# AN ANALYSIS OF CAMERA CONFIGURATIONS AND DEPTH ESTIMATION ALGORITHMS FOR TRIPLE-CAMERA COMPUTER VISION SYSTEMS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Jared Peter-Contesse

December 2021

COMMITTEE MEMBERSHIP

|  |  |
|---|---|
| TITLE: | An Analysis of Camera Configurations and Depth Estimation Algorithms for Triple-Camera Computer Vision Systems |
| AUTHOR: | Jared Peter-Contesse |
| DATE SUBMITTED: | December 2021 |
| COMMITTEE CHAIR: | Andrew Danowitz, Ph.D.<br>Associate Professor of Electrical Engineering |
| COMMITTEE MEMBER: | Jane Zhang, Ph.D.<br>Professor of Electrical Engineering |
| COMMITTEE MEMBER: | Lynne Slivovsky, Ph.D.<br>Computer Engineering Department Chair |

## ABSTRACT

An Analysis of Camera Configurations and Depth Estimation Algorithms for Triple-Camera Computer Vision Systems

Jared Peter-Contesse

The ability to accurately map and localize relevant objects surrounding a vehicle is an important task for autonomous vehicle systems. Currently, many of the environmental mapping approaches rely on the expensive LiDAR sensor. Researchers have been attempting to transition to cheaper sensors like the camera, but so far, the mapping accuracy of single-camera and dual-camera systems has not matched the accuracy of LiDAR systems. This thesis examines depth estimation algorithms and camera configurations of a triple-camera system to determine if sensor data from an additional perspective will improve the accuracy of camera-based systems. Using a synthetic dataset, the performance of a selection of stereo depth estimation algorithms is compared to the performance of two triple-camera depth estimation algorithms: disparity fusion and cost fusion. The cost fusion algorithm in both a multi-baseline and multi-axis triple-camera configuration outperformed the environmental mapping accuracy of non-CNN algorithms in a two-camera configuration.

## ACKNOWLEDGMENTS

Thanks to:

- My parents, Alex and Kathy, for providing unwavering support throughout all my years of schooling
- The Cal Poly Robotics Club, for providing an environment for learning and building friendships
- Andrew Danowitz, for his expertise and advice throughout my research and drafting of my thesis
- Kayla, for her help in editing drafts

TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# Chapter 1

# INTRODUCTION

Autonomous vehicle technology has been a major research topic for many years. Attempts were made as far back as 1984 to create a vehicle that would be able to drive autonomously, with Carnegie-Mellon's Navlab that could drive very slowly along a winding road. The limitation of the vehicle at that time was the computational power of the hardware running the image processing and control algorithms [5]. With ever increasing advances in sensor technology, computer hardware performance, and more recently, machine learning, the push to develop autonomous driving tech has ramped up.

There are four major tasks that autonomous vehicle systems need to complete. These include environment analysis, vehicle localization, path planning, and vehicle control [6]. This thesis focuses on the environment analysis component of autonomous vehicles. An understanding of the vehicle's surrounding is necessary for a vehicle to travel along roads, follow traffic laws, and avoid collisions with pedestrians, cyclists, and other vehicles. Environmental analysis is commonly achieved by creating an accurate map of the surroundings and locating important objects and features in the map [6]. Approaches to mapping vary significantly depending on the type of sensor used to capture data of the environment. Common sensors include radar, ultrasonic distance, LiDAR, and various types of cameras (RGB, Infrared, light field, etc) [6]. Some approaches rely on a single type of sensor while a majority use multiple different types in tandem to improve accuracy and increase reliability [7]. The process of combining data from different sources to produce an output with more accurate or useful information is known as sensor fusion [8].

This thesis explores one approach to sensor fusion called stereo vision and its applications for autonomous vehicles. Stereo vision is the process of gathering 3D information of a scene from 2D images captured from different perspectives [3]. In the simplest case, depth can be perceived from two images captured by cameras displaced horizontally, in a process similar to the human vision system. However there is no limit on the number or relative positioning of cameras used to capture an environment. Multi-camera systems show promise in improving computer vision applications over two-camera systems by providing additional data to the computer vision algorithms [9].

## 1.1 Motivation

The inspiration and basis for this thesis came in part from work by a research group at Cornell University working on what they called Pseudo-LiDAR for 3D object detection [10]. This team saw that the leading 3D object detection methods were dominated by those that used data captured by a LiDAR sensor. LiDAR sensors are expensive, especially in comparison to sensors like the camera, which limit their ability to be used in consumer autonomous vehicles [11]. Consequently, researchers have attempted to develop image-based algorithms for autonomous vehicle tasks. However, the accuracy of image-based 3D object detection is currently significantly lower than LiDAR methods. The Psuedo-LiDAR team's proposal was to convert data captured by non-LiDAR sensors into "LiDAR form," which could be fed into the best LiDAR 3D Object Detection algorithms. They found that this approach achieved a significant 3D Object Detection accuracy improvement over the existing image-based algorithms, but was still less accurate than the cutting-edge LiDAR algorithms.

## 1.2 Objectives

This thesis examines an approach for improving the accuracy of image-based depth estimation. An increase in the accuracy of depth estimation should improve the accuracy of 3D Object Detection and Localization when the enhanced depth estimator is included in the Pseudo-LiDAR pipeline. Since a majority of the image-based depth estimation research has been focused on monocular (single image) or binocular (two image) systems, this thesis will examine the feasibility of using a trinocular (three image) system to improve depth estimation. Adding an additional camera to the camera system may be able to increase the accuracy of image-based depth estimation enough to eliminate the need for LiDAR.

This thesis makes the following contributions:

1. Demonstration of a platform for creating virtual datasets of specific computer vision autonomous vehicle tasks.
2. Quantitative and qualitative analysis of the performance of various depth estimation algorithms when using the virtual dataset
3. A performance comparison of various depth estimation algorithms
4. A performance comparison between two-image and three-image depth estimation systems

Chapter 2

# RELATED WORK

The inspiration for this thesis came from the Pseudo-LiDAR research and extension of their work, Pseudo-LiDAR++, conducted by a Cornell University team.

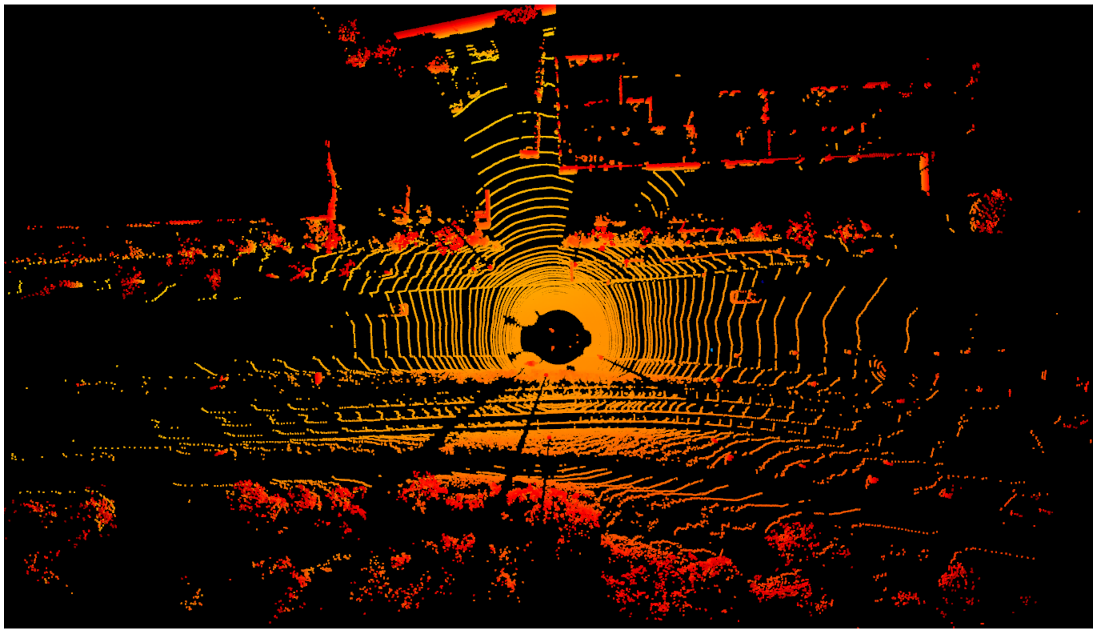


**Figure 2.1: Point cloud representation of LiDAR data from the KITTI dataset [1]**

## 2.1 Relevant Concepts

This section briefly describes the concepts and terminology necessary to understand the results of their research. These topics are LiDAR, the KITTI dataset and benchmarks, and the 3D Object Detection Benchmark.

### 2.1.1 LiDAR

Light Detection and Ranging, commonly known as LiDAR, is a sensor that has seen significant interest for 3D mapping due to its ability to produce highly precise measurements. The LiDAR sensor emits pulses of light and times how long it takes for light beams to bounce off objects and return back to the sensor. Knowledge of how fast light travels through the air can be used to determine distance using the time-of-flight principle [12]. A LiDAR sensor with 360-degree rotating lasers is commonly mounted on the top of autonomous vehicles to provide mapping of the vehicle's full surroundings. These sensors produce a highly accurate 3D point cloud at frequencies of 5-20Hz. The main issue with LiDAR is that sensors with a high resolution are extremely expensive [11]. LiDAR is the most commonly used sensor for autonomous vehicles but LiDAR hardware is the most expensive component of these systems [10]. An important thing to note about LiDAR is that the sensors produce data in the form of a 3D point cloud. An example of a point cloud is shown in Figure 2.1.

### 2.1.2 The KITTI Dataset

The KITTI dataset is one of the most popular datasets used for autonomous driving research. The KITTI group recorded 6 hours of real traffic scenarios using a variety of sensors mounted on a vehicle including LiDAR, color and grayscale stereo cameras, IMU measurements, and high precision GPS. The KITTI dataset is also available under a Creative Commons License to download and use [1]. To promote research in vision recognition systems, the group also created a series of benchmarks for specific computer vision tasks such as depth mapping, scene flow evaluation, object detection, and multi-object tracking [13]. The benchmarks were created by selecting a subset of the larger KITTI dataset and generating ground truth data (true measurements) for each of the tasks. Evaluation metrics measuring the performance of algorithms in comparison to ground truth are used to rank and compare algorithms on the KITTI leaderboard [14]. The Pseudo-LiDAR project uses a KITTI benchmark to evaluate the performance of their algorithm.

### 2.1.3 KITTI 3D Object Detection Benchmark

The vision recognition task that the pseudo-LiDAR team attempted was 3D Object Detection. For this task, the goal is to generate 3D bounding boxes around all vehicles, bicycles, and pedestrians in the view of the front facing cameras. The KITTI 3D Object Detection benchmark contains raw images, point clouds, and ground-truth data labeled with the type and bounding box location of relevant objects [13]. The benchmark evaluates performance by calculating two metrics: Average Precision (AP) and bounding box overlap. Average Precision evaluates how well the algorithm can detect and classify objects in the image, while bounding box overlap evaluates how close an objects predicted bounding box is to its ground truth location [15].

## 2.2 Pseudo-LiDAR

The research in the original Pseudo-Lidar paper began by examining existing 3D object detection algorithms on the KITTI leaderboard. The team noticed that the leading approaches were dominated by LiDAR with the best algorithms achieving Average Precision percentages on the benchmarks of about 73%. The best image-only algorithms achieved a mere 10% AP [10]. The Pseudo-LiDAR team notes that image-based depth estimation is inherently less accurate than LiDAR, but that inaccuracy should not account for such a large difference in 3D Object Detection performance between the two approaches.

The Pseudo-LiDAR team found that the cause for the performance gap between image-based approaches and LiDAR was that image-based approaches represented depth data in a way that makes it difficult for machine learning algorithms to do 3D Object detection. The LiDAR signal is represented as 3D point-clouds. In this data representation, apparent object sizes are invariant to depth. Using a camera, object size does vary based on depth and objects appear smaller in the resulting image (captured by less pixels) the further away they are from the sensor. 2D convolutional networks applied over the image/depth map can have a difficult time classifying objects that can be of varying sizes. The team converted the output of image-based depth estimation algorithms into point-cloud representations, which they call Pseudo-LiDAR, and found that they could significantly increase the accuracy of 3D object detection by leveraging existing LiDAR-based 3D object detection pipelines. Their custom pipeline was able to achieve an Average Precision of 45.3% on the KITTI 3D Object Detection benchmark, which was an improvement of 350% over existing image-based approaches [10].

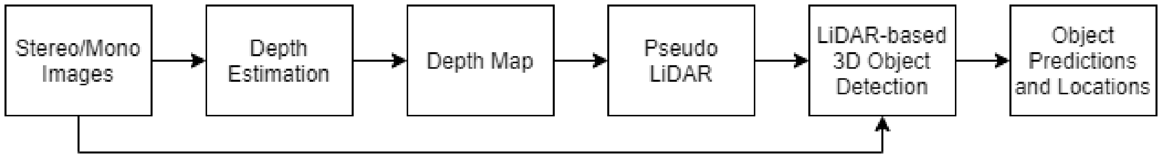

Stereo/Mono Images
Depth Estimation
Depth Map
Pseudo LiDAR
LiDAR-based 3D Object Detection
Object Predictions and Locations

**Figure 2.2: The Pseudo-LiDAR 3D object detection pipeline**

Although the performance improvement was significant, this approach is still behind the state-of-the-art LiDAR algorithms [16, 17]. The team noted two reasons for the discrepancy: low resolution images and the lower accuracy of image-based depth estimation, especially at far distances.

## 2.3 Psuedo-LiDAR++

Pseudo-LiDAR++ was an extension of Pseudo-LiDAR to determine why the performance in 3D Object Detection and Localization of Pseudo-LiDAR was lower than when using LiDAR point clouds. The team found that the higher 3D object detection error stemmed from the inherent higher error in the image-based depth estimation [18]. Two approaches were attempted in order to improve the accuracy of image-based depth estimation. In the first approach, the team modified an existing convolutional neural network depth estimation algorithm to make the algorithm more sensitive to areas in the image that were further away during training. This improved the accuracy of the Pseudo-LiDAR 3D object detection pipeline by about 10% over the original Pseudo-LiDAR implementation [18]. The second approach tested to see if a sparse LiDAR (4 beams rather than 64-128 beams) could be used to de-bias the image-based depth estimators. Sparse LiDAR sensors are significantly cheaper and are just as accurate as dense sensors but produce a lower resolution point-cloud. The team developed a sensor fusion algorithm that could combine the sparse but accurate LiDAR point-cloud with the dense but inaccurate, image-based point-cloud. This sensor fusion approach was able to achieve a 37% performance improvement over the original Pseudo-LiDAR algorithm. At distances less than 30m, Pseudo-LiDAR performed comparably to state-of-the-art LiDAR. However, the accuracy of Pseudo-LiDAR began to fall off at distances greater than 30 meters [18].

# Chapter 3

# BACKGROUND

This chapter provides background information in order to better understand the algorithms discussed later.

## 3.1 Camera Modeling

The camera is an essential component for computer vision systems. A mathematical model is necessary to understand how this sensor interacts with the world. The most common and most simplistic way of modeling a digital camera is the pinhole model. This model describes how points in the 3D world are mapped onto a 2D plane through the projection of light through a “pinhole” onto an image plane.

### 3.1.1 Intrinsic Parameters

The formal pinhole model can be constructed from the geometry of Figure 3.1.

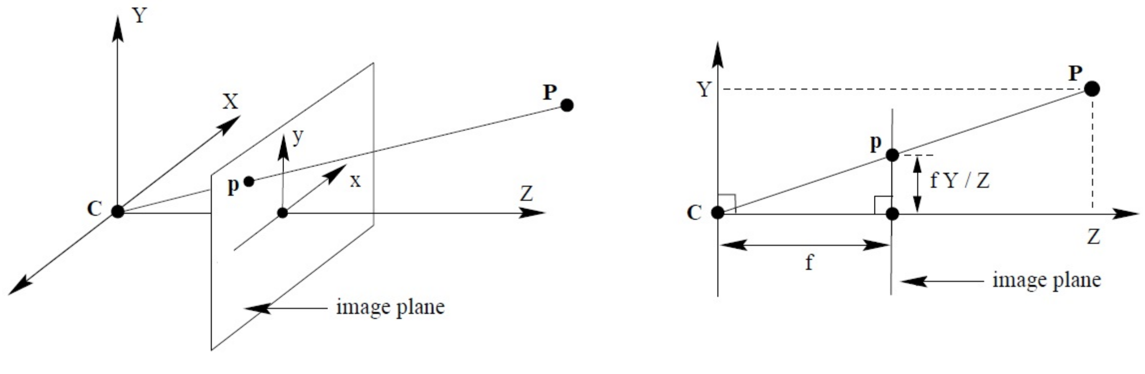


**Figure 3.1: Geometry of the pinhole model [2]**

The derived equations for the projection of a 3D point $P = (X, Y, Z)$ onto a 2D image plane $p = (x, y)$ are shown below in Equation 3.1.

$$x = f\frac{X}{Z} \quad \text{and} \quad y = f\frac{Y}{Z} \tag{3.1}$$

In order to make solving the system of equations easier, the projection equations are commonly converted into matrix form. The transformation matrix from 3D space to a 2D image plane is shown below in Equation 3.2.

$$\begin{bmatrix} x' \\ y' \\ z \end{bmatrix} = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

In this particular camera model, some additional camera parameters are included to account for a non-ideal camera [2]. $f_x$ and $f_y$ represent the focal length ratios in the vertical and horizontal axis, $c_x$ and $c_y$ are the offsets of the principal point $p$ from the camera center in the vertical and horizontal axis, $s$ represents the skew distortion of non-rectangular pixels in a camera, and the euclidean coordinates $p = (x, y)$ can be found using $x = \frac{x'}{z}$ and $y = \frac{y'}{z}$. This matrix can be decomposed even further into Equation 3.3.

$$\begin{bmatrix} x' \\ y' \\ z \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K \begin{bmatrix} I_{3x3} & 0_{3x1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.3}$$

The matrix K, commonly known as the camera calibration matrix, contains the intrinsic parameters of the camera. This matrix transforms a set of 3D coordinates referenced from the center of the camera to the 2D image plane. These parameters can be determined by referencing the camera's datasheet or, more commonly, by a method called camera calibration which is discussed in Section 3.2.

### 3.1.2 Extrinsic Parameters

So far, the mapping of 3D coordinates to the image plane has been performed from the perspective of the camera. If information about the 3D world is given in another coordinate system, then an additional transformation is required to convert the external reference systems points to the camera reference system. This mapping consists of two transformations: rotation and translation.

