DE-41 (MTS) TALHA, M.ASHHADUDDIN, NOOR, MUHAMMAD

MULTI-MODAL SENSOR FUSION (CAMERA, LiDAR) FOR ENVIRONMENTAL DEPTH PERCEPTION FOR AUTONOMOUS VEHICLES

COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING NATIONAL
UNIVERSITY OF SCIENCES AND TECHNOLOGY
RAWALPINDI
2023

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

DE-41 MTS

PROJECT REPORT

MULTI-MODAL SENSOR FUSION (CAMERA, LiDAR) FOR ENVIRONMENTAL DEPTH PERCEPTION FOR AUTONOMOUS VEHICLES

Submitted to the Department of Mechatronics Engineering
in partial fulfilment of the requirements
for the degree of
Bachelor of Engineering
in
Mechatronics
2023

Sponsoring DS:

Dr. Tahir Habib Nawaz (Supervisor)

Dr. Umar Shahbaz Khan (Co-Supervisor)

Submitted By:

Talha Tahir Khurshid

Muhammad Ashhad Uddin Umer Kazi

Noor UL Huda Rasool

Muhammad Malik

ACKNOWLEDGMENTS

To conduct this thesis on the perception module of self-driving cars, we would like to express our gratitude to Allah Almighty, who bestowed His blessings to carry out this extensive research. Further, we would like to extend our humble gratitude to our supervisor, Dr. Tahir Habib Nawaz, whose valuable advice helped us surmount many hurdles. We are also thankful to our co-supervisor, Dr. Umer Shahbaz Khan, for providing great assistance. Finally, a huge thanks to our parents, friends, and colleagues for their support.

Lastly, we are grateful to the IGNITE Foundation for giving us a sponsorship fund of 80,000 rupees.

ABSTRACT

With the rise of Artificial Intelligence and recent technological advancements, there has been considerable progress towards the development of autonomous vehicles. Autonomous driving can be broken down into four major phases: perception, object detection, path planning, and actuation. Our project focuses on the perception stage of the self-driving problem, specifically on developing a sensor fusion system that combines data from both Lidar and Stereo sensors to form a robust depth map for long ranges.

This project intends to develop a platform onto which other sensors can be added for increased precision and accuracy. The fusion technique being used is the Extended Kalman filter, which will combine the outputs of LiDAR and Stereo sensors.

The entire algorithm will be ported onto an embedded platform, the Jetson Nano, which is designed to manage the computational requirements of the fusion algorithm. This will enable our system to run in real time, making it suitable for use in autonomous vehicles.

The plan is to assess and evaluate the performance of this sensor fusion system using real-world data and scenarios. The project aims to contribute to the development of autonomous vehicles by providing a more accurate and reliable perception system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

α = Depth map from stereo camera.

β = Depth map from LiDAR

γ = Fused depth map

μ = Fusion Algorithm.

A_1 = Uncalibrated left image from Stereo Camera.

A_2 = Uncalibrated right image from Stereo Camera.

A_1' = Rectified right left from Stereo Camera.

A_2' = Rectified left right from Stereo Camera.

ρ = Point Cloud.

f = Focal length of ZED in pixels.

u = Disparity map.

$baseline$ = Distance between the two cameras of ZED.

k_1, k_2, k_3 = Radial distortion coefficients.

p_1, p_2 = Tangential distortion coefficients.

R = Rotation matrix.

T = Translation vector.

E = Essential matrix.

F = Fundamental matrix.

r_1, r_2 = Rectification transforms.

P_1, P_2 = Projection matrices.

Q = Disparity to depth mapping matrix.

A = State matrix of the model.

B = Control matrix of the model.

C = Measurement matrix

Q = Process noise covariance matrix.

R = Measurement noise covariance matrix.

U = Control vector.

X = State vector.

P = Error covariance

S = Innovation covariance matrix.

z = Measurement vector.

y = Measurement residual.

K = Kalman gain.

Chapter 1 - INTRODUCTION

1.1. Overview:

This overview summarizes a research project aimed at developing a robust perception model for autonomous vehicles. The project addresses the limitations of stereo cameras in accurately producing depth maps at long ranges by proposing a multi-modal sensor fusion algorithm. It involves obtaining depth maps from both stereo cameras and LiDAR through rectification and disparity map calculations. The background knowledge highlights the strengths and weaknesses of LiDARs and stereo cameras and emphasizes the importance of well-calibrated sensors and fusion techniques. The project's contribution is the creation of a research platform for self-driving vehicle development. The thesis structure includes chapters on existing problems, methodologies, results, and future directions.

1.2. Motivation:

An autonomous vehicle is dependent on the efficient working of its perception module. Perception acts as a visionary sense of self-driving cars. It acquires data from sensors and estimates the distance of objects in the environment to perform the required action.

Stereo cameras are exceptionally good at calculating high quality depth maps at short ranges. However, the issue with stereo cameras is that it cannot produce accurate depth maps at long ranges. To overcome this shortcoming of stereo cameras, this project intends to use a multi-modal sensor fusion algorithm that will fuse the output depth maps of stereo camera and LiDAR and get a robust depth map at long ranges.

1.3. Problem Formulation:

To get a fused depth map, a fusion algorithm 'μ' is used which requires a depth map from both the stereo Camera and LiDAR.

To get the depth map from the stereo Camera 'α', uncalibrated left and right images 'A_1' and 'A_2' respectively are obtained, on which rectification is applied to obtain rectified left and right images 'A'_1' and 'A'_2' respectively. After acquiring the rectified images, the disparity map 'δ' is calculated and then 'α' is found using the depth formula.

$$\alpha = \frac{f * baseline}{u} \qquad (1)$$

For obtaining the depth map from LiDAR 'β' a point cloud 'ρ' is obtained in polar coordinates which are converted to cartesian coordinates using the coordinate transformation.

At the end, 'μ' is used to fuse both 'α' and 'β' to get a fused depth map 'γ'.

1.4. Background Knowledge:

The most common perception sensors used in the autonomous industry are LiDARs, stereo cameras, and radars. Using a single sensor for producing an effective perception model often results in weak solutions. LiDAR depicts the surroundings by emitting a laser from the device forming a 3-dimensional point cloud. It scans a full 360 degrees and can have a range of scanning frequencies[1]. A stereo camera produces depth images by comparing the disparity between two images produced individually by its two cameras. Hence it is more convenient and accurate to perceive the surrounding environment. However, it does not provide an accurate depth map at long ranges, nor does it measure the distance between opaque and transparent objects [2]. In the case of LiDAR, it provides poor results in harsh weather and foggy conditions. Hence the drawback of each sensor can be mitigated and the strengths of each complement other by implementing the multi-modal algorithm of the fused results.

For ideal multi-modal perception systems, well-calibrated sensors are an important prerequisite. [3] However, a high probability of misalignments exists in the deployment of the perception models. The problem can be mitigated by using existing intrinsic and extrinsic calibration techniques. Online various datasets exist that include KITTI, NuScenes, A2D2, ApolloScape and Waymo, Tusimple and CuLane [3]. These datasets are further divided into Comprehensive and characteristic types based on the features they provide.

Once the sensors provide the information about the surroundings, the next task is to pre-process the data acquired, by combining the information into a common coordinate frame [4] using calibration methods. This procedure is followed by fusion techniques. When implementing a fusion method, it is critical to determine what, when and how to fuse the results of information acquired from various sensors. [3]

Different fusion techniques are employed on the fusion model to estimate the current position, orientation, and velocity of object. Most common techniques are Kalman filtering, extended Kalman filtering, Bayesian filtering and Fusion using Deep Learning. These techniques differ in terms of accuracy and applications they are used for.

1.5. Contribution:

This project tends to create a platform for development and research on self-driving vehicles to which more modalities can be added for experimentation. This is one of the first experiments in the field of self-driving cars in the department of Mechatronics Engineering at the College of Electrical and Mechanical Engineering, NUST.

1.6. Organization of the thesis:

This thesis is further written in the following fashion:

Chapter 2: Provides a Literature Review regarding the existing problems in the perception model and existing methodologies for developing an effective and robust fused model.

Chapter 3: Presents the details of methodology employed by our team in completing this project and explains the inner working of the project.

Chapter 4: Briefly discusses the summary of results and findings and comparison of accuracy with existing methodologies in terms of a more accurate depth perception model.

Chapter 5: Concludes the report and explores future possibilities and directions in which the project can be taken.

Chapter 2 - LITERATURE REVIEW

The autonomous industry has had significant research in the last decades. Many advancements have been made by the advent of Artificial Intelligence and strong computational power CPU have assisted the progress in the industry. Considering the general structure of autonomous driving systems e.g., localization, perception, prediction, action planning, and control, the first two are the key areas of our interest. The robust, and efficient localization and perception module are requisite for safe autonomous driving.

For the automotive vehicle, the optimal target is to develop a fully automated system that can perceive, predict, decide, plan, and execute their decisions in the real world. Perception, planning, and control are three broad modules in designating a path using an autonomous vehicle. Perception is a prerequisite to and path planning among the three. Over the years, extensive research has been implemented in the field of perception, introducing many sensors and sensor-fusion algorithms. In this research, we attempted to overview the state-of-art sensor fusion algorithms and identify the challenges faced by the prevailing methods to locate the gap.

An ideal perception system should be robust, accurate, and real-time [1]. While developing the ideal scenario is beyond success at present, the recent advancement in the fields and continued interest by research scientists portrayed the future of the growing industry in automotive vehicles. For perception, the rudimentary task is the selection of sensors for data acquisition. In this regard, various datasets offer large acquisition of images from different sensors in diverse environments to obtain an accurate multi-modal fusion algorithm.

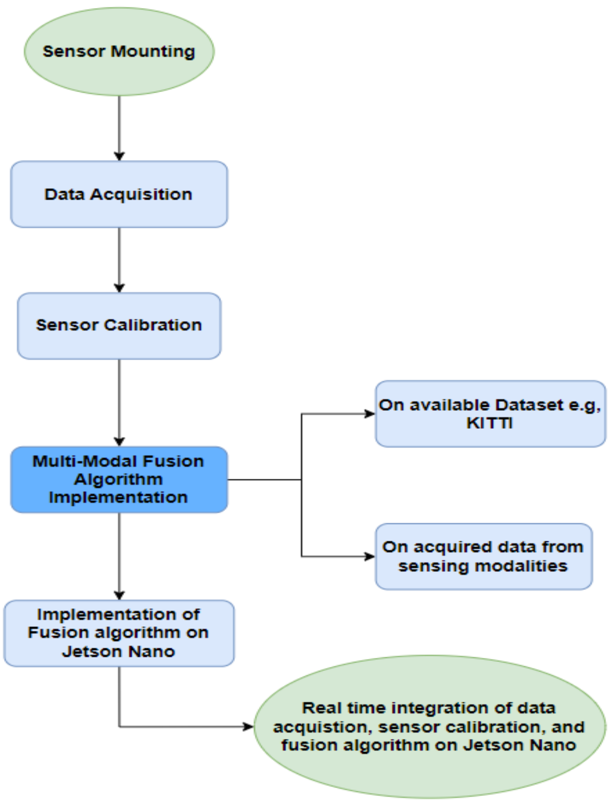
Sensor Mounting
Data Acquisition
Sensor Calibration
Multi-Modal Fusion Algorithm Implementation
On available Dataset e.g, KITTI
On acquired data from sensing modalities
Implementation of Fusion algorithm on Jetson Nano
Real time integration of data acquistion, sensor calibration, and fusion algorithm on Jetson Nano

FIGURE 1. WORKFLOW OF MULTI-MODAL SENSOR FUSION METHOD USED IN THIS PAPER.

2.1. Sensing Modalities:

Sensors are used for object identification, scene understanding, segmentation, and object localization. Further, the number of sensors used and their location on the vehicle has highly influenced the data acquired through these sensors. Quality sensors are used to obtain reliable data, useful for the control system. Diverse kinds of sensors operating in the autonomous vehicle are LiDAR, Camera, Radar, IMUs, and GPS [2]**.** These sets of internal and external sensors have distinct characteristics and limitations that, when combined to develop a multi-modal algorithm, boost the system's reliability, and compensate for another's shortcomings [3]. Internal sensors measure the physical variables of the vehicles while the external sensors build the relationship between the vehicle and the environment [2]**.** This paper focuses on fusing LiDAR and stereo Camera because, this will allow us to increase the range of depth perception for the autonomous vehicles, since stereo cameras can only display depth up to a range of 20m, while LiDAR can calculate depth up to a range of 100-200m. The aim of this project is to provide proof of concept for building a framework upon which further sensing modalities such as radar and sonar could be easily added and that is why we are only focusing on fusing stereo Camera and LiDAR.

TABLE 1. DIFFERENT SENSING MODALITIES USED IN AUTONOMOUS INDUSTRIES [4]

Sensors	**Function**	**Limitations**
Stereo Camera	Provide detailed colour, contrast, and texture information. Effective sensor for object detection scenarios.	Sensitive to adverse weather (fog, rain, etc.) and varying illumination conditions. Cannot directly provide depth information.
LiDAR	Provides accurate depth information in the form of 3D points. Traditionally LiDAR's of wavelength of 940 nm were not suitable for harsh weather conditions but the upcoming LiDAR's are able to use a longer wavelength of 1500 nm., which mitigates the effects of the weather conditions.	Not effective for object classification as their points become sparse with distant objects.
Radar	Robust weather conditions. Used for Assistive cruise control and traffic jam assistance.	Due to low resolution, it is not suitable for object classification.
Ultrasonics	These are used for low-speed and near-range object detections.	Affected by external parameters, Air, humidity, temperature, and dirt.
GNSS and HD Maps	It provides an accurate 3D position by a satellite system. A combination of GNSS and HD maps is used in the path-planning and localization of automotive vehicles.	Expensive and noise-affected approach to be used in automobiles.
IMU and Odometry	Used for accurate localization to measure vehicle's acceleration and odometry as they provide vehicles internal information.	It results in the accumulation of error in position determination over time

2.2. Sensor Calibration:

As research in the autonomous industry advances, a combination of LiDAR and cameras has become indispensable. Fusion of these two results in rich and contemporary data to sense and infer about the surroundings [7]. The First step prior to fusion is the calibration of the two sensors, the state of art describes two methods intrinsic and extrinsic calibration. Well-calibrated sensors are the prerequisite for accurate and robust multi-modal perception systems.

Calibration of sensors is a challenging task, as poor calibration of sensors will lead to less impact of fusion techniques and divergence of results. LiDAR and the Camera are both extrinsically calibrated by finding the transformation and rotation matrix and later identifying 3D calibration targets. Extrinsic calibration combines the information from multiple sensors into a common coordinate frame by identifying the relative position and direction of the sensors. Before it, intrinsic parameters are calibrated using **Zhang Calibration** as discussed in [1]. Tools used for calibration are a calibration object board and calibration sphere. By Zhang's method as described in [8] the checkerboard method is used for intrinsic camera calibration. For LiDAR, the position of the front LiDAR with respect to the reference frame is measured to be more accurate than all the sensors' positions are located with reference to the front LiDAR. For external calibration we use a calibration sphere to calculate the centre of the calibration sphere from LiDAR and the camera. We set a threshold to judge the projection of the point cloud to an image.

2.3. Techniques for Fusion:

Three methods of fusion exist as described in the paper [4][13]. Pre Fusion, Post Fusion, and Middle Fusion techniques. Early Fusion has low computation and low memory budget but is less flexible and sensitive to temporal data. On the other hand, late fusion is highly flexible and modular but requires huge computation costs and memory requirements. Lastly, middle fusion is the combination of early and late fusion that makes it highly

flexible yet determining the optimal intermediate layer to fuse the two sensing modalities is hard to determine. Here we will describe two fusion techniques as described by [14] and [6] respectively.

The advantage of the method as described in [14] is it allows the fusion of 2D lidar with 3D stereo camera. It shows that due to the fusion of stereo camera with 2D LiDAR we can detect objects that could not have been detected using only LiDAR. The disadvantage of this method is that by using only 2D LiDAR and not 3D LiDAR a sizeable portion of data in terms of heights is lost which can be used to improve the accuracy of the object detection algorithm.

2.3.1. Method for Pre Fusion:

Normally there is one Global disparity interval D for the entire Image, however this paper proposes to use LIDAR estimate to produce a range which can be used as an estimate for each pixel. There are two advantages to this, as it allows the optimization to not fall into a local minimum and reduces the computation as the disparity interval D is reduced so does computation time.

By using the proposed techniques in [6], we can perform accurate depth estimation on texture less surface and be able to reduce the time required for depth estimation however the algorithm has some difficulty when lidar data is not properly acquired such as reflection from the windscreen which causes the algorithm to fail in that particular region.

In the paper [15], it was shown by using sonar and odometry sensors for localization of autonomous vehicles that by using Kalman filter based sensor fusion. It was shown that by applying Adaptive Fuzzy logic system to monitor the covariance of the Kalman filter allows for better results from the filter. However, without the use of Adaptive Fuzzy Logic system, the fusion algorithm does not output smooth motion for the autonomous vehicle.

Vineet Gandhi proposes in [16] a novel growing correspondence algorithm which fuses time of flight LiDAR sensors and stereo camera to produce accurate depth maps. However, the disadvantage of their implementation is that it takes 5 seconds to calculate a depth map of an image, however it was shown in [17] that this technique can be implemented real time by using a normal CPU without parallel hardware.

