INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

WeatherDepth: Monocular Depth Estimation in Adverse Weather Conditions with Adversarial Training

A dissertation by

Mr. Avishka Amunugama

Supervised by

Mr. Thiloshon Nagarajah

Submitted in partial fulfilment of the requirements for the BEng in Software Engineering degree at the University of Westminster.

Jul 2022

Declaration

I hereby certify that this dissertation and all associated components are my own work and that all ethics guidelines were strictly followed throughout the project's development. I further declare that this dissertation or any associated component has not been submitted before, nor is it being submitted for any other degree program.

Full Name : Avishka Amunugama
Registration No : 2018152 | W1714904

Signature : 

Date : 11th of July 2022

Abstract

Estimating the depth from a single RGB image is a challenging task that finds applications in various fields such as autonomous driving, robotics, 3D modelling, scene understanding, etc. Though there has been plenty of research in recent years on this field, of which many have produced outstanding results on daytime clear weather conditions, very few researches have been carried out that aim to solve monocular depth estimation in adverse weather conditions. The lack of research in adverse weather monocular depth estimation systems and the inability of the current state-of-the-art approaches that produce remarkable results in clear daytime conditions to produce accurate and consistent results when subjected to adverse weather conditions such as rain, snow or fog has been a significant setback in the autonomous driving industry and has been a key factor forcing most autonomous driving companies to still focus on expensive sensor-based approaches. This dissertation presents WeatherDepth, a novel robust monocular depth estimation approach that is capable of producing high-resolution accurate depth maps in adverse weather conditions such as rain, snow, and fog with the help of transfer learning and the CityscapeWeather dataset, which is an adverse weather depth estimation dataset based on the popular Cityscape dataset. The proposed approach, WeatherDepth, utilizes an adversarially trained autoencoder-based architecture. Experiments on the CityscapeWeather and vKITTI datasets demonstrate that our approach outperforms the state-of-the-art monocular depth estimation systems in generalization capabilities when subjected to adverse weather conditions.

Keywords - Monocular Depth Estimation, Generative Adversarial Network, Autoencoder, U-Net, Transfer Learning, Adverse Weather Conditions

Acknowledgement

I would like to express my sincere gratitude to Mr Thiloshan Nagarajah for supervising my project and for continuously providing support and guidance from the point I started working on the project till the very end, and for his invaluable feedback every step of the way, which helped greatly to shape the project to bring it to the standard that it is now. I would like to thank Dr Nihal Kodikara for his valuable feedback and advice on the project, which helped me decide on the project idea and scope it down to something doable within the limited time available. Also, I would like to thank Mr Dulaj Weerakoon for his valuable support and insights on the project domain, which was a significant eye-opener and extremely helpful in coming up with the approach proposed in this dissertation.

Finally, I would like to thank my family and friends for their support and encouragement throughout the project's development, which helped me immensely in completing the project.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
DNN	Deep Neural Network
AV	Autonomous Vehicle
AR	Augmented Reality
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
VAE	Variational AutoEncoder
GAN	Generative Adversarial Network
Lidar	Light detection and ranging
Radar	Radio detection and ranging
ADCU	Autonomous Driving Control Unit
ADAS	Advanced Driver-Assistance System

CHAPTER 01: INTRODUCTION

1.1. Chapter Overview

Autonomous driving depth perception has seen tremendous growth during the last few years. Remarkable results have been achieved using DNN-based approaches for monocular depth estimation. However, most of these approaches can only work well in the daytime. Few attempts have been made in the past to adapt monocular depth estimation to adverse weather conditions. Most of these attempts use additional sensors such as lidar and radar to aid the monocular depth estimation process in adverse weather conditions. The current state-of-the-art monocular depth estimation approaches mainly focus on clear daytime conditions, and they fail to maintain accuracy and consistency when subjected to adverse weather. The author of this research attempts to design, develop and evaluate a monocular depth estimation approach that can produce more accurate and consistent depth estimates in adverse weather conditions than the state-of-the-art without any additional sensor data.

1.2. Problem Domain

The autonomous driving industry has been out there for decades and has grown ever so slowly until recently. With the recent breakthroughs in deep learning, the growth in autonomous driving technologies has skyrocketed. Several large-scale tech companies are now actively involved in developing autonomous tech, and it has become an extremely competitive race to release the first fully autonomous vehicle to the public. However, unfortunately, we are still a long way off from achieving full autonomy. We still have many more advanced problems to solve to achieve full autonomy (Dia, 2021). One such problem that needs further improvement is depth perception (Adams, 2020). AVs should be able to accurately perceive the depth of objects in their surroundings, unaffected by the time of the day or weather conditions to be safe in public among pedestrians and other vehicles; hence, depth estimation is a crucial step in autonomous vehicles perception systems. The sections below give a detailed overview of how computer vision is used to estimate the depth through camera images and how well the cameras compete with other sensors.

1.2.1. Computer Vision And Depth Estimation

Computer Vision is a field of Artificial Intelligence that enables computers to represent the visual world (Computer Vision: What it is and why it matters, 2021). Measuring distance

relative to a camera using computer vision remains a challenging problem. It has become an area of interest in many fields of study, such asAR, 3D scene reconstruction and autonomous driving, in which depth is a crucial prerequisite to performing multiple tasks such as perception, navigation, and planning (Perception – Towards Data Science, 2021). The goal of depth estimation is to recover the lost depth dimension from 2-dimensional images to recreate a 3-dimensional representation of the scene in an image that helps understand its spatial structure. Two principal methodologies exist in extracting depth from images using computer vision.

1. Monocular depth estimation (using a single ordinary RGB image)
2. Stereo depth estimation (using a pair of RGB images)

1.2.2. Monocular Vs Stereo Depth Estimation

Since the 90s, the earliest computer vision-based depth estimation methodologies used stereo vision, which mimicked stereopsis mathematically. The stereo methods work through triangulation (Tan, 2021). The 3D position of points in an image is computed using the disparity map and the geometry of the stereo setting of the two images (Luca, 1998). Monocular depth estimation recently gained popularity with the development of DNNs, and means of extracting depth directly from an image were introduced. Monocular methods use neural networks to convert every pixel in the single input RGB image to distant estimates forming a depth map (Institute, 2021).

The main reason stereo depth estimation has become unpopular in recent years is that stereo depth estimation is heavily dependent on the calibration of the two cameras and its inherent limitations. Careful calibration and setup demand expert knowledge, often requiring special post-processing (Atapour-Abarghouei and Breckon, 2018). Stereo systems have inherent limitations such as depth inhomogeneity, missing depth (holes) in depth maps, computationally intensive and tend to fail depth estimation of texture-less surfaces where correspondences cannot be reliably found (Saxena, 2007).

On the other hand, monocular depth estimation suffers from far fewer limitations even though the estimation process is comparatively more complex. Several pieces of research, including (Eigen, Puhrsch and Fergus, 2014), demonstrated that CNNs and other DNNs could accurately estimate dense depth maps from single images better than stereo depth estimation techniques.

1.2.3. Cameras Over Other Sensors

Among the most commonly used types of sensors for AV perception, i.e. camera, lidar and radar, lidar has become the go-to sensor for depth estimation in the automotive industry and robotics. A large number of existing approaches to depth estimation rely on lidar, and this is because they are capable of producing very accurate 3D point clouds of their environment directly. Although lidar is still more accurate in in-depth estimation than cameras, cameras stand out as the most viable option for multiple reasons. Firstly, lidar sensors are too expensive to be used in commercial vehicles, incurring a hefty premium for autonomous driving hardware (Wang et al., 2020). Secondly, they are bulky and need to be mounted on top of vehicles for them to work correctly, which is a massive hit on car aesthetics and aerodynamics, making the cars look weird and inefficient.

Similar to cameras, lidar sensors are also affected by bad weather conditions such as rain, snow and fog. Radar sensors are much cheaper than lidar and unaffected by adverse weather conditions, but their measurements are sparse and much noisier than other sensors, making radar data extremely challenging to work with (Lin, Dai and Gool, 2020). Furthermore, over-reliance on a single sensor would be a safety risk if the sensor malfunctions, so having a viable secondary depth estimation approach to fall back on would be of great value (Wang et al., 2020).

Cameras stand out as the cheapest and the most available among the various sensors used for depth estimation. Cameras operate at a much higher frame rate, producing a much denser depth map, unlike the inherently limited radar signals.

Rating: H = High, M=Medium, L = Low	Camera	Radar	LiDAR	Autonomous Requirement
Object Detection	M	H	H	H
Classification	H	M	L	H
Close-Proximity Detection	M	H	L	H
Speed Detection	L	H	M	H
Lane Detection	H	L	L	H
Traffic Sign Recognition	H	L	L	H
Range	H (200m)	H (250m)	M (120m)	Full range
Work in Rain, Fog, Snow	L	H	M	H
Work in Low Light	L	H	H	H
Work in Bright Light	M	H	H	H
Size	Small	Small	Medium	Mix
Cost	$	$$	$$$$	Mix

Figure 1: Comparison of current capabilities of commonly used sensors in AV perception.

1.3. Problem Definition

With the rapid development in deep learning, a variety of DNNs such as CNNs (Garg et al., 2016), RNNs (Wang et al., 2019) and VAEs (Chakravarty, Narayanan and Roussel, 2019) have been used widely in depth estimation. With their excellent ability to recover pixel-level depth maps from a single image in an end-to-end manner based on deep learning, deep learning-based approaches have manifested monocular depth estimation approaches by producing remarkable results in terms of depth estimation accuracies (Xiaogang et al., 2020). Most research on monocular depth estimation has only focussed on clear daytime conditions. These approaches that have shown outstanding performance in daytime conditions fail miserably when tested in adverse weather conditions (Vankadari et al., 2020). The majority of the research carried out in the past to address depth estimation in adverse weather conditions has mainly focussed on sensor-based approaches and utilised expensive sensors such as lidar or radar or used combinations of all or several sensors in sensor-fusion-based approaches. Some tried to address each weather condition separately, which will finally have to be combined into a multi-modal approach. Such systems would be computationally intensive and comparatively slow in terms of inferencing times, thus making such approaches infeasible to be used in autonomous vehicles.

1.3.1. Problem Statement

The lack of research in adverse weather depth estimation systems and the inability of the current state-of-the-art approaches that produce remarkable results in clear daytime conditions to produce accurate and consistent results when subjected to adverse weather conditions such as rain, snow, or fog has been a significant setback in the autonomous driving industry and has been a key factor forcing autonomous driving companies to still focus on expensive sensor-based approaches. Therefore introducing a monocular depth estimation approach that can produce consistent and accurate depth estimates in adverse weather conditions will be of utmost importance as this would help bring the attention of autonomous communities to camera-based approaches for depth estimation over sensor-based approaches.

1.4. Research Motivation

Estimating the depth of a scene from a single RGB image is a challenging problem that finds applications in a wide range of fields such as autonomous driving, robotics, 3D modelling, scene understanding, etc. (Bhoi, 2019). Though there has been plenty of research in recent

years on this field, of which many have produced outstanding results on daytime clear weather conditions, very few researches have been carried out that aim to solve depth estimation in adverse weather conditions. Testing out these state-of-art approaches that produce phenomenal results in clear daytime conditions in adverse weather conditions such as rain, snow, fog, etc. have produced heavily inaccurate and inconsistent results (Vankadari et al., 2020) hence proving the fact that these are unable to cope up with adverse weather conditions. This inability of these state-of-the-art methods has been a significant factor for the autonomous driving industry to rely mostly on expensive sensor-based rather than image-based approaches.

1.5. Existing Work

1.5.1. Monocular Depth Estimation Approaches

Monocular depth estimation has been a topic of broad interest among autonomous driving researchers in the last few years and has seen tremendous growth as never before. The table below summarises the most popular and well-performing monocular depth estimation approaches.

Citation	Brief Description	Limitations	Improvement
(Ranftl et al., 2020) MiDAS	Proposed a novel depth estimation approach with a vision transformer-based backbone instead of a convolutional network which most depth estimation approaches usually use. The approach provided finer-grained and more globally coherent depth predictions than fully-convolutional networks.	Even though the approach produces remarkable results in the daytime, under adverse weather conditions, the depth predictions lack consistency, with depth holes appearing randomly and overall depth maps tend to look smudged as the network is only built targeting daytime conditions.	At the time of this research was carried out, this approach was the best performing monocular depth estimation approach among several benchmarks and produced far superior depth estimates with over a 28% improvement in relative performance in terms of accuracy and inference time.

(Alhashim and Wonka, 2019) DenseDepth	Proposed a convolutional neural network-based approach which predicts depth at high-resolution with the help of transfer learning. The encoder-decoder architecture proposed with the help of the high-performing Densenet pretrained network outperforms the state-of-the-art in prediction accuracies and object boundary detection.	Similar to the previous approach, the network produces inaccurate and inconsistent results when tested with adverse weather images.	The transfer learning-based network architecture produced high accuracy and higher quality depth maps using much fewer parameters and training iterations.

Table 1: Overview of the existing monocular depth estimation methods

1.5.2. Adverse Weather Depth Estimation Approaches

The following sections provide an overview of the current state-of-the-art monocular depth estimation that had previously attempted adverse weather depth estimation.

1.5.2.1 Multiple Sensor Fusion Approaches

The table below provides a brief overview of the current systems that make use of a multitude of sensors such as lidar, radar and camera.

Citation	Brief Description	Limitations	Improvement	Sensors Used
(Gasperini et al., 2021)	Proposed a set of techniques to integrate radar into a self-supervised monocular depth	Use of additional sensors. Uses radar data during training and inference time in	It is able to produce accurate depth estimates in all weather and illumination	Monocular Camera and sparse radar sensor

	estimation framework.	addition to images.	conditions, and radar signals are unaffected by weather.	
(Siddiqui, Vierling and Berns, 2020)	Proposed an extension to (Alhashim and Wonka, 2019) by fusing RGB images with sparse radar signals recorded on various weather conditions.	Uses additional sensors. The approach requires radar signals during training and inference to produce desirable results.	By fusing RGB images with sparse radar, the authors revealed that slight improvements were made over the original approach. Significant improvements were seen on top-view and zoomed images compared to the original approach.	Monocular Camera and sparse radar sensor

Table 2: Overview of the multiple sensor based adverse weather depth estimation methods

1.5.2.2 Monocular Approaches

To the best of our knowledge, only two pieces of work were published that attempted to address monocular depth estimation in adverse weather conditions at the time of this research. A brief overview of these approaches is provided in the table below.

Citation	**Brief Description**	**Limitations**	**Improvement**
(Zhao, Tang and Sun, 2021) ITDFA	Proposed an image transfer-based unsupervised domain adaptation approach by extending the	The approach performs well in dark rainy, and snowy conditions, but performance tends to degrade significantly	While maintaining the outstanding accuracy of depth estimation in day-time conditions of the original approach (Godard

	monodepth2 proposed in (Godard et al., 2018) to address depth estimation in challenging conditions such as night, rainy night and snowy night environments.	when rain streaks, snowdrops, or fog is present in images.	et al., 2018) model, the approach brought major improvements in the night, rainy night and snowy night conditions, which produced extremely unpredictable outcomes without these improvements and inconsistent results.
(Choi et al., 2020) SAFENet	Proposed a multi-task learning approach that incorporates schematic awareness and geometric knowledge to improve depth estimation in daytime conditions and is shown to generalise to various other conditions, including low-light and adverse weather.	According to the author, the approach has only been tested in virtual adverse weather conditions such as rain and fog and has only produced satisfactory results under adverse weather.	Being a self-supervised approach, it outperformed the baseline (Godard et al., 2018) approach on clear daytime depth estimation while bringing good predictions in adverse weather conditions.

Table 3: Overview of adverse weather monocular depth estimation methods

1.6. Research Gap

Accurate depth estimation in adverse weather conditions is an essential part of autonomous driving perception. Most current state-of-the-art adverse weather depth estimation approaches utilise expensive sensors such as lidar, and significantly less interest has been shown in the research of camera-based adverse weather depth estimation systems; hence monocular depth estimation in adverse weather conditions still remains to be a largely under-explored domain. The current state-of-the-art monocular depth estimation approaches that perform significantly well in clear weather fail to maintain accuracy and consistency when subjected to adverse weather conditions such as rain, snow and fog. So an approach to produce consistent depth

estimates in all these adverse weather conditions without any major performance degradations will help address a significant limitation in image-based depth estimation methods.

1.7. Contribution To The Body Of Knowledge

1.7.1. Technological Contribution

In this research, the author designs and develops a monocular depth estimation approach capable of producing consistent and accurate depth estimates than the state-of-the-art methods in adverse weather conditions such as rain, snow and fog. Furthermore, the depth estimation performance of the built model will be evaluated against the state-of-the-art models, and a detailed overview of the existing state-of-the-art depth estimation approaches will be provided.

1.7.2. Domain Contribution

A successful depth estimation approach using a monocular camera that produces consistent and accurate depth maps in adverse weather conditions will make image-based approaches more dependable and bring more community reliance which will be a huge step forward in the autonomous industry to move on from the expensive sensor-based approaches to camera-based approaches. This research contributes to the autonomous driving community by investigating a solid approach to estimating depth more accurately in adverse weather conditions.

1.8. Research Challenge

From the preliminary study of existing literature, it is understood that the adverse weather depth estimation framework should be able to generalise to input images of various weather conditions such as rain, snow and fog, which have a significant domain shift in between them and produce depth estimates accurately, consistently and overall high quality without causing unexpected drops in quality or performance degradations in various weather conditions. The approach should be able to produce depth maps with equal to or comparable quality to depth maps produced through state-of-the-art lidar sensors. The network architecture chosen should be able to learn quickly without excessive use of resources during training as resources will be limited and should be capable of producing depth maps quickly in fractions of a second using less computational power. Based on these core functionalities, it was understood that the research challenge mainly lies within the following areas:

The table below sums up the research challenges faced and what technologies were used to overcome them.

Research Challenge	**Overcome Using**
The learning approach should learn using fewer computational resources and be capable of producing high-quality depth estimates.	From the study of existing literature, it was understood that this could be solved by using a **transfer learning-based** approach.
The approach should generalise well to input images of adverse weather conditions, including rain, snow and fog and produce accurate and consistent depth maps.	To overcome this, an **adversarial autoencoder-based** approach was used to learn a mapping function to map images from various weather conditions and an optimal combined loss function to train these mappings.

Table 4: Overview of research challenges

1.9. Research Questions

RQ1: What type of learning approach could estimate depth accurately in a monocular fashion whilst consuming less computational resources during training?

RQ2: Would a two-staged approach which would first convert the adverse weather images to clear and then estimate depth from it, perform better than a robust single-stage depth estimation approach that can directly generalise to images of various weather conditions?

RQ3: What machine learning paradigm (supervised, unsupervised or self-supervised) would be most beneficial for depth estimation in adverse weather, given the nature of the dataset available?

1.10. Research Aim

The research aims to design, develop and evaluate a solid monocular depth estimation approach that can produce consistent and accurate depth estimates when subjected to various

adverse weather conditions, unlike most state-of-the-art methods, which fail to maintain accuracy and consistency when tested in adverse weather conditions.

1.11. Research Objective

The following table provides a detailed overview of the research objectives essential to be accomplished in order to complete the research project successfully and maps them with their respective learning outcomes.

Research Objectives	**Explanation**	**Learning Outcome**
Literature Review	- Examine previous publications on monocular depth estimation to find a feasible research gap that could lead to an exciting and challenging project with potential for future research. - Review existing literature on depth estimation and gain a solid understanding of the fundamentals of how depth maps are produced from a single image using computer vision techniques. - Analyse and identify computer vision techniques for developing a depth estimation approach for varying weather conditions. - Perform critical review of existing systems in close research areas such as object detection and segmentation that have investigated generalisation between multiple domains of varying distributions. - Identify technologies, frameworks, and pre-built packages that could aid the development process.	LO1, LO4, LO8, LO6, MA1, MA2, MA3
Data Gathering and Analysis	- Conduct requirement elicitation and gather the requirements of an adverse weather depth estimation system. - Get insights from domain experts on building a robust depth estimation system.	LO2,LO3, LO5, LO7, LO6, MA2, MA3, MA5, MA6

	- Define the software requirements based on the gathered requirements and identify the components of the system.	
Design	- Design an approach that could learn to produce accurate depth estimates using less computational resources whilst generalising well to input images from various weather conditions such as rain, snow and fog with zero to very minimal adverse effects or loss in overall depth map quality.	LO5, LO6, MA1, MA4, MA5, MA6
Development	- Develop the designed approach to address the limitations of current state-of-the-art approaches in estimating depth in adverse weather conditions.	LO9, LO6, MA2, MA3, MA5
Testing	- Create test cases, and the prototype will be subjected to several rounds of thorough testing using various testing strategies such as unit tests, integration, and functional testing. - The prototype will be evaluated by comparing error and accuracy metrics against the state-of-the-art methods using the results published on public benchmark sites such as vKITTI and Cityscape. - Validate the requirements of the system against the functional and non-functional requirements.	LO9, LO6, MA2, MA3, MA5

Table 5: Research objectives

1.12. Project Scope

After an intensive analysis of the existing literature and based on the research project's aim, resources, and time available, the project's in-scope and out-scope features were defined as follows.

1.12.1. In-scope

- The research predominantly aims to build a depth estimation approach capable of producing accurate depth estimations in varying weather conditions.
- The model will be trained using data collected under rainy, foggy and snowy adverse weather conditions.

- The depth estimation approach will only use a single RGB image to produce the depth maps.

1.12.2. Out-scope

- The research will not cover the depth estimation in low illuminated or nighttime images due to the project's increasing knowledge complexity and strict time constraints.
- This research will not address the unintended effects caused by adverse weather conditions, such as smooth surfaces being reflective under rainy weather, as overcoming reflections during estimating depth is another area of research.

1.12.3. Prototype Feature Diagram

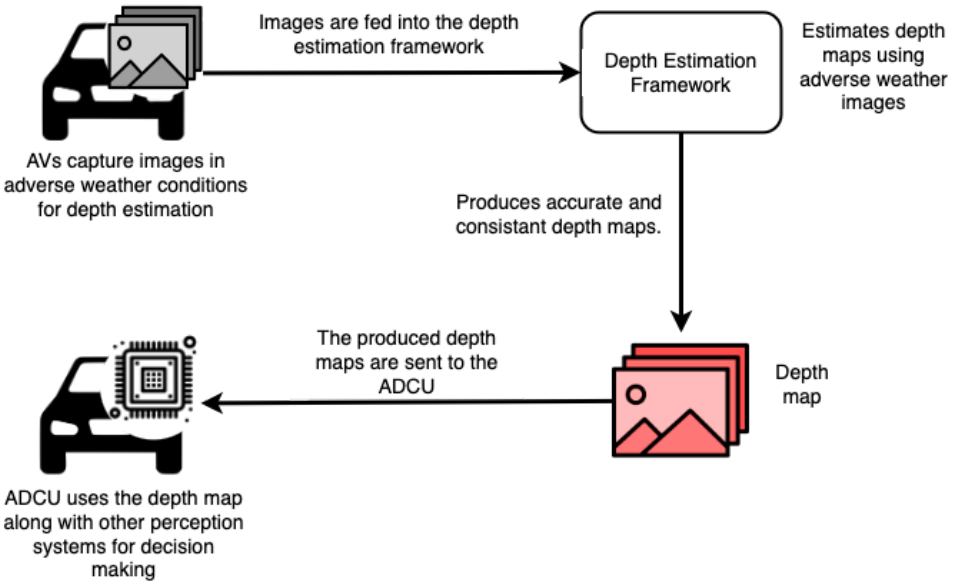


Figure 2: Prototype feature diagram

1.13. Resource Requirements

Based on the research objectives, the following hardware, software, skill and data requirements are essential for successfully completing the research project.