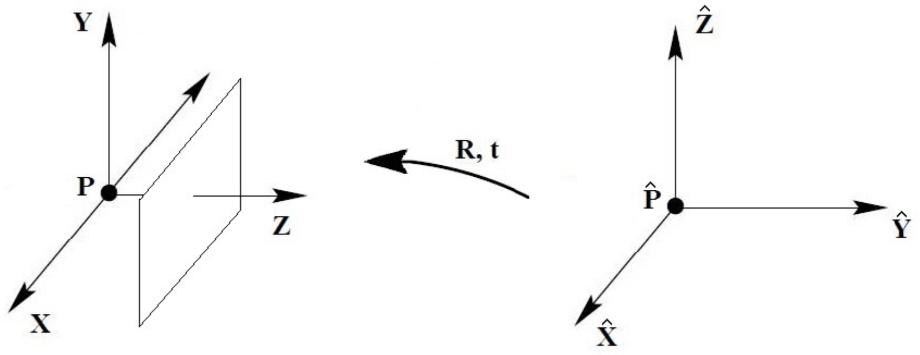


**Figure 3.2: Transformation between the world and camera coordinate frames**

The transformation describing the mapping from the external reference system $\hat{P} = (\hat{X}, \hat{Y}, \hat{Z})$ to the camera reference system $P = (X, Y, Z)$ can be represented using nine parameters for rotation and three parameters for translation in the following matrix form [19]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3x3} & t_{3x1} \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{bmatrix} \tag{3.4}$$

Combing the intrinsic and extrinsic mappings produces the following equation that maps points in the external reference system $\hat{P} = (\hat{X}, \hat{Y}, \hat{Z})$ to points on the cameras image frame $p = (x, y)$.

$$\begin{bmatrix} x' \\ y' \\ z \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{bmatrix} = K \begin{bmatrix} R_{3x3} & t_{3x1} \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{bmatrix} \tag{3.5}$$

### 3.1.3 Lens Distortion

The pinhole model is an idealized approximation which ignores the fact that most cameras have lenses. Lenses can cause the mapping between the camera reference systems and image plane to become distorted. The two major types of lens distortions are radial distortion and tangential distortion [20]. *Radial distortion* causes straight lines to appear curved. The traditional model for representing radial distortion is a non-linear equation where $r$ is the distance from the point to the distortion center [3].

$$\begin{aligned} x_{distorted} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{distorted} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{3.6}$$

With the coefficients $k_1, k_2$, and $k_3$ describing the radial distortion.

Similarly, *tangential distortion* causes objects in some regions of the image to look nearer than expected. This is caused by the lens not being perfectly parallel to the image plane. A similar model to radial distortion is commonly used [3].

$$\begin{aligned} x_{distorted} &= x + [2p_1xy + p_2(r^2 + 2x^2)] \\ y_{distorted} &= y + [p_1(r^2 + 2y^2) + 2p_2xy] \end{aligned} \tag{3.7}$$

With the coefficients $p_1$ and $p_2$ describing the tangential distortion.

## 3.2 Camera Calibration

There are five intrinsic camera parameters, six extrinsic camera parameters, and five lens distortion parameters that all describe the mapping from the 3D world to the 2D image plane. It is difficult to analytically determine these parameters, so the common method is to measure them in a process called camera calibration. Since we have models describing the system, all we need are a set of inputs and outputs that could be used to solve for these internal parameters. The common technique is to take pictures of a known object at various angles. The pictures and knowledge about the 3D real world points of the object allow us to work backwards to determine the camera parameters [3]. Computer Vision API's, like MATLAB's Camera Calibration Toolbox [21] and OpenCV's Camera Calibration functions [22], use a chessboard as the pattern for calibration which was a technique proposed by Zhengyou Zhang in his original camera calibration paper [23]. Knowledge about the size of the squares on a chessboard and how they appear in the series of images is used to solve for all the camera parameters [22].

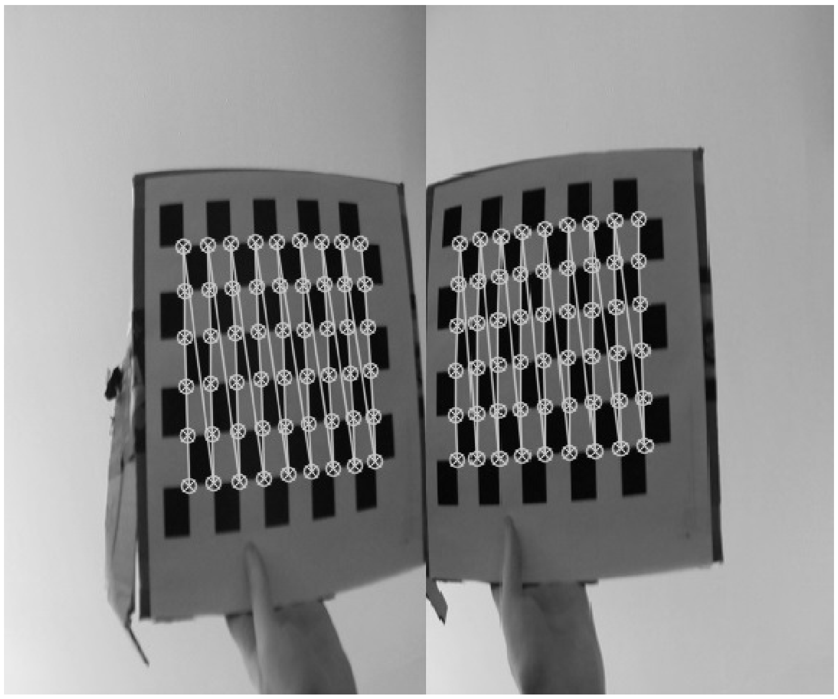


**Figure 3.3: Examples of grayscale chessboard calibration images**

At this stage, exact models describing how a point in the 3D world will be mapped onto the 2D image plane of a camera have been determined. This knowledge is essential for stereo rectification which is discussed in Section 3.3.2.

## 3.3 Stereo Vision

A camera's process of translating an objects 3D coordinates onto a 2D plane is a well defined model, however, determining 3D coordinates from a single 2D image is an impossible task because depth information is lost during projection. In Figure 3.4, point $p$ in the left image can correspond to any 3D point $P$ along the line extending from the camera center $(C)$ through point $p$. To determine the true location of that point on the line, a different perspective is required. The process of extracting 3D depth information from a scene using multiple 2D images is known as *Stereo Vision* [24].

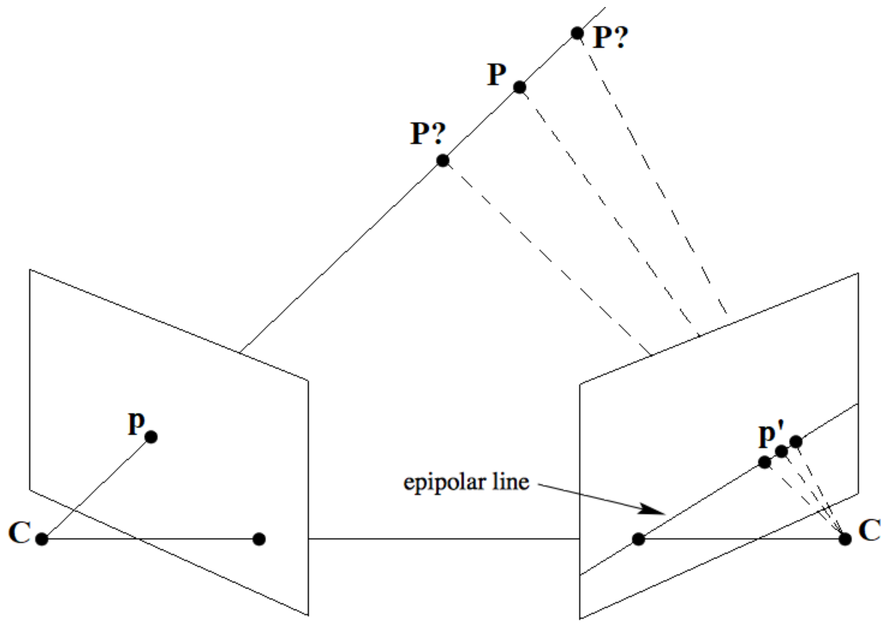


**Figure 3.4: Epipolar geometry of a two image system**

### 3.3.1 Epipolar Geometry

To find the depth of a 3D point from the perspective of one of the images, a number of geometric relations must be determined. This geometry is known as *Epipolar Geometry* [25]. A diagram showing the relationship between two arbitrary image planes and a point $P$ in 3D space is shown in Figure 3.4. An important thing to note from this diagram is that line extending from the left cameras center ($C$) through point $p$ projects a line onto the right image plane. This line is known as the *epipolar line* and every point in the left image plane has a corresponding epipolar line in the right image plane. If the geometric relationship between the two image systems is known, and we were searching for the point corresponding to point $p$ in the right image plane, we only need to search along the epipolar line in the right image plane rather than the entire image. If the points corresponding to the 3D point are known in both the left and right image planes, then the exact 3D location of that point can be determined through triangulation.

### 3.3.2 Stereo Rectification

If the locations of the image planes are arbitrary, then the geometries to determine the epipolar lines and perform triangulation will not be consistent for all camera systems. To simplify the geometries for stereo vision, an algorithm known as *Stereo Rectification* projects each image onto a common image plane.

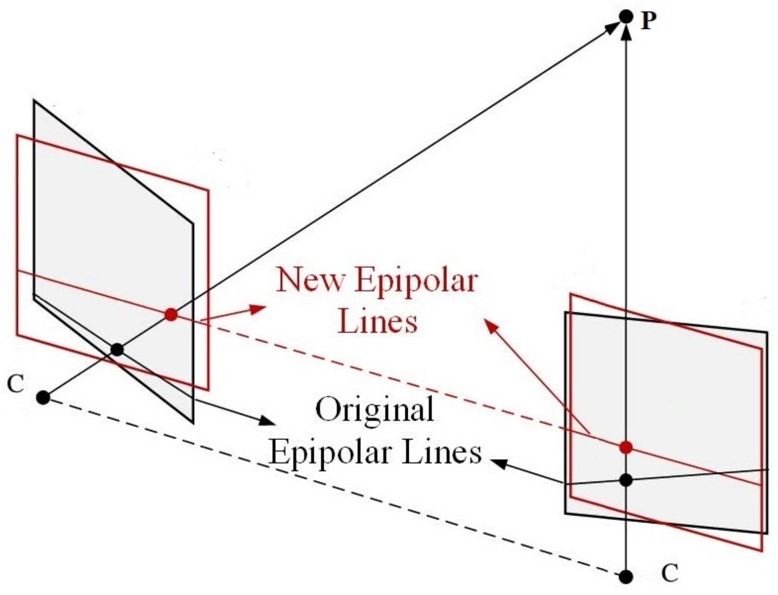


**Figure 3.5: Stereo rectification geometry [3]**

This significantly simplifies the geometry of stereo vision because this causes the epipolar lines to be parallel to the x-axis of the image plane. For digital images, the epipolar line for a point in one image will be the corresponding row in the other image. This will make the stereo depth estimation algorithms, discussed in a later section, faster and more reliable.

The first step in the stereo rectification process is to remove the lens distortions. This is done by mapping all the distorted pixels to their corresponding undistorted locations according to Equations 3.6 and 3.7 discussed previously.

The next step is to project the undistorted images onto a common plane. The equations used by MATLAB's Camera Calibration Toolbox [21] and OpenCV's Camera Calibration functions [22] to map points onto a common plane are Equations 3.8 and 3.9 [19].

$$p'_l = \frac{f}{z'} R_{rect} p_l \tag{3.8}$$

$$p'_r = \frac{f}{z'} R R_{rect} p_r \tag{3.9}$$

$p_l$ is a point in the left image plane with its corresponding point $p'_l$ on the common plane. The rotation matrix $R_{rect}$, built from the extrinsic translation parameters between the two camera reference frames determined during camera calibration, and a scaling factor $\frac{f}{z'}$, where $f$ is the focal length and $z'$ is the z-coordinate of the common plane, perform the translation of points from the left image plane to the common plane.

For the points in the right image plane $p'_r$, an additional transformation has to be performed to align the right image plane with the left image plane before performing the mapping to the new common plane. This matrix $R$ is the rotation matrix found during camera calibration.

After rectification, the resulting images are usually oddly shaped and contain regions with unknown pixel values near the edges. The last stage of the stereo rectification algorithm is to crop these irregularities out and find a rectangular region that covers as much of both of the images as possible. Figure 3.6 shows a summary diagram of the rectification process.

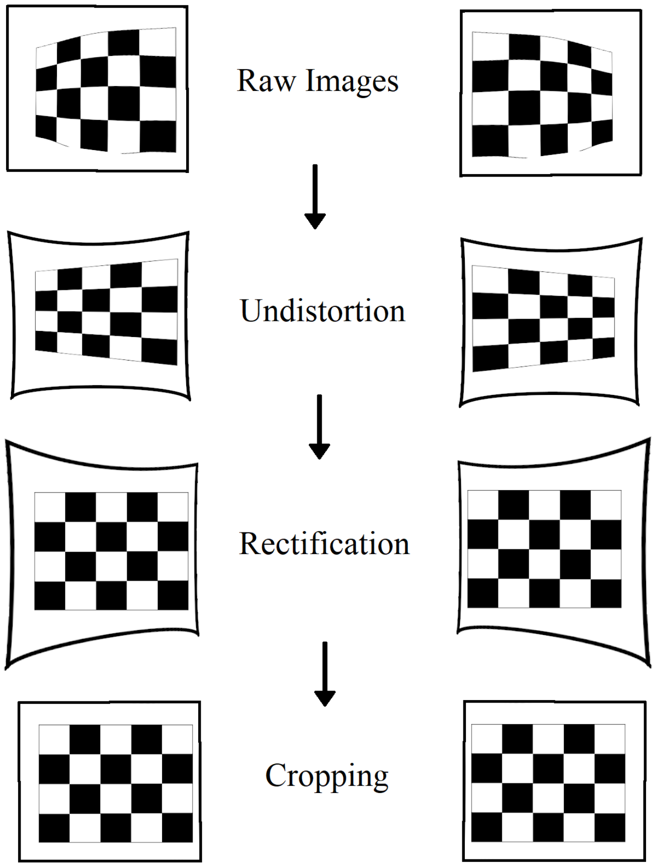


**Figure 3.6: Main steps of stereo rectification**

### 3.3.3 Stereo Geometry

The process of rectification aligns the two image planes and makes the geometry of stereo vision significantly simpler as shown in Figure 3.7.

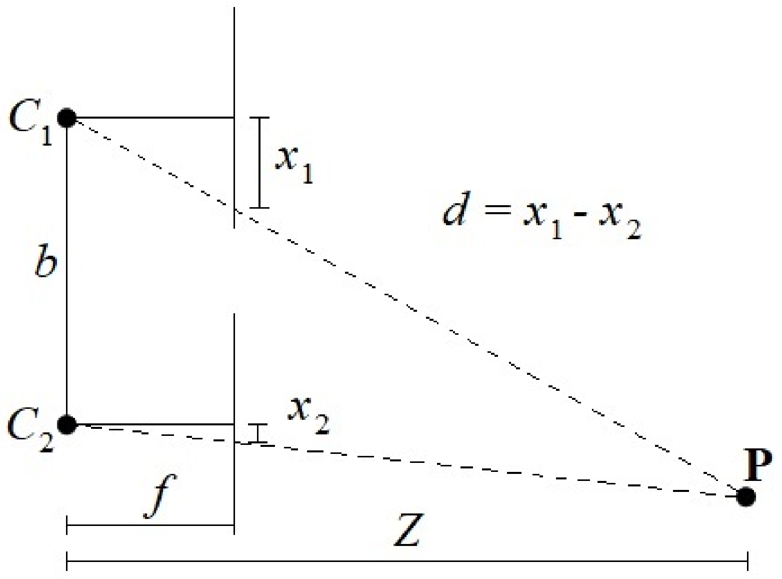


**Figure 3.7: Stereo camera geometry of a rectified stereo camera set**

If point $P = (X, Y, X)$, defined in the reference system of the left camera, is projected onto the image coordinates $(x_1, y_1)$ in the left image and $(x_2, y_2)$ in the right image, relative to the center of each camera image plane, then the exact location of $P$ can be determined in 3D space. $Z$ can be found using a simple ratio of similar triangles. In this figure, $b$ is the distance between the two camera centers which is known as the *baseline*, $f$ is the focal length of the cameras, and the difference $d = x_1 - x_2$ is known as *disparity*, which represents the relative offset in the location of the projection on the two image planes. Note that the units of disparity is pixels.

$$\frac{Z}{b} = \frac{f}{d} \tag{3.10}$$

$$Z = \frac{bf}{d} \tag{3.11}$$

Equation 3.11 is the essential equation for stereo vision because it defines how to determine depth from disparity. The inverse relationship between depth and disparity causes points with projections that have a large disparity to be closer, while points with a smaller disparity are further away. The depth $Z$, along with ratios of similar triangles can be used to determine the other 3D coordinates of $P$ as well.

$$X = x_1 \frac{Z}{f} \tag{3.12}$$

$$Y = y_1 \frac{Z}{f} \tag{3.13}$$

The goal of Stereo Vision is to determine the 3D location of points using a set of images. The geometric derivation demonstrated that this goal can be achieved by finding the correct projection of that point in each camera and then the disparity can be used to determine the points exact location. That point is measured from the perspective of one of the cameras in the camera stereo system, which is known as the *reference camera.* All measurements are calculated from the perspective of the reference camera. The next section discusses how to determine a matching point in two offset images.

### 3.3.4 Stereo Depth Estimation

Stereo depth estimation is performed by stereo matching algorithms. These algorithms take rectified images as input and perform pixel matching and calculation of disparities. The output of these algorithms is a 2D array of disparities, called a disparity map, where each array element contains an estimated disparity for a pixel in the reference image to the image pair.

Stereo depth estimation algorithms are characterized into either local, global, or semi-global methods [3]. Local stereo algorithms select a small section of the reference image and attempt to match it with sections of the other images. This approach is based on the assumption that pixels in the matching neighborhood will be similar regardless of the perspective [26]. How similar the matching neighborhoods are to each other is determined by a cost function. The location on the search space with the minimum cost is determined to be the matched pixel.

Global methods seek to minimize a energy function that is defined over all pixels in image. Given a set of images and a disparity map, the energy function returns a single value defining how well a disparity map matches pixels between the images. Since all pixels are accounted for in this function, this method is less sensitive to noise than local methods [27]. Global methods provide better results than local methods but come at a significantly higher computational cost because it is very difficult to minimize a function with so many possible parameters ($Width \times Height$ parameters).