2.4. Fusion Algorithm:

For optimal results, the measurements of different sensors are fused using different fusion architectures. While many different fusion algorithms are currently in practice and research continues to produce an improved version of the algorithm, at present most methods in practice today are deep learning code, Extended Kalman Filter (EKF), and Adaptive Fuzzy Logic system (AFLS). [9]– [11][34]

2.4.1. Adaptive Fuzzy Logic:

In adaptive fuzzy systems we model the process as a fuzzy system and parametrize it. We also apply an adjustment mechanism which allows the controller to monitor the output of the system and change the parameters of the system to get the required output.

2.4.2. Deep Learning:

It is a subbranch of Machine Learning which allows the computer to learn weights from input images by training on them. So, we provide input images and their respective depth maps to the deep learning algorithm and after it trains, it learns the parameters to calculate depth map using just input images.

2.4.3. Kalman Filter:

It is an algorithm designed to estimate the values for unknown variables by factoring in statistical noise and other inaccuracies. Usually, our sensor provides us with a value. It is done so by taking the average of multiple sensor inputs. However, this process takes a long time which is why it cannot be used in real time problems as perception of a self-driving vehicle. So, what Kalman filter does is that it takes a measurement from the sensor and predicts the next value for the sensor while obtaining input from the sensor also, if the value from the sensor turns out better than its prediction then it assigns more weight to the input from the sensor otherwise it relies more on the predication value. This allows the algorithm to quickly reach the real value of the sensor instead of waiting for multiple inputs from the sensor. Kalman filter is a very useful filter, however its disadvantage is that it only works for linear systems so another filter must be used for non-linear problems.

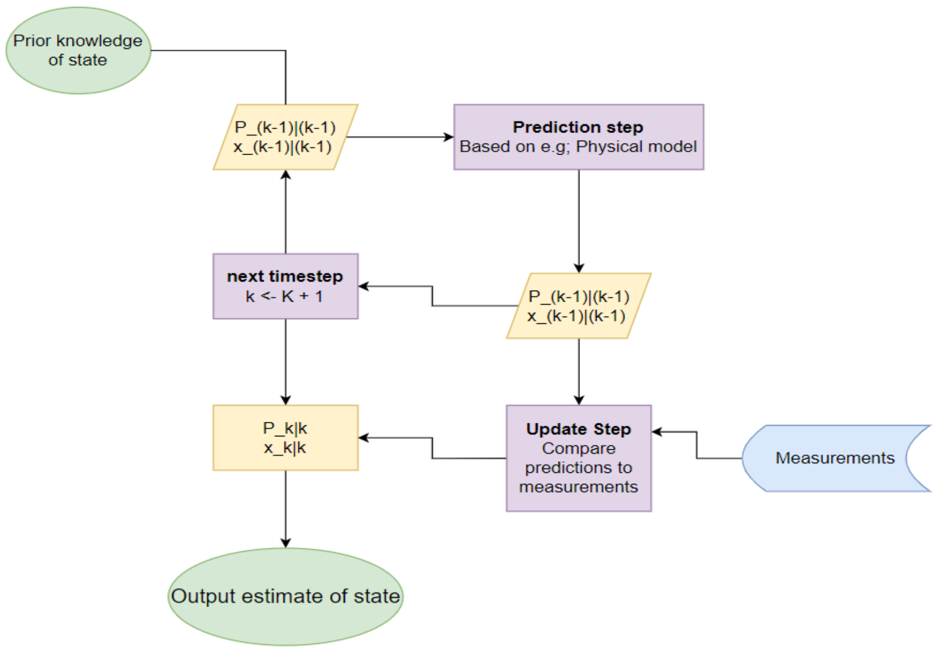

Figure 2. Flow chart of Kalman filtering

2.4.4. Extended Kalman Filter:

Due to the Kalman filter working only on linear problems, the extended Kalman filter was designed to counter this problem, it linearizes the problem by using Taylor series for a model around a certain working point. However, the disadvantage of Extended Kalman Filter is that if the initial estimate is wrong or the process is not modelled correctly, it can quickly diverge from the solution[35] [41].

The equations of discrete time predict, and update are as follows:

Note: Notation $x_{n|m}$ represents the estimation of x at time n given observations up to and including at time $m \leq n$.

$\boldsymbol{Predict\ equations}$:

Predicted state estimate:

$$x_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \tag{2}$$

Predicted covariance estimation:

$$P_{k|k-1} = (F_k)\big(P_{k-1|k-1}\big)(F_k^T) + Q_k \tag{3}$$

$\boldsymbol{Update\ equations}$:

Innovation or measurement residual:

$$\hat{y}_k = z_k - h(\hat{x}_{k|k-1}) \tag{4}$$

Innovation or residual covariance:

$$S_k = (H_k)(P_{k|k-1})(H_k^T) + R_k \tag{5}$$

Near optimal Kalman gain:

$$K_k = (P_{k|k-1})(H_k^T)(S_k^{-1}) \tag{6}$$

Updated state estimate:

$$x_{k|k} = x_{k|k-1} + (K_k)(y_k) \tag{7}$$

Updated covariance estimate:

$$P_{k|k} = [I - (K_k)(H_k)] * P_{k|k-1} \tag{8}$$

Where the **state transition** and **observation matrices,** respectively, are defined to be the following Jacobians:

$$F_k = \frac{\partial f}{\partial x}|_{\hat{x}_{k-1|k-1,u_k}} \tag{9}$$

$$H_k = \frac{\partial f}{\partial x}|_{\hat{x}_{k|k-1}} \tag{10}$$

After sensor calibration and annotating datasets, different fusion architectures are explored to obtain the most reliable techniques. [3] propose a method known as the Sequential Forward Floating Selection algorithm and a T transform method as the best alternative approach to selecting the most effective sensor fusion architecture. The methodology of when to fuse and what to fuse is explained by [4]. The point clouds obtained via LiDAR sensors and RGB images from the camera are processed separately and later fused. The optimal method is to process both point clouds and RGB images using 2D convolutional layers. Different techniques compared and illustrated in the paper by [4] were addition, ensemble, and concatenation of feature maps, where the latter is the most common one. The fusion of multi-sensors also followed several ways with their own advantages and disadvantages. Different fusion methods are early fusion, late fusion, and middle fusion. Early fusion has low computation and low memory budget but is less flexible, and sensitive to spatial-temporal data. Late fusion is highly flexible and modular but has high computation costs and memory requirements. Lastly, middle fusion is highly flexible but finding the optimal intermediate layer to fuse the sensing modalities is hard to find.

2.5. Datasets:

To perform the sensor fusion algorithms, multiple datasets are available online covering diverse scenes through different sensors. [8] aims to develop a dataset with diverse weather conditions. The dataset was acquired by using 4 LiDAR and 6 cameras mounted on vehicles and had images of sunny, rainy, foggy days. Data is collected from Oct – Nov'20 and recorded in ROS kinetic kame system. The many other datasets [17] available for research in the field include KITTI, Cityscapes, Apollo, Tusimple, and Waymo. The common thing is that all of them attempt to provide multimodal information in various conditions to enrich the scene library of auto-driving models. In OpenMPD, the dataset is annotated in the form of 2D and 3D bounding box annotations and 2D and 3D semantic segmentation, all in a single keyframe [12].

TABLE 2. COMPARISON OF STATE-OF-ART DATASETS [8]

Datasets	Year	Annotations	Degree of Camera View	Annotation of Point Cloud	Point Cloud Beams
NuScenes	2020	3D bounding box	360	3D bounding box	32
H3D	2019		180		64
Apollo Scape	2019	2D semantic segmentation	360	2D semantic segmentation	
A2D2	2020	2D semantic segmentation	270	2D semantic segmentation	5*16
		3D bounding box		3D bounding box	
Waymo Open	2020	2D bounding box	270	2D bounding box	75, 4*20
Level 5 lyft	2019	2D bounding box	360	2D bounding box	64, 2*40
KITTI	2012	2D bounding box	180	2D bounding box	64
		3D bounding box		3D bounding box	
OpenMPD	2021	2D bounding box	360	2D bounding box	128, 40, 2*16
		3D bounding Box		3D bounding box	
		2D semantic segmentation		2D semantic segmentation	

Chapter 3 -SENSOR FUSION FRAMEWORK

3.1. Overview:

This chapter explains the entire sensor fusion framework implemented in this project, the structure for this chapter is as follows. First the sensors used in this final-year project are introduced. Then the specification of those sensors is explained. Following that the software used to implement the sensor fusion is explained. Finally, it explains the detailed procedure of how to implement the multi-modal fusion algorithm on the data acquired from 2D LiDAR and stereo camera sensing modalities.

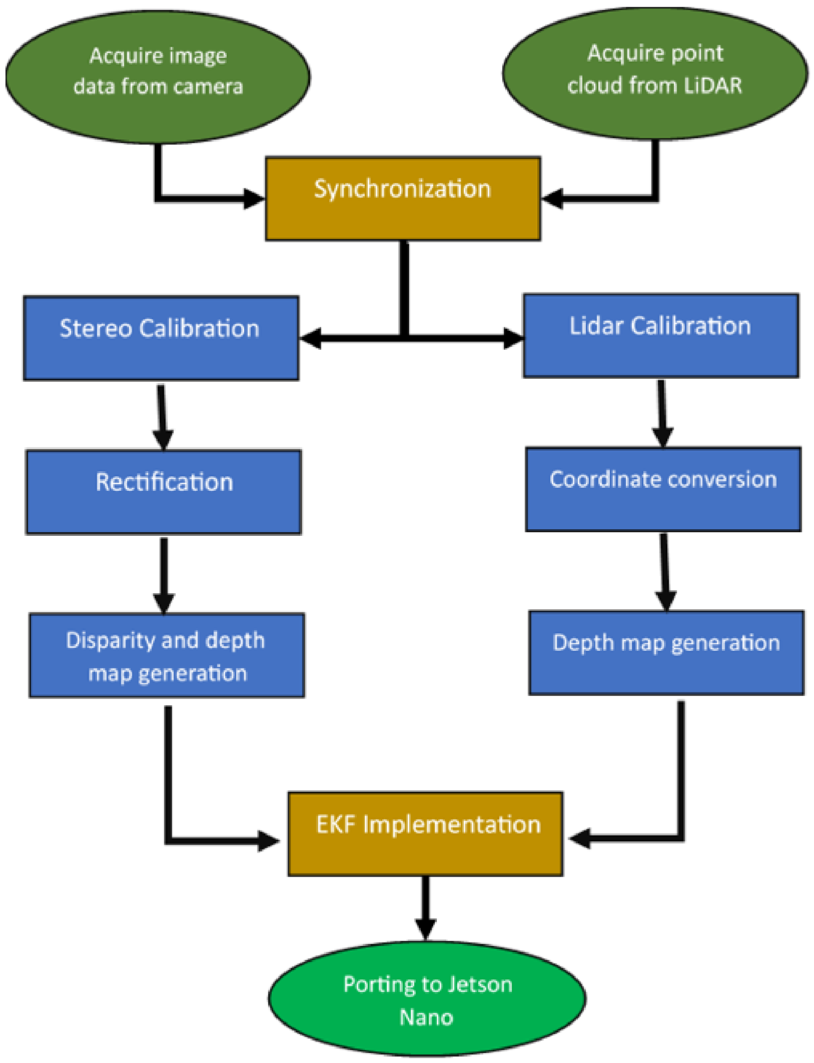

Figure 3. Flowchart of Depth perception algorithm

3.2. Hardware Specification:

3.2.1. LiDAR

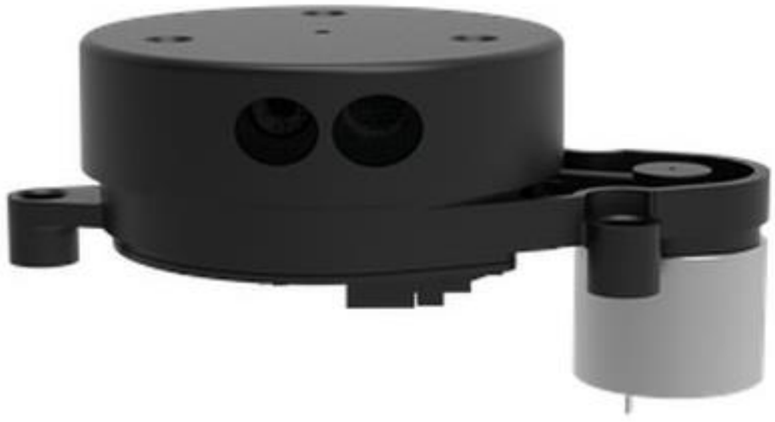

Figure 4. Image of YD-LiDAR TX20

LiDAR stands for Light Detection and Ranging. It is a sensing device used to examine the nature of the environment. It works by emitting light as a pulsated laser sensor beam and producing a 3-dimensional shape of the surrounding. It allows the user to examine the nature of the environment with better precision, accuracy, and flexibility[36].

The product application used in this project is YD-LIDAR TX20. It is a 360-degree two-dimensional LiDAR. It is equipped with advanced optics, electricity, and algorithm design to provide a distance measurement with better frequency and precision. The device rotates with 360 degrees angle of revolution to provide continuous results of angle information and point cloud data.

a) **Development kit:**

The development kit of YD-Lidar is designed to facilitate the user's performance. By using the development kit and matching evaluation software, point cloud data of the environment is scanned and can be visualized on a computer screen.

It has following parts:

- TX20 LiDAR,
- USB Cable, and
- USB Adapter board.

b) Product Specifications:

TABLE 3. SPECIFICATIONS OF LIDAR

Parameters	Items	Description
Product Parameter	Ranging frequency	4000 Hz
	Ranging distance	0.1 - 20 m
	Scanning angle	0 ~ 360°
	Motor frequency	5 - 12 Hz
	Ranging accuracy	+/- 4 cm
	Angle resolution	0.63°
Electrical Parameter	Supply voltage	4.8 - 5.2 V
	Voltage ripple	50 - 100 mV
	Starting current	200 - 400 mA
	Working current	200 - 380 mA
Data Communication	Baud rate	115200 bps
	High Signal level	1.8 - 3.5 v
	Low level signal	0.5 V
Optical Characteristics	Laser wavelength	905 nm
	Laser power	15 W
Other	Operating temperature	20ºC
	Lighting environment	8000 Lux
	Weight	100 g

3.2.2. Stereo Camera:

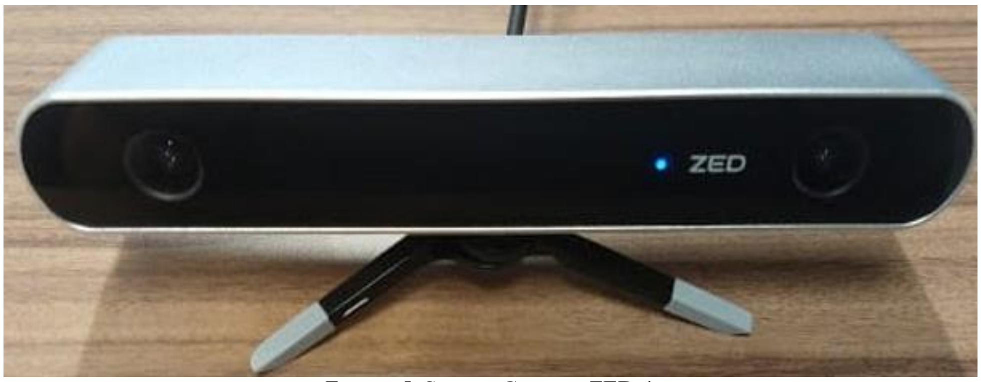

FIGURE 5. STEREO CAMERA ZED-1

Another sensor that is used in this project is ZED-1. ZED is a stereo camera from Stereo Labs, which provides a robust measurement of environmental depth. It is used in various applications such as security systems, aerial surveillance, intelligent systems, and remote sensing.

ZED-1 has a tracking API that allows its user to monitor objects captured by the camera. This API also allows the user to use the neural depth sensing mode that computes depth twice as fast as compared to other depth sensing modes given in the camera such as quality or performance. Therefore, allowing for more computations in less time.

Additionally, ZED-1 also provides OpenCV support which was used in this project for rectification and depth map generation, coupled with wide VGA support. It also boasted a superior field of view and improved image quality, elevating the overall visual experience.

Table 4 shows the technical specifications of ZED-1 stereo cameras.

TABLE 4. ZED-1 SPECIFICATIONS

Dimensions	175x30x30
Weight	170 g
Power	380mA /5V USB powered
Operating temperature	0° C to 45 ° C
Baseline	120 mm (4.7”)
Depth range	0.5 m to 25 m
Depth Map resolution	Native video resolution (in Ultra mode)
Depth Accuracy	<2% up to 3m <4% up to 15 m
Output resolution	Side by Side 2x (2208x1242) @ 15fps 2x(1920x1080) @ 30fps 2x(1280x720) @60fps 2x (672x376) @ 100fps
Field of View	Max. 90° (H) x 60° (W) x 100° (D)
RGB sensor type	1/3” 4MP CMOS
Active array Size	2688 x 1520 pixels per sensor (4MP)
Focal length	2.8mm
Interface	USB 3.0 – Integrated 1.5m cable
Shutter	Electronic synchronized rolling shutter

3.3. Software:

- **Python:** The main benefit of using Python is its scalability, as we can process large data sets. It is efficient as it can manage data and process it without consuming excessive memory. It is versatile, it tends to be applicable to a wide range of tasks such as data processing, web development, and scientific computing. Lastly, it has a lot of libraries that allow for easy processing of computer vision-related tasks, making it the suitable choice to use in this project. The most common libraries used in our project to implement codes are as follows:

    1) **OpenCV:** A Python library used for computer vision and image processing tasks. It provides easy-to-use functions for loading, manipulating, and saving image data. It was used for the display and saving of depth maps in this project.

