

Enhancing spatial image datasets for utilisation in a simulator

presented by

LEPEKOLA IGNATIUS LENKOE

Dissertation submitted in fulfilment of the requirement for the degree:

Master of Engineering in Electrical Engineering

in the

Department of Electrical, Electronic and Computer Engineering

of the

Faculty of Engineering, Built Environment and

Information Technology

at the

Central University of Technology, Free State

Supervisor: Dr B Kotze: Doctor of Technologiae: Engineering: Electrical

Co-Supervisor: Mr P Veldtsman: Master of Technologiae: Engineering:
Electrical

Bloemfontein

2022

Declaration

I, **LENKOE LEPEKOLA IGNATIUS**, student number _____, do hereby declare that this research project which has been submitted to the Central University of Technology, Free State, for the degree Master of Engineering in Electrical Engineering, is my independent work and complies with the code of academic integrity as well as the other relevant policies, procedures, rules and regulations of the Central University of Technology, Free State and has not yet been submitted before by me in fulfilment of the requirements of any qualification.

Date: July 2022

Dedication

I dedicate this dissertation to my family and friends. A special dedication to my parents, Mahase and Molibalisio Lenkoe for their encouragement through tough times. This is also dedicated to my son, Obohlokoa Bokamoso.

Acknowledgements

I thank the Almighty God for granting me the opportunity to study and the strength to complete this research study.

I'm very profound and sincerely grateful and would like to convey my message of gratitude to my supervisor, and mentor, Dr Ben Kotze, who was genuinely supportive throughout this study. His continuous support, encouragement, knowledge, motivation and guidance was invaluable together with the contribution from my co-supervisor, Mr Pieter Veldtsman.

I further would like to acknowledge Dr Tshepo Kukuni for encouraging me to continue with my studies and for the efforts and time to make sure that my dissertation is of high quality and factual beyond reasonable doubt.

Abstract

The introduction of Google Street View, which is an integral part of Google Maps, has brought to the surface a method of roof-mounted mobile cameras on vehicles. This is regarded as one of the highly known and adopted methodically for capturing street-level images. Computer vision as one of the frontier technologies in computer science has allowed for the use of building artificial systems to extract valuable information from images. This approach has a broad range of applications in various areas such as agriculture, business, and healthcare.

This dissertation contributes to the development and implementation of Image-Based Rendering (IBR) techniques by presenting a method that makes use of hexagon-based camera configuration for image capturing. Upon the image capturing, each segmented image was stored in a specific folder relative to the camera number. Following this process, the images were chosen based on their timestamp and GPS coordinates and copied to a master folder where the rendering took place.

However, before rendering can take place, the master folder was called inside Blender software. The reason for placing the master folder inside Blender3D was to ensure smooth blending of different image datasets with fewer resources and low computing power during the rendering process. This is feasible as all the image datasets are in one folder as compared to the calling of multiple datasets from different directories which might affect the processing power. Subsequently, OpenCV algorithms were utilised for the Structure from Motion and points of cloud simulation. These techniques and algorithms were based on the available image datasets that were created in the master folder.

Following the optimal image rendering, a process of image blending took place inside the Blender3D software where the captured images (dataset) were rendered for utilisation in the simulator. The use of the Structure from Motion algorithm was utilised for the development of the dense point image, feature, and matching detection. Furthermore, the process for extraction of a depth map model from the three-dimensional (3D) mesh was also highlighted as well as the image restoration process utilising the 3D warping approach. In addition, after these processes were completed, the IBR technique was utilised again for rendering the scenes from the multiple datasets that were captured from the Hexagon Camera Configuration Model to present a scenery that can allow for bidirectional movement. It is therefore noted that the entire work done in this dissertation was substantiated using simulations, genuine data, and physical

analysis based on the physically gathered raw data and results from the analysis. The study objectives were therefore achieved by presenting a framework that allows for virtual driving and bidirectional movement of the scene from a Hexagon Camera Configuration Model. Furthermore, the image datasets showed an improvement in the visuals, spatial details and quality of the panoramic images for location identification based on GPS coordinates. Additionally, the rendered images were observed to be smaller than the originally captured images.

The study contribution was based on the GPS module which was utilised to observe and project the scene altitude and coordinates. Moreover, the contribution results process allows for free movement within the 3D-rendered scene to allow for back and forward motion as compared to a slide show that only allows for forwarding motion. In evaluating the efficacy of this research study, the objective argument highlights that through the use of a Hexagon Camera Configuration Model, the user is permitted to move in both the forward and reverse direction within a simulator as opposed to the one-directional movement. These results demonstrate the feasibility of utilising an alternative model for image capture as opposed to the utilisation of a 360° omnidirectional camera and image stitching protocol. Furthermore, the study results demonstrate that the more the input image data, the higher the realism of such a model. In contrast, for 364 image datasets, the output scene is high as a result of a large number of input image datasets with the scene realism observed for both points of cloud and mesh-based on 106110 points.

List of Figures

Figure 1.1: Street-view to overhead view image matching [10, p. 2].....	2
Figure 1.2: Geometric-based views of the driving environment [11].....	3
Figure 1.3: Intersection intervals methodology between perspective images	6
Figure 1.4: Architectural proposition for data collection and rendering using IBR technology.	7
Figure 1.5: An image capturing and acquisition overview.	8
Figure 2.1: Quantitative analysis of the relationships between depth and texture information, number of input images, and rendering resolution.	18
Figure 2.2: IBR difference between computer vision and graphics.....	19
Figure 2.3: Different rendering techniques.....	22
Figure 2.4: Geometric view of a GSV [62].....	24
Figure 3.1: System Architectural Model.....	28
Figure 3.2: Go Xtreme Rebel Full HD camera.	30
Figure 3.3: System components configuration setup in a stationary position.	33
Figure 3.4: System model layout.	35
Figure 3.5: Hexagon Architectural Camera Configuration Model.	36
Figure 3.6: Pinhole Camera Model.....	37
Figure 3.7: Hexagon camera configuration setup.	39
Figure 3.8: (a) Captured image dataset with timestamp and date; (b) Lossy image compression at 100% quality with file size 2.68MB; (c) Lossy image compression at 9% quality with file size 1.05MB.	42
Figure 3.9: Compression structure.....	43
Figure 3.10: Black and white test match on a chessboard.	48
Figure 3.11: Preview of the .xml directory file.....	49
Figure 3.12: a) Original captured image; b) Application of SIFT algorithm for keypoints detection.....	51
Figure 3.13: Feature detection using SIFT feature extraction and brute force matching.	53
Figure 3.14: Data points collection flow diagram.	54
Figure 3.15: Application of a point of cloud in an image.....	55
Figure 3.16: Mesh output from the Multiview stereo.....	60
Figure 3.17: Textured scene overview.....	60

Figure 4.1: Input captured image scenery.....	64
Figure 4.2: Feature detection and matching in Structure from Motion.	65
Figure 4.3: Point cloud and camera position reconstruction.....	66
Figure 4.4: Dense point of cloud with Multiview stereo.	67
Figure 4.5: Mesh output from Multiview stereo.....	68
Figure 4.6: Generated texture.	69
Figure 4.7: Model of the street without texture before simulation.	70
Figure 4.8: 3D-rendered street view.	71
Figure 4.9: a) Rendered scene in a left direction; b) Rendered scene in a reverse direction...72	
Figure 4.10: Comparison results between the captured and rendered image.	72

List of Tables

Table 2.1: Rendering comparison between IBR and MBR techniques [33].....	16
Table 3.1: Go Xtreme Rebel Full HD camera specifications.	30
Table 3.2: Acer Aspire ES14 laptop specifications.	31
Table 4.1: Efficacy comparison table based on image sample size.	74

Acronyms, Abbreviations

IBR	Image-Based Rendering
MBR	Model-Based Rendering
GPS	Global Position System
GSV	Google Street View
SID	Spatial Image Datasets
SFM	Structure from Motion

Keywords

Image-Based Rendering

Blender

Simulation

Image acquisition

Datasets

Street views

Model-Based Rendering

Image Processing

Computer Vision

Table of Contents

Declaration.....	ii
Dedication	iii
Acknowledgements	iv
Abstract.....	v
List of Figures.....	vii
List of Tables	ix
Acronyms, Abbreviations.....	x
Keywords	xi
Chapter 1: Introduction to the dissertation research.....	1
1.1 Introduction	1
1.2 Problem statement.....	4
1.2.1 The research question for this research study is as follows:.....	4
1.3 The objective of the research study.....	4
1.4 Fact-finding technique and design	5
1.5 Challenges of the research study	10
1.6 Expected outcomes.....	10
1.7 Publications and presentations during the study	10
1.8 Dissertation structure.....	11
Chapter 2: Overview of the study conducted on panoramic image and spatial data models	12
2.1 Introduction	12
2.2 Review on approaches of the existing systems	13
2.3 Technological inclination.....	17
2.3.3 Image recognition and identification	20
2.3.4 Image matching gauge	20
2.4 Different construction methods.....	21

2.4.1	Construction standard measurement	21
2.5	3D Panoramic image generation	23
2.5.1	Construction of panoramic image	24
2.5.2	Google Street View application interface	25
2.6	Summary	26
Chapter 3: Development of the image capture model and the use of IBR technique to render the captured images.....		27
3.1	Introduction	27
3.2	System apparatus selection.....	28
3.2.1	HD cameras.....	29
3.2.2	Acer laptop.....	31
3.2.3	USB hubs and power supply.....	31
3.3	The system`s software design process	32
3.4	System development process	34
3.4.1	Image capturing and collection.....	40
3.4.2	Image compression	40
3.4.3	Camera calibration.....	44
3.4.4	Image rendering procedure	50
3.4.5	Data simulation	61
3.5	Summary	61
Chapter 4: Results and discussions		63
4.1	Introduction to the results chapter.....	63
4.2	Input image results	63
4.3	Rendering simulation outcomes	65
4.4	Output simulation results	69
4.5	Conclusion.....	76
4.5.1.	Panoramic image rendering.....	76

4.5.2. Data simulation.....	77
Chapter 5: Conclusion & future work.....	78
5.1. Conclusions of the research done.....	78
5.2. Contributions from this research study	79
5.3. Final dissertation remarks	80
5.4. Suggested future work.....	80
References.....	82
Annexure A: Lossy flow chart and code algorithm	91
Annexure B: Code algorithm.....	94
Annexure C: Simulation of Hexagon Spatial Image Datasets for Free-Motion in Simulator for Smart City Bidirectional Navigation Purposes	98
I. Introduction.....	98
II. Problem statement.....	98
III. Objectives of this study is therefore to:	98
IV. Original contributions of this research paper.....	99
V. Literature Review	99
VI. Methodology.....	99
A. System apparatus setup.....	100
B. Image compression	101
C. Image calibration	101
D. Application of Structure from Motion Model	102
I. Results	103
IV. Conclusion	104
References.....	104
Annexure D: Enhancing Spatial Image Datasets For Utilisation in A Simulator for Smart City Transport Navigation	105
I. Introduction	105
II. Problem statement	105

III. Aims and objectives.....	2
IV. Literature review.....	2
V. Methodology.....	3
VI. Results.....	5
VII. Conclusion	6
Acknowledgment.....	6
References.....	6
Annexure E: Modelling Input.....	74
Annexure F: Feature Detection	75
Annexure G: Rendered output	76
Annexure H: Image Calibration Code.....	77

Chapter 1: Introduction to the dissertation research

This chapter appraises the introduction to the research work to be conducted. The problem statement is outlined detailing the research question for testing the feasibility for free movement within the 3D-rendered scene to allow for back and forward motion as compared to a slide show that only allows for forwarding motion. Additionally, the study objectives, limitations, and expected outcomes are also outlined.

1.1 Introduction

Over the years different kinds of techniques have been proposed for image data collection and image rendering. Such techniques include but are not limited to Image-Based Rendering (IBR) and Model-Based Rendering (MBR). However, according to the technical merits that are found in the IBR technique, the technique is deemed relevant and useful for utilisation in this research study. This is due to its benefits such as the ability to capture real-world effects and detail the imperfections of the real world [1]. It has been noted in the past few years that the IBR technique has gained much attention mainly in image processing, computer vision, and the computer graphics community due to its potential to create realistic images [2]. In addition to IBR interest in communities, IBR is also used within the spatial knowledge framework, such as in geoscience, urbanism, geography, etc. that are inherent to sustainable cities, reduction in air pollution, and improvement of human living conditions.

Additionally, to delineate the Virtual Map (VM) and real scenery, Google provides a service known as Google Maps that also possesses scenery called street view by which users can view the scenery as if they are driving, riding, or walking in it.

Street View has debilitated previous restrictions on the availability of data sources for evaluating streets [3, pp. 337-345], [4]. Furthermore, large-scale scenes within the cities are associated with MBR and IBR methods [5]. However, the scalability of these two techniques is their implementation and developmental capabilities for Google Street View.

In addition, the MBR technique is defined by depicting it as a classified rudimentary method for reconstructing a virtual view from any arbitrary viewpoint by using an explicit 3D geometric model and texture information about the scene, while IBR is a method that constructs a virtual

view by using several images captured beforehand [6, p. 87]. In an MBR system, the images can provide valuable information about the incident such as its location. The location has the exact Global Positioning System (GPS) coordinates, which can also be an estimation of the location.

In 2005 Google launched what is currently known as Google Maps (Google 2014c) which is a free-to-use web-based mapping service that combines conventional cartography maps with satellite imagery and high-resolution aerial photography [7, pp. 1-16]. Images from Google Street View provide geographical information and service with high coverage which shapes the street view as a true database. Furthermore, cross-view globalisation has been addressed utilising a technique or method called deep learning which was identified and observed through scientific results that outperform the hand-craft technique [8, pp. 5007-5015], [9, pp. 2794-2802]. Figure 1.1 presents the graphical representation of inquiring a query image to the reference database of the captured Google Street View imagery to find the match between a stored image and the query image presented by ref. [10, p. 2].

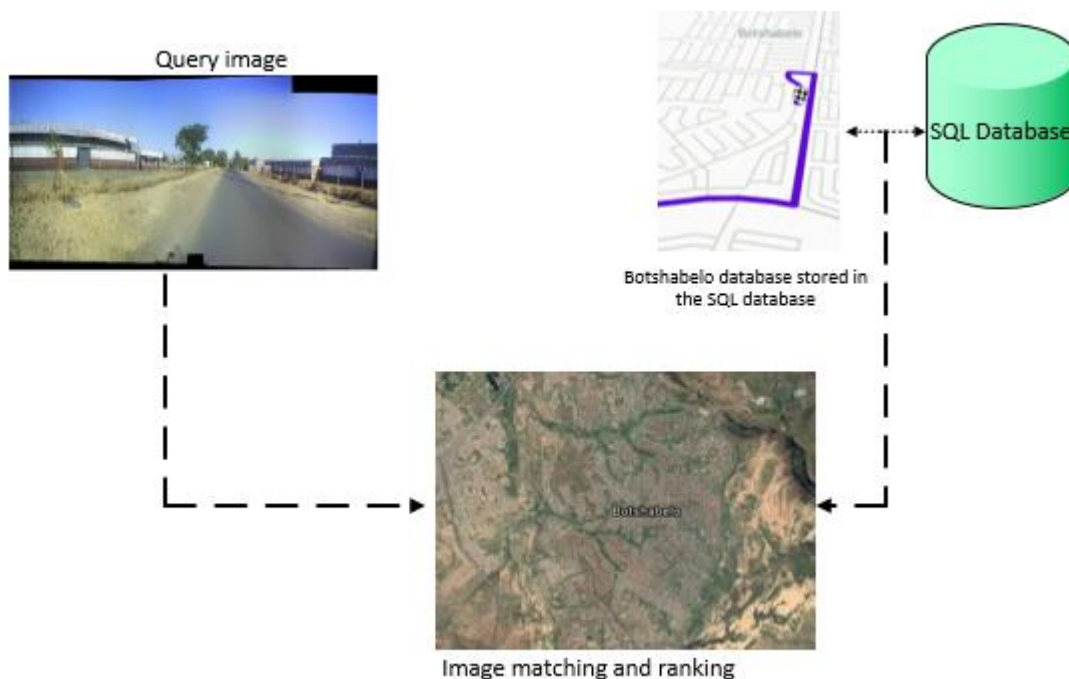


Figure 1.1: Street-view to overhead-view image matching [10, p. 2]

In addition to Figure 1.1, the conventional driving simulators in driving schools or railways provide the driver with the view in a geometric view as indicated in Figure 1.2 [11]. Figure 1.2

further outlines the merged roads based on the simulation of multiple images to design a view as indicated.



Figure 1.2: Geometric-based views of the driving environment [11].

Geometric-based views as presented in Figure 1.2 have a relatively less data size and their view is a result of poor photo-reality. It is for such reasons that the geometric-based rendering technique is not ideal for such a simulation. The selection of this rendering model is based against the traditional process-based 3D modelling and rendering. This can allow users the flexibility to change the external conditions with the time required for rendering increasing exponentially. The IBR model allows for the generation of rendering results of an unknown viewpoint by interpolating through discrete input images. As a result, geometric and IBR techniques are used according to their roles depending on the forecast objective of the study. Due to the reasons outlined earlier the use of the IBR technique is aimed to produce high output efficacy. This can be subjective, as the factual output result will indicate in Chapter 4. The model efficacy will also be outlined.

1.2 Problem statement

The 3D data acquisition process provides the probe position and orientation that remain in static order to produce accurate datasets. A single 3D capacity is not able to support the translational motion of the simulated probes, thus the need to develop a methodology for recording and capturing of single 3D images and amalgamating the images into multiple 3D image datasets within a single unit.

The issue of emulating street-view images for multiple image transitions for application in geolocalisation for utilisation in a simulator needs to be investigated. Additionally, the increase in the coverage renders the opportunity for image dataset scanning within the simulator. Hence the importance for investigating the feasibility of image rendering and permitting for bidirectional movement of the scene from multiple image capture in the simulator.

1.2.1 The research question for this research study is as follows:

Is it feasible to capture multiple images and render them for a bidirectional movement?

- This research study seeks to explore the feasibility for image capture by making use of the Hexagon Camera Configuration Model. The question further seeks to investigate the possibility and feasibility for creating a rendered scene that can be used in a simulator from a Hexagon Camera Configuration Model and permitting for bi-directional movement within the simulator.

1.3 The objective of the research study

This research study aims to:

- Incorporate the image dataset into the simulation system in real-time for increasing the reality of the simulation system in different geographical locations based on the Hexagon Camera Configuration Model.
- Simulate a rendering technique for improvement of visual and spatial images, and the quality of the panoramic images for location identification.

- Present a framework that allows for virtual driving and bidirectional movement of the scene.
- Model the image data collection technique using multiple cameras.

1.4 Fact-finding technique and design

There are other methodologies that are associated with image rendering and image smoothing transition [2]. Other methodologies include hybrid image-based rendering and pure image-based rendering encompassing 3D wrapping and layered depth images. In this research study, a Hexagon Camera Configuration Model is used and is prompted to achieve the same results despite the number of cameras utilised, and this output is made in comparison to the use of the omnidirectional camera. Figure 1.3 outlines the intersection of the image intervals between multiple images that are captured utilising multiple camera configuration models. Furthermore, this process can be denoted through the angular flow between multiple images. This research study makes use of a six camera configuration model at an angular distance of 60° between cameras.

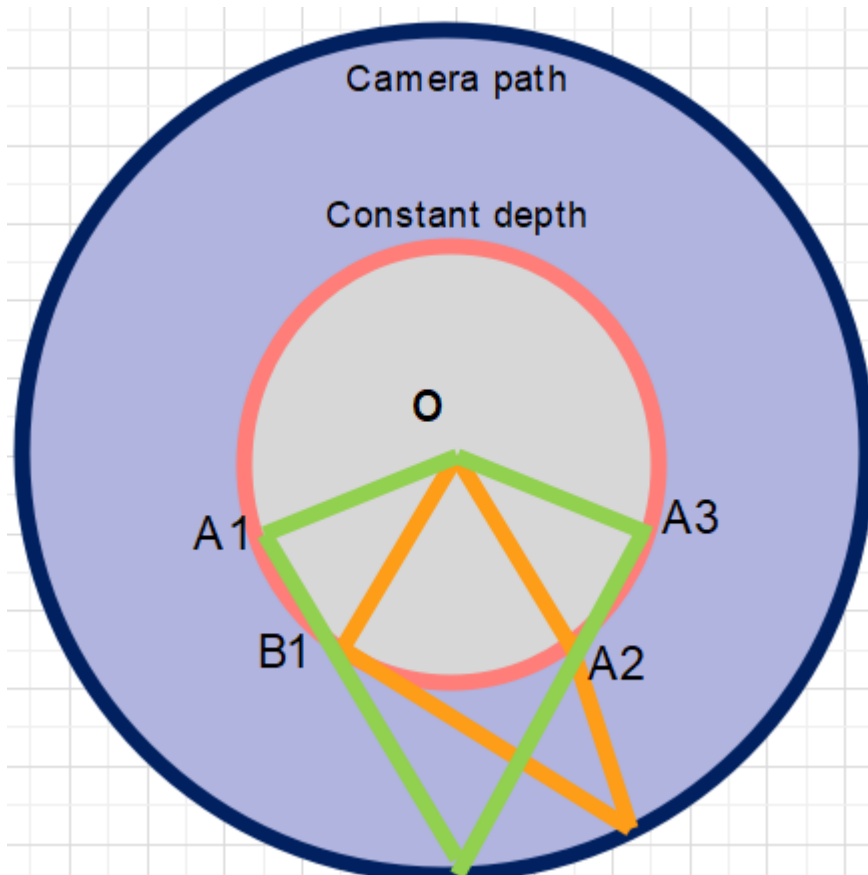


Figure 1.3: Intersection intervals methodology between perspective images

Figure 1.3 further enables the concept of image capture as outlined in Figure 1.4 where the camera setup model is highlighted for image camera capturing citing Figure 1.3 as the basic configuration model. O is the circumference of the camera holder. A (A1-A3) represents the camera placement, while B (B1) represents the camera lens holders and the intersection intervals between O and A images. The reason for adding an extra layer for the camera lens holder is to ensure that the lens is protected. Figure 1.4 depicts the camera configuration to be utilised in this research study to capture images. Additionally, this model can be altered depending on the testing site, and theoretically, the number of cameras does not affect the results depending on the study's aim and objectives.

Six (6) camera configuration model on top of a vehicle

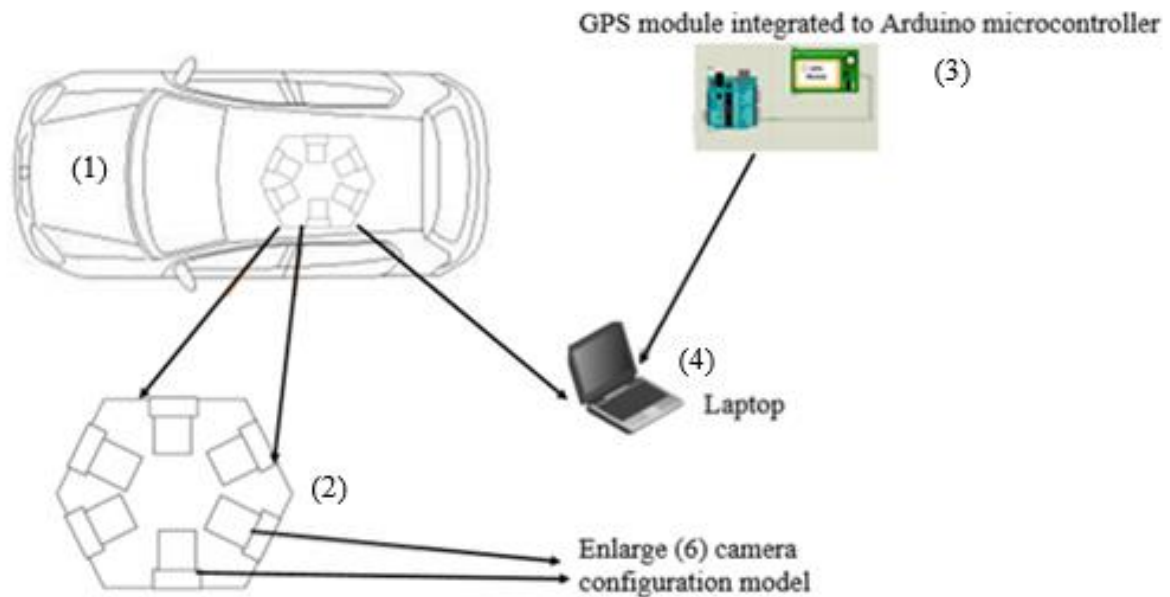


Figure 1.4: Architectural proposition for data collection and rendering using IBR technology

The preliminary data collection model is conducted by the use of a vehicle that is mounted with six (6) cameras at the top as indicated in Figure 1.4. In Figure 1.4 the components that are utilised are as follows: 1) test vehicle; 2) camera configuration model; 3) Arduino microcontroller integrated with GPS module; 4) laptop. However, the system modelling and development made use of Open-Source Computer Vision (OpenCV) and Unity3D for image processing, image compression, and image rendering processes. Additionally, from the camera formation, six (6) of the cameras will be placed in a hexagon formation at a 60° angular difference between the cameras to obtain a full 360° camera view. The selection for utilisation of a multi-camera configuration model, as opposed to the use of a 360° omnidirectional camera, is to test the feasibility for bidirectional movement of the scene within a simulator utilising other camera configuration models as opposed to the use of a 360° omnidirectional camera.

All six cameras will be connected to a laptop as depicted in Figure 1.4, and the laptop will be used for image storage, image data processing, and computation purposes. OpenCV is defined as a free computer vision library that allows for the manipulation of images and videos to accomplish a variety of tasks - from displaying the feed of a webcam to potentially teaching a robot to recognise real-life objects, and it will be utilised for image processing with a focal point

on image compression [12]. In Addition, Blender3D software will be used for image rendering and processing of the compressed image datasets. This process will be executed simultaneously, hence the need for a laptop with high processing power and multi-threading capability needs to be used. Each image that is obtained from the datasets will be Geotagged using a Global Positioning System (GPS) module which will also be attached to Arduino microcontroller. Arduino microcontroller will be used to process coordinates from the GPS module with the laptop running lossy compression algorithm for the compression of images of the same area (GPS-based images but different cameras to obtain the 360° view).

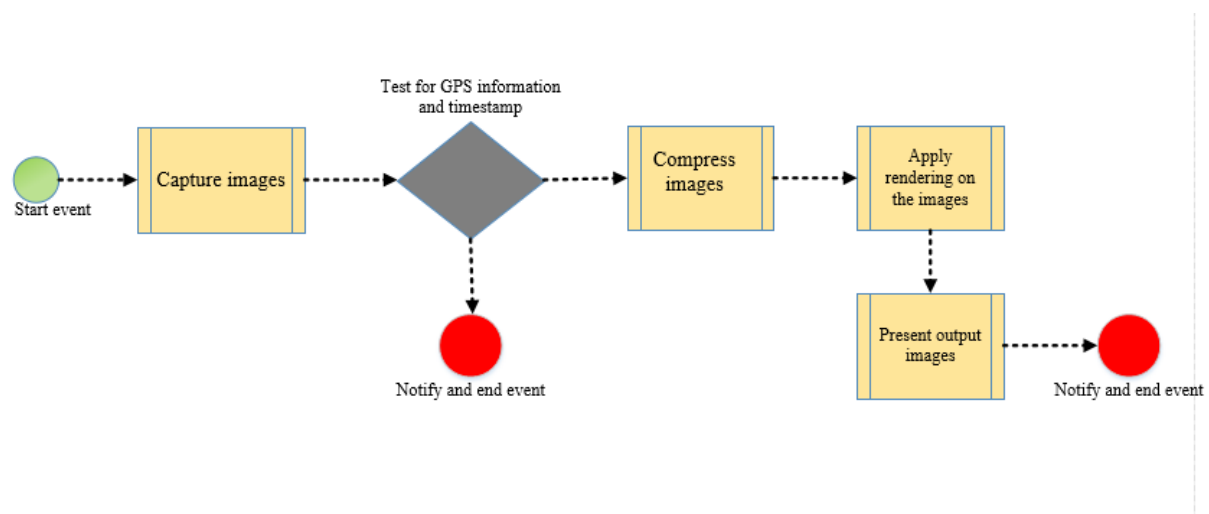


Figure 1.5: An image capturing and acquisition overview

Figure 1.5 outlines the methodological overview of the study representing the image acquisition process. Firstly, the process will be started by allowing the cameras to capture the images, and a code will be written in Python using OpenCV to allow for interval camera capture between six cameras. A mathematical approach for counting the camera capture interval will be discussed in the later chapters. During the image capture phase, the images are stored automatically into the folders assigned per camera, for example “camera 1 = storage folder 1”, and during this process it is important that the images have GPS coordinates attached to them. The reason for obtaining GPS coordinates is to ensure that the images can be merged into one dataset to obtain a 3D panoramic view. However, before the rendering and compression can occur, the images are sorted manually and placed into the master folder to create an image

dataset. Upon the successful image dataset creation, the dataset is called into Blender3D software for compression before the IBR technique can be applied.