1.13.1. Hardware Requirements

- Intel Core i7 or equivalent - High compute power is required for processing Deep Learning and Computer Vision algorithms.

- 16GB RAM or more - Handling the large datasets used in the project consumes a large amount of application memory.
- Disk space of 40GB or more - A large amount of storage space is required to store the large autonomous driving datasets.
- Nvidia GPU with CUDA support. - GPUs with CUDA significantly reduce the time it takes to perform compute-intensive tasks.

1.13.2. Software Requirements

- **Operating System (Windows 10 / MacOS / Ubuntu)** - A 64-bit OS that efficiently handles resource-intensive tasks is required.
- **Pytorch / Tensorflow** - Using a machine learning library such as these, is crucial. It makes coding neural networks much more straightforward and would help focus on the system's logic rather than on syntax and other low-level functions.
- **VSCode / Pycharm / Jupyter Notebooks / Google Colab -** Working on an IDE helps improve code quality and greatly helps save time by identifying compile time errors.
- **Zotero** / **Mendeley -** A research assistant tool helps manage and backup research papers and easily cite and refer to them when needed.
- **MS Office / Google Docs** - Word processing software is essential for documenting the research process and writing reports.
- **Google Drive / OneDrive -** Cloud storage service is required to back up the work carried out on the project.

1.13.3. Data Requirements

- CityScapesWeather Dataset - a large-scale dataset put together by the author of this research by augmenting the Cityscapes dataset (Cordts et al., 2016) into various adverse weather conditions such as rain, snow and fog.

1.13.4. Skills Requirements

- Research and analysis skills - is essential for researching and analysing existing literature.

- Programming skills - is crucial for understanding existing systems and successful implementation of the prototype.
- Time management skills - is essential for planning and controlling the time spent on specific tasks in order to complete each deliverable on time.

1.14. Document Structure

The dissertation is structured into ten chapters. A brief description of the content of each chapter is provided below.

- **Chapter 01** - Introduction chapter provides an introduction to the project and its domain, defines the research gap and documents the research aim, contribution and scope of the research, along with the research objectives and motivations.
- **Chapter 02** - The literature review chapter documents the problem domain in much more detail and provides an in-depth review of existing literature.
- **Chapter 03** - Methodologies chapter documents the research and project management methodologies and presents the work breakdown chart, and activity schedules.
- **Chapter 04** - Requirement gathering chapter documents the various requirement elicitation methods followed and the functional/nonfunctional requirements gathered.
- **Chapter 05** - This chapter documents the legal, ethical and professional issues encountered during the development of the project and their mitigation process.
- **Chapter 06** - The design chapter documents the design choices taken and provides an overview of the complete architectural design of the system.
- **Chapter 07** - The implementation chapter documents the implementation process of the proposed system and various tools, technologies, and algorithms adapted.
- **Chapter 08** - The testing chapter covers the various tests to validate the developed system's functional and non-functional requirements. This chapter also benchmarks the developed system against the state-of-the-art approaches.
- **Chapter 09** - The evaluation chapter covers the process in which the developed system was evaluated based on expert feedback and the authors' self-evaluation.
- **Chapter 10** - Conclusion chapter provides final remarks on the project and reflects on the overall achievements of the project and possible future work.

CHAPTER 02: LITERATURE REVIEW

2.1 Chapter Overview

This chapter discusses in-depth the existing work in monocular depth estimation and means of coping with different weather conditions. The pros and cons of various approaches and a review of their performance in terms of their depth estimation accuracies under different weather conditions are extensively explored in this section.

2.2 Concept Map

With the knowledge gathered from the study of existing literature, a concept map was drawn. The concept map covers the entire scope of the problem domain and all concepts related to the problem domain. The concept map can be found in **Appendix A**.

2.3 Problem Domain

2.3.1 Autonomous Driving

Automobiles are a leading cause of death and injuries all around the world. Each year approximately 1.3 million deaths and 20-50 million non-fatal injuries are caused by automobile-related incidents (Road traffic injuries, 2022). According to the surveys carried out by WHO in 2018, most of these accidents are caused by human errors. Inattention and distractions while driving, driving too fast, misinterpretation of other traffic participants, falling asleep and drunk driving are some of the significant factors that have led to this massive death toll and fatalities count (Global status report on road safety 2018, 2018). A study conducted by the Insurance Institute for Highway Safety (IIHS) on the prospect of expanding existing automated driver assistance safety features revealed that almost one-third of these accidents could have been prevented, saving more than half-a-million lives annually if all vehicles had automated safety features like forward collision avoidance, lane departure warning, side views blind spot assistance and adaptive headlights (Brett, 2016). AVs are expected to reduce the number of accidents and increase safety substantially as the amount of human intervention needed by AVs is minimal, thereby minimising the number of incidents that human errors could cause to a deficient level.

This potential of AVs to improve automobile safety is the primary motivation for the development of autonomous vehicles, and it keeps the field moving forward. In addition to the safety improvements, AVs bring to the table several other benefits such as reduced energy use

and emissions by being able to be programmed to operate more efficiently by maintaining more efficient speeds and eliminating unnecessary braking and acceleration. AVs hold the ability to be programmed to form platoons which would massively minimise drag on vehicles, helps them conserve energy and makes them much more fuel-efficient. With instant reaction speeds, car-to-car communication and reduced car accidents, AVs could also benefit society by reducing traffic congestion (Frank, 2020).

The task of autonomous driving is often treated as a modular task, and the overall problem is divided into several well-studied subtasks as perception, state estimation & localisation, path planning, decision making and control. By modularising the problem into sub-tasks, each sub-task can be separately developed and tested, making identifying the root cause of errors much more accessible and straightforward and accumulated know-how and expertise can be transferred directly (Bachute and Subhedar, 2021).

2.3.2 Perception For Autonomous Driving

The perception system of AVs is responsible for understanding the surrounding environment by processing data collected through various sensors and making this information available to the decision-making control unit for taking appropriate actions based on the situation the vehicle is in (Dholakia, 2019). Tasks commonly performed by the perception systems of AVs include:

- Scene understanding, i.e. position, depth and size of objects in 3D, orientation and velocities of moving objects in the surrounding, etc.,
- Semantic interpretation of the identified objects and intention recognition of objects in the surrounding, i.e. identifying and understanding traffic signs, identifying the type of objects, whether its a person, bicycle or a car which helps in predicting their near-future behaviour such as gesture recognition of cyclists and head and body rotation of pedestrians.

Depth estimation, a significant component of geometric scene understanding, often uses data from a combination of sensors such as lidar, radar and cameras to process and produce depth estimates of its surroundings. The pros and cons of each sensor and why the author chose the camera as the preferred sensor for depth estimation are explained in detail in the Introduction section.

As mentioned in the introduction chapter, the depth of a scene can be extracted using cameras in 2 approaches, namely by monocular and stereo depth estimation. Since stereo depth

estimation suffers from a variety of limitations compared to monocular depth estimation (explained in more detail in the Introduction section), monocular depth estimation was chosen as the preferred approach in developing the proposed system.

2.3.3 Monocular Depth Estimation

Monocular depth estimation is the task of producing a depth map consisting of pixel-wise depth estimates from a single RGB image. It is often referred to as an ill-posed problem, as a single 2D RGB image can originate from an infinite number of different 3D scenes due to monocular shape and scale ambiguities (Piven, 2021). Depending on the availability of data and ground truth supervision, depth estimation from monocular cameras is generally achieved in supervised, self-supervised or unsupervised learning approaches. The following topics briefly explain the depth estimation process from these commonly used learning approaches.

2.3.3.1 Supervised Approaches

Before convolutional networks, monocular depth estimation problems were tackled using in-direct approaches by considering it as a sub-task of 3D reconstruction, which relies heavily on hand-crafted features by making strong assumptions about the 3D structure of the scene (Mertan, Duff and Unal, 2022). In supervised learning approaches, this task is generally considered a regression problem and approached by using deep convolutional network-based architecture models to make inferences using the pixel-wise depth information obtained from the ground-truth data (Liu, Shen and Lin, 2015), thereby producing a similarly sized output image as the input image with depth predictions for every pixel instead of the original RGB values. The only major downside of this approach is that these approaches require an accurate source of supervision to learn correctly, and most commonly, as the primary supervision source, lidar data is used, which is costly to collect. In recent years to address this limitation in supervised approaches, a significant number of researches have been done focusing on self-supervised and unsupervised depth estimation approaches in which training with very little or no labelled ground truth data is possible.

2.3.3.2 Self-Supervised Approaches

Unlike supervised approaches, which rely heavily on ground-truth data, self-supervised monocular depth estimation methods rely on geometry as the primary source of supervision.

Self-supervised approaches use view synthesis as the meta-objective, and depth estimates are obtained as a by-product necessary to perform projections for image warping, unlike supervised approaches, which formulate the task of estimating depth as regression of the depth information obtained from ground-truth data (Piven, 2021).

2.3.3.3 Unsupervised Approaches

Similar to self-supervised approaches, unsupervised approaches use geometry as the primary source of supervision. The main difference between self-supervised and unsupervised approaches is that in unsupervised approaches, depth estimates are obtained using structure-from-motion (SfM) (Pinard et al., 2019) techniques using monocular image sequences consisting of camera motion or monocular video data to train. Since unsupervised approaches are based on SfM assumptions, the training data are expected to be properly 3D-reconstructed, e.g. camera motion, static scenes, and no occlusions or reflective surface (Gonzales, 2020).

Unfortunately, most of the self-supervised and unsupervised approaches fall far behind supervised approaches in terms of depth estimation accuracy and still, monocular depth estimation benchmarks are often led by supervised approaches. Also, another limitation of self-supervised and unsupervised approaches is that even though they are monocular depth estimation approaches, they still require an input of stereo images or image sequences while training, which is not as easy to collect as single monocular images. Compared to supervised approaches, unsupervised and sefl-supervisedapproaches are much more computationally intensive and require much more resources and time when training. So, therefore, by considering these factors, it was decided to utilise a supervised training approach for the system proposed in this research.

2.3.4 Effects Of Adverse Weather Conditions On Monocular Depth Estimation

A significant challenge in monocular depth estimation is dealing with adverse weather conditions such as rain, snow, fog, etc., and poor lighting conditions. Since most state-of-the-art depth estimation approaches are designed and tested only on ideal clear weather conditions, they get severely affected and produce inconsistent and unintended results in adverse weather conditions. As AVs are considered safety-critical automobiles, this

limitation of current state-of-the-art approaches in adverse weather conditions is a critical issue. Unlike ADAS systems, which can still miss out on challenging occasions as the driver is still in charge, AVs should handle all conditions at all times well without needing any intervention from the driver. The most frequently occurring adverse weather conditions that often affect the performance of the depth estimation models are rain, snow and fog. Overcoming the challenges these weather conditions introduce on depth estimation is the primary goal of this research.

2.3.4.1 Rain

Rain can heavily affect visibility depending on the level of rain, orientation and size of the droplets. Rain introduces sharp intensity fluctuations in images causing the image quality to degrade and may cause temporal variations in scenes. In rainy weather, smooth surfaces such as vehicle bodies and puddles on road surfaces could be made reflective, which leads to false depth estimations (Zang et al., 2019).

2.3.4.2 Snow

Snow significantly affects visibility as heavy snow causes scattering and can increase image intensity, which can obscure accurate detection of object edges in a scene, producing inaccurate depth estimates (Rajderkar and Mohod, 2013).

2.3.4.3 Fog

Foggy conditions drastically affect visibility by shortening the visible distance and making the environment darker, which results in degradation of the image quality by reducing contrast and introducing excessive noise. Fog could also cause moisture to build up on camera lenses, resulting in a similar impact as during rain and snow, negatively influencing depth estimation. (Zang et al., 2019).

2.3.6 Proposed Architecture

The proposed architecture diagram is provided in the Algorithm Design section in the System Architecture and Design chapter, along with clear explanations as to why such an architecture is utilised.

2.4 Existing Work

2.4.1 Monocular Depth Estimation Approaches

(Eigen, Puhrsch and Fergus, 2014) This paper proposed the earliest approach that directly estimated depth from images by considering monocular depth estimation as a regression problem in a supervised learning manner. The approach proposed in the paper works in a two-stage process using a single neural network with two components. The first component (coarse-scale network) estimates the global structure of the image, and the second component (fine-scale network) refines the estimated global structure using local information. The coarse-scale network consists of five feature extraction layers of convolutional layers pretrained on the ImageNet dataset and max-pooling layers followed by two fully connected layers with dropout added to the final fully-connected layer. The max-pooling layers help concatenate information from different parts of the original image to a smaller spatial dimension, thereby enabling the network to use the global information of the entire image to predict the depth. The fine-scale network consists of a series of convolutional layers led by a single max-pooling layer for scaling the depth map to the exact size of the original input image and a final linear convolutional layer for outputting the depth map. The coarse prediction is fed into this network as an additional low-level feature map. The network then enhances the fed in coarse prediction by incorporating the local details such as object and wall edges. Both networks are trained in an end-to-end fashion where the coarse network is first trained with the ground truth data, and then the fine-scale network is trained while keeping the output of the coarse network fixed by freezing the weights of the coarse network. With the zero-padded convolutions in the fine-scale network, the final output depth map from the fine-scale network will retain the same spatial size as the output from the first pooling layers of the fine-scale network.

DenseDepth (Alhashim and Wonka, 2019) proposed a high-performing approach using a much simpler architecture than most depth estimation approaches available at the time. The approach produced depth maps with higher accuracy and significantly higher visual quality than the other supervised approaches that treated depth estimation as a regression problem. Since the release, the approach remains among the top 5 in most clear daytime-only depth estimation benchmarks. The network consists of a U-Net (Ronneberger, Fischer and Brox, 2015) architecture in which a pretrained DenseNet-169 (Huang et al., 2018) model as the deep feature encoder was originally built to extract features for image classification trained on the Imagenet dataset was used. The decoder of the proposed system consists of two bilinear

upsample layers followed by two standard deconvolutional layers, which together up-samples the output vector from the encoder to form the final depth map of half the size of the original input image. By producing a depth map of half the spatial size of the original input image and later on rescaling the depth map back to the original size using simple image interpolation separate from the decoder it was found out to be extremely resource-efficient with no loss of depth map accuracy and visual quality.

The loss function used in the approach is a combination of 3 different loss functions defined to minimise the difference between the estimated depth map and the ground truth depth map. The final loss of the approach is defined as the weighted sum of point-wise L1 loss defined on the depth values, L1 loss defined on the gradient g of the depth values and SSIM loss defined on the depth values.The equation for the loss function is defined as follows, where lambda = 0.1, y = ground truth depth , y hat = estimated depth map :

$$L(y, \hat{y}) = \lambda L_{depth}(y, \hat{y}) + L_{grad}(y, \hat{y}) + L_{SSIM}(y, \hat{y})$$

2.4.2. Adverse Weather Depth Estimation Approaches

This section provides an in-depth review of the current state-of-the-art monocular depth estimation that had previously attempted adverse weather depth estimation by fusing data obtained with a multitude of sensors and using monocular cameras along.

2.4.2.1 Multiple Sensor Fusion Approaches (Depth Completion)

R4Dyn (Gasperini et al., 2021) Proposed an approach that can use sparse radar data during training and as a weak supervision signal during inference time to enhance the robustness of depth estimation in dynamic scenes and some adverse weather conditions. The approach uses monocular video, radar data, and vehicle odometry for scale awareness in a self-supervised manner. The approach works by simultaneously predicting depth and estimating the pose transformation between the target and source frame. Which then is used to warp source frames into the reconstructed target views from which an appearance-based error is calculated. The sparse depth estimates and the weak velocity supervision provided by the radar sensors are then used as a weak supervision signal to predict accurate estimate depths and achieve scale awareness in dynamic scenes.

(Siddiqui, Vierling and Berns, 2020) proposed a multi-modal depth estimation approach that extended the approach proposed in (Alhashim and Wonka, 2019) by fusing sparse radar signals collected under various adverse conditions with the input RGB images.

Similar to (Alhashim and Wonka, 2019), an encoder-decoder architecture was used and trained with transfer learning. Instead of just using a single RGB image as input to the encoder, an RGB image fused with sparse radar point cloud data was used. The Encoder then maps the inputs into a feature representation which is then fed to the decoder, and a full-resolution depth map is produced instead of the half-scaled depth map as it was done in (Alhashim and Wonka, 2019). As the author mentioned, the approach brought significant improvements in estimating depth in zoomed and top-view images under adverse weather, but only slight improvements were observed in general depth estimation in adverse weather conditions.

2.4.2.2. Monocular Approaches

ITDFA (Zhao, Tang and Sun, 2021) Proposed an image transfer-based domain feature adaptation approach that produces reliable and accurate depth estimates in low illuminated nighttime conditions and complex environments such as rainy nights and snowy winter night scenarios. Similar to (Vankadari et al., 2020) for which, this paper finds an improved approach for, adapts a state-of-the-art unsupervised day-time only monocular depth estimation approach (Godard et al., 2019) to low-illuminated complex environments. The approach works by encoding images of complex environments to the same feature space as day-time images using a pre-trained CycleGan image translation model. The network consists of an encoder-decoder architecture that, during training, uses two encoders. One pretrained encoder encodes the day-time images and obtains their feature maps, while the other encodes images of complex environments and obtains their feature maps in the same feature space as the day-time images. The weights of both pretrained encoder and decoder are fixed during training, and the weights of the encoder, which translates the images of complex environments, are updated. The shared pretrained decoder is trained to decode the features of both these feature maps and obtain the depth maps. The approach uses a combination of 3 losses, a feature consistency loss to ensure that both encoders produce features maps in the same feature space, a depth consistency loss on the decoder to minimise the error between the decoded depth maps produced from each feature map and finally a smoothness loss which utilises edge-aware smoothness (Godard et al., 2019) between the input day time image and its corresponding depth map to promote smoothness of the generated depth map.

SAFENet (Choi et al., 2020) proposed a self-supervised monocular depth estimation that leverages semantic information to overcome the limitations in photometric loss, which is used to provide self-supervision instead of ground truth depth maps. The approach introduced a multi-task learning scheme using the semantic-aware depth features from a video sequence

or a single image that incorporates semantic and geometric knowledge into the representation of depth features. Extensive experiments on various datasets, including KITTI, vKITTI and Nuscenes, demonstrated that the approach generalises better and more robustly than other state-of-the-art methods in various conditions, such as low-light or adverse weather.

DeFeat-Net (Spencer, Bowden and Hadfield, 2020) proposed an approach that learns monocular depth and cross-domain dense feature representation in a fully self-supervised manner using only a stream of monocular images as an input to produce depth estimates in adverse weather conditions. Defeat-Net combines three complementary networks to solve for feature representation simultaneously, depth and ego-motion, namely DispNet, PoseNet and FeatNet. DispNet, which, when given an image, produces a corresponding depth map using a disparity estimation network formed by a ResNet encoder and decoder with skip connections. PoseNet predicts the pose, representing a composition of a rotation in axis-angle and a translation vector scaled by a factor of 0.001 using a multi-image ResNet encoder followed by a 4-layer convolutional decoder. Finally, FeatNet, composed of a residual block encoder-decoder network with skip connections, produces a dense n-dimensional feature map using the image fed into the network. In challenging environments such as during nighttime and highly occluded scenarios, the DeFeat-Net outperformed the state-of-the-art depth estimation approaches since the approach is unaffected by light and heavily dependent on the image features. However, the approach has a way of enforcing feature consistency across adverse weather conditions, which could result in lost essential features and inaccurate depth maps.

2.5 Review Of Algorithms

2.5.1 Transfer Learning

Transfer learning is a process in which knowledge acquired from learning one task is used to help learn a different but related task more efficiently. This allows us to utilise knowledge (features and weights) obtained by learning on tasks with significantly large amounts of data consuming enormous amounts of computational power and generalise this knowledge for a related task with significantly fewer data and computational power.

Transfer learning is widely used in deep learning, especially for computer vision and natural language processing tasks where pre-trained models are used as starting points to networks which usually require vast computing and massive datasets to obtain favourable results. This approach also eliminates the need to build separate deep neural network models

from scratch for similar tasks and provides the opportunity to use well-performing pretrained models to speed up training and improve performance (Brownlee, 2017).

2.5.2 Autoencoders

Autoencoders are a type of neural network that was primarily designed to output a reconstruction of the input such that the output is as similar as possible to the original input. Autoencoder consists of 2 sub-models, an encoder and a decoder. During training, the encoder downsamples the input data and learns a set of features known as the latent representation, and the decoder learns to reconstruct/upsample the input data back into the original input based on the latent representation learned by the encoder. After training, the trained weights and model can be saved and used to predict inputs that are not previously seen, or the decoder can be discarded, and the encoder can be used as a feature extractor to train different machine learning models (Brownlee, 2020). Several variations of autoencoder exist; in this research, the author uses a U-Net autoencoder, a general autoencoder with skip connections between layers of the encoder and decoder, to prevent spatial information from being lost during downsampling.

2.5.3 Generative Adversarial Networks (GANs)

Initially proposed by Ian Goodfellow in 2014, GANs are a type of deep generative neural network composed of 2 separate sub-models, the generator and the discriminator, trained simultaneously to compete against each other. The generator, a generative model whose ultimate goal is to produce output data identical to the input data, captures the input data distribution and produces fake data with similarities to the original input data. The discriminator is trained simultaneously to identify fake images that are being produced by the generator and penalise them accordingly. The discriminator compares the output produced by the generator and the actual input data to the network and applies a loss value depending on how different the produced image is from the actual image. The process continues for several epochs until the generator successfully learns to create images with identical features to the input images, and the discriminator fails to identify it as a fake image (Goodfellow et al., 2014).

2.6 Review Of Evaluation Techniques

The method's performance will be evaluated and compared using both the error and accuracy metrics proposed by (Eigen, Puhrsch and Fergus, 2014), which has since become the industry standard for monocular depth estimation. The error metrics include Absolute Relative Difference (AbsRel), Root Mean Square Error (RMSE), RMSE (log) and Square Relative Error (SqRel). Accuracy metrics include δ<1:25t, where t = 1,2,3. For all equations mentioned, d_i and d_i^* are the ground truth and predicted depth at pixel i and N is the total number of pixels.