2) **NumPy:** This library is used for numerical computation. It can be used to manipulate large matrices and vectors. Different mathematical models like Fourier analysis, and random number generation can be performed using NumPy. In this project, the library was used to create the system matrices and for computation of the Extended Kalman Filter.
3) **Matplotlib:** This library is used for creating visualization and plots in Python. Different plots, for example line plots, scatter plots, bar charts, and histograms are easily visualized. It also supports 3D plots and visualizations. The library was used to display the LiDAR plot in real-time in this project.
4) **Anaconda:** It is a Windows-based software platform for the distribution of Python and R programming languages. It includes an exceptionally large repository of 1500 data science packages including NumPy, Pandas, and Matplotlib. It also allows the user to create a virtual environment with a tendency to install packages of different versions without conflicts. Other features, for example include package management, Jupiter notebook interface, execution and deployment of code, and integration of various libraries. In this project, anaconda was used for creating and storing the virtual environment.
5) **CUDA:** It is a computing platform and programming model developed by NVIDIA that uses Graphics Processing Units (GPUs) to speed up computing applications. Developers can program in common languages such as C++, Python, and MATLAB and express parallelism through extensions in the form of keywords. The CUDA Toolkit includes libraries, a compiler, and development tools to develop GPU-accelerated applications. When using CUDA, the workload is split between the CPU and thousands of GPU cores in parallel to speed up the compute-intensive part of the application. In this project, CUDA was used in the ZED SDK without which the stereo camera API would not work.

3.4. Algorithm generation:

3.4.1. Data Acquisition Using stereo camera:

To acquire data from stereo cameras, the ZED-1 API was to get images from the stereo camera. However, the stereo camera can also act as a USB Video Class (UVC) camera, from which 'A_1' and 'A_2' can be taken, then those images are manually rectified using the calibration parameters of the stereo camera giving 'A_1'' and 'A_2''. After that 'α' can be calculated. The reason for using ZED-1 API was that it returned better images as compared to the manually computed images [18].

3.4.2. Data Acquisition using 2D LiDAR:

To obtain 'β' from the 2D LIDAR we need to convert 'ρ' into 'β'. So, to receive 'ρ', YD-LiDAR API was used to collect 'ρ' and store it in an array for further processing [19].

3.4.3. Synchronization of LiDAR and STEREO:

The data from the two sensors was retrieved separately and now the procedure of synchronization of the two sensors is implemented [21]. The following steps were deployed for synchronization:

1) The stereo camera and LiDAR data points were loaded.

2) The LiDAR and stereo camera coordinate systems were defined.

3) The results were synchronized using timestamps, by forcing the LiDAR to output results matching the speed of the stereo camera.

4) Then we save both the inputs from stereo camera and LiDAR with the current timestamps

5) Then those synchronized inputs are stored for further processing such as rectification in the later steps.

3.4.4. Stereo Camera Calibration:

The stereo camera was calibrated using the in-built calibration application given by Stereo Labs which uses Zhang's calibration method. The application displays a checkerboard on the devices' screen. The stereo camera is calibrated by moving the camera at various distances from the checkerboard pattern, then which the camera takes pictures to compare the feature points of the left and right images. After this procedure, the calibration parameters are acquired from the py-zed library which is used to access ZED-1 API. After getting the parameters, images from the stereo camera are rectified [37].

The calibration parameters include two types of parameters which are explained below:

a) **Intrinsic parameters:**

   It refers to internal characteristics of how images are captured by the camera. Some intrinsic parameters include:

   - **Focal Length:** This is the distance between the camera sensor and the centre of the lens. It is usually expressed in pixels or millimetres.
   - **Image Centre:** It is the point where the optical axis of the camera intersects the image plane. It is usually expressed in pixels.
   - **Skew:** It refers to the non-perpendicular of the image plane with respect to the optical axis. It is usually close to zero for most cameras.
   - **Distortion Coefficients:** Distortion occurs when a camera lens fails to produce a straight line when imaging a straight object. There are two types of distortion, radial and tangential. The radial distortion coefficients are related to how much the lens curves and are usually denoted by k1, k2, and k3. The tangential distortion coefficients are related to how the lens is not aligned perfectly parallel to the image plane and are usually denoted by p1 and p2.

- **Rectification Parameters:** The rectification parameters are used to transform the images so that they can be processed more easily in stereo vision algorithms. The rectification parameters include the rotation matrix and translation vector between the two cameras, the projection matrix for each camera, and the disparity-to-depth mapping function

Accurate determination of these intrinsic parameters is crucial for achieving accurate 3D reconstructions from a stereo-camera system.

b) Extrinsic parameters:

It refers to the position and orientation of each camera in the stereo camera system relative to a common coordinate system. Here are the extrinsic parameters of a stereo camera calibration:

- **Rotation Matrix:** The rotation matrix specifies how one camera is rotated relative to the other camera. It is usually denoted by R.
- **Translation Vector:** The translation vector specifies the distance and direction between the two cameras. It is usually denoted by T.
- **Essential Matrix:** The essential matrix relates the corresponding image points in the left and right camera images. It is computed from the rotation matrix and translation vector and is denoted by E.
- **Fundamental Matrix:** The fundamental matrix describes the epi-polar geometry of the stereo camera system. It relates the corresponding image points in the left and right camera images and is denoted by F

Accurate determination of these extrinsic parameters is crucial for achieving accurate 3D reconstructions from a stereo-camera system. The extrinsic parameters can be computed by taking images of a calibration object from various positions and orientations, and then using these images to compute the relative position and orientation of each camera in the stereo camera system.

3.4.5. Stereo Camera Rectification:

Now calibration parameters can be accessed, which can be used to apply rectification on the stereo images to get rectified images.

Rectification is the process in which stereo images are transformed such that their epi-polar lines become horizontal and disparity between the points can be easily computed [38]. Without rectification, depth cannot be properly computed as the disparity will have errors.

So, the rectification parameters such as focal length, principal point and distortion coefficients are taken from the intrinsic parameters and the rotation matrix and translation vector are taken from the extrinsic parameters. To get these parameters the configuration file of the stereo camera was accessed where these calibration parameters were stored.

Using these parameters, the rectification transforms 'R_1' and 'R_2', the projection matrices 'P_1' and 'P_2', and the disparity-to-depth mapping matrix 'Q' are calculated, which are then used to calculate the rectification maps. The rectification maps make the epi-polar lines of the images aligned and horizontal, as shown in figure 6, then those maps are applied on the input images to get Rectified images.

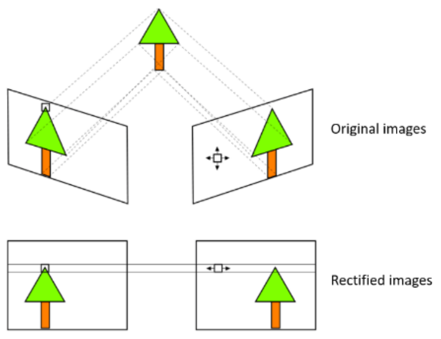

Figure 6. Rectification process simplified [43].

3.4.6. Disparity Map Calculation:

To generate the disparity map from the pair of rectified stereo images, the Semi-Global Matching (SGM) algorithm was used. The steps below explain how the SGM works in detail [39].

1) **Cost Computation:**
   Once the stereo images are rectified, the SGM algorithm computes a cost volume. The cost volume represents the dissimilarity or matching cost between pixels in the left and right images. For each pixel in the left image, the algorithm computes the cost for a set of candidate disparities (horizontal offsets) in the right image. The cost can be calculated using various metrics, such as the sum of absolute differences (SAD), sum of squared differences (SSD), or normalized cross-correlation.
2) **Cost Aggregation:**
   The cost aggregation step aims to refine the cost volume by considering the context and similarity of neighbouring pixels. It helps to reduce noise and inconsistencies in the disparity estimation. SGM performs cost aggregation by propagating and aggregating costs along different directions (e.g., left-right, right-left, top-bottom, bottom-top) within a local window around each pixel. The aggregation can be done using various techniques, such as computing the minimum cost along the aggregation paths or applying a weighted average of the costs.
3) **Disparity Optimization:**
   After cost aggregation, SGM performs disparity optimization to determine the best disparity value for each pixel. It aims to find the disparity that minimizes the overall cost, considering both the local cost and the aggregated costs from neighbouring pixels. SGM typically uses dynamic programming to find the optimal disparity configuration by evaluating different paths through the cost volume. The optimization process ensures that the selected disparity values are consistent and globally optimal.

4) **Disparity Refinement:**
   Once the initial disparity map is obtained, SGM applies refinement techniques to improve the accuracy and smoothness of the disparity estimates. This may involve sub-pixel refinement, where the disparity values are interpolated to sub-pixel accuracy by considering the neighbouring disparities and their corresponding costs. Sub-pixel refinement helps to achieve more precise depth estimation.
5) **Post-processing:**
   In the last step, post-processing techniques can be applied to further enhance the quality of the disparity map. This may involve additional filtering or regularization to reduce artifacts, fill in missing or occluded regions, and preserve depth discontinuities or edges. Common post-processing techniques include noise removal, hole filling, edge-aware filtering, and outlier rejection.

3.4.7. Depth map Generation:

After getting the disparity map from the previous step, the triangulation formula for depth is used to convert the disparity map into a proper depth map where f is the focal length of the ZED in pixels, u is the disparity map and baseline is the distance between the two cameras of the ZED 1 [40].

$$\alpha = \frac{f * baseline}{u}$$

3.4.8. LiDAR calibration:

Conversion of coordinates is needed to synchronize the depth map of LiDAR with a stereo camera [20][22]. For coordinates conversion, the following steps were followed:

a) First, the angle, range, and intensity data are extracted from the LiDAR point cloud into an array.

b) Using Python programming language and various mathematical libraries the angle values are changed from degrees to radians and the conversion of polar coordinates to Cartesian occurs using the formulas given below:

$$x = r * cos(\theta) \tag{11}$$

$$y = r * sin(\theta) \tag{12}$$

$where\ r = magnitude\ of\ vector\ in\ polar\ coordinates$

c) Then the transformed x- and y- coordinates are saved to another array for further post processing.

d) In the post processing step, we must remove all points which had an intensity value of 0.

e) The final Cartesian coordinates are transported to the plot, as shown in Figure 7, that displays the surroundings as captured by 2D LiDAR.

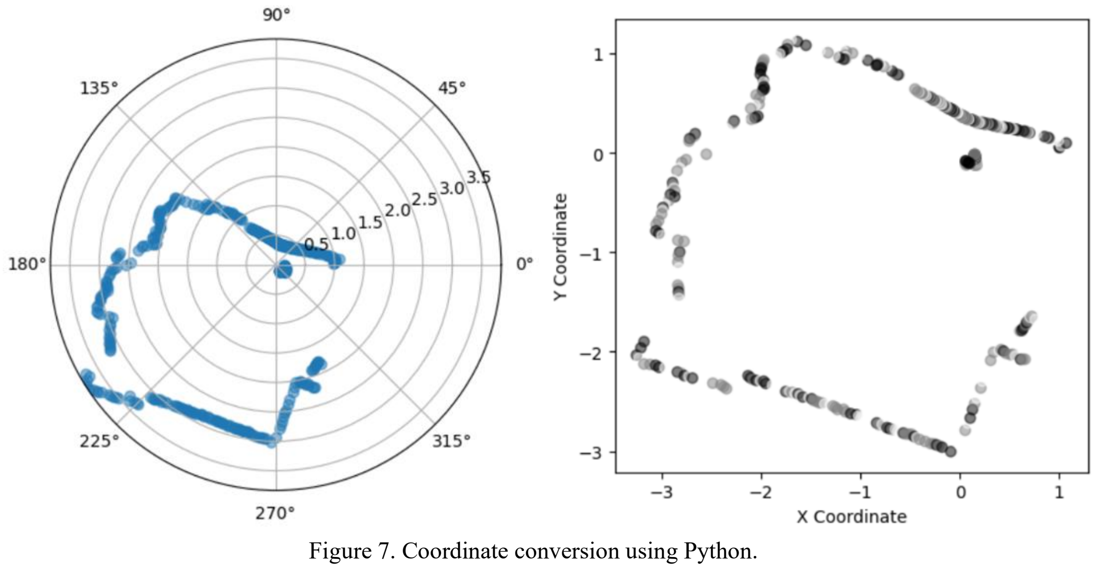

Figure 7. Coordinate conversion using Python.

3.4.9. Point Cloud to the depth map generation:

The next target was to generate the depth map that corresponds to the Cartesian points generated in the previous step using 2D LiDAR. To implement this, we followed the following steps:

a) The Cartesian coordinates are loaded into a separate array.

b) Determine the zero point of the point cloud.

c) Next, only the Y- axis values are selected for the points corresponding to the Field of View (FOV) of the stereo camera and store those in a separate array.

d) Finally, the 2D LiDAR depth map is sent to the fusion algorithm.

3.4.10. Sensor Fusion using Extended Kalman Filtering

The Sensor Fusion algorithm chosen to implement in this project is called Extended Kalman filter or EKF. EKF is a sub-technique of Kalman Filtering which is a powerful algorithm for estimating and predicting the state variable of the systems. These variables can be the position and velocity of the moving vehicle, here it is used to track the depth of each pixel in the depth map and update it based on the stereo and LiDAR measurements and the predication model.

The Kalman filter is used to estimate values for any system by relying on a prediction model and sensor measurements, where it estimates the next state of the system using the previous state estimate and the Kalman gain. It checks its estimates based on the next sensor input, if the next input is more accurate as compared to the model prediction more weightage is given to the sensor measurements but if the predication model is correct then less weightage is given to the sensor measurements in the next value prediction. For example, if Kalman gain is near to 1 then more weightage is given to the sensor measurement, but if the Kalman gain is close to 0 then more weightage is given to the predication in the next state estimate.

Below is the explanation regarding state model.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{13}$$

A is the state matrix of the model which shows that the system changes with time, where first value which is '1' represents the depth estimate but is taken as a constant since depth is not changing in a single image, while the other value ‘1’ in the diagonal represents the depth rate of change.

$$B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad (14)$$

B is the control matrix which relates the control vector ‘U’ to the state vector ‘X’ and since the system has no control input to the system then matrix B is also, zero.

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad (15)$$

C is the measurement matrix which relates the state vector ‘X’ to the measurement vector z, this matrix is used in the updated step of the EKF.

$$Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.04 \end{bmatrix} \qquad (16)$$

Q is the process noise covariance matrix which tells the system how much trust should be put in the model predications, it is a 2x2 representing covariance for depth estimates and depth rate of change.

$$\mathrm{R} = [0.1] \qquad (17)$$

R is the measurement noise covariance which tells the system how much trust should be put in the measurements from the sensors, which in this case are LiDAR and stereo camera.

$$U = [0] \qquad (18)$$

U is the control vector which shows the input to the system, which is zero in this case.

$$X = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad (19)$$

X is the state vector which contains the initial depth estimate and it is updated at each time step.

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (20)$$

P is the Error covariance matrix which shows how much uncertainty exists in the current state estimate, it is updated at each time step.

S is the innovation covariance matrix which gives the uncertainty between the predicated measurement and the actual measurement given by the sensors.

z is the measurement vector which contains the measurement of both LiDAR and stereo camera.

y is the measurement residual which calculated the difference between the predicted value and the actual measurement from the sensors.

K is the Kalman gain which is used to update the state estimate and process error covariance matrix, it decides which should be given more weightage in the next prediction, either the sensor measurements or the model predication.