Due to the test vehicle velocity and camera capture speed, oversampling might occur during the image capturing process. Therefore, a mathematical approach to reduce or mitigate that possibility needs to be implemented. Due to the study focus area and objectives oversampling is not accounted for, and as a result, it is the prerogative of the researcher to determine the number of images required to develop a full image dataset that can be rendered. The camera capture speed will, however, be explored in Chapter 3. There are several kinds of IBR techniques such as Vector Quantization (VQ) or Disparity Compensation Prediction (DCP) that may be considered for utilisation for this research study depending on the literature review chapter. These techniques can be altered depending on the results obtained during the development phase. Furthermore, for image rendering, the techniques Unstructured Lumigraph Rendering (ULR) and Light Field Rendering (LFR) will also be considered, since they make use of fewer geometry principles. The technique selection can also be altered depending on the results.

In addition, other non-geometric methods that utilises computer vision application such as plenoptic function which is defined as the intensity of light rays passing through the camera center at every location at every possible angle for every wavelength at every time [13], will be tested for a better algorithm aimed at producing satisfactory study results output.

The plenoptic function can be expressed as follows:

$$P_7 = P(L_x, L_y, L_z, \theta, \phi, \lambda, t) \quad (1.1)$$

Where:

(L_x, L_y, L_z) = GPS location

(θ, ϕ) = angle for forward movement

λ = wavelength between the eye and the object focal point

t = time spent looking in the same direction

All the variables are highlighted in the plenoptic definition.

The images will be projected from the laptop, and keyboard inputs will be used to navigate through the rendered scene for both forward and backward movement. Furthermore, the GPS

information (e.g. longitude, latitude, and elevation) from the scene will be written to a text file which will be stored in a specified directory.

Cylindrical panorama will be utilised due to its ease-to-build method for the single unit camera, and due to the research proposition, the cylindrical panorama will be modified to accommodate multiple camera input sources. This is due to the study objective focusing on the development of a panoramic simulation model using image data from the multiple camera feed.

1.5 Challenges of the research study

- The development of an automated data integration algorithm into the simulator.
- Incorporation of the fisheye camera image sample into the dataset due to the test environment.

1.6 Expected outcomes

- The incorporation of a 3D panoramic model into the simulation model in real-time for increasing the reality of the simulation system in different geographical locations.
- Simulating a rendering technique for improvement of visual and spatial images and quality of the panoramic images for location identification.
- Development of a technique that allows for bidirectional movement within a simulator from a Hexagon Camera Configuration Model.

1.7 Publications and presentations during the study

- Lepekola Lenkoe, Ben Kotze: “Enhancing Spatial Image Datasets for Utilisation in a Simulator for Smart City Transport Navigation” The Tenth International Conference on Smart Cities, Systems, Devices and Technologies” 30 May – 03 June 2021, Valencia, Spain.
- Lepekola Lenkoe, Ben Kotze, Pieter Veldtsman: “Simulation of Hexagon Spatial Datasets for Free Motion in a Simulator for Smart City Bidirectional Navigation

Purposes” The Sixth International Conference on Applications and Systems of Visual Paradigms” 18 July – 22 July 2021, Nice, France.

1.8 Dissertation structure

This dissertation has been arranged into five chapters, consisting of the introductory chapter, literature chapter, methodology chapter, results chapter, and finally the conclusion chapter.

Chapter 1 outlines the introduction to this dissertation focusing on the background, problem statement, objectives, methodology, and expected outcomes for the research conducted that are consolidated to explain the reason for conducting this study.

Chapter 2 presents the overview of the different methodologies and studies conducted that relate to spatial data, image rendering techniques, and the current studies focusing on the method used for data collection using datasets in the simulator.

Chapter 3 presents the design and modelling techniques for IBR rendering, image capturing techniques, and the configuration model focusing on camera calibration and the manipulation of imaging techniques such as image acquisition and image compression to obtain noise-free images before image rendering can occur.

Chapter 4 outlines the results obtained from the model developed in Chapter 3. In this chapter, the results are evaluated, analysed, and discussed with reference to the study's aims and objectives.

Chapter 5 outlines the dissertation's conclusion based on the method and results obtained. This chapter further focuses on the research contribution to the field of computer vision and the application of 3D panoramic image rendering techniques on the simulators. This chapter further presents the dissertation remarks and suggests the future works emanating from this research study that were not addressed due to the research scope and also presents the shortcomings.

Chapter 2: Overview of the study conducted on panoramic image and spatial data models

This chapter outlines the theoretical overview of the studies conducted on panoramic images using spatial data modelling techniques. Furthermore, different literature is also outlined together with the shortcomings and advantages from the results obtained by other authors in this field of study.

2.1 Introduction

This chapter evaluates the theoretical background relating to the currently available methods and techniques that are used for image data collection and image rendering techniques such as IBR and MBR techniques. The use of such datasets in a simulator is mainly for image construction and reconstruction and also image rendering that are viewed relevant for this research study. Additionally, the camera content acquisition in a 3D content view can be viewed as an alternative modelling approach for such a study. In this chapter, several methods and techniques are reviewed, analysed, and implemented to address the research question.

The study of Li and Ratti [14, pp. 109-119] resonates with this current research study to a certain degree, where the author outlines the significance of cities in the context of global warming and urbanisation that is also interpreted and analysed for further developments. Furthermore, Li and Ratti outline the significance of having both the ID and data information of the image. However, this thought process depends on the research objective. For this research study, it is important to have the Global Positioning System (GPS) coordinates on each image, since multiple image datasets will be compressed. In addition, Martinez-Carranza *et al.* [15] present an image data capture model that makes use of a drone in real time. The author makes use of image mosaicking and 3D point cloud generation as well as the tessellation. The author does not clearly outline how many cameras were used to capture either the images or the video. Additionally, the system has a drawback of precision alignment when the number is high (in this context the author did not articulate or indicate what high implies). This chapter further discusses the historical background of Google Maps' development in conjunction with Google Street View.

2.2 Review on approaches of the existing systems

Typically, there are massive amounts of image data collections that are presented as slideshows which are arguably the practical way. However, with the current technological advancement, these methodological approaches are deemed not engaging, and as a result, new techniques and models need to be developed and implemented to investigate their efficiency and efficacy as compared to the slideshow models. The reason for the investigation of other models is due to the fact that the slideshow is deemed not to be engaging as it is one-sided, and the user cannot go back and forth within the view. Dong *et al.* [16] outline the fast development of Structure from Motion (SFM) and depth camera-based 3D scene reconstruction. Bianco *et al.* [17] presents the evaluation of Structure from Motion utilising the real dataset. In their study, these authors evaluate the reconstruction errors, as well as the estimation errors of the camera pose used in the reconstruction. This research study makes use of Structure from Motion that is utilised as a tool for converting 2D images into 3D image datasets and as a result, the error or loss of data during the conversion is not considered. Furthermore, Schonberger and Fram, [18] and Xu *et al.* [19] present the method for capturing high-quality geometric images of an indoor scene to improve even the mirrors and glasses. This paper's results make it possible to argue that image capturing in a controlled environment is more content as opposed to an outdoor controlled environment due to factors such as setting up of the ambient light. Hence the complexity of the current research study as it focuses on an outdoor setup. Additionally, the change in scenery for such a model is due to technological improvements, and this change has led to a wide study pool that also cites the research conducted by Sivic. Sivic *et al.* [20] highlight the connection of clustering visually similar images together to create a virtual space in which the users are free to change position from one image to another. This virtual space modelling can be obtained by utilising intuitive 3D control objects such as moving left/right, zooming in/out, and rotating. Sivic's study resonates with the current study to a certain degree, hence the importance of ensuring that each image has GPS coordinates to enable for change in position should the need arise.

Sivic further supports his work by outlining that the displayed images in a correct geometric and photometric alignment concerning the current photo results in a smooth transition between multiple images. In addition, Kopf *et al.* [21] present a method of combining images in the street view system by stitching the image side views. This approach means you have to be standing on the street and looking in either the left or right direction of a certain street together to generate

a long street slide for users to quickly browse if the street is feasible for motion or not, despite the foundational way of viewing the side scene of a certain street from Kopf's methodology. Furthermore, the practicality of this approach remains scientifically arguable, as this approach focuses on a one-sided approach that is not always the case while driving or walking, which opens a window for further research.

Peng *et al.* [22, p. 1] present a system that takes the start and endpoint as inputs and automatically connects them to Google Maps utilising street view to achieve the route planned result and scenery along the route. The system can generate smooth scenic video from the starting point to the destination and combine Google Maps to provide better route recognition to users. Despite the advantage of smooth scenic video that Peng proposes, the element of duration for downloading images from Google Maps still needs to be investigated. Furthermore, Peng's study represents the combination of Google Maps, which is seen as an addition of a few image datasets to the existing dataset rather than the capturing and the development of a full 2D dataset. There are other proposals that do not make use of the feature matching model but of metrics similarities which are calculated from the pixel values. Additionally, there are other techniques based on local traits that allow for the rapid search of the correspondence features, and this improves the image alignment accuracy and processing time [23], [24]. Despite the improvements outlined by Kokate *et al.*, [24], the use of feature detection and matching is still deemed as the best approach for IBR rendering, especially for multiple datasets. One of the different approaches is presented by Rzotkiewicz *et al.* [25] where the authors agree that GSV imagery strength includes low cost, ease of use, and it saves time. In retrospect to the study by Rzotkiewicz *et al.* [25], the research was based on the availability of literature on health research which resulted in the final review of 54 articles relevant to the health GSV. Furthermore, the lack of relevant or extended weakness of accessing the GSV might be as a result of high-definition imagery that may already be in some areas and not others, and the inconsistency of image resolution quality and date which may result in error inclusion/exclusion. Rzotkiewicz's study can be argued citing Coronelli and Rinaudo *et al.* [26] and Wahbeh *et al.* [27], stating that nowadays images and open data that are obtained online are increasing exponential with 3D reconstruction available from generic touristic photos gathered from the web and frames extracted from the videos.

Kopf *et al.* [28] present street slide methodology that combines the nature of bubbles provided by perspective stripe panoramas. Their study further presents an integrated annotation and a mini-map within the user interface to provide geographic information as well as additional

affordances for navigation. Kopf's work relates to that of Gortler *et al.* [29] due to their classic approaches of using image-based rendering such as the Lumigraph technique. Kopf's work is further supported by Agarwala *et al.* [30, pp. 853-561], by emphasising the utilising of the correlation alignment techniques for aligning adjacent vertical strips instead of modelling a full 3D geometric proxy.

Najafizadeh *et al.* [31, p. 1] present a methodology for utilising the Google Street View feature as a three-stage classification framework by manually labelling the accessibility problems in one time period. The classification of the labelled image patch into one of five accessibility categories, and the patch is then localised in all previous snapshots. Najafizadeh and his co-authors' work is supported by Serferling and co-authors, basing their arguments on image relationships. Serferling *et al.* [32] present a computer vision application that is utilised to qualify urban tree cover at street level by employing open-source image data of city streetscapes that are abundant by Google Street View. This application is achieved by displaying a computer vision algorithm segment that qualifies as a percentage of the tree cover in the streetscape images, as well as by modelling the relationship between neighbouring images along the city street segment. However, these authors have not presented their data collection model. In these papers, they only highlighted the use of Google Street View, but did not indicate the compatibility of the dataset with their applications.

Furthermore, is important to compare the two basic rendering models namely the IBR and MBR. The two rendering models rely primarily on the original and/or trained image dataset to produce new and virtual views. Table 2.1 depicts the comparison between MBR and IBR. Table 2.1 also gives context into the selection of the IBR as the relevant technique for utilisation in this research study.

Table 2.1: Rendering comparison between IBR and MBR techniques [33].

IBR technique	MBR technique
Direct use of collection of images	Explicit use of 3D models
Based on interpolation	Uses conventional rendering pipeline
Speed independence depends on scene complexity	Speed depends on scene complexity
Relies on processor speed	Relies on hardware accelerator speed
Realism depends on input images	Requires sophisticated software for realism

Table 2.1 depicts the difference between MBR and IBR techniques. As a result, the above-mentioned differences direct the selection of the appropriate technique such as realism, which requires sophisticated software on the MBR site, while on the IBR realism depends on the input data. Additionally, the cost of rendering in an IBR model is independent of the scene complexity, while in an MBR model the cost of rendering depends on the objects or complexity of the scene and the number of facets.

Furthermore, Voumard *et al.* [34], in their study, focused on the performance of an online imagery dataset utilising photographs. Voumard's study was based on a scaled imagery size of approximately 50 and 60 image datasets for the rock survey. In addition, the scaled datasets by Voumard resonated with Gribble *et al.* [35] and presented an approach to rendering large, time-varying particle-based simulation datasets using programmable graphics hardware on a desktop computer system. Yet, this study also focuses on image rendering.

Kang [36] and Lengyel [37] presented an IBR method that is based on a trade-off by estimating how many input images are needed and how much is known about the scene geometry. Kang's study also includes capturing or data collection based on nine camera configurations. A smaller number of research studies have been conducted in this discipline despite much traction in this field. Theoretically, the number of input camera configuration model does not change the required output, but rather depends on the data collection environment.

2.3 Technological inclination

The technological inclination section gives more context to the historical developments around the Google Street View and its influence on the modern era. The process of development and the background is outlined also quoting the shortcomings and how this technology can be improved citing the literature in this discipline.

2.3.1 Historical background on Google Street View

Google has revolutionised the online mapping environment by creating an efficient and immediate spatial data collection method. This method is supported by the affluent misuse of the principles of stereopsis presented by Hu and Ming [38, pp. 2161-2191]. Stereopsis in this context is denoted as a perception of the depth and the 3D structure obtained based on visual information derived from two eyes by individuals with normal developed binocular vision.

It is highlighted that Google Street View (GSV) was first put into action in May 2007 with the primary objective of capturing pictures in major cities. It is recommended that a third-party mode overpass the abstract map and the distant aerial view environments by providing street-level imagery. Google Street View is a model that presents the world as a fact, mapped, and documented view in an approximation of the street condition [39]. Furthermore, in the context of data collection for GSV, the images were collected (historical context) with a panoramic camera model mounted on a different type of vehicle or backpack as per this research study's data collection method of mounting cameras on top of a moving vehicle [40].

This technique of camera mounting or the utilisation of a backpack is also supported by Dragomir Anguelov *et al.* [41], and he also outlines the data capturing technique using the same technique of mounting cameras on top of a moving vehicle, but the difference is the number of cameras and the camera configuration technique that he uses. Additionally, the technique is also supported by Lenkoe *et al.* [42], who highlighted the advantages of using a Hexagon Camera Configuration Model as compared to the utilisation of the use of a 360° omnidirectional camera. In this research study the camera configuration model will be altered as per Lenkoe's study. Additionally, the details and reasons for altering the camera numbers will be discussed in the design chapter. The proposed camera configuration model is based on the test environment; hence the utilisation of the six cameras.

2.3.2 Image-based rendering techniques

This research study aims at exploring the IBR technique citing the work presented by Shum and Kang [43], who presented the advanced image pixel indexing methodologies utilising the IBR technique. The advanced IBR method was presented by Mao *et al.* [44] and Shi *et al.* [45], who outlined the depth map containing associated pixels to the reference image in a 3D environment. Furthermore, the concept has been explored more with the application development focusing on the estimation of depth values from a single monocular image value. The result of this application is seen as an issue due to depth estimation, as it requires a reformulation of (colour) image to depth image generation [46]. In addition to the parametric methods for extracting depths, many non-parametric depth sampling approaches have also been proposed to automatically convert monocular images into stereoscopic images with good performances. In addition to the real-world environments, these models can be generated using a 3D scanner or by applying computer vision techniques to the captured images. Unfortunately, vision techniques are not robust enough to recover accurate 3D models, as indicated in Figure 2.1. Additionally, this makes it difficult to capture visual effects such as reflections and transparency utilising a single texture-mapped model.

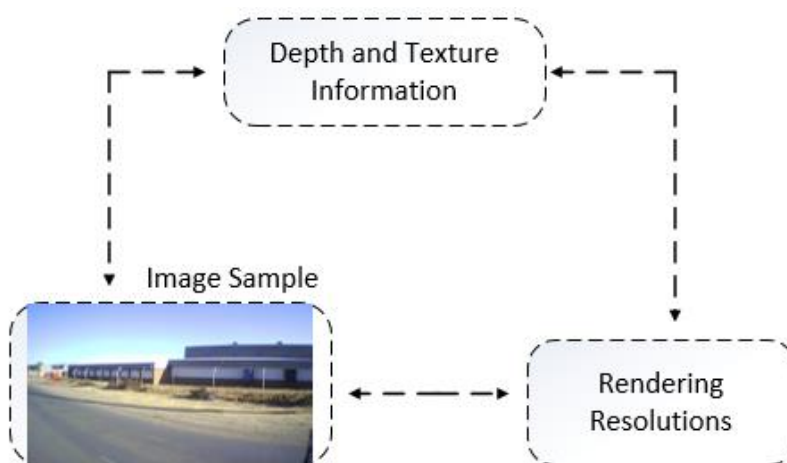


Figure 2.1: Quantitative analysis of the relationships between depth and texture information, number of input images, and rendering resolution

Figure 2.1 depicts the qualitative analysis of the relationship between depth and texture information. This is very essential in an image processing system where a certain type of input dataset in a specific type needs to be first put into a depth and texture model. Furthermore, as indicated in Figure 2.1, the input dataset can still be rendered directly, but in most cases the issue is around the processing time and system accuracy. However, such a process can be optimised by obtaining the visual effects of the reconstructed environment and implementing them in a view-dependent and texture mapping model. Nonetheless, the texture mapping process in this context is used for rendering new views by warping and compositioning several input image datasets of an environment.

Additionally, rendering techniques are associated with computer graphics to enable better processing of images [47, p. 25]. Many rendering techniques can be used to address such a topic; yet, this study focuses on the use of the IBR technique as compared to other traditional techniques. One of the fundamental reasons for this selection is the ability of the IBR technique to use 3D computer graphics and images as an elementary description, and also citing Table 2.1 for in-depth reasons. The IBR technique brings about a bridge in the gap between computer graphics and computer vision as indicated in Figure 2.2, which also adds to the motivation for the selection of this technique to be used in this research study.

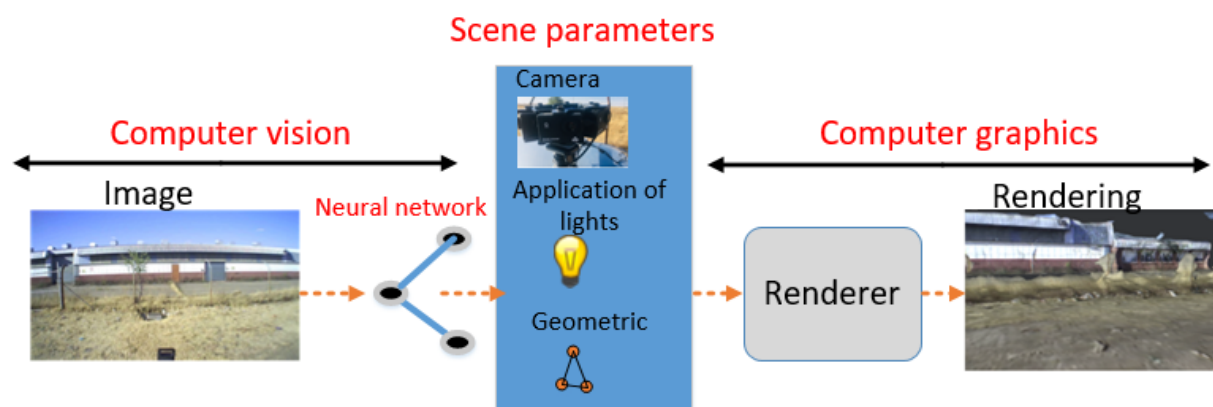


Figure 2.2: IBR difference between computer vision and graphics

Figure 2.2 depicts the difference between computer vision and computer graphics. Many people interpret the two as a single word. Computer vision is defined as a field of study that seeks to

develop techniques to help computers see and understand the content of digital image and videos [48], while computer graphics depict the creation or manipulation of images on a computer including animated images [49]. Furthermore, it is highlighted that some of the IBR techniques do equally manipulate the geometric data during the rendering process. This data manipulation has led to the current development of Google Maps demonstrating great attention to the transition between cartographic and photographic modelling techniques [50]. As a result, a wide window has opened for further research in this subject line. Furthermore, IBR classification in the context of this research study defines image recognition and how the image is matched. There are other classifications such as restraining the view, space, and the introduction of resource descriptors also resonate with the objectives of this research study.

2.3.3 Image recognition and identification

There are several features associated with computer vision applications due to their roles in image data collection and processing features. The interpretation of the words “Computer Vision” – similar to any other words – is based on the context of utilisation within the sentence. Other authors give context to this terminology in relation to their experience. As a result, this discussion of the pixel indexing scheme has contributed to the methodological development for IBR mainly in the geometric view, and this method is widely accepted in the computer vision research community.

2.3.4 Image matching gauge

The spatial image datasets using IBR functions are important, since the pixel coordinates from two or more different intensity images correspond to the same point in the world data. The mathematical equations are developed to properly analyse the absolute as well as the square difference of the function computation. The Sum of Absolute Difference (SAD) and the Sum of Squared Difference (SSD) are classified as popular computationally inexpensive image regions for matching the measured images [51]. These methods are not entirely justified in terms of image feature matching, and further to that, Gaussian noise is still an unsolved phenomenon as outlined by Bhat and Nayar [52] referring to equations 2.1 and 2.2. Equations

2.1 and 2.2 depicts the Sum of Squared Difference and Sum of Absolute Difference which are the equations to be used in this research study.

$$SAD = \sum_{(i,j) \in U} |I_1(x + i, y + j) - I_2(x + d_x + i, y + d_y + j)| \quad (2.1)$$

$$SSD = \sum_{(i,j) \in U} (I_1(x + i, y + j) - I_2(x + d_x + i, y + d_y + j))^2 \quad (2.2)$$

Where: I_1, I_2, x, y denotes compatible regions

: x_1, d_x, y, d_y denotes the local coordinates of the image

The IBR method comprises of several functions such as plenoptic, light field, and other concentric mosaic functions. These functions are associated with geometric rendering. The plenoptic function is representative of the intensity of the light ray passing through the camera centre at a 3D spatial location mostly in the (x, y, z) algorithm.

2.4 Different construction methods

In this section, construction and rendering are used interchangeably.

In determining the best construction method, the comparison analysis of different construction methods needs to be outlined. In this section, the rendering construction methods are discussed also citing their advantages over each other.

2.4.1 Construction standard measurement

IBR techniques are currently receiving much attention due to their powerful alternative to traditional geometric-based techniques that are outlined in this subsection for image synthesis.

There are several types of construction matrix namely geometric, matrix, and no geometric construction. In a rendering process, it is very important to note that the vector images form part of the pre-processing process and also contribute to the ability of the model to resolve the visibility issue. The acceleration of this process relies much on the axis-aligned bounding boxes

for each fragment that is computed [53]. As a result of this approach, a more detailed and graphical representation of this approach will be elaborated on in detail in the design chapter, as the same approach is going to be followed in this research study. Subsequent to these approaches, rendering displacement in maps requires the surface to be adaptively re-tessellated [54, pp. 171-180]. The significant difference between these methods is outlined in Figure 2.3. A high-level sample of these techniques are also indicated as follows:

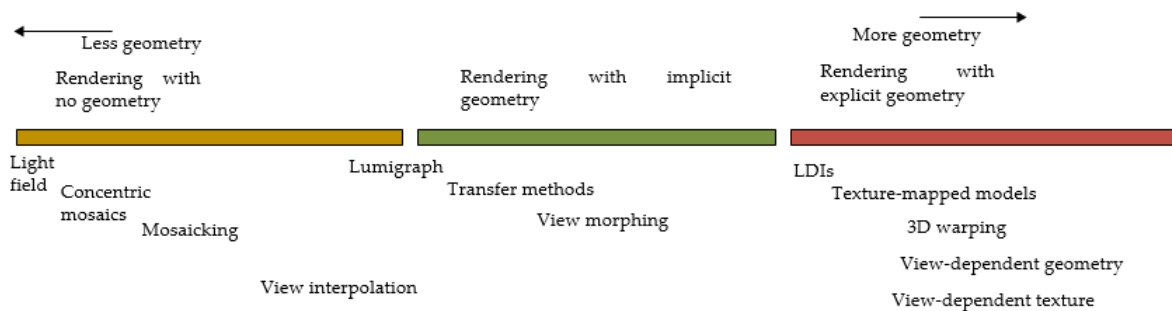


Figure 2.3: Different rendering techniques

Figure 2.3 depicts the different and most common rendering techniques. Figure 2.3 presents three geometry functions namely less geometry, rendering with implicit geometry and more geometry. The less geometry presents a function of no geometric rendering, which is an expensive method due to data acquisition and storage requirements. More geometry presents a function of explicit geometry and rendering with implicit geometry presenting the view morphing. Rendering with no geometry comprises of light field and concentration mosaics which are functions of the view interpolation. Additionally, the rendering with implicit geometry constitutes the transfer methods, while more geometry includes the 3D warping and view-dependent geometry.

Geometric construction relates to both implicit and explicit construction [55]. There are several construction methods that are associated with image rendering techniques. Explicit geometric rendering relies on the use of the approximate geometric view of the Lumigraph rendering. This is as a result of the view dependency, which means that the explicit geometric rendering much relies on the known approximate environment [56]. The use of the explicit rendering becomes much more difficult in an informal environment or settlement due to the tiring exercise of data

collection, which is skewed since residents can build on the road and mountains. This is deemed as one of the main concerns due to the need for a frequently updatable data collection model that needs to be implemented and monitored.

2.5 3D Panoramic image generation

Multiview image processing has been receiving increased attention lately with the advent of interactive navigation applications and immersive communication [57]. In addition, the use of computer-based models is growing exponentially in support of urban planning and management. The evolution of such a solution or innovation is mainly based on the increased data resources, multiple spatial datasets, and tools for processing and computation [58], [59]. Subsequently, the panorama creation of the 360° panorama is limited to the horizontal axis, hence allowing the virtual spin around the axis [60, pp. 261-271]. However, this study focuses on the use of multiple cameras rather than the use of a single 360° omnidirectional capable camera with its shortcomings. The word omnidirectional refers to the ability to sense in all directions (a full 360° view), and is part of the shortcomings identified in such a camera. The feasibility for a new model utilising hexagon camera configuration has to be exploited. Borg [61] depicted that achieving a full omnidirectional vision is possible in practice since the sensors themselves (despite how small they might be) must hide part of the view.

The intrinsic movement parallax and single effective viewpoint are associated with a panoramic view of a scene, bearing in mind that the panorama is from two different panoramic views. In a panoramic image generation, a new phenomenon of image stitching surfaces. Image stitching (mosaicking) is denoted as a process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image [62, p. 1]. Image stitching can be grouped into two approaches, direct and indirect feature-based techniques. Direct technique constitutes the dependent comparison of pixel intensities of the images and decreases the total differences among overlapping pixels. In contrast, the feature technique can recognise panoramas by utilising an automatic discovery of associations among the undistorted images [63]. Furthermore, intrinsic and extrinsic parameters are deemed crucial in obtaining camera calibration and pose estimation for the selection of an appropriate technique [64].