Semi-global methods attempt to find the middle-ground between algorithm performance and computational complexity. These methods attempt to minimize a global energy function, similar to global methods, but have techniques to avoid solving the minimization of the energy function across the entire image. Semi-global methods are the most popular method for stereo matching. Most semi-global methods are based off of the original semi-global algorithm, Semi-Global Matching (SGM) [28].

Recently, machine-learning based methods have begun to explore using convolutional neural networks (CNN) to perform stereo matching. Architectures like the Pyramid

Stereo Matching Network (PSMNet) [29] and Group-wise Correlation Stereo Network (GWCNet) [30] are used in the top performing image-based algorithms on the KITTI leaderboards.

In summary, stereo vision is the process of determining 3D information from a set of images using stereo calibration, rectification, and matching. Figure 3.8 shows the high level diagram of a Stereo Vision systems.

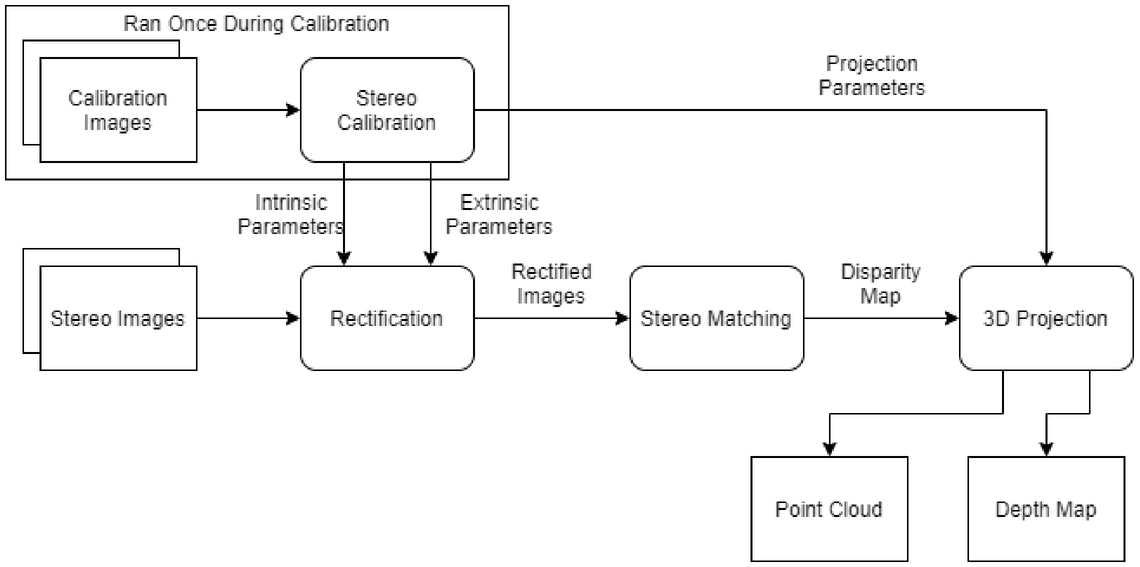


**Figure 3.8: Flowchart summarizing the main steps of Stereo Vision**

Disparity maps are often converted into depth maps through Equation 3.11, where each element in the depth map holds depth information in a unit like meters. The data in this form is more useful for applications like autonomous vehicles. Since this conversion is computationally simple, disparity map and depth map are terms that are used interchangeably to describe the output of stereo estimation algorithms.

# Chapter 4

# METHODOLOGY

The motivation for this thesis came from the Pseudo-LiDAR research group who found that image-based depth estimation could be used as a replacement for LiDAR sensor data for 3D Object Detection and Localization algorithms. The issue was that the image-based 3D Object Detection and Localization (Pseudo-LiDAR) had worse performance than state-of-the-art LiDAR 3D Object Detection and Localization, notably due to the lower accuracy of image-based depth estimation, especially at long distances. Pseudo-LiDAR++ improved the accuracy of image-based depth estimation by fusing sparse LiDAR sensor data with the image-based depth estimation data. This thesis aims to examine if an additional camera is able to improve the accuracy of image-based depth estimators over traditional two-camera stereo vision methods.

## 4.1 Camera Configurations

Many three-camera configurations are possible. To determine which configurations has the best chance of improving depth estimation algorithms, further analysis is necessary. The first step in this process is examining existing stereo camera systems.

### 4.1.1 Analysis of the accuracy of image-based depth estimators

Understanding the parameters that effect the accuracy of stereo depth estimators are crucial to determining how to improve the accuracy of these systems. Looking at the camera model, only a discrete number of points can be mapped onto the cameras image plane, because the camera sensor has a limited resolution. Pixels can only capture a single intensity over a certain area. Figure 4.1 shows multiple "cameras" looking at a point in space.

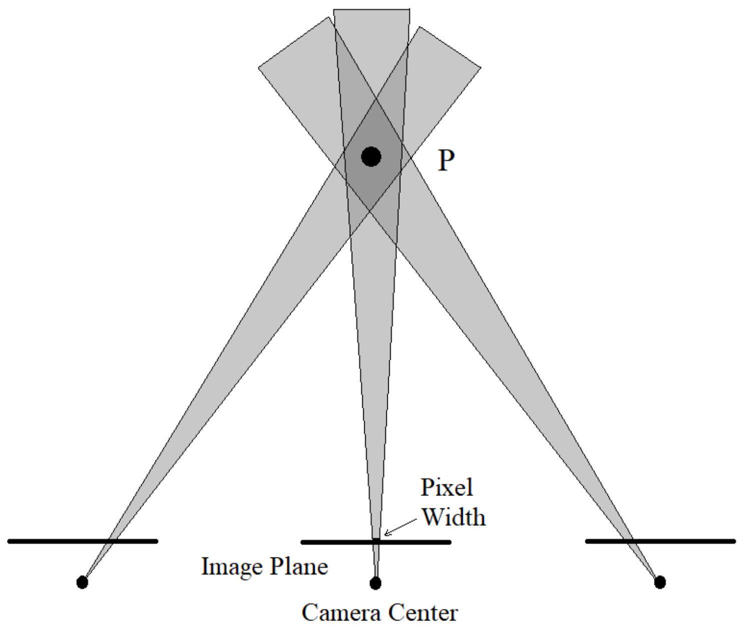


**Figure 4.1: Field of view of pixels from multiple camera perspectives looking at a point *P* in space**

The cone coming from the optical center of the cameras demonstrates how the area that a single pixel captures increases as the distance from the image plane increases. The area that the cones overlap illustrate the possible true location of the point $P$ that was detected by each camera in a stereo camera system. Figure 4.1 is significantly exaggerated, but demonstrates that distance estimation by a camera is inherently limited by the physical limitations of the camera sensor. This is why the accuracy of stereo camera depth estimation may never reach the accuracy of LiDAR systems.

The precision of a stereo system is called *depth resolution*, and refers to the theoretical limit for how accurately a stereo vision system can estimate depth given the specifications of the stereo camera system [31]. Equation 4.1 defines the relationship between the resolution of the camera and the depth error $\Delta_Z$.

$$\Delta_Z = \frac{Z^2}{fb}\Delta_{px} \tag{4.1}$$

Depth error refers to the accuracy with which a stereo vision system can estimate changes in the depth of a surface and is a way to measure depth resolution. In Equation 4.1, $\Delta_{px}$ is the length of a pixel in the horizontal direction, $Z$ is the depth of an object measured in the cameras coordinate system, $f$ is the focal length of the camera, and $b$ is the baseline of the stereo camera system. From this equation it can be determined that depth error $\Delta_Z$ has an inverse relation to the baseline. As the baseline of a stereo system increases (the distance between the centers of the two cameras) the depth error decreases, or the accuracy of the stereo system increases. Other parameters, such as focal length and camera resolution, have a relationship to depth error as well, but are much more difficult to control in the stereo camera system because they are parameters within the cameras themselves.

The problem with just increasing the baseline in an attempt to reduce depth error is that large baselines can cause more occlusions and discontinuities (Parts of objects that are not visible from all perspectives) which reduces the accuracy of the stereo matching algorithms [19]. In addition, research with real stereo vision systems have shown that, in general, shorter baselines perform better when objects are at closer distances, whereas longer baselines perform better when objects are at greater distances [32]. The idea is that adding an additional camera with a larger baseline to the stereo camera set would improve the accuracy of the depth estimation, particularly at large distances, while maintaining the accuracy of the short baseline.

### 4.1.2 Mixed-Axis Camera Placement

Studies indicate that the orientation of the stereo cameras matters when detecting certain features in images. A horizontal camera pair is superior than a vertical camera pair at detecting vertical structures in images, while the opposite is true for horizontal structures [33]. Since autonomous vehicles are driving on roads containing many different types of objects, it may be beneficial to have a stereo pair configured in the vertical direction that could better detect horizontal structures. A depth estimation algorithm that can exploit both the vertical baseline stereo setup's accuracy of horizontal structures with the horizontal baseline stereo setup's accuracy of vertical structures may be able to significantly improve depth estimation accuracy.

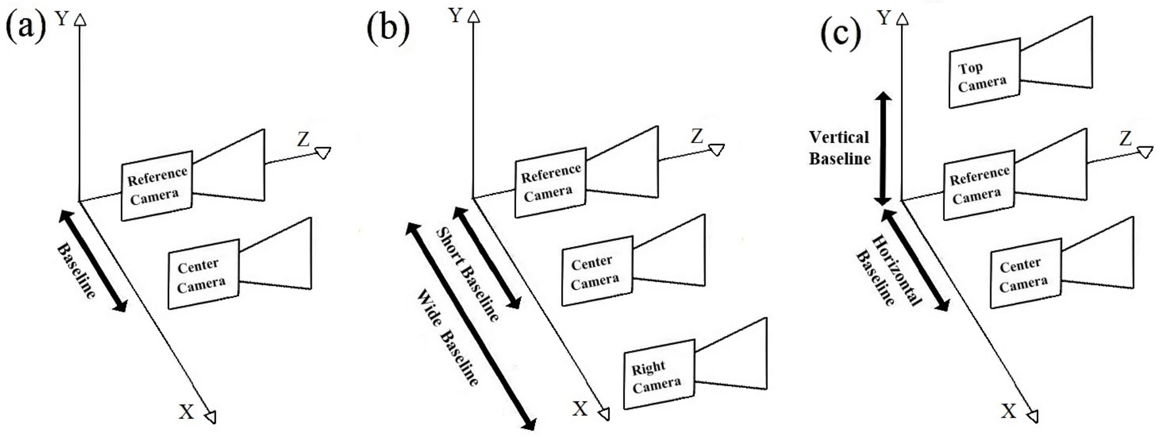


**Figure 4.2: Diagram of camera configurations used. (a) Traditional stereo, (b) Multi-baseline stereo, and (c) Multi-axis stereo**

### 4.1.3 Proposed Camera Configurations

Rather than placing the three cameras in an arbitrary fashion, there may be some benefit to configuring cameras at multiple baselines, to benefit from the combination of short and wide baseline stereo, or along perpendicular axis, to benefit from the improved detection of structures in multiple directions. The multi-baseline setup is aligned on a single axis, while the mixed-axis setup have camera pairs aligned perpendicular to each other. These camera configurations simplify the geometry which makes the stereo matching algorithms easier to implement. The camera configurations shown in Figure 4.2 are the ones used in this thesis.

## 4.2 Data Collection

The Pseudo-LiDAR and Pseudo-LiDAR++ research team used the images, LiDAR point-clouds, and ground truth data from the KITTI dataset and benchmarks to test their proposed algorithms. Since the team used algorithms that just required stereo images and some sparse LiDAR data, KITTI provides the ideal platform to perform their research. However, the raw KITTI data was collected with a single fixed stereo camera rig meaning that multi-camera depth estimation research cannot be performed using this dataset. This leaves three options: capturing data with a custom test rig, finding an existing dataset other than KITTI, or using a simulation to create the necessary data for the research.

### 4.2.1 CARLA

Because of time constraints, complexity, cost associated with capturing data, and lack of other viable datasets, this project uses simulation. After some experimentation with different simulation environments, the simulator CARLA (Car Learning to Act) was selected [4]. CARLA is an open-source simulator, developed on the Unreal Engine, designed specifically for training, prototyping, and developing autonomous driving models. CARLA was created to be used as an environment to test fully autonomous vehicle models that have path planning and vehicle control algorithms. The developers and artists working on the project created custom maps of urban and rural driving environments. These environments contain many unique buildings, vehicles, and pedestrians that try to make the simulator as realistic as possible. The maps contain various driving scenarios with working traffic signals and automatic vehicle and human AI that traverse the roads and sidewalks during simulation. An example of one of the streets in a CARLA map is shown in Figure 4.3.

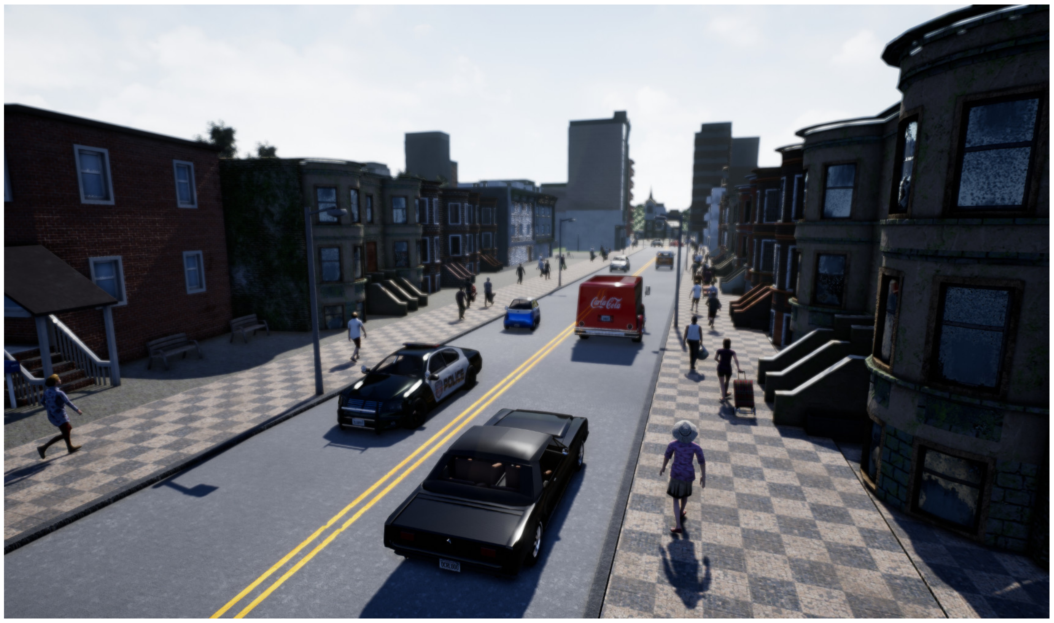


**Figure 4.3: One of the populated streets in Town 2 of the CARLA simulator [4]**

Using a Python or C++ API, objects can be created and controlled in the CARLA simulated environment. One of the most important features in the CARLA API is the ability to create and attach different sensors to vehicles in the environment. All of the sensors commonly used in autonomous vehicle systems like LiDAR, radar, and different types of cameras have virtual counterparts in the CARLA simulator that can gather data from the simulated environment.

#### 4.2.1.1 Pros and Cons of using a Simulator

One of the benefits of using a simulator is the ability to completely control the environment. The placement of objects, lighting, weather, and landscape can be controlled and modified relatively easily. New features and props can be easily updated and integrated.

For this thesis, having complete control over the specifications and positions of the cameras was important for gathering data. As discussed previously, camera calibration and rectification is a significant aspect of stereo depth estimation algorithms. In this simulator, all parameters for camera calibration can be retrieved and set through the simulator. Additionally the goal of stereo rectification is to align all images onto the same plane, if the cameras are positioned correctly in the simulator, the stereo rectification process does not need to be performed because the images will already be aligned. These luxuries are not available in real world conditions because of inherent inaccuracies of camera parameters and the impossibility of positioning cameras perfectly.

Another benefit is that simulators can measure ground truth data perfectly. In the case of depth estimation, CARLA has a virtual sensor that can save the current depth to every point in the environment in an image format. This data is the perfect depth map for the scene and can be used to analyze the performance of depth estimation algorithms. For datasets that use real data, like the KITTI dataset, the ground truth is typically manually generated and annotated containing inaccuracies and skewed performance statistics based on the biases of the annotator [13].

The main downside of simulators is that they are models of the true system and do not contain all of the factors involved. Although the props and maps of CARLA are relatively realistic, they are noticeably different than a real environment. The performance of the depth estimation algorithms when using images generated in CARLA may not match with the performance of these algorithms when using images of real scenes.

### 4.2.2 CARLA Custom Environment

The first step in generating the dataset used to test the depth estimation algorithms is to create a custom environment in CARLA. The goal of this environment was to contain a couple large objects that would be easy to examine when analyzing the performance of the depth estimation algorithms. For this reason, two vehicles, a large column, traffic lights, the road, ground, and sky are the only objects in this scene. The two cars were placed at different distances from the stereo camera array, one at 20m and the other 35m away. A large column is also located 35m away to provide a shape in the background. The vehicles were chosen as the main objects in the scene because the specific application of depth estimation in this thesis is object localization for autonomous vehicles. An example image captured from the perspective of the reference camera is shown in Figure 4.4.