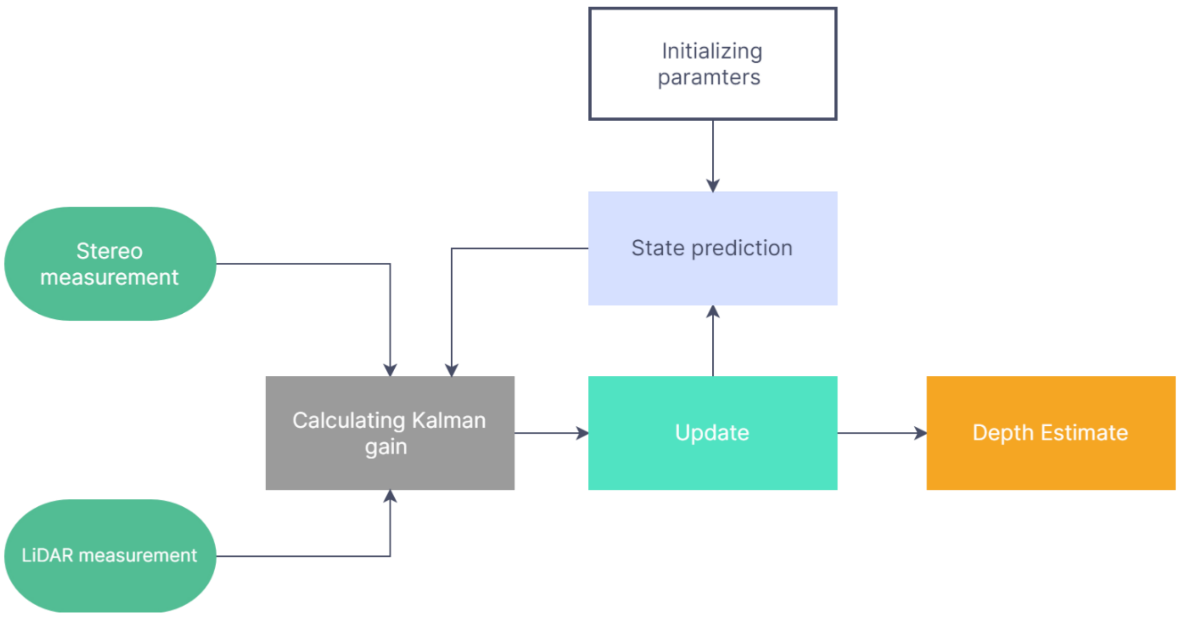

Figure 8. Flowchart for EKF implementation

The following steps are employed for the implementation of EKF:

1) Model and observation: First determine the state variable and observation to be used in the above equations. In the case of sensor fusion, state variable is the depth estimate of each pixel in the image, and measurement are the LiDAR and stereo camera measurements from their respective depth maps.

2) Initialization: Next, initialize the EKF model by providing initial value to state variable and covariance. The covariance matrix

determines the uncertainty in the initial state estimates.

3) Model forecast/ predictor: In this step, the EKF model predicts the state of the system at the next time step based on the initial values of state estimation and covariance matrix.

4) First the state estimate x is predicted based on the previous state estimate. This is done by taking a dot product with the matrix A the system matrix, while B the control matrix is taken as a dot product with u the control vector. However, since both B and u are zero their effect is neglected in the system.

$$x = (A * x) + (B * u) \tag{21}$$

5) Next the Error covariance P is predicted by using the previous error covariance and the system matrix A. Q which is process noise covariance is also added in at the final stage.

$$P = [(A * P) * (A.T)] + Q \tag{22}$$

6) Before Kalman gain can be calculated, the innovation covariance matrix S is calculated which is based on the measurement model and measurement noise R.

$$S = [(C * P) * (C.T)] + R \tag{23}$$

7) Next, we calculate the Kalman gain which is calculated using the innovation covariance matrix S and the error covariance matrix P.

$$K = \big(P * (C.T)\big) * S^{-1} \tag{24}$$

8) Update Step: Based on the new Kalman gain, the measurement residual y is calculated and using that and the Kalman gain, the state estimate x is updated for the next time step. In this same step the error covariance is also updated using the Kalman gain is used to update the state estimate and covariance matrix.

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \tag{25}$$

$$y = z - (C * x) \tag{26}$$

$$x = x + \big(K * (y.T)\big) \tag{27}$$

9) Data Assimilation/ Correction step: The updated depth estimate for that pixel is stored in a separate array and the step 3-9 are repeated for each pixel in the depth map.

3.4.11. GUI Implementation of the code:

In the project, a GUI code was developed to display the LiDAR point cloud, stereo depth map, and fused depth in a single window. The reason for developing this GUI interface was to create a presentable visual interface for displaying the results of the fused algorithm as well as the input depth map in one comprehensive window.

3.4.12. Porting algorithm on Jetson Nano:

The process of deploying the framework to the Jetson Nano platform involved the following steps:

1) The hardware requirements of the Jetson Nano were tested and set up.

2) The necessary libraries were installed on the system, and it was integrated with the PC using the available CUDA toolkit.

3) After installing the required libraries, the fusion algorithm was optimized to run on the limited resource of the Jetson Nano, one of which was rescaling the input depth maps to a smaller size for faster computation of the EKF algorithm.

4) The algorithm was tested on the Jetson Nano to ensure smooth running.

5) Finally, the code was deployed to the Jetson Nano platform.

3.5. Summary:

This chapter describes the framework used in this project, after following the above given procedures , the key outcomes are as follows:

- Pair of rectified stereo camera images were obtained from uncalibrated images.
- Disparity Map was obtained from rectified stereo camera images.
- Depth map was obtained from disparity map.
- Lidar depth map was obtained from point cloud.
- Fused depth map was obtained by fusing LiDAR and stereo depth maps using EKF.

The following chapter will describe the evaluation criteria used to test this fused depth map and the results obtained from this fusion. The algorithm will be tested in two different setting, first is the KITTI dataset and second is the real word dataset.

Chapter 4 -EXPERIMENTAL RESULTS AND ANALYSIS

4.1. Overview:

Chapter 4 presents the experimental results and analysis of the depth perception module. The chapter begins by describing the experimental setup, including the use of the KITTI dataset and real-world data. The KITTI dataset, a benchmark for self-driving cars, provides depth maps for evaluation. The integration of sensors on an All-Terrain Vehicle (ATV) and the utilization of the NVIDIA Jetson Nano Developer Kit for processing are explained. Qualitative results demonstrate the data acquisition process and visual representations of fused results. The chapter concludes by summarizing the key findings and insights derived from the analysis.

4.2. Data Collection and Experimental Setup:

To assess the accuracy and robustness of our depth perception module, we conducted experiments in two distinct environments. Initially, we implemented the fusion model and EKF algorithm using datasets obtained from the KITTI platform, which encompasses road scenes from rural areas and highways in Karlsruhe city. Subsequently, we tested our algorithm on real-world data collected from 2D lidar and stereo cameras situated in the vicinity of our university.

4.3. KITTI Datasets:

The KITTI dataset, developed by the Karlsruhe Institute of Technology and Toyota Technological Institute, is widely recognized, and employed in the research and advancement of mobile robotics and autonomous driving [42]. This dataset comprises information captured through an array of advanced sensing modalities, including Velodyne 3D LiDARs, stereo cameras, and GPS. The data was acquired by driving a vehicle within urban environments and encompasses high-resolution images, point clouds, and GPS maps. Researchers employ these datasets to evaluate various models for tasks such as object tracking, state estimation, and visual odometry. The KITTI benchmark has emerged as the primary choice for comparing and assessing the performance of different computer vision algorithms within the domain of self-driving cars.

We employed the “KITTI-Depth dataset” that provided depth maps from projected LiDAR point clouds captured using 3D LiDAR and matched against the depth estimation of stereo cameras. The dataset included 86k training images, 7k validation images, and 1k test set images on the benchmark server.

4.3.1. Evaluation Criteria:

Two types of datasets were used to evaluate the depth maps. The two datasets are:

- **KITTI dataset:**

  To assess the accuracy of the fusion depth map used in this project, Mean Absolute Error (MAE) is chosen as the evaluation criteria, the SGM algorithm is used to first calculate a depth map using the KITTI rectified images and store them separately as an SGM depth map. Subsequently, the LiDAR depth map of the same is integrated with a copy of the SGM depth map, resulting in a fused depth map.

  To evaluate the fused depth map, comparisons between KITTI's SGM depth map and fused depth map are performed against the depth map already provided by the KITTI dataset. By examining the differences between these depth maps, the consistency and quality of the fusion technique is assessed. Specifically, the **mean absolute error** is calculated of the fused depth map and SGM depth map as they are subtracted from the depth map provided by the KITTI data set. This meticulous analysis allows for the accuracy of the developed algorithm to be tested when applied to the KITTI dataset, shedding light on its efficacy in capturing the intricate depth information within the captured scenes.

- **Real-world dataset:**

  To calculate the standard deviation, mean of the depth values should be calculated. It is calculated by adding all of the depth values and dividing by the number of depth values. In this case, there are 15 depth values, and the mean is 1.3230.

After getting the mean value, variance is calculated, which is calculated by subtracting the mean from each depth value, squaring the difference, and then dividing by the number of depth values minus ‘1’. In this case, the variance is 0.0021.

At the end, Standard deviation is calculated. This is done by taking the square root of the variance. The standard deviation is 0.0139.

Since there was no ground truth to compare the sensor fusion algorithm running on collected dataset, so instead of calculating MAE, the variance and standard deviation were calculated to get an idea of the accuracy of the fusion model.

4.3.2. Testing on the KITTI Dataset:

The Fusion Algorithm was tested on the KITTI however the procedure was slightly different as compared to the standard framework, as the dataset provided pre-rectified images with stereo camera and LiDAR depth maps. First the stereo camera images were used to produce a disparity map using the SGM algorithm, after which a scaling factor of 16 had to be applied which allowed the actual values of disparity to be extracted from the disparity map. Then those disparity maps were converted in a depth map and fused with their corresponding LiDAR depth map to produce the Fused depth map. Finally, the above stated evaluation criteria were used to test the effectiveness of the fusion algorithm.

4.4. Hardware Setup:

To acquire data from sensors, and display the fused results on the screen, a prototype setup was made using a four-wheeler all-terrain vehicle. Sensors are mounted at the top of the vehicle at the measured distance using extrusion frames. The v-slots were set up in 3 layers to accommodate sensors, jetson nano, and supporting material.

To set up sensing modalities, cad modelling of sensor mounts was developed. This step is executed in the following steps.

4.4.1. Computer Aided Design:

A 3D model design of the sensor mount is developed using SolidWorks software. By identifying and measuring the right dimension in terms of shape and size, the 3D model is designed considering the requirements of a stereo camera, LiDAR, and Jetson Nano. The designed model accommodated the necessary clearance, and adjustments needed for fixing the sensors. The Computer Aided Design (CAD) model is exported in other formats for manufacturing and analysis.

1) Stereo camera:

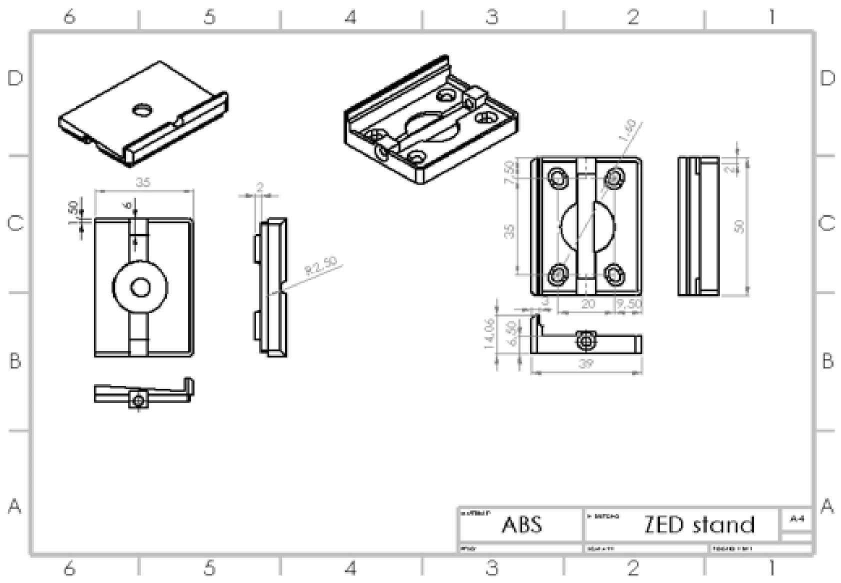

Figure 9. Cad Model of Stereo Camera Mount

2) LiDAR:

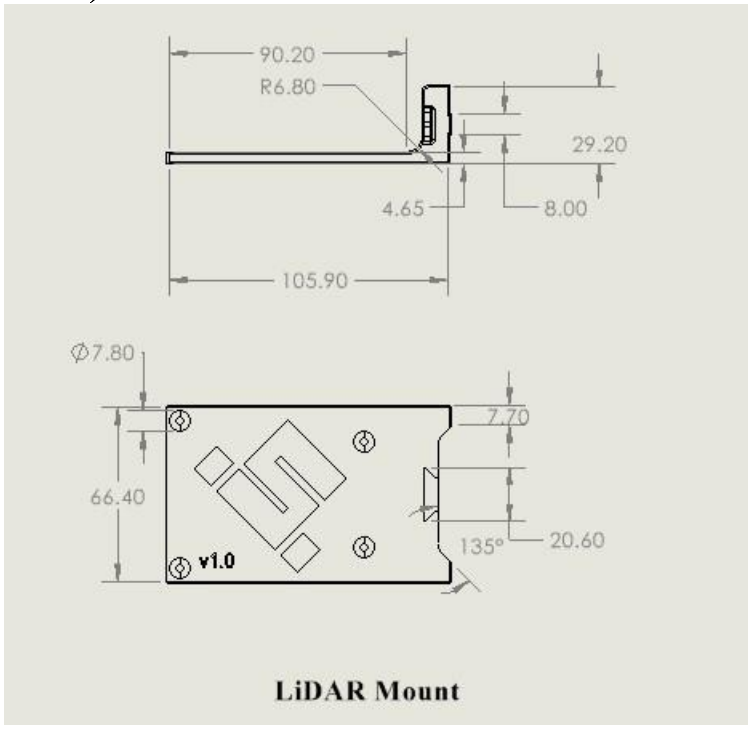

Figure 10. CAD Model of LiDAR Mount

3) Jetson Nano:

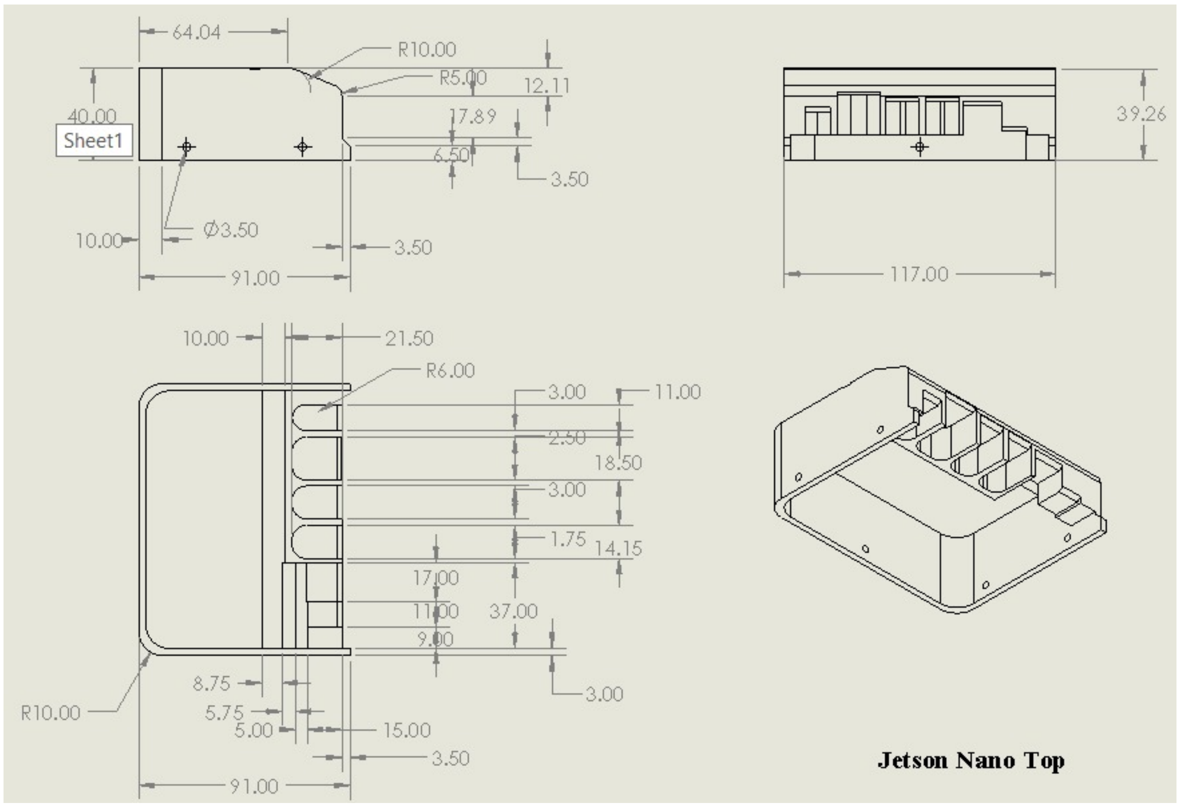

Figure 11. CAD Model of Jetson Nano Cover

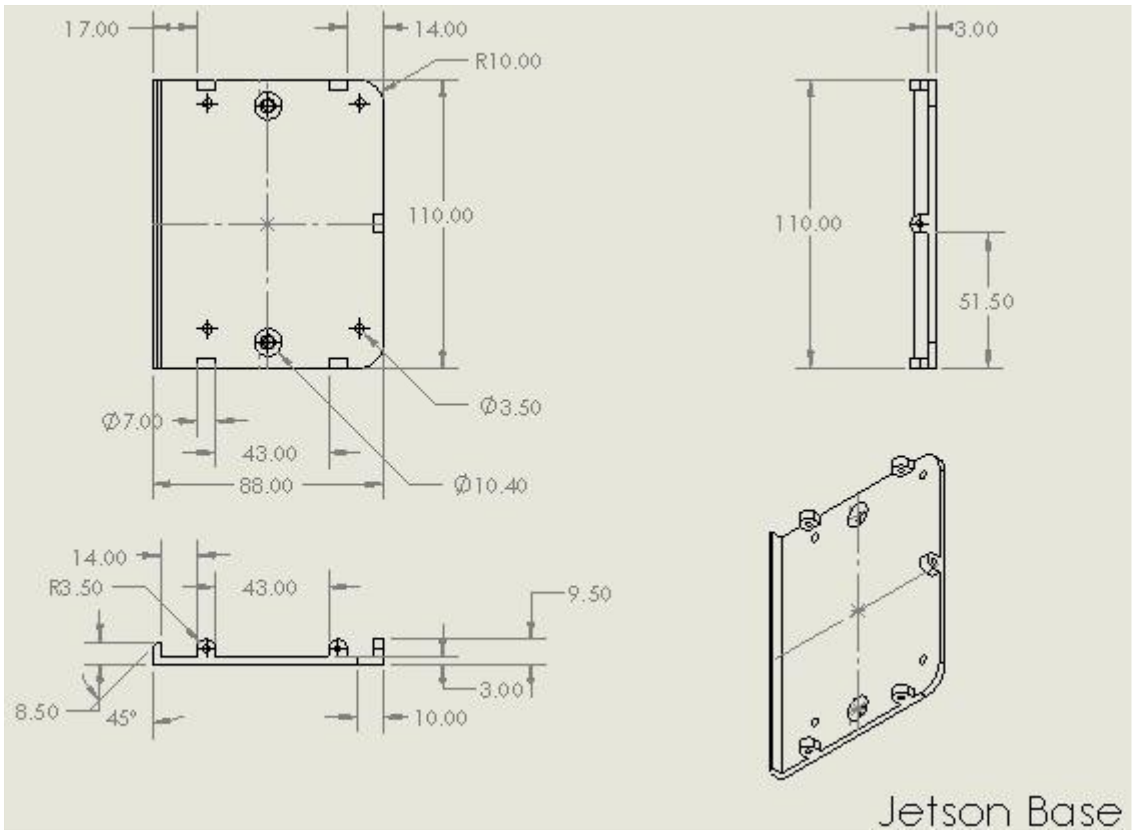

Figure 12. CAD Model of Jetson Nano Base

4.4.2. 3D Printing:

For the final product Polyethylene Terephthalate Glycol (PETG) material was used to print the 3D model. The reason for the selection of material is due to the high-temperature requirement, which helps with the proper printing of the material. It also possessed a high adherence capability with few errors. By importing the CAD model into a 3D printing software, we selected the suitable parameters, and a desired printing file is generated. Later this file is sent to a 3D printer using additive manufacturing to create the physical object layer by layer.

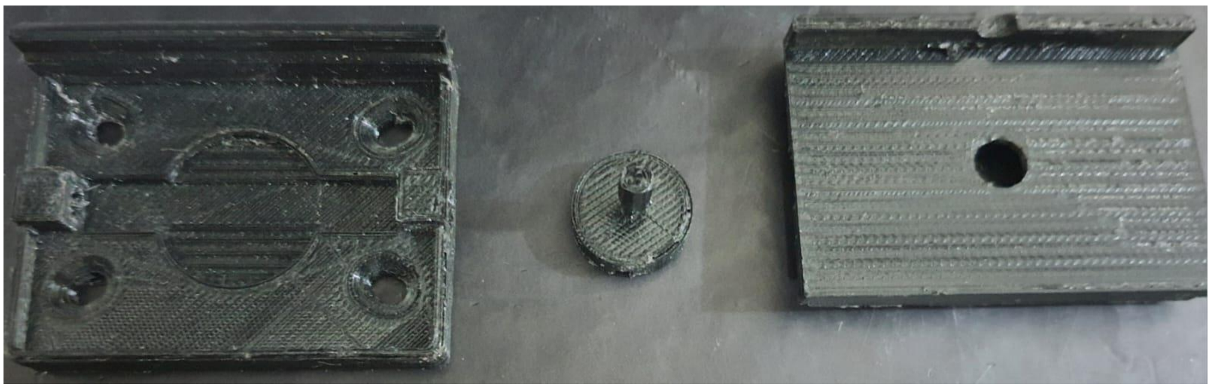
Figure 13. 3D Printed mounts

In summary, a sensor mount involved designing the CAD design to create a 3D model of the mount, and 3D printing to manufacture the physical part. By following these steps, the sensor mount was optimized for performance and reliability, which was essential for accurate and robust stereo vision.

4.4.3. All-Terrain Vehicle:

For proof of concept, we mounted LiDAR and stereo camera sensors on the All-Terrain Vehicle (ATV), to capture real-time data and process the algorithms accordingly.

ATV is a small, open, and agile vehicle with four or more wheels, designed for use on diverse types of terrain, including off-road conditions. ATVs are typically used for recreational purposes, such as racing or off-roading, but also have a variety of practical applications, including transportation and work-related tasks in agriculture, forestry, mining, and emergency services. Due to their manoeuvrability and ability to traverse difficult terrain, ATVs are popular in outdoor industries and remote areas where traditional vehicles may not be practical or accessible.

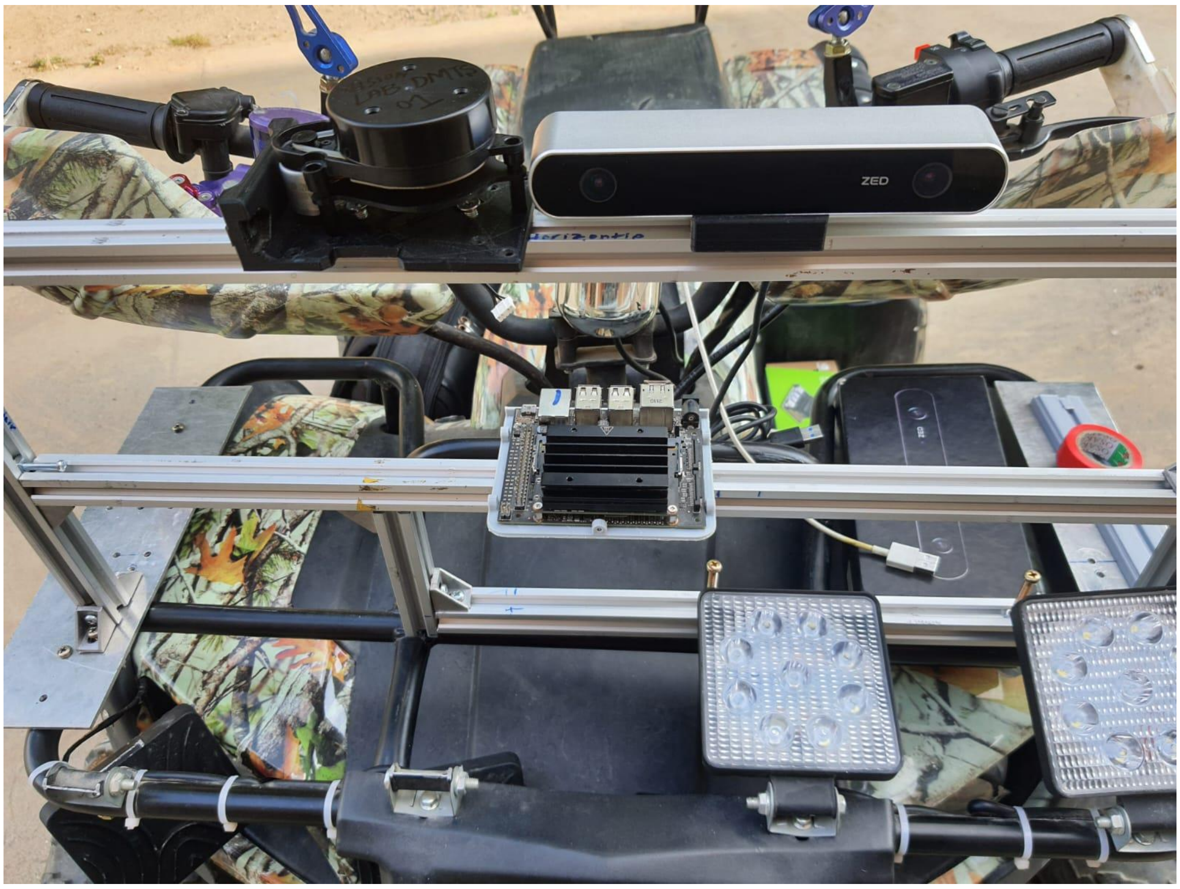

Figure 14. ATV with mounts

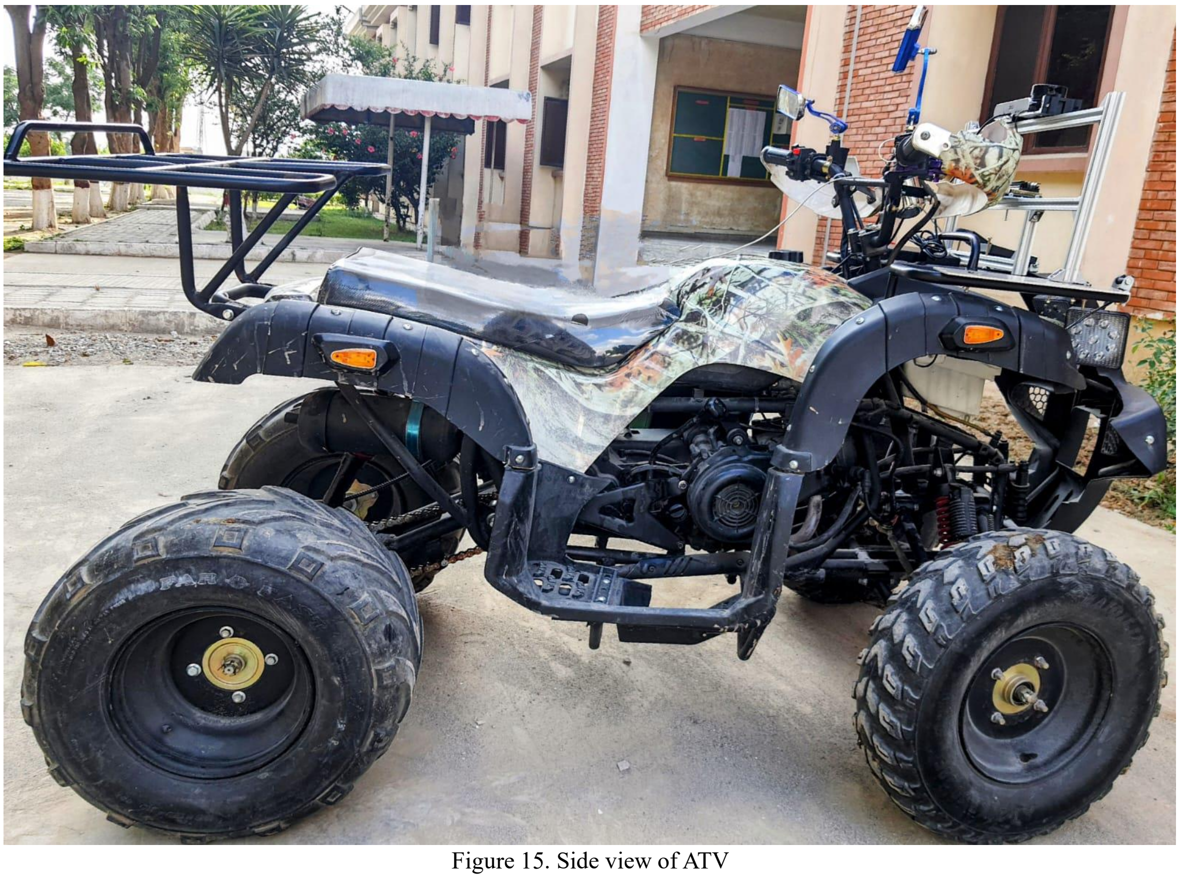

Figure 15. Side view of ATV

4.4.4. Jetson Nano

The NVIDIA® Jetson Nano™ Developer Kit is a compact and robust computer that enables the concurrent operation of multiple neural networks for tasks such as image classification, object detection, segmentation, and speech processing. It's user-friendly, low-power, and requires only a microSD card with a system image to get started. The NVIDIA Jet-Pack SDK, used across the entire NVIDIA Jetson™ product line, simplifies the development process by being compatible with the leading AI platform for training and deploying AI software. This reduces complexity and effort for developers.

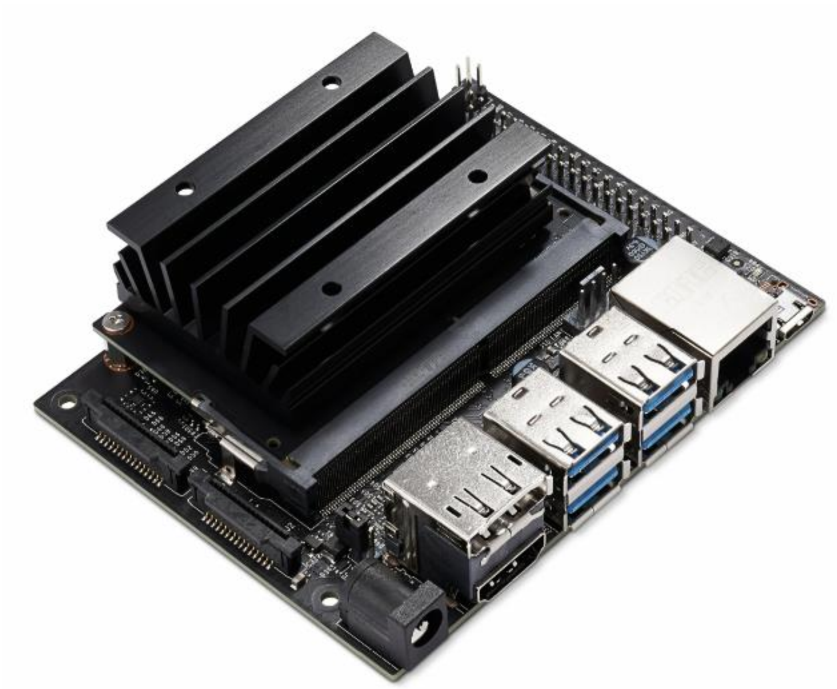
Figure 16. Jetson Nano kit

The Jetson Nano is a small and powerful computer designed for embedded applications and AI Internet of Things (IoT) devices. It costs $99 for a minimum order of 1000 modules. You can start using it quickly with the Jet-Pack SDK, which has features for deep learning, computer vision, graphics, multimedia, and other things. The Jetson Nano has the ability to manage modern AI workloads, making it simple to add advanced AI to your next product. To begin developing it, check out the Jetson Nano Developer

Kit helping guide. Jetson Nano Technical specifications are listed in table 5.

TABLE 5. JETSON NANO SPECIFICATIONS

Features	Description
GPU	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
CPU	Quad-core ARM Cortex-A57 MP Core processor
Memory	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Storage	16 GB eMMC 5.1
Camera	12 lanes (3x4 or 4x2) MIPI CSI-2 D-PHY 1.1 (1.5 Gb/s per pair)
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^{2}C, I^{2}S, SPI, UART
Mechanical	69.6 mm x 45 mm 260-pin edge connector

a) Functional overview:

The benefits of using Jetson Nano are to utilize the maximum computational performance of a high-power computer in limited environments. Integrating a multi-advanced video, audio, and image processing outline into 260-pin ISO-DIMM. The Maxwell GPU architecture combines several architecture designs to extract maximum performance per watt.

Core components of Jetson Nano include:

1) NVIDIA Maxwell GPU:

The Graphics Processing Cluster (GPC) in GPUs manages rasterization, shading, texturing, and compute, with multiple Streaming Multiprocessor (SM) units and a Raster Engine. The Maxwell GPU architecture introduced redesigned SMs, optimized data flows, and improved power management. Each Maxwell SM is partitioned into four processing blocks with their own instruction buffer, scheduler, and 32 CUDA cores. The SMM CUDA cores handle shading and compute, texture units perform texture filtering, and SFUs manage transcendental and graphics interpolation instructions. The

SMM geometry and pixel processing performance is suitable for complex gaming applications, and the power efficiency of the Maxwell GPU makes it suitable for devices with power-limited environments. It can be utilized to speed up complex processing such the fusion algorithm used in this project.

2) **ARM quad-core Cortex-A57 CPU Complex:**
   The CPU complex consists of a high-performance Multi-Core SMP cluster of four ARM Cortex-A57 CPUs with 2MB of L2 cache. It has features like dynamic branch prediction, TLBs, I-cache, D-cache, ARMv8 architecture, ETM, PMU, CTI, Cryptographic Engine, interface to an external Generic Interrupt Controller, and power management. The CPU frequency and voltage are actively managed by Tegra Power and Thermal Management Software, and the frequency may be throttled at higher temperatures. Observed chip-to-chip variance is due to NVIDIA's ability to maximize performance within the available power budget.
3) **4 GB Low Power Double Data Rate 4 (LPDDR4) memory:**
   The Jetson Nano has 4GB of fast memory, and the Memory Controller (MC) manages memory requests to make sure the memory is used efficiently. It has security features and power-saving modes. The MC is good at managing many different types of memory requests, especially for multimedia, and it saves power when not in use.
4) **16 GB storage:**
   Jetson Nano provides 16 GB of embedded multi-media card (eMMC) storage. This eMMC storage is helpful for storing operating systems, applications, and data. The benefit of using such types of storage is fast computation and independent working.