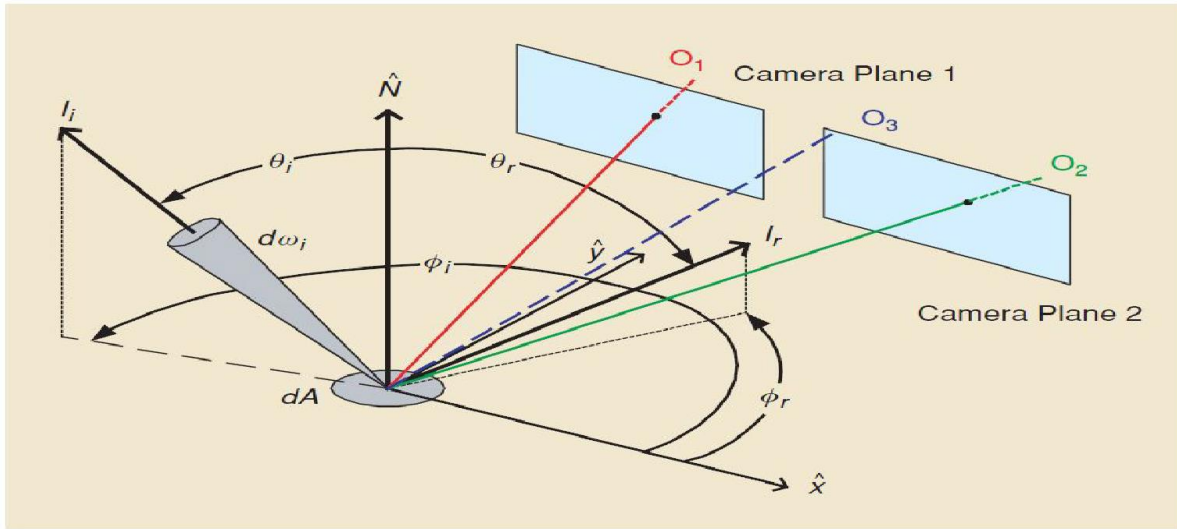


Figure 2.4: Geometric view of a GSV [62].

Figure 2.4 outlines the geometric view of Google Street View where the camera pose is highlighted for taking multiple image shoots. This approach makes use of multiple cameras and the image capture takes place along the 180° plane. The camera focuses and the angular distance for proper or overhead image stitching must be correctly aligned.

Figure 2.4 depicts camera planes (camera plane 1 and camera plane 2) for a geometric architecture that is almost identical in theoretical overview than the proposed research model. Each camera in the context of this research study is placed at 60° Ø angle between each other with the camera focal point at position dA. As a result, this gives the holistic view of the 180° plane in the context of Figure 2.4, while in the conducted research study, this approach will present a full 360° view utilising the Hexagon Camera Configuration Model.

2.5.1 Construction of panoramic image

Several methods are used in the construction of panoramic images. The two most common and easy-to-construct methods are cylindrical and spherical panoramas. Shum [65] outlined the cylindrical panorama as the easier one of the two methods. The cylindrical panorama is constructed in a 2D model to a 3D model which makes it an easier approach, as both the focal point and the field of view are known. The cylindrical panoramas (CP) are mostly mapped in

the following manner CP = (X, Y, Z) into a 2D model ($\emptyset$, v) using both formulae 2.3 and 2.4, and CP is an abbreviation for cylindrical panorama.

$$\emptyset = \tan^{-1} \left(\frac{X}{Z} \right) \quad (2.3)$$

$$v = \tan^{-1} \left(\frac{Y}{\sqrt{X^2 + Z^2}} \right) \quad (2.4)$$

Where:

$\emptyset$ denotes the panning angle

v denotes the scanline

The GSV panoramas will be requested using image indexing and WiFi as compared to inputting coordinates through the Application Programming Interface (API). Subsequently, Li and Ratti [66] utilised the same request methodology of ID and date information which resonates with the fundamental study aims of indirect improvement of this system by using other parameters apart from only the date and ID tag. A more detailed methodology structure will be outlined in Chapter 3.

2.5.2 Google Street View application interface

The GSV model has a distinctive quality due to factors such as a possible API interface to the existing platform. The image database of Google Street View is a network of adjacent 360° high-resolution panorama images, which are divided into quadratic tiles [67].

The GSV's API interface from google comprises the four (4) distinct APIs that are freely available for download. The API interface is outlined as follows:

- Thumbnail from GPS coordinates – use GPS coordinates (latitude and longitude), requires the street view image not to be close to the available position (...*output=thumbnail*&*w*=[SX]&*h*=[SY]&*ll* = [LAT, LNG]).
- XML – is based on the request and retrieval of GPS coordinates from a panoramic image ...*output=xml*&*ll* = [LAT, LNG], yet this approach differs from one database directory to another.

- Tile - given the panorama ID, the API can be used to access the panorama image using this ID in the request.

(...output = tile&panoid=[ID] & zoom=[z] &x=[X] &y = [Y]

- Thumbnail from ID

(...output=thumbnail&w=[SX] &h=[SY] &panoid=[ID]

2.6 Summary

Google Street View is an integral part of image processing with different kinds of data acquisition models such as remote sensing, which models are adopted methods that are mostly utilised for urban planning. Google Street View has the advantage of covering large areas in both rural and urban areas as opposed to other technologies. It is noted that IBR sampling is crucial in a multi-dimensional signal processing problem. This chapter has further outlined the work done thus far in the field of image processing, computer vision, and data requisition using Google Street View and Image-Based Rendering technology. Several methods have been outlined citing both the advantages and shortcomings of these models and reasons for the selection of the IBR techniques as the best approach to achieve the study objectives. It is seen through the literature that there is a piece of strong feasibility evidence that the study in hand can be conducted and further improved utilising IBR and other techniques. Despite the feasibility for utilising other methods for such a research study, the IBR technique is still deemed the most appropriate model in terms of data processing with fewer resources, and its efficiency is deemed optimal.

Chapter 3: Development of the image capture model and the use of IBR technique to render the captured images

This chapter presents the design and modelling techniques for capturing images utilising the hexagon image capture model. This model converts the images into a set of image datasets which are then calibrated utilising Blender3D software. The IBR technique is then called by focusing the calibrated images, resizing and aligning the camera number with the datasets, and then applying texture onto each image before the model can be simulated. This dataset manipulation is then optimised by calibrating individual images in the datasets to obtain noise-free images that are ready for simulation and rendering.

3.1 Introduction

The IBR modelling techniques are newly observed with high attention as one of the powerful alternative rendering techniques as opposed to geometric-based techniques. As opposed to the geometric-based technique, the IBR model makes use of a small number of images that generate the feature correspondence. In addition, the scene depth is important for accomplishing different tasks such as 3D modelling. Subsequently, the image capture proposition obtained by a conventional camera contains blurred scenes that are out of focus that are commonly known as Circle of Confusion (COC) [68, p. 1].

The research study test environment was conducted at Botshabelo (Place of Refugee), which is a small town located 45KM outside Bloemfontein (the capital city of the Free State province in South Africa). The reason for the selection of the test place was the prerogative of the researcher, and the technical factors taken by the research will be highlighted later in the chapter. Moreover, the test site brought about changes to the initial camera configuration model. The reason is due to the fact that Botshabelo township does not have high buildings, and as a result the configuration had to be altered from fisheye model with 9 cameras to the Hexagon Camera Configuration Model. The Hexagon Camera Configuration Model was then chosen as the prerogative of the researcher at a 60° angle between the cameras, and this selection does not affect the study objective in any way.

3.2 System apparatus selection

Before the system setup model is developed, a series of apparatus and system model configurations had to be identified. As a result, the hexagon-based cameras pose model was selected due to the reasons outlined in section 1.3. Following the successful system architectural model development of the hexagon configuration model, the apparatus were selected based on the power output for both the inverters and batteries, while for the cameras, the selection was based on the camera quality and pixels size.

The components outlined in Figure 3.1 depict the primary components or tools, namely the cameras and laptop for processing as well as the USB hubs and the power supply to be used for powering the cameras in this research study. Nonetheless, this does not in anyway discredit the importance of both the GPS module and the Arduino microcontroller. It is important to note that the architectural system model development was constructed before any physical development process could take place to visualise and direct the components selected for the physical model. This system modelling task needed to take place before the system components can be acquired/bought and the functionality test can be conducted either in a stationary or dynamic environment. The system design model architecture is based on the initial model designed on Microsoft® Visio Professional 2016, as indicated in Figure 3.1.

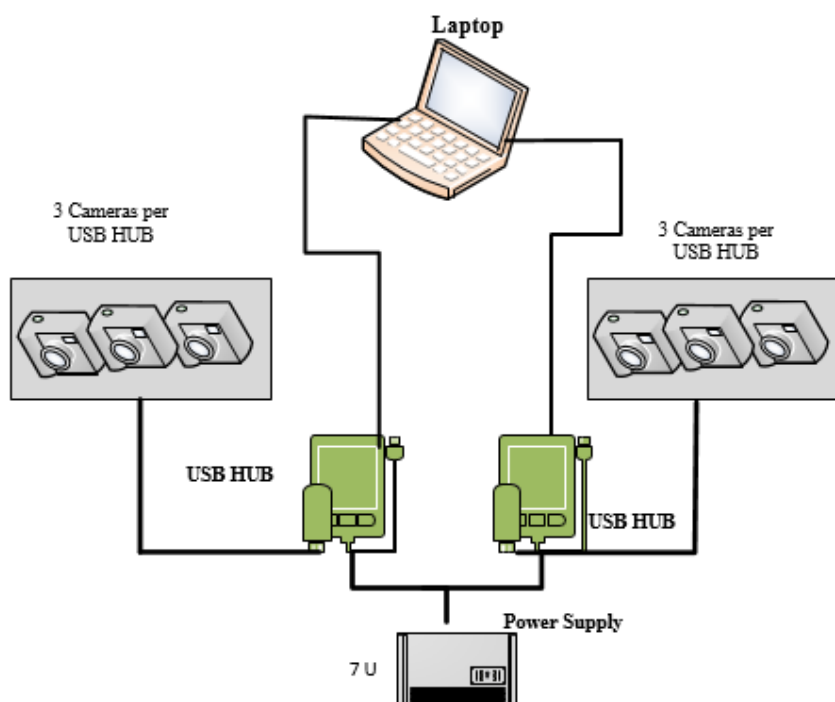


Figure 3.1: System architectural model

Figure 3.1 depicts the Hexagon Camera Configuration Model in an architectural view. The model consists of the following components:

- 6 HD cameras
- Laptop
- 2 USB HUBs
- Power Supply

These components are based on the initial model configuration of utilising six cameras at 60° distance apart to capture the entire 360° view.

3.2.1 HD cameras

Figure 3.1 highlights the components selected for this research study in the context of how the components will be placed and utilised. The full camera holder will hold the cameras that are mounted on the test vehicle rooftop, while the inverter converts the Direct Current (DC) to an Alternating Current (AC) and provides stability in the charging of the cameras during the testing phase. The six Full HD Action cameras were selected and utilised to capture quality images that do not require any customisation, which might delete some important data from the image such as image descriptor/GPS coordinates. Blender3D was selected and utilised due to its ability to convert and remodel files other than the .bli files, and also because it has the capabilities for code reuse. The detailed description of how the components are mounted and used on the live site will be addressed at a later stage. Furthermore, these cameras were utilised with the specifications with reference to Table 3.1 [69]:



Figure 3.2: Go Xtreme Rebel Full HD camera

Figure 3.2 depicts the Go Xtreme Rebel Full HD camera. This camera was selected based on its specifications outlined in Table 3.1.

Table 3.1: Go Xtreme Rebel Full HD camera specifications

Technical Specifications	Specifications
- Resolution	- 1080p @30fps 720p@30fps
- Effective pixels	- 16MP*, 12MP*, 10MP*, 8MP*, 5MP*, 3MP*, 2MP*
- Sensor type	- 2"/5cm display
- USB type	- Interface: USB 2.0
- WIFI	- Yes
- Durability	- Waterproof case upto 30m
- Battery	- 900mAh lithium battery
- Product length	- 6.000000
- Product width	- 3.000000
- Product height	- 4.000000

The camera model is selected based on its specifications that are outlined in Table 3.1, that demonstrates the feasibility of achieving the study objectives. The HD camera in Figure 3.2 will be used for image acquisition and the acquired data images will be converted into an image dataset that will be used to render the captured scene.

3.2.2 Acer laptop

In this research study, an Acer Aspire ES14 laptop with the following specifications outlined in Table 3.2 is selected.

Table 3.2: Acer Aspire ES14 laptop specifications

Technical specification	Old specifications	Modified specification
- Display size	- 14.00 inch	- 14.00 inch
- Display resolutions	- 1366X768 pixels	- 1366X768 pixels
- Processor	- Intel core 5	- Pentium quad core
- RAM	- 2GB	- 4GB
- Operating system	- Windows 10 Pro	- Linux
- Hard disk	- 500 GB	- 5TB
- Weight	- 2.40 kg	- 2.40 kg

Table 3.2 outlines the laptop specifications. Due to data processing and large datasets, the laptop was modified by adding a new processor, and RAM as well as increasing the hard disk space for better data processing power and data storage, as indicated in Table 3.2. Furthermore, it is theoretically proven that rendering requires much hard disk space, hence the hard disk upgrade.

3.2.3 USB hubs and power supply

Due to the camera specifications outlined in Table 3.1, an additional external battery worth 1200mAh was purchased and used to power the two hubs. This extra battery acts as an additional power supply to the computer due to the high amount of power needed to keep the system active. It is important to keep the system active without interruptions as this will negatively affect the data collection, mainly the coordinates and timestamp. In such a study that mainly focuses on accurate and timely data capture processes, any distortion that might arise will affect the coordinates and timestamp which are crucial in the final rendered output.

Furthermore, the system test duration was calculated as indicated in equation (3.1) as follows:

$$\begin{aligned}
 B_d &= \frac{C_c * C_{bv} * N_{bs}}{L_c} \\
 &= \frac{0.9Ah * 3.8v * 3}{11.4w} \\
 &= 0.9hr \\
 &= \underline{1 \text{ hr testing}}
 \end{aligned}
 \tag{3.1}$$

Where:

C_c = camera battery capacity

B_d = battery duration

C_{bv} = camera battery voltage capacity

N_{bs} = number of batteries connected in series

L_c = load connected in watts

Note that both the series and parallel connection in the context of equation (3.1) does not make a significant change in the calculation in terms of the power dissipation. Furthermore, equation (3.1) represents a single hub, therefore each hub power supply can last for at least one hour.

3.3 The system's software design process

The system's software design and development is based on the use of Blender3D software as outlined earlier. Blender3D is an open-source 3D creation suite. It also supports the entirety of the 3D pipeline modelling, rigging, animation, simulation, rendering, compositing, motion tracking, video editing, and 2D animation pipeline [70]. The reason for the selection of this software is the ability to convert and remodel files other than the .bli files and to re-sue codes. Additionally, the use of this software is deemed useful for this research study as it focuses on the simulation and rendering components of the suite.

Figure 3.3 depicts the connected system in a stationary position for testing.

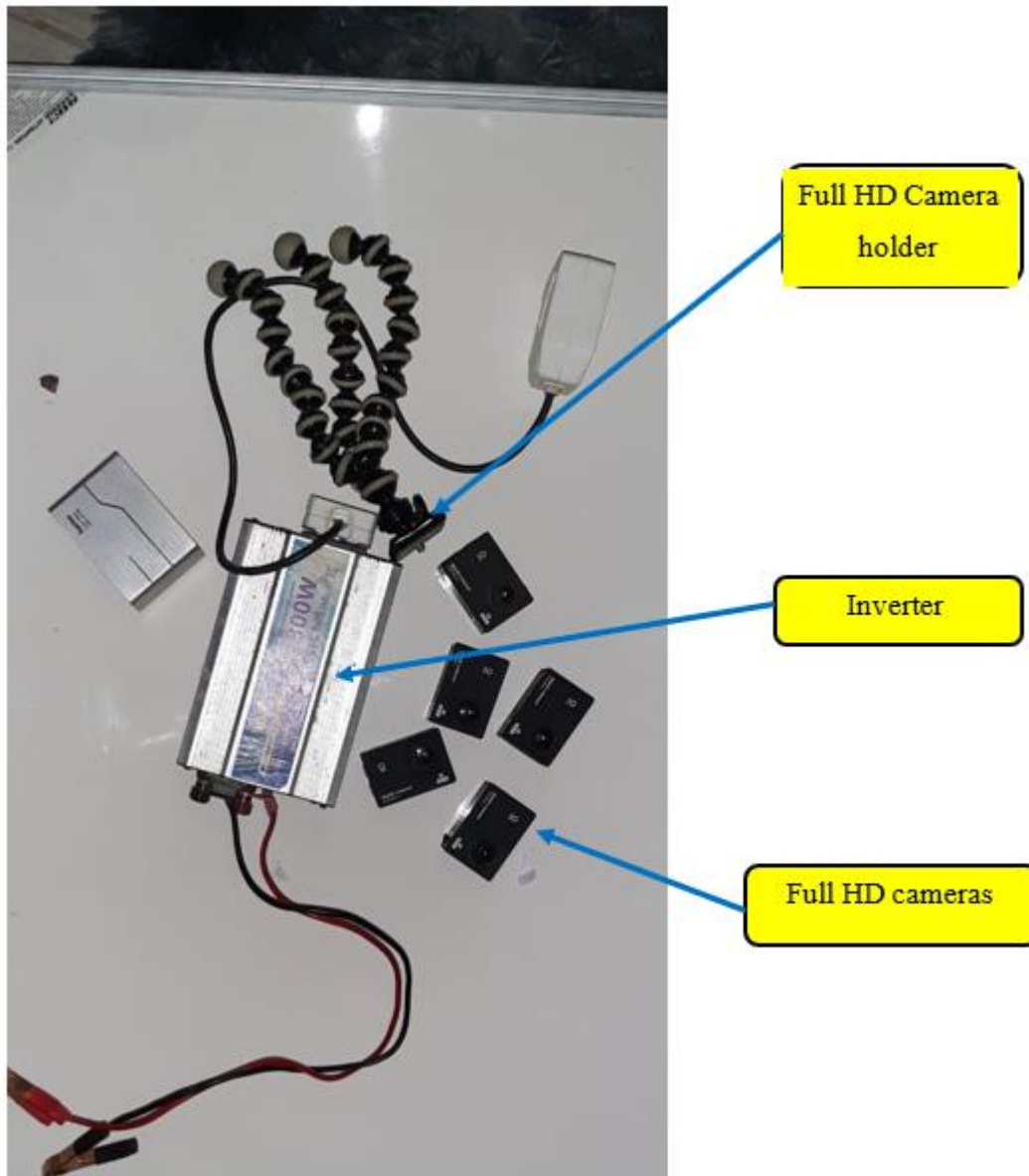


Figure 3.3: System components configuration setup in a stationary position

Figure 3.3 depicts the Hexagon Camera Configuration Model's physical system components in a stationary position. The components are connected without programming involved to test the feasibility of this physical model before it is mounted on the vehicle rooftop. Upon successful mounting of the stationary model, the system was placed on top of the vehicle. The same six-camera model was mounted on top of the vehicle and hooked at the edges of the vehicle roof so that the system is firm and rigid and does not fall during the testing phase. Once this system was mounted, a quick test of checking the firmness of the mounted system was conducted with the vehicle traveling at 80km/h on the test route.

3.4 System development process

The current spatial urban models have unique requirements in terms of data for parameterisation such as data on the urban extension. Remote sensing products are widely used to provide such datasets and have the capability to improve existing datasets [71, pp. 369-399]. The hexagon configuration model was constructed from a 2D model configuration as indicated in Figure 1.4 of Chapter 1. This was accomplished through the addition of the z-coordinate. This model was later transposed into a 3D model. The current technological capabilities enable the use of an omnidirectional camera to perform this activity, but due to the reasons already highlighted in Chapter 1 and Chapter 2 to answer the research question, a different model and approach had to be taken. The other reason for not utilising the omnidirectional camera is that the rendering construction specifically for this research study requires individual camera feeds as opposed to a single 360° feed. The use of the 360° omnidirectional camera will mean that the system only gets one set of feeds, which will not meet the study objectives. However, during the construction and development assessment it was noted that despite the researcher's prerogative, the following shortcomings regarding the testing site had to be considered:

- the applicable time with traveling from Botshabelo to Bloemfontein to conduct testing and data collection;
- traffic jams in the capital city (Bloemfontein); and
- high possibility of noise and skewed data due to high volumes of traffic.

After carefully considering, the assessment report outlined that it is critical to amend the scope of work and develop a Hexagon Camera Configuration Model that will act as an x-axis rotational 360° camera configuration at a 60° angle between each camera.

This process was then supported on the basis that Botshabelo township is a township close to the researcher's residential area, and it has less traffic and a low noise ratio (this has not been determined scientifically, but the conclusion was taken on those bases and also the township building infrastructure). The system development was then initiated based on the model layout as indicated in Figure 3.4.

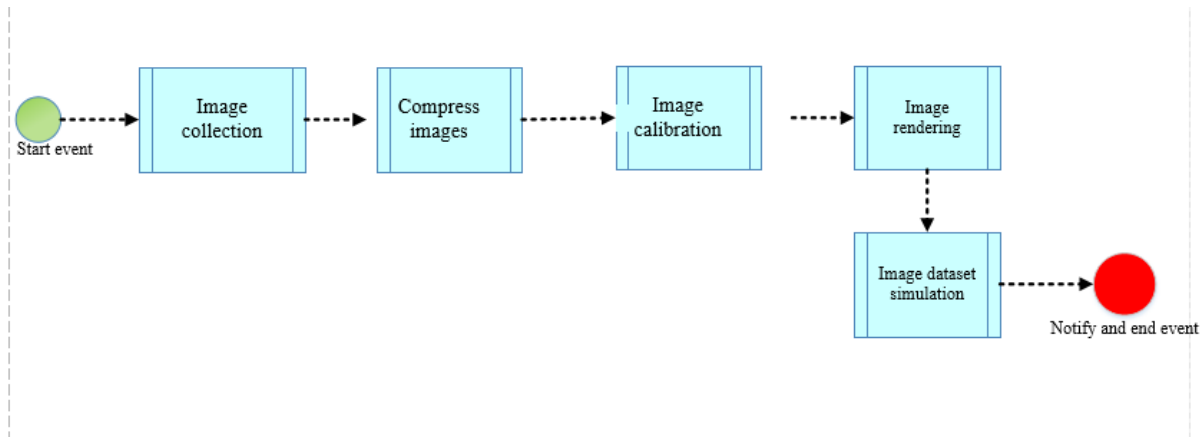


Figure 3.4: System model layout

Figure 3.4 depicts the system model layout in terms of the processes required before the system can be functional. The system commences by collecting the data utilising all six cameras in an asynchronous format with time intervals between each capture process. The system process model works as follows:

- image collection – the images are captured by six independent cameras with a 60° space between each camera;
- compress images – once the images are captured they are stored in a master folder and placed inside Blender3D software. Lossy algorithm is then applied to reduce the image dataset size; and
- image calibration – the dataset is then calibrated and rendered to produce an output simulation that permits for bidirectional movement within the simulator.

Due to the image dataset size, the images are compressed and then calibrated. After the images are calibrated, they are rendered inside the Blender3D software, and finally simulation takes place where the user can move forward and backward in the simulator. Figure 3.5 depicts the architectural model of the system when placed on the rooftop of the test vehicle.



Figure 3.5: Hexagon architectural camera configuration model

Figure 3.5 depicts the architectural model of the developed system model. The camera configuration model is placed on the vehicle rooftop. The system is then connected to the laptop that is seated in the car, and the GPS integrated module is connected to the Arduino microcontroller, which in turn is connected to the laptop for powering of both the GPS module and the Arduino microcontroller, as well as the simulation of the algorithm.

The study methodology is based on the flow diagramme and architectural models as indicated in figures 3.4 and 3.5. In these figures the images are captured concurrently with the six cameras and stored in different folders. Each camera has its own storage folder, for example “Camera 1 = folder 1”, and “Camera 2 = folder 2” up until the sixth camera. The image post-processing only includes the images that are selected based on their timestamp and GPS coordinates. The selected images are placed in one master folder to allow for accurate image processing and less processing power during the rendering and simulation of the image dataset. The selection of

images based on the timestamp and GPS coordinates was to reduce the number of images which would result in the computation taking longer.

The pinhole model which is an alternative model that is commonly known and used for non-sensor-based cameras [72] is utilised for this research study. The reason for opting to utilise this method is because the study does not entail or have any significance for a built-in sensory camera. This is due to data processing and image data manipulation that is concluded after the images are captured and datasets are created. Figure 3.6 depicts the pinhole model that is utilised in this research study.

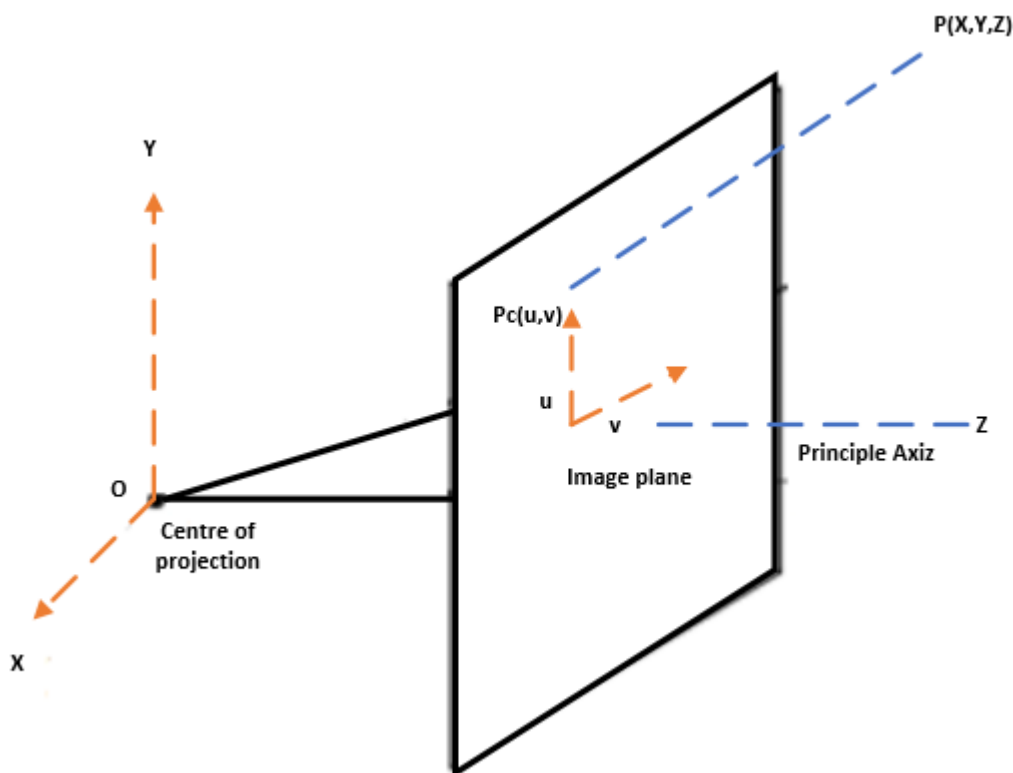


Figure 3.6: Pinhole camera model

The pinhole model is a model used for the image acquisition-based technique for sensorless cameras, and as a result it is based on the mathematical approach. Figure 3.6 depicts the camera centre projection for the pinhole as well as the principal axis.

The camera's centre of projection is defined by a character O and also the principal axis that is parallel to the Z -axis. The image plane is at the focal length f which is away from the centre of projection in the camera lens.

The P variables X, Y, and Z outline the 3D components of the camera lens.

The camera's image plane coordinates as indicated in Figure 3.6, which highlights the camera calibration matrix. The same calibration matrix of 3 X 3 is applied for mapping the 3D P to 2D Pc.

Equation 3.2 is used to locate the 3D coordinates using a similar triangle as depicted in Figure 3.6:

$$\frac{f}{Z} = \frac{u}{X} = \frac{v}{Y} \quad (3.2)$$

Where:

f = focal point

u, v = camera image plane coordinates

Equation 3.2 can further be simplified as follows to construct equations 3.3 and 3.4:

$$u = \frac{fX}{Z} \quad (3.3)$$

$$v = \frac{fY}{Z} \quad (3.4)$$

Alternatively, these equations can be addressed utilising homogeneous coordinates to match or formulate the calibration to a 3 X 3 matrix as follows:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (3.5)$$

The implementation of these equations resonates with the chessboard algorithm. The introduction of these equations are based on the pinhole model. As the camera capture is executed, the pattern matching algorithm makes use of these calculations to address the homogeneous coordinates. This is important, as the image datasets are supposed to be calibrated; hence, the emphasis of these calculations in the introductory part of the system development subsection.