- $Absolute\ Relative\ Difference\ (AbsRel) = \frac{1}{N}\sum \frac{\left|d_i - d_i^*\right|}{d_i}$
- $Root\ Mean\ Square\ Error\ (RMSE) = \sqrt{\frac{1}{N}\sum \left|d_i - d_i^*\right|^2}$
- $RMSE(log) = \sqrt{\frac{1}{N}\sum \left|logd_i - logd_i^*\right|^2}$
- $Square\ Relative\ Error\ (SqRel) = \frac{1}{N}\sum \frac{\left|d_i - d_i^*\right|^2}{d_i}$
- Accuracy with threshold ($\delta < thr$) : % of d_i such that $max\left(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}\right) < thr$, where $thr = 1.25, 1.25^2, 1.25^3$

2.8 Chapter Summary

The chapter begins by providing a concept map of the problem domain and an in-depth review of the problem domain. The existing approaches in the monocular depth estimations were then critically analysed, and their shortcomings and strengths were clearly reviewed. With the knowledge acquired by reviewing existing literature in the domain, the architecture of the proposed system was decided, and the architecture diagram of the proposed system was illustrated. Finally, a detailed review of the critical concepts of the decided architecture was provided, followed by a review of the standard evaluation techniques used in the domain of monocular depth estimation.

CHAPTER 03: METHODOLOGY

3.1. Chapter Overview

This chapter provides an overview of the research, project management and development methodologies followed throughout the duration of the research project. Details on each chosen methodology, along with justifications for choosing to follow a specific methodology, will be provided in this chapter.

3.2. Research Methodology

Project quality is defined by three main factors: time, cost, and the well-defined scope. Methodologies should be defined in order to maintain project quality. The following list tabulates the methodologies chosen in respective sectors.

Research Philosophy	From the possible candidates of research philosophies pragmatism, positivism, realism and interpretivism, the author of the research has selected **pragmatism** as the research philosophy as the research involves gathering data or knowledge by the usage of a mix of quantitative and qualitative approaches such as through experiments which are quantitative and interviews which are qualitative to evaluate different aspects of the research problem.
Research Approach	The author adopts a **deductive** data collection which is more favourable for the project compared to an inductive approach as the research process involves studying existing theories in monocular depth estimations and various theories built upon domain adaptation and formulation of hypotheses and their subjection to testing on testing and the purposes of analysing the collected data a quantitative approach was chosen as quantitative approach better suits the project in consideration with the research philosophy rather than a qualitative approach.
Research Strategy	**Case-study** was chosen as the preferred research strategy as the research involved an in-depth investigation of existing depth estimation techniques through the use of different types of data such as observations, interviews and analysis of documents.

Research Choice	**Mixed-method** was chosen as the research choice because as described both quantitative data such as the numerical performance values obtained from a variety of experiments carried out and qualitative data such as the feedback from the interviews conducted is used throughout the research.
Time horizon	**Cross-sectional** time horizon was chosen as the preferred time horizon for the research as the research was limited to a specific time frame and not repeated over an extended period.

Table 6: Overview of identified research methodologies

3.3. Development Methodology

3.3.1. Life Cycle Model

Prototyping model was chosen as the preferred SDLC model among the various SDLC models because the project involves initially developing a prototype of the proposed system and continuously making refinements and alterations according to the test results and feedback received until the system is fully developed and an acceptable outcome is achieved.

3.3.2. Design Methodology

Structured System Analysis and Design Modelling (SSADM) was chosen as the preferred design methodology for the research. SSADM mainly focuses on the process and procedures of the system and allows concentrating more on developing its functionalities rather than focussing on the data structures and real-world objects, and since the project is using Python programming language in a structured manner, SSADM was ideal.

3.4. Project Management Methodology

3.4.1. Gantt Chart

Gantt Chart illustrating the project schedule can be found in **APPENDIX B** section.

3.4.2. Deliverables

The following table lists the project deliverables and their expected due dates that need to be achieved in time for successful completion of the project.

Deliverable	Date
Project Proposal Document The document defines the research project and provides an initial overview of the project.	11th November 2021
Software Requirement Specification (SRS) The document specifies the requirements of the proposed system based on the data gathered through elicitation.	25th November 2021
Proof of Concept Version 1 The preliminary implementation is presented to the mentor to get feedback.	6th December 2021
Interim Progress Report (IPR) The document presents the initial evaluation findings.	27th January 2022
Presentation of Prototype The prototype will be presented to the mentor after completing the core functionalities to get further feedback.	21st February 2021
Final Project Report The final document reviews the overall research process and decisions carried out.	11th July 2022

Table 7: Deliverables and dates

3.4.3. Risk Management

The following table contains an overview of the possible risks associated with the research project and relevant mitigation procedures to overcome or avoid such risks.

Risks	Severity	Mitigation Plan
Not having enough computational resource power for resource-intensive tasks	High	Could be avoided by training the models on cloud-based solutions such as Google Colab or Google Cloud Platform.

Lack of domain knowledge	High	Could be avoided by intensive self-study of existing literature and systems and learning new skills.
Proposed approach not performing as expected.	High	Could be avoided by using unbiased and clean datasets and by addressing the problem with the proper understanding of the problem domain.
Data loss due to hardware failures	Medium	Could be overcome by regularly backing up the work to cloud-based solutions.
Running out of time	Medium	Could be overcome by maintaining proper time management throughout the research process.
Unforeseen risks such as health related issues	Low	Requires application of mitigation forms and research submissions will have to be deferred.

Table 8: Evaluation of risks and mitigation procedures

3.5. Chapter Summary

The chapter starts with a detailed explanation of the various available research methodology and provides justification for choosing a specific research methodology. Then the chapter explains the various development methodologies chosen and justifications for considering a specific development methodology. Finally, the project management methodologies chosen were explained, followed by a project plan, various deliverables and a discussion of various mitigation procedures for risks associated with the project.

CHAPTER 04: SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Chapter Overview

This chapter presents a detailed review of the process of gathering requirements from the identified stakeholders and analysing these requirements to identify the essential ones for the proposed system. The identified stakeholders, their roles in the proposed system, the various requirement gathering techniques used, the use-cases of the system and finally, the scope for the project will be defined by identifying the functional and non-functional requirements of the system.

4.2 Rich Picture Diagram

The identified stakeholders and their roles on the problem domain are illustrated in the following diagram.

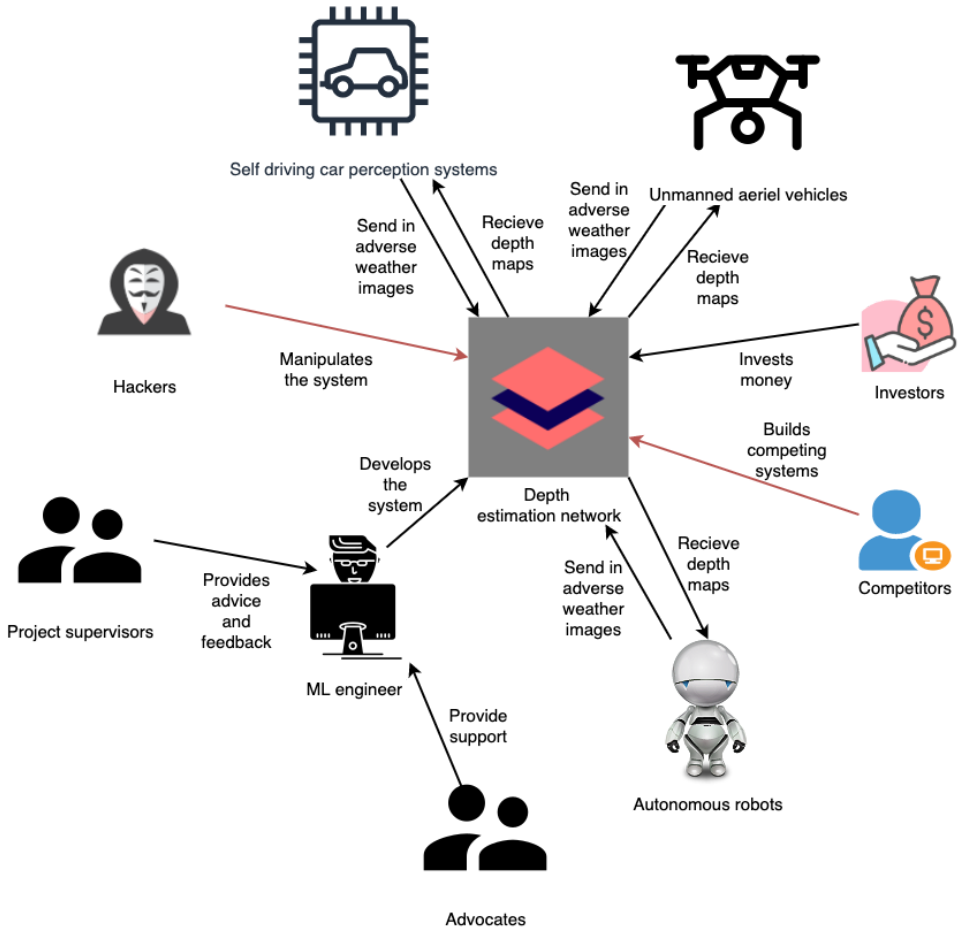


Figure 3: Rich Picture Diagram

4.3 Stakeholder Analysis

The established stakeholders with direct involvement with the proposed system are illustrated in the following diagram, along with a description of their roles in the system.

4.3.1 Stakeholder Onion Model

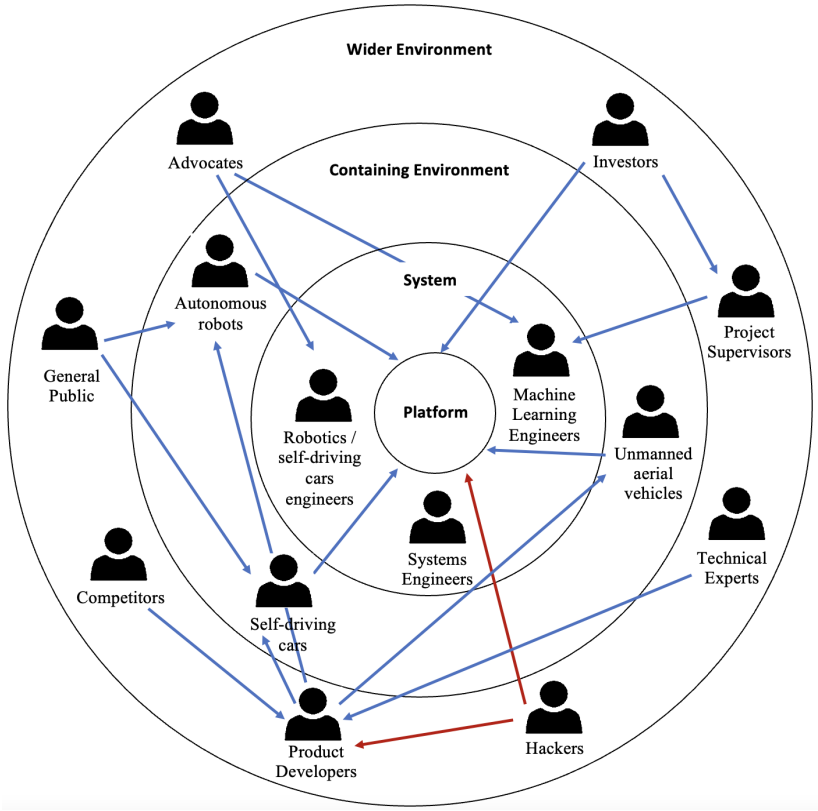


Figure 4: Stakeholder Onion Model

4.3.2 Stakeholder Viewpoints

Stakeholder	Role	Description
System Stakeholder		
Machine Learning Engineers, Computer Vision Engineers	Operational - Maintenance	Maintains and improves the system and keeps the system functioning at all times.
Robotics engineer, Self-driving car engineers	Operational - Administration	Integrates the system to the containing system and makes sure the system

		coordinates with the rest of the containing system.
Systems engineer	Operational - Support	Develops and maintains the containing system and makes sure the conditions for the system to function are met at all times.
Containing System Stakeholders		
Self-driving cars	Functional beneficiary	Self-driving cars will use the system to perceive depth of objects in its surroundings.
Autonomous robots		Autonomous robots will use the system to estimate distance of obstacles and in all other locomotion tasks under any weather condition.
Unmanned aerial vehicles (UAVs)		UAVs will use the system for estimating the distances of obstacles and trajectory estimation in all weather conditions.
Wider Environment Stakeholders		
Advocates	Advisory	Provides the guidance and support required to use the system successfully.
Project Supervisors		Provides guidance to complete the system successfully.
Investors	Financial beneficiary	Invest money during the development stages of the system expecting a larger financial return once the system is deployed to production.
Electrical / Mechanical engineers	Engineering Staff	Works on the development and maintenance of the environment the proposed system

		would be deployed in, such as general non-self-driving vehicles.
General public	Functional beneficiary	Uses the system once it has been deployed into production vehicles.
Competitors	Negative stakeholder	Creates systems with improved features to the proposed system.
Hackers		Intends to disrupt or infiltrate the system in order to access or destroy data.
Technical Experts	Expert	Evaluates the system and provides feedback in order to further improve the system.

Table 9: Stakeholder viewpoints

4.4 Selection Of Requirement Elicitation Methods

In order to overcome the limitations of different elicitation methods and identify the essential system requirement from the selected stakeholders, several elicitation methods were utilised. The table below provides an overview of the chosen elicitation methods and what factors led each elicitation method to be chosen.

Method 1: Literature Review	
Reviewing existing research on the problem domain helps identify the current systems' strengths, weaknesses and limitations.	
Advantages	– Provides access to well-documented research, which helps in correctly understanding and refining the requirements of the proposed system. – Strengths, weaknesses and limitations of existing systems will be clearly stated, which helps identify the requirements of the proposed system.
Disadvantages	– Thorough reviewing of existing literature would be time-consuming.
Method 2: Observing Existing Systems	

By observing and analysing the existing systems, the feature gaps of the systems can be identified, which helps determine the essential requirements of the proposed system.	
Advantages	– The essential requirements of similar systems can be identified.
Disadvantages	– Identifying the limitations of existing systems through only observing them would be a lot more time-consuming than a literature review and may not guarantee all limitations of the system to be identified.
Method 3: Interviews	
By interviewing domain and technology experts, the proposed system's requirements can be easily identified and essential when deciding on the functional and non-functional requirements.	
Advantages	– Provides an opportunity to clear out problems faced during requirement identification and implementation.
Disadvantages	– Reaching out to many domain and technology experts would be difficult.
Method 4: Prototyping	
Following a prototyping approach, it allows the proposed system to continuously make refinements and improvements based on test results. This also provides an excellent opportunity to gather and refine the functional and non-functional requirements, which would improve the system's quality.	
Advantages	– Provides an opportunity to determine how important certain features/requirements would be for the overall system.
Disadvantages	– Would be time-consuming, and relying mainly on prototyping as a source of requirement gathering could lead to poorly engineered systems, and less time would be spent analysing existing systems and literature.

Table 10: Requirement elicitation methods

4.5 Discussion Of Findings Through Different Elicitation Methods

The following table provides a review of the findings obtained from each elicitation method.

Method 1: Literature Review
After an extensive review of existing literature on the domain, it was found that there had been a considerably low number of researches carried out to address the limitations of monocular depth estimation in adverse weather conditions. Certain researchers have proposed sensor fusion approaches where lidar or radar sensors are used together with cameras to enhance the depth estimation in adverse weather conditions, but due to the current extreme cost of these sensors, it is impractical to be used in commercial vehicles.
Method 2: Observing Existing Systems
By observing the current state-of-the-art monocular depth estimation systems in action, it was observed that these systems produce phenomenal results in clear daytime conditions but fail to maintain accuracy and consistency when tested out in adverse weather conditions such as rain, fog or snow. The systems showed significant limitations, such as random dark spots with no depth estimates in some regions of the images with low texture, and some showed unexpected losses of image details resulting in very poor depth estimates in adverse weather conditions.
Method 3: Interviews
Several experts from both the domain and technological sides were interviewed to identify the requirements of the proposed system. All interviews were conducted online, and details of the various domain and technology experts that were interviewed are provided in **Appendix D**. A thematic analysis of the conducted interviews is presented in the table below.

Theme	**Analysis**
Research gap and depth of scope	All of the interviewees stated that this would be a challenging task as the domain distribution of each weather condition may vary significantly, and all stated that the research gap and depth of scope are adequate for the final year project and that this is a valid research gap which is helpful for many other applications in addition to depth estimation.
Features of the prototype and suggestions	It was stated during several interviews that the three weather conditions, rain, snow and fog well suffice for

	the prototype of the system and suggested including night time-depth estimation as well into the system if there was spare time after the development of the system and clearly stated that it would increase the complexity of the system furthermore.
Using autoencoders for mapping weather images directly to depth maps	During two out of four interviews with industry experts in the field of computer vision, it was pointed out that this problem is kind of related to image decomposition and denoising, and it was suggested to try using autoencoders as they stand out in this kind of task and requires considerably less computational resources to train them.
Using generative adversarial networks to translate weather images to clear images	It was pointed out during almost all interviews that I brought up the question that this approach might fail if the input images to the system are very different to each other and might lead to mode collapsing. It was suggested to try using supervised image-to-image translation methods initially and see how well the models would learn.
Any other possible approaches for tackling the problem	It was suggested to try out shallow learning approaches such as image priors and feature matching approaches, which might work depending on the training dataset.
Method 4: Prototyping	
The validity and the tangibility of the proposed system were understood by prototyping. Continuously experimenting using different methodologies and technologies helps identify the most suited approach for building the system. Several different approaches were experimented ranging from domain adaptation to transformer-based autoencoders, and finally, it was deduced that the best approach for solving the problem was using an adversarially trained autoencoder. Furthermore, repetitive experimentation helps identify the weaknesses of the	

selected development approach and aids in fine-tuning the system by identifying the optimal hyperparameters.

Table 11: Findings through different elicitation methods

4.6 Summary Of Findings

A summary of the findings obtained from each elicitation method is presented in the table below.

Finding	Literature Review	Observing Existing Systems	Interviews	Prototyping
Current state-of-the-art depth estimation approaches fail on adverse weather conditions	✓	✓		
It would be a major contribution to the body of knowledge to develop a system that could outcome the weakness of current depth estimation approaches.			✓	
The proposed system should be able to function properly better than state-of-the-art approaches in rain, snow and fog weather conditions.	✓		✓	
The system would require synthetic paired data with accurate depth maps to train with.	✓			✓
Autoencoder based approaches could be used to produce depth maps	✓	✓	✓	

directly from adverse weather images.				
Generative adversarial networks could be used to translate images between weather conditions.	✓	✓	✓	
Training an autoencoder in an adversarial manner by integrating a discriminator could potentially increase the generalisation capabilities of the system.	✓			✓
Image decomposition techniques could be used to help obtain a clear image from an adverse weather image.	✓		✓	

Table 12: Summary of the findings

4.7 Context Diagram

The diagram below illustrates the interactions between the system and the external factors from which the boundaries of the system are defined.

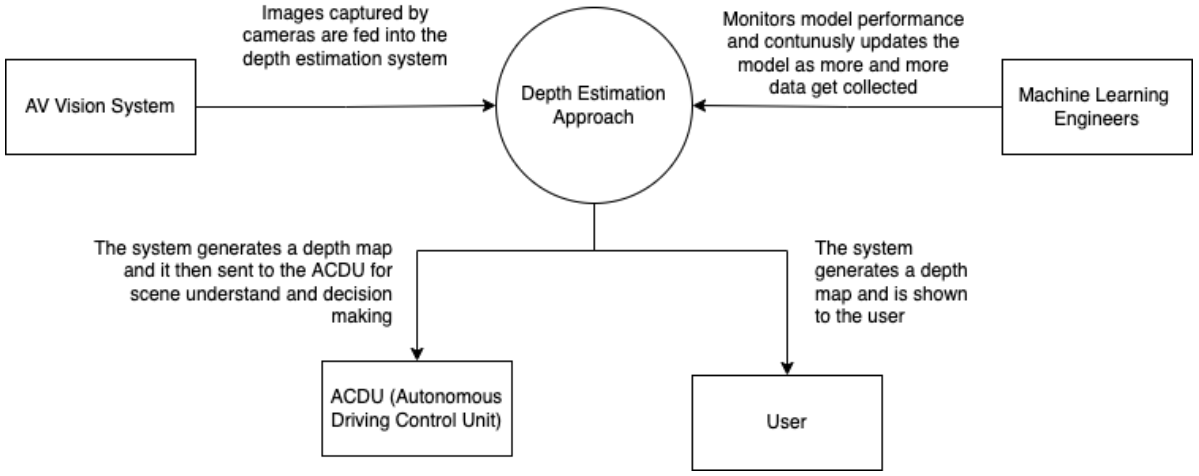


Figure 5: Context diagram of the proposed system

4.8 Use Case Diagram

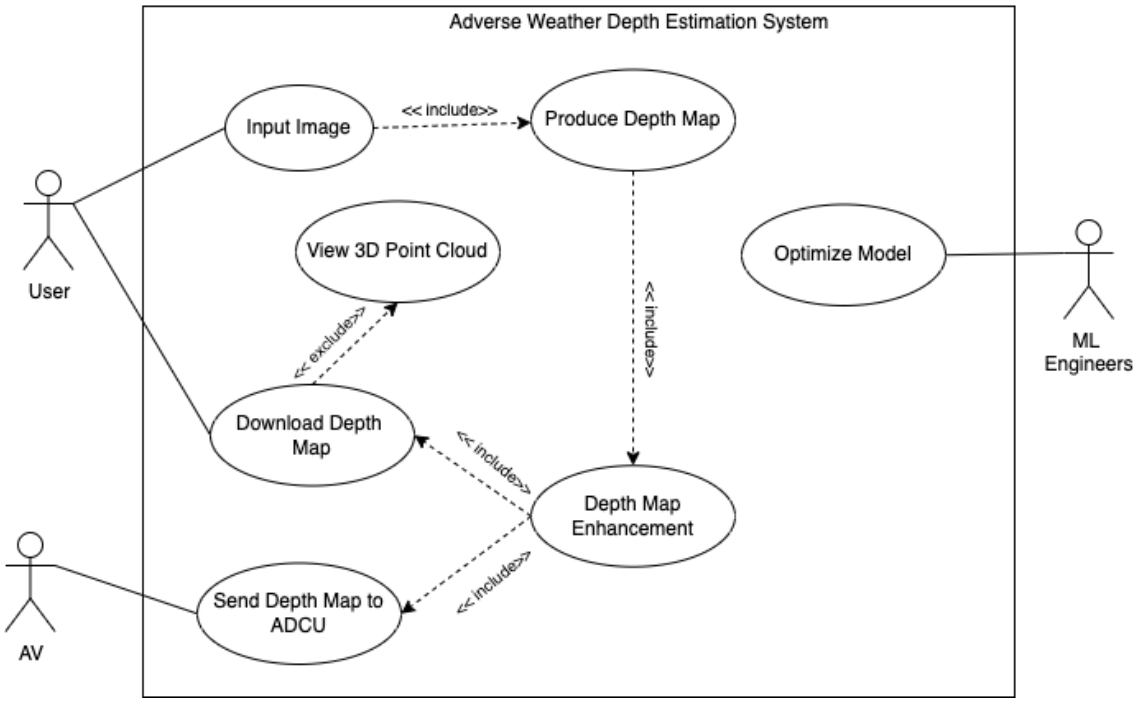


Figure 6: Use case diagram

4.9 Use Case Description

Use Case #1	Input image
Description	A video stream captured under various weather conditions by autonomous vehicle cameras is fed into the model as an image sequence, from which frame by frame will be fed to the system to produce the depth maps.
Participating Actor	Autonomous vehicles (AVs)
Preconditions	The image should be captured using a monocular camera.
Extended Use Cases	None
Included Use Cases	Produce Depth Map
Main Flow	1. Split the video into separate image frames. 2. Resize the image to fit the modal parameters. 3. Input to the system to get the depth map.