5) **Gigabit ethernet:**

   Gigabit ethernet port on Jetson Nano allows a local network or the internet to connect at a high speed of up to 1000 Mbps. Hence even with minimum latency, high-quality videos, and large files are transferred quickly. It also enables communication between Jetson Nano to computers, servers, and IoT devices. It also provides the feature of remote access and control.

6) **On-chip temperature sensors**

   There are several sensors mounted on Jetson Nano on various parts including GPU, CPU, and memory. These sensors guide the system by monitoring the temperature of the device and provide simultaneous feedback to the system's thermal management software which is helpful in adjusting the CPU frequency and voltage to optimize performance and avoid overheating. If the temperature exceeds the provided safe limit these sensors enable to trigger alarms or shut down the device completely.

b) Operation:

The working of Jetson Nano is based upon the following steps:

1) Familiarize with hardware components. Make necessary connections and integrate other devices that are not available in the developer kit.
2) Use a microSD card as a boot device and for main storage, and mini-USB power supply.
3) Next step is to set up the microSD card, for this first download the required software package and format the microSD card using the SD Memory card formatter from the SD association. Next using an etcher, write the Jetson nano developer kit SD card image to a microSD card. Once the microSD card is ready, process to developer kit.

4) There are two ways to interact with the developer kit. Either by attaching the kit with a display, mouse, and keyboard or in a "headless mode" via a connection from another computer.

TABLE 6. COMPARISON OF INITIAL SETUP USING HEADLESS MODE WITH DISPLAY ATTACHED.

	Initial setup with display attached	**Initial setup in headless mode**
Monitor, Keyboard, and mouse	Required	Not required
Extra computer	Not required	Required
Power options	Either Micro-USB or DC power supply can be used	DC power supply is needed

5) After initial setup, a green LED will blink as the developer kit is turned on.
6) Next using the NVIDIA Developer platforms online and user guide, explore more details and start running sample data provided there.
7) We are using Jetson Nano with display attached in our experimental setup where the display is taken on a laptop screen.

4.5. Qualitative results

This section presents the qualitative results obtained from the implementation of our algorithm. Firstly, we illustrate the data acquisition process, displaying the point cloud obtained from the 2D LiDAR sensor and the corresponding images captured by the stereo camera. These visual representations provide a clear understanding of the data captured by the sensors.

Subsequently, we conducted a comprehensive analysis using the data acquired from the KITTI dataset. By leveraging this dataset, we compare and evaluate our algorithm's performance. The fused results, which combine information from multiple sensors, are displayed and analysed. This visualization allows for a detailed examination of the effectiveness and accuracy of our algorithm in integrating and processing the data from different sources.

Through these visual representations and analysis, we gain valuable insights into the capabilities of our algorithm and its ability to provide accurate and meaningful results in the fusion of sensor data.

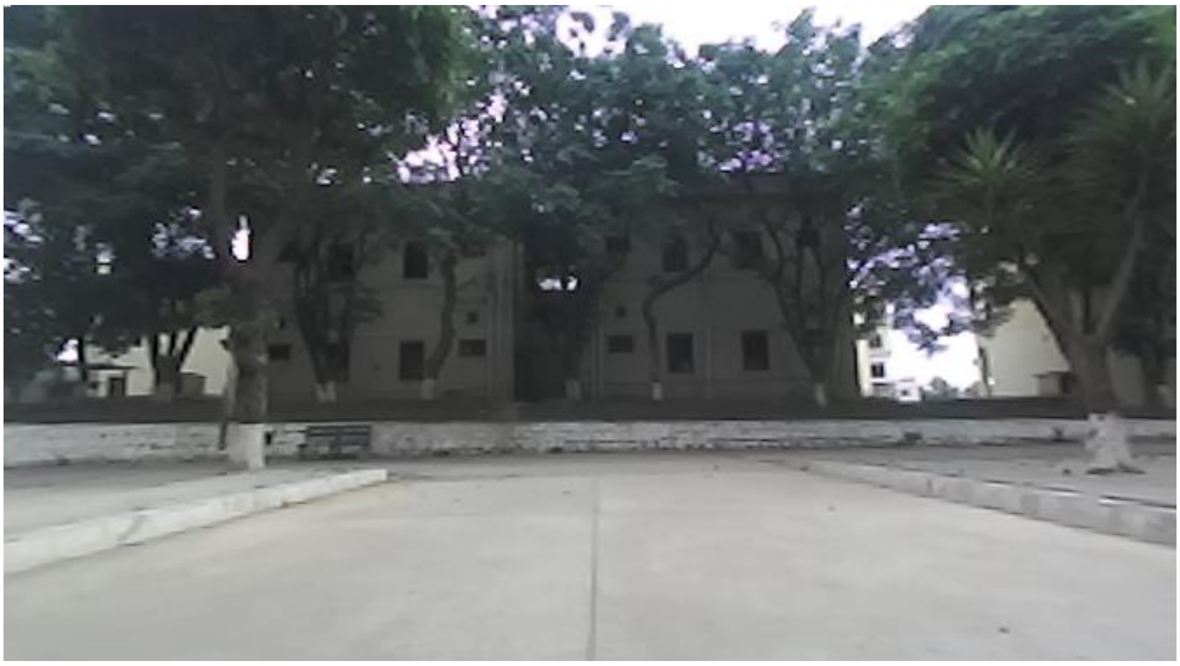

Figure 17. Raw Image from stereo camera.

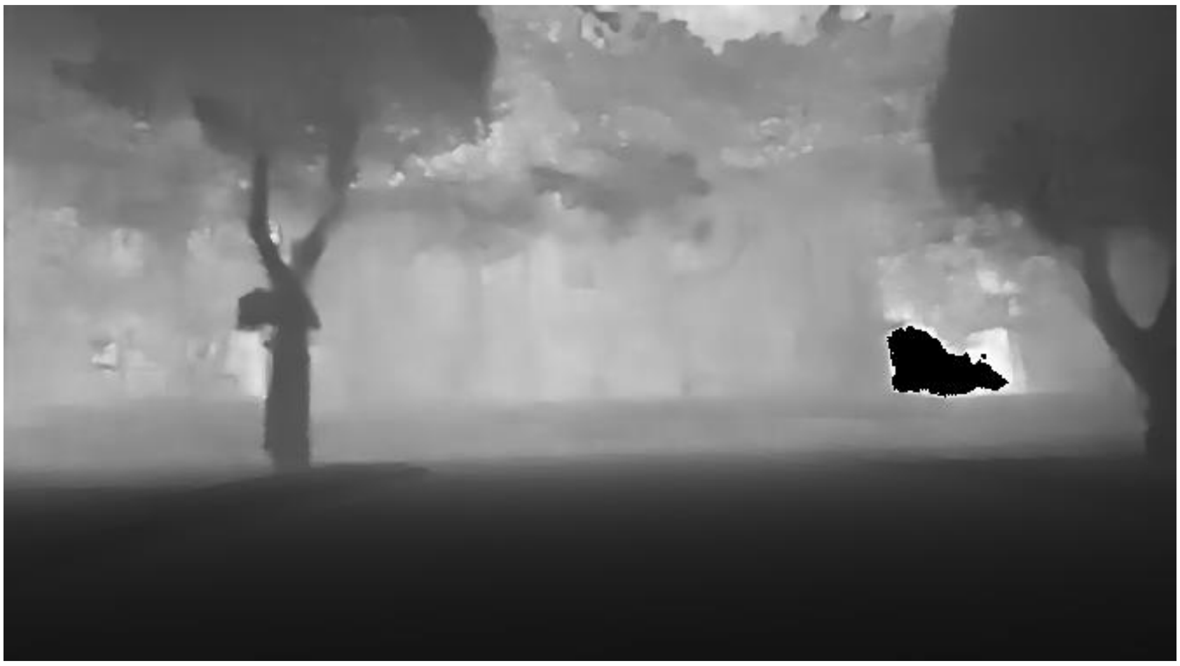

Figure 18. Depth map from stereo camera.

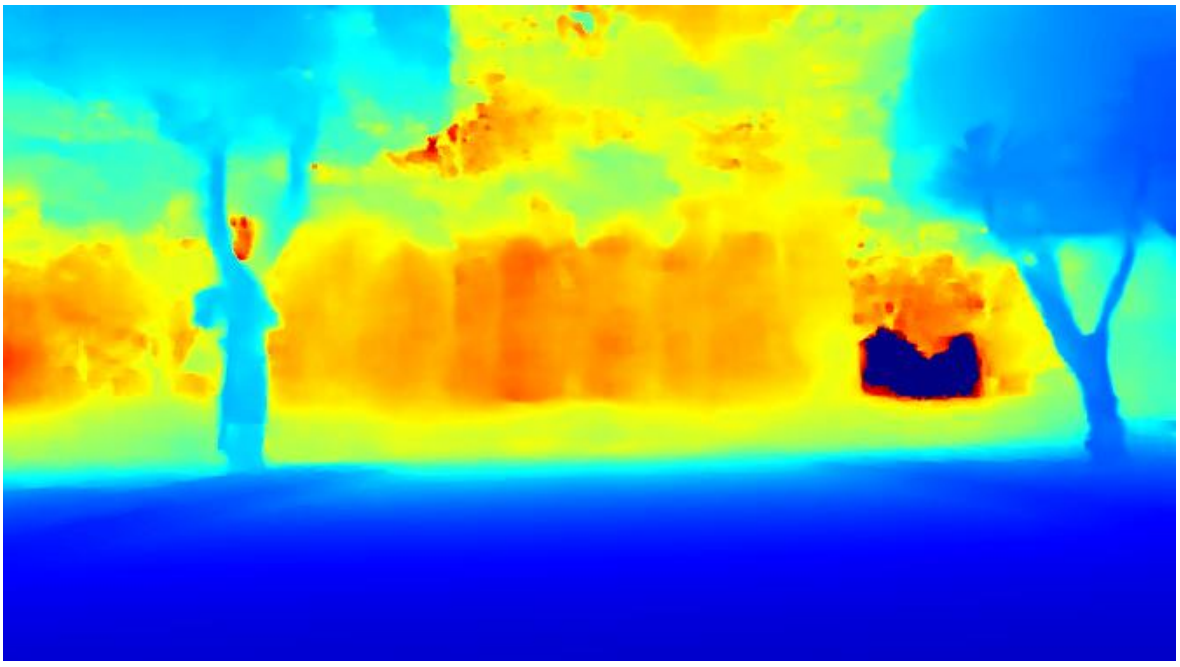
Figure 19. Fused depth map.

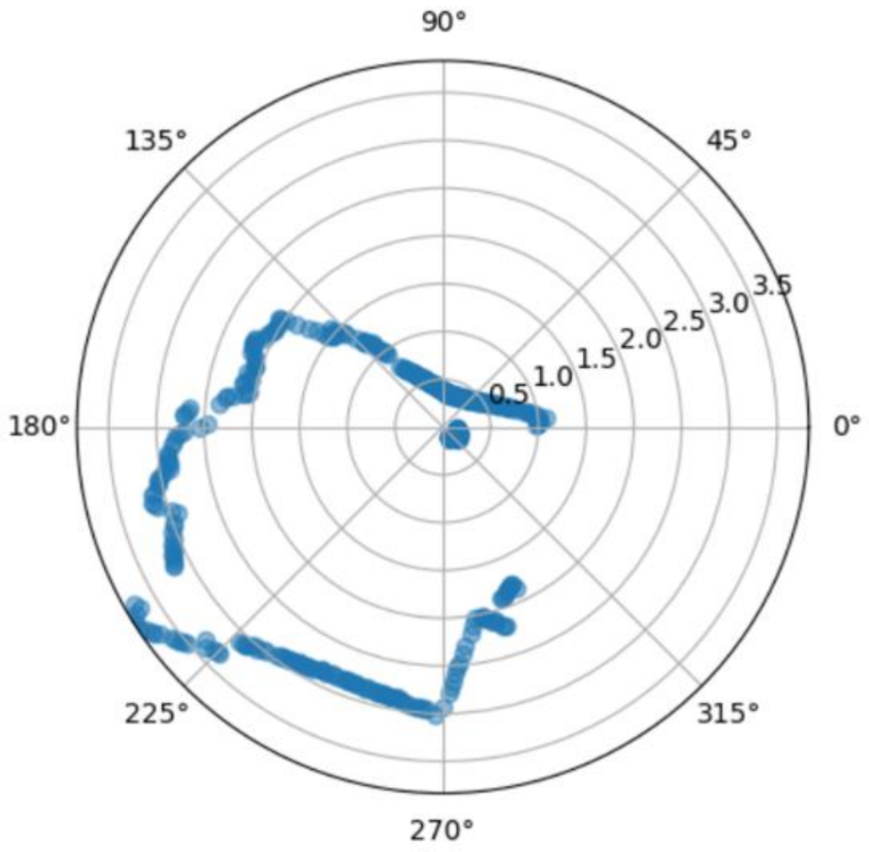

Figure 20. Real-Time Point Cloud Data Acquisition from 2D YDLIDAR

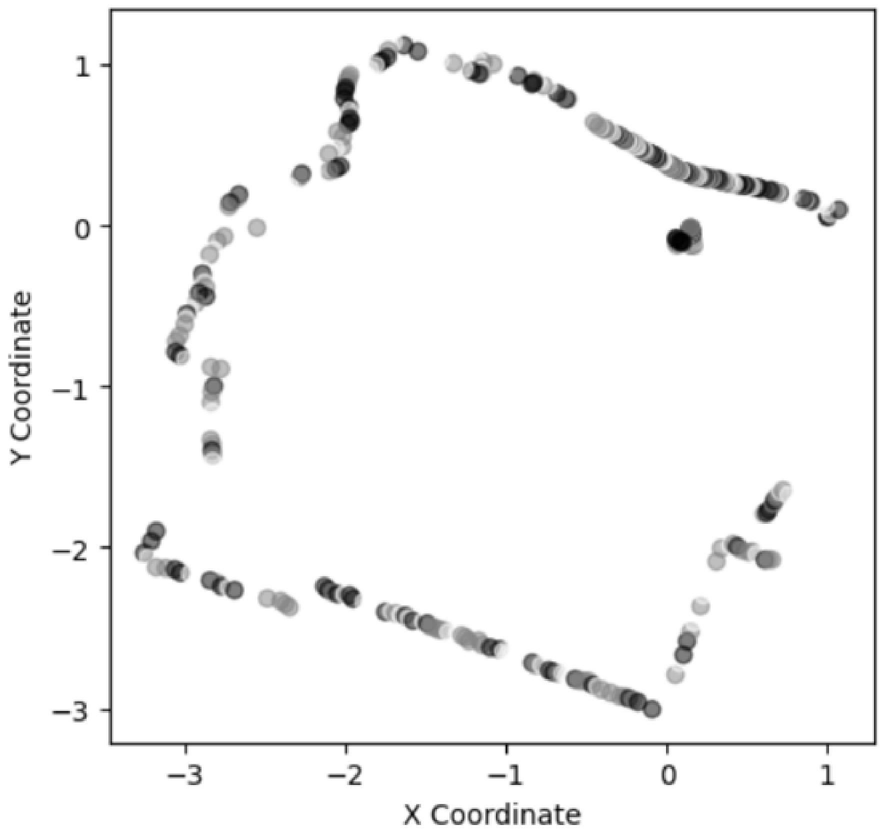

Figure 21. Conversion of Polar Coordinates to Cartesian Coordinates

4.5.1. Fused results:

We have assessed our algorithms on three different environmental conditions:

- City
- Road
- Residential

Results are displayed in the following way:

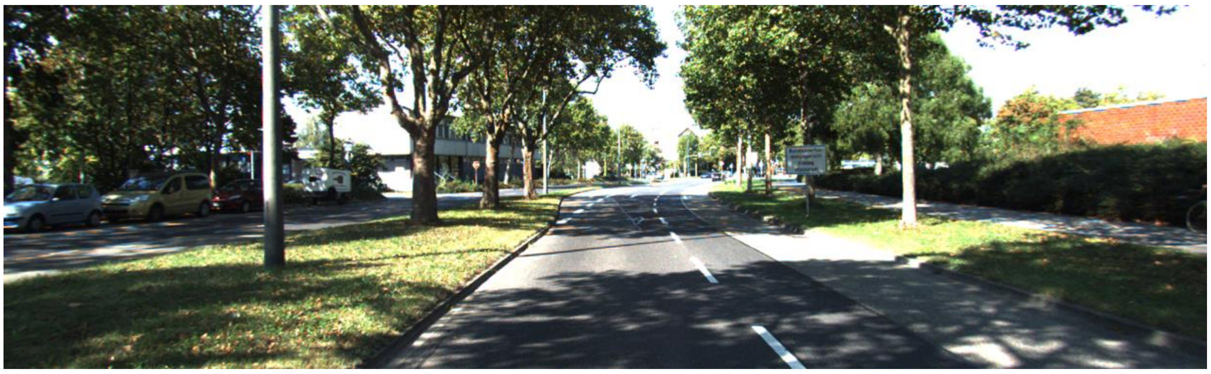
Figure 22. Input image - City

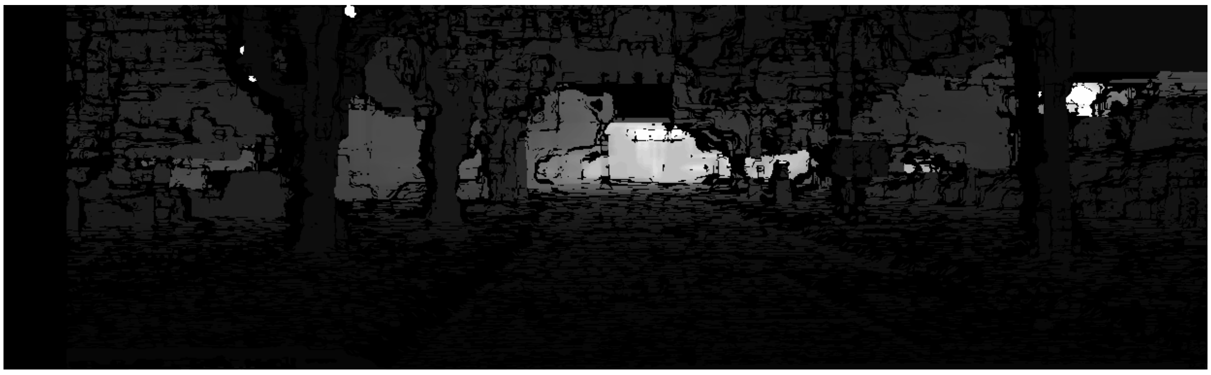
Figure 23. Depth map from stereo camera of the city.