Figure 3.7: Hexagon camera configuration setup

Figure 3.7 depicts the hexagon camera model attached to the test vehicle. Furthermore, Figure 3.7 outlines the model connection also displaying the rigid camera holder with the configuration model enlarged for a better view. Before system testing can commence, the second stationary test has to be conducted to ensure the full and correct functionality of the apparatus. Following the secondary stationary test results, the test vehicle speed together with the frame speed are calculated to obtain the best test speed for this research test. Equation 3.6 depicts the test vehicle and camera frame speed for the selection of the best test speed to be utilised for this research study. The method used for this application is derived from the video image equation [73]. Since the camera configuration model is based on a fixed model status, the equation is firstly obtained by calculating the vehicle speed as follows:

$$v = \frac{\Delta p}{\Delta t} \quad (3.6)$$

Where:

v = instantaneous velocity

Δp = displacement vector

Δt = time interval

In addition to estimating the vehicle speed, the instantaneous velocity vector is computed by adding the n points to the equation as follows to obtain speed at each n^{th} position with reference to camera frames:

$$v_i(t) = \frac{i}{n} \sum_{i=1}^n v_i(t) \quad (3.7)$$

Additionally, the camera calibration matrix is important for determining the intrinsic parameters of the camera imaging process. As a result, the conversion between the raw image plane and the retinal plane occurs in this process.

3.4.1 Image capturing and collection

The image capture section consists of the following apparatus: camera, images, and GPS module. The six-mounted camera configuration model captures images while the vehicle is in motion, and each image is treated as a frame. During the image capturing process each camera image dataset is stored in its specific folder as outlined earlier. Post this action process, image compression activity commences.

3.4.2 Image compression

The scene depiction utilising multiple depth images in a dataset format is compressed. The image samples are captured and obtained from the camera capture by delaying the camera switching algorithm between multiple cameras by 3ms. The camera switching duration was selected to allow for proper transition between cameras taking into account the number and size of the images. The 3ms switching duration was the prerogative of the researcher based on the IBR algorithm that does not require a large dataset for rendering. Image compression is utilised due to its advantages that correlate with the advantages outlined in [74].

The switching algorithm was processed from Arduino microcontroller as follows:

$$T = \frac{A}{S} \quad (3.8)$$

Where: T = time

S = speed (480MB/s) – USB (2.0 speed)

A = data transfer speed (1.5MB) raw data

$$T = \frac{1.5MB}{480MB/s} \quad (3.9)$$

$$T = 0.003125s$$

$$\underline{T = 3.125ms}$$

In this research study lossy and lossless compression are investigated and the best algorithm is selected and implemented. As a result, the two algorithms were tested and the best algorithm with the best capabilities for achieving the study objectives was chosen, which in this case was the lossy compression. The use of a lossy image compression algorithm was based on the ability to compress images by removing redundancy, however, the process is reversible as outlined in ref [75, 76]. Lossless image compression algorithm performs redundant processing on image information according to the human principle that the human eye is insensitive to certain visual features.

The performance view of the compression algorithms and the methods in which the two algorithms can be used are determined by making use of a lossy image compression algorithm. In this research study, lossy image compression algorithm is aimed at reducing the size of an image to a few kilobytes without compromising the image quality. The lossy image compressions algorithm utilised in this research study makes use of the .JPEG image formatting to effectively compare the image texture between different views. The sample .JPEG image formatting is indicated in Figure 3.8. Additionally, the image dataset is simulated without noise for better performance verification.

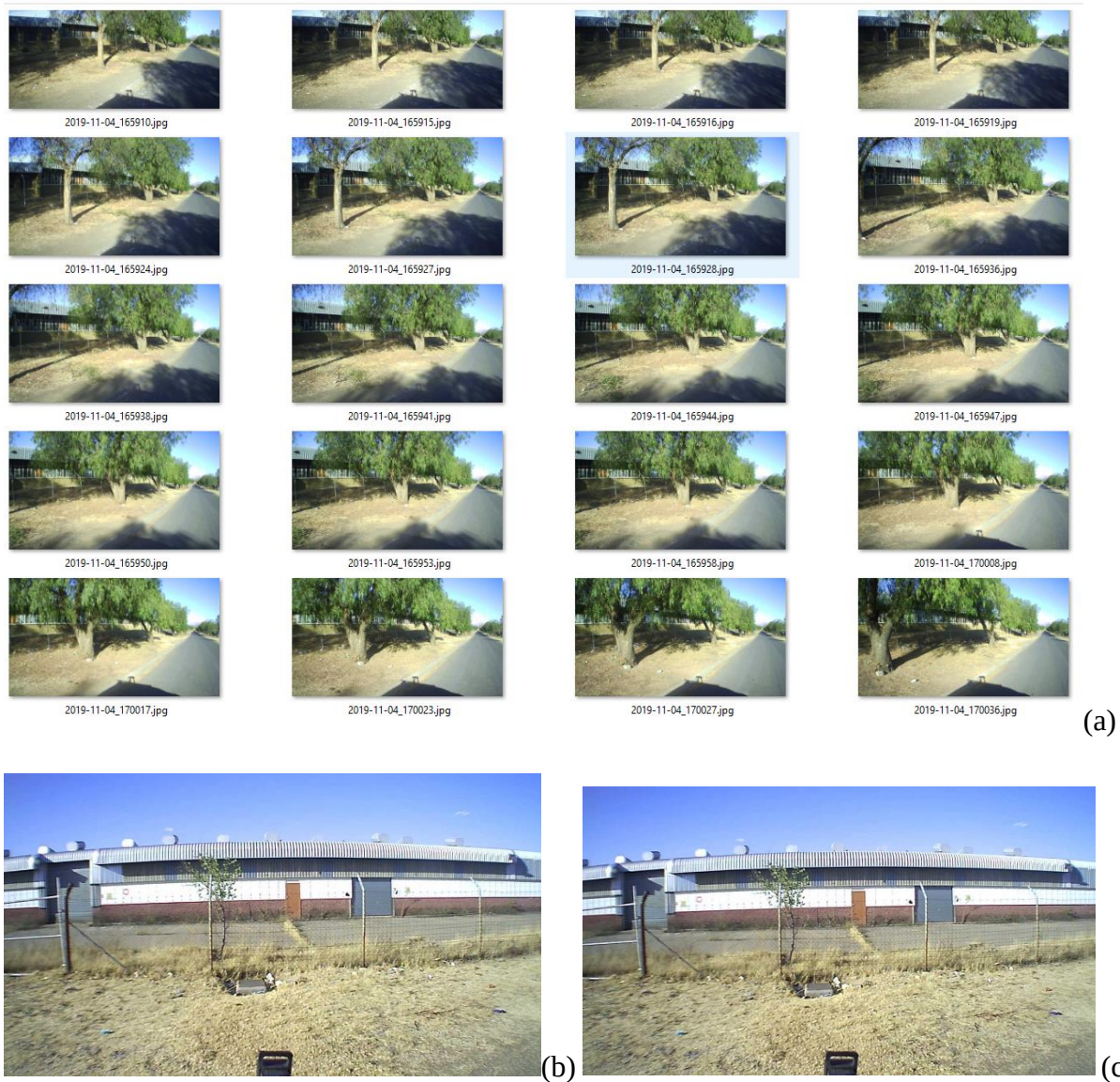


Figure 3.8: (a) Captured image dataset with timestamp and date; (b) Lossy image compression at 100% quality with file size 2.68MB; (c) Lossy image compression at 9% quality with file size 1.05MB

Figure 3.8 depicts the image types for the captured images and also their date and timestamp within the camera storage folder. The images are outlined in Annexure E. Before Lossy compression algorithm is applied to a .JPEG image file, the file size is huge - in this context it is 2.68MB. After applying the lossy image compression algorithm, the file size reduces and this algorithm does not affect the image or dataset quality. In this research study, after the lossy image compression algorithm was applied, the file size was reduced to 1.05MB and the image quality and data were not affected. As a result, the file was ready for rendering.

Additionally, the image compression framework used to obtain the results output is outlined in Figure 3.9.

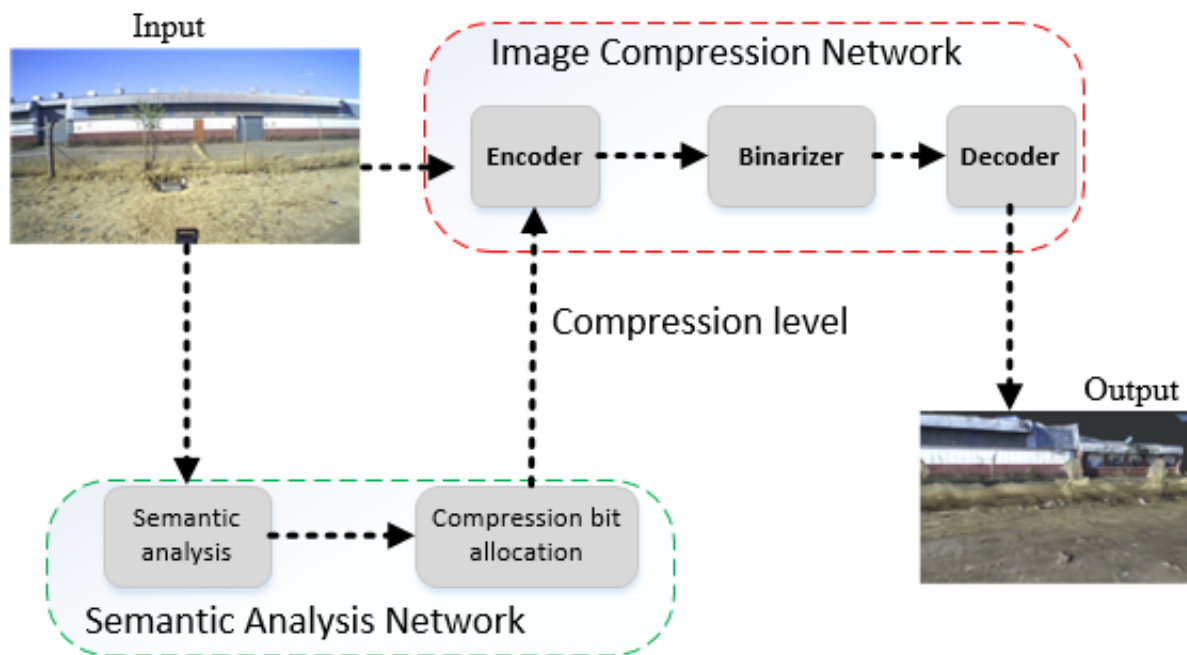


Figure 3.9: Compression structure

Figure 3.9 shows the compression structure that is utilised in this research study. The input image datasets are then used for the network training sample, by specifically setting the image dataset for image recognition where the compressed image datasets are compared against the raw image dataset. Additionally, the image compression bit allocation is then used to calculate the compression alterations and the alterations depend on the size of the dataset. In the context of this research study, each image captured was placed inside the master folder and converted into a full image dataset. The execution of these image datasets is accomplished by utilising the principles of compression structure, and the laptop is modified to increase the processing power and enable multithreading. Hence the image dataset compression is utilised, as opposed to the individual image compression, and this is also to retain of the dataset quality due to the use of the lossy compression algorithm. The reason for the laptop modification was that the compression execution was taking long and in some instances, the laptop was crashing and it was observed to be impractical to compress images individually. The individual image compression can be done for a smaller dataset, and it will be impractical in a bigger dataset.

Furthermore, Figure 3.9 comprises of input image, semantic analysis network, image compression network and output image. The input image depicts the original input image to the

semantic analysis network and the image compression network. From the input image the semantic analysis network is used to extract semantic regions of the input image, then calculate the compression level corresponding to each area. The image compression network is used to compress and decompress the image hierarchically.

3.4.3 Camera calibration

The data collection and image compression process are determined by the reduction in the image size whilst still keeping the image resolutions intact. The image calibration model utilises the pinhole camera model that introduces some image distortions. These image distortions that are seen in this process are classified as either radical or tangential image distortion. The code below depicts the sample image calibration code for the intrinsic matrix, which is the matrix utilised in this research study. The execution of this code is to detect the focal point of the intrinsic image dataset by locating the focal point of the image during the calibration process. Due to the online accessibilities of libraries, it was deemed not necessary to re-write certain libraries, but rather import and modify/customise them to meet the study requirements or output results. Following the successful library import into the working directory of the code sheet, the calibration path was assigned and the dataset is loaded and analysed.

The chessboard matrix corners are then detected through calibration of image corners and points and eventually, they are saved as different .xml files in the camera params directory. This is important in reducing the execution time of the files and also security around the corruption of any file.

```
#Importing and loading of libraries

import cv2
import numpy as np
import glob
from tqdm import tqdm
import PIL.ExifTags
import PIL.Image

chessboard_size = (9,6)
obj_points = []
img_points = []

objp = np.zeros((np.prod(chessboard_size),3),dtype=np.float32)
objp[:, :2] = np.mgrid[0:chessboard_size[0],
                      0:chessboard_size[1]].T.reshape(-1,2)
calibration_paths = glob.glob('calibration_images/Front_center/*')#Iterate
```

```

over images to find intrinsic matrix
for image_path in tqdm(calibration_paths):#Load image
    image = cv2.imread(image_path)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    print("Image loaded, Analyzing...")

    #find chessboard corners

    ret, corners = cv2.findChessboardCorners(gray_image, chessboard_size, None)
    if ret == True:
        print("Chessboard detected!")
        print(image_path)
        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
        cv2.cornerSubPix(gray_image, corners, (5,5), (-1,-1), criteria)
        obj_points.append(objp)
        img_points.append(corners)
        # Calibrate camera
    ret, K, dist, rvecs, tvecs = cv2.calibrateCamera(obj_points, img_points,
        gray_image.shape[::-1], None, None)
    # Save parameters into numpy file
    np.save("camera_params/ret", ret)
    np.save("camera_params/K", K)
    np.save("camera_params/dist", dist)
    np.save("camera_params/rvecs", rvecs)
    np.save("camera_params/tvecs", tvecs)
    #Get exif data in order to get focal length.
    exif_img = PIL.Image.open(calibration_paths[0])
    exif_data = {
        PIL.ExifTags.TAGS[k]:v
        for k, v in exif_img._getexif().items()
        if k in PIL.ExifTags.TAGS}
    focal_length_exif = exif_data['FocalLength']
    focal_length = focal_length_exif[0]/focal_length_exif[1]
    np.save("./camera_params/FocalLength", focal_length)

```

The radical image distortion was calculated as follows:

$$X_{distortion} = x(1 + k_1r^2 + k_2r^2 + k_3r^6) \quad (3.10)$$

$$Y_{distortion} = y(1 + k_1r^2 + k_2r^2 + k_3r^6) \quad (3.11)$$

Where: x = original x location on the imager

y = original y location on the imager

k = radical distortion coefficient

r = radical distortion form Taylor series

Additionally, the tangential image distortion was also tested. This image distortion test occurred mostly when the image lenses were not aligned perfectly and in parallel to the image plane. As

a result, some areas within the image appeared nearer than expected or than the actual area. These tests were validated using the following equations:

$$X_{distortion} = x + [2p_1xy + p_2(r^2 + 2r^2)] \quad (3.12)$$

$$Y_{distortion} = y + [p_2(r^2 + 2y^2) + 2p_2xy] \quad (3.13)$$

Where: x = original x location on the imager

y = original y location on the imager

p = tangential distortion coefficient

r = radical distortion form Taylor series

For the camera to be fully calibrated, five parameters known as distortion coefficients given using equation (3.14) needs to be known. Therefore, it is empirical to calculate this variable before the image calibration process can be performed.

$$D_c = k_1k_2p_1p_2k_3 \quad (3.14)$$

Where: k = radical distortion coefficient

p = tangential distortion coefficient

D_c = Distortion coefficients

Additionally, other camera information including but not limited to intrinsic and extrinsic camera properties are important during the camera calibration, and, in most cases, they are camera specific. Moreover, the information about the focal length (f_x, f_y) and optical centre (C_x, C_y) is critical. The focal length and optical centre are then used to create the camera matrix, which is used to remove the distortion due to the lenses which are camera specific. The camera matrix is then calculated utilising the focal length and the optical centres for the reused of images that are captured utilising a specific camera model as follows:

$$C_m = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.15)$$

Where: C_m = camera matrix

(C_x, C_y) = optical centres

(f_x, f_y) = focal length

The system setup approach makes use of the chess pattern for the camera calibration setup. Some calibration methods in the literature rely on three-dimensional objects through the tests conducted, and therefore the flat chessboard pattern approach is deemed appropriate for this research study. This is due to the method being less complex and easily understood even by non-technical individuals. Furthermore, the usage of altering the black-and-white squared pattern ensures that there is no bias towards either side in the measurement algorithms.

The main aim of the camera calibration is to determine the geometric parameters of the image formation process. This is crucial for many computer vision applications, especially when the metric information about the scene is required [78, pp. 517-531].

The camera calibration process is as follows:

- Capture 20 chessboard images from different poses – in the context of this research study, 60 or more images are needed for calibration.
- Find the chessboard corners.
- Find the intrinsic matrix, distortion coefficients, rotation vectors, and the translation vector.
- Store the parameters to the .xml file.

Following the process completion, OpenCV for the python library is utilised to compute the results from the .xml file. This, therefore, allows for the reuse of the code for multiple cameras, which is relevant for our study that utilises a Hexagon Camera Configuration Model with a rotational image capture technique.

In order to achieve this task, an OpenCV python code was programmed. The reason for this line of code was to take the stored images and enable a camera reading as outlined below:

```
ret, frame = cam.read()
```

Following this process, the image dataset was converted to Gray Scale as indicated below:

```
grey=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
```

An image calibration utilising a chessboard was utilised as a squared pattern match. The finding is outlined as indicated in Figure 3.10.



Figure 3.10: Black-and-white test match on a chessboard

Figure 3.10 depicts a black-and-white match-finding test on a chessboard utilising an intrinsic matrix for the reduction of image biasing. This model was deemed less complicated in matching the images utilising the Kd-tree approach. It was noted that the CasHash-based approach can have a faster computational time compared to the Kd-tree-based image matching, depending on the number and size of the datasets. The precision ratio becomes lower, hence the continuation of the Kd-tree-based image matching approach.

The first process to achieve the results in Figure 3.10 is as follows:

- Load the chessboard images into the model

```
calibration_paths = glob.glob('calibration_images/*')
```

- Convert the dataset to grayscale

```
image=cv2.imread(image_path)  
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

- Find image corners

```
ret,corners = cv2.findChessboardCorners(gray_image, chessboard_size, None)
```

Upon finding the image corners, the intrinsic matrix, distortion coefficient, rotation vectors, and the translation vector were found utilising the below line of code.

```
ret,K,dist,rvecs,tvecs=cv2.calibrateCamera(obj_points,img_points,  
gray_image.shape[:-1], None,None)
```

After this is achieved the .xml files are stored utilising the below code.

```
np.save("camera_params/ret",ret)  
np.save("camera_params/K",K)  
np.save("camera_params/dist",dist)  
np.save("camera_params/rvecs",rvecs)  
np.save("camera_params/tvecs", tvecs)
```

Figure 3.11 depicts the sneak preview of how the .xml files are stored in the directory.

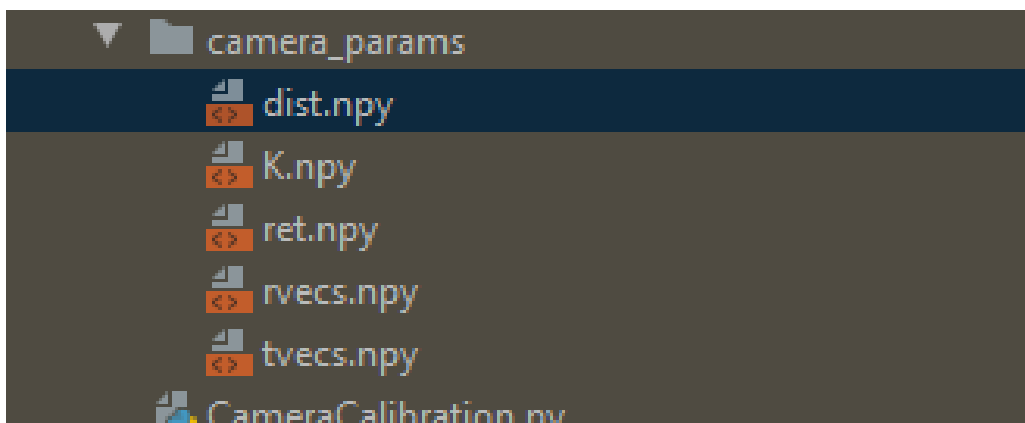


Figure 3.11: Preview of the .xml directory file.

Figure 3.11 depicts the camera parameters for the saved .xml files that are used for rendering after the compression process is complete.

3.4.4 Image rendering procedure

The image rendering technique in the context of this research study focuses on the known camera parameters and undistorted images for the rendering of the scenes. These images are reconstructed, and the texture is applied to their structure before rendering simulation can be executed.

The first step in reconstructing a 3D object from its original 2D digital image form is through its corresponding matching points. Correspondence matching points refer to those points across the images which projections are of the same 3D point of the object being imaged [79]. For correspondence matching, a set of distinctive feature points is detected in the image of the dataset. The feature points are searched for the correspondence across the rest of the images within the dataset. Additionally, the feature points are then detected in an image reducing the number of matched correspondence.

Furthermore, the exhaustive matching of all the image pixels against each other is deemed expensive, and as a result, the robust feature points are readily distinguishable and invariant to image transformations. Thus, the computation time of the correspondence matching is greatly reduced by detecting feature points in an image. SIFT algorithm is utilised for the detection of the feature point in an image as outlined below:

```
img=cv2.imread('2019-11-07_122716.jpg')
#converttogreyscale
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
sift=cv2.xfeatures2d.SIFT_create()
keypoints,descriptors=sift.detectAndCompute(img,None)
#drawthedetectedkeypoints
sift_image = cv2.drawKeypoints(gray, keypoints, img)
```

The outlined code depicts how the keypoints are computerised and drawn as indicated in Figure 3.12.

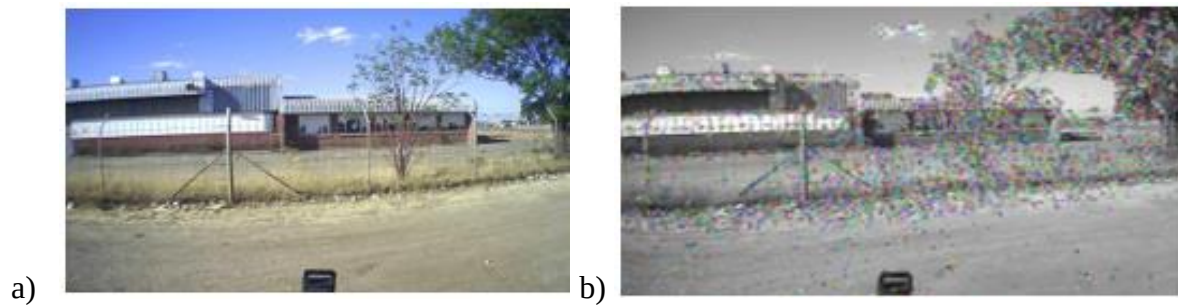


Figure 3.12: a) original captured image; b) application of SIFT algorithm for keypoints detection.

Figure 3.12 a) depicts the originally captured image, while Figure 3.12 b) depicts the calibrated image and the application of the SIFT algorithm for the detection of the keypoints in an image. The keypoints are highlighted in Figure 3.12 (b), and this enables ease of access to features when executing or applying the Structure from Motion Model within a dataset.

For the image dataset to be fully rendered and simulated, the following processes must take place namely:

- Structure from Motion
- Multi-view stereo
- Texturing

3.4.4.1 Structure from Motion

Structure from Motion is a technique that uses a series of two-dimensional images of a scene or object to reconstruct its 3D structure [80]. Furthermore, SFM permits for three-dimensional reconstruction starting from collection of images, hence the application and utilisation of SFM and its relevance to this study. For Structure from Motion, similar to any other algorithm, sequencing is very important mainly when processing a large amount of data. Structure from Motion follows the following steps:

- An image pair, having some overlapping portion of the scene/object between them, is selected from the image sequence.

- All the feature points from one image are inserted into the leaves of the k-d tree. The feature points belonging to the other image of the pair are used as queries to the first image.
- Then, the k-d tree algorithm is used for the efficient search of K-nearest neighbour of a feature point X. The threshold distance R is selected according to the image resolution.
- This search across the k-d tree leaves results in correspondence matching the feature points in the image pair.
- The matched points are then checked for accuracy using RANSAC-based estimation of the fundamental matrix. Those matched points which are not satisfying the fundamental matrix equation are rejected as mismatches.

The following code outlines how the features are detected in an image with the output depicted in Figure 3.13.

```
#loading of images
img1=cv2.imread('2019-11-07_122716.jpg')
img2=cv2.imread('2019-11-07_122706.jpg')

#convertimagestograyscale
img1=cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
img2=cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)

#createSIFTobject
sift=cv2.xfeatures2d.SIFT_create()

#detectSIFTfeaturesinbothimages
keypoints_1,descriptors_1=sift.detectAndCompute(img1,None)
keypoints_2,descriptors_2=sift.detectAndCompute(img2,None)
bf=cv2.BFMatcher(cv2.NORM_L1,crossCheck=True)

#matchdescriptorsofbothimages
matches=bf.match(descriptors_1,descriptors_2)
matches=sorted(matches,key=lambda x:x.distance)
```

```
#drawfirst50matches
```

```
matched_img = cv2.drawMatches(img1, keypoints_1, img2, keypoints_2, matches[:50],  
img2, flags=2)
```



Figure 3.13: Feature detection using SIFT feature extraction and brute force matching.

Figure 3.13 depicts the output obtained by applying the SIFT feature algorithm and the brute force matching algorithm to an image. This model still requires further refinement; hence the introduction of the point of cloud. These matched image points are then computed as a collection of data points by a given coordinate system as outlined in Figure 3.14.

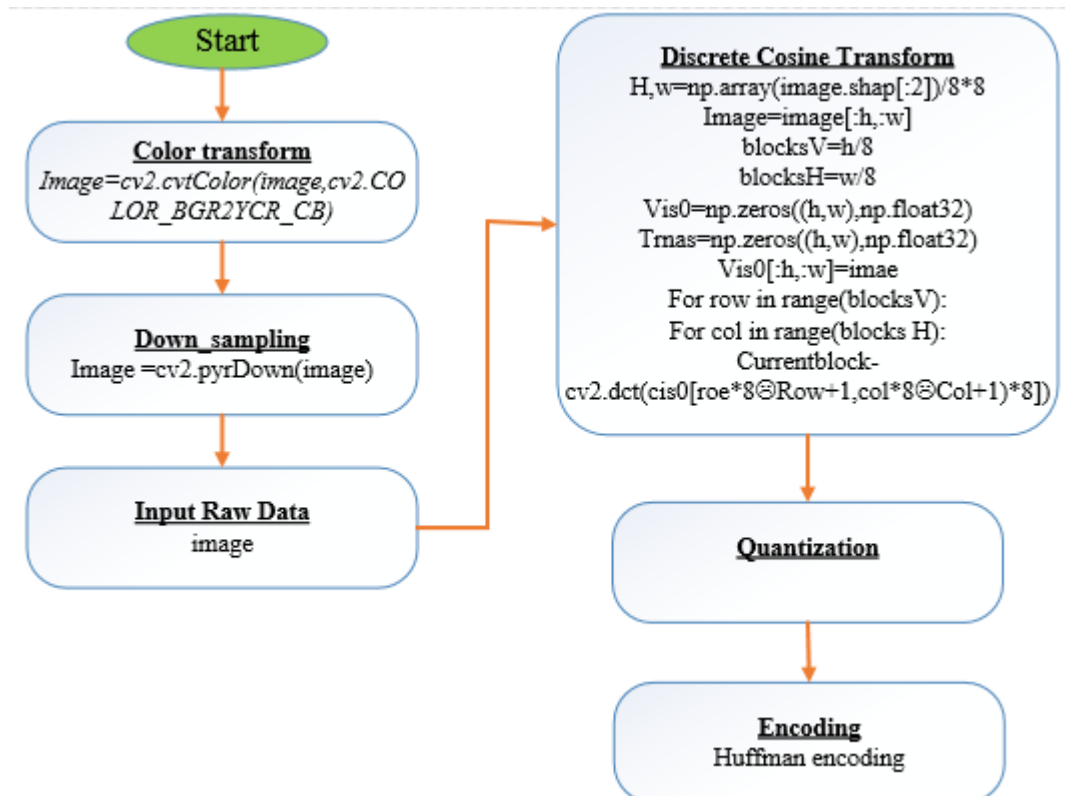


Figure 3.14: data points collection flow diagram

Figure 3.14 depicts the data points data collection. The image dataset is placed into the model, and the block matching objects are then created from the input dataset. The matching objects after execution creates the disparity map which is then computed to generate a new dataset height and weight. After this is completed, the focal length is loaded into the model, the 3D points are projected, and the model itself gets rid of value 0 points and masks the colour before the final output file is produced.

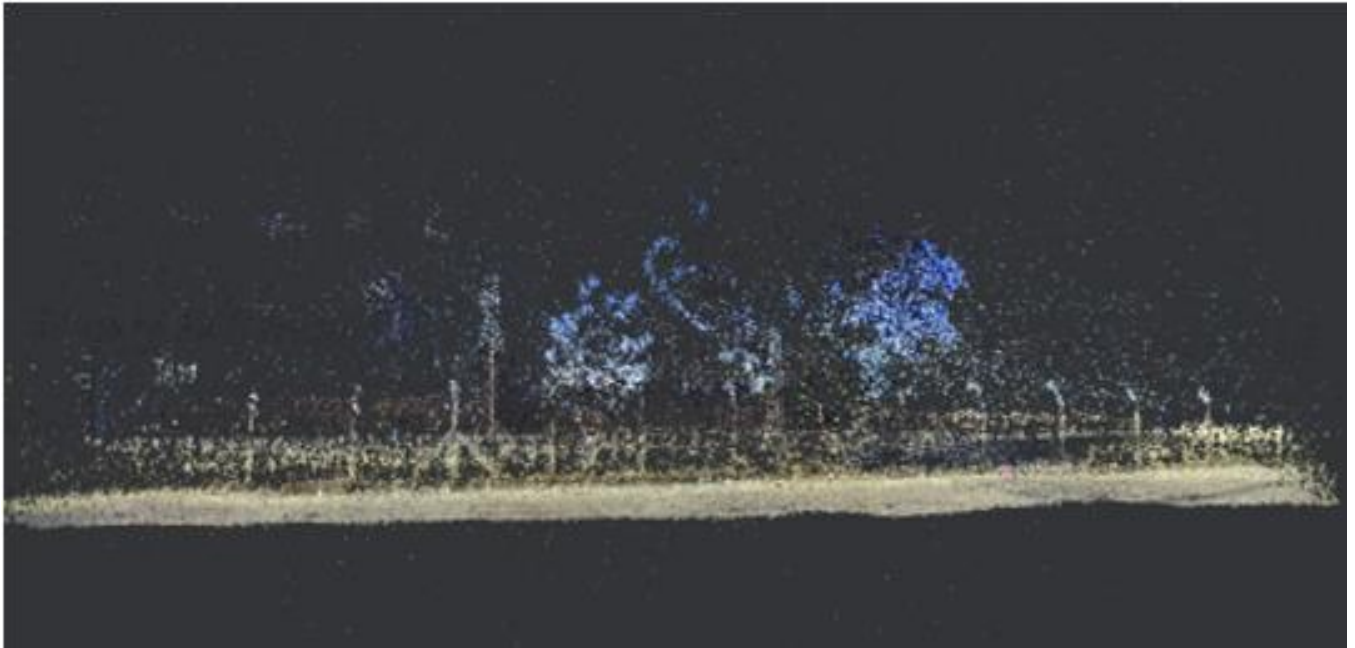


Figure 3.15: Application of a point of cloud in an image

Figure 3.15 depicts the application of a point of cloud in an image. To achieve the coloured point of cloud in this research study, the 2D camera images are extended over the 3D points so that the information is assigned to each 3D point; hence the view of different colours in Figure 3.15. Additionally, the extraction of the 3D point in the point of cloud was achieved by using the extrinsic calibration between the camera and the device. The below code depicts the key point computation code as follows:

```
win_size=5
min_disp=-1
max_disp=63
num_disp=max_disp-min_disp
stereo=cv2.StereoSGBM_create(minDisparity=min_disp,
    numDisparities=num_disp,
    blockSize=5,
    uniquenessRatio=5,
    speckleWindowSize=5,
    speckleRange=5,
    disp12MaxDiff=2,
```

```

P1=8*3*win_size**2,#8*3*win_size**2,
P2=32*3*win_size**2)

print("\nComputing the disparity map...")
disparity_map=stereo.compute(img_1_downsampled,img_2_downsampled)

print("\nGenerating the 3D map...")

h,w=img_2_downsampled.shape[:2]

#Load focal length.
focal_length=np.load('./camera_params/FocalLength.npy')

Q=np.float32([[1,0,0,-w/2.0],
              [0,-1,0,h/2.0],
              [0,0,0,-focal_length],
              [0,0,1,0]])

Q2=np.float32([[1,0,0,0],
              [0,-1,0,0],
              [0,0,focal_length*0.05,0], #Focal length multiplication obtained experimentally.
              [0,0,0,1]])

#Reproject points into 3D
points_3D=cv2.reprojectImageTo3D(disparity_map,Q2)
#Get color points
colors=cv2.cvtColor(img_1_downsampled,cv2.COLOR_BGR2RGB)

#Get rid of points with value 0 (i.e no depth)
mask_map=disparity_map>disparity_map.min()

#Mask colors and points.
output_points=points_3D[mask_map]
output_colors=colors[mask_map]

```

```
output_file='left.ply'
```

During the image matching process, the flag is passed using the match flag function of the FLANN algorithm. The image match function in the context of this research study was used with reference to the matrix outlined in Figure 3.6, and was computed after the track construction was obtained as follows:

```
#Draw keypoint ()  
  
detIMG=cv2.drawKeyPoints(gray,  
kP,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS_cv2.imwrite  
(‘sift_keypoints.jpg’,detIMG))
```

Under normal circumstances mainly utilising exhaustive matching, the computation is usually time-consuming and complex; hence the selection of the FLANN algorithm in this research study. The FLANN algorithm utilises a tree-based approach by storing the image datasets within efficient data structures and utilising the Kd-tree approach. The Kd-tree approach was selected based on the literature from [81], [82], highlighting the construction of the tracks for the matches.

In the track construction, the essential and fundamental matrices of a camera are computed. These matrices specify the camera motion in terms of the rotational and translational components. This function is called in OpenCV to find the fundamental camera matrix as indicated:

```
cv2.findFundamentalMat(self.match_pts1,self.match_pts2, cv2.FM_RANSAC, 0.1, 0.99).
```

For the essential matrix, the solve for the structure library is called as follows:

- Solving the Structure from Motion from 2D tracks

At this stage, triangulation is performed. Triangulation is defined as a process for determining the point in 3D space given its projection onto two or more images [83]. Direct linear transformation namely a P3P algorithm was used for the triangulation. In the context of this

research study, two sets of triangulation methods, namely linear and non-linear triangulation, were used. The reason for utilising triangulation is to find the intersections of two or more known rays in space. A pinhole model was utilised for the camera projection matrices utilising the following equations: :

$$S_1[u_1, v_1, 1]^T = P_1[X, Y, ZW]^T \quad (3.16)$$

$$S_2[u_2, v_2, 1]^T = P_2[X, Y, ZW]^T \quad (3.17)$$

Where: S_1, S_2 = two scalars

i^{th} = number of rows

This linear triangulation is not limited to two images. It can also be extended to the rows per dataset which can be obtained in a matrix format. Following this process, the non-linear triangulation was applied to minimise the measured error citing the optimisation model developed by Zhang [84, pp. 161-195]. Another model based on the Bundle Adjustment (BA) algorithm was also utilised by refining the Structure from Motion (SFM) model and applying it to the calibrated dataset.

At this stage, the Bundle Adjustment (BA) is performed. The problem relating to the BA is the simultaneous refining of the 3D coordinates describing the scene geometry. The parameters of the relative motion and the optical characteristics of the camera (s) are employed to attain the images. As a result, particularly for this research study, the CERES algorithm is utilised as well to mitigate any errors that might occur in conjunction with the non-linear triangulation.

3.4.4.2 Multi-view stereo

In this section, the camera parameters are captured, and the patch-based stereo and semi-global matching are used to generate point tracks, the depth-maps as well as the points cloud. Following the successful generation of these variables, a mesh of the scene is created as indicated in Figure 3.16. Finally, all the refined depth maps are merged to get the final reconstruction model. To obtain the Multiview stereo, the following sequence was followed:

- Few image frames with minimal camera motion are selected from the image sequence. By doing this, image redundancy is addressed, as image frames having large camera motion between them have less amount of overlapping object regions. So, adding two frames with large camera motion does not give many corresponding points.
- Dense stereo matching is ran in these camera frames to obtain a dense 3D point cloud of the overlapping object region. A real-time plane sweeping algorithm is used in each of the image frames during stereo matching. The basic idea behind this process is epipolar geometry. Two corresponding points in a stereo setup follow an epipolar constraint, resulting in the corresponding point in an image frame laying along the epipolar line. This constraint reduces the correspondence point search area from 2D plane across the entire image, to a one-dimensional epipolar line existing in the stereo image pair counterpart.
- A real-time plane-sweeping method is implemented by sweeping a plane through 3D space, across the image frames following the camera positions. Light rays from all the pixels of the images are projected into the respective imaging planes.
- The rays are back-projected towards the reconstructed object/scene at each intersection of a 3D point. These points come from all the points across the images. So, the intersection of these rays results in a dense collection of points in 3D space, which represent the object/scene being imaged. The obtained point cloud preserves the object geometry and forms a dense 3D point cloud as indicated in Figure 3.15.

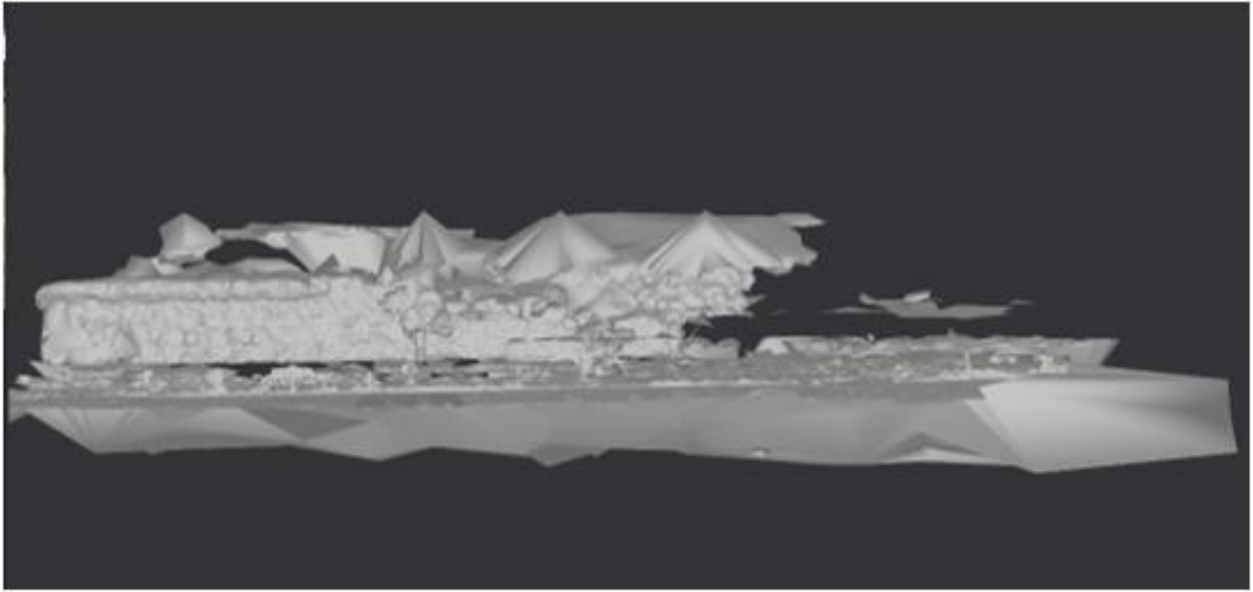


Figure 3.16: Mesh output from the Multiview stereo

The multi-view stereo algorithm is further used as a Semi Global Matching algorithm (SGM), where it consists of calculations, aggregated costs, disparity computation, and the extension for multi-baseline matching. Since the mesh output is obtained, the Multiview stereo is operated as a 3D model as seen in Figure 3.16.

3.4.4.3 Texturing

After all the patches are formed onto the faces of the model, the texture patches and colours are adjusted. This is achieved by adjusting colour between different adjacent patches. This results in a seamless texture being achieved across the entire model.

Figure 3.17 depicts the texture scene overview model.

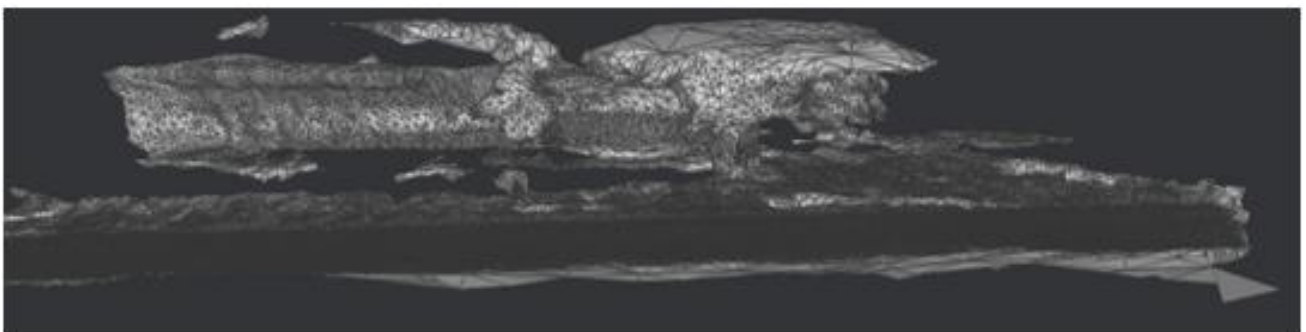


Figure 3.17: Textured scene overview

Figure 3.17 depicts the overview model of the textured scene after the multi-view is applied to the image dataset. Once the texture is applied to the dataset, the dataset can be simulated to provide the required output, which in this case is permitting bidirectional movement. Furthermore, the 3D textured model is obtained from each pixel of the datasets. The index of the disparity datasets serves as an index for both the left and right image datasets. This is pivotal as the textured model needs to be rendered for a panoramic view. If the model is only rendered, the full view of the model to enable 360° view that permits for back-and-forth movement cannot be realised, and as a result the study objectives could not be met. Hence, the importance of the panoramic view in this research study.

3.4.5 Data simulation

This section presents the rendered output in Blender3D software after the mesh and textured models are applied on the simulated model. The model with texture consists of the path/road and environment (trees and buildings), lighting, camera, and collides are added. The lighting is added to the images with the sole purpose to simulate the light from the sun.

Furthermore, the movement of the camera simulates a vehicle's motion through the path/road created within the model. The movement gets the inputs from the keyboard. The colliders are also created in Blender3D software as objects that provide physical attributes to the model. These physical attributes are added to prevent the user from moving beyond the required space within the simulator.

3.5 Summary

In this chapter, the image-based rendering technique for utilisation in a simulator model was presented together with the supporting tools for the development of this model. “The pinhole technique” was used for camera configuration and Python programming language was utilised for camera switching at 3ms time intervals without the use of camera internal sensors.

Subsequent to the camera configuration model, the process for capturing the images from the fixed Hexagon Camera Configuration Model was outlined, as well as the modelling of the configuration model utilised in this research study. Following the image capture stage, the captured images were stored on a specified folder per camera, i.e. camera 1 = folder 1. This applies to the other five cameras, meaning that all six cameras each had its own folder with

image datasets. The datasets were then compressed and calibrated inside Blender3D software to reduce noise in the images. Lossy compression algorithm was utilised during the compression stage and contributed to the reduction of the dataset size, yet not affecting the image size and quality.

The created image datasets were then calibrated and the Structure from Motion Model and texturing were applied to the datasets inside Blender3D software. Additionally, the image rendering procedure and data simulation were achieved. The results discussion of the two processes namely rendering and simulation are discussed in Chapter 4.

Chapter 4: Results and discussions

In this chapter, the study results are outlined based on the implementation and methodology chapter. The results are analysed, then compared against the aim and objectives outlined in the introduction chapter. Subsequently, the results are presented outlining the bidirectional motion that is achieved using the panoramic view of the input image database obtained from the Hexagon Camera Configuration Model.

4.1 Introduction to the results chapter

The results outlined in this section depict the image rendering framework for the 364 .JPEG images that were captured on each camera at a total dataset worth 2184 .JPEG images at a high resolution of 1280 X 720 pixels at a total size of 39.8MB. The system required a dynamic scene with six cameras arranged at a 2D arc at a spanning of about 60° angle apart from each other. Additionally, each camera frame comprised of 364 .JPEG images that were captured from the real scene, and only then the process of image matching and texturing was applied using “depth maps. This resulted in an output of a textured .PNG image of 25.2 MB of 8192 X 9192 pixels resolution. The depth map components in this research study are treated as graphics models that can be easily rendered. Additionally, this image dataset utilises the blending approach that computes the weights of each depth image.

The selected depth maps in the textured image datasets produce acceptable output quality that can be utilised for this research study. The quality of the image reconstruction depends on the image quality and as a result of outdoor image capture, the texture and lighting were applied on each image dataset to enhance the image quality.

4.2 Input image results

To simulate a virtual curved mirror, the simple setup was arranged where a six-camera configuration arrangement was set up by placing this configuration model on the test vehicle’s rooftop, whilst the display output of the camera was on the computer. The model configuration on its own was insufficient, because the viewpoint of the camera depended on the position of

the viewer. Therefore, an alternative test had to be conducted. The second scenario had to be formulated where a curved mirror surface with the scene points are observed as indicated in Figure 4.1.



Figure 4.1: Input captured image scenery

The main challenge around the simulation of the curved mirror in the rendered scene is the display of active content correctly depending on the viewer's perspective. However, to simulate a large curved mirror surface, the camera display system must be able to capture the 3D environment while rendering the new view based on the viewer's position.

Furthermore, all the tasks were accomplished in real time. Otherwise, the virtual mirror system would lose the instant visual feedback required to provide the realism of the mirror. The IBR algorithm therefore seeks to advance the geometry and surface properties with images, and this is based on the image geometry and material attributes. A method from Szeliski's [85] study was inherited and utilised for image rendering in this research study.

Figure 4.1 further outlines the multiple captured images based on the Hexagon Camera Configuration Model and real-time image capture. The images outlined in Figure 4.1 are images that are captured from individual camera feeds and are stored in a folder based on their timestamp, and they are transposed to a master folder.

4.3 Rendering simulation outcomes

The technique of combining Street View imagery with other known image datasets makes the experience of using panoramic view mainly in the virtual world even richer. The use of Street View is essential as it is used as a tool to ease the livelihood with the ability to convert it into an application tool for driving directions. In the process of acquiring such results, the feature detection and matching utilising Structure from Motion technique is important.

Figure 4.2 presents the output results for feature detection and matching from the Structure from Motion algorithm. This output is based on the use of the SIFT feature detection algorithm that was utilised in Chapter 3. The SIFT algorithm was presented highlighting the number of interest points to detect the images utilising the Difference of Gaussian function as indicated in Figure 4.2. Furthermore, the feature tracks and the point of clouds are matched to capitulate the feature tracks. This is accomplished by tracking each feature in the corresponding image and testing it against the 3D point in the Structure from Motion on the drive testing route.

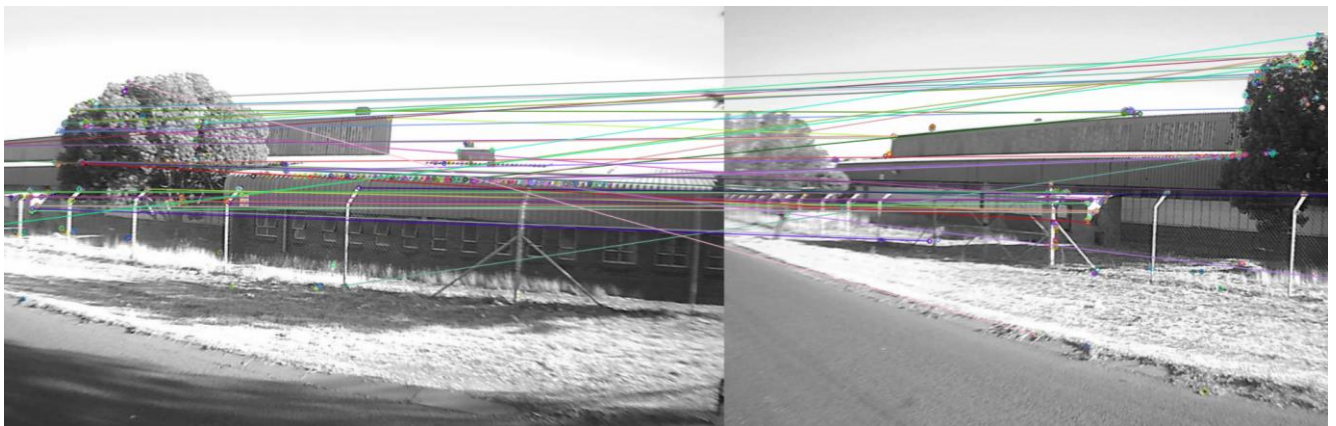


Figure 4.2: Feature detection and matching in Structure from Motion

Another element outlined in Figure 4.3 depicts the point of cloud where all the feature tracks are visible in the image. The incremental Structure from Motion (SFM) points are then used in the reconstruction of the scene and result in the formulation of the points of cloud as indicated in Figure 4.3. The output results are derived based on the camera position as indicated.

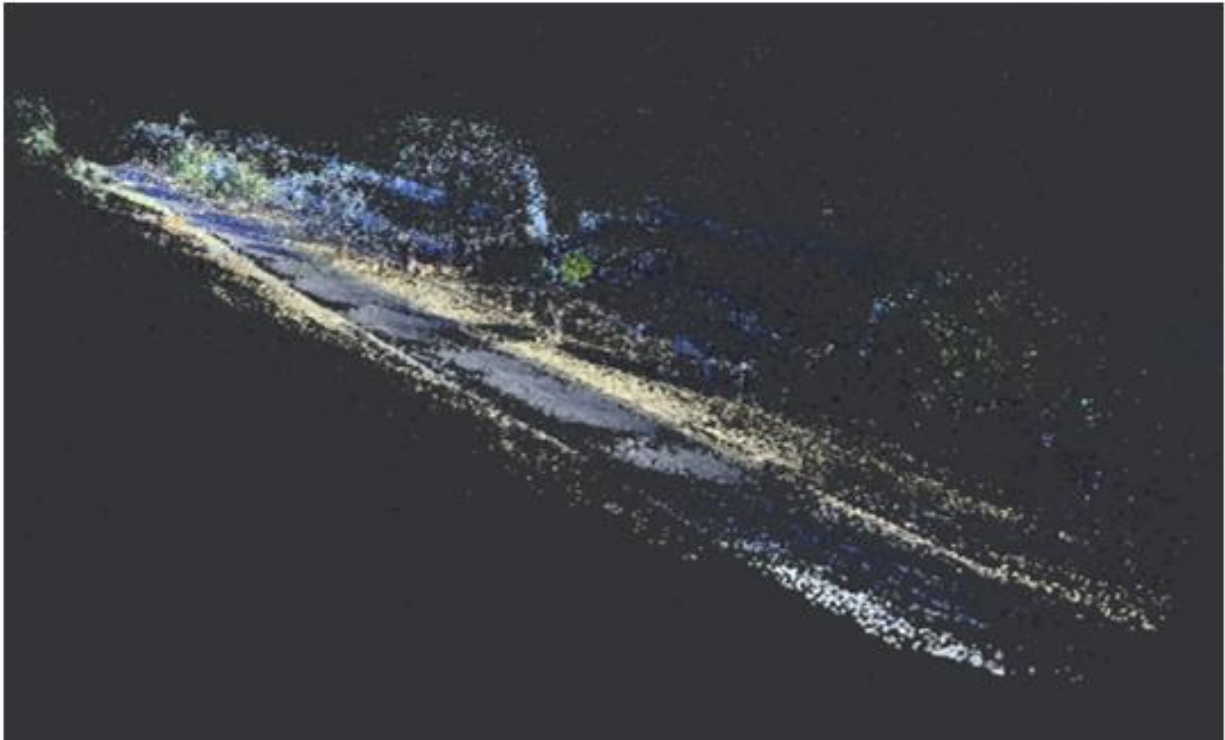


Figure 4.3: Point cloud and camera position reconstruction

Figure 4.3 depicts the dense geometry reconstruction of the scene. This is achieved by performing the multi-view stereo algorithm. The application of the multi-view stereo produces the depth maps and dense point cloud and with this, the algorithm can perform the map recovery. Following the completion of this process, a mesh of a scene is reconstructed by fusing the depth maps and the dense cloud that produces the mesh explained in line with Figure 4.4 for further apprehension of these results.



Figure 4.4: Dense point of cloud with Multiview stereo

Figure 4.4. depicts the output results in a form of a dense point of cloud with Multiview stereo. The reason for the conversion of the dense point of cloud was to find small and smooth points with a good fit to the plane.

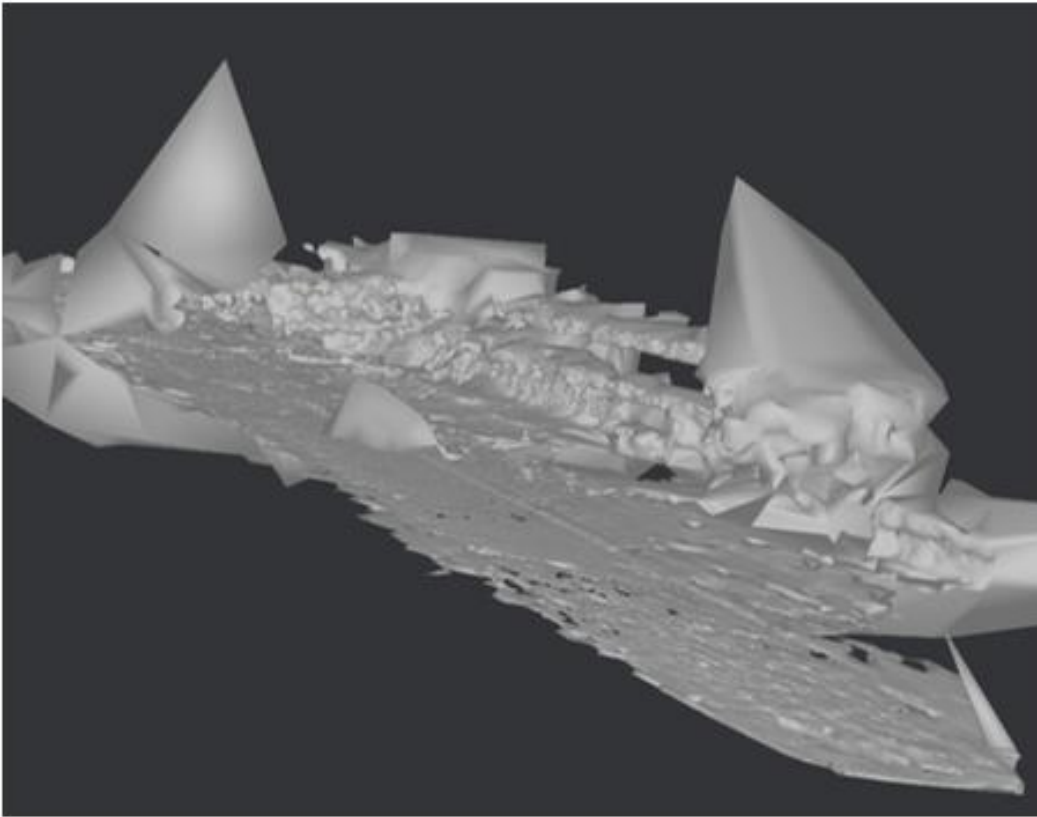


Figure 4.5: Mesh output from Multiview stereo

Following the creation of the mesh output, the texture was then generated by taking models, images, and the camera position (this was achieved through the use of the GPS coordinates). This was accomplished in the meshroom function by selecting 8 192 texture slides and unwrapping them to produce the desired expected output generate texture as indicated in Figure 4.6.

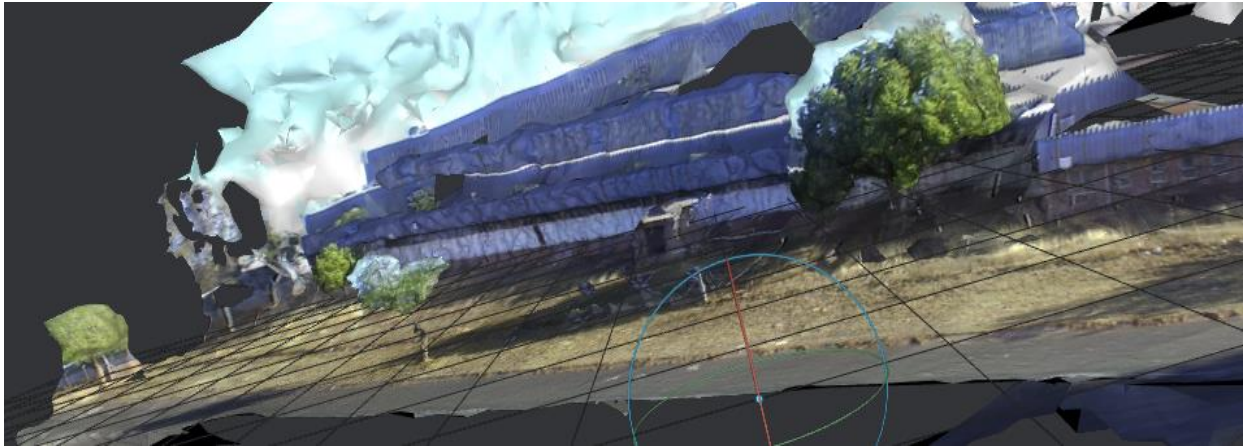


Figure 4.6: Generated texture

The depth map restoration and colour images were paired up to create the application of texture to the 3D mesh model and warp to the new viewpoint. Furthermore, the extraction of the depth map model from the 3D mesh, and the shortcomings were observed to be the delay by which the mesh update duration is long and as a result, affects the depth map extraction. The long duration observed for mesh update processing did not affect the expected results.

4.4 Output simulation results

To complete the simulation mesh model constructed from the different cameras, the integration of different models had to be applied. In addition, the scene construction was developed in Blender3D software including the camera, light, and colliders that are added to make the scene complete. The camera is then used to simulate the movement of the vehicle through the constructed scene, and the light is then added to illuminate the scene. The colliders are also added to simulate the physical attributes such as coordinates, gravity, and road limitation of the scene, as indicated in Figure 4.7.

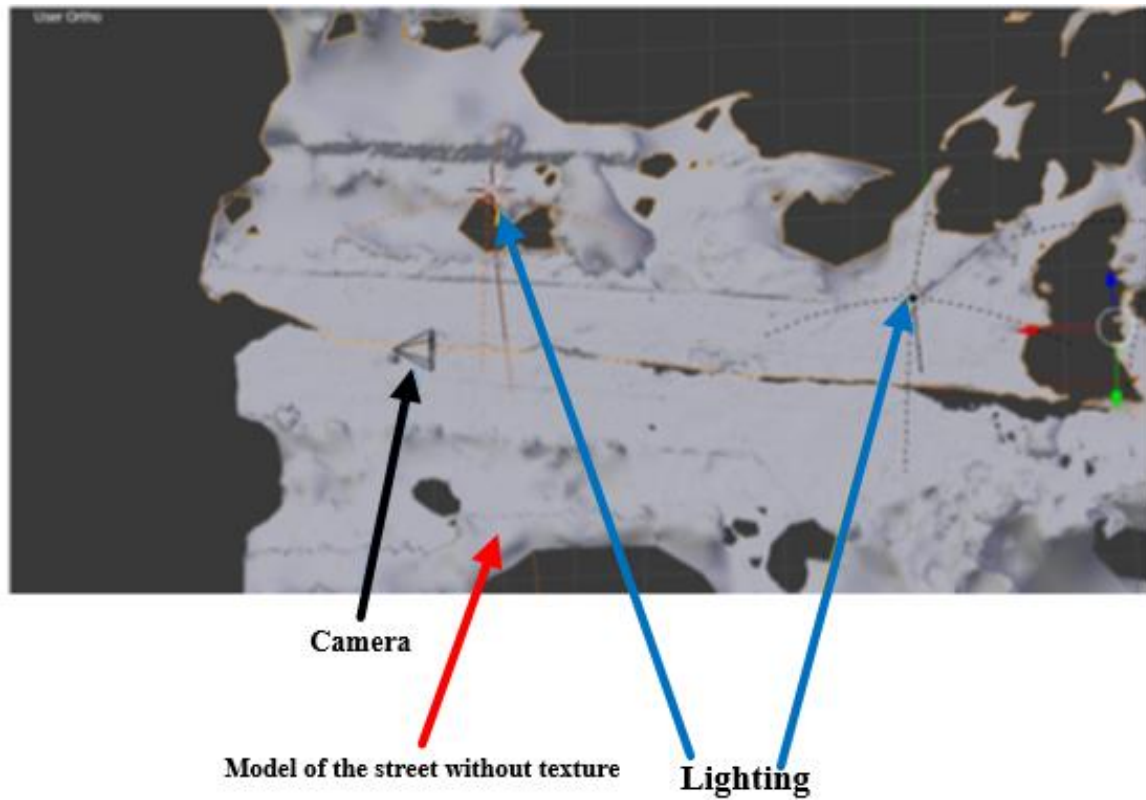


Figure 4.7: Model of the street without texture before simulation

The rendered street view panorama and its utilisation in a simulator are indicated in Figure 4.8. The rendering process takes place inside Blender3D, and the output is simulated and projected as depicted in Figure 4.8.



Figure 4.8: 3D-rendered street view

The technique for combining street view imagery with other known image datasets was extracted from the Hexagon Configuration Model. The experience of using a panoramic view mainly in the virtual world is deemed essential, as it is used as a tool to ease the livelihood with the ability to convert the view into an application tool that is used for driving or bicycle riding directions. This application enables individuals to move in a scene that was initially captured by a Hexagon Camera Configuration Model with a bidirectional movement option available. Additionally, all the depth maps are combined into a single large single-point cloud to produce a mesh that can easily be simulated.

A scale value is attached to every point which indicates the actual size. The actual .JPEG images were captured at 1280 X 720 resolutions and this is the default camera resolution. After applying texture the .PNG out images had a resolution of 8192 X 8192 pixels. These parameters depict the final mesh image that is extracted when the texture is applied to the resulting output dataset as indicated in figures 4.9 (a) and (b).

The final output that is represented in figures 4.9 (a) - 4.9(b) depicts the rendered view of the panoramic street views captured in Botshabelo. The panoramic street view in the context of this research study can be viewed from multiple viewing angles, and the navigation of the scene that was captured from a hexagon camera configuration is also feasible, as indicated in figures 4.9

(a) - (b). The depicted images are shown in a rendered street view from a top view perspective. This is achieved and projected from a horizontal street view in an omnidirectional manner. These results are outlined to provide the freedom of movement and the views that are obtained from the Hexagon Camera Configuration Model. These views allow for bidirectional movement within the simulator as indicated in figures 4.9 (a) and (b).

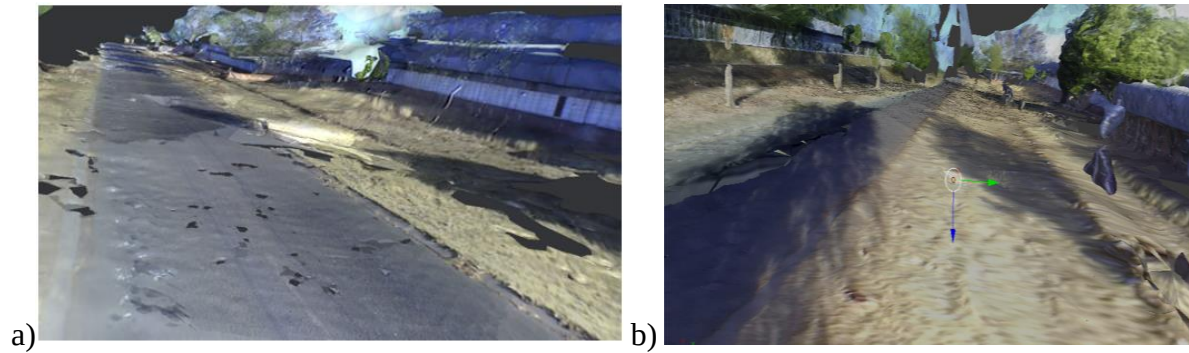


Figure 4.9: a) Rendered scene in a left direction; b) rendered scene in a reverse direction

The omnidirectional view of the rendered scene is deemed important as compared to the slider show that only allows for one-directional motion within the rendered scene. Figures 4.9 (a) and (b) presents the reversed horizontal view of the results outlining the effectiveness of this model. In addition, Figure 4.10 presents the comparison results between the rendered images and the initial captured image.

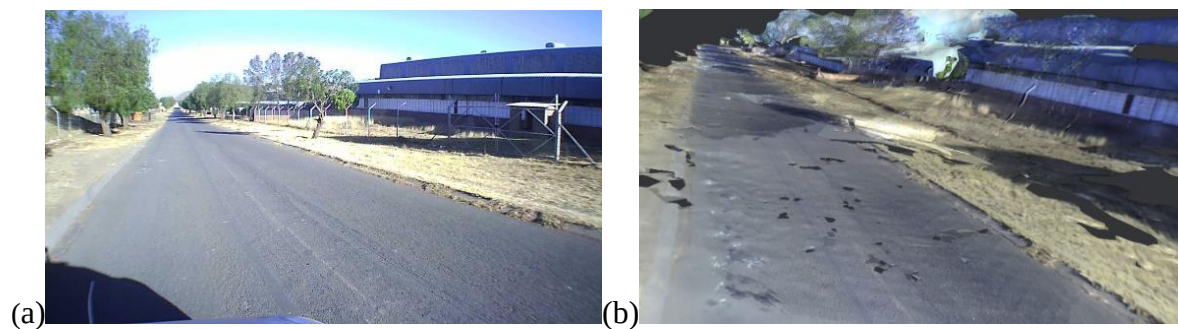


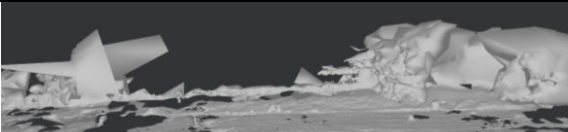
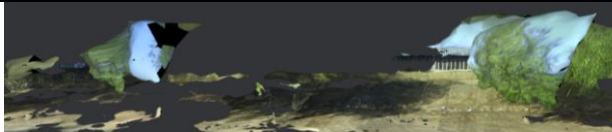
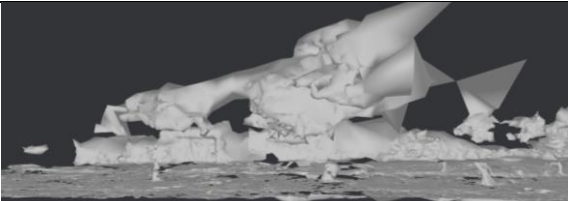
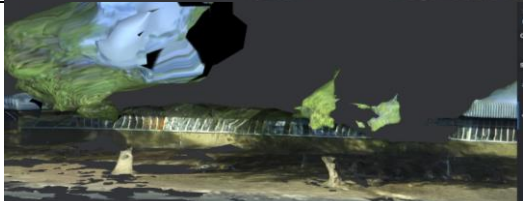
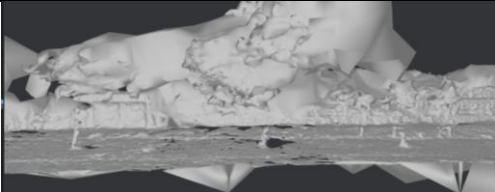
Figure 4.10: Comparison results between the captured and rendered image

Figure 4.10 (a) presents the originally captured image from one of the cameras. It is important to note that these images are obtained from six cameras and they are used as input datasets. Figure 4.10 (b) presents the output rendered scene. The process for image dataset constitutes a process within the value chain comprising of camera calibration, image compression, and

application of SFM and texture to the image dataset. The results shown in figures 4.10 (a) and (b) depict that through the completion of the successful application of the value chain process images can be converted into a dataset that can permit for bidirectional movement. As a result, the model efficacy can be determined by comparing the 2D input model against the rendered output model as outlined in figures 4.10 (a) and (b).

To determine the efficacy of the research output, Table 4.1 depicts the comparison output results based on the sample size to justify the study results.

Table 4.1: Efficacy comparison table based on image sample size

Number of samples	Point cloud	Number of points	Mesh	Scene	Efficacy
91		16 881			Low
182		39 690			Moderate
273		74 070			Medium

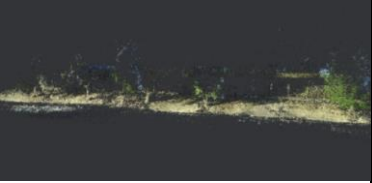
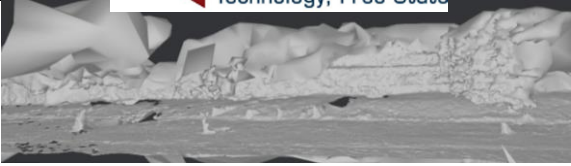
364		106 110			High
-----	---	---------	--	---	------

Table 4.1 presents the efficacy test results. The results are based on several sample images taken during the capture phase. The number of images in total were 364; however, the sampling was conducted on the multiple of 91 images per dataset. The set number of images taken or utilised for each sample image dataset was deemed sufficient as a prerogative of the researcher. For 91 image dataset, the number of points are 16 881, and as a result the efficacy of the study is determined by the realism of the rendered output scene from the point of cloud and mesh output.

For the 91 image dataset, the output scene is low as the realism of the entire scene is not complete. For 182 image dataset, the rendered output scene efficacy is moderate but still with low contrast of the realism, and the realism for the actual scene is imminent. For 273 image dataset, the output scene realism is observed as medium due to the high volume of the input image dataset. Furthermore, for 364 image dataset, the output scene is high as a result of a high volume of input image dataset with the scene realism observed for both points of cloud and mesh. This observation does demonstrate that the more images there are in a dataset, the higher the realism can be observed in a system.

4.5 Conclusion

In this chapter, the results of the tests that were conducted in Chapter 3 were evaluated and analysed with respect to the simulation and rendering of bidirectional panoramic views. The evaluation was observed based on the feasibility of the user to move forward and backward within the simulator. The prototype development produced good quality rendered frames with acceptable data size. In the context of this research study, acceptable data size refers to the quick simulation of a full dataset within a five minute period. The time frame was not scientifically proven, but it was the prerogative of the researcher and it was compared against the initial captured dataset simulation time frame. The success of the developed prototype was based on the ability of the model to permit bidirectional movement of the scene in a simulator. The results were then separated into scene simulation and panoramic image rendering applications as follows:

4.5.1. Panoramic image rendering

In this process, multiple application processes entailing several algorithms for image capture, compression, and texturing were utilised before the rendering could take place. It was noted that image compression is a very important process in the execution of such a computer vision application. The utilisation of the lossy algorithm compensated for the image dataset size of 38.9 MB and quality at 1270 X 720 pixels. This algorithm only reduced the dataset size but also increased the resolution size to 8192 X 8192 pixels at 25.2MB. Furthermore, the SIFT

algorithm was utilised for feature and point of interest detection for a set of 2D images. These image datasets were then compared against the camera pose related to the previous view. It was of high importance to detect the image features to enable for smooth image matching application. The reason for rendering the scene in a panoramic view was to allow the user to view around the rendered scene at any angle, and to permit them to move forward and backward within a simulator. Furthermore, the image rendering application utilising multi-view stereo was able to produce the depth maps and dense point cloud and enable the performance recovery of the maps.

4.5.2. Data simulation

The data simulation was performed on the meshed dataset that is attained from the Hexagon Configuration Model. The initial output mesh simulation light was not satisfactory, and as a result the colliders were added to the mesh inside Blender3D to complete the scene and increase the output light on the simulated dataset. This process was claimed following the application processes were SFM, and Multiview were applied. The introduction of the SFM meant that the pose of the calibration camera from a hexagon dataset of 2D images could be estimated, and as a result, the information of the 3D reconstruction point of cloud can be applied to the unknown objects. The objective of this research study was to test the feasibility for achieving bidirectional movement of a scene without the use of a 360° omnidirectional camera.

The results demonstrate that this task is feasible based on the tested Hexagon Camera Configuration Model. Figures 4.9 (a) and (b) presents the output results and the feasibility of a concept utilising a six-camera configuration model. It was important to test the feasibility of the same concept utilising a Hexagon Camera Configuration Model. These results are measured against the study objectives and the research question for testing of the feasibility of a developed 360° bidirectional view of a scene. The feasibility testing utilised a Hexagon Camera Configuration Model as compared to the use of a 360° omnidirectional camera. Furthermore, the input images are broken into two sets of input data, namely left and right cameras. The reason for this is to enable bidirectional movement within the simulator. The results also show the possibility for dataset creation from the SFM reconstruction with satisfactory output results.

Chapter 5: Conclusion & future work

This chapter outlines the conclusion to the research study and recommendations outlining the work that was not covered during the execution of this research study and also the study contribution are made. This chapter further makes mention of what was achieved in this research study.

5.1. Conclusions of the research done

This research study has contributed to human knowledge by testing the feasibility of the development of a Hexagon Camera Configuration Model that can permit for bidirectional movement of the scene within a simulator. Furthermore, this research study has presented an optimal utilisation and development model for spatial image enhancement without the use of a 360° omnidirectional camera but with the use of the 2D Hexagon Camera Configuration Model. The configuration method was developed by placing six cameras on the test vehicle rooftop and converting the captured 2D images to 3D images by adding the z-coordinate on the images. The drive test was conducted in Botshabelo township, and the rendering and scene creation were developed in Blender3D software.

The use of the image-based rendering technique utilising hexagon camera configuration was proposed as an ideal method due to the landscape of the testing site. To accomplish this, a technique with several algorithms based on SFM and SIFT was utilised. The feature detection and matching technique was observed as the best technique for detecting and matching the images from multiple image datasets. Subsequently, the process of data extraction from a 3D depth map to the mesh was outlined, and the restoration of the image data by utilising the 3D warping restoration approach was utilised.

Furthermore, it was proven that the use of Blender3D software is feasible and effective in rendering panoramic images from multiple image datasets and can also allow for code reuse. The use of computer vision algorithms were easily integrated into the Blender3D software and has proven the feasibility of the enhancement of spatial image data in a short period and with more accurate results.

The result and outcome of this research study concludes that the use of Google Street View and image rendering can be easily integrated into the existing simulations, and that they are quite useful in delivering beneficial information from camera input datasets. Furthermore, in evaluating the efficacy of this research study, the objective argument highlights that through the use of machine vision, the user can move back and forth within the simulator utilising multiple datasets that were captured from the Hexagon Camera Configuration Model. This contributes to the feasibility aspect of utilising multiple cameras to capture images and rendering these datasets to permit for bidirectional movement. In comparison to the use of a 360° omnidirectional camera, rendering of multiple scenes (datasets) permits the researcher to have control over the compression of images by labelling every dataset against its origin. This is complex using the 360° omnidirectional camera, as the entire dataset is stored into a master folder and images have to be selected individually.

The fundamental research questions for this research study was based on whether it was feasible to develop a bidirectional model from the Multimode Camera Configuration Model. The research study objectives were looking at the application of IBR techniques and enabling for bidirectional movement within a simulator, and the following objectives were achieved, as indicated:

- To incorporate the system into the simulation system in real time for increasing the realism of the simulation system in different geographical locations.
- To simulate a rendering technique for improvement of visual and spatial images, and the quality of the panoramic images for location identification.

The following outcomes were achieved as follows:

- The development of a rendered panoramic image model for the enhancement of virtual driving through incorporated image datasets for utilisation in a simulator.
- A possible image capturing technique for improvement of human interaction with the virtual world while driving or riding a bicycle.

5.2. Contributions from this research study

This research study has produced the following contributions in furthering the knowledge contribution in the field of computer vision as follows:

- 6 Degree of Freedom (6 DoF) from the Hexagon Camera Configuration Model - where the user can move in any direction as opposed to the use of a single slide that allows for one-directional movement in a street view scene.
- The development of the Hexagon Camera Configuration Model that is used to capture images and produce a rendered scene that allows for bidirectional movement of the scene within a simulator.
- Application of rendered images for the enhancement of virtual driving as proof of concept.
- Image capturing technique for improvement of human interaction with the virtual world while traveling through a smart city.

5.3. Final dissertation remarks

In view of this research's methods and results, it is evident that the Hexagon Camera Configuration Model with a set of camera models from the same manufacturer and focal length can provide a new dimension of Multiview. Furthermore, this camera configuration model allows for a scene of interest to be viewed from a simulator and can permit for bi-directional movement within the scene.

5.4. Suggested future work

Following the completion of this research study, a window for advanced future investigations in this field of study was noted. Due to the current study framework and scope, there were limitations, and as a result, not all identified aspects of this research topic were addressed. Therefore, the continuing investigation of the study can further be researched by addressing the following:

- Error evaluation algorithm for input view computation.
- The improvements on the rendering quality through the use of better surface meshing processing techniques. In this research study, it was not crucial to look at the rendering scene quality as it was out of scope. This study focuses on the image capture techniques and the ability to permit bidirectional movement within the rendered scene.

- The use of one of the cameras to determine the viewpoint and adapt the display accordingly.
- Investigation on the effect of using different camera models with different focal lengths and from different manufacturers as opposed to utilisation of the same camera models from the same manufacturer.
- Developing a quantitative evaluation approach to determine the feasibility and importance of such a research study.

References

- [1] J. Foley, “Getting There: The Ten Top Problems Left,” *IEEE Computer Graphics and Application*, vol. 20, no. 1, pp. 66-68, 2000.
- [2] M. Oliveira, “Image-Based Modeling and Rendering Techniques: A survey,” vol. 9, no. 2, p. 38, 2002.
- [3] L. Yin, Q. Cheng, Z. Wandy and Z. Shao, “Big data for pedestrian volume: Exploring the Sasol,” *Exploring the use of Google street view images for pedestrian counts*, no. 63, pp. 337-345, 2015.
- [4] N. Runge, P. Samsonov, D. Degraen and J. Schoning , “No more autobahn: Scenic route generation using Googles Street View,” *In Proceedings of the I*, 7-10 March 2016.
- [5] L. Stamos and P. Allen, “Geometry and texture recovery of scene of large scale,” vol. 88, no. 2, pp. 94-118, 2002.
- [6] R. Sato, S. Ono, H. Kawasaki and K. Ikeuchi, “Photo-Realistic Driving simulator using Eigen Texture and Real-Time Restoration Techniques by GPU,” vol. 6, no. 2, p. 87, 2 December 2008.
- [7] C. Vandeviver, “Applying Google Maps and Google Street View in Criminological Research,” pp. 1-16, 2014.
- [8] T. Y. Lin, Y. Cui, S. Belongie and J. Hays, “Learning deep representations for ground to aerial geolocalization,” *In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5007-5015, 2015.
- [9] F. Wang and A. Gupta, “Unsupervised learning of visual representations using videos,” *In: Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794-2802, 2015.
- [10] N. N. Vo and J. Hays, “Localizing and Orienting Street View Using Overhead Imagery,” p. 2, 2016.

- [11] S. Ono, K. Ogawara, M. Kagesawa, H. Kawasaki, M. Onuki, J. Abeki, T. Yano, M. Nerio, K. Honda and K. Ikeuchi, “A photo-realistic driving system for mixed-reality traffic experiment space,” p. 2, 6-9 June 2005.
- [12] J. Minichino and J. Howse, “Learning OpenCV 3 Computer Vision with Python,” p. 1, 2015.
- [13] E. H. Adelson and J. Bergen, “The plenoptic function and elements of early vision. In computational Model of Visual Processing,” in *MIT Press*, Cambridge, MA, 1991.
- [14] X. Li and C. Ratti, “Mapping the spatial distribution of shade provision of street tree in Boston using Google Street View panoramas,” pp. 109-119, 25 February 2018.
- [15] J. Martinez-Carranza, M. Varela, L. Oyuki, A. Ponce and R. Dominguez-Colin, “An open source based software to capture aerial images and video using drones 22 Reality, DATA and space international Journal of Statistics and Geography,” vol. 11, no. 1, pp. 22-37, January 2020.
- [16] S. Dong, K. Xu, Q. Zhou, A. Tagliasacchi, S. Xin, M. Niebner and B. Chen, “Multi-Robot Collarative Dense Scene Reconstruction,” *ACM Trans Graph*, vol. 38, no. 4, pp. 1-16, 12 July 2019.
- [17] S. Bianco, G. Ciocca and D. Marelli, “Evaluating the performance of Structure From Motion Pipelines,” pp. 1-18, 1 August 2018.
- [18] J. L. Schonberger and J. Frahm, “Structure From Motion Revisited.,” pp. 4104-4113, 2016.
- [19] K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-OR and H. Huang, “Autonomous Reconstruction of Unknown Indoor Scenes Guided by Time-Varying Tensor Fields,” vol. 36, no. 6, p. 15, November 2017.
- [20] J. Sivic, B. Kaneva, A. Torralba, S. Avidan and W. T. Freeman, “Creating and Exploring a Large Photorealistic Virtual Space,” *Proc. IEEE Workshop on Internet Vision*, pp. 1- 8, 2008.

- [21] J. Kopf, B. Chen, R. Szeliski and M. Cohen, “Street slide: Browsing Street Level Imagery,” *ACM Trans.Graphics*, vol. 29, no. 4, pp. 2-9, July 2010.
- [22] C. Peng, B.-Y. Chen and C.-H. Tsai, “Integrated Google Maps and Smooth Street View Video for Route Planning,” p. 1, 16-18 December 2010.
- [23] T. M. Cover and J. A. Thomas, “Elements of Information Theory,” John Wiley & Sons, 2012.
- [24] M. D. Kokate, V. Wankhde and R. S. Patil, “Survey: Image Mosaicing based on feature extraction,” *International Journal of Computer Applications*, vol. 165, no. 1, pp. 26-30, May 2017.
- [25] A. Rzotkiewicz, A. Pearson, B. Dougherty, A. Shortridge and N. Wilson, “Systematic review of the use of Google Street View in Health research: Major themes, strengths, weaknesses and possibilities for future research,” *Elsevier*, vol. 52, pp. 240-246, 2018.
- [26] F. Condorelli and F. Rinaudo, “Cultural heritage reconstruction from historical photographs and videos,” *International Arch Photogramm. Remote Sensing. Spatial Inf.Sci*, vol. 2, pp. 259-265, 2018.
- [27] W. Wahbeh, S. Nebiker and G. Fangi, “Combining public domain and professional panoramic imagery for the accurate and dense 3D reconstruction of the destroyed bel temple in Palmyra,” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 5, pp. 81-88, 2016.
- [28] J. Kopf, B. Chen, R. Szeliski and M. Cohen , “Street Slide: Browsing Street Level Imagery,” *ACM Transaction on Graphics*, vol. 29, no. 4, pp. 1-8, July 2010.
- [29] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, “The Lumigraph. In Computer Graphics Proceedings, Annual conference series,” in *ACM SIGGRAPH, Proc. SIGGRAPH 96*, New Orleans, 1996.
- [30] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin and R. Szeliski, “Photographing long scenes with multi-viewpoint panoramas,” *ACM Transactions on Graphics* 25, pp. 853-861, 3 August 2006.

- [31] L. Najafizadeh and J. E. Froehlich, "A feasibility study of using Google Street View and Computer Vision to Track the Evaluation of Urban Accessibility," p. 1, 22-24 October 2018.
- [32] I. Serferling, N. Naik, C. Ratti and R. Proulx, "Green street - Qualifying and mapping urban trees with street-level imagery and computer vision," pp. 93-101, 13 May 2017.
- [33] S.B. Kang, "A survey on image-based rendering techniques," in *SPIE International Symposium on Electronic Imaging: Science and Technology*, San Jose, CA, 1999.
- [34] J. Voumard, A. Abellan, P. Nicolet, I. Penna, M. A. Chanut, M.-H. Derron and M. Jaboyedoff, "Using street view imagery for 3-D survey of rock slope failure," p. 2093, 1 December 2017.
- [35] C. P. Gribble, A. J. Stephens, J. E. Guilkey and S. G. Parker, "Visualizing particle-based simulation datasets on the desktop," p. 1, 2006.
- [36] S. B. Kang, "A survey of image-based rendering," *In VideoMertics*, vol. 3641, pp. 2-16, 23-29 January 1999.
- [37] J. Lengyel, "The convergence of graphics and vision," *Technical report. IEEE Computer*, July 1998.
- [38] Z. Hu and Y. Ming, "Modeling stereopsis via markov random field," *Neural Computation*, vol. 22, no. 8, pp. 2161-2191, 2010.
- [39] C. Gilge, "Google street view and image as experience," *GeoHumanities*, vol. 2, no. 2, pp. 469-484, 2016.
- [40] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent and J. Weaver, "Google Street View: Capturing the world t street level, Computer," vol. 43, pp. 32-38, 2010.
- [41] D. Anguelov, "Google Street View: Capturing the world at street level," p. 34, 2010.
- [42] L. Lenkoe, B. Kotze and P. Veldtsman, "Simulation of Hexagon Spatial Dataset for Free Motion in a Simulator for Smart City Bidirectional Navigation Purposes," in *The Sixth*

International Conference on Applications and Systems of Visual Paradigms - Visual 2021,
Nice, France, 2021.

- [43] H. Y. Shum and S. B. Kang, "A Review of image-based rendering techniques," vol. 213,
p. 24, May 2000.
- [44] Y. Mao, G. Cheung, A. Ortega and Y. Ji, "Expansion hole filling in depth-image-based
rendering using graph-based interpolation," pp. 1859-1863, 2013.
- [45] S. Shi, K. Nahrstedt and R. Campbell, "A real-time remote rendering system for
interactive mobile graphics," *ACM Transactions on Multimedia Computing,
Communications and Applications (TOMCCAP)*, vol. 8, no. 3, pp. 1-20, 2012.
- [46] M. Song and W. Kim, "Depth estimation from a single image using guided deep learning,"
IEEE Transactions and Journals, vol. 4, pp. 1-12, 2016.
- [47] A. I. Audu, "Camera positioning for 3D panoramic image rendering," p. 25, 2015.
- [48] H. Lawaniya, "A gentle introduction to computer vision," *IET Computer Vision*, 2020.
- [49] D. Eck, "Introduction to Computer Graphics," 2021.
- [50] L. McMillan, "An Image based approach to three dimensional computer graphics," Chapel
Hill, 1997.
- [51] A. Speers and A. Jenkin, "Tuning stereo image matching with stereo video sequence
processing," in *Joint International Conference on Human-Centered Computer
Environments*, vol. HCCE 2012, pp. 208-214, 2012.
- [52] D. N. Bhat and S. K. Nayar, "Ordinal measures for image correspondence," *Pattern
Analysis and Machine Intelligence, IEEE Transaction on*, vol. 20, no. 4, pp. 415-423,
1998.
- [53] Z. Qin, M. McCool and C. Kaplan, "Precise Vector Textures for Real-Time 3D
Rendering," p. 4, 2008.
- [54] K. Moule and M. McCool, "Efficient bounded adaptive tessellation of displacement maps,"
In proc. of GI, pp. 171-180, 2002.

- [55] H. Y. Shum and S. B. Kang, “A review of image based rendering techniques,” vol. 4067, pp. 2-13, May 2000.
- [56] P. E. Debevec, C. J. Taylor and J. Malik, “Modeling and Rendering architecture from photos: A Hybrid geometry- and image-based approach,” *In ACM SIGGRAPH*, pp. 11-20, 1996.
- [57] P. Akyazi and P. Frossard, “Graph-Based Interpolation for Zooming in 3D scene,” *25th European Signal Processing Conference*, p. 1, 2017.
- [58] R. E. Klosterman, “The What if? Collaborative planning support system,” *Environment and Planning B: Planning and Design*, pp. 393-408, 1999.
- [59] D. Z. Sui, “GIS-based urban modeling: practices, problem and prospects.,” *International Journal of Geographical Information Science*, no. 12, pp. 651-671, 1998.
- [60] I. Hoelzi and R. Marie, “Google Street View: navigating the operative image,” vol. 29, no. 3, pp. 261-271, September 2014.
- [61] M. Borg, “Github,” [Online]. Available: <https://mark-borg.github.io/assets/omniapp-chapter2.pdf>. [Accessed 20 June 2021].
- [62] E. Adel, M. Elmogy and H. Elbakry, “Image stitching,” *International Journal of Computer Applications*, vol. 99, no. 6, pp. 1-8, 2014.
- [63] M. Z. Bonny and S. Uddin, “Feature-based Image Stitching Algorithms,” pp. 198-203, 12-13 December 2016.
- [64] T. Siddique, Y. Rehman, T. Rafiq, M. Nisar, M. Ibrahim and M. Usman, “3D Object localisation using 2D estimates for computer vision applications,” in *2021 Mohammad Ali Jinnah University International Conference on Computing*, 2021.
- [65] H. Y. Shum, “Construction of Panoramic Image Mosaic with Global and Local Alignment,” *Kluwer Academic Publishers*, p. 1, 1999.
- [66] X. Li and C. Ratti, “Mapping the spatial distribution of shade provision of street trees in Boston using Google Street View panoramas,” p. 3, March 2018.

- [67] J. Salmen, S. H. Houben and M. Schlipsing, “Google Street View Images Support the Development of Vision-Based Driver Assistance,” p. 2, 2012.
- [68] S. S. Tung and W. L. Hwang, “Depth Extraction from a Single Image and Its Application,” p. 1, 13 February 2019.
- [69] Amazon,
“https://www.amazon.com/dp/B01N8TX7PD/ref=psdc_7161074011_t3_B074ZFXJTL,”
Amazon, [Online]. Available:
https://www.amazon.com/dp/B01N8TX7PD/ref=psdc_7161074011_t3_B074ZFXJTL.
[Accessed 27 03 2020].
- [70] Blender, [Online]. Available: <https://www.blender.org/>. [Accessed 25 02 2020].
- [71] M. Herold, H. Couclelis and K. Clarke, “The role of spatial metrics in the analysis and modeling of urban land use change,” pp. 369-399, 3 December 2003.
- [72] “<https://www.ics.uci.edu/~majumder/vispercep/cameracalib.pdf>,” [Online]. Available:
<https://www.ics.uci.edu/~majumder/vispercep/cameracalib.pdf>. [Accessed 30 03 2020].
- [73] A. Dehghani and A. Pourmohammad, “Single Camera Vehicle Speed Measurement,” pp. 190-193, September 2013.
- [74] D. Rojatkhar, N. Borkar, B. Naik and R. Peddiwar, “Image compression techniques: Lossy and Lossless,” *International Journal of Engineering Research and General Science* , vol. 3, no. 2, pp. 912-917, 2015.
- [75] D. Venugopal, S. Mohan and S. Raja, “An efficient block based lossless compression of medical images,” *Optik*, pp. 754-758, 2016.
- [76] M. Ferroukhi, A. Ouahabi, M. Attari, Y. Habchi and A. Taleb-Ahmed, “Medical video coding based on 2nd-generation wavelets,” *Performance Evaluation*, p. 88, 2019.
- [77] M. Mandal, “Data Science Blogathon,” Analytics Vidhya, [Online]. Available:
<https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>.
[Accessed 12 05 2022].

- [78] G. T. Laureano, M. Paiva, A. Soares and C. Coelho, “A topological approach for detection of chessboard patterns for camera calibration,” *Emerging Trends in Image Processing, Computer Vision and Pattern Recognition*, pp. 517-531, 2015.
- [79] P. Miro, Alba, Casas, R. Josep , R. Hidalgo and Javier, “Correspondence matching in unorganised 3D point clouds using Convolution Neural Networks,” *Image and Vision Computing*, Vols. 83-84, pp. 51-60, 2019.
- [80] A. Moncef and A. Mhamed, “A review on 3D Reconstruction Techniques from 2D images,” p. 9, February 2020.
- [81] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz and R. Szeliski, “Building Rome in a day,” pp. 72-79, 2009.
- [82] D. Crandall, A. Owens, N. Snavely and D. P. Huttenlocher, “Discrete continuous optimisation for large-scale structure from motion,” pp. 3001-3008, 2011.
- [83] A. Henrich, “3D Reconstruction and Camera Calibration from 2D images,” p. 3, 2000.
- [84] Z. Zhang, “Determining the Epipolar Geometry and its Uncertainty,” *A Review. The International Journal of Computer Vision*, vol. 27, no. 2, pp. 161-195, March 1998.
- [85] R. Szeliski, “Image Alignment Stitching,” pp. 1-87, 10 December 2006.
- [86] P. Ahirwar and D. Nagar, “Image compression: A review,” *International Journal of Advanced Research in Science, Communications and Technology*, vol. 7, no. 2, pp. 184-189, 2021.

The annexure section presents the information that was included in the body of the dissertation, with the reason either being size or space. The annexure section comprises of eight sections presenting additional images, figures, and codes that were used for the execution of the prototype. The image size difference and the reason for selection between lossy and lossless compression algorithms are outlined in Annexure A, followed by the full algorithm code which is broken into pieces in the dissertation and in line with the subject content in Annexure B.

Annexures C and D present a copy of the two internationally published conference papers. Additionally, Annexure E depicts the modelling input of the captured images before image sorting based on the timestamp can be actioned, while Annexure F depicts the feature detection algorithm applied to the image datasets, with Annexure G presenting the final rendered output panoramic view, and Annexure H presenting the image calibration code.

Annexure A: Lossy flow chart and code algorithm

Lossless compression(5% quality) file size 26.8KB

Input raw data

image



Lossless compression (100% quality) file size 398KB



Lossy compression(100% quality) file size 2,68MB



Lossy compression (9% quality) file size 1.05MB



Annexure B: Code algorithm