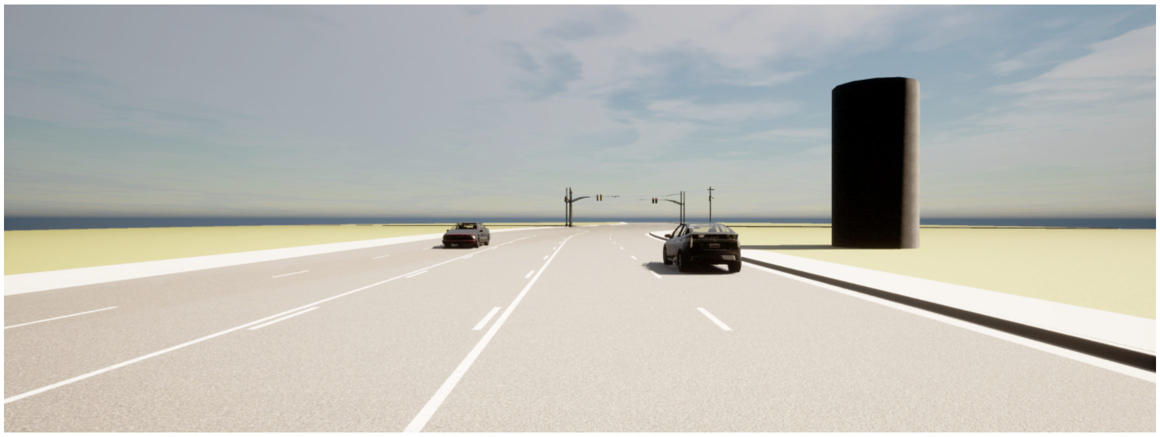


**Figure 4.4: Image of the CARLA custom environment captured from the perspective of the reference camera**

### 4.2.3 CARLA City Environment

Next one of the prebuilt CARLA maps was used to create a realistic city scene. One of the 5-way intersections in a CARLA town was selected that contained medium and large sized buildings, a large overpass, foliage in one of the street blocks, and a number of street lights and traffic signals. The prebuilt CARLA maps do not contain any vehicles or pedestrians which were added to this intersection. The environment that was created from this map contained twelve vehicles and three pedestrians. The goal of this environment was to provide a more realistic scene than the Custom environment. The images captured in this environment are more similar to those encountered on a real street.

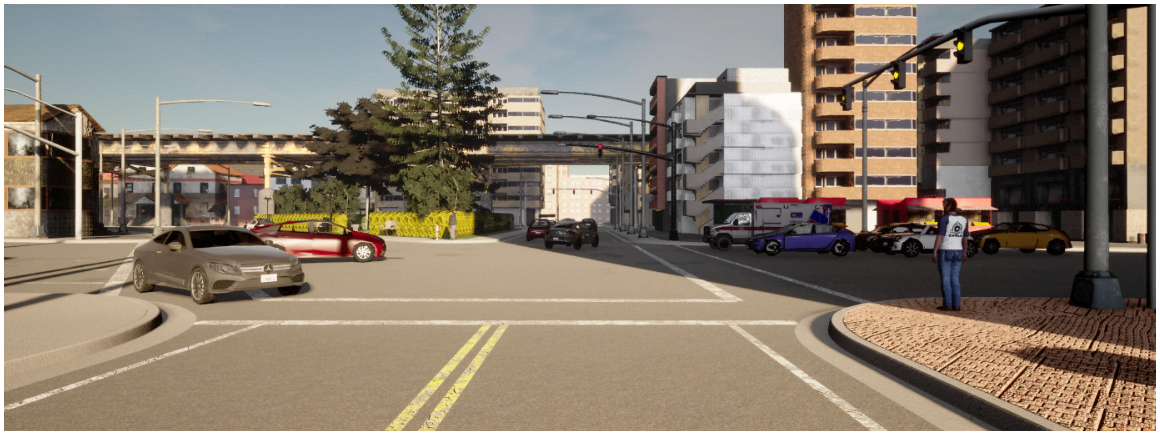


**Figure 4.5: Image of the CARLA city environment captured from the perspective of the reference camera**

In the two environments, a series of 1382x512 pixel images and ground truth data were captured using CARLA's virtual sensors in the three different camera configurations diagrammed in Figure 4.2. This specific image resolution was chosen because the images in the KITTI dataset were of this dimension. The reference camera for each of the camera configurations was located in the same exact spot, 1.6 meters above the road surface (the height of a typical vehicle) and all the images captured fall on the same image plane which is facing in the direction a typical car would be driving in. The placement of the cameras around the reference camera is based on the diagram in Figure 4.2. The traditional stereo configuration has a baseline of 0.5 meters, the multi-baseline configuration has a short baseline of 0.5 meters and a long baseline of 1 meter, and the multi-axis configuration has a horizontal baseline of 0.5m and a vertical baseline of 0.5 meters.

The images generated from these two environments were the main focus for this thesis, however to validate the results across a wider range of data, 8 additional sets of images were captured. The additional set of images was captured using the same camera configuration as the Custom and City environments, but of different scenes in the CARLA simulator.

## 4.3 Stereo Matching Algorithms

Using the synthetic images generated from the CARLA environments, the next step is to perform stereo matching. Stereo matching algorithms create disparity maps that can be used to determine depth of all the points in the reference image. To provide a reference point for the performance of triple-camera algorithms, stereo algorithms were run on the images generated from the traditional stereo rig. The algorithm that will be examined in depth is called Semi-Global Matching (SGM).

### 4.3.1 Semi-Global Matching

SGM was an algorithm proposed by Hirschmuller to solve the stereo matching problem by combining the benefits of both local and global stereo matching methods [28]. This approach combines fast computation of pixel matching and approximating a global

2D smoothing constraint by combining many 1D constraints. Similar to the global methods, SGM attempts to minimize an energy function $E(D)$ of the disparity image $D$.

$$E(D) = \sum_{p}(C(p, D_p)) + \sum_{q \in N_p} P_1 I(|D_p - D_q| = 1) + \sum_{q \in N_p} P_2 I(|D_p - D_q| > 1) \quad (4.2)$$

$p$ is a pixel in the reference image and $D_p$ is the disparity for that pixel. $q$ is a pixel in the stereo image pair and $D_q$ is the disparity for that pixel. $C(p, d)$ defines the matching cost function and $P_1$ and $P_2$ define penalties that smooth the output disparity map. If a nearby disparity is different by one, the penalty $P_1$ is added to the cost. If a nearby disparity is different by more than one, the larger penalty $P_2$ is added. The process of producing a disparity map that minimizes this energy function to produce the best possible result can be separated into four steps: matching cost computation, cost aggregation, disparity optimization, and disparity refinement [3].

#### 4.3.1.1 Matching Cost Computation

The matching cost computation is a way to determine how well a pixel in the stereo pair matches a pixel in the reference image. The cost $C(p, d)$ can be defined in many ways but overall a larger cost means that the pixels under scrutiny are less likely to be a correct match. Sum of Absolute Difference (SAD) and Sum of Squared Difference (SSD) are some of the simplest cost functions [3].

$$C_{SAD}(p, d) = |i_{ref}(p) - i_{off}(p - d)| \quad (4.3)$$

$$C_{SSD}(p, d) = (i_{ref}(p) - i_{off}(p - d))^2 \quad (4.4)$$

The cost metric used by the original SGM algorithm is the hamming distance (count of the number of different bits in two bit strings) between bit strings calculated from the census transform of each image. The census transform calculates a census string for each pixel in an image which describes the surroundings of that pixels. Given a pixel at location $i, j$ with intensity $g(i, j)$, the census string is defined

$$\begin{aligned} \mathcal{C}(i,j) = [\ \dots\ , \mathcal{H}(g(i+l_v, j+l_h) - g(i,j)),\ \dots\ ] \\ \text{with} -0.5(w_v-1) \leq l_v \leq 0.5(w_v-1) \\ \text{and} -0.5(w_h-1) \leq l_h \leq 0.5(w_h-1) \end{aligned} \tag{4.5}$$

where $w_h$ and $w_v$ are the horizontal and vertical window sizes and the heavyside function $\mathcal{H}(x)$ is given by

$$\mathcal{H}(x) = \begin{cases} 0, & \text{if } x < 0. \\ 1, & \text{otherwise.} \end{cases} \tag{4.6}$$

The matching cost between a pixel in the reference image and a pixel in the stereo pair is then computed from each of the census transformed images.

$$C(i,j,d_h) = \Delta_{hamming}(\mathcal{C}_{ref}(i,j),\ \mathcal{C}_{off}(i,j+dh)) \tag{4.7}$$

For each pixel in the reference image, a cost is calculated for each pixel along the epipolar line in the stereo image pair. The output is a 3D array where the third dimension holds all the costs calculated for that specific reference pixel.

#### 4.3.1.2 Cost Aggregation

At this point, each pixel has a whole list of possible disparities that have a cost associated with them. Finding the disparities that minimize the energy function, defined in Equation 4.2, across the entire image, with smoothing constraints $P_1$ and $P_2$, is actually a NP-complete problem [28]. To simplify this problem, the solution proposed in SGM was to perform optimizations along many 1D paths around the pixel to approximate a global minimization.

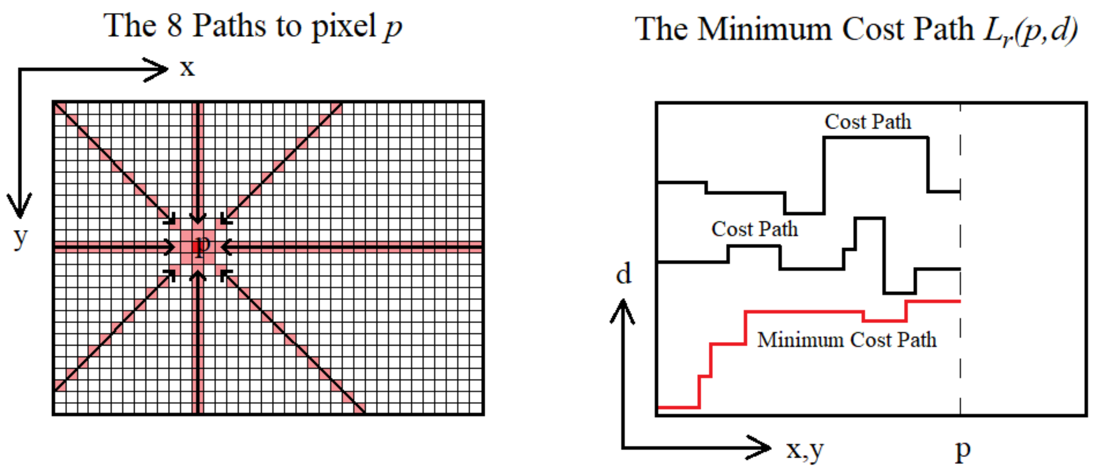


**Figure 4.6: The 8 paths used to aggregate costs up to a pixel $p$ and the minimum cost path through pixels $x$, $y$ up to p**

The aggregated (smoothed) cost $S(p, d)$ for pixel $p$ at disparity $d$ is calculated by summing the costs of all 1D minimum cost paths that end in pixel $p$ and disparity $d$.

$$S(p, d) = \sum_r L_r(p, d) \tag{4.8}$$

In equation 4.8, $L_r(p, d)$ defines the sum of the minimum cost path to pixel $p$ and disparity $d$ from direction $r$. The output at the cost aggregation stage is still a 3D array, but instead of costs, the third dimension holds the new aggregated costs.

#### 4.3.1.3 Disparity Selection

The next step is selecting the disparity with the minimum cost. This involves searching through the third dimension and selecting the index that contains the minimum value. The map of disparity values produced is the solution to minimizing the energy function from Equation 4.2 using a "semi-global" method.

#### 4.3.1.4 Disparity Refinement

The output of the disparity selection stage is a 2D array of disparity values called a disparity map. These disparity values are integers, usually ranging from 0-64, which causes the output depth map to have distinct levels of depth rather than a continuous curve. To get smoother disparity values to better fit the continuous geometry of the real world, a method known as sub-pixel approximation can be used to produce decimal disparities. One common sub-pixel approximation method is parabolic approximation that finds the minimum of a parabola fitted to the costs around the selected best disparity [34]. The local extrema of a parabolic function

$$f(x) = Ax^2 + Bx + C \tag{4.9}$$

is found by differentiating the parabolic function and setting the output to zero.

$$\begin{aligned} \frac{dy}{dx} &= 2Ax + b = 0 \\ x &= \frac{-B}{2A} \end{aligned} \tag{4.10}$$

Given a minimum cost $c_0$ at disparity $d_0$ and costs $c_+$ and $c_-$ located at $d_+ = d + 1$ and $d_- = d - 1$ respectively, where the disparities sit on the x-axis and the corresponding costs sit on the y-axis, the sub-pixel approximated disparity can be found by substituting each point into Equation 4.9. The math can be simplified by offsetting all disparities by making $d_0 = 0$. The offset can be added at the end of the sub-pixel approximation to get the actual estimated disparity.

$$\begin{aligned} c_- &= A - B + C \\ c_0 &= C \\ c_+ &= A + B + C \end{aligned} \tag{4.11}$$

Equation 4.11 shows the costs in terms of the parabolic function parameters. When the cost equations are substituted into Equation 4.10 and simplified, an equation for the sub-pixel approximated disparity given the three cost values is formed (Equation 4.12).

$$d_{sub} = \frac{c_+ - c_-}{4c_0 - 2(c_+ + c_-)} \tag{4.12}$$

Since $d_0$ was already determined to be the location of the minimum, $d_{sub}$ is also guaranteed to be the location of a minimum. This approximation finds a more refined disparity using the shape of the costs surrounding the minimum cost. This process is replicated for all minimum costs in the image. Many other disparity refinement techniques exist [35, 36], and currently disparity refinement is a active research area for computer vision.

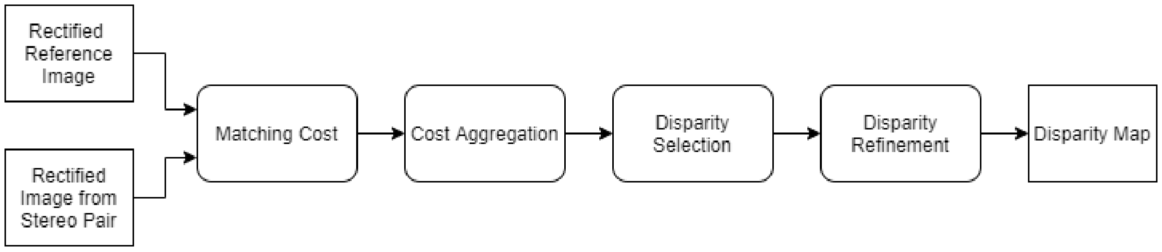


**Figure 4.7: Flowchart illustrating the main steps of the SGM algorithm**

Figure 4.7 shows the pathway that a pair of images takes to produce a disparity map using the SGM algorithm. The output of SGM is a refined 2D disparity map containing decimal disparities calculated between a reference image and its stereo pair.

## 4.4 Triple-Camera Matching Algorithms

Determining depth from a set of three images is a significantly less researched topic than two camera stereo vision. In two camera stereo, depth is determined by finding matching pixels between a reference image and a stereo image pair. In three camera systems there are now two images to match pixels to a reference image. The approaches for adding and combining this information to improve the matching output vary significantly. Some methods attempt to combine the output disparity maps by considering the system as just two stereo systems with different baselines. Other methods try to use the information gleaned from the two stereo systems in order to create a better cost map that can produces a better disparity match. The thesis examines both types of approaches which are called disparity fusion and cost fusion.

### 4.4.1 Disparity Fusion

Disparity fusion is the simplest approach to triple-camera matching algorithms. Given a reference camera and two other cameras offset by baselines $b_1$ and $b_2$, simply compute disparity maps for each extra camera and combine the disparity maps together. In the original SGM paper, this was the approach proposed for multi-camera matching [28]. The only complication is that the computed disparity maps can be in reference to different baselines, which means that the disparities have to be normalized to some common baseline. The equation offered in the original SGM paper is a weighted mean of all disparities within one pixel of the median disparity. In a triple camera system with only two calculated disparities per pixel, this just becomes a weighted mean.

$$D_{fused}(x,y) = \frac{D_1(x,y) + D_2(x,y)}{b_1 + b_2} \tag{4.13}$$

Using Equation 4.13, the resulting disparity map will have their disparities correspond to a stereo vision system that has a baseline of 1 (in whatever units the original baselines were measured in).

The benefits of this approach is that it is simple, easy to implement, and any existing stereo matching algorithm can be used to compute the disparity maps for either pair. The biggest drawback is that the computation time of this approach is more than double traditional stereo because the algorithms must be run for each stereo pair and then combined after. Algorithm computation time is an important factor to consider for any autonomous vehicle task because autonomous vehicles need to be able to react to environmental changes in real time.

### 4.4.2 Cost Fusion

Cost fusion is a triple-camera matching approach designed specifically for algorithms that use costs to determine disparities. These algorithms attempt to fuse information from the two perspectives before the computation of disparities. Since minimum cost is what determines what disparity is selected, having a more accurate cost curve will hopefully select a better disparity. As an example, Figure 4.8 shows matching cost curves (Map 1 and Map 2) calculated for a single pixel in the reference image to two different image pairs. The minimum cost is located at significantly different disparities for each of the individual cost curves (about 25px for Map 1 and and 62px for Map 2), meaning that different disparities would be selected depending on which cost curve was used. Using disparity fusion, the resulting disparity would be the mean of the individual disparities (43.5px for disparity fusion), which is far from either of the individual disparities. Using cost fusion, the matching cost curve confirms with Map 1 that the minimum cost is located at a disparity of 25 pixels. By fusing cost information, the selected minimum cost will likely be more accurate than just averaging individual disparities.

