guide-self-driving-cars
53. Self-driving Car: Path Planning To Maneuver the Traffic. /@jonathan_hui - https://medium.com/@jonathan_hui/self-driving-car-path-planning-to-maneuver-the-traffic-ac63f5a620e2
54. How Does Path Planning For Autonomous Vehicles Work - Dzone Iot Paul Ryabchuk - https://dzone.com/articles/how-does-path-planning-for-autonomous-vehicles-wor
55. Safe Central Compute https://www.nxp.com/applications/solutions/automotive/adas-and-highly-automated-driving/safe-central-compute:SENSOR-FUSION-SYSTEM

56. Pid Theory Explained https://www.ni.com/en-ie/innovations/white-papers/06/pid-theory-explained.html
57. An Introduction To Self-driving Cars FutureCar - https://www.futurecar.com/351/An-Introduction-to-Self-Driving-Cars
58. Haeberli, P., & Segal, M. (1993, June). Texture mapping as a fundamental drawing primitive. In *Fourth Eurographics Workshop on Rendering* (Vol. 259, p. 266).
59. Förstner, W.: 3D-City Models: Automatic and Semiautomatic Acquisition Methods. *Proceedings Photogrammetric Week '99*, pp. 291-303, Wichmann-Verlag, 1999.
60. RERUM NATURALIUM (2006). Real-time Visualization of 3D City Models.
61. Gkeli, M., Ioannidis, C., Potsiou, C. (2017). Review of the 3D Modelling Algorithms and Crowdsourcing Techniques - An Assessment of their Potential for 3D Cadastre. In: FIG Working Week 2017 – ‘’Surveying the world of tomorrow – From digitalisation to augmented reality’’, Helsinki, Filand, pp. 1-23.
62. S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi. (2018). Discovery of latent 3D keypoints via end-to-end geometric reasoning. In NIPS.
63. Rıza Alp Guler, Natalia Neverova, Iasonas Kokkinos. (2018). "DensePose: Dense Human Pose Estimation in The Wild".
64. DensePose COCO Dataset arXiv:1802.00434v1 [cs.CV] 1 Feb 2018. https://arxiv.org/pdf/1802.00434
65. Biljecki, F., Ledoux, H., & Stoter, J. (2017). Generating 3D city models without elevation data. *Computers, Environment and Urban Systems*, *64,* 1-18.
66. Zhongdi, Y. U., Hui, L. I., Fang, B. A., & Zhaoyang, W. A. N. G. (2018). 3D city model construction based on a consumer-grade UAV. *Remote Sensing for Land & Resources*, *30*(2), 67-72.
67. Yang, B., & Lee, J. (2019). Improving accuracy of automated 3-D building models for smart cities. *International journal of digital earth*, *12*(2), 209-227.
68. Kang, J., Deng, F., Li, X., & Wan, F. (2016). Automatic texture reconstruction of 3d city model from oblique images. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, *41*.

69. Wu, B., Xie, L., Hu, H., Zhu, Q., & Yau, E. (2018). Integration of aerial oblique imagery and terrestrial imagery for optimized 3D modeling in urban areas. *ISPRS journal of photogrammetry and remote sensing*, *139*, 119-132.
70. Openstreetmap. https://www.openstreetmap.org
71. Death Of Elaine Herzberg. https://en.wikipedia.org/wiki/Death_of_Elaine_Herzberg
72. https://developers.google.com/maps/documentation/javascript/streetview
73. 3d Design Software: 3d Modeling on the Web. https://www.sketchup.com/
74. List Of Self-driving Car Fatalities. https://en.wikipedia.org/wiki/List_of_self-driving_car_fatalities

# VITA AUCTORIS

| | |
|---|---|
| NAME: | Sumit Khairnar |
| PLACE OF BIRTH: | Ahmedabad, India |
| YEAR OF BIRTH: | 1996 |
| EDUCATION: | Bachelor of Engineering, 2013-2017<br>Gujarat Technological University, Ahmedabad, Gujarat, India<br>Master of Science in Computer Science, co-op, 2017-2019<br>University of Windsor, Windsor, ON |