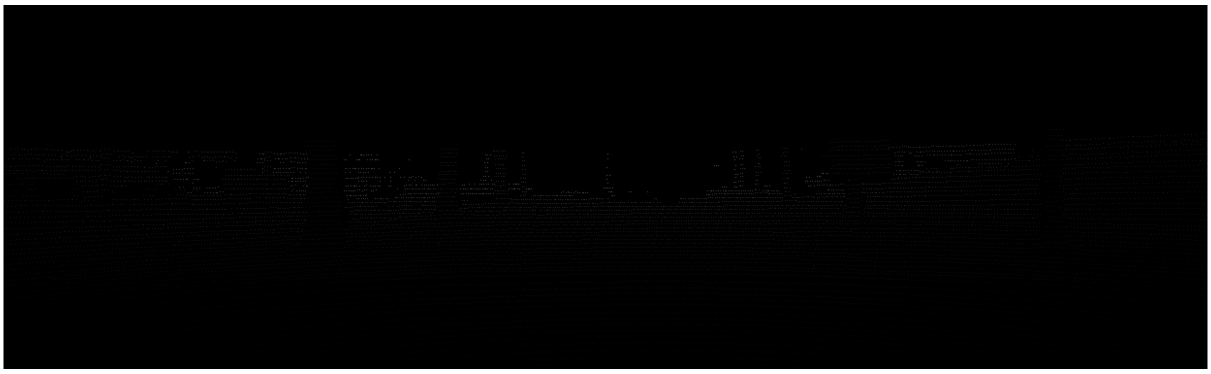
Figure 24. Point Cloud of the city.

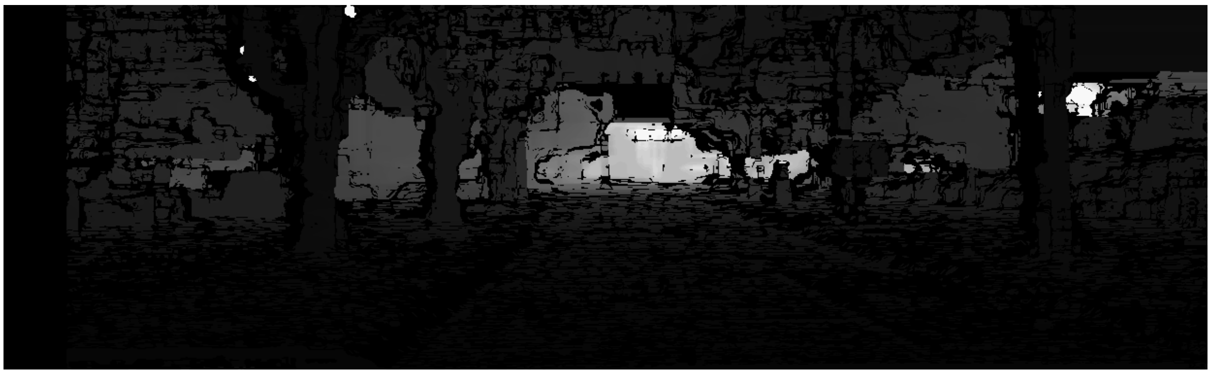
Figure 25. Fused results of synchronized depth estimates of the city

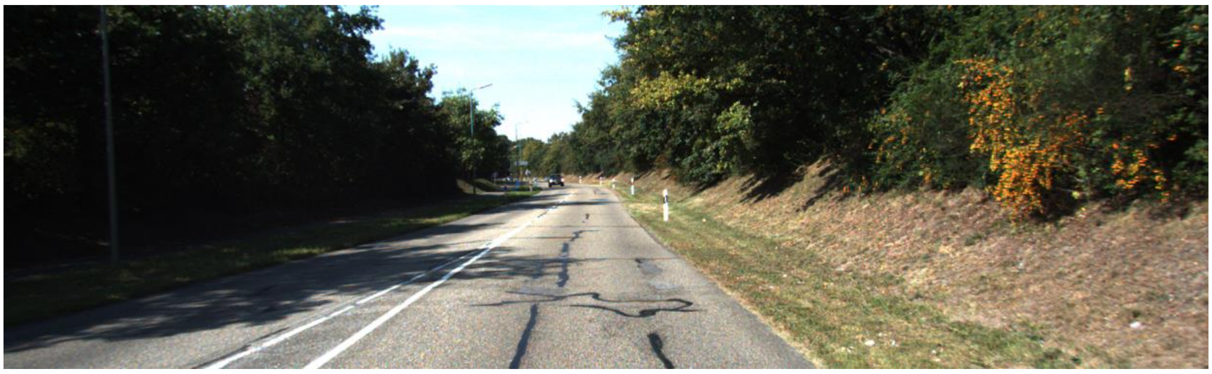

Figure 26. Input image - Road

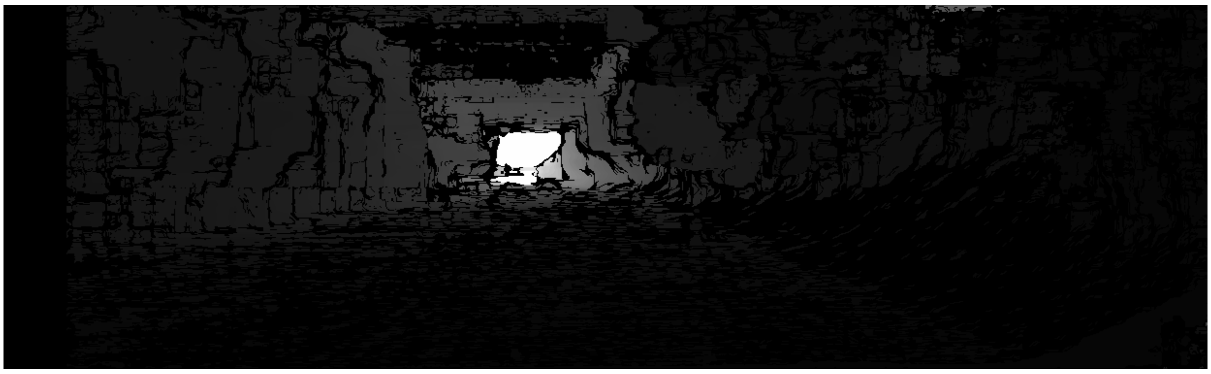

Figure 27. Depth map from stereo camera of road

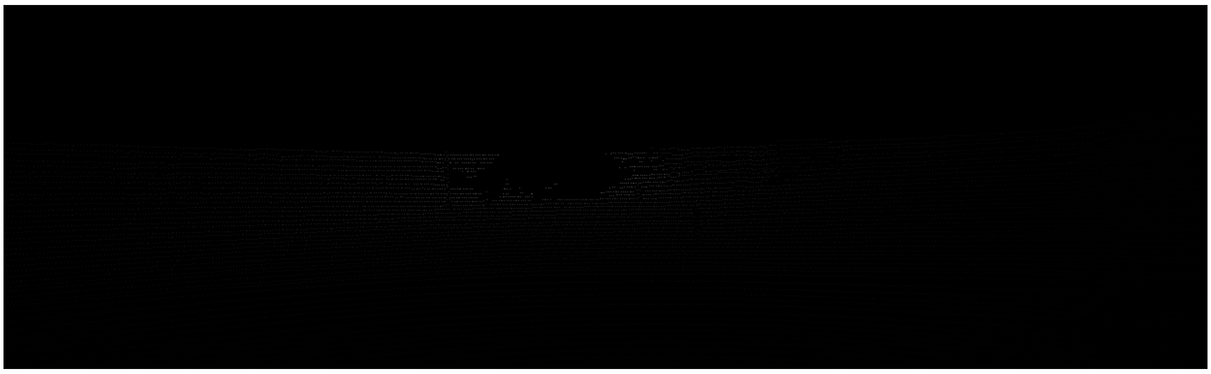

Figure 28. Point Cloud of road

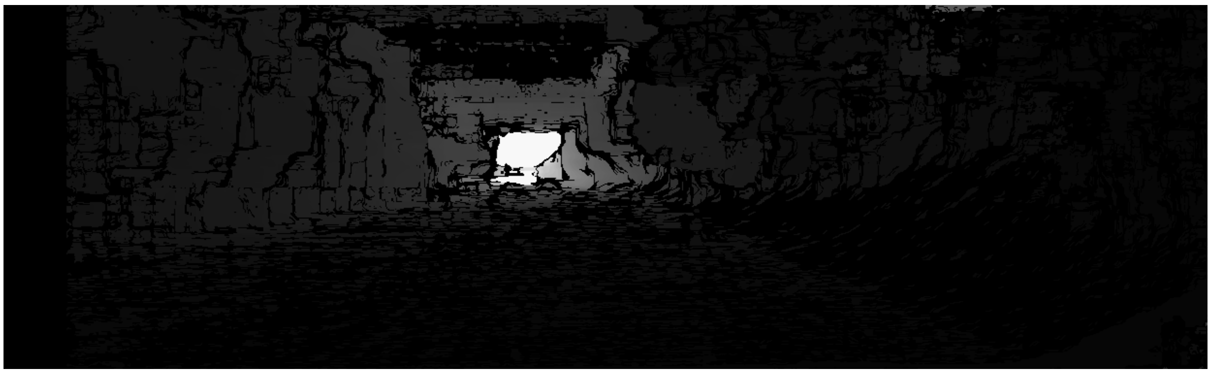

Figure 29. Fused results of synchronized depth estimates of the city

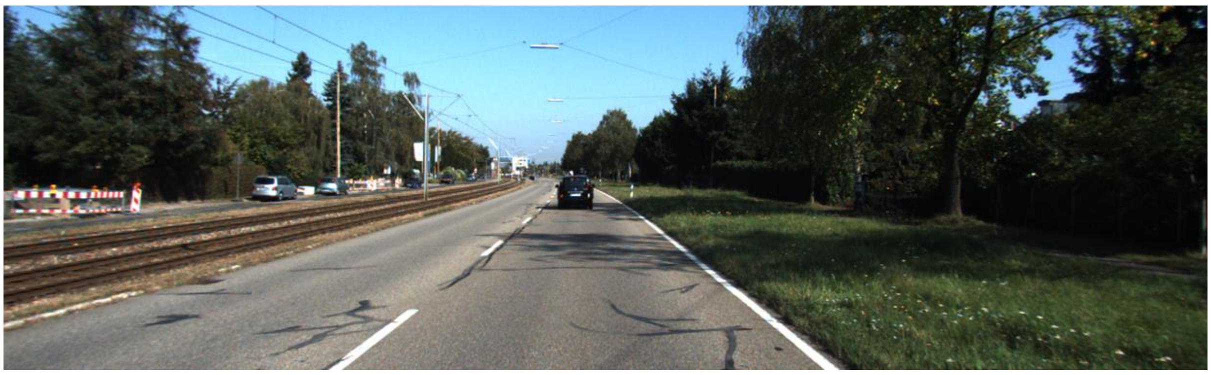

Figure 30. Input image - Residential Area

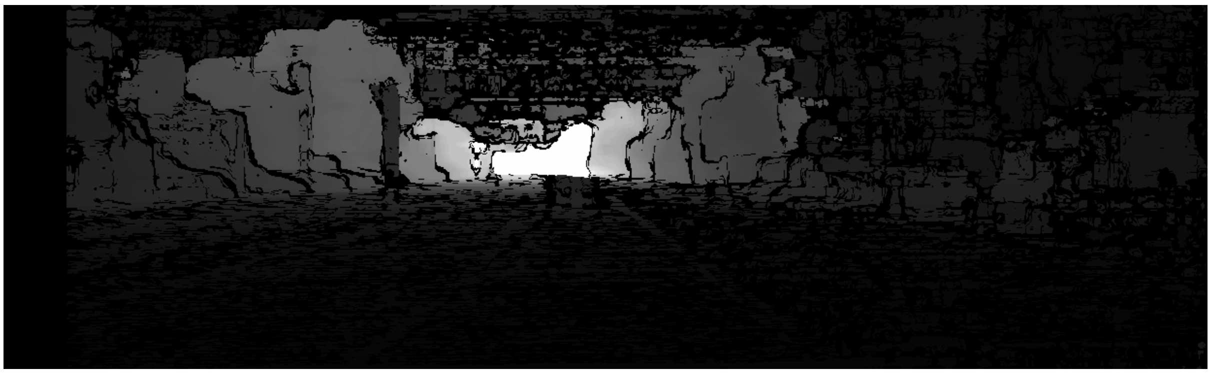

Figure 31. Depth map from stereo camera of residential area.

Figure 32. Point Cloud of residential area.

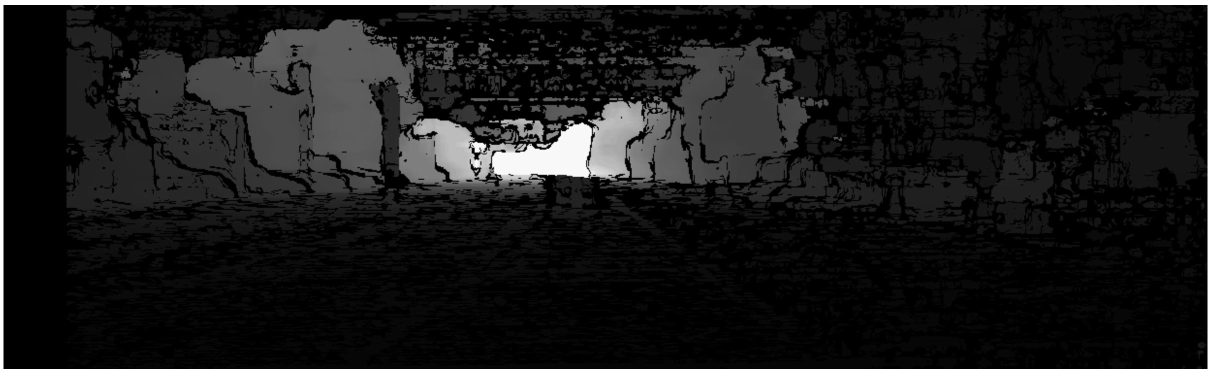

Figure 33. Fused results of synchronized depth estimates of residential area:

4.6. Quantitative results

The following section elaborates on the mean error between the evaluated fused results when compared to ground truth provided by KITTI datasets. Comparisons of three errors are computed for three different scenes as explained in subsequent chapters.

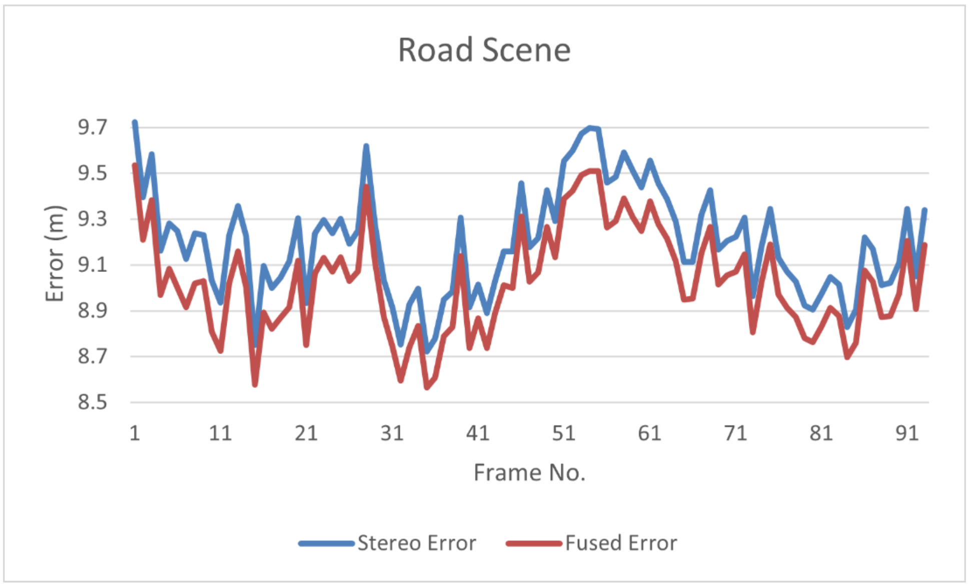

Figure 34. Comparative analysis of errors computed from fused output to the ground truth of a road scene.