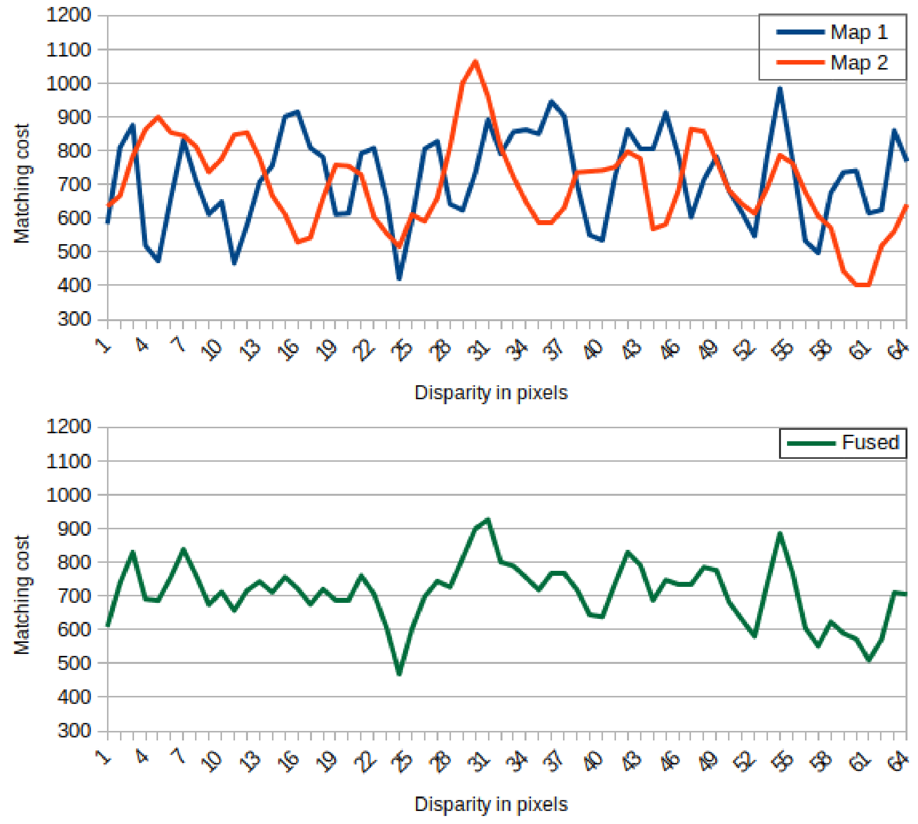
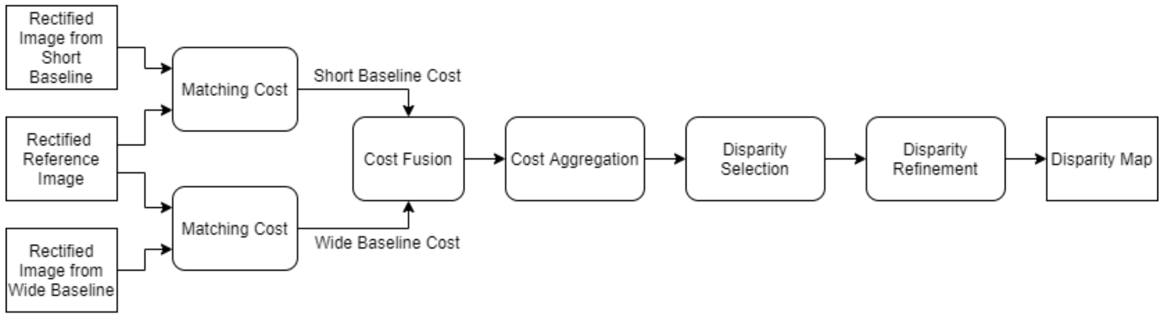
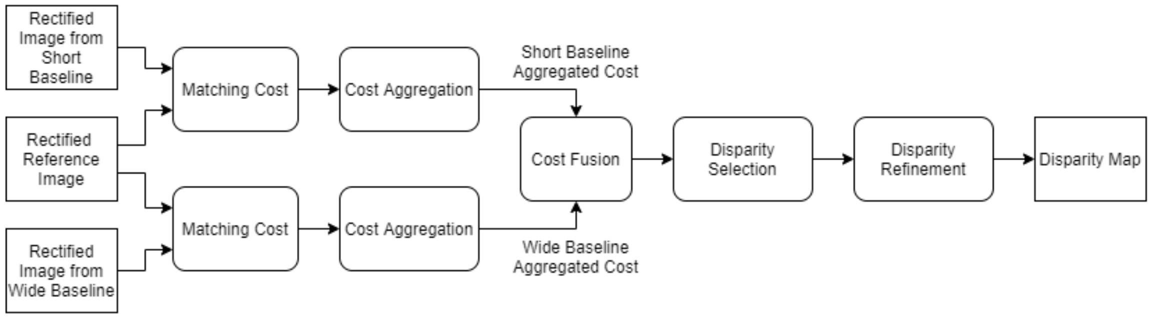


**Figure 4.8: Example of individual and fused matching cost curves of a single pixel in the cost fusion step of the SGM algorithm**

For the SGM algorithm, combining the cost curves can occur either before or after cost aggregation. Papers argue for both fusing it before [33] and fusing after [37]. The benefit for performing fusion before is that cost aggregation only needs to be performed once, which is the most computationally intensive part of the algorithm. This thesis tests both cost fusion methods.



(a) Cost fusion before aggregation



Rectified Image from Short Baseline
Rectified Reference Image
Rectified Image from Wide Baseline
Matching Cost
Matching Cost
Cost Aggregation
Cost Aggregation
Short Baseline Aggregated Cost
Wide Baseline Aggregated Cost
Cost Fusion
Disparity Selection
Disparity Refinement
Disparity Map

(b) Cost fusion after aggregation

**Figure 4.9: Flowcharts of the two multi-baseline cost fusion algorithms showcasing modifications of the SGM algorithm**

Cost fusion can be performed by just adding associated costs in a manner shown in Equation 4.14. Since the minimum value of this cost function is all that matters, no cost normalization is necessary.

$$C_f(x, y, d_s) = C_s(x, y, d_s) + C_w(x, y, a_{ws} d_s) \tag{4.14}$$

In Equation 4.14, $C_s(x, y, d_s)$ is the cost of pixel $(x,y)$ at disparity $d_s$ of the short baseline stereo pair. The wide baseline cost is computed by multiplying the disparity by a scaling factor $a_{ws}$ because its baseline is different and has to be normalized to the units of the short baseline. The scaling factor $a_{ws}$ is simply the ratio of the short and wide baseline distances.

$$a_{ws} = \frac{b_w}{b_s} \tag{4.15}$$

Cost fusion produces a cost map with disparities in reference to the short baseline. A similar cost fusion function and an inverted scaling factor can be used to produce disparities in reference to the wide baseline. If the scaling factor $a_{ws}$ produces steps that are not integers (costs are only available at integer disparities), then the cost can be interpolated to non-integer disparity values in a similar process to disparity refinement described in section 4.3.1.4. Another method for cost interpolation is cubic Hermite splines [33]. This method fits a piecewise cubic polynomial to the cost curve rather than just a parabolic function.

Different baseline widths also cause issues with the size of the cost maps. A wider baseline will cause a larger disparity and thus need a larger disparity search space. As an example, if the wide baseline is twice the distance of the short baseline, the scaling factor will be 2. This means that every other disparity in the wide baseline set will be used, and to match every disparity in the short baseline map, costs need to be computed for double the number of disparities in the wide baseline cost map.

### 4.4.3 Extension to Multi-axis stereo systems

One of the camera configurations that is examined in this thesis is the multi-axis stereo configuration. Previous sections about matching algorithms, cost fusion, and disparity fusion were only considering scenarios where cameras were aligned along a single axis. In vertical stereo systems, the epipolar lines in the image pair are vertical. To make the lines align with the horizontal pair, both the reference image and vertical pair are rotated by 90°. This allows the matching algorithms to compute disparity based on the assumption that the epipolar lines are row-aligned. For disparity fusion the process is exactly the same, except the disparity maps have to be rotated back 90° before fusion. The equations for cost fusion are also identical to those for multi-baseline fusion.

$$C_f(x, y, d_h) = C_h(x, y, d_h) + C_v(x, y, a_{hv}d_h) \tag{4.16}$$

Where the subscript $h$ denotes the horizontal pair and the subscript $v$ denotes the vertical pair. The scaling factor is still a ratio of the baselines.

$$a_{hv} = \frac{b_v}{b_h} \tag{4.17}$$

Similar to multi-baseline cost fusion, cost fusion for multi-axis configurations can be performed before and after cost aggregation. Figure 4.10 shows the pathway that three images captured by a multi-axis camera configuration take to produce a disparity map using cost fusion before aggregation. The rotation of the images and cost matrices is the only difference to the multi-baseline cost fusion algorithm (Figure 4.9).

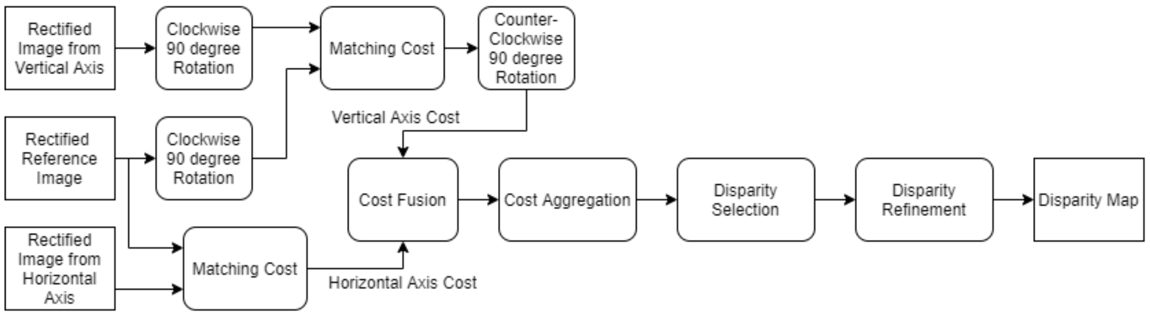

Rectified Image from Vertical Axis
Clockwise 90 degree Rotation
Matching Cost
Counter-Clockwise 90 degree Rotation
Vertical Axis Cost
Rectified Reference Image
Clockwise 90 degree Rotation
Cost Fusion
Cost Aggregation
Disparity Selection
Disparity Refinement
Disparity Map
Rectified Image from Horizontal Axis
Matching Cost
Horizontal Axis Cost

**Figure 4.10: Flowchart of the multi-axis cost fusion before aggregation algorithm showcasing a modification of the SGM algorithm**

## 4.5 Depth Map Evaluation

The performance of a depth estimation algorithm is typically measured depending on what application it is used for. Some depth estimation algorithms may impose significant smoothing on the output depth map which can decrease the number of falsely matched pixels, but may filter out important details. Other algorithms may be more accurate across the entire depth map but have difficulties with regions like flat surfaces [38]. Different applications require different metrics to determine how successful each depth algorithms is. Because the specific goal of this thesis is to improve the accuracy of the depth map generated, some quantitative depth metrics evaluating the accuracy of each point in the depth map need to selected. Since the application for this thesis is 3D object detection and localization, some qualitative metrics determining how well a disparity map describes the shapes of objects will also be employed.

### 4.5.1 Quantitative Metrics

Quantitative metrics use measured data to evaluate the performance of depth maps. Quantitative methods offer the advantage of being objective against human-related biasing [38]. Metrics to estimate the accuracy of depth maps fit into two categories: those that compute error from a disparity map, and those that compute error from depth maps. One thing to note about error metrics is that they return a measure of the amount of error which means that the best performing depth maps will have the lowest error metric.

#### 4.5.1.1 Disparity Error Metrics

Disparity metrics measure how well an estimated disparity map matches the ground truth disparity map. The most common metric is Bad Matched Pixels, and is the most straightforward approach to defining error in disparity maps.

**Bad Matched Pixels (BMP)** is a metric that compares the difference in disparity between the estimated and ground truth maps. If the difference is greater than some threshold $\delta$, it is considered a "badly matched" pixel.

$$\epsilon(x,y) = \begin{cases} 1, & \text{if } |D_{GT}(x,y) - D_{est}(x,y)| > \delta \\ 0, & \text{otherwise.} \end{cases} \tag{4.18}$$

where $D_{GT}$ is the ground truth disparity and $D_{est}$ is the estimated disparity. The threshold parameter is typically either 1, 2, or 3 pixels which denotes how close the disparities must be to regard the estimated disparity as a correct match. The BMP metric can therefore be calculated with Equation 4.19.

$$BMP = \frac{1}{N} \sum_{x,y} \epsilon(x,y) \tag{4.19}$$

where $N$ is the number of pixels in the image. The BMP metric measures the percentage of pixels in the image that are "badly matched". The issue with this metric is that the absolute error of the pixels are not considered. A disparity map can have a good BMP score with a small number of very significant errors. These significant errors could cause issues with 3D reconstruction and results later in the pipeline, even though the depth map received a good BMP score [2].

**Bad Matched Pixels Relative Error (BMPRE)** attempts to fix this issue by considering the relative error as well [39]. For all the pixels that were incorrectly matched according to the threshold parameter, the absolute difference is calculated. Since it is more difficult to accurately estimate larger disparities than smaller disparities, the absolute difference is normalized by the ground truth disparity. This makes the error scale relative to the depth triangulation rather than the absolute difference [2]. The relative difference error for each pixel is defined in the following way to avoid division by zero

$$\tau(x,y) = \begin{cases} \frac{|D_{GT}(x,y) - D_{est}(x,y)|}{D_{est}(x,y)}, & \text{if } D_{est}(x,y) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{4.20}$$

BMPRE can be calculated in a similar manner to BMP by adding the relative error if the pixel is incorrectly matched according to the threshold parameter $\delta$.

$$BMPRE = \sum_{x,y} \begin{cases} \tau(x,y), & \text{if } |D_{GT}(x,y) - D_{est}(x,y)| > \delta \\ 0, & \text{otherwise.} \end{cases} \tag{4.21}$$

BMPRE allows for a deeper understanding of the accuracy of disparity maps by considering error magnitude as well. A larger BMPRE metric means that the disparity maps contain more relative difference. BMP is still the most prevalent metric for comparing matching algorithms, and in this thesis both metrics are used.

#### 4.5.1.2 Depth Error Metrics

Depth metrics measure the performance of a depth map given a ground truth depth map. Framing the problem in terms of depth can make the magnitude of the metrics be easier to understand because the units are a known physical quantity.

**Mean Absolute Error (MAE)** is one of the simplest error metrics to determine the magnitude of difference between two points. The formula for the mean absolute error can be found simply with

$$\text{MAE} = \frac{1}{n} \sum_{x,y} |D_{GT}(x,y) - D_{est}(x,y)| \tag{4.22}$$

which determines how far, on average, the estimated depth of a pixel is from its true location.

**Mean Squared Error (MSE)** measures the squared difference between the estimated and ground truth depth values. MSE is the second moment of the error and thus includes variance in its measurements. This means that a large difference between the estimated and true values will have a more significant impact on the error than smaller differences. MSE can be found using Equation 4.23.

$$MSE = \frac{1}{n}\sum_{x,y}(D_{GT}(x,y) - D_{est}(x,y))^2 \tag{4.23}$$

A challenge with depth metrics is that there is no way to distinguish whether or not there are a large number of small errors or a small number of large errors. This information can be determined from BMP, which is why all these metrics are used in tandem to get a better picture of the accuracy of the depth estimation algorithms.

### 4.5.2 Qualitative Metrics

Qualitative metrics are metrics that do not depend on formulas or measurements. They instead depend on subjective properties determined by an observer [40]. For depth estimation algorithms this usually involves examining either the estimated disparity map or depth map in a visual form, looking for problem areas. Depth or disparity maps can be expressed as a grayscale image by normalizing all the depths or disparities between 0 and 255. In a disparity map, small depths will be lighter pixels while large depths will be darker pixels. The inverse is true for a depth map because depth has an inverse relationship to disparity. These images can be compared to the visualized versions of ground truth data. An example of a grayscale disparity map is shown in Figure 4.11.

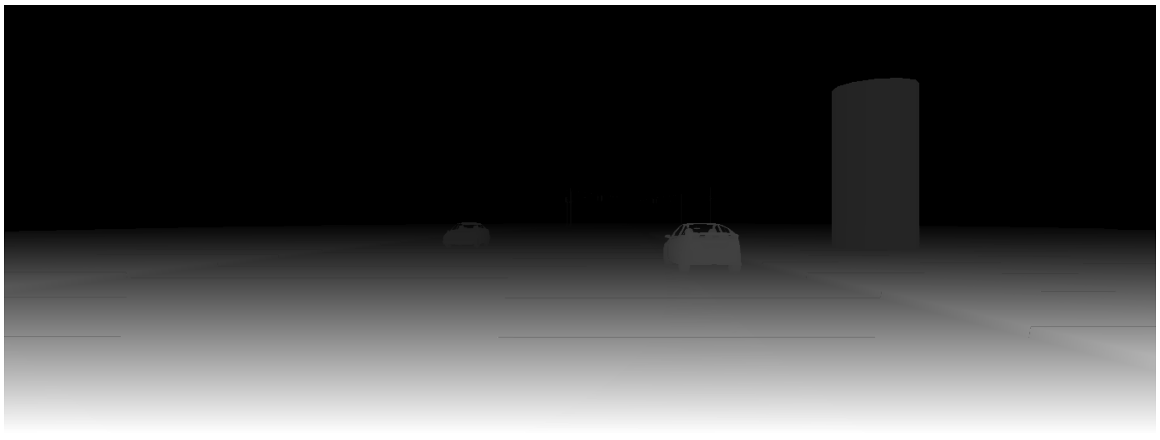


**Figure 4.11: Ground truth disparity map visualized in grayscale**

Another way to visualize the output of depth estimation algorithms is in point-cloud form. In this thesis point-cloud form is the data representation that is input into the pipeline for 3D object detection and localization. Point-cloud form also allows visualizing the data from multiple perspectives which can be better when determining how well certain objects are depicted in a depth map. Point-clouds are three dimensional data that require 3D rendering software to display on a computer. Images of the 3D rendered point-clouds are shown in this thesis. An example of the ground truth point-cloud from the CARLA custom scene is shown in Figure 4.12. The points in the point-clouds for this thesis are all black except for the points corresponding to the two vehicles which are red. Since vehicles are the most important features in the scene, the points corresponding to the vehicles are under the most scrutiny.

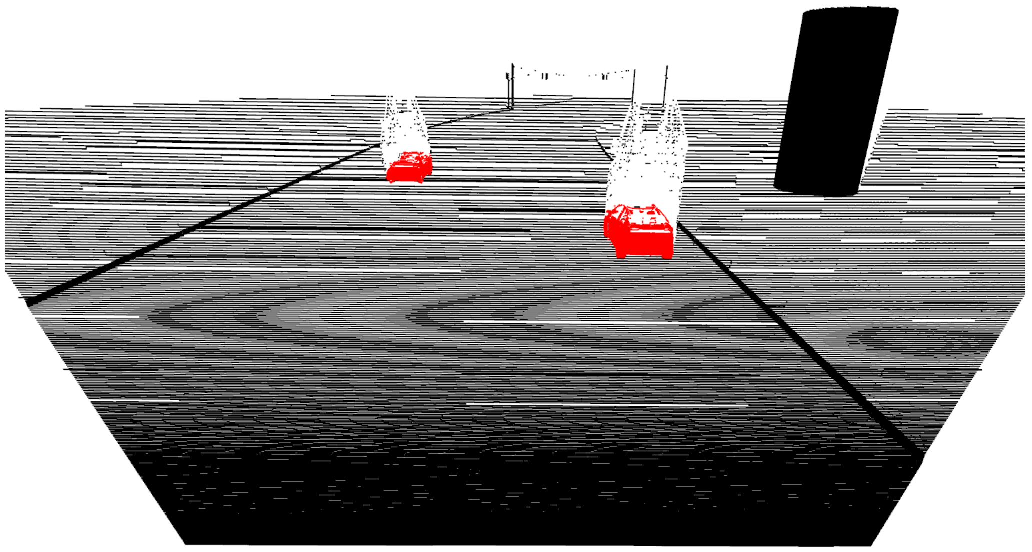


**Figure 4.12: Ground truth depth map visualized in point-cloud form**

# Chapter 5

# RESULTS

The evaluation of triple-camera depth estimation was split into three sections based on the three camera configurations: Traditional Stereo, Multi-baseline stereo, and Multi-axis stereo. For each of the camera configurations, quantitative and qualitative metrics were determined from the depth and disparity maps generated by each algorithm when using images captured of the two environments. Metrics were calculated across the entire depth and disparity maps and over specific regions in the depth and disparity maps. When using images of the CARLA custom environment, the region of pixels containing the near and far vehicles and depths in 10-meter intervals from 0 to 150 meters were examined. When using images of the CARLA city environment, metrics were only calculated across the entire depth and disparity maps.

## 5.1 Traditional Stereo

The purpose of evaluating a traditional stereo configuration was to determine the baseline performance of common stereo algorithms on the new synthetic image dataset. Five different stereo algorithms were used: OpenCV's StereoBM [41], OpenCV's StereoSGBM [41], PSMNet [29], GWCNet [30], and a custom implementation of the SGM algorithm discussed previously.

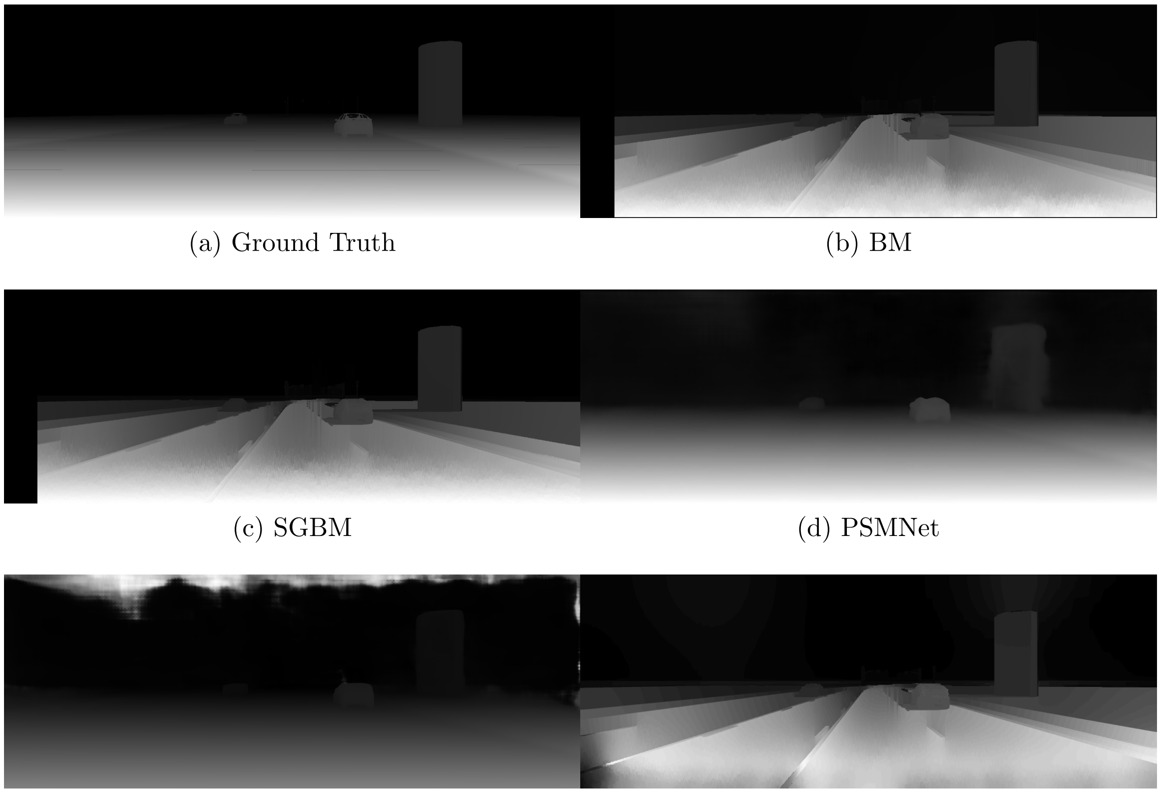
OpenCV's StereoBM algorithm is an implementation of Konolige's block matching algorithm which is a local method that matches blocks of pixels using area correlation [42]. OpenCV's StereoSGBM algorithm is OpenCV's version of SGM discussed in this thesis, but rather than working at the pixel level, cost is computed per block [41].

A block size of 5 was selected and cost aggregation was performed in 8 directions similar to the custom implementation of SGM. Both OpenCV algorithms were post-filtered using OpenCV's WLS filter which uses a grayscale image to align the disparity map edges and propagate disparity values to occluded regions [41]. In contrast, PSMNet [29] and GWCNet [30] are both convolutional neural networks (CNN) that perform stereo matching. Both algorithms were created and trained on the KITTI dataset and are among the top performing stereo matching algorithms on the KITTI leaderboards. PSMNet was one of the algorithms used in the original Psuedo-Lidar paper for depth estimation. A pretrained version (trained on the KITTI 2015 dataset) of each model was used to perform stereo matching on the new dataset. Note that because these models were trained using an entirely different dataset, this study does not determine their true performance and is only used as a comparison tool for the other models.

### 5.1.1 Short Baseline

The first configuration tested was a stereo camera rig with a baseline of 0.5 meters. In Figure 4.2 this corresponds to configuration (a) which captures images from the reference and center cameras. Each algorithm was used to generate a disparity map and this disparity map was normalized and converted into a grayscale image. These disparity maps along with the ground truth disparity are shown in Figure 5.1.

The first thing to notice about these disparity map representations is that the BM and SGBM algorithms have black areas in the far-left side of their depth maps. These regions are implemented as invalid regions in the respective algorithms. Otherwise, the big takeaway from these images is that the CNN matching algorithms (PSMNet and GWCNet) produce very smooth disparity maps, but fail to differentiate objects located far away from the background. PSMNet makes the edges of the pillar wavy,


(a) Ground Truth (b) BM

(c) SGBM (d) PSMNet

(e) GWCNet (f) Custom SGM

**Figure 5.1: Disparity maps generated using various algorithms in a short baseline camera configuration**

while there are artifacts in the sky in the GWCNet disparity map. In contrast, the non-CNN algorithms (BM, SGBM, and Custom SGM) produce many edges in their disparity maps, which makes sense because these methods focus on matching specific regions to other regions, and edges are some of the easiest regions to match. These algorithms struggle with flat regions like the road that have very little texture and make it difficult to determine a match given a points surroundings. It seems like the CNN algorithms have ”learned” about the flat gradient of the road in their training which is one of the notable problem areas in the non-CNN algorithm disparity maps.

Next, quantitative metrics for each algorithm were calculated from the disparity maps and ground truth data. First, the metrics calculated across entire disparity maps are analyzed. Table 5.1 and Table 5.2 show the BMP and BMPRE metrics across the entire disparity map.

**Table 5.1: BMP values across entire disparity maps generated by different algorithms in a short baseline camera configuration**

| BMP | | | |
|---|---|---|---|
| Algorithm | $\delta$=3px | $\delta$=2px | $\delta$=1px |
| BM | 0.475 | 0.597 | 0.745 |
| SGBM | 0.372 | 0.458 | 0.596 |
| PSMNet | 0.035 | 0.115 | 0.405 |
| GWCNet | 0.042 | 0.067 | 0.129 |
| Custom SGM | 0.648 | 0.732 | 0.844 |

Note that the BMP values range from 0 to 1 and as the threshold $\delta$ increases, BMP decreases. This is expected because BMP calculates the percentage of pixels over a certain threshold and as that threshold increases, the number of pixels above can only ever decrease. This means that a lower BMP value is better, and noticeably the two CNN algorithms significantly outperform the non-CNN algorithms. This is likely because these algorithms perform significantly better on the road area which constitutes most of the pixels in the image.

**Table 5.2: BMPRE values across entire disparity maps generated by different algorithms in a short baseline camera configuration**

| BMPRE | | | |
|---|---|---|---|
| Algorithm | $\delta$=3px | $\delta$=2px | $\delta$=1px |
| BM | 1,108,368 | 1,111,279 | 1,113,580 |
| SGBM | 941,552 | 944,229 | 946,488 |
| PSMNet | 12,290 | 18,737 | 26,245 |
| GWCNet | 13,880 | 16,839 | 19,188 |
| Custom SGM | 79,048 | 83,099 | 86,350 |

This same trend is reflected in the BMPRE values, where the CNN algorithms are an order of magnitude lower than the non-CNN algorithms. One observation is that the Custom SGM algorithm has a significantly lower BMPRE than the other non-CNN algorithms while at the same time having a slightly higher BMP. This means that

Custom SGM has a large number of small errors, while the algorithms BM and SGBM have a smaller number of errors, but these errors are considerably larger.

Next, the metrics for each depth map were determined for various distances from the stereo camera setup. Figure 5.2 shows how each algorithm performed over regions located various distances from the stereo camera setup.

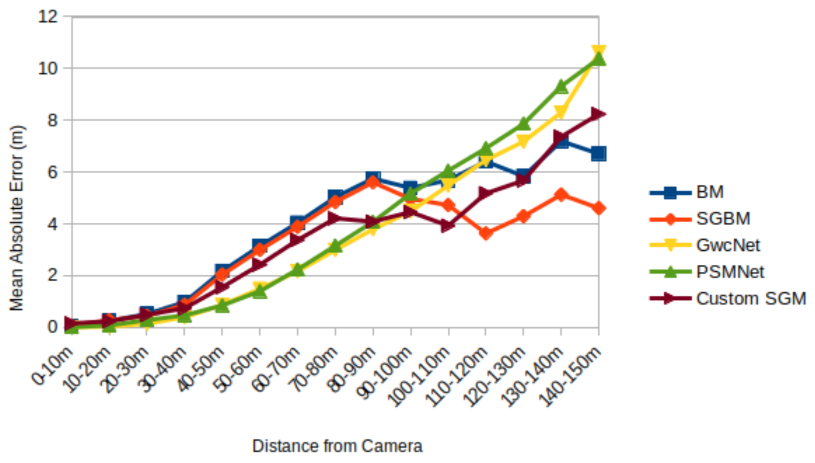


**Figure 5.2: Comparison of MAE values from depth maps generated by different algorithms in a short baseline camera configuration at various 10m intervals**

In this plot, the MAE increases as the distance from the camera increases, which is expected because depth error is proportional to the square of the distance as noted in Equation 4.1. The CNN algorithms are smoother than the non-CNN algorithms and perform better at smaller distances, but are worse at distances greater than 100m.

The plot for the metric MSE showed similar results to MAE (Figure 5.3). Smaller distances showed a more pronounced difference with MSE than MAE, but the overall shape of the plots was similar except for a bump occurring at the 100-110m range in SGBM.

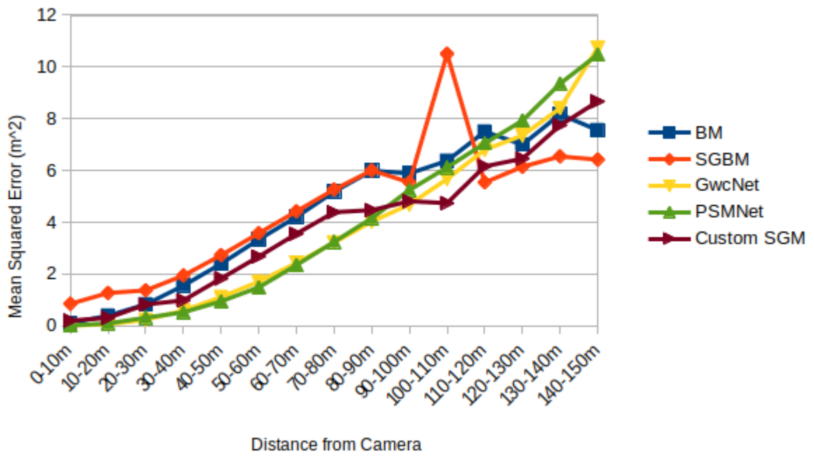


**Figure 5.3: Comparison of MSE values from depth maps generated by different algorithms in a short baseline camera configuration at various 10m intervals**

The final regions examined were the near and far vehicles. The purpose was to examine how these algorithms performed around specific objects of interest. The plot in Figure 5.4 compares the MAE between each vehicle for each algorithm.

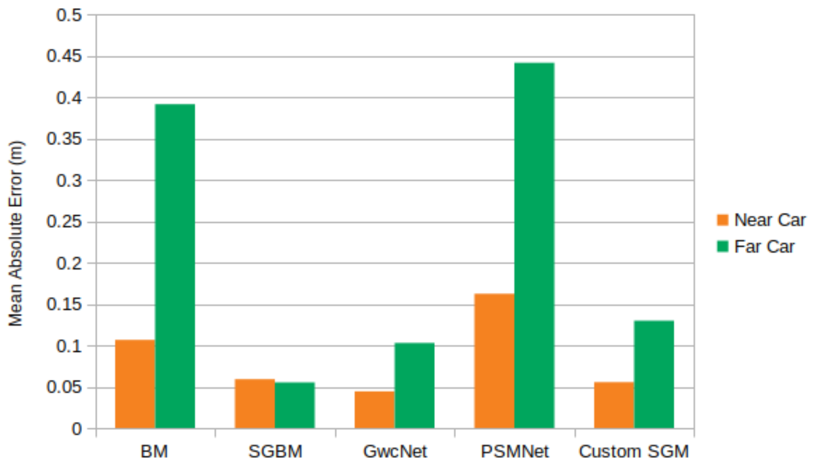


**Figure 5.4: Comparison of MAE values from depth maps generated by different algorithms in a short baseline camera configuration of the near and far vehicles**

In this plot, the CNN algorithms do not significantly outperform the others. This is likely because vehicles contain many edges and features that can be locally matched which the non-CNN algorithms excel with. Also, in all algorithms except SGBM, the error for the near vehicle region is higher than the far vehicle region. This is consistent with the results from the range of distance plots which found that as distance from the stereo camera setup increases, the depth error also increases.

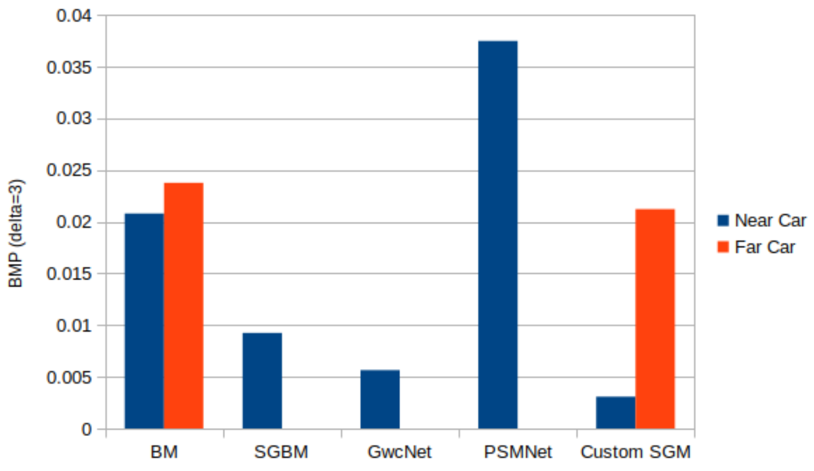
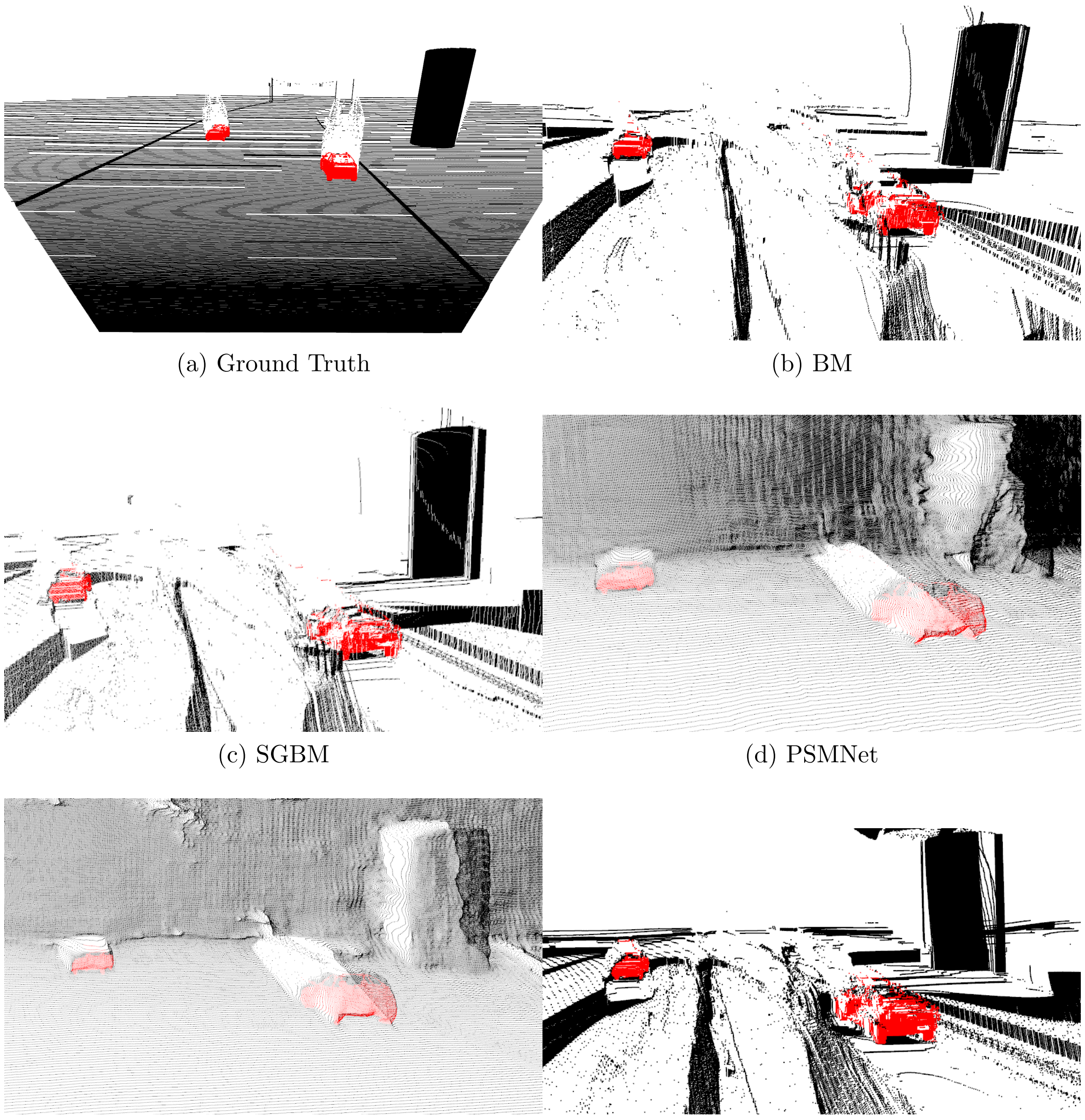


**Figure 5.5: Comparison of BMP ($\delta$=3px) values from disparity maps generated by different algorithms in a short baseline camera configuration of the near and far vehicles**

The BMP metrics show a slightly different result. The relatively large absolute error seen for the SGBM, GWCNet, and PSMNet algorithms actually results in a zero BMP ($\delta$=3px) percentage. This means that all the estimated disparities are within 3px of the true disparity however on average they are far enough away to cause a large absolute error. Therefore a combination of metrics should be used to truly evaluate the performance of stereo matching algorithms. Despite this, BMP is commonly the only metric used.

Moving to qualitative metrics, the point-cloud representations of the depth maps for each algorithm are shown in Figure 5.6. Like the disparity maps, the CNN-based algorithms (PSMNet and GWCNet) produce a smoother point-cloud which is more in-line with the ground truth point-cloud, especially for regions like the ground and road. CNN-based algorithms struggle with objects located far away like the column which has very uneven texture in the point-cloud representation which should be smooth. Additionally, CNN-based algorithms seem to have a depth limit which is shown as the large wall in the background of the point-clouds. This wall occurred at about a depth of 60 meters from the camera setup for both PSMNet and GWCNet and no objects beyond this limit are represented.



(a) Ground Truth

(b) BM

(c) SGBM

(d) PSMNet

(e) GWCNet

(f) Custom SGM

**Figure 5.6: Point-cloud representation of the depth maps generated using various algorithms in a short baseline camera configuration**

All the Non-CNN based algorithms had similar point-clouds to each other. These algorithms struggle with areas like the ground and road because they have very little texture for the algorithms to match points across the images. For objects like the vehicles and the column that have many unique edges, matching is significantly easier. For some algorithms, there are red points (Points corresponding to the vehicle in the image) in the point-cloud that stretch far past their ground truth location. Those points are outliers that cause an increase in BMP for those algorithms.

### 5.1.2 Wide Baseline

The next camera configuration tested was a stereo camera rig with a baseline of 1 meter. This is double the baseline of the short baseline stereo setup. As noted in section 4.1, increasing the baseline should improve the accuracy of depth estimation algorithm because depth error has an inverse relationship with baseline length. However, large baselines can cause more occlusions and discontinuities in the resulting images which reduce the accuracy of matching algorithms.

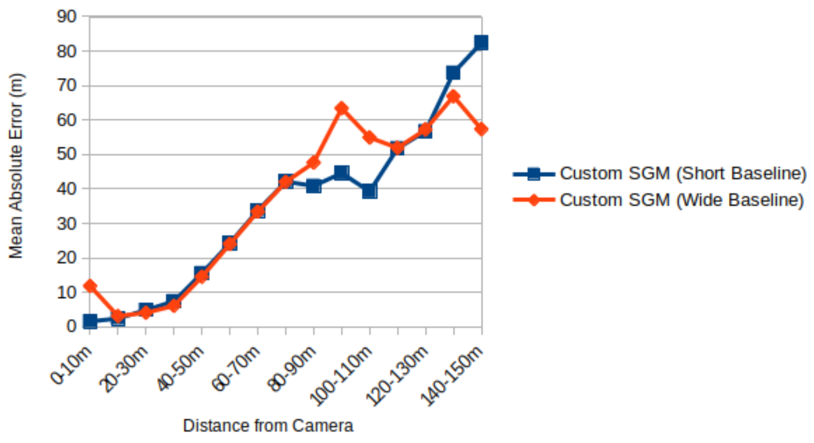


**Figure 5.7: Comparison of MAE values from depth maps generated by the custom SGM algorithm in a wide baseline camera configuration at various 10m intervals**

Figure 5.7 shows a comparison of the performance of the custom SGM algorithm for both the short and baseline systems over a range of distances. For this algorithm, the performance does not differ significantly between the short and wide baselines. Only at the extremes (small and large distances) there are any significant differences.

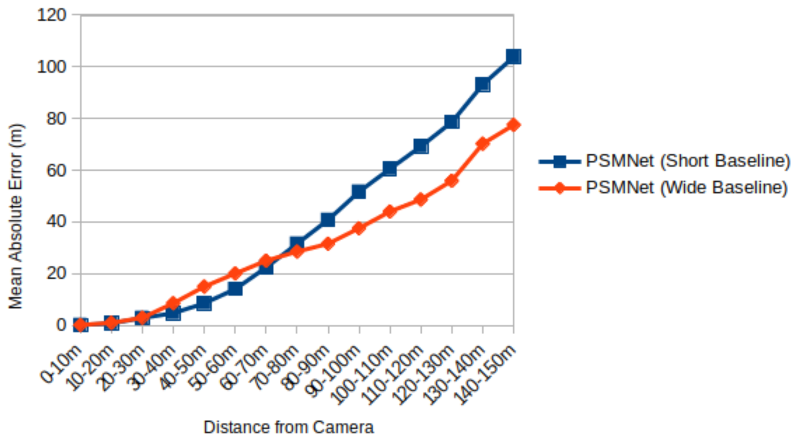
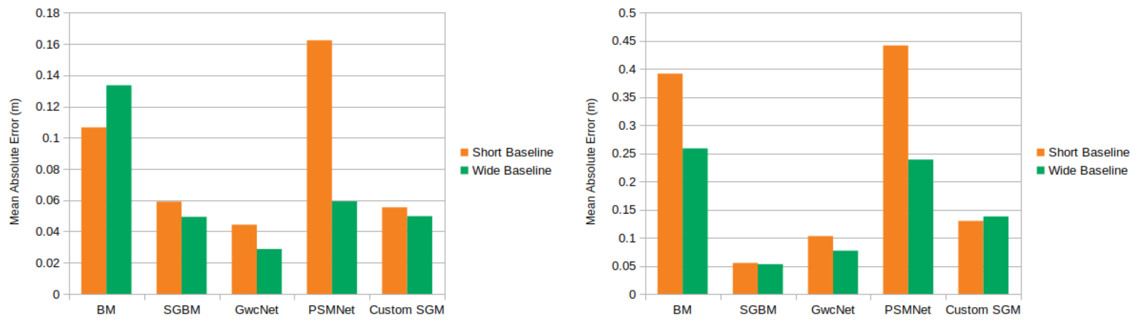


**Figure 5.8: Comparison of MAE values from depth maps generated by PSMNet in a wide baseline camera configuration at various 10m intervals**

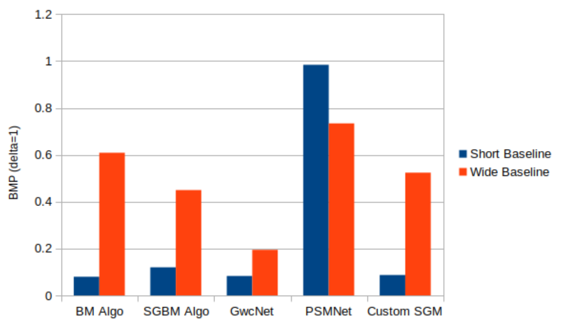
In contrast, Figure 5.8 shows a noticeable difference in the performance of the PSMNet algorithm for the short and wide baselines. At smaller distances, the MAE is larger for the wide baseline meaning that the short baseline has better performance. After a turning point at around 70m, the opposite is true.



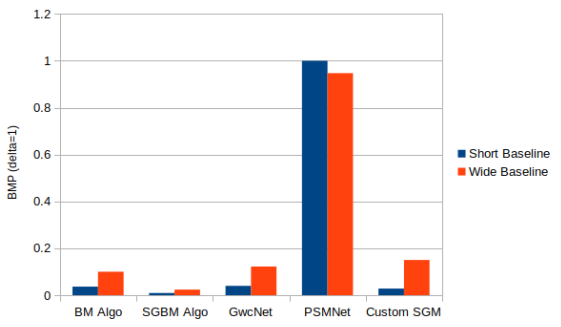
(a) Near Vehicle (b) Far Vehicle

**Figure 5.9: Comparison of MAE between short and wide baseline camera configurations from depth maps generated by different algorithms**

Looking specifically at the vehicle regions in the depth maps, it is interesting to see that the wide baseline configuration had a higher BMP (Figure 5.10) but at the same time a lower MAE (Figure 5.9) in comparison to the short baseline configuration. This means that the short baseline configuration had more exact matches, but overall had a larger average difference for each pixel in the depth map. Ignoring baseline, the MAE for the far vehicle is higher than the near vehicle for all algorithms, but the opposite is true for BMP metric where the far vehicle has the lowest error for all algorithms.



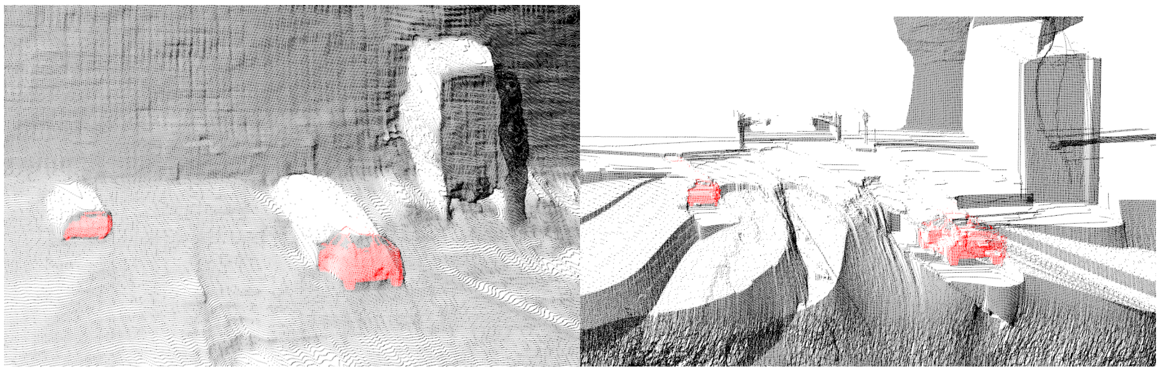
(a) Near Vehicle



(b) Far Vehicle

**Figure 5.10: Comparison of BMP ($\delta = 1$) between short and wide baseline camera configurations from disparity maps generated by different algorithms**

The point-clouds generated by the algorithms in the wide baseline configuration overall are very similar to the point-clouds generated by the algorithms in the short baseline configuration. However there are some differences. In the Custom SGM point-cloud, there are large holes in the ground in the regions closest to the camera. As the baseline increases, the disparity for the closer points can increase significantly. A matched pixel can have a disparity so large that it is out of range of the possible disparities checked ensuring that an incorrect disparity is chosen. The wide baseline configuration is poor at depth estimation for small distance but can detect the traffic lights located about 150 meters away which was the first algorithm to do so.



(a) PSMNet (b) Custom SGM

**Figure 5.11: Point-cloud representations of the depth maps generated using various algorithms in a wide baseline camera configuration**

Another takeaway is that the PSMNet algorithm depicts the shape of the vehicle well in the point-cloud, but the BMP value for both vehicles is almost one meaning that most of the pixels are off by at least one pixel. This means that PSMNet is good at describing the shapes in the images but struggles to accurately localize the points in the shape. This is likely due to the model being trained on a different dataset.

Overall, it is difficult to determine if a larger baseline improves the accuracy of stereo vision systems. At smaller distances, a short baseline is better, while at larger distances, a wider baseline is better. Next, the combination of the short and wide baseline configurations is examined.

## 5.2 Multi-Baseline Stereo

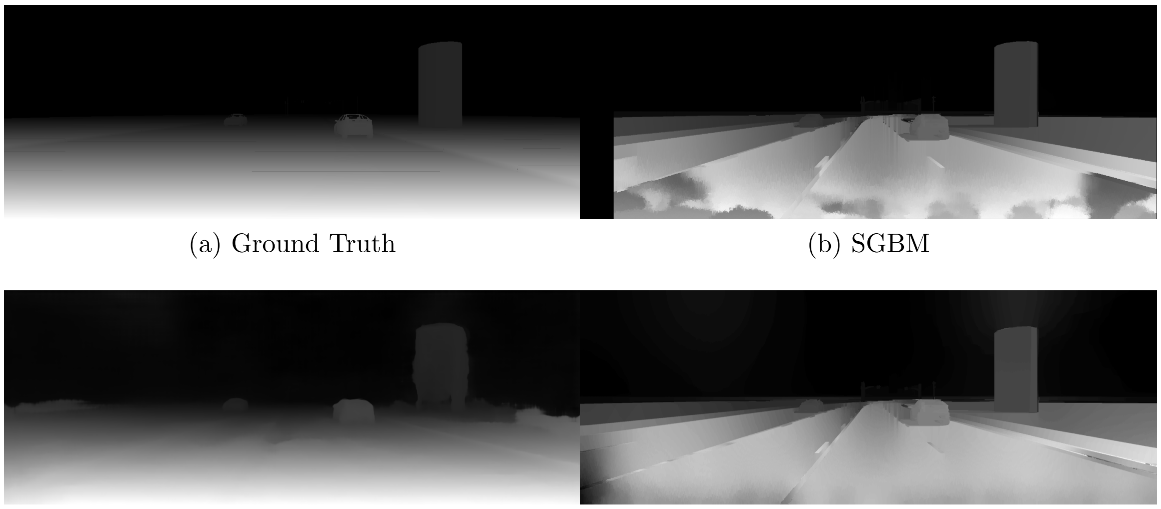
The next configuration investigated was the multi-baseline stereo configuration. In Figure 4.2 this corresponds to configuration (b) which uses images captured from a reference camera and two other cameras (center and right cameras) along a horizontal axis. Using the camera positions from both the short and wide baseline traditional stereo setups, this section analyzes if a combination of the stereo systems into a triple camera system has improved performance. Both disparity fusion and cost fusion methods are examined.

### 5.2.1 Disparity Fusion

With a short baseline length of 0.5m and a wide baseline length of 1m, the disparity fusion equation becomes

$$D_{fused}(x, y) = \frac{D_1(x, y) + D_2(x, y)}{1.5} \tag{5.1}$$

where $D_1$ and $D_2$ are the calculated disparities from the short baseline systems and wide baseline systems respectively. The benefit of disparity fusion is that any traditional stereo matching algorithm can be used. For this test, the SGBM, PSMNet, and Custom SGM algorithms were selected from the five algorithms examined previously.


(a) Ground Truth (b) SGBM

(c) PSMNet (d) Custom SGM

**Figure 5.12: Disparity maps generated using disparity fusion for various algorithms in a multi-baseline camera configuration**

The grayscale representation of the disparity fusion maps shown in Figure 5.12 are very similar to the traditional stereo configuration maps shown in Figure 5.1. One difference is that the disparity fusion maps have patches of disparity regions significantly different than the surrounding areas when these areas should be continuous. This is most obvious in the SGBM algorithm disparity map in the areas closest to the stereo camera. This likely happens because the wide baseline configuration has the highest error when matching regions located closest to the camera and the fusion of both disparities causes the wide baseline errors to propagate to the output disparity map.

Since this method performs a weighted mean of the disparities, it is expected that the fusion performance will fall somewhere in between the performance of the short and wide baseline stereo system. Looking at the plots for the three algorithms over a range of distances (Figures 5.13-5.15), this prediction held true. The mean absolute difference of the fusion algorithm falls at about the average between the short and wide baseline configurations. It is always better than the worst performing baseline length but worse than the best performing baseline length.

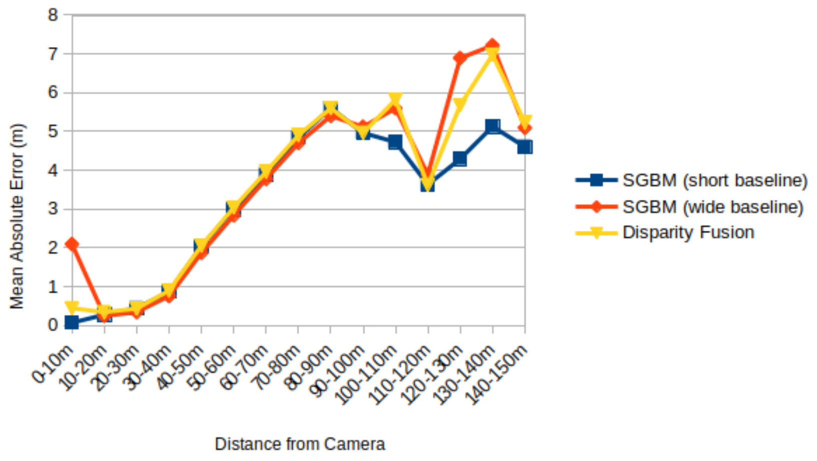


**Figure 5.13: Comparison of MAE between disparity fusion and traditional stereo configurations for the algorithm SGBM at various 10m intervals**

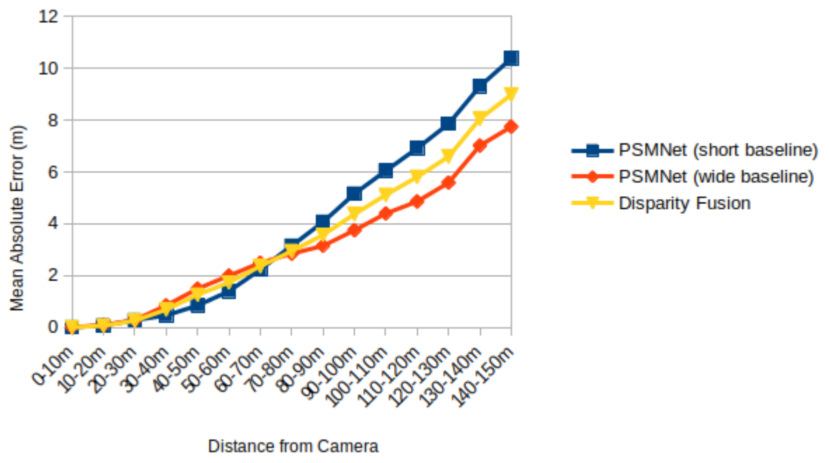


**Figure 5.14: Comparison of MAE between disparity fusion and traditional stereo configurations for the algorithm PSMNet at various 10m intervals**

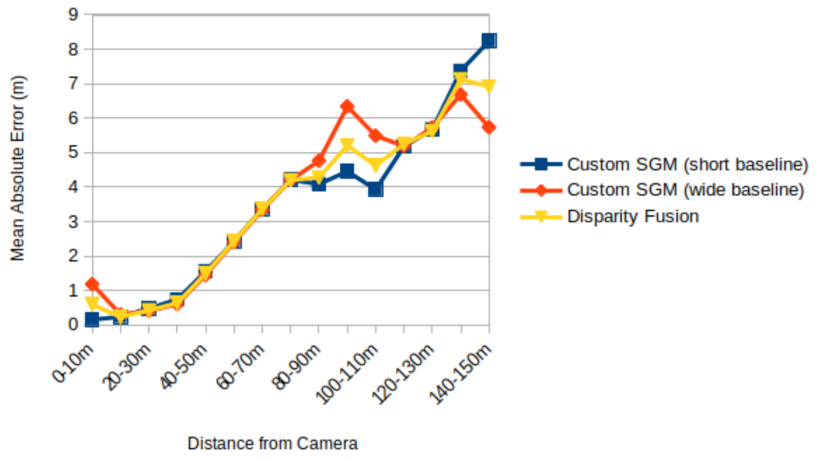


**Figure 5.15: Comparison of MAE between disparity fusion and traditional stereo configurations for the algorithm Custom SGM at various 10m intervals**

What is interesting, however, is that the BMP for the vehicles in the fusion map is higher than both the short and wide baseline disparity maps. This may be due to the issue discussed previously where errors in a disparity map may cause a good prediction in the other disparity map to be averaged into a bad prediction in the disparity fusion map.

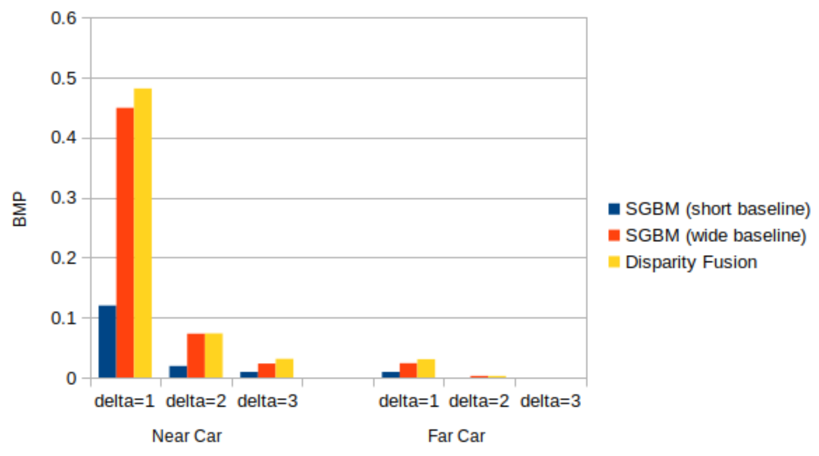


**Figure 5.16: Comparison of BMP between disparity fusion and traditional stereo configurations for the algorithm SGBM of the two vehicles**

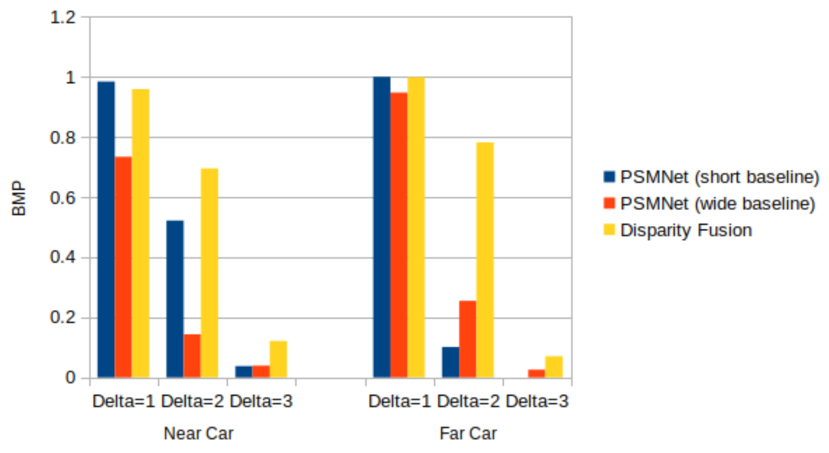


**Figure 5.17: Comparison of BMP between disparity fusion and traditional stereo configurations for the algorithm PSMNet of the two vehicles**

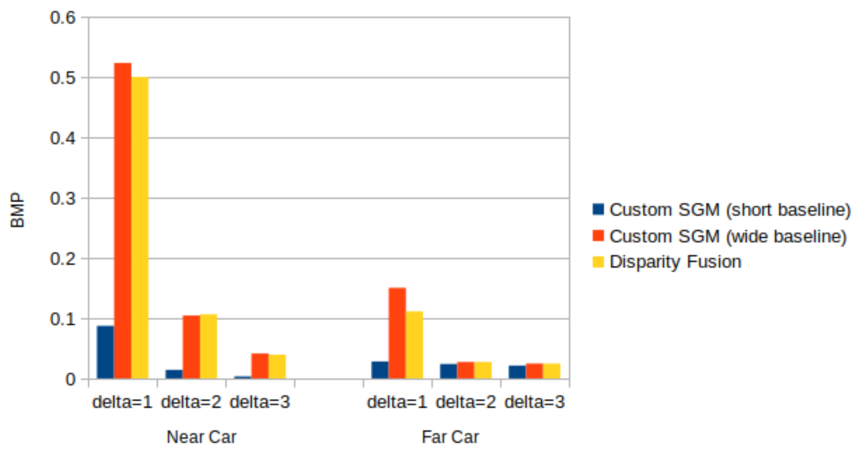
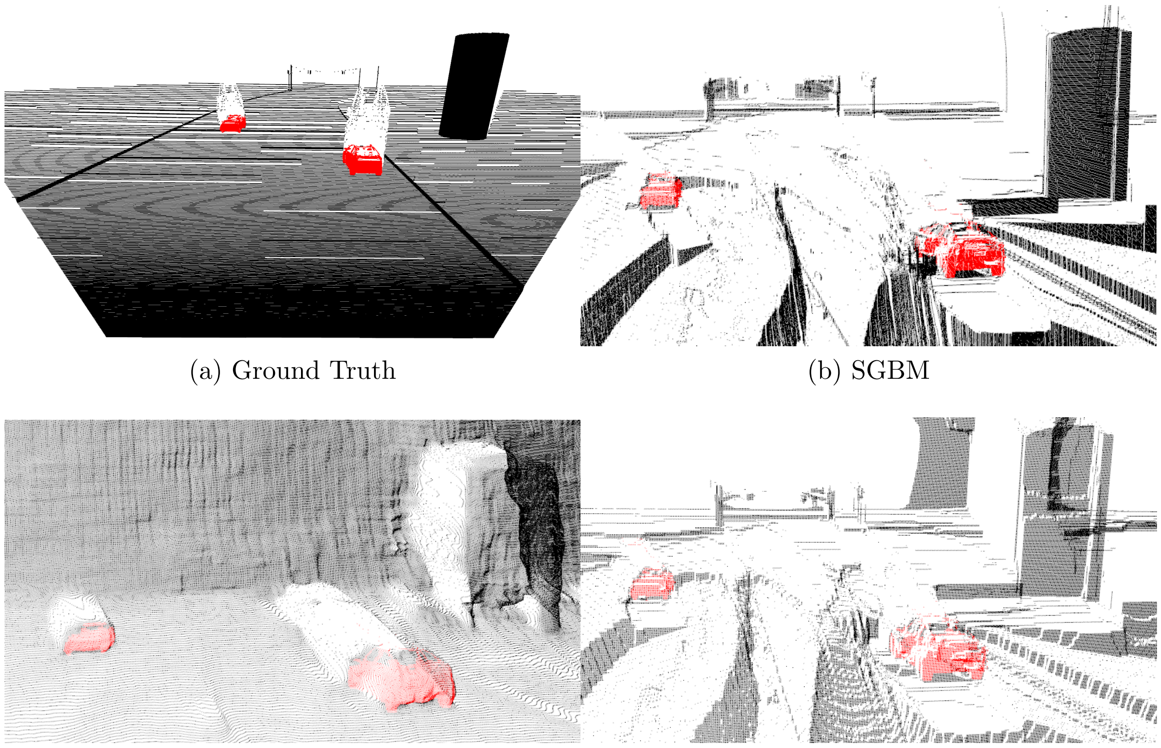


**Figure 5.18: Comparison of BMP between disparity fusion and traditional stereo configurations for the algorithm Custom SGM of the two vehicles**

The point-cloud representations of the depth maps are very similar to the traditional stereo configuration point-clouds. The point-clouds generated by SGBM and Custom SGM were able to incorporate features from the wide baseline configuration like the traffic lights, but at the same time included unwanted features like the holes in the ground in regions closest to the cameras.



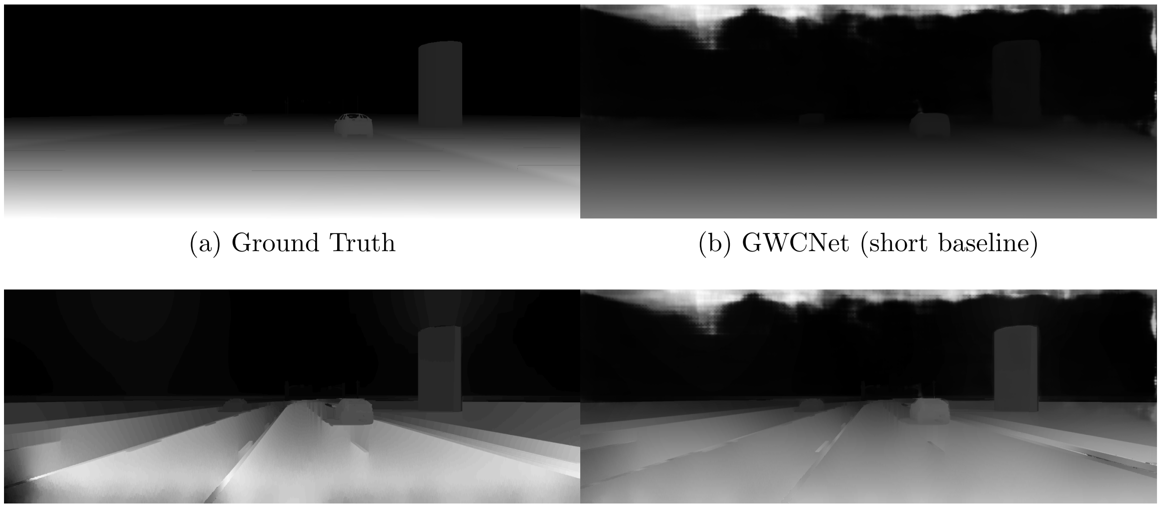
(a) Ground Truth (b) SGBM

(c) PSMNet (d) Custom SGM

**Figure 5.19: Point-cloud representations of depth maps generated with disparity fusion using various algorithms in a multi-baseline stereo configuration**

#### 5.2.1.1 Multi-Algorithm Disparity Fusion

One of the benefits of disparity fusion is that any algorithm can be used to produce the disparity map for either baseline configuration. Disparity maps from different algorithms can be fused together. The next investigation examines the feasibility of fusing a disparity map from a CNN algorithm with a disparity map from a non-CNN algorithm. For this experiment, the disparity map generated by the algorithm GWCNet in the short baseline camera configuration and the disparity map generated by the algorithm Custom SGM in the wide baseline camera configuration were chosen as the two inputs to disparity fusion.


(a) Ground Truth

(b) GWCNet (short baseline)

(c) Custom SGM (wide baseline)

(d) Disparity Fusion (GWCNet and Custom SGM)

**Figure 5.20: Comparison between disparity maps generated by algorithms in a traditional camera configuration and disparity maps generated by disparity fusion**

Looking at the disparity fusion map of the GWCNet and Custom SGM algorithms, features from both depth maps are integrated into the result. The sharper edges from the Custom SGM disparity map, and the artifacts in the sky from the GWCNet disparity map are the most noticeable. Overall, the GWCNet disparity map smoothed out the rough edges of the Custom SGM disparity map which makes it look more similar to the ground truth disparity map.

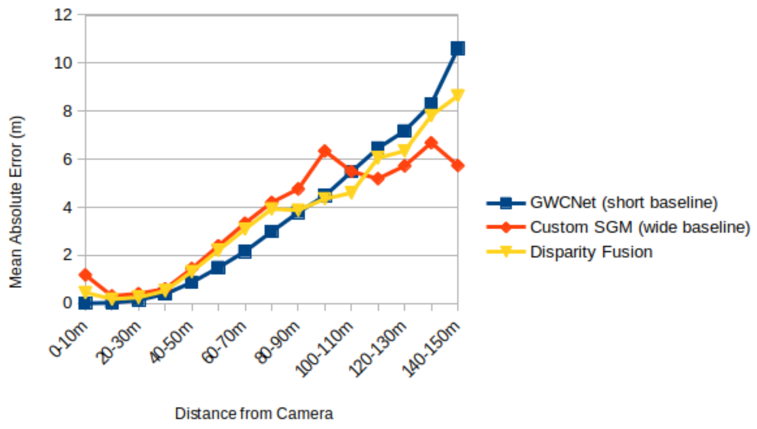


**Figure 5.21: Comparison of MAE between disparity fusion and traditional stereo configurations at various 10m intervals**

The metrics of the fused depth map are consistent with the results from the other disparity fusion depth maps. Both the MAE and BMP values for the fusion disparity maps fall between the metrics of the individual disparity maps.

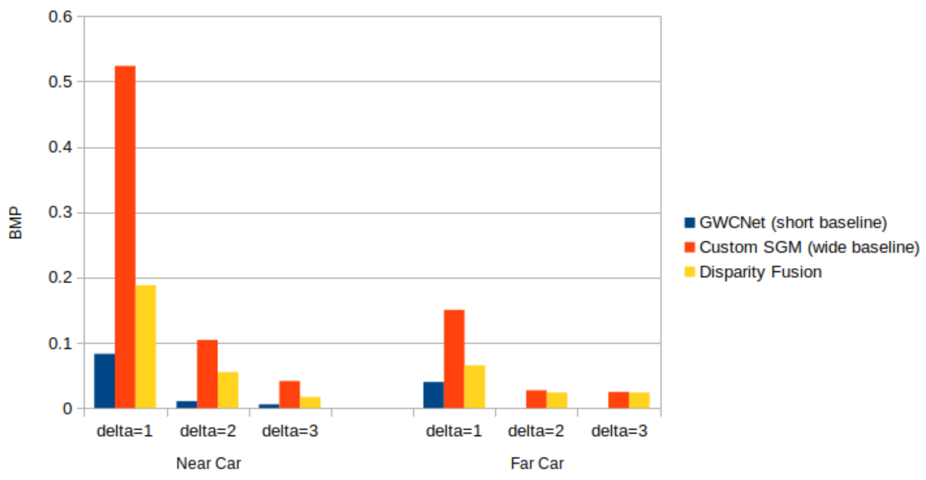
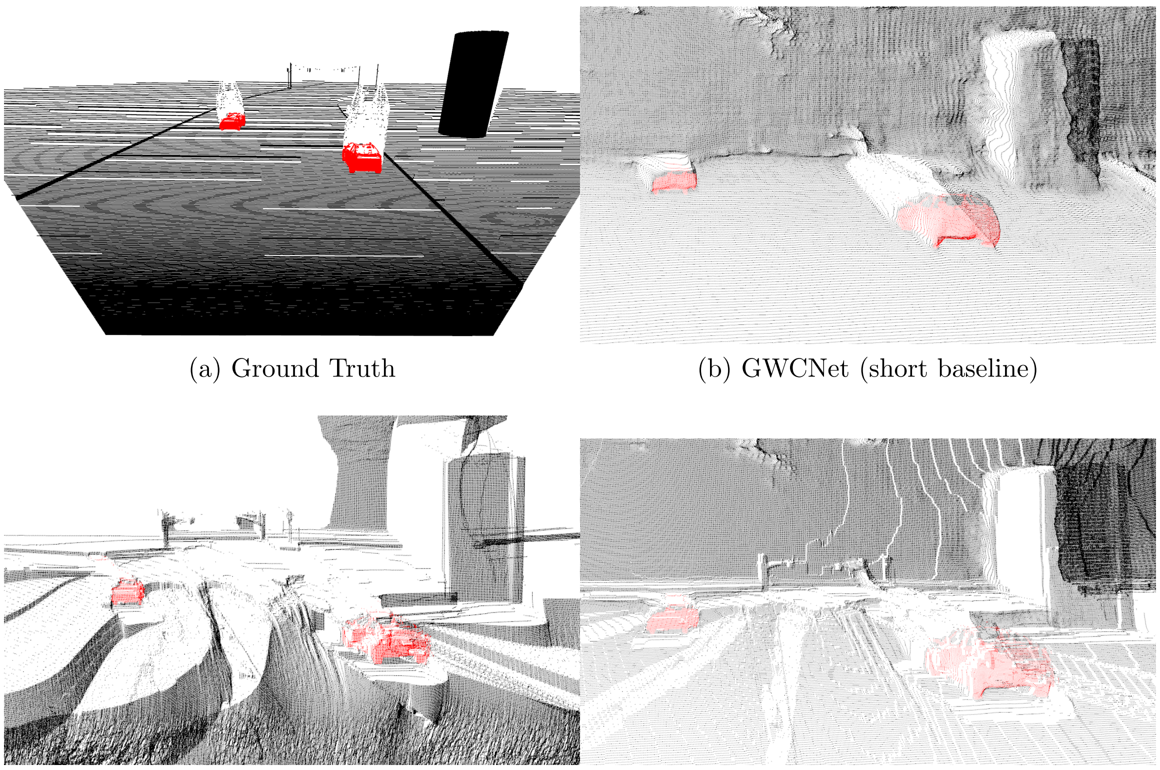


**Figure 5.22: Comparison of BMP between disparity fusion and traditional stereo configurations for the near and far vehicles**

The point-cloud representation of the fusion depth map shows a similar story to the grayscale disparity maps, where both the sharp edges around objects from the Custom SGM disparity map and the smooth gradient of the ground from the GWCNet disparity map are incorporated together. The difficulties that the wide baseline Custom SGM algorithm had with matching in non-textured regions, especially at the smallest distances, were smoothed over by the GWCNet algorithm. At the same, GWCNet struggled with objects that had sharp edges like the streetlights and large column which became clearer in the disparity fusion point-cloud.


(a) Ground Truth

(b) GWCNet (short baseline)

(c) Custom SGM (wide baseline)