```
import cv2

cam = cv2.VideoCapture(0)

cv2.namedWindow("Capturing Images")

img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break

    cv2.imshow("Capture Calibration", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE pressed
        img_name =
"calibration_images/back/calibration_image_{}.png".format(img_counter)

        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

```
import cv2
import numpy as np
import glob
from tqdm import tqdm
import PIL.ExifTags
import PIL.Image
from matplotlib import pyplot as plt

def create_output(vertices, colors, filename):
    colors = colors.reshape(-1,3)
    vertices = np.hstack([vertices.reshape(-1,3), colors])

    ply_header = '''ply
format ascii 1.0
element vertex %(vert_num)d
property float x
property float y
property float z
property uchar red
property uchar green
```

```

        property uchar blue
    end_header
'''

with open(filename, 'w') as f:
    f.write(ply_header %dict(vertnum=len(vertices)))
    np.savetxt(f, vertices, '%f %f %f %d %d %d')

def downsample(image, reduce_factor):
    for i in range(0, reduce_factor):
        #Check if image is color or grayscale
        if len(image.shape) > 2:
            row, col = image.shape[:2]
        else:
            row, col = image.shape

        image = cv2.pyrDown(image, dstsize= (col//2, row // 2))
    return image

ret = np.load('camera_params/ret.npy')
K = np.load('camera_params/K.npy')
dist = np.load('camera_params/dist.npy')

img_path1 = '2019-11-07_122706.jpg'
img_path2 = '2019-11-07_122716.jpg'

img_1 = cv2.imread(img_path1)
img_2 = cv2.imread(img_path2)

#Get height and width. Note: It assumes that both pictures are the same
size. They HAVE to be same size and height.
h,w = img_2.shape[:2]

#Get optimal camera matrix for better undistortion
new_camera_matrix, roi =
cv2.getOptimalNewCameraMatrix(K,dist,(w,h),1,(w,h))

img_1_undistorted = cv2.undistort(img_1, K, dist, None, new_camera_matrix)
img_2_undistorted = cv2.undistort(img_2, K, dist, None, new_camera_matrix)

cv2.imshow('undistorted_1',img_1_undistorted)
cv2.imshow('undistorted_2',img_2_undistorted)

gray = cv2.cvtColor(img_2_undistorted, cv2.COLOR_BGR2GRAY)
sift = cv2.xfeatures2d.SIFT_create()
keypoints, descriptors = sift.detectAndCompute(img_2_undistorted, None)
# draw the detected key points
sift_image = cv2.drawKeypoints(gray, keypoints, img_2_undistorted)

img1 = downsample(img_1_undistorted,3)
img2= downsample(img_2_undistorted,3)

cv2.imwrite('undistorted_left.jpg', img1)
cv2.imwrite('undistorted_right.jpg', img2)

win_size = 5
min_disp = -1
max_disp = 63 #min_disp * 9
num_disp = max_disp - min_disp # Needs to be divisible by 16

```

```
#Create Block matching object.
stereo = cv2.StereoSGBM_create(minDisparity= min_disp,
    numDisparities = num_disp,
    blockSize = 5,
    uniquenessRatio = 5,
    speckleWindowSize = 5,
    speckleRange = 5,
    disp12MaxDiff = 2,
    P1 = 8*3*win_size**2,
    P2 = 32*3*win_size**2)

disparity_map = stereo.compute(img1, img2)

h,w = img2.shape[:2]
focal_length = np.load('camera_params/FocalLength.npy')
Q1 = np.float32([[1,0,0,0],
    [0,-1,0,0],
    [0,0,focal_length*0.05,0],
    [0,0,0,1]])

points_3D = cv2.reprojectImageTo3D(disparity_map, Q1)

colors = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)

mask_map = disparity_map > disparity_map.min()

output_points = points_3D[mask_map]
output_colors = colors[mask_map]

#Define name for output file
output_file = 'reconstructed.ply'

create_output(output_points, output_colors, output_file)
```

```
import cv2
import numpy as np
import glob
from tqdm import tqdm
import PIL.ExifTags
import PIL.Image

chessboard_size = (9,6)
obj_points = []
img_points = []

objp = np.zeros((np.prod(chessboard_size),3),dtype=np.float32)
objp[:, :2] = np.mgrid[0:chessboard_size[0],
                       0:chessboard_size[1]].T.reshape(-1,2)
calibration_paths = glob.glob('calibration_images/Front_center/*')
for image_path in tqdm(calibration_paths):
    image = cv2.imread(image_path)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    print("Image loaded, Analyzing...")

    ret, corners = cv2.findChessboardCorners(gray_image, chessboard_size, None)
    if ret == True:

        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
        cv2.cornerSubPix(gray_image, corners, (5,5), (-1,-1), criteria)
        obj_points.append(objp)
        img_points.append(corners)

ret, K, dist, rvecs, tvecs = cv2.calibrateCamera(obj_points, img_points,
gray_image.shape[::-1], None, None)

np.save("camera_params/ret", ret)
np.save("camera_params/K", K)
np.save("camera_params/dist", dist)
np.save("camera_params/rvecs", rvecs)
np.save("camera_params/tvecs", tvecs)

exif_img = PIL.Image.open(calibration_paths[0])
exif_data = {
    PIL.ExifTags.TAGS[k]:v
    for k, v in exif_img._getexif().items()
    if k in PIL.ExifTags.TAGS}
focal_length_exif = exif_data['FocalLength']
focal_length = focal_length_exif[0]/focal_length_exif[1]
np.save("./camera_params/FocalLength", focal_length)
```

Annexure C: Simulation of Hexagon Spatial Image Datasets for Free-Motion in Simulator for Smart City Bidirectional Navigation Purposes

Simulation of Hexagon Spatial Image Datasets for Free Motion in a Simulator for Smart City Bidirectional Navigation Purposes

Lepekola I. Lenkoe

Department of Electrical,
Electronic and Computer
Engineering
Central University of
Technology, Free State
Bloemfontein, South Africa
email: pexonl3@gmail.com

Ben Kotze

Department of Electrical,
Electronic and Computer
Engineering
Central University of
Technology, Free State
Bloemfontein, South Africa
email: bkotze@cut.ac.za

Pieter Veldtsman

Department of Electrical,
Electronic and Computer
Engineering
Central University of
Technology, Free State
Bloemfontein, South Africa
email: pveldtsm@cut.ac.z

Abstract—It has been noted that Google Street View serves millions of users with panoramic imagery across the globe. However, the configuration model such as the use of a 360° omnidirectional camera is utilised. However, an optimal model for capturing, calibrating, and compressing the captured images differs. It is for these reasons that a Hexagon Camera Configuration Model is investigated, including but not limited to imagery capture and simulation of results to allow for bidirectional navigation within a simulator with the ability to view the initially uncaptured scenery. On the configuration model, cameras were placed at a 60° angle apart to obtain the full 360° scenery view. It is seen that this model can optimally produce a full 360° panoramic view with the capabilities for a bidirectional view and movement of the initially uncaptured scene/ view through the application of the image rendering technique.

Keywords – *Simulation; Image-Based Rendering; Spatial datasets; Google Street View; Smart cities;*

I. Introduction

The smart city concept has the potential to capture real-time data that communicates with stakeholders for optimising decision-making utilising artificial intelligence and a low latency response rate. In addition, the modelling and visualisation of complex processes and data are significant. Hence, the hype in the furthering the concept of Google Street View (GSV) for mapping and documenting of rendered spatial built environment [1].

Furthermore, GSV is deemed as a technology implemented in several Google services to provide the user interested in viewing a particular location on the map with panoramic images [2]. In addition, the GSV implementation in most cases is achieved utilising the Image-Based Rendering (IBR) technique. Despite the selection of the IBR technique in this paper, several modelling techniques can be utilised to achieve the same

results such as the Model-Based Rendering (MBR) technique. However, the construction and implementation of such techniques rely on techniques such as Structure from Motion, which is used to build 3D models from both structured and unstructured image collection depending on the dataset model [3].

II. Problem statement

In most GSV-based applications, a 360° omnidirectional camera is utilised for image capture. However, as a result of having a single dataset, the application becomes limited and reduces the application of a full 3D panoramic view processing power. It is for such reasons that a new image capture technique utilising six (6) camera configurations at a 60° angle needs to be tested and investigated and modelled for a full 3D panoramic view to observe the feasibility for free motion in a simulator for bidirectional imagery viewing of the initially captured images utilising an alternative model configuration as opposed to 360° omnidirectional camera.

III. Objectives of this study is therefore to:

- Simulate a rendering technique for improvement of visual and spatial images, and quality of the panoramic images for location identification.
- Present a framework that allows for omnidirectional virtual driving.
- Model image data collection technique utilising Hexagon Camera Configuration Model.

IV. Original contributions of this research paper

This research paper has produced the following contributions in furthering the knowledge contribution in the field of computer vision as follows:

- Development of a rendered panoramic image model for the enhancement of virtual driving through incorporated image datasets for utilisation in a simulator.
- Image capturing technique for improvement of human interaction with the virtual world while traveling through a smart city.

This paper is arranged as follows: Section II gives background to the work conducted in this field also pointing out the shortcomings from the research conducted. Section III presents the configuration model for image capture and data processing techniques, while Section IV discusses the results obtained from the study. Section V awards the conclusion of the results obtained from the study.

V. Literature Review

The use of datasets in a simulator for constructing image rendering and camera content acquisition in a 3D content view is regarded as a significant approach in such a study.

It is with such reasons, that a study from Li *et al.* [4] outlines the significance of cities in the context of global warming and urbanisation that is also interpreted and analysed for further developments.

This is as a result of the view dependency, which means that the explicit geometric rendering much relies on the known approximate environment [5]. The use of the explicit rendering becomes complex in an informal environment/settlement due to the tiring exercise of data collection, which is skewed since residents can be built on any topography. However, the observation of the feature detection and matching technique utilising IBR technique for multiple image datasets is feasible to achieve the objectives of such a study.

It is with such reasons supporting the use of the advanced IBR method presented by Mao *et al.* [7] and Shi *et al.* [8] outlining the depth map containing associated pixels to the reference image in a 3D environment. It is noted according to the literature that depth estimation from a single image is an important issue in understanding a 3D scene.

In addition to the parametric methods for extracting depths, many non-parametric depth sampling approaches have also been proposed to automatically convert monocular images into stereoscopic images with good performances.

Unfortunately, vision techniques are not robust enough currently to recover accurate 3D models. In addition, it is difficult to capture visual effects such

as highlights reflections and transparency making use of single texture-mapped model.

It is for such reasons that the rendering techniques are associated with computer graphics to enable better processing of images [9].

The evolution of such an innovation is mainly based on the increased data resources, multiple spatial datasets, and tools for processing and computation [10] [11]. This paper focuses on the use of multiple cameras rather than the use of a single 360° omnidirectional camera to allow for bidirectional imagery viewing of the initially uncaptured scene. It is noted against the literature that much work has been conducted in this field of study. However, the exploration of IBR technic utilising another configuration model apart from the use of a 360° omnidirectional camera has not yet been fully explored. Furthermore, the use of the 360° omnidirectional camera has still not yet proved the concept of imagery view of the uncaptured scene.

VI. Methodology

Figure 1 depicts the camera configuration setup. However, this model can be altered depending on the testing site, and theoretically, the number of cameras does not affect the results, but rather affects the resources required to process the output.

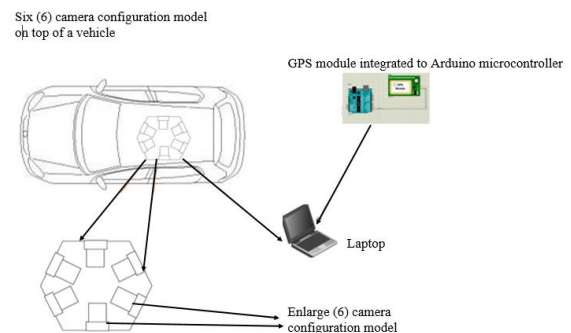


Figure 1. Architectural proposition for data collection and rendering using IBR technology

The data collection was conducted utilising a vehicle that is mounted with six cameras on its roof with the processing of the algorithms simulated in Blender3D for both image compression and image rendering. The camera configuration model consists of six cameras that are placed in a hexagon formation at a 60° angle between the cameras.

All six cameras are connected to a laptop that is used for image storage, data processing, and data computation. Each image is Geotagged using a Global Positioning System (GPS) module that is attached to the Arduino microcontroller.

The process is started by allowing the cameras to capture individual images at a 3ms switched interval between the cameras. The switching duration was selected to allow for proper transition between the cameras taking into account the number and size of the images. The switching algorithm is processed from Arduino Microcontroller as follows:

$$T = \frac{A}{S} \quad (1)$$

Where: A = data transfer speed (1.5 MB) raw data,

S = speed (480 MB/s) USB 2.0 speed

Following the image capture phase, the captured images are downloaded and manually placed into a single folder based on their timestamp.

The reason for obtaining GPS coordinates is to ensure that the images can be merged to obtain a panoramic view with approximately the same timestamp and location.

Other non-geometric methods that utilises computer vision functions, such as plenoptic function are utilised for allowing the intensity of light rays to pass through the camera centre at every location (L_x, L_y, L_z) and at every possible angle (θ, ϕ) for every wavelength λ , at every time t were tested [87]. In this paper, two algorithms were tested namely; Lossy and Lossless algorithms. However, it was noted that the Lossy compression algorithm was the best algorithm for utilisation in this research based on the reduced image size and less reduction of the image quality as compared to Lossless compression that can still do the same but with the image sizes being a concern.

The plenoptic function is expressed as follows:

$$P_7 = P(L_x, L_y, L_z, \phi, \theta, \lambda, t) \quad (2)$$

The images are projected from the laptop, and keyboard inputs are being used to navigate through the rendered scene. Furthermore, the GPS information such as longitude, latitude, and elevation from the scene is written to a text file, which is stored in a specified directory.

The cylindrical panorama was also utilised due to its ease-to-build method for the single unit camera.

A. System apparatus setup

The following components were selected as indicated in Figure 2 as the primary apparatus for this research. It is important to note that the development of the physical model was constructed before any process can take place.

This task needed to take place for the system component's functionality to be tested in a stationary environment before they can be placed on the test vehicle.

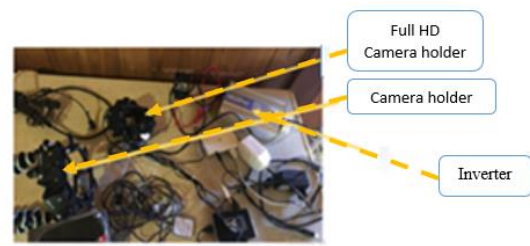


Figure 2. System components setup

The six full HD action cameras were selected and utilised to capture quality images that do not require any customisation which might delete some important data from the image, such as image descriptor/GPS coordinates.

Blender3D was selected and utilised due to its ability to convert and remodel files other than the .bli files, and also because it has the capabilities for code re-use. Furthermore, Blender3D was compared against Cinema4D as indicated in Table I as follows:

TABLE I. CINEMA4D COMPARISON TO BLENDER3D

	Cinema 4D	Blender3D
Availability	Paid, R700 – R1 200 per month	Free
Source	Closed	Open
Applications	Animation	Animation
	Rendering	Rendering
	Texturing	Texturing
Learning curve	Easy to learn	3D printing
		A hard learning curve at the beginning
User interface	User-friendly	No such intuitive

Table I depicts the reasons for the selection of Blender3D over Cinema4D. There are other softwares available; however, for this paper, it was deemed necessary to evaluate only these two softwares due to resource constraints.

Furthermore, the system design model is based on the initial model designed on Microsoft® Visio Professional 2016 as indicated in Figure 3.

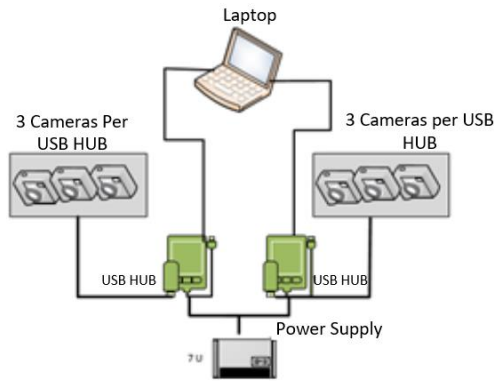


Figure 3. Camera configuration setup

The camera configuration setup outlined in Figure 3 depicts the system connections of the two hubs, which are connected to an ACER laptop that processes the incoming data from the cameras. There are six cameras, which are divided into two sectors/groups of three cameras per hub.

Due to the much-required processing power, the laptop specifications had to be modified for rendering and processing to take place without any interruptions. Additionally, an external battery with a 1200mAh capacity was purchased and used to power the two hubs and act as an extra power supply to the laptop due to the high amount of power needed to keep the system active.

Furthermore, the system test duration was calculated as indicated in (3) as follows:

$$\begin{aligned}
 B_d(A) &= \frac{C_c * C_{bv} * N_{bs}}{L_c} \quad (3) \\
 &= \frac{0.9Ah * 3.8V * 3}{11.4W} \\
 &= 0.9hr \\
 &= \underline{1 \text{ hrs testing}}
 \end{aligned}$$

Where:

$B_d(A/B)$ = battery duration for Hub (A or B); C_c = camera battery capacity; C_{bv} = camera battery voltage capacity; N_{bs} = number of batteries connected in series; L_c = load connected in watts

B. Image compression

The Lossless compression algorithm is performed for the redundant processing of image information. Additionally, the image dataset is simulated without noise for better performance verification.

C. Image calibration

Other camera information including but not limited to intrinsic and extrinsic camera properties are important during the camera calibration. However, they are deemed to be camera-specific.

Moreover, the information about the focal length (f_x, f_y) and optical center (C_x, C_y) is critical. The focal length and optical center are then used to create the camera matrix, which is used to remove the distortion due to the lenses, which are camera-specific.

The camera matrix is then calculated utilising the focal length and the optical centers for the reuse of images that are captured utilising a specific camera model as follows:

$$C_m = \begin{bmatrix} f_x & 0 & C_x & 0 & f_y & C_y & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Where: C_m = camera matrix, (C_x, C_y) = optical centres, (f_x, f_y) = focal length

The system setup approach makes use of the chess pattern for the camera calibration setup. Some calibration methods in the literature rely on 3D objects.

The camera calibration process is as follows:

- capture 20 chessboard images from different poses;
- find the chessboard corners;
- find the intrinsic matrix, distortion coefficients, rotation vectors, and the translation vector; and
- store the .xml file.

Following the process completion, OpenCV for the Python library is utilised to compute the results stored in the .xml file.

The black-and-white squared pattern match-finding is outlined as indicated in Figure 4.



Figure 4. Black-and-white test match on a chessboard

Figure 4 depicts the black-and-white match-finding test on a chessboard utilising an intrinsic

matrix for the reduction of image biasing. This model was deemed less complicated in matching the images utilising the Kd-tree approach. However, it was noted that the CasHash based approach can have a faster computational time as compared to Kd-tree-based image matching, depending on the number and size of the datasets. However, the Kd-tree-based approach was used since this project was based on few datasets.

D. Application of Structure from Motion Model

The basic operation of the Structure from Motion Model in this research follows the following pipeline:

- Detection of 2D features on every image;



Figure 5. Detecting the image feature using SIFT algorithm

Figure 5 depicts the captured image scene with the application of feature detection utilising the SIFT feature detection algorithms. This is obtained by creating an orientation histogram with 36 bins that cover 360° of the captured image. As a result, this creates the key points with the same location and scale but with different directions.

Table II depicts the SIFT feature detection algorithm that is utilised for the detection of the image scene outlined in Figure 5. These detected image features are detected from a .JPEG image format.

TABLE II. SIFT ALGORITHM FOR FEATURE DETECTION

Algorithm: *sift.detect()*

Function: *Begin()*

Import cv2

Import numpy as np

img = cv2.imread('cam1Image.JPG')

gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

sift = cv2.SIFT()

kP = sift.detect(gray, NONE)

img = cv2.drawKeypoints(gray,kP)

cv2.imwrite('sift_keypoints.jpg', img)

END

The use of the *sift.detect()* function is to find the keypoints, which are outlined in Figure 6. Furthermore, the *cv2.drawKeyPoints()* function is utilised to draw the small circles concurrently with the *sift.detect()* function. During this process, it was noted that the keypoint structure has many features for example the (X, Y) coordinates, size of neighbours, and keypoints strength.