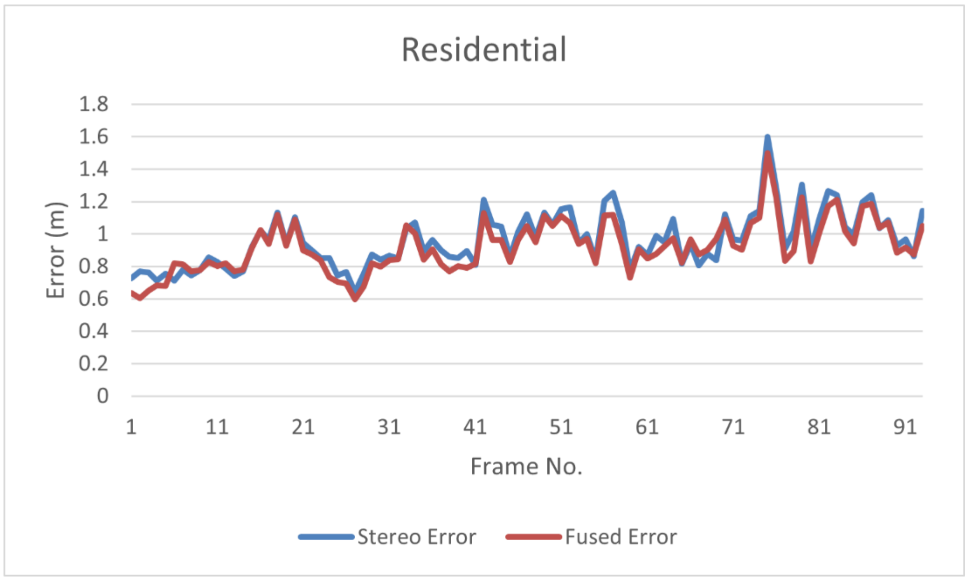

Figure 35. Comparative analysis of errors computed from fused output to the ground truth of a residential scene.

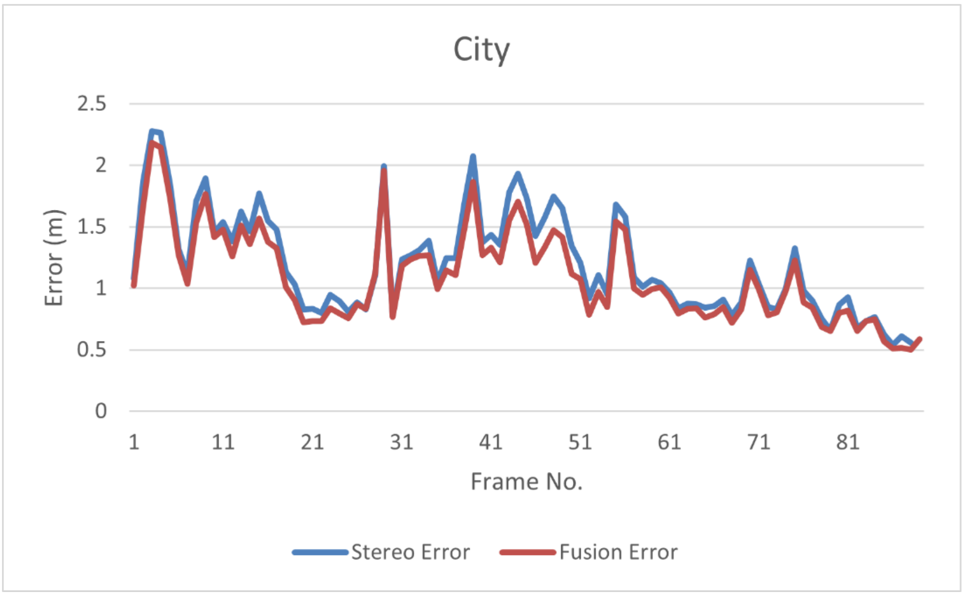

Figure 36. Comparative analysis of errors computed from fused output to the ground truth of a city scene.

TABLE 7. RESULTS DEPICTING MEAN ABSOLUTE ERROR OF FUSED DEPTH AS COMPARED TO STEREO CAMERA DEPTH MAP.

Mean Absolute Error Difference	
Road	0.169808
Residential	0.138450
City	0.124000

4.6.1. Discussion:

As observed in the above results, the Extended Kalman filter shows an improvement on all the scenes in the KITTI dataset however the most improvement was shown in the "road" scene from the KITTI data which shows that the fusion algorithm works better, where depth maps at long ranges need to be calculated as compared to scenes with short range depth maps.

As for results using the collected dataset, due to the LiDAR being used was a 2D LiDAR, so proper fusion could only be achieved in a single plane, but it showed that the platform could be used as is with a 3D LiDAR and the framework would still work.

The low value of the standard deviation shows that the fusion algorithm is correctly predicting values near the ground truth, and since 2D LiDAR was used this will be true for short ranges but when a proper 3D LiDAR is used the algorithm performance can be further improved.

In summary, this chapter provides comprehensive experimental results and analysis of the depth perception module. The algorithm showed high accuracy and robustness through testing with the KITTI dataset and real-world data. Comparisons between algorithm-generated depth maps and ground truth maps confirmed performance. Fused results exhibited effective processing of depth information in diverse environments. The findings validate the module's potential in mobile robotics and autonomous driving. This chapter establishes a foundation for further advancements in depth perception algorithms.

Chapter 5 -CONCLUSIONS

5.1. Summary of achievements:

The following are the achievements of the research:

This program is funded by IGNITE Foundation, through its National Grassroots Research Initiative (NGIRI) program. Their support highlights the recognition of its importance and potential impact.

This project also qualified for the second stage of the FICs Competition.

Undertaking a final year project on multi-modal sensor fusion for environmental depth perception using stereo cameras and LiDAR offers several significant benefits. Firstly, it allows for the exploration and understanding of innovative technologies in the field of autonomous vehicles and perception systems. Combining the results of two sensing modalities LiDAR and a stereo camera, this project delivered an elaborative analysis of the environmental depth estimation supplying robust and accurate results mitigating the limitations of individual sensors. Secondly, working on the perception module of self-driving cars facilitates the learning and enhancement of skills like sensor integration, data synchronization, and fusion algorithms which are emerging technologies in the industry. These skills are essential for the advancement of autonomous vehicles and robotics, making it an excellent benchmark for enhancing one's technical expertise. Lastly, by addressing the challenges associated with multi-modal sensor fusion, this project contributes to the broader research community's understanding of perception systems, potentially leading to advancements in autonomous driving technology and improved safety on the roads. This project allowed for the development of a platform on which more sensing modalities can be added which would lead to further investigation in the domain of Autonomous Vehicle navigation in the department of Mechatronics Engineering. Overall, undertaking this final year project offers invaluable knowledge, skill development, and a chance to make a meaningful contribution to the field of autonomous vehicles and multi-modal sensor fusion.

5.2. Future recommendations:

- Obtain Real-word dataset using a 3D LiDAR to ensure proper fusion.
- Improve the computation cost and time needed for the Extended Kalman filter by utilizing GPU and multi-threading.
- Try out another sensor fusion algorithm and evaluate their efficacy against the current sensor fusion algorithm.
- Add other sensing modalities to the framework to research new sensor fusion techniques.

REFERENCES

[1] M. Y. Kim, “A Sensor Fusion System with Thermal Infrared Camera and LiDAR for Autonomous Vehicles: Its Calibration and Application,” pp. 361–365, 2021.

[2] J. Z. Sasiadek and P. Hartana, “Sensor Data Fusion Using Kalman Filter,” no. August 2000, 2015, Doi: 10.1109/IFIC.2000.859866.

[3] E. Molino-minero-re, A. A. Aguileta, and R. F. Brena, “Improved Accuracy in Predicting the Best Sensor Fusion Architecture for Multiple Domains,” pp. 1–17, 2021.

[4] D. Feng *et al.*, “Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2021, Doi: 10.1109/TITS.2020.2972974.

[5] N. Kemsaram, A. Das, and G. Dubbelman, “A Stereo Perception Framework for Autonomous Vehicles,” 2020.

[6] H. Badino, D. Huber, and T. Kanade, “Integrating LIDAR into stereo for fast and improved disparity computation,” *Proceedings - 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, 3DIMPVT 2011*, pp. 405–412, 2011, Doi: 10.1109/3DIMPVT.2011.58.

[7] L. C. Point, “LiDAR-Camera Calibration using 3D-3D Point correspondences,” pp. 1–19.

[8] X. Zhang *et al.*, “OpenMPD: An Open Multimodal Perception Dataset for Autonomous Driving,” *IEEE Trans Veh Technol*, vol. 71, no. 3, pp. 2437–2447, 2022, Doi: 10.1109/TVT.2022.3143173.

[9] “Vision-Based Multi-Sensor Fusion for Robust Unmanned Aerial Vehicles Autonomous Navigation,” 2019.

[10] V. John, “Sensor Fusion and Registration of Lidar and Stereo Camera without Calibration Objects Sensor Fusion of Lidar and Stereo Camera without Calibration Objects,” no. February 2017, Doi: 10.1587/trans. E0.

[11] A. Pfeuffer and K. Dietmayer, “Optimal Sensor Data Fusion Architecture for Object Detection in Adverse Weather Conditions”.

[12] H. Raei, Y. Cho, and K. Park, “Autonomous landing on moving targets using LiDAR, Camera and IMU sensor Fusion,” no. Ascc, pp. 419–423, 2022.

[13] D. Das, N. Adhikary, and S. Chaudhury, “Sensor fusion in autonomous vehicles using LiDAR and camera Sensor with Odometry,” vol. 1, pp. 4–9.

[14] M. Zhang, D. Tang, C. Liu, X. Xu, and Z. Tan, “A LiDAR and camera fusion-based approach to mapping and navigation,” *Chinese Control Conference, CCC*, vol. 2021-July, pp. 4163–4168, 2021, Doi: 10.23919/CCC52363.2021.9549993.

[15] Sasiadek J, Hartana P, Sensor data fusion using Kalman filter, *“Proceedings of the 3rd International Conference on Information Fusion, FUSION 2000,* Doi: 10.1109/IFIC.2000.859866

[16] Vineet Gandhi, Radu Horaud, “High-Resolution Depth Maps Based on TOF-Stereo Fusion,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol 2012, doi:10.1109/ICRA.2012.6224771.

[17] A. Zhu, “Multi Vehicle Stereo Event Camera Dataset,” daniilidis-group.github.io. https://daniilidis-group.github.io/mvsec/.(accessed Nov. 24, 2020).

[18] E. Kirsten, L. C. Inocencio, M. R. Veronez, L. G. Da Silveira, F. Bordin, and F. P. Marson, “3D Data Acquisition Using Stereo Camera,” IEEE Xplore, Jul. 01, 2018. https://ieeexplore.ieee.org/document/8519568 (accessed May 28, 2023).

[19] A. N. Catapang and M. Ramos, “Obstacle detection using a 2D LIDAR system for an Autonomous Vehicle,” *IEEE Xplore*, Nov. 01, 2016. https://ieeexplore.ieee.org/abstract/document/7893614 (accessed Nov. 24, 2020).

[20] F. Itami and T. Yamazaki, “A Simple Calibration Procedure for a 2D LiDAR With Respect to a Camera*,” IEEE Sensors Journal*, vol. 19, no. 17, pp. 7553–7564, Sep. 2019, doi: https://doi.org/10.1109/jsen.2019.2915991. (accessed May 26, 2023).

[21] S. Schneider, M. Himmelsbach, T. Luettel, and H.-J. Wuensche, “Fusing vision and LIDAR - Synchronization, correction and occlusion reasoning,” *IEEE Xplore*, Jun. 01, 2010. https://ieeexplore.ieee.org/document/5548079 (accessed May 28, 2023). (accessed May 26, 2023).

[22] S. Li, L. Wang, J. Li, B. Tian, L. Chen, and G. Li, “3D LiDAR/IMU Calibration

Based on Continuous-Time Trajectory Estimation in Structured Environments," *IEEE Access*, vol. 9, pp. 138803–138816, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3114618. (accessed May 26, 2023).

[23] G. Terejanu, "Extended Kalman Filter Tutorial." [Online]. Available: https://homes.cs.washington.edu/~todorov/courses/cseP590/readings/tutorialEKF.pdf (accessed: May 26, 2023.)

[24] "*ZED 2 ZED 2 Camera and SDK Overview.*" Available: https://cdn2.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf (accessed: May 26, 2023.)

[25] Gabriel, "*ZED 1.0 is here, adds Positional Tracking and 3D Reconstruction*," *Stereolabs*, Jun. 28, 2016. https://www.stereolabs.com/blog/positional-tracking-3d-reconstruction-and-more-with-zed-camera/ (accessed May 26, 2023).

[26] "ZED Depth Sensors," *encyclopedia.pub*. https://encyclopedia.pub/entry/20326 (accessed May 26, 2023).

[27] *"YDLIDAR|Focus on lidar sensor solutions,"* www.ydlidar.com. https://www.ydlidar.com/Public/upload/files/2019-12-18/YDLIDAR%20TX20%20Datasheet.pdf (accessed May 26, 2023).

[28] *"YDLIDAR|Focus on lidar sensor solutions,"* www.ydlidar.com. https://www.ydlidar.com/Public/upload/files/2019-12-18/YDLIDAR%20TX20%20User%20Manual.pdf (accessed May 26, 2023).

[29] N. O. and A. A. US Department of Commerce, *"What is LIDAR,"* oceanservice.noaa.gov, Feb. 26, 2021. https://oceanservice.noaa.gov/facts/lidar.html#:~:text=Lidar%2C%20which%20stands%20for%20Light (accessed May 26,2023)

[30] *"Jetson Nano,"* NVIDIA Developer, Mar. 06, 2019. https://developer.nvidia.com/embedded/jetson-nano. (accessed May 28, 2023).

[31] NVIDIA, *"Jetson Nano Developer Kit,"* NVIDIA Developer, Mar. 06, 2019. https://developer.nvidia.com/embedded/jetson-nano-developer-kit. (accessed May 28, 2023).

[32] *“Getting Started with Jetson Nano Developer Kit,”* NVIDIA Developer, Mar. 05, 2019. https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit. (accessed May 28, 2023).

[33] *“OpenMPD: An Open Multimodal Perception Dataset for Autonomous Driving”* by X. Zhang *et al, IEEE Trans Veh Technol*, vol. 71, no. 3, pp. 2437–2447, 2022.

[34] J. Dong, D. Zhuang, Y. Huang, and J. Fu, *“Advances in Multi-Sensor Data Fusion: Algorithms and Applications,”* Sensors, vol. 9, no. 10, pp. 7771–7784, Sep. 2009, doi: https://doi.org/10.3390/s91007771. (accessed May 28, 2023).

[35] *“Kalman and Extended Kalman Filters: Concept, Derivation and Properties,”* docplayer.net. https://docplayer.net/21170209-Kalman-and-extended-kalman-filters-concept-derivation-and-properties.html. (accessed May 28, 2023).

[36] “Lidar 101: An Introduction to Lidar Technology, Data, and Applications Coastal Remote Sensing Program,” 2012. Available: https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf. (accessed May 28, 2023).

[37] J. Weng, P. Cohen, and M. Herniou, “Camera calibration with distortion models and accuracy evaluation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992, doi: https://doi.org/10.1109/34.159901. (accessed May 28, 2023).

[38] D. V. Papadimitriou and T. J. Dennis, “Epipolar line estimation and rectification for stereo image pairs,” *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 672–676, Apr. 1996, doi: https://doi.org/10.1109/83.491345. (accessed May 28, 2023).

[39] S. K. Gehrig, F. Eberli, and T. Meyer, “A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching,*” Lecture Notes in Computer Science*, pp. 134–143, 2009, doi: https://doi.org/10.1007/978-3-642-04667-4_14. (accessed May 28, 2023).

[40] Y.-C. Chen, Y.-C. Wu, C.-H. Liu, W.-C. Sun, and Y.-C. Chen, “Depth map generation based on depth from focus,” *IEEE Xplore*, Apr. 01, 2010. https://ieeexplore.ieee.org/document/5503103 (accessed May 28, 2023).

[41] S. Yang and M. Baum, “Extended Kalman filter for extended object tracking,” 2017

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar. 2017, doi: https://doi.org/10.1109/icassp.2017.7952985. (accessed May 28, 2023).

[42] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Aug. 2013, doi: https://doi.org/10.1177/0278364913491297. (accessed May 28, 2023).

[43] G. Gerig, "Image Rectification (Stereo)." Available: http://www.sci.utah.edu/~gerig/CS6320-S2013/Materials/CS6320-CV-F2012-Rectification.pdf (accessed May 28, 2023).