Alternate Flow	None
Exceptional Flow	None
Post Conditions	The fed image will undergo a series of transformations if required and finally an accurate depth map will be produced.

Use Case #2	Download Depth Map
Description	The user should be able to view the produced depth map and download it if required.
Participating Actor	User
Preconditions	The produced depth map should be the same dimensions of the original input image and should accurately depict the pixel-wise depth estimates of the original image fed to the system.
Extended Use Cases	View 3D Point Cloud
Included Use Cases	None
Main Flow	1. Get the predicted depth map from the depth estimation network. 2. Present the depth map to the user in the application UI and provide the ability to download it by clicking on a button.
Alternate Flow	None
Exceptional Flow	None
Post Conditions	The depth map will be saved to their local storage and will be able to be used for their intended purpose.

Use Case #3	Send Depth Map to ADCU
Description	The predicted depth map should be made available for ACDU to be used in processes such as localisation and perception along with the inputs of other sensors.
Participating Actor	AV
Preconditions	The produced depth map should be the same dimensions of the original input image and should accurately depict the pixel-wise depth estimates of the original image fed to the system.
Extended Use Cases	None
Included Use Cases	None
Main Flow	3. Get the predicted depth map from the depth estimation network. 4. Send it to the ACDU to be used in localisation and other perception tasks.
Alternate Flow	None
Exceptional Flow	None
Post Conditions	ACDU will use the depth map to understand how far away objects and obstacles are on the surrounding and localisation and other perception tasks .

Use Case #4	Optimise modal
Description	Machine learning engineers would closely keep an eye on the system performance and would optimise the system to overcome failure cases.
Participating Actor	ML Engineers

Preconditions	Poor performance or failure cases should be monitored and identified
Extended Use Cases	None
Included Use Cases	None
Main Flow	1. ML engineers failure cases. 2. Debug and analyse the system to identify the root cause. 3. Plan a solution to overcome the failure case
Alternate Flow	None
Exceptional Flow	None
Post Conditions	Optimised models will be deployed and used to estimate depth maps in adverse weather conditions.

Table 13: Use case descriptions

4.10 Requirements Specification

This section defines the functional and nonfunctional requirements from the identified requirements of the system. The requirements are prioritised using the MoSCoW prioritisation technique.

4.10.1 Functional Requirements

ID	Requirement	Priority Level	Use Case Mapping
FR1	Users should be able to input an adverse weather image to system to predict a depth map	M	Input Image
FR2	Input should be validated before feeding into the model	M	Input Image
FR3	System should function by only using single camera images.	M	Produce Depth Map

FR4	The depth estimation network should produce consistent and accurate depth estimations	M	Produce Depth Map
FR5	The produced depth maps should be the same dimensions as the original image	M	Depth Map Enhancement
FR6	The depth maps should have very minimal to no negative effects from the adverse weather conditions	M	Produce Depth Map
FR7	The system should be able to produce accurate depth maps from rain, fog, snow and clear images.	M	Produce Depth Map
FR8	Users should have the ability to save the produced depth maps.	S	Download Depth Map
FR9	Users should have the ability to visualise the depth maps in multiple styles.	S	Download Depth Map
FR10	Users should have the ability to visualise a 3D point cloud from the produced depth map.	C	View 3D point cloud

Table 14: Functional requirements

4.10.2 Non-Functional Requirements

ID	Requirement	Description
NFR1	Accuracy	The system must produce accurate and consistent depth predictions as autonomous vehicles would totally rely on this system to understand the distance of objects ahead of the vehicles, so less accurate systems would result in unforeseen consequences.
NFR2	Performance	The system should produce depth estimates very quickly, as each frame will need to be processed when a video is fed to the system. Also, the system should not be resource-heavy, as inference will need to be done on the fly.

NFR3	Reliability	The weather conditions could constantly change, so the system needs to function equally well in all weather conditions, as unreliability and inconsistency in-depth estimates of certain weather conditions could bring disastrous outcomes.

Table 15: Non-functional requirements

4.11 Chapter Summary

This chapter provided an in-depth review of how the requirements of the proposed system were gathered from the stakeholders of the system using several elicitation techniques such as literature review, observing existing systems, interviews and prototyping. Finally, the chapter concluded by defining the functional and non-functional requirements of the system based on gathered requirements and the gathered requirements were prioritised according to the importance of each requirement to the system.

CHAPTER 05: SOCIAL, LEGAL, ETHICAL AND PROFESSIONAL ISSUES

5.1 Chapter Overview

This chapter provides an overview of the social, legal, ethical and professional issues that may likely occur during the development of the proposed system and defines the mitigation procedure for such issues.

5.2 SLEP Issues And Mitigation

Issue	Mitigation
Social	– The details of the technical and domain experts interviewed were included in the thesis with their consent. – Details of all technical and domain experts obtained were only used for the intended purpose; no details were shared with any third party. – No religious, ethnic or political bias was followed during the development of the project.
Legal	– All third-party tools and frameworks used for developing the proposed system were used adhering to the rules and regulations imposed by the GPL licence. – The proposed system was developed strictly following the General Data Protection Regulation (GDPR).
Ethical	All authors of other researches or existing similar systems that information gathered or code were borrowed during the research and development of the project were rightfully credited at all times.
Professional	– Research guidelines and rules and regulations imposed by the institutions on the Final Year Project were strictly followed. – All software and tools used to develop the system were open-source or under student licence provided by the institution. – Critical analysis of existing work was done without causing any alterations or making false accusations regarding the work.

	– The limitations of the project are clearly mentioned, and all code and complete documentation of the system is publicly available on GitHub for anyone to easily continue the project.

Table 16: SLEP issues and mitigation procedures

5.3 Chapter Summary

This chapter explained in detail all social, legal, ethical and professional issues that could be caused and clearly stated how a procedure was taken to mitigate such issues during the development of the project.

CHAPTER 06: SYSTEM ARCHITECTURE & DESIGN

6.1. Chapter Overview

This chapter provides a detailed overview of the architectural design of the proposed system. The design choices made throughout the development process of the proposed system will be descriptively illustrated using various high-level and low-level diagrams along with concise and clear justifications.

6.2. Design Goals

Design Goal	Description
Correctness	Correctness is a crucial factor for the proposed system as the system's primary goal is to produce depth estimates in adverse weather conditions with higher accuracy and quality than the state-of-the-art.
Performance	Similarly to correctness, the system should be performant. The inference time of the system should be fractions of a second and should be able to be performed at a low computational cost.
Scalability	The system should be built so that new features should be added without much effort, and the system should be able to be trained on new datasets with new weather conditions that have not been seen before.
Reusability	The system should be modularized so that the components, such as the depth estimation network, image enhancement module, etc., could be decoupled from the system and be used as extensions to other systems.

Table 17: Design goals

6.3. System Architecture Design

6.3.1. Tiered Architecture

The system is designed in a 3-tier architecture which includes the data, logic, and presentation tiers. Following a tiered architecture allows to scale and upgrade each component separately much more easily than when the system is tightly coupled. Tiered architecture is also considered to be more secure than other architectures due to its decoupled nature. The main contributions of the research lie within the logic and data tier, which acts as a backbone

to the presentational tier. The diagram and the explanation following it provide a much clearer idea of each tier and its subcomponents.

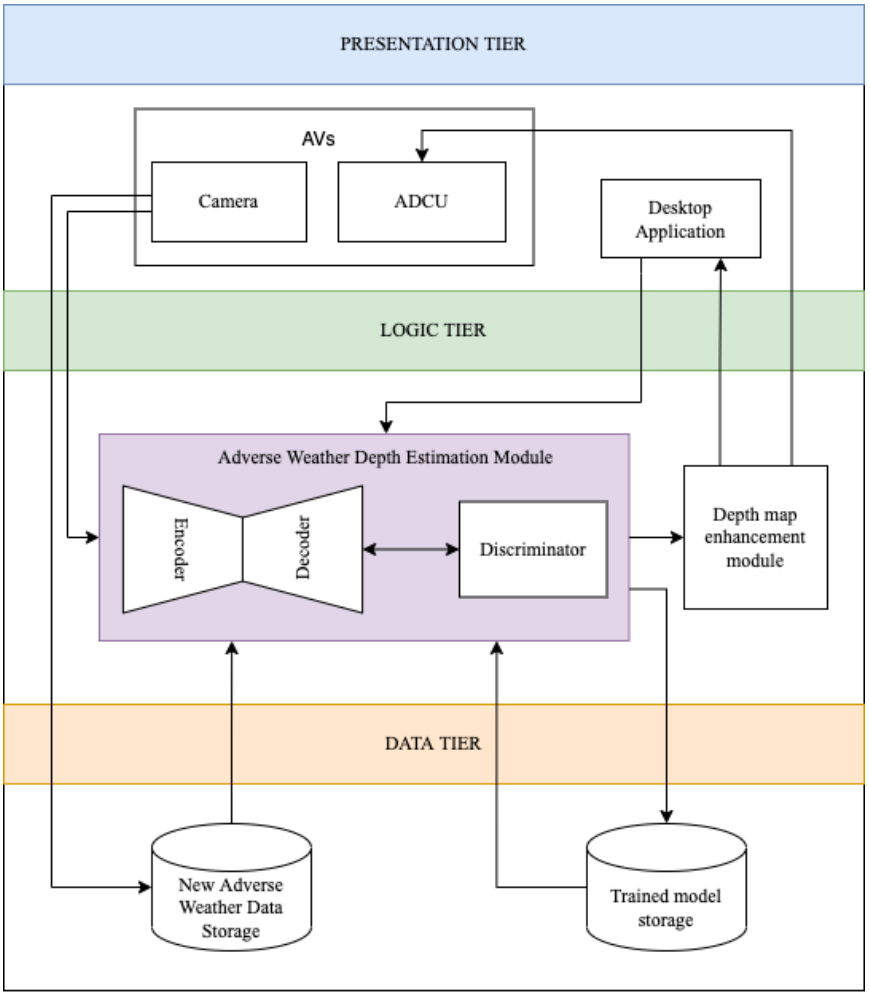


Figure 7: Tiered architecture diagram

Data tier

- Trained model storage- Trained models which predict depth maps using images captured under adverse weather conditions are stored here.
- New Adverse Weather Data Storage- The new images captured by AVs should be stored in the data tier until it is used for training and further enhancing the model.

Logic tier

- Depth estimation module - This is the system's primary and most crucial part. The depth maps will be produced using the input from cameras or the user's input.

- Depth map enhancement module- The depth maps produced will be of half the original scale and will be normalized to be values of between 0 to 255, so depth maps are upscaled and transformed to be in the range of max and min depth.

Presentation tier

- Desktop application - Users can add an image to the application, and the depth map for the input image will be produced. The produced depth map will then be able to be saved or converted to a 3D point cloud if they wish to.
- Camera Input Module - Cameras in AVs capture images and input them into the system. The images will also be saved in the data tier for training the model.
- ADCU Module - ADCU in AVs receive fully processed depth maps from the enhancement module for decision making.

6.4. System Design

6.4.1. Choice Of The Design Paradigm

Choosing the right paradigm for the project is a crucial task in the software development process. The right paradigm for a project depends solely on the nature of the project and its various factors, such as the type of software being developed, various requirements brought in by the users, and the time constraint for the development of the project. The most prominent design paradigms that are being heavily used in industry and academia for software development tasks are Structured Systems Analysis and Design Method (SSADM) and Object-Oriented Analysis and Design Method (OOADM). SSADM was chosen as the preferred design paradigm by taking the following points into consideration:

- SSADM mainly focuses on the process and procedures of the system and allows concentrating more on developing its functionalities rather than focusing on the data structures and real-world objects.
- SSADM allows users to develop the software modularly and integrate it much more efficiently in a structured manner than OOADM, where the system will be mapped together in an object-based manner.

6.4.2 Data Flow Diagram(DFD)

a. DFD Level - 1

The following diagram illustrates all the major components and their relationship with each other. Each of these components is modeled according to the flow of data through the system from start to finish.

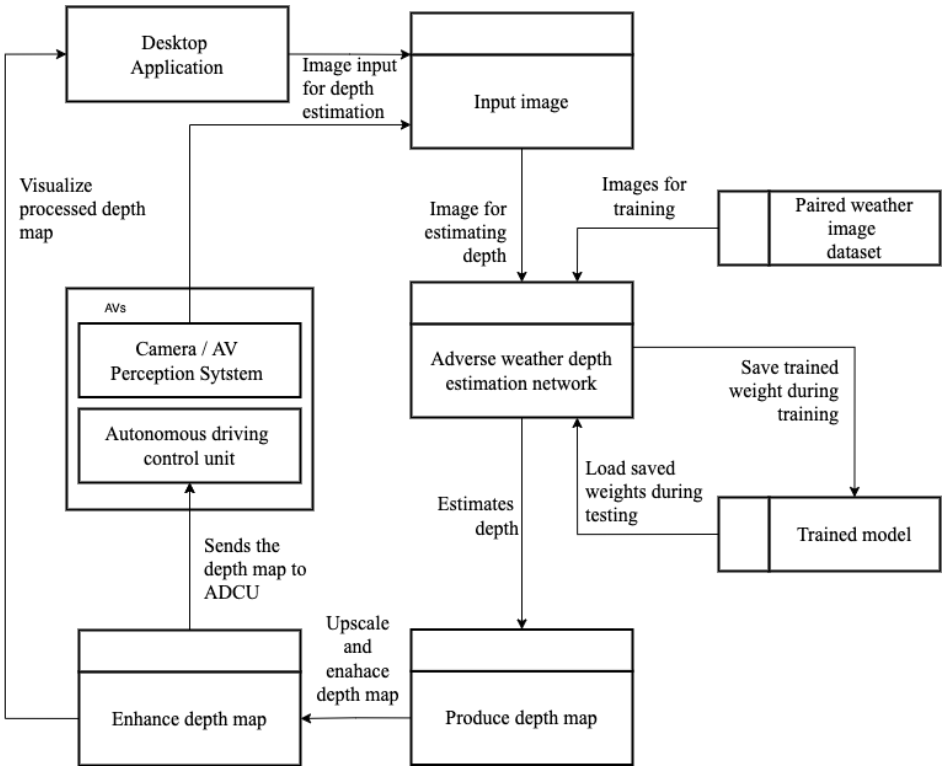


Figure 8: Data Flow Diagram Level 1

b. DFD Level - 2

The following diagram illustrates the depth estimation component of the system shown in the DFD Level 1 diagram above, which plays the most crucial part in the proposed system. The relationship between various sub-components and the flow of data among them are illustrated in the following diagram.

Adverse weather depth estimation module - The depth estimation network is composed of an autoencoder paired with a discriminator trained in an adversarial manner to enhance the depth estimation performance on the input images.

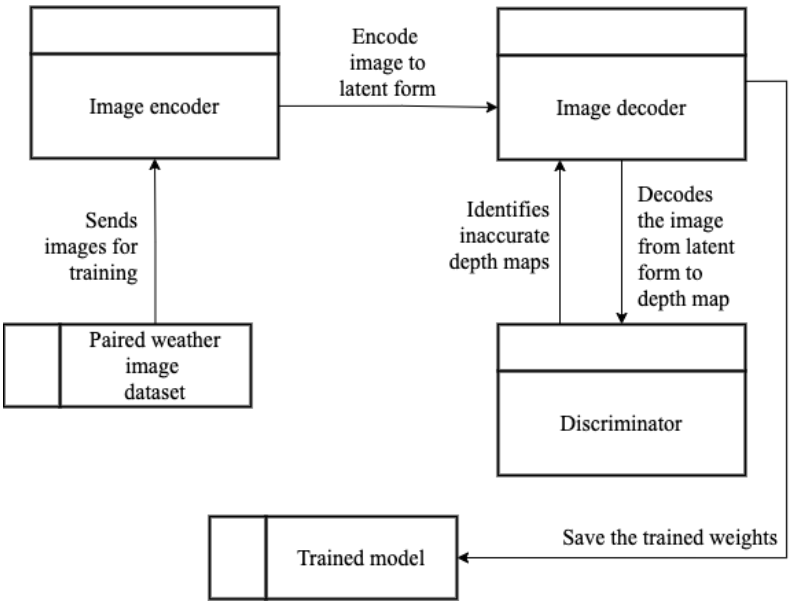


Figure 9: Data Flow Diagram Level 2 Module 2

6.4.5 Sequence Diagram

The diagram below illustrates the contributions from each of the major components in the proposed depth estimation system in the order in which they take place, starting once the user inputs an image and end once the processed depth map is displayed to the user.

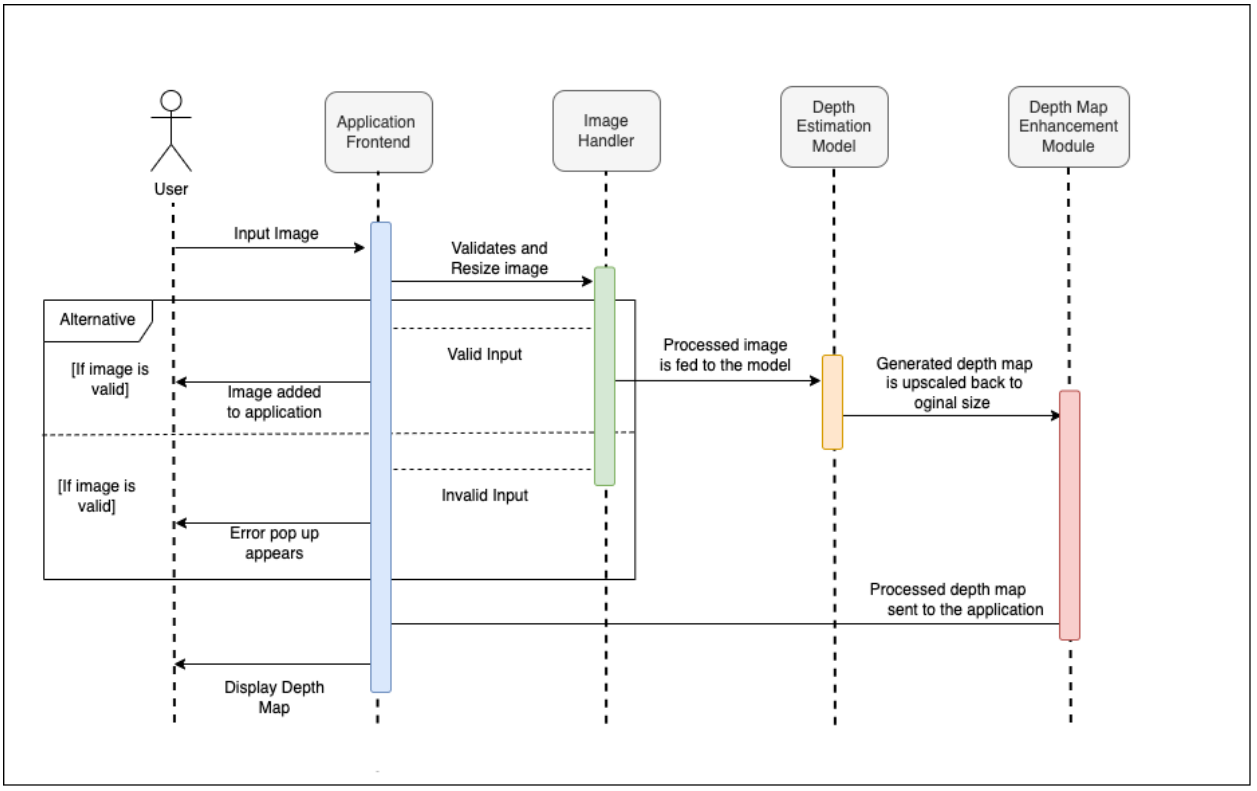


Figure 10: Sequence Diagram

6.4.6 Algorithm Design

6.4.6.1. Overall Architecture

Supervised monocular depth estimation can be achieved by using a variety of architectures. During the implementation of the project, the author initially tried utilizing an ordinary generative adversarial network. After some training epochs, it was noticed that the model suffered from mode collapse, where the generator continuously learns a mapping function to fool the discriminator by producing the same output for any input image. Shown below are some depth maps suffering from mode collapse.

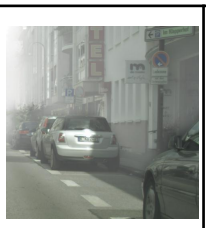
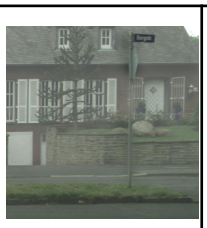
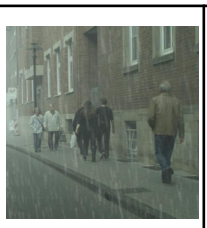
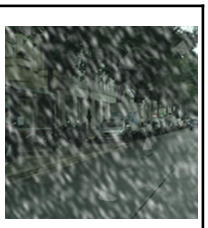
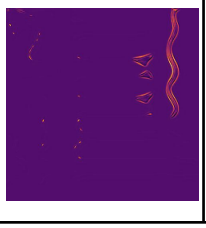
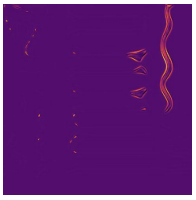
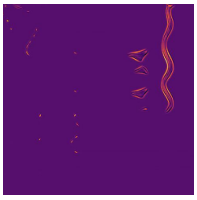
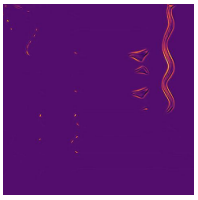
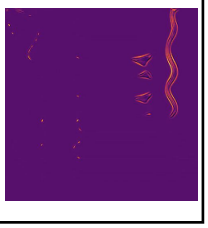
Input					
Output					

Table 18: Mode collapse

One set of model architectures that avoids mode collapse and learns how to reconstruct the data is an autoencoder. As it was explained in the Review of Algorithms section in the Literature Review chapter, autoencoders are composed of two sub-neural networks called the encoder and decoder. The encoder compresses the data to a lower-dimensional representation called a latent code, and the decoder decompresses/reconstructs the data to the required output, in this case to an image identical to the ground truth depth map fed to the network. An inherent limitation of autoencoders is that autoencoders utilize an L1 (Mean Absolute Error) or L2 (Mean Squared Error) reconstruction loss which averages out pixel values over surrounding pixels and generates sub-optimal blurry output. If we were to produce sharper, crispier output, the best-suited architecture would be Generative Adversarial networks, as GANs are well reputed for superior reconstruction capabilities over autoencoders. So, to sum up, using either architecture, an ordinary plain GANs, or an ordinary plain autoencoder would introduce limitations to the project; therefore to obtain the best of both worlds and mitigate the limitations introduced by each architecture, the author proposes using an adversarially trained

autoencoder-based approach for robust depth estimation following the work done by (Makhzani et al., 2016). A discriminator is added to the end of the autoencoder during training so that whenever the autoencoder produces blurry, washed-out output, it will be picked up by the discriminator, and a significant loss value will be added to the overall loss function penalizing the result causing the autoencoder to improve further. So, in the end, a novel architecture that combines the reconstruction power of GANs and the sampling power of autoencoders to produce a generative model that does not suffer from mode collapse was introduced to the task of monocular depth estimation. The diagram below illustrates the proposed architecture.

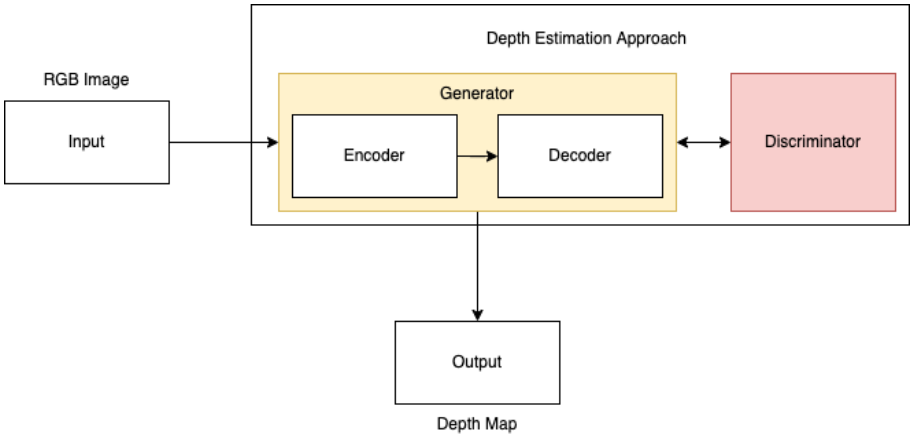


Figure 11 : Proposed architecture

6.4.6.2. Choice Of Generator

After studying existing literature that utilizes autoencoder-based approaches for depth estimation, it was found that a U-Net autoencoder would be ideal for the generator in the proposed architecture. After experimenting with several approaches that utilize a U-Net, the architecture proposed by (Alhashim and Wonka, 2019) was chosen as the ideal architecture for the generator as it would make use of transfer learning and train with less computation power compared to most other U-Net-based architectures. Two pretrained models were tested for the encoder, the Densenet-169 and MobileNet V2 model, out of which DenseNet gave the best results and was chosen as the preferred pretrained model for the encoder. The diagram below breaks down the layers and illustrates the concatenation process of the output features and skip connections forming the U-Net that is used for the generator of the proposed system.

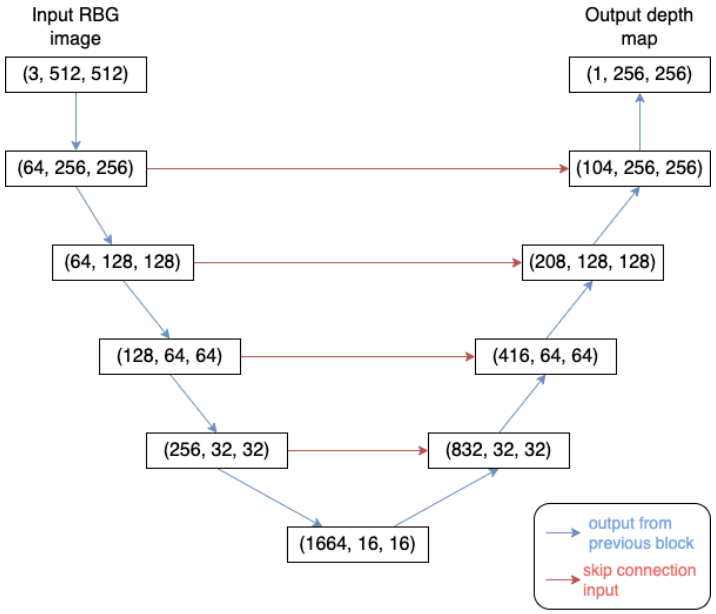


Figure 12 : U-Net generator breakdown

Note that the U-Net shown in the diagram above produces the output depth map at half the scale of the original input. This was done on purpose to reduce computation complexity without losing quality in the final output depth map following the approach of (Alhashim and Wonka, 2019).

After extensive experimentation and with the knowledge gathered from existing literature, the optimal loss function for the generator composed was found to be a combination of 3 loss functions L1 (Mean Absolute Error) loss, SSIM (Structural Similarity Index Measure) loss, and GAN loss of BCE(Binary Cross Entropy). More details and results obtained by experimenting with various loss functions are provided in the Model Testing section in the Testing chapter of this thesis.

6.4.6.3. Choice Of Discrimintor

Patch Discriminator was first proposed for image-to-image translation tasks by (Isola et al., 2018). After experimenting with both an ordinary discriminator and a patch discriminator, it was found that the patch discriminator gives far superior results over an ordinary discriminator. A patch discriminator divides the image into patches and returns a probability for each patch being real or fake, forming a matrix of probabilities instead of a single probability for the whole image being real or fake as it is done with ordinary discriminators. Following the discriminator architecture proposed in (Isola et al., 2018), the patch size was set to be 70x70 pixel patches, which gives the optimal results for image translation tasks such as

monocular depth estimation. The loss function used for the patch discriminator was BCE (Binary Cross Entropy) loss which was ideal for distinguishing the differences between the real and fake images.

6.4.7 UI Design

The wireframe of the desktop application built is shown below. Using the application, a user can feed an image into the system and obtain its depth map. The UI will no longer be part of the system once the system is deployed in an AV; the ACDU will directly process the output from the proposed system.

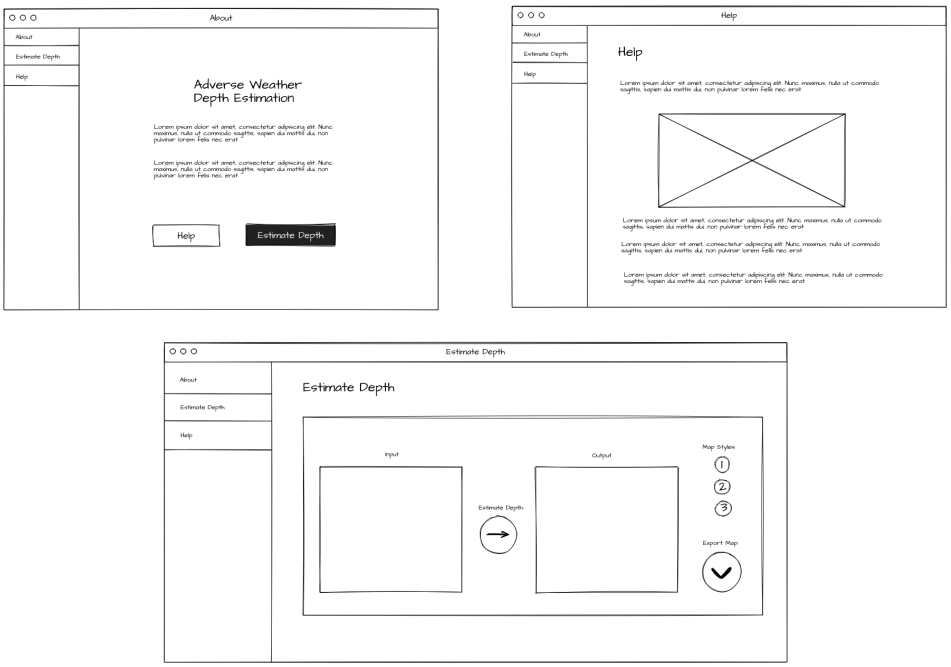


Figure 13: UI Design

6.5. Chapter Summary

This chapter provided an in-depth overview of the design and architecture of the proposed system. The chapter started by clearly defining the design goals, followed by a tiered architecture diagram in which all system functionalities were illustrated in a 3-tiered architecture depicting how each functionality is wired up internally within the system. The choice of the design paradigm was then justified, and data flow diagrams were then used to visualize the flow of data through the system and significant components of the system. Finally, the chapter concluded with an illustration of the UI wireframe of the proposed system.

CHAPTER 07: IMPLEMENTATION

7.1. Chapter Overview

This chapter provides an in-depth review of the implementation process of the proposed system. The chapter will provide justifications for the critical decisions made and the choice of tools, datasets, and various other libraries used during the implementation process. Code snippets for important functionalities, along with detailed explanations of procedures taken to solve problems faced during the implementation process, will be provided in this chapter.

7.2. Technology Selection

7.2.1. Technology Stack

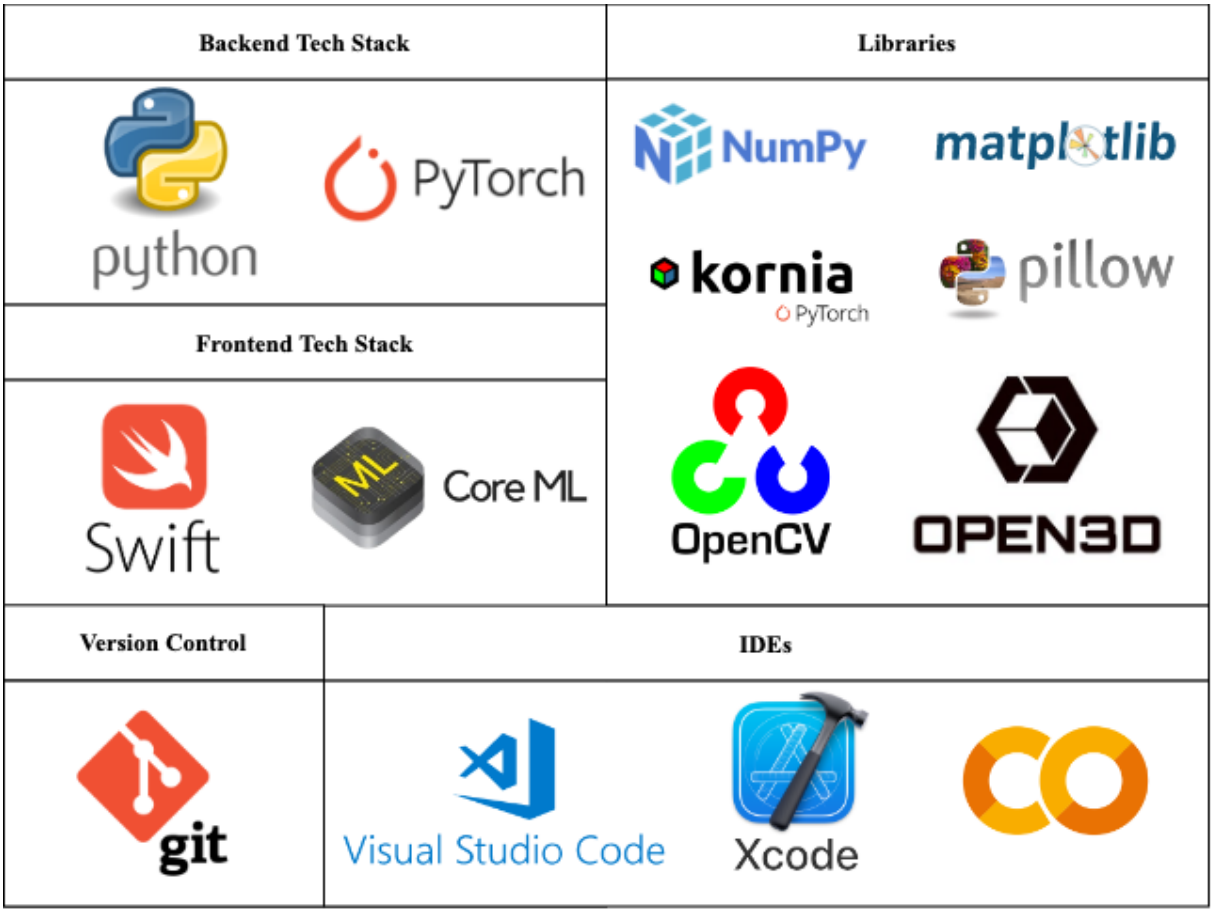


Figure 16: Technology stack

7.2.2. Data Selection

As this is one of the first attempts to address multiple adverse weather conditions using a single approach, a custom dataset was created by combining several augmented versions of the Cityscapes dataset with rain, snow, and fog weather conditions. The dataset consisted of

real-world images, augmented ultra realistic images under various adverse weather conditions of rain, snow, and fog, and corresponding ground truth depth maps produced from high fidelity lidar sensors. Accurate ground truth data was necessary as the proposed depth estimation system is a supervised learning approach, and the system's output depth maps directly depend on the quality of the ground truth supervision data provided to the model during training. The various versions of the cityscape dataset used are as follows:

- CityScapes clear image dataset (Cordts et al., 2016)
- CityScapes foggy image dataset (Sakaridis, Dai and Van Gool, 2018)
- CityScapes rainy image dataset (Hu et al., 2019)
- CityScapes snowy image dataset (Zhang et al., 2021)

7.2.3. Selection Of Development Framework

PyTorch was chosen as the preferred framework for implementing the proposed system among the various available development frameworks for deep learning projects. Pytorch is an optimized tensor processing framework based on the Torch framework built using the Python programming language. Created by a group of developers at Facebook, soon it was favored by scientists, developers, and neural network debuggers mainly due to its usage of dynamic computation graphs and its completely Pythonic nature, bringing tough competition to its predecessors such as TensorFlow by Google (Heller, 2022). Pytorch also supports strong GPU acceleration support and Automatic Differentiation for creating and training deep neural networks, which has been a critical factor to elevate it from its competitors (Heller, 2022). Unlike other deep learning frameworks, PyTorch allows advanced network enhancements and allows the code to be structured more logically, which helps deepen the understanding of how the neural network actually works. Like other deep learning development frameworks, PyTorch provides easy-to-understand abstracted high-level functions. These abstracted high-level functions make writing neural networks and modifying them according to our needs much more straightforward and help developers focus more on tasks related to the system's logic and underlying architectures rather than solely being distracted by the Python syntax and basic constructs.

7.2.4. Programming Language

Among various programming languages used in the industry for deep learning, such as Python, Lua, C#, C/C++, etc., Python stands out as the most prominent language of choice for deep

learning applications, leading language popularity charts by significant margins. Python was chosen as the preferred programming language for implementing the proposed system because, compared to other languages, Python offers inequivalent advantages, particularly for machine learning-related tasks. Some of them are as follows:

- Extensive Collection of Libraries and Packages - libraries built into the python programming language greatly help users on machine learning tasks by avoiding the need to rewrite code for most basic to advanced data manipulation functions, which leads to massive reductions in the development time and increase in productivity.
- Code Readability - code readability is a vital part of a programming language; Python excels in this very well. The short, concise, and readable classes offered by Python and its simple syntax boost the code readability and help users focus on the program logic instead of worrying about complex language syntax and constructs, which most other competing languages do not offer.
- Flexibility - unlike other languages, Python offers multi-paradigm programming and allows users to use procedural, functional, object-oriented, or imperative programming for their projects.

7.2.5. Libraries Utilised

The following table briefly describes the most commonly used libraries in the project, along with justifications for their use in the project implementation.

Library	About	Justification
matplotlib	A python based statistical data visualisation library	Matplotlib was used in the project to plot images and charts for comparison purposes.
numpy	A python based data manipulation library.	Numpy was used in the project to perform image vector and array manipulations and image tensor conversions.
pillow	A python based library for opening, manipulating and saving images.	Pillow was used in the project to open images from datasets, perform basic manipulations such as cropping and save processed images and depth maps to disc.

kornia	An opencv-python based library which allows to integrate classical computer videos functionalities into deep learning models	Kornia was used to access loss functions such as SSIIM and PSNR which were passed into the model while training to optimise the model.
tensorboard	A library by Google Tensorflow which provides the visualisation and tooling needed for machine learning experimentation	Tensorboard was used to observe the model training behaviour.

Table 19: Libraries utilised

7.2.6. IDEs Utilised

VSCode was chosen as the preferred IDE for developing the project backend and machine learning component due to several benefits over other Python IDEs. VSCode provides all capabilities and functionalities of a full-fledged IDE such as Pycharm while being several times lighter. It also offers quick and easy integration with remote servers, which most other IDEs lack. Various remote GPUs, especially Google Colab and Saturn Cloud servers, were heavily utilized during the training process, which provided the high graphical processing power required to accelerate the training of the deep generative model-based architecture of the proposed system. During the development of the frontend application of the proposed system, XCode was primarily used as the preferred IDE of choice. Xcode stands out as the best IDE with complete, up-to-date support for developing all iOS and macOS features and supports Objective-C and Swift programming languages.

7.2.7. Summary Of Technology Selection

Component	**Tools**
Backend Development Tech Stack	Python, Pytorch
Frontend Development Tech Stack	Swift (UIKit), Coreml

Libraries	Matplotlib, numpy, pillow, kornia, OpenCV, Open3D
IDE	VSCode, Xcode, Google Colab
Version Control	Git

Table 20: Summary of technology selection

7.3. Implementation Of Core Functionalities

As explained in the Algorithm Design section in the Design chapter, the architecture of the proposed system consisted of a U-Net autoencoder combined with a patch discriminator to maximize the generalization capabilities of the system to the input images from various weather conditions and improve the overall depth map quality.

7.3.1 Encoder Implementation

The encoder utilizes transfer learning using the Densenet-169 (Huang et al., 2018) pretrained model, with the weights obtained from training on the Imagenet dataset.

```python
# Encoder uses the Densenet-169 pre-trained on Imagenet dataset
class Encoder(nn.Module):
    def __init__(self):
        super(Encoder, self).__init__()
        self.original_model = models.densenet169( pretrained=True )

    def forward(self, x):
        features = [x]

        ''' Starting with the input image, the output received by feeding the last stored feature into each module of the Densenet
        model is stored in features list '''

        for name, module in self.original_model.features._modules.items():
            features.append( module(features[-1]) )
        return features
```

✓ 0.3s Python

Figure 14: Encoder Implementation

7.3.2 Decoder

The decoder was built with up-sample blocks of convolutional and Leaky Rectified Linear Unit (leakyRelu) layers with 2x bilinear upsampling of the feature output from the previous block concatenated with output from skip-connections from the encoder block. The skip connection ensures that no spatial information is lost during the decoding process and that the output depth map is of the exact spatial dimensions as the input image. Shown in Figure 15 is a single up-sample block used in the decoder.

```python
# Upsample block with convolutional layers concatenated with 2x bilinear upsampling and skip connections
class UpSample(nn.Sequential):
    def __init__(self, input_features, output_features):
        super(UpSample, self).__init__()
        self.convA = nn.Conv2d(input_features, output_features, kernel_size=3, stride=1, padding=1)
        self.leakyreluA = nn.LeakyReLU(0.2)
        self.convB = nn.Conv2d(output_features, output_features, kernel_size=3, stride=1, padding=1)
        self.leakyreluB = nn.LeakyReLU(0.2)

    # On each forward pass the input from skip connection is upscaled and concatenated with output from previous block
    def forward(self, x, concat_with):
        up_x = F.interpolate(x, size=[concat_with.size(2), concat_with.size(3)], mode='bilinear', align_corners=True)
        return self.leakyreluB( self.convB( self.convA( torch.cat([up_x, concat_with], dim=1)  ) )  )
```

✓ 0.2s Python

Figure 15: A single upsample block from the decoder

Shown below in Figure 16 is the complete decoder model with line-by-line comments explaining the concatenation process of the output features and skip connections forming the U-Net.

```python
class Decoder(nn.Module):
    def __init__(self, num_features=1664, decoder_width = 1.0):
        super(Decoder, self).__init__()

        # Defines the first conv2d layer to transform the output from encder depending on the size of the decoder / decoder_width
        features = int(num_features * decoder_width)
        self.conv2 = nn.Conv2d(num_features, features, kernel_size=1, stride=1, padding=0)

        # Defines the upsample blocks with input_features = output_features from previous layer + features from skip connection
        self.up1 = UpSample(input_features=features//1 + 256, output_features=features//2)
        self.up2 = UpSample(input_features=features//2 + 128,  output_features=features//4)
        self.up3 = UpSample(input_features=features//4 + 64,  output_features=features//8)
        self.up4 = UpSample(input_features=features//8 + 64,  output_features=features//16)

        # Defines final conv2d layer to transform the number of channels from 104 to 1
        self.conv3 = nn.Conv2d(features//16, 1, kernel_size=3, stride=1, padding=1)

    def forward(self, features):
        # Skip connections from encoder
        x_block0, x_block1, x_block2, x_block3, x_block4 = features[3], features[4], features[6], features[8], features[12]
        # First conv2D layer transforms the output of the encoder based on the size of the decoder / decoder_width
        x_d0 = self.conv2(F.relu(x_block4))
        # First Upsample Block: concats output from encoder of shape (1664, 16, 16) with (256, 32, 32) block from encoder  ---> (832,32,32)
        x_d1 = self.up1(x_d0, x_block3)
        # Second Upsample Block: concats output of first upsample block with (128, 64, 64) block from encoder  ---> (416,64,64)
        x_d2 = self.up2(x_d1, x_block2)
        # Third Upsample Block: concats output of second upsample block with (64, 128, 128) block from encoder  ---> (208,128,128)
        x_d3 = self.up3(x_d2, x_block1)
        # Fourth Upsample Block: concats output of third upsample block with (64, 256, 256) block from encoder  ---> (104,256,256)
        x_d4 = self.up4(x_d3, x_block0)
        # Final conv2D layer transforms the output from fourth upsample layer into feature vector of shape (1,256,256)
        return self.conv3(x_d4)
```

✓ 0.3s Python

Figure 16: Decoder Implementation

7.3.3 Generator Implementation

Shown in Figure 17 is the generator of the proposed Adversarial Network, which consists of the U-Net autoencoder explained above.

```python
# The Unet autoencoder formed by using the encoder and decoder is used as the generator
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.encoder = Encoder()
        self.decoder = Decoder()

    def forward(self, x):
        out2 = self.encoder(x)
        out3 = self.decoder(out2)
        return out3
```

✓ 0.4s Python

Figure 17: Generator Implementation

7.3.4 Discriminator Implementation

Shown in Figure 18 is the implementation of the Patch Discriminator with clear step-wise explanations for each line of code.

```python
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        # In channels count is equal to the sum of channels in feature vector x and y
        in_channels = 4

        ''' Defining the CNN blocks with conv2D layers with output channels(filters) of size 64, 128, 256, 512
        and a kernel size of 4 and stride of 2 except for the last block in order to form a patch size of 70x70 '''

        ''' All CNN blocks except for the first block will have a conv2D layer followed by a batch normalization
        and a leaky relu layer '''
        self.block_0 = nn.Sequential(
            nn.Conv2d(in_channels, out_channels=64, kernel_size=4, stride=2, padding=1, padding_mode='reflect'),
            nn.LeakyReLU(0.2)
            )
        self.block_1 = CNNBlock(in_channels=64, out_channels=128, stride=2)
        self.block_2 = CNNBlock(in_channels=128, out_channels=256, stride=2)
        self.block_3 = CNNBlock(in_channels=256, out_channels= 512, stride=1)
        self.block_4 = nn.Conv2d(in_channels=512, out_channels=1, kernel_size=4, stride=1, padding=1, padding_mode='reflect')


    def forward(self, x, y):
        # The predicted image x is concatenated with the ground-truth/ excepted image and passed into the discriminator blocks
        x = torch.cat([x, y], dim=1) #  Shape of x = (4, 512, 512)
        '''
        As mentioned in this thread "https://github.com/phillipi/pix2pix/issues/49#issuecomment-310819127"
        The patch size of the discriminator can be calculated as follows:
            patch_size = (output_size - 1) * stride + kernel_size
        '''
        x_0 = self.block_0(x)   #  Shape of x_0 = (64, 256, 256)    Patch_size = [(34-1) * 2 + 4] = 70
        x_1 = self.block_1(x_0) #  Shape of x_1 = (128, 128, 128)   Patch_size = [(16-1) * 2 + 4] = 34
        x_2 = self.block_2(x_1) #  Shape of x_2 = (256, 64, 64)     Patch_size = [(7-1) * 2 + 4] = 16
        x_3 = self.block_3(x_2) #  Shape of x_3 = (512, 63, 63)     Patch_size = [(4-1) * 1 + 4] = 7
        x_4 = self.block_4(x_3) #  Shape of x_4 = (1, 62, 62)       Patch_size = [(1-1) * 1 + 4] = 4
        return x_4

# Each CNN Block consists a conv2D followed by a BatchNorm and lealy relu layer
class CNNBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride):
        super(CNNBlock, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=4, stride=stride, padding=1, bias=False, padding_mode='reflect'),
            nn.BatchNorm2d(out_channels),
            nn.LeakyReLU(0.2)
            )

    def forward(self, x):
        return self.conv(x)
```

✓ 0.4s Python

Figure 18: Patch Discriminator Implementation

7.3.5 Generator Loss function

The loss function used for the generator consists of a combination of L1, SSIM, and BCE adversarial loss. After experimenting for several rounds with different weight ratios, the optimal ratio was found to be L1:100, SSIM:10, BCE:1. The experimentation results are provided in the Testing chapter. Shown in Figure 19 is the implementation of the loss function of the generator.

```python
# Binary cross entropy loss is used calculate how likely each image patch is real or fake.
# BCEWithLogitsLoss because it combines BCEloss with a sigmoid layer for better stability.
BCE = nn.BCEWithLogitsLoss()
L1_LOSS = nn.L1Loss()

def generatorLoss(real_image, real_depth):
        # Fake depth map is generated by passing the real image to generator
        fake_depth = gen(real_image)

        '''
        1st Loss: The mean absolute error between fakedepth and the ground truth real depth  is calculated
        and it is multiplied by l1_lamba of 100 to maximize the significance of this this loss over the other.'''
        G_L1 = L1_LOSS(upscale_depth_tensor(fake_depth), real_depth) * configs.L1_LAMBDA

        # The exact matrix of probabilites of real_image and fake_depth
        D_fake = disc(real_image, upscale_depth_tensor(fake_depth))

        '''
        2nd Loss: The binary cross entropy loss of how likely D_Fake is 100% real is taken and
        the generator is tricked into understanding the D_fake probability matrix as being real.
        '''
        G_fake_loss = BCE(D_fake, torch.ones_like(D_fake))

        '''
        3rd Loss: The structural similarity loss
        '''
        structural_similarity = ssim(upscale_depth_tensor(fake_depth), real_depth)
        structural_similarity = torch.clamp((1 - structural_similarity) * 0.5, 0, 1).mean().item()
        SSIM_LOSS =  structural_similarity * configs.SSIM_LAMBDA
        G_loss = SSIM_LOSS + G_L1 + G_fake_loss
```

Python

Figure 19: Generator Loss

7.3.6 Discriminator Loss function

The loss function of the discriminator is an average loss of 2 BCE(Binary Cross Entropy) losses. They are calculated as follows.

- Passing the actual depth and input image to the discriminator and the output probability matrix through a BCE loss function along with a matrix of all ones (denoting maximum probability being real).
- Passing the predicted depth map with the actual image to the discriminator and the output probability matrix is passed into a BCE loss function along with a matrix of all zeros (denoting maximum probability being fake). Shown in Figure 20 is the implementation of the loss function of the patch discriminator.

```python
def discriminatorLoss(real_image, real_depth):
    # Fake depth map is generated by passing the real image to generator
    fake_depth = gen(real_image)

    # The exact matrix of probabilites of real_image and real_depth
    D_real = disc(real_image, real_depth)
    # The exact matrix of probabilites of real_image and fake_depth
    D_fake = disc(real_image, upscale_depth_tensor(fake_depth.detach()))

    # Returns the likelihood the D_real is 100% real (Probability matix with all ones)
    D_real_loss = BCE(D_real, torch.ones_like(D_real))
    # Returns the likelihood the D_fake is 100% fake (Probability matix with all zeros)
    D_fake_loss = BCE(D_fake, torch.zeros_like(D_fake))

    # The final loss of the discriminator will be the average of D_real_loss and D_fake_loss
    D_loss = (D_real_loss + D_fake_loss) / 2
    return D_loss
```

Python

Figure 20: Patch Discriminator Loss

7.3.7 Training The Network

```python
# Start training...
for epoch in range(configs.NUM_EPOCHS + 1):

    loop = tqdm(train_loader, leave=True)
    N = len(train_loader)

    # Switching both models to train mode
    disc.train()
    gen.train()

    # Iterating through the traning dataset
    for idx, sample_batched in enumerate(loop):

        # RGB input image
        image = torch.autograd.Variable(sample_batched['image'].to(configs.DEVICE))
        # Ground truth disparity map
        depth = torch.autograd.Variable(sample_batched['depth'].to(configs.DEVICE))
        # Get the actual depth map from the ground truth disparity map
        depth_n = DepthNorm(depth)

        # Training the patch discriminator
        with torch.cuda.amp.autocast():
            D_loss, D_real = discriminatorLoss(image, depth_n)

        # Updating the discriminator model weights with combined discriminator loss
        disc.zero_grad()
        d_scaler.scale(D_loss).backward()
        d_scaler.step(opt_disc)
        d_scaler.update()

        # Training unet generator
        with torch.cuda.amp.autocast():
            G_loss, D_fake = generatorLoss(image, depth_n)

        # Updating the generator model weights with combined generator loss
        opt_gen.zero_grad()
        g_scaler.scale(G_loss).backward()
        g_scaler.step(opt_gen)
        g_scaler.update()
```

Figure 21: Training the complete adversarial network

7.4. Implementation Of Apis

7.4.1 Converting The Saved Pytorch Model To A Coreml Model

The saved Pytorch model is converted to a CoreML model to support integration with iOS/macOS applications. Shown in Figure 22 is the implementation of the conversion process to CoreML.

```python
# Importing saved weights from checkpoint
def load_checkpoint_state(checkpoint_file, model):
    print('=> Loading checkpoint state')
    checkpoint = torch.load(checkpoint_file, map_location=configs.DEVICE)
    model.load_state_dict(checkpoint['state_dict'])

model = Generator().to(configs.DEVICE)
load_checkpoint_state(configs.CHECKPOINT_GEN, model)
model.eval()
model = model.to("cpu")

# Converting to coreML model
dummy_input = torch.rand(1, 3, 512, 512) # Creating dummy input
traced_model = torch.jit.trace(model, dummy_input) # Tracing the model using the dummmy input
input_image = ct.ImageType(name="image", shape=(1, 3, 512, 512), scale=1.0/255.0) # Creating the input image type
coreml_model = ct.convert(traced_model, inputs=[input_image]) # Converting the model
spec = coreml_model.get_spec() # Loading spec from model
ct.utils.rename_feature(spec, "var_2795", "depthmap") # Renaming the output mutliarray type from spec
coreml_model = ct.models.MLModel(spec) # Updating the model from the updated spec
coreml_model.save('coreML/AWDE_Model.mlmodel') # Saving the CoreML model
```

Python

Figure 22: CoreML conversion

7.5. Implementation Of The User Interface

Shown in Figure 26 are the various screens in the macOS application built.

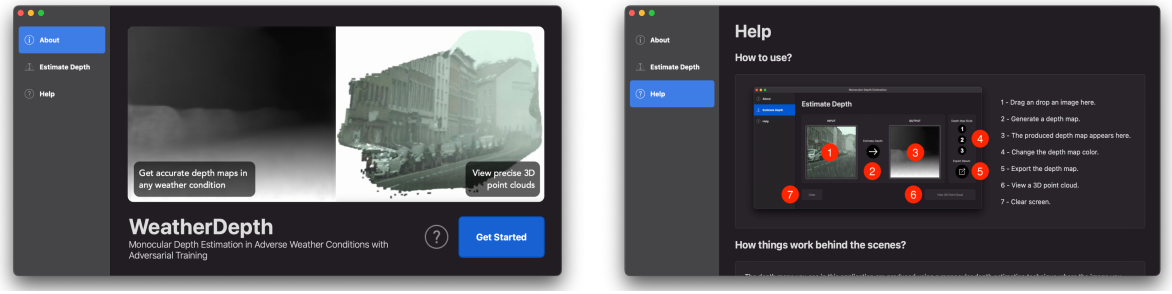


Figure 23: Home and help screens of the MacOS application

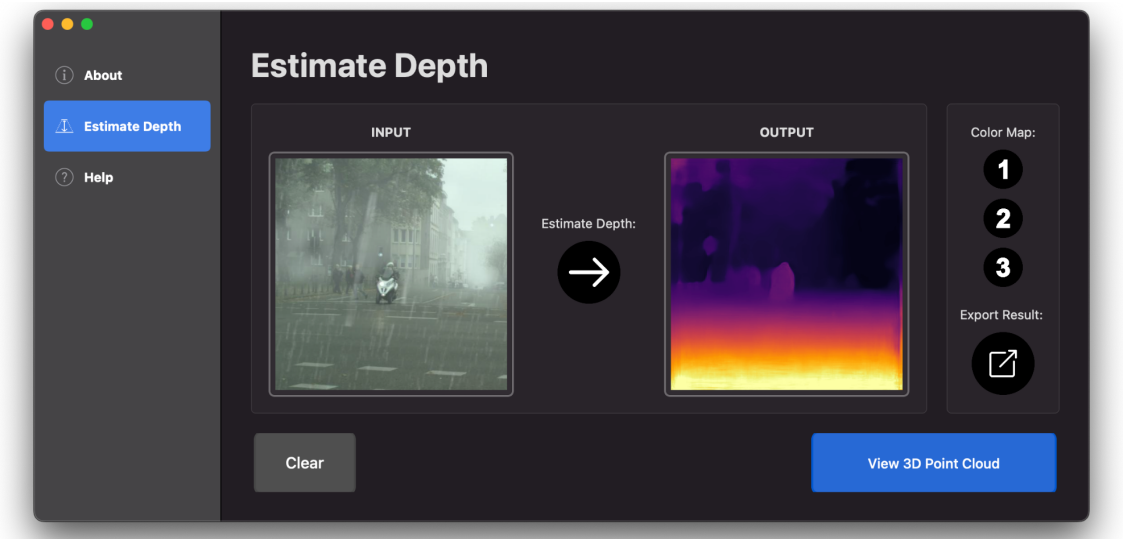


Figure 24: Depth estimation screen of the MacOS application

Shown in Figure 25 is the 3D point cloud screen that visualizes the generated interactive 3D point cloud.

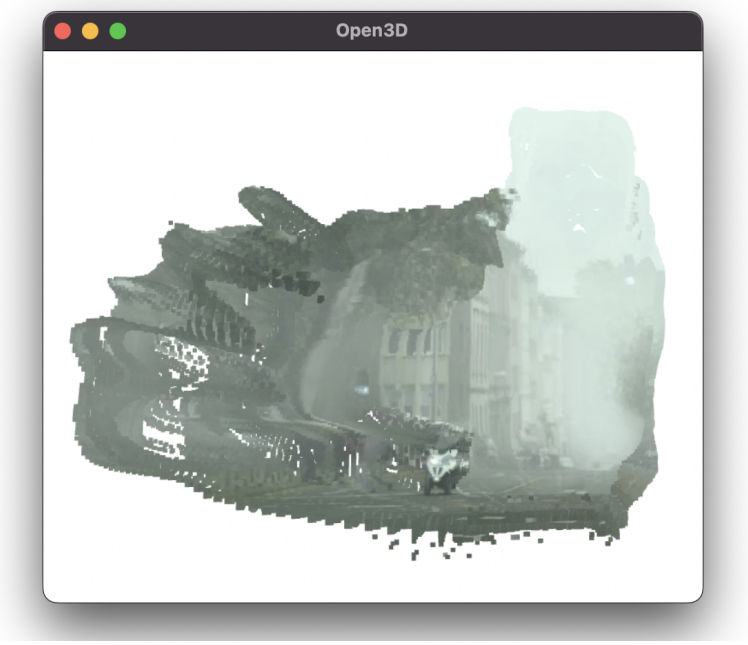


Figure 25: 3D Point Cloud Screen of the MacOS application

7.6. Chapter Summary

This chapter provides a detailed overview of the implementation of the proposed system. Code snippets of the system's core functionalities were provided, and results from the current implementation of the project were discussed.

CHAPTER 08: TESTING

8.1. Chapter Overview

This chapter covers the various tests that were carried out on the developed system, such as model testing to validate the model's performance and accuracy, benchmarking against state-of-the-art approaches in order to compare their performance against the developed system, and functional and non-functional tests to validate its various functional and non-functional requirements that were defined at the beginning of the project.

8.2 Objectives And Goals Of Testing

The main objective of carrying out various tests on the system is to verify if the system covers all the defined functional requirements and if it is performing as expected. The following test goals were defined:

- Validate if all proposed functional requirements are implemented.
- Validate if all proposed non-functional requirements are implemented.
- Verify if coding best practices are followed throughout the development process.
- Make sure the system is bug-free.

8.3. Testing Criteria

All test that is carried out on the system falls under the following criteria:

- Functional Testing - This validates if the system satisfies all the functional requirements, adheres to all information processing standards, and does not contain any defects.
- Structural Testing - This validates the coding standards and program logic and finds errors in data structure usage.

8.4. Model Testing

The model was tested using Absolute Relative Difference (AbsRel), Root Mean Square Error (RMSE), RMSE (log), and Square Relative Error (SqRel). The results obtained by testing the model by using each metric are provided in the following sections.

Dataset	AbsRel	RMSE	RMSE (log)	SqRel
CityScapesWeather	0.116	9.425	0.272	1.897
vKITTI	0.119	4.598	0.289	1.566

Table 21: Results obtained by testing model in various datasets

The CityScapesWeather dataset consists of augmented adverse weather images of rain, snow, and fog weather conditions from real-world clear daytime urban images from the CityScapes dataset. The table below shows some samples of generated depth maps using this research's proposed approach.

Input	GT	WeatherDepth
		
		
		

Table 22: Depth maps obtained by testing on CityScapesWeather dataset

vKITTI is a photo-realistic synthetic dataset commonly used to learn and evaluate several computer vision-related deep learning tasks such as object detection and multi-object tracking, scene-level and instance-level semantic segmentation, optical flow, and depth estimation. The table below shows samples of generated depth maps using the proposed approach.

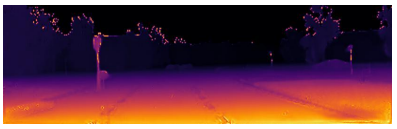
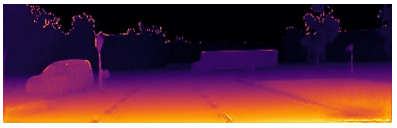
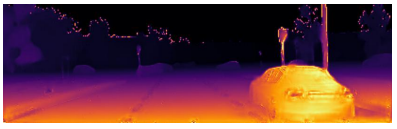
Input	GT	WeatherDepth
		
		
		

Table 23: Depth maps obtained by testing on vKITTI dataset

The results shown in the sections above were obtained using the best-combined loss function, which was found by training the model for several rounds with various loss functions and comparing its performance based on the metrics mentioned above. A list of all results obtained by various combinations of loss functions can be found in Appendix C.

8.5. Benchmarking

Several benchmarking suites exist for clear weather day-time monocular depth estimations, but since adverse weather depth estimation is a significantly less studied subject, there are no existing benchmarking marking suites built explicitly for adverse weather depth estimation. Therefore it was decided to benchmark the proposed approach and the state-of-the-art with vKITTI Dataset, which is not explicitly built for adverse weather depth estimation, but contains some scenes under adverse weather conditions, including rain and fog.

8.5.1 Benchmarking With CityScapesWeather Dataset

Two loss functions were tested, one with Gradient Loss and one without Gradient Loss. The results obtained are provided in the table below. Each model was trained for 15 epochs

with the same loss function, and in both instances, the WeatherDepth model performed better than the DenseDepth model.

Model	**AbsRel**	**RMSE**	**RMSE(log)**	**SqRel**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Generator Loss: L1+SSIM+BCE							
WeatherDepth	0.183	**10.262**	**0.295**	**2.834**	**0.758**	**0.902**	**0.952**
DenseDepth	**0.182**	12.427	0.332	3.086	0.751	0.899	0.949
Generator Loss: L1+SSIM+BCE+Gradient Loss							
WeatherDepth	**0.099**	**9.060**	**0.253**	**1.615**	**0.900**	**0.954**	**0.971**
DenseDepth	0.111	8.562	0.237	1.916	0.869	0.944	0.969

Table 24: Benchmarking on WeatherCityDataset

Shown below are some sample depth maps generated by each model with L1+SSIM+BCE after 15 epochs of training.

Input	**GT**	**DenseDepth**	**WeatherDepth**
			
			

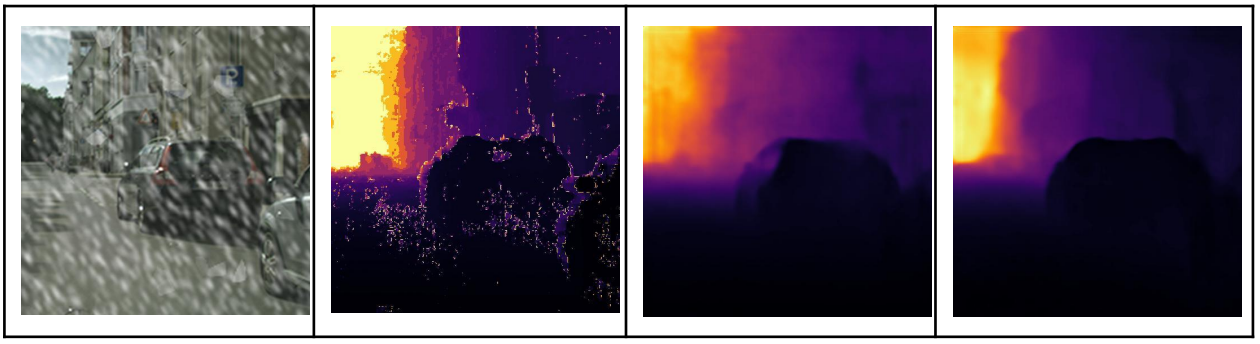


Table 25: Comparison of results different models on CityScapeWeather Dataset

Below are some examples of the depth maps generated by the WeatherDepth model using the 2 generator loss variants.

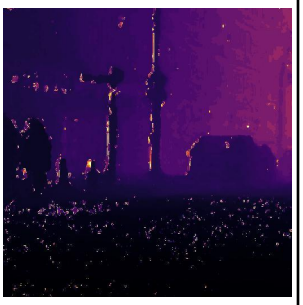
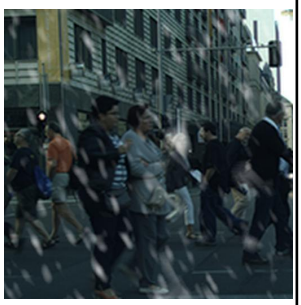
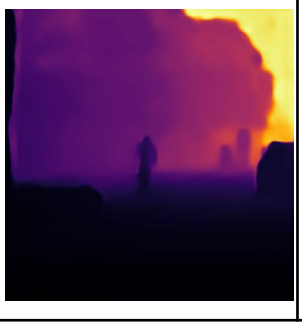
Input	**GT**	**L1 + SSIM + BCE**	**L1 + SSIM + BCE + Gradient Loss**
			
			
			

Table 26: Comparison of results from different generator loss function

As seen in the examples above, the object boundaries are much clearer with the addition of the gradient loss function, and small objects with more minor details are picked up properly. This brought in some drawbacks too. For example, the skies looked distorted most of the time after the addition of the gradient loss; this is mainly due to the noise in the original ground truth data that had been collected via a lidar sensor, and the noise in the depth maps was picked up more than it was before. However, as seen in a comparison of various loss function combinations in Appendix C, the overall depth estimation accuracies were higher with gradient loss, but because far away objects and structures looked significantly distorted, the 3D point clouds generated from these depth maps were not accurate as compared to without gradient loss. Therefore the L1 + SSIM + BCE loss combination was chosen as the preferred combined loss function for the generator of the proposed approach.

8.5.2 Benchmarking With vKITTI Dataset

vKITTI (Humenberger, 2020) dataset is a publicly available depth estimation and image segmentation benchmark suite. The dataset only contains adverse weather images of fog and rain conditions. Similar to benchmarking on the CityscapeWeather Dataset, the system was benchmarked using both generator loss combinations mentioned above. The outcome was very similar to the CityscapeWeather Dataset; the proposed system was outperforming the state-of-the-art with both loss functions, and although the loss function with gradient loss function was performing much better in terms of accuracy and error metrics, the far-away objects were much distorted compared to what was produced without gradient loss. Below is a comparison against the state-of-the-art approaches on the vKITTI dataset using the L1 + SSIM + BCE loss combination for the generator of the proposed approach.

Weather	Model	AbsRel	RMSE	RMSE(log)	SqRel	δ<1.25	δ <1.25^2	δ< 1.25^3
Fog	WeatherDepth	**0.159**	**6.965**	0.391	**1.981**	**0.854**	**0.935**	**0.960**
	SAFENet	0.213	9.018	**0.317**	2.478	0.690	0.872	0.936
	Monodepth	0.218	10.392	0.370	2.823	0.686	0.871	0.919
Rain	WeatherDepth	**0.132**	**4.186**	0.314	1.213	**0.898**	**0.953**	0.972
	SAFENet	0.145	6.349	**0.222**	**1.114**	0.800	0.937	**0.977**

	Monodepth	0.200	6.965	0.263	1.907	0.734	0.901	0.961

Table 27: Benchmarking on vKITTI Dataset

8.6. Functional Testing

All proposed functional requirements of the system are provided in the following table, along with their results.

Test Case	ID	Requirement	Expected Result	Actual Result	Result
1	FR1	Input a valid image to the system.	Successfully gets added to the system	Successfully gets added to the system	Passed
2	FR2	The input image is validated	Input gets validated successfully	Input gets validated successfully	Passed
3	FR3	Produce depth maps using a single image.	Successfully generate a depth map.	Successfully generate a depth map.	Passed
4	FR4	Depth maps generated are accurate with no distortions.	Accurate depth maps with no distortions gets generated	Accurate depth maps with no distortions gets generated	Passed
5	FR5	The scale and dimension of the depth maps in consistent with the input image	Produced depth maps are of same scale and dimension as input	Produced depth maps are of same scale and dimension as input	Passed
6	FR6 , FR7	The system should be able to produce accurate depth maps	Accurate depth maps are produced for all weather conditions	Accurate depth maps are produced for all weather conditions	Passed

		from rain, fog, snow and clear images.			
7	FR8	Users should have the ability to save the produced depth maps.	The produced depth maps gets saved	The produced depth maps gets saved	Passed
8	FR9	Users should have the ability to visualise the depth maps in multiple styles.	User can change the colour maps of produced depth maps	User can change the colour maps of produced depth maps	Passed
9	FR10	Users should have the ability to visualise a 3D point cloud from the produced depth map.	Users are able to view a point cloud from the depth map.	Users can view the 3D point cloud from the depth map.	Passed

Table 28: Overview of functional tests carried out

8.7. Module And Integration Testing

Module	Input	Expected Output	Actual Output	Result
Input validation	Image input	Image is added to the system.	Image is added to the system.	Passed
	Other file input	Image is not added to the system and error shown.	Input is not added and pop up alert is shown	Passed
Estimate depth for input image	Rain Image	Successfully generate a depth map	Depth map gets generated successfully	Passed
	Snow image	Successfully generate a depth map	Depth map gets generated successfully	Passed
	Fog Image	Successfully generate a depth map	Depth map gets generated successfully	Passed

Visualise depth map	Adverse weather image	The depth map is successfully visualised	The depth map is successfully visualised	Passed
Save depth map	Adverse weather image	The depth map is successfully saved	The depth map is successfully saved	Passed
Visualise 3D point cloud	Adverse weather image	Users can successfully see a 3D depth map of the scene in the image	Users can successfully see a 3D depth map of the scene in the image	Passed

Table 29: Overview of module and integration test

8.8 Non-Functional Testing

The non-functional requirements defined for the proposed system were accuracy, performance, and reliability. The following sections include the testing procedure for each of these non-functional requirements.

8.8.1. Accuracy Testing

The accuracy of the proposed systems was tested by using thresholded accuracy metrics $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$ first proposed by (Ladicky, Shi and Pollefeys, 2014). The table below shows the thresholded accuracy of the proposed system. The accuracy was tested on the test set of the CityScapesWeather dataset with 1500 images.

Dataset	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
CityScapesWeather	0.875	0.945	0.968
vKITTI	0.921	0.964	0.977

Table 30: Accuracy test results

8.8.2. Performance Testing

The performance of the proposed approach is tested by measuring how fast the depth estimations are produced in non-GPU-powered systems during inferencing. The table below shows the proposed system's inferencing time against some other state-of-the-art approaches. The testing was carried out on a personal computer with no GPU acceleration. The details are stated below:

- Operating System: MacOS
- CPU: 3.2 GHz Apple Silicon M1
- Memory: 8GB

Model	Architecture	Time(s)	Training Paradigm
BTS (Lee et al., 2020)	Autoencoder	**0.22**	Supervised
DenseDepth (Alhashim and Wonka, 2019)	Autoencoder	0.35	Supervised
Monodepth2 (Godard et al., 2019)	CNN	0.56	Self-supervised
WeatherDepth	Adversarial Autoencoder	0.78	Supervised
PackNet-SfM (Guizilini et al., 2020)	CNN	0.97	Self-supervised
DORN (Fu et al., 2018)	Autoencoder	0.98	Supervised

Table 31: Performance test results

8.8.3. Reliability Testing

The reliability of the developed system depends on how well the depth estimation system performs in various weather conditions. Ideally, the system should produce depth estimates with very close or equal accuracy when tested in various weather conditions. The table below records the accuracy of the system for each weather condition. The model was evaluated on a dataset of 500 images of each weather type separately.

Weather Condition	AbsRel	RMSE	RMSE (log)	SqRel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Rain	0.081	8.117	0.228	1.192	0.926	0.964	0.977
Snow	0.144	10.475	0.306	2.489	0.826	0.926	0.959
Fog	0.127	9.977	0.284	2.444	0.871	0.943	0.965

Table 32: Testing model reliability

As it can be seen from the results shown in the table above, the developed approach functions well in all adverse weather conditions without causing significant degradation in performance in any specific weather type and produces depth maps within a close (± 0.02) thresholded accuracy range.

8.9. Limitations Of The Testing Process

Certain limitations were observed during testing the developed system against the state-of-the-art. Below are the limitations faced in the testing process:

- Specific adverse weather depth estimation systems included in the literature review were not compared during testing due to the models not being publicly available and ones that required extreme compute power to retrain on a different dataset.
- Inferencing from the trained models for testing purposes was carried out on a personal computer, and the performance may slightly vary once deployed to AVs or when tested on a different computer as the computational power would vary.

8.10. Chapter Summary

The chapter defined the various goals and objectives for testing the developed system. The results obtained by testing the model using various metrics were provided in this chapter, along with benchmarking results with the vKITTI dataset. Finally, the functional and non-functional requirements defined at the beginning of the research project were tested, and the completion rate was found to be 100%.

CHAPTER 09: EVALUATION

9.1. Chapter Overview

This chapter provides a detailed overview of how the proposed system was evaluated, and the chosen evaluation methodology and various other aspects will be explained in detail. Feedback collected from various stakeholders, along with the author's self-evaluation of the system, will be discussed in this chapter.

9.2. Evaluation Methodology And Approach

The developed depth estimation approach was evaluated using both qualitative and quantitative approaches. Since the target audience of the proposed system will be engineers working on self-driving cars or robotics and will not be of much use to the general public, a interview-based evaluation approach was ideal rather than a questionnaire-based one. The developed system was presented to the domain and technological experts, and a evaluation was received based on the criteria explained in the following section. The system was also quantitatively evaluated by benchmarking the system against public depth estimation benchmark suites, and the results were compared against the state-of-the-art.

9.3. Evaluation Criteria

The qualitative evaluation performed on the system will be based on the criteria shown in the following table.

Criterion	Evaluation Purpose
Overall Project Concept	To gather insight about the system from the target audience’s point of view.
Project Scope	To learn experts’ views on the project scope.
Design and architecture	To learn if the system follows good coding standards and adheres to standard engineering principles.
Solution and Prototype	To evaluate if the developed system addresses the identified research gap.

Accuracy	To validate if the system is performing accurately enough to be a reasonable solution.
Performance	To validate if the system is performing fast enough to be deployed in AVs.
Choice of features and overall design of the user interface	To gain an understanding of its functionalities, ease of use, and how appealing it is to the users.

Table 33: Evaluation criteria

9.4. Self-Evaluation

The author self-evaluated the developed system based on the evaluation criteria defined in the previous section. The evaluation results can be found in the table below.

Criterion	**Self-evaluation**
Overall Project Concept	Estimating depth from a single 2D camera image without any additional support from other sensors such as radar or lidar is a challenging research area that has recently grown in popularity due to recent advancements in deep neural networks. The benefits that could be brought by a fully functioning monocular system that can estimate depth accurately are vast, so research on a proper adverse weather depth estimation system is essential for the autonomous driving community.
Project Scope	The project's scope, as mentioned in several sections in the thesis, only attempts to find a solution for adverse weather conditions, which is one of the edge cases affecting depth estimation in autonomous driving. The main reason for the author limiting the research for only adverse weather was the limited time available to complete the research project; an all-in-one system that attempts to address all edge cases affecting camera-based depth estimation is likely possible if more time could be allocated.

Design and architecture	The adversarial depth estimation approach designed by the author gave a satisfactory outcome. All components initially planned to be part of the system were successfully implemented, and it was shown that the proposed system is capable of handling adverse weather conditions better than state-of-the-art approaches.
Solution, Prototype and it performance	The prototype built successfully showed the full capabilities of the proposed depth estimation system. The proposed solution was thoroughly compared in terms of accuracy, performance, and overall depth estimation quality to the state-of-the-art, and it was proven with sufficient evidence that the proposed solution is valid.
Choice of features and overall design of the user interface	The features included in the prototype and the overall design of the UI were carefully chosen to fully show the evaluators the system's full capabilities and convey an understanding of how useful it will be once deployed in an AV.

Table 33: Evaluation criteria

9.5. Selection Of The Evaluators

The developed system was evaluated by several domain experts and technical experts related to the system. The list of all evaluators from whom the project was evaluated, and received feedback is listed in **Appendix D**.

9.6. Evaluation Result And Expert Opinion

The opinions of experts gathered by conducting interviews, explaining the project, and demoing it are summarised in the table below in a thematic analysis form.

Criterion	Evaluator Group	Summary of Expert Opinion
Overall Project Concept	Technological	Depth estimation from 2D images is a challenging task using machine learning. The identified gap is valid, and the depth of the research is adequate for a final year thesis.

	Domain	Depth estimation in adverse weather is a topic that not much focus has been shed on in previous research; researching a possible approach to handle such a situation will be very helpful. The importance of the topic would grow with time as more and more self-driving vehicles move to software-based approaches for perceptive tasks rather than using sensors.
Project Scope	Technological	The scope chosen is broad enough and requires adequate research to develop the system. The project is well scoped for the undergraduate level.
	Domain	It would be ideal if the system could handle low light instances as well, as it is a limitation in current depth estimation approaches, but given the limited time available for development, the chosen project scope is acceptable.
Design and architecture	Technological	The go-to approach for tasks similar to depth estimation would be autoencoder; the fact that the student was able to achieve better results than a well-established autoencoder-based approach by training the autoencoder in an adversarial manner is outstanding.
Solution, Prototype and its performance	Technological	The proposed solution is good and seems to have an in-depth analysis of the existing approaches. Model evaluation is acceptable, and enough evidence was provided to prove its performance over existing approaches.
	Domain	The approach produces good enough depth estimates to be tested out on actual self-driving vehicles under adverse weather conditions. The prototype is good,

		and it is able to show the usefulness of the developed approach.
Choice of features and overall design of the user interface	Technological	The features built into the prototype are adequate to show the viewers the quality of the depth estimates produced.
	Domain	The point cloud feature is imposing, and the option to save the depth map feature would be helpful if we were to test the generated depth maps through other applications. The choice of UI framework could have been improved as a python backend is ideal.

Table 34: Evaluation results based on domain and technology expert feedback

9.7. Limitations Of Evaluation

There were a few limitations faced during the evaluation phase of the project. A significant limitation during the evaluation was the lack of domain experts specializing in self-driving technology in Sri Lanka. Therefore domain-related feedback had to be taken from experts in close fields of study, such as Electrical Engineering, that were well experienced in working with deep learning tasks. Another limitation was that the previous approaches used different datasets for evaluation, so the author had to re-train the model in various datasets they had used for their system.

9.7. Evaluation On Functional Requirements

Evaluation of the functional requirements expected from the system has resulted in a completion rate of 100%. Evaluation results for all functional requirements are provided in Appendix E.

9.9. Evaluation On Non-Functional Requirements

Evaluation of the non-functional requirements expected from the system has resulted in a completion rate of 100%. Evaluation results for all non-functional requirements are provided in Appendix F.

9.10. Chapter Summary

The chapter discussed the evaluation approach and methodology followed when evaluating the system. Depending on various aspects of the research project, the evaluation criteria were defined, and the expert's evaluation and author's self-evaluation were performed based on the defined criteria. The feedback from various domain and technological experts was then summarised into a thematic analysis.

CHAPTER 10: CONCLUSION

10.1. Chapter Overview

This chapter provides a final remark on the project, reflecting on the achievement of aims and objectives of the research, problems faced during the research, utilization of knowledge gained throughout the last four years of the degree program, and new skills gained during the research. The chapter will also provide insight on future work and contributions to the body of knowledge made from the research.

10.2. Achievements Of Research Aims & Objectives

The research aims to design, develop and evaluate a solid monocular depth estimation approach that can produce consistent and accurate depth estimates when subjected to various adverse weather conditions, unlike most state-of-the-art methods, which fail to maintain accuracy and consistency when tested in adverse weather conditions.

The aim of the project was successfully achieved. A monocular depth estimation approach that can produce significantly better depth estimates in adverse weather conditions than the state-of-the-art approaches was successfully developed and evaluated.

10.3. Utilisation Of Knowledge From The Course

The modules from which the knowledge gained throughout the degree program that helped lay the foundation for the research project are as follows:

- **Programming Principles I & II** - The fundamentals of programming, such as language constructs learned during the module, helped in choosing the programming language and tools required for the development of the research project, and the python programming skills gained were instrumental when learning machine learning.
- **Software Development Development Group Project** - This module helped build an understanding of how to conduct research, standard research methodologies, and knowledge of writing the thesis.
- **Mobile Native Application Development** - The swift language programming skills gained in this module helped build an interactive UI for the depth estimation model.

10.4. Use Of Existing Skills

Several skills learned during the degree program from outside resources and from within the degree curriculum were very useful and heavily utilized in developing the system. The skills attained are as follows:

- **Machine Learning** - Online courses such as "Machine Learning with PyTorch" from David Mertz and "Introduction to Computer Vision" from Udacity helped the author understand the basics of machine learning and computer vision, which was very useful when developing the project.
- Intro to **Self-driving cars** - "Introduction to Self-Driving Cars" Course from the University of Toronto through Coursera helped lay a good foundation on self-driving tech and helped in coming up with the project idea.
- **Swift Programming** - The programming skills gained from the "Hacking With Swift" series by Paul Hudson was extremely useful when developing the front-end macOS application of the system.

10.5. Use Of New Skills

- Even though the author had a basic understanding of computer vision concepts, the research project helped him gain a thorough understanding and experience in working with advanced deep learning algorithms such as Generative Adversarial Networks and autoencoders which were new to the author.
- The research project also provided the opportunity to learn and work with new python deep learning libraries such as Pytorch.
- The author had no prior experience in MacOS development; the research project provided an opportunity to learn MacOS development and work with cutting-edge frameworks provided by Apple, such as CoreML.

10.6. Achievement Of Learning Outcomes

The learning outcomes achieved by the end of the research project are summarised on the table below.

Description	Learning Outcome
- The research provided an opportunity for the author to gain a solid understanding of monocular depth estimation, a good understanding of previous research on it, and good knowledge of the domain enough to evaluate them critically. - The research project gave the author the knowledge required to identify research gaps in an area of interest and conduct research properly, maintaining the research standards.	LO1, LO2, L04
- The study of existing research gave a good understanding of the tools and techniques used to develop monocular depth estimation approaches. - The research provided the opportunity to meet and discuss the proposed system with domain and technical experts in the field, which helped identify such a system's requirements.	LO3
- The development of the proposed approach required a great deal of knowledge of advanced concepts of deep learning and computer visions such as adversarial networks and autoencoders and a good understanding of the basics so that it was possible to successfully come up with an approach that helps obtain best of both learning techniques.	LO5
The project strictly followed the SLEP issues and its mitigation procedures.	LO6
- The research project produced a fully functional working prototype of the system, which clearly shows the capabilities of the proposed system, and the system was extensively experimented with using a wide range of techniques to prove its validity	LO7
- The final thesis thoroughly documented the complete research process step by step; the results obtained by evaluating the system against state-of-the-art approaches were also presented. A detailed overview of the new skills gained and the actual project plan followed during the development of the project was documented in the thesis.	LO8

Table 35: Overview of achieved learning outcomes

10.7. Problems And Challenges Faced

Problem/Challenge	Solution
Extremely resource-intensive during training	Since the proposed approach was based upon a U-Net being adversarially trained, the system was very computationally intensive during training, and training on local desktop GPUs was not feasible. So for training, a cloud GPU server with a high-end Tesla T4 GPU was used, dramatically improving training time compared to local consumer-grade GPUs.
Multiple methods of implementation were possible	After research on existing literature, it was identified that a depth estimation network could be built in several ways; previous research was built using U-Net Autoencoders, Generative Adversarial Networks, and transformer-based approaches. Countless hours were spent studying each possible approach, identifying its pros and cons, putting together a working system on each possible approach, and testing out how well each approach seems to fit the dataset I created.
Steep learning curve	A significant amount of effort and time was spent on learning all the technologies and tools used in the development of similar previous research, and given that the author had minimal knowledge of the concepts of adversarial networks and autoencoder at the start of the research, there was a very challenging steep learning curve.
There were no dataset that exactly met all my needs	Since research on depth estimation of adverse weather conditions was a very ill-studied problem, no publicly available dataset met the proposed system's needs and contained images captured under the rain, snow, and fog weather conditions. So a dataset was put together by augmenting a publicly available clear weather depth estimation dataset.

Table 36: Overview of problems and challenges faced

10.8. Limitations Of The Research

1. A significant limitation during development was that the MacOS applications do not support representing images in float (float32) values; the depth predictions of type float were converted to type integer (uint8), resulting in a slight loss of detail in the point clouds representation.
2. The project was tightly scoped to only address adverse weather conditions due to limited time available to conduct the research.
3. The input data were cropped to a size of 512x512 in order to obtain a good balance between the computation power required and the quality of the produced depth maps. If more computational resources were available, it would be possible to input a high-resolution image and output a higher resolution, more accurate depth map.
4. Since the author did not have access to a local decently powered GPU, only depth maps from single images were attempted; producing depth maps directly from the camera stream is entirely possible with the developed system if a decently powered GPU is locally available.

10.9. Future Enhancements

1. The current limitation of NSimage and CoreML frameworks to represent float values is expected to be fixed very soon in an upcoming update at the end of 2022 or the beginning of 2023. So with some minor changes to the system, the prototype will be able to produce point clouds with lossless quality.
2. A significant improvement that could be made to the project would be to make the system also perform well under low light conditions as often low light and adverse weather conditions could occur together.

10.10. Achievement Of The Contribution To Body Of Knowledge

When subjected to adverse weather conditions, the state-of-the-art monocular depth estimation approaches were significantly limited in depth estimation accuracy and overall performance. This inability had been one of the significant factors that encouraged the autonomous drinking industry to still mostly depend on sensor-based approaches even though they were significantly more costly than camera-based approaches. This research addressed this gap in depth estimation and developed a system that could successfully estimate depth in adverse

weather conditions. The 3 main contributions to the body of knowledge made by this research are :

1. This research proposed an adversarially trained autoencoder architecture for robust monocular depth estimation in adverse weather conditions, which helps solve a significant gap in monocular depth estimation for self-driving vehicles. The ins and outs of the chosen architecture and justifications were provided in the implementation chapter.
2. This research also tests out a variety of loss functions and finds an optimal loss combination for the task of monocular depth estimation in adverse weather conditions using the novel proposed approach.
3. A new large-scale dataset is introduced with monocular images and ground truth data for monocular depth estimation in adverse weather conditions.

The system developed in this research hopes to help bring the autonomous industry one step closer to achieving full autonomy and complete autonomous driving perception using only camera-based approaches. Also, this research hopes to help camera-based depth estimation approaches draw more attention from various autonomous driving communities that still believe and utilize only sensor-based approaches for tasks such as depth estimation that can be achieved well with only cameras.

10.11. Concluding Remarks

This chapter concludes the end of the research project and thesis. The chapter provided final remarks on how well the aim and objective of the project were met and how existing knowledge gathered from the degree program and other resources were utilized in developing the proposed system. The chapter also discussed in detail the various challenges faced during the development process, how they were overcome, existing limitations of the system, and possible improvements that could be made to the system as future work. Finally, the chapter provided an overview of the contribution to the knowledge body.

REFERENCES

- Adams, E. (2020). Why we're still years away from having self-driving cars. Vox. Available from https://www.vox.com/recode/2020/9/25/21456421/why-self-driving-cars-autonomous-still-years-away [Accessed 3 November 2021].
- Aleotti, F. et al. (2019). Generative Adversarial Networks for Unsupervised Monocular Depth Prediction. In: Leal-Taixé, L. and Roth, S. (eds.). Computer Vision – ECCV 2018 Workshops. Lecture Notes in Computer Science. Cham: Springer International Publishing, 337–354. Available from https://doi.org/10.1007/978-3-030-11009-3_20 [Accessed 13 October 2021].
- Alhashim, I. and Wonka, P. (2019). High Quality Monocular Depth Estimation via Transfer Learning. arXiv:1812.11941 [cs]. Available from http://arxiv.org/abs/1812.11941 [Accessed 18 April 2022].
- Atapour-Abarghouei, A. and Breckon, T.P. (2018). Real-Time Monocular Depth Estimation Using Synthetic Data with Domain Adaptation via Image Style Transfer. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. June 2018. Salt Lake City, UT: IEEE, 2800–2810. Available from https://doi.org/10.1109/CVPR.2018.00296 [Accessed 12 October 2021].
- Bachute, M.R. and Subhedar, J.M. (2021). Autonomous Driving Architectures: Insights of Machine Learning and Deep Learning Algorithms. Machine Learning with Applications, 6, 100164. Available from https://doi.org/10.1016/j.mlwa.2021.100164.
- Bhoi, A. (2019). Monocular Depth Estimation: A Survey. arXiv:1901.09402 [cs]. Available from http://arxiv.org/abs/1901.09402 [Accessed 6 September 2021].
- Brett, J.A. (2016). Thinking Local about Self-Driving Cars: A Local Framework for Autonomous Vehicle Development in the United States. 126.
- Brownlee, J. (2017). A Gentle Introduction to Transfer Learning for Deep Learning. Machine Learning Mastery. Available from https://machinelearningmastery.com/transfer-learning-for-deep-learning/ [Accessed 20 June 2022].
- Brownlee, J. (2020). Autoencoder Feature Extraction for Classification. Machine Learning Mastery. Available from https://machinelearningmastery.com/autoencoder-for-classification/ [Accessed 20 June 2022].

- Chakravarty, P., Narayanan, P. and Roussel, T. (2019). GEN-SLAM: Generative Modeling for Monocular Simultaneous Localization and Mapping. arXiv:1902.02086 [cs]. Available from http://arxiv.org/abs/1902.02086 [Accessed 13 October 2021].
- Choi, J. et al. (2020). SAFENet: Self-Supervised Monocular Depth Estimation with Semantic-Aware Feature Extraction. arXiv:2010.02893 [cs]. Available from http://arxiv.org/abs/2010.02893 [Accessed 28 December 2021].
- Computer Vision: What it is and why it matters. (2021). Available from https://www.sas.com/en_us/insights/analytics/computer-vision.html [Accessed 3 November 2021].
- Cordts, M. et al. (2016). The Cityscapes Dataset. Available from https://www.cityscapes-dataset.com/wordpress/wp-content/papercite-data/pdf/cordts2015cvprw.pdf.
- Dholakia, S. (2019). Perception: How Self-Driving Cars 'see' the World. Medium. Available from https://swarit.medium.com/perception-how-self-driving-cars-see-the-world-ae630636f4c [Accessed 12 October 2021].
- Dia, H. (2021). 'Self-driving' cars are still a long way off. Here are three reasons why. The Conversation. Available from http://theconversation.com/self-driving-cars-are-still-a-long-way-off-here-are-three-reasons-why-159234 [Accessed 23 September 2021].
- Eigen, D., Puhrsch, C. and Fergus, R. (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. Advances in Neural Information Processing Systems. 2014. Curran Associates, Inc. Available from https://proceedings.neurips.cc/paper/2014/hash/7bccfde7714a1ebadf06c5f4cea752c1-Abstract.html [Accessed 18 April 2022].
- Frank, B.A. (2020). How Self-Driving Cars Will Reduce Traffic Congestion. The Startup. Available from https://medium.com/swlh/how-self-driving-cars-will-reduce-traffic-congestion-8bad5594c5d0 [Accessed 3 December 2021].
- Fu, H. et al. (2018). Deep Ordinal Regression Network for Monocular Depth Estimation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. June 2018. Salt Lake City, UT: IEEE, 2002–2011. Available from https://doi.org/10.1109/CVPR.2018.00214 [Accessed 3 September 2021].

- Garg, R. et al. (2016). Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue. In: Leibe, B. Matas, J. Sebe, N. et al. (eds.). Computer Vision – ECCV 2016. 2016. Cham: Springer International Publishing, 740–756.
- Gasperini, S. et al. (2021). R4Dyn: Exploring Radar for Self-Supervised Monocular Depth Estimation of Dynamic Scenes. arXiv:2108.04814 [cs]. Available from http://arxiv.org/abs/2108.04814 [Accessed 21 October 2021].
- Global status report on road safety 2018. (2018). Available from https://www.who.int/publications-detail-redirect/9789241565684 [Accessed 3 December 2021].
- Godard, C. et al. (2019). Digging Into Self-Supervised Monocular Depth Estimation. 2019 IEEE/CVF International Conference on Computer Vision (ICCV). October 2019. Seoul, Korea (South): IEEE, 3827–3837. Available from https://doi.org/10.1109/ICCV.2019.00393 [Accessed 7 August 2021].
- Gonzales, D. (2020). Unsupervised monocular depth estimation: Learning to generalize [Thesis]. Available from https://www.ideals.illinois.edu/handle/2142/108020 [Accessed 4 December 2021].
- Goodfellow, I. et al. (2014). *Generative Adversarial Nets*. Available from https://arxiv.org/pdf/1406.2661.pdf.
- Guizilini, V. et al. (2020). 3D Packing for Self-Supervised Monocular Depth Estimation. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). June 2020. Seattle, WA, USA: IEEE, 2482–2491. Available from https://doi.org/10.1109/CVPR42600.2020.00256 [Accessed 21 October 2021].
- Heller, M. (2022). What is PyTorch? Python machine learning on GPUs. InfoWorld. Available from https://www.infoworld.com/article/3658989/what-is-pytorch-a-popular-alternative-to-tensorflow-for-model-development-with-gpus.html [Accessed 3 July 2022].
- Hu, X. et al. (2019). Depth-Attentional Features for Single-Image Rain Removal. 2019. 8022–8031. Available from https://openaccess.thecvf.com/content_CVPR_2019/html/Hu_Depth-Attentional_Features_for_Single-Image_Rain_Removal_CVPR_2019_paper.html [Accessed 6 July 2022].
- Huang, G. et al. (2018). Densely Connected Convolutional Networks. Available from https://doi.org/10.48550/arXiv.1608.06993 [Accessed 3 July 2022].

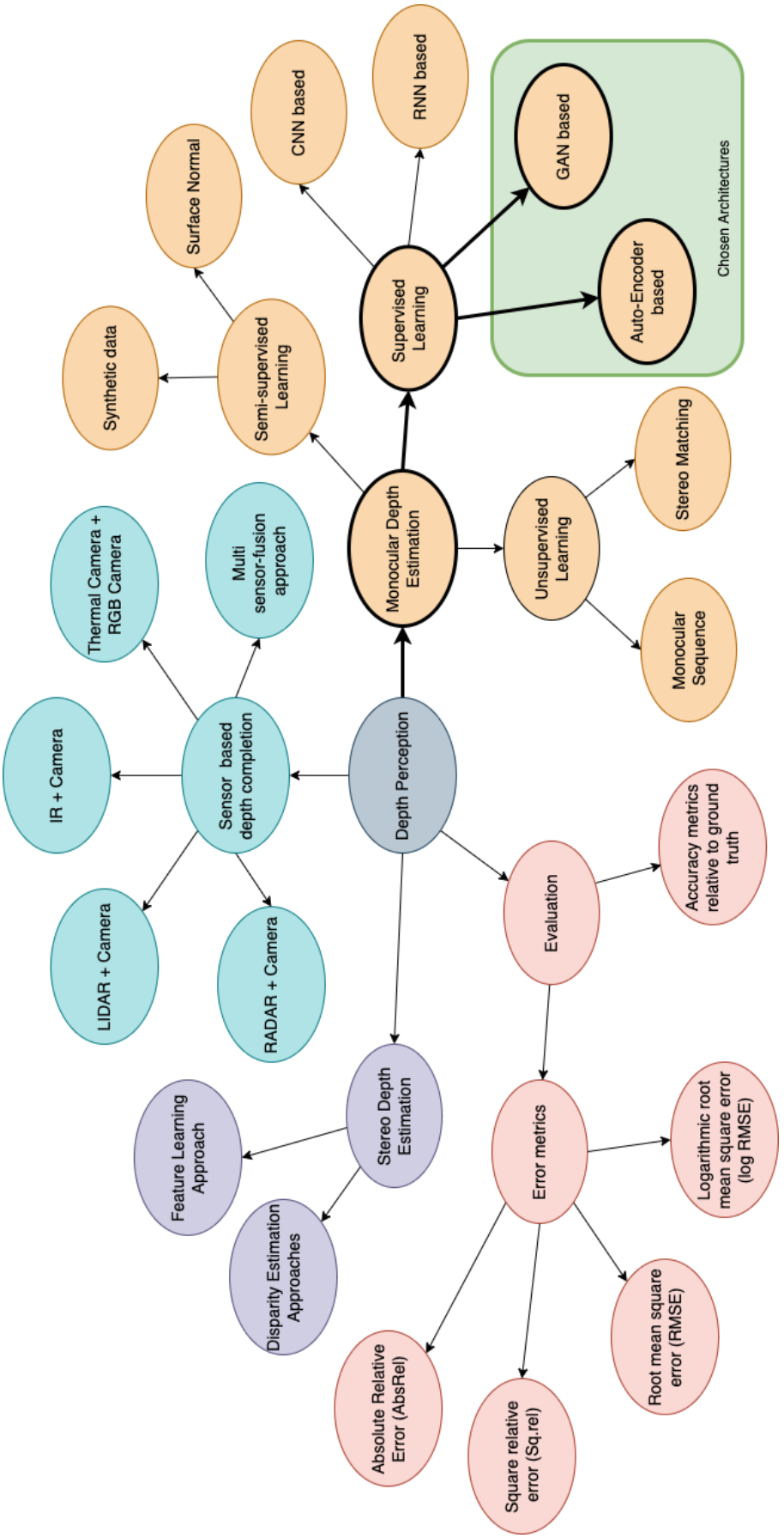
- Humenberger, M. (2020). Announcing Virtual KITTI 2. Naver Labs Europe. Available from https://europe.naverlabs.com/blog/announcing-virtual-kitti-2/ [Accessed 8 July 2022].
- Institute, T.R. (2021). Self-supervised Learning in Depth — part 1 of 2. Toyota Research Institute. Available from https://medium.com/toyotaresearch/self-supervised-learning-in-depth-part-1-of-2-74825baaaa04 [Accessed 15 September 2021].
- Isola, P. et al. (2018). Image-to-Image Translation with Conditional Adversarial Networks. Available from http://arxiv.org/abs/1611.07004 [Accessed 3 July 2022].
- Ladicky, L., Shi, J. and Pollefeys, M. (2014). Pulling Things out of Perspective. 2014 IEEE Conference on Computer Vision and Pattern Recognition. June 2014. Columbus, OH, USA: IEEE, 89–96. Available from https://doi.org/10.1109/CVPR.2014.19 [Accessed 23 June 2022].
- Lee, J.H. et al. (2020). From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation. arXiv:1907.10326 [cs]. Available from http://arxiv.org/abs/1907.10326 [Accessed 16 September 2021].
- Lin, J.-T., Dai, D. and Gool, L.V. (2020). Depth Estimation from Monocular Images and Sparse Radar Data. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). October 2020. 10233–10240. Available from https://doi.org/10.1109/IROS45743.2020.9340998.
- Liu, F., Shen, C. and Lin, G. (2015). Deep convolutional neural fields for depth estimation from a single image. 1 June 2015. 5162–5170. Available from https://doi.org/10.1109/CVPR.2015.7299152.
- Luca, L. (1998). Stereo Vision: Triangulation. Available from http://www.diag.uniroma1.it//~iocchi/stereo/triang.html [Accessed 12 October 2021].
- Makhzani, A. et al. (2016). Adversarial Autoencoders. Available from http://arxiv.org/abs/1511.05644 [Accessed 8 July 2022].
- Mertan, A., Duff, D.J. and Unal, G. (2022). Single Image Depth Estimation: An Overview. Digital Signal Processing, 123, 103441. Available from https://doi.org/10.1016/j.dsp.2022.103441.
- Perception – Towards Data Science. (2021). Perception – Towards Data Science. Available from https://towardsdatascience.com [Accessed 3 November 2021].
- Pinard, C. et al. (2019). Learning Structure-from-Motion from Motion. In: Leal-Taixé, L. and Roth, S. (eds.). Computer Vision – ECCV 2018 Workshops. Lecture Notes in

Computer Science. Cham: Springer International Publishing, 363–376. Available from https://doi.org/10.1007/978-3-030-11015-4_27 [Accessed 6 July 2022].

- Piven, Y. (2021). Single image scene-depth estimation based on self-supervised deep learning : For perception in autonomous heavy duty vehicles. Available from http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-455948 [Accessed 4 December 2021].
- Rajderkar, D. and Mohod, P.S. (2013). Removing snow from an image via image decomposition. 1 March 2013. 576–579. Available from https://doi.org/10.1109/ICE-CCN.2013.6528565.
- Ranftl, R. et al. (2020). Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. Available from http://arxiv.org/abs/1907.01341 [Accessed 8 July 2022].
- Road traffic injuries. (2022). Available from https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries [Accessed 2 December 2021].
- Ronneberger, O., Fischer, P. and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science*, 234–241. Available from https://doi.org/10.1007/978-3-319-24574-4_28 [Accessed 8 July 2022].
- Sakaridis, C., Dai, D. and Van Gool, L. (2018). Semantic Foggy Scene Understanding with Synthetic Data. International Journal of Computer Vision, 126 (9), 973–992. Available from https://doi.org/10.1007/s11263-018-1072-8.
- Saxena, A. (2007). Depth Estimation Using Monocular and Stereo Cues. 7.
- Siddiqui, S.A., Vierling, A. and Berns, K. (2020). Multi-Modal Depth Estimation Using Convolutional Neural Networks. arXiv:2012.09667 [cs]. Available from http://arxiv.org/abs/2012.09667 [Accessed 22 April 2022].
- Tan, D. (2021). Depth Estimation: Basics and Intuition. Medium. Available from https://towardsdatascience.com/depth-estimation-1-basics-and-intuition-86f2c9538cd1 [Accessed 15 September 2021].
- Vankadari, M. et al. (2020). Unsupervised Monocular Depth Estimation for Night-time Images using Adversarial Domain Feature Adaptation. arXiv:2010.01402 [cs]. Available from http://arxiv.org/abs/2010.01402 [Accessed 9 August 2021].
- Wang, Y. et al. (2019). Anytime Stereo Image Depth Estimation on Mobile Devices. arXiv:1810.11408 [cs]. Available from http://arxiv.org/abs/1810.11408 [Accessed 5 August 2021].

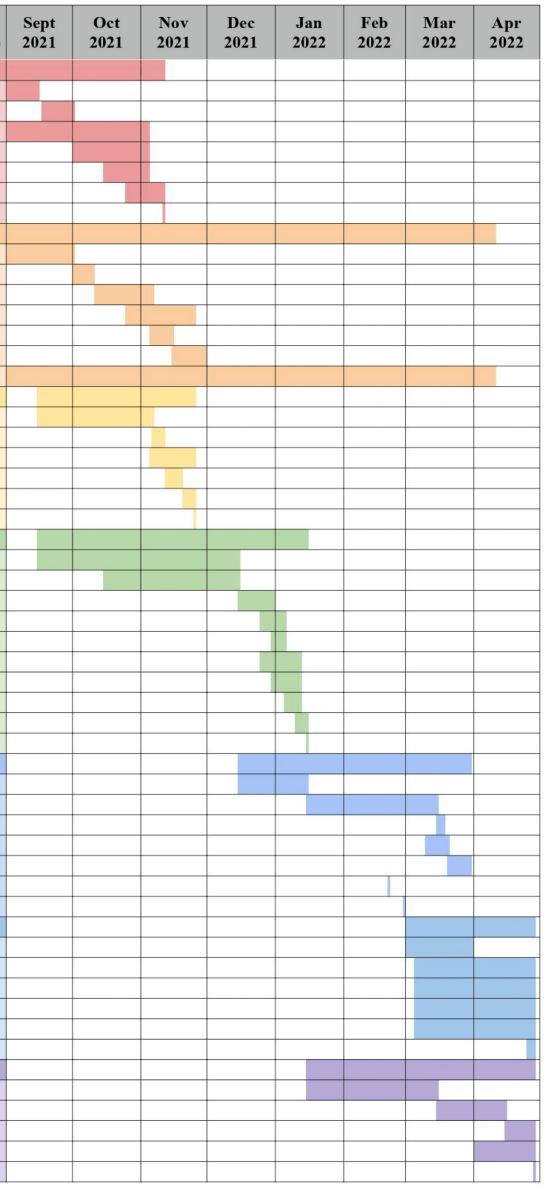
- Wang, Y. et al. (2020). Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. arXiv:1812.07179 [cs]. Available from http://arxiv.org/abs/1812.07179 [Accessed 5 August 2021].
- Xiaogang, R. et al. (2020). Monocular Depth Estimation Based on Deep Learning:A Survey. 2020 Chinese Automation Congress (CAC). November 2020. 2436–2440. Available from https://doi.org/10.1109/CAC51589.2020.9327548.
- Zang, S. et al. (2019). The Impact of Adverse Weather Conditions on Autonomous Vehicles: How Rain, Snow, Fog, and Hail Affect the Performance of a Self-Driving Car. IEEE Vehicular Technology Magazine, 14 (2), 103–111. Available from https://doi.org/10.1109/MVT.2019.2892497.
- Zhang, K. et al. (2021). Deep Dense Multi-scale Network for Snow Removal Using Semantic and Geometric Priors. IEEE Transactions on Image Processing, 30, 7419–7431. Available from https://doi.org/10.1109/TIP.2021.3104166.
- Zhao, C., Tang, Y. and Sun, Q. (2021). Unsupervised Monocular Depth Estimation in Highly Complex Environments. arXiv:2107.13137 [cs]. Available from http://arxiv.org/abs/2107.13137 [Accessed 9 August 2021].

APPENDIX A - CONCEPT MAP



APPENDIX B - GANTT CHART

	Task	Start Date	End Date	Duration (In Days)
1	**Project Initiation**	**1-Sep-2021**	**11-Nov-2021**	**72**
	Selecting a research problem and topic	1-Sep-2021	15-Sep-2021	15
	Mentor approval of the research topic	17-Sep-2021	1-Oct-2021	15
	Performing initial review of existing literature	1-Sep-2021	4-Nov-2021	65
	Preparing the Project Proposal (PP)	1-Oct-2021	4-Nov-2021	35
	Mentor and peer review of PP	15-Oct-2021	4-Nov-2021	21
	Self-evaluating and refining PP	25-Oct-2021	11-Nov-2021	18
	Submission of Project Proposal	11-Nov-2021	11-Nov-2021	1
2	**Literature Review**	**1-Sep-2021**	**10-Apr-2022**	**222**
	Research on monocular depth estimation	1-Sep-2021	1-Oct-2021	31
	Identifying the ML algorithms to tackle the research problem	1-Oct-2021	10-Oct-2021	10
	Research on identified ML algorithms	11-Oct-2021	6-Nov-2021	27
	Preparing the draft Literature Review (LR)	25-Oct-2021	25-Nov-2021	32
	Mentor and peer review of draft LR	5-Nov-2021	15-Nov-2021	11
	Self-evaluating and refining LR	15-Nov-2021	30-Nov-2021	16
	Make amendments to LR as research progress	1-Sep-2021	10-Apr-2022	222
3	**Requirement Gathering**	**15-Sep-2021**	**25-Nov-2021**	**72**
	Performing requirement elicitation	15-Sep-2021	6-Nov-2021	53
	Finalizing the functional and nonfunctional requirements	6-Nov-2021	11-Nov-2021	6
	Preparing the Software Requirement Specification (SRS)	5-Nov-2021	25-Nov-2021	21
	Mentor and peer review of SRS	12-Nov-2021	19-Nov-2021	8
	Self-evaluating and refining the SRS	20-Nov-2021	25-Nov-2021	6
	Submission of Software Requirement Specification	25-Nov-2021	25-Nov-2021	1
4	**System Design**	**15-Sep-2021**	**15-Jan-2022**	**123**
	Identifying components of existing systems	15-Sep-2021	15-Dec-2021	92
	Selecting components and boundaries of the prototype	15-Oct-2021	15-Dec-2021	62
	Creating high level architecture	15-Dec-2021	31-Dec-2021	17
	Mentor review of high level architecture	25-Dec-2021	5-Jan-2022	12
	Self-evaluating and refining the high level architecture	30-Dec-2021	5-Jan-2022	7
	Creating system design	25-Dec-2021	12-Jan-2022	19
	Preparing draft of Design Specification Document (DSD)	30-Dec-2021	12-Jan-2022	14
	Mentor review of DSD	5-Jan-2022	12-Jan-2022	8
	Self-evaluating and refining the DSD	10-Jan-2022	15-Jan-2022	6
	Submission of Design Specification Document	15-Jan-2022	15-Jan-2022	1
5	**Prototype Implementation**	**15-Dec-2021**	**30-Mar-2022**	**106**
	Planning the prototype implementation	15-Dec-2021	15-Jan-2022	32
	Implementing prototype	15-Jan-2022	15-Mar-2022	60
	Refactoring code following best practices	15-Mar-2022	18-Mar-2022	4
	Preparing the interim report	10-Mar-2022	20-Mar-2022	11
	Preparing prototype report	20-Mar-2022	30-Mar-2022	11
	Demoing the prototype to the mentor	21-Feb-2022	21-Feb-2022	1
	Submission of prototype report	28-Feb-2022	28-Feb-2022	1
6	**Testing and Evaluation**	**1-Mar-2022**	**28-Apr-2022**	**59**
	Preparing test cases	1-Mar-2022	31-Mar-2022	31
	Conducting unit tests	5-Mar-2022	28-Apr-2022	55
	Conducting functionality tests	5-Mar-2022	28-Apr-2022	55
	Conducting integration tests	5-Mar-2022	28-Apr-2022	55
	Conducting performance tests	5-Mar-2022	28-Apr-2022	55
	Final evaluation of the prototype	25-Apr-2022	28-Apr-2022	4
7	**Documentation and Conclusion**	**15-Jan-2022**	**28-Apr-2022**	**104**
	Creating draft Project Report (PR)	15-Jan-2022	15-Mar-2022	60
	Mentor and peer review of PR	15-Mar-2022	15-Apr-2022	32
	Refining PR	15-Apr-2022	28-Apr-2022	14
	Proofreading PR	1-Apr-2022	28-Apr-2022	28
	Submission of the Final Project Report	28-Apr-2022	28-Apr-2022	1



APPENDIX C - COMBINED LOSS FUNCTION COMPARISON

[Note: The following results were obtained by only training the model with each loss function for 5 epochs since it was only done to monitor the initial training progress with each loss combination and as each epoch takes about 45 mins to train in a high-end Tesla T4 GPU with 16GB of memory. Therefore, the number of epochs had to be limited due to time complexities.].

#	Loss functions : weight ratio	AbsRel	RMSE	RMSE (log)	SqRel
1	L1:100, BCE:1	35.841	450.928	2.968	43505.480
2	L1:1, SSIM:1, BCE:1	17.996	1010.870	2.8355	17362.349
3	(L1:1, SIM:10):100, BCE:1	0.180	25.049	0.349	5.902
4	L1:100, SSIM:10, BCE:1	**0.209**	**12.064**	**0.336**	**3.522**
5	(L1:0.1, SSIM:1):100, BCE:1	91.957	2352.904	4.062	155552.515
6	L1:100, SSIM:10, BCE:1, Gradient Loss:1	0.142	10.001	0.271	2.855

APPENDIX D - INTERVIEWEE DETAILS

Stakeholder	Name and Affiliations
Machine Learning Expert - Technology Expert	Dr. Nihal Kodikara Current HOD at IIT Former Professor at University of Colombo
Machine Learning Expert - Technology Expert	Mr. Prasan Yapa Lecturer - Software Engineering Former Senior Consultant Technology - Virtusa (Pvt) Ltd
Machine Learning / Self-driving Car Expert - Technology / Domain Expert	Mr. Dulaj Weerakoon Senior Electronic Engineer Zone24x7
Self-driving Car Expert - Domain Expert	*PHD Candidate Self-driving car engineer currently working on autonomous driving perception and control sections. Moscow State University, Russia

* these interviewees chose to remain anonymous

APPENDIX E - EVALUATION OF FUNCTIONAL REQUIREMENTS

ID	Requirement	Priority Level	Evaluation
FR1	Users should be able to input an adverse weather image to system to predict a depth map	M	Implemented
FR2	Input should be validated before feeding into the model	M	Implemented
FR3	System should function by only using single camera images.	M	Implemented
FR4	The depth estimation network should produce consistent and accurate depth estimations	M	Implemented
FR5	The produced depth maps should be the same dimensions as the original image	M	Implemented
FR6	The depth maps should have very minimal to no negative effects from the adverse weather conditions	M	Implemented
FR7	The system should be able to produce accurate depth maps from rain, fog, snow and clear images.	M	Implemented
FR8	Users should have the ability to save the produced depth maps.	S	Implemented
FR9	Users should have the ability to visualise the depth maps in multiple styles.	S	Implemented
FR10	Users should have the ability to visualise a 3D point cloud from the produced depth map.	C	Implemented

Functional Requirement Completion Percentage = (10/10) × 100 = 100%

APPENDIX F - EVALUATION OF NON-FUNCTIONAL REQUIREMENTS

ID	Requirement	Priority Level	Evaluation
NFR1	Accuracy	Important	Implemented
NFR2	Performance	Important	Implemented
NFR3	Reliability	Important	Implemented

Non-Functional Requirement Completion Percentage = (3/3) × 100 = 100%