- Matching of the 2D points between images:

Upon feature identification, the key points are matched by applying the Fast Library Approximate Nearest Neighbour (FLANN) as indicated in Figure 6.

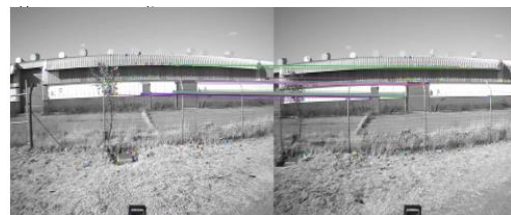


Figure 6. Matching of 2D points using the FLANN library

During the image matching process, the flag is passed using the match flag function. The image match function in the context of this application was used with reference to the chessboard intrinsic matrix and was computed after the track construction was obtained as follows:

detIMG=cv2.drawKeyPoints(gray,kP,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

cv2.imwrite('sift_keypoints.jpg', detIMG)

Under normal circumstances mainly utilising exhaustive matching, the computation is usually time-consuming and complex. Hence, the need to utilise the FLANN algorithm in this application. The FLANN algorithm utilised a tree-based approach by storing the image datasets within efficient data structures and utilising the Kd-tree approach. The Kd-tree approach was selected based on [88] [89].

- Construction of the tracks for the matches:

In track construction, the essential and fundamental matrices of a camera are computed. These matrices specify the camera motion in terms of rotational and translational components. This function is called in OpenCV to find the fundamental camera matrix as indicated:

```
cv2.findFundamentalMat(self.match_pts1,self
.match_pts2, cv2.FM_RANSAC, 0.1, 0.99).
```

For essential matrix, the solve for structure library is called.

- Solving of the Structure from Motion from 2D tracks.

At this stage, the triangulation is performed for determining the point in a 3D space given its projection onto two or more images [15]. In a direct linear transformation, P3P, the algorithm is used for triangulation.

- Refine the SFM model using bundle adjustment algorithm;

At this stage, the Bundle Adjustment (BA) is performed. However, the problem relating to the BA is the simultaneous refining of the 3D coordinates describing the scene geometry. The parameters of the relative motion and the optical characteristics of the camera (s) are employed to attain the images. However, in this paper, the CERES algorithm is utilised.

I. Results

The technique of combining Street View imagery with other known image datasets makes the experience of using panoramic view mainly in the virtual world even richer. The use of Street View is essential as it is used as a tool to ease the livelihood for directions while driving. In the process of acquiring such results, the feature detection and matching utilising the Structure From Motion (SFM) technique are important.

Another element outlined in Figure 7 depicts the point of cloud where all the feature tracks are visible in the image. The incremental SFM points are then used in the reconstruction of the scene and result in the formulation of the points of cloud. The output results are derived based on the camera position.

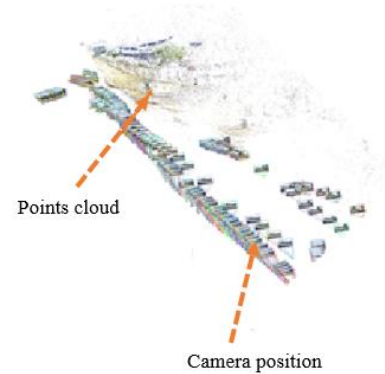


Figure 7. Point cloud and camera position reconstruction

Figure 8 depicts the dense geometry reconstruction of the scene. This is achieved by performing the multi-view stereo algorithm. The application of the multi-view stereo produces the depth maps and dense point cloud and with this, the algorithm can perform the map recovery. Once this process is complete, a mesh of a scene is reconstructed by fusing depth maps and dense cloud as indicated in Figure 8.



Figure 8. Denser point of cloud with Multiview stereo

Following the creation of the mesh output, the texture was then generated by taking models, images, and the camera position (this was achieved using the GPS coordinates). Furthermore, meshroom function selecting 8192 texture slides was unwrapped.

The depth map restoration and colour images were paired up to create the application of texture to the 3D mesh model and warp to the new viewpoint. Furthermore, the depth map model was extracted from the 3D mesh, and the shortcomings were observed to be the delay since the mesh update duration is long, and as a result it affects the depth map extraction.

The final output that is represented in Figure 9 depicts the rendered view of the panoramic street views. The depicted images are shown in a rendered street view from a top view perspective. This is

achieved and projected from a horizontal street view in an omnidirectional manner.

These results are outlined to provide the freedom of movement. The views that were not captured by the camera but through rendering the uncaptured scene can be viewed as indicated in Figure 9.



Figure 9. Rendered bidirectional 3D scenery view

The panoramic street view can be viewed in multiple viewing angles as depicted in Figure 9. Furthermore, the depicted image outlines a rendered street view from a top view perspective with the bidirectional capability of scenes that were originally not captured.

IV. Conclusion

The result and outcome of this paper demonstrate the application of the use of Street View and image rendering in a real-life environment based on the hexagon camera configuration. Furthermore, it was established that the IBR technique can easily allow for faster processing of multiple datasets based on the Kd-tree approach. Additionally, these techniques can allow for the movement and view of the uncaptured scene without the need for any extra application or tools.

References

- [1] W. Wabbeh, S. Nebiker and G. Fangi, "Combining public domain and professional imagery for the accurate and dense 3D reconstruction of the destroyed bel temple in Palmyra," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 88, no. 81, p. 5, 2016.
- [2] N. Bruno and R. Roncella, "Accuracy assessment of 3D models generated from Google Street View imagery," in *8th Intl. Workshop 3D-ARCH "3D Virtual Reconstruction and Visualization of Complex Architectures"*, Bergamo, Italy, 2019.
- [3] J. M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y. H. Jen, E. Dunn, B. Clipp, S. Lazebnik and M. Pollefeys, "Building Rome on a cloudless day," pp. 368-381, 2010.
- [4] X. Li and C. Ratti, "Mapping the spatial distribution of shade provision of street tree in Boston using Google Street View panoramas," pp. 109-119, 25 February 2018.
- [5] P. E. Debevec, C. J. Taylor and J. Malik, "Modeling and rendering architecture from photos: A hybrid geometry- and image-based approach," in *ACM SIGGRAPH*, pp. 11-20, 1996.
- [6] I. L. Lenkoe and B. Kotze, "Enhancing Spatial Image Datasets For Utilisation In A Simulation for Smart City Transport Navigation," in *Smart 2021: The Tenth International Conference on Smart Cities, Systems, Device and Technologies*, Valencia, Spain, 2021.
- [7] Y. Mao, G. Cheung, A. Ortega and Y. Ji, "Expansion hole filling in depth-image-based rendering using graph-based interpolation," pp. 1859-1863, 2013.
- [8] S. Shi, K. Nahrstedt and R. Campbell, "A real-time remote rendering system for interactive mobile graphics," *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, vol. 8, no. 3, pp. 1-20, 2012.
- [9] A. I. Audu, "Camera positioning for 3D panoramic image rendering," London, 2015.
- [10] R. E. Klosterman, "The What if? Collaborative planning support system.," *Environment and Planning B: Planning and Design*, pp. 393-408, 1999.
- [11] D. Z. Sui, "GIS-based urban modeling: practices, problem and prospects.," *International Journal of Geographical Information Science*, no. 12, pp. 651-671, 1998.
- [12] E. H. Adelson and J. Bergen, "The plenoptic function and elements of early vision," in *Computational Model of Visual Processing*, in MIT Press, Cambridge, MA, 1991.
- [13] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz and R. Szeliski, "Building Rome in a day," vol. 54, no. 10, pp. 1-8, October 2009.
- [14] D. Crandall, A. Owens, N. Snavely and D. P. Huttenlocher, "Discrete continuous optimisation for large-scale structure from motion," pp. 3001-3008, July 2011.

Annexure D: Enhancing Spatial Image Datasets For Utilisation in A Simulator for Smart City Transport Navigation

Enhancing spatial image datasets for utilisation in a simulator for smart city transport navigation

Lepekola I. Lenkoe

Department of Electrical, Electronic and
Computer Engineering
Central University of Technology, Free State
Bloemfontein, South Africa
email: pexonl3@gmail.com

Ben Kotze

Department of Electrical, Electronic and
Computer Engineering
Central University of Technology, Free State
Bloemfontein, South Africa
email: bkotze@cut.ac.za

Abstract – The introduction of Google Street View has brought to the surface a method for roof-mounted mobile cameras on vehicles. This method is regarded as one of the highly known and adopted methodologies for capturing street-level images. This article contributes to the development and implementation of Image-Based Rendering techniques by presenting a technique that makes use of hexagon-based camera configuration for image capturing. Upon the image capturing, each segmented image is stored in a specific folder relative to the camera number (i.e camera 1 = folder 1). Subsequently, the optimal image rendering process of each image blending takes place inside Blender3D software where image datasets are rendered for utilisation in the simulator. Utilising the Structure from Motion algorithm, the dense point image, and its features, match detection is obtained. The article further contributed to the results process that allows for free movement within the 3D-rendered scene by permitting for back and forward movement as compared to a slide show that only allows for forwarding motion.

Keywords – *Image-Based Rendering; Blender3D, Simulation, Datasets; Google Street View; Smart City*

I. Introduction

Over the years different techniques have been proposed for image data collection and image rendering. However, in the past few years, the Image-Based Rendering (IBR) technique has gained much attention mainly in image processing, computer vision, and the computer graphics community. In addition to IBR's interest in communities and spatial knowledge, which is regarded as an essential subject that makes use of geospatial statistics such as geoscience, geography has shown growth with regards to multi-functional ecosystems.

Street View has debilitated previous restrictions on the availability of data sources for evaluating streets [1]-[2]. Furthermore, Model-Based Rendering (MBR) is classified as an easy method for reconstructing virtual view from any arbitrary viewpoint by using explicit 3D geometric and model and texture information about the scene, while IBR is a method that constructs virtual view by using several images captured beforehand [3].

The captured images can provide valuable information about the incident, e.g. location. The location has the exact Global Positioning System (GPS) coordinates, which can also be an estimation of the location. Figure 1 presents the graphical representation of inquiring a query image to the reference database to find the match between a stored image and the query image [4].

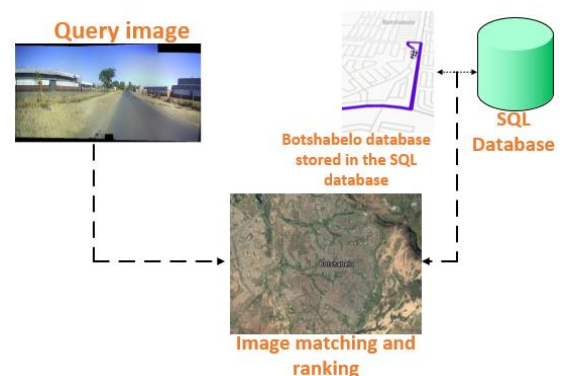


Figure 1: Street-view to overhead view image matching

II. Problem statement

The 3D data acquisition process provides the probe position and orientation that remain in static order to produce accurate datasets. A single 3D

capacity is not able to support the translational motion of the simulated probes, thus the need to develop a methodology for recording and capturing single 2D images and amalgamating the images into multiple 3D images within a single unit.

Furthermore, the issue of emulating street-view images for multiple image transitions for application in geolocalisation and utilisation in a simulator needs to be investigated.

III. Aims and objectives

This study seeks to develop a 3D-rendered model from 2D captured images.

I. The objective of this study is therefore to:

- Identify datasets with capabilities such as frame position, frame elevation, and frame indexing.
- Incorporate the system into the simulation system in real-time for increasing the reality of the simulation system in different geographical locations.

J. Original contributions of this research article

This research article has produced the following contributions in furthering the knowledge contribution in the field of computer vision as follows:

- six degrees of freedom where the user can move in any direction as opposed to the use of a single slide show that allows for one-directional movement in a street view scene.
- The ability of the application to use multiple cameras between three to six inputs as opposed to the use of single omnidirectional camera feedback and still obtain the same output rendered panoramic and simulated results.

IV. Literature review

Quintessentially, massive amounts of image collections are presented as slideshows, which are arguably the practical way, but with the current technological advancement these methodological approaches are deemed not engaging.

The change in scenery due to technological improvements has led to a wide study pool which also cites the research conducted by Sivic. Sivic *et*

al., [91] highlight the connection of clustering visually similar images together to create a virtual space in which the users are free to change position from one image to another. This virtual space modelling can be obtained by utilising intuitive 3D control objects such as move left/right, zoom in/out and rotate.

Sivic further outlines that the displayed images in a correct geometric and photometric alignment concerning the current photo result in a smooth transition between multiple images. In addition, Kopf *et al.* [92] present a method of combining images in the Street View system by stitching the image side views. This approach means that it is as if one is standing on the street and looking in either the left or right direction of a certain street together. This generates a long street slide for users to quickly browse whether a street is feasible.

Despite the excellent way of viewing the side scene of a certain street from Kopf's methodology, the practicality of that method is not always the case while driving or walking.

Kopf *et al.* [93] present street slide methodology which combines the nature of bubbles provided by perspective stripe panoramas. Kopf further presents integrated annotations and a mini-map within the user interface to provide geographic information as well as the additional affordances for navigation.

However, Kopf's work relates to Gortler *et al.* [8] due to their classic approaches of image-based rendering such as Lumigraph. Kopf's work is further supported by Agarwala *et al.* [9], emphasising the utilising of the correlation alignment techniques for aligning adjacent vertical strips instead of modeling the full 3D geometric proxies.

Subsequent to these approaches, rendering displacement in maps requires the surface to be adaptively retessellated [10].

The synthetic environments for the extraction of the depth information during the rendering process. The geometric construction relates to both implicit and explicit construction [94]. This is as a result of the view dependency, which means that the explicit geometric rendering relies much on the known approximate environment [12]. The use of the explicit rendering becomes much tricky in an informal environment/settlement due to the tiring exercise of data collection, which is skewed since residents can build on the road and mountains.

V. Methodology

The simulator model design is developed for a driver or person riding a bicycle inside the simulator following a track in either forward or reverse direction. With this said, the initial thought design was to develop the Spatial Image Datasets (SID) based on the nonagon (9) camera configuration model.

However, with the current technological capabilities, an omnidirectional camera could have been utilised to conduct this activity. The reason for not utilising the omnidirectional camera is because the rendering construction specifically for this research article requires individual camera feeds as opposed to one 360° feed.

The system development was then initiated based on the model layout as indicated in Figure 2. Figure 2 depicts the technical approach for the design and development of the simulator utilising the Hexagon Camera Configuration Model as follows:

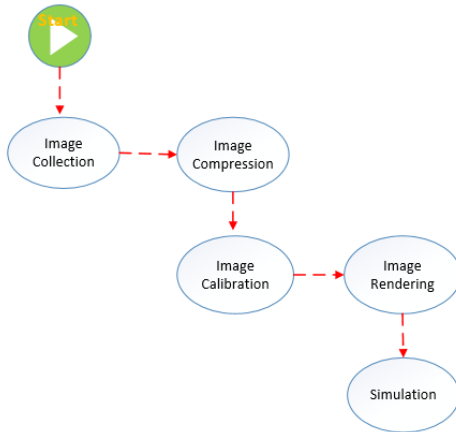


Figure 2: System model layout

A. Image capturing and collection

The image capture section consists of the following apparatus: camera, images, and GPS coordinates. The six (6) Mounted Camera Configuration Model is used for image capturing while the vehicle is in motion, and each image is treated as a frame. During the image capturing process, each camera image dataset is stored in its specific folder, i.e. camera 1 = folder 1.

B. Image compression

The scene depiction utilising multiple depth images in a dataset format is compressed. The image samples are then captured and obtained from the camera capture by delaying the camera switching

algorithm between multiple cameras by 3ms (the delay period was based on the trial and error test conducted between 1ms – 5ms switching).

Furthermore, the Lossy compression algorithm is performed for the redundant processing of image information. Additionally, the image dataset is simulated without noise for better performance verification as outlined in Figure 3 utilising the Lossy compression algorithm.



Figure 3: image compression on a JPE file

Figure 3 depicts the image types for the captured images, subsequently showing the timestamp and the GPS coordinates for the image dataset created. Additionally, the image compression framework is used to obtain the results output outlined in Figure 4.

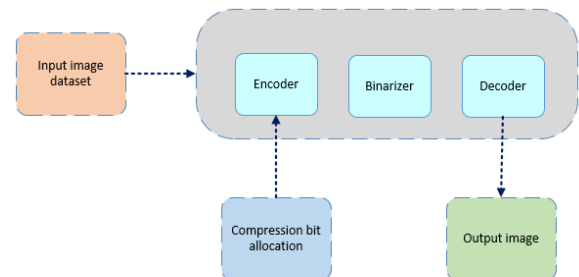


Figure 4: Compression structure

Figure 4 depicts the compression structure that utilised the input image datasets for the network training sample. This is obtained by specifically setting the image dataset for image recognition where the compressed image datasets are compared against the raw image datasets.

Additionally, the image compression bit allocation is then used to calculate the compression alterations. However, the alterations depend on the size of the dataset. Notably, each image in the dataset is compressed individually to retain the image quality. However, this is deemed as a tedious process, especially for huge datasets.

C. Image calibration

The data collection and image compression process are determined by the reduction in the image size whilst keeping the image resolutions intact. The image calibration model utilises the pinhole camera model that introduces some image distortions. These image distortions that are seen in this process are classified as radical image distortion. The radical image distortion was calculated as follows:

$$X_{distortion} = x(1 + k_1r^2 + k_2r^2 + k_3r^6) \quad (1)$$

$$Y_{distortion} = y(1 + k_1r^2 + k_2r^2 + k_3r^6) \quad (2)$$

Where: x = original; x location on the imager

y = original y location on the imager

k = radical distortion coefficient

r = radical distortion form Taylor series

The system setup approach makes use of the chess pattern for camera calibration setup. Some calibration methods in the literature rely on 3D objects. However, through the tests conducted, the flat chessboard pattern approach is deemed appropriate for this research study due to the method been less complex and easily understood even by non-technical individuals.

The camera calibration process is as follows:

- capture 20 chessboard images from different poses;
- find the chessboard corners;
- find the intrinsic matrix, distortion coefficients, rotation vectors, and the translation vector; and
- store the .xml file.

Following the process completion, OpenCV for the Python library is utilised to compute the results from the .xml file. This, therefore, allows for the reuse of the code for multiple cameras, which is relevant for this study which uses a Hexagon Camera Configuration Model with a rotational image capturing technique.

The black-and-white squared pattern match-finding is outlined as indicated in Figure 5.



Figure 5: Black-and-white test match on a chessboard

A. Image rendering procedure

The image rendering technique in the context of this research article focuses on the known camera parameters and undistorted images for the rendering of the scenes. These images are reconstructed, and the texture is applied to their structure before rendering simulation can be executed.

B. Structure from Motion

The basic operation of the Structure from Motion Model in this research follows the Detection of 2D features on every image. In this step, a 2D feature is detected using the Scale-Invariant Feature Transform (SIFT) algorithm as indicated in Figure 6. Figure 6 depicts the original image captured with one of the mounted cameras from the Hexagon Configuration Model.



Figure 6: Detecting the image feature using SIFT

C. Multiview stereo

The camera parameters are captured, and the patch-based stereo and semi-global matching are used to generate point tracks, depth-maps as well as the points cloud. Upon successful generation of these variables, a mesh of the scene is created as indicated in Figure 7. Finally, all the refined depth maps are merged to get the final reconstruction.

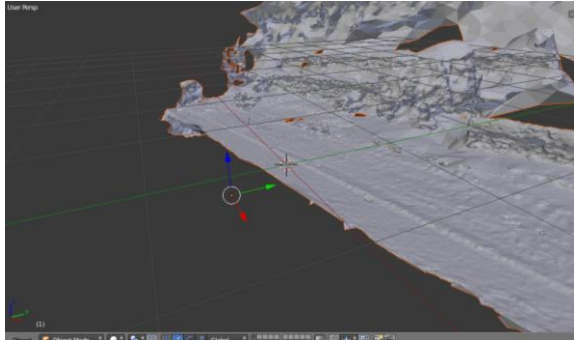


Figure 7: Mesh output from the Multiview stereo

The multi-view stereo algorithm is further used as a Semi Global Matching algorithm (SGM), where it consists of calculation, aggregated costs, disparity computation, and the extension for multi-baseline matching.

D. Texturing

Patches are formed onto the faces of the model, and the texture patches colours are adjusted. This is achieved by adjusting colour between adjacent patches. This results in seamless texture across the model.

E. Data simulation

The model with texture consists of the path/road and environment (trees and buildings). Lighting, camera, and collides are added. Lighting is added to illuminate the model to simulate the light from the sun.

The movement of the camera simulates a vehicle moving through the path/road created with the model. The movement gets its inputs from the keyboard. Colliders are Blender3D objects that provide physics attributes to the model, and they are added to prevent the user from moving beyond the required space within the simulator.

VI. Results

The results outlined in this section depict the image rendering framework for the 364 .JPEG images that were captured on each camera at a total dataset worth 2184 JPEG images at a high resolution of 1280X720 pixels at a total size of 39.8MB. The system required a dynamic scene with six. A 2D arc was arranged at a spanning of approximately 600cm apart from each other.

Additionally, each camera frame comprised of 364 JPEG images captured from the real scene, and only then the process of image matching and texturing was applied using “depth maps resulting” with the output textured PNG image of 25.2 MB of 8192X9192 pixels resolution.

A. Rendering simulation outcomes



Figure 8: Denser point of cloud with Multiview stereo

Following the creation of the mesh output, the texture was then generated by taking models, images, and the camera position (this was achieved through the use of the GPS coordinates). This was accomplished in the meshroom function by selecting 8192 texture slides and by unwrapping this method as indicated in Figure 8.

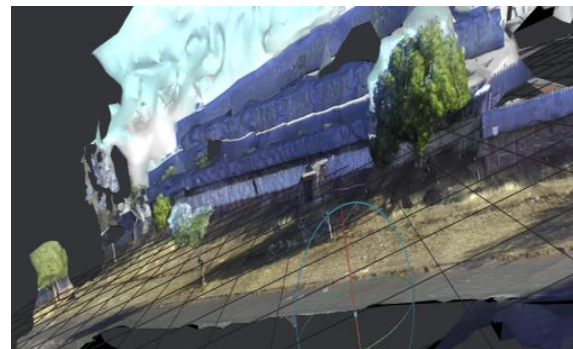


Figure 9: Generated texture

Figure 9 depicts the depth map restoration, and colour images were paired up to create the

application of texture to the 3D mesh model and warp to the new viewpoint. Furthermore, the extraction of the depth map model from the 3D mesh and the shortcomings were observed to be the delay by which the mesh update duration is long and as a result, affects the depth map extraction.

The final output that is represented in Figure 10 depicts the rendered view of the panoramic street views. The panoramic street view can be viewed in multiple viewing angles as indicated in Figure 10.

This is achieved and projected from a horizontal street view in an omnidirectional manner. These results are outlined to provide the freedom of movement, and the views which were not captured by the camera but through rendering the uncaptured scene can be viewed.

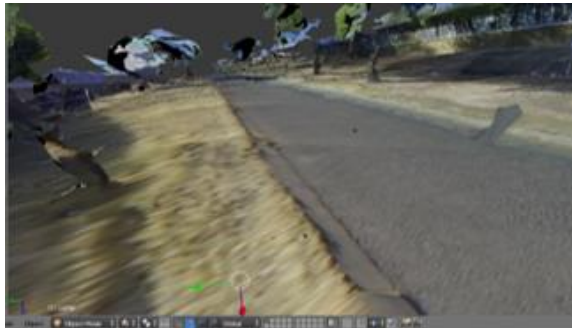


Figure 10: Rendered image horizontal view

VII. Conclusion

The feature detection and matching technique was observed as the best technique in detecting and matching the images from multiple image datasets. As a result, the use of the image-based rendering technique utilising the Hexagon Camera Configuration Model was proposed as an ideal method in this study.

The objectives looking into the integration of IBR and the simulator were achieved as indicated:

- The incorporation of the system into the simulation system in real time for increasing the reality of the simulation system in different geographical locations.
- To simulate a rendering technique for improvement of visual, spatial, and quality of the panoramic images for location identification.

Acknowledgment

Acknowledgment for financial assistance and academic assistance from the Central University of

Technology, Free-State (CUT) and Mr T.G Kukuni for his academic expertise and unending support.

References

- [1] L. Yin, Q. Cheng, Z. Wandy and Z. Shao, "Big data for pedestrian volume: Exploring the Sasol," Exploring the use of Google Street View images for pedestrian counts, no. 63, pp. 337-345, 2015.
- [2] N. Runge, P. Samsonov, D. Degraen and J. Schoning, "No more autobahn: Scenic route generation using Googles Street View," In Proceedings of the I, 7-10 March 2016.
- [3] R. Sato, S. Ono, H. Kawasaki and K. Ikeuchi, "Photo-Realistic Driving simulator using Eigen Texture and Real-Time Restoration Techniques by GPU," vol. 6, no. 2, p. 87, 2 December 2008.
- [4] N. N. Vo and J. Hays, "Localizing and Orienting Street View Using Overhead Imagery," p. 2, 2016.
- [5] J. Sivic, B. Kaneva, A. Torralba, S. Avidan and W. T. Freeman, "Creating and Exploring a Large Photorealistic Virtual Space," Proc. IEEE Workshop on Internet Vision, 2008.
- [6] J. Kopf, B. Chen, R. Szeliski and M. Cohen, "Street slide: Browsing Street Level Imagery," ACM Trans.Graphics, vol. 29, no. 4, 2010.
- [7] J. Kopf, B. Chen, R. Szeliski and M. Cohen, "Street Slide: Browsing Street Level Imagery," ACM Transaction on Graphics, vol. 29, no. 4, July 2010.
- [8] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, "The Lumigraph. In Computer Graphics Proceedings, Annual Conference Series," in ACM SIGGRAPH, Proc. SIGGRAPH 96, New Orleans, 1996.
- [9] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin and R. Szeliski, "Photographing long scenes with multi-viewpoint panoramas," ACM Transactions on Graphics 25, pp. 853-861, 3 August 2006.
- [10] K. Moule and M. McCool, "Efficient bounded adaptive tessellation of displacement maps," In proc. of GI, pp. 171-180, 2002.
- [11] H. Y. Shum and S. B. Kang, "A review of image-based rendering techniques," p. 1.
- [12] P. E. Debevec, C. J. Taylor and J. Malik, "Modeling and Rendering Architecture from Photos: A Hybrid geometry- and image-based approach," In ACM SIGGRAPH, pp. 11-20, 1996.

Annexure E: Modelling Input



Annexure F: Feature Detection



Annexure G: Rendered output



Annexure H: Image Calibration Code

```
import cv2
import numpy as np
import glob
from tqdm import tqdm
import PIL.ExifTags
import PIL.Image

chessboard_size = (9,6)
obj_points = []
img_points = []

objp = np.zeros((np.prod(chessboard_size),3),dtype=np.float32)
objp[:, :2] = np.mgrid[0:chessboard_size[0],
                       0:chessboard_size[1]].T.reshape(-1,2)
calibration_paths = glob.glob('calibration_images/Front_center/*')#Iterate
over images to find intrinsic matrix
for image_path in tqdm(calibration_paths):#Load image
    image = cv2.imread(image_path)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    print("Image loaded, Analyzing...")
    #find chessboard corners
    ret, corners = cv2.findChessboardCorners(gray_image, chessboard_size, None)
    if ret == True:
        print("Chessboard detected!")
        print(image_path)
        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
        cv2.cornerSubPix(gray_image, corners, (5,5), (-1,-1), criteria)
        obj_points.append(objp)
        img_points.append(corners)
        # Calibrate camera
    ret, K, dist, rvecs, tvecs = cv2.calibrateCamera(obj_points, img_points,
    gray_image.shape[::-1], None, None)
    # Save parameters into numpy file
    np.save("camera_params/ret", ret)
    np.save("camera_params/K", K)
    np.save("camera_params/dist", dist)
    np.save("camera_params/rvecs", rvecs)
    np.save("camera_params/tvecs", tvecs)
    #Get exif data in order to get focal length.
    exif_img = PIL.Image.open(calibration_paths[0])
    exif_data = {
        PIL.ExifTags.TAGS[k]:v
        for k, v in exif_img.getexif().items()
        if k in PIL.ExifTags.TAGS}
    focal_length_exif = exif_data['FocalLength']
    focal_length = focal_length_exif[0]/focal_length_exif[1]
    np.save("./camera_params/FocalLength", focal_length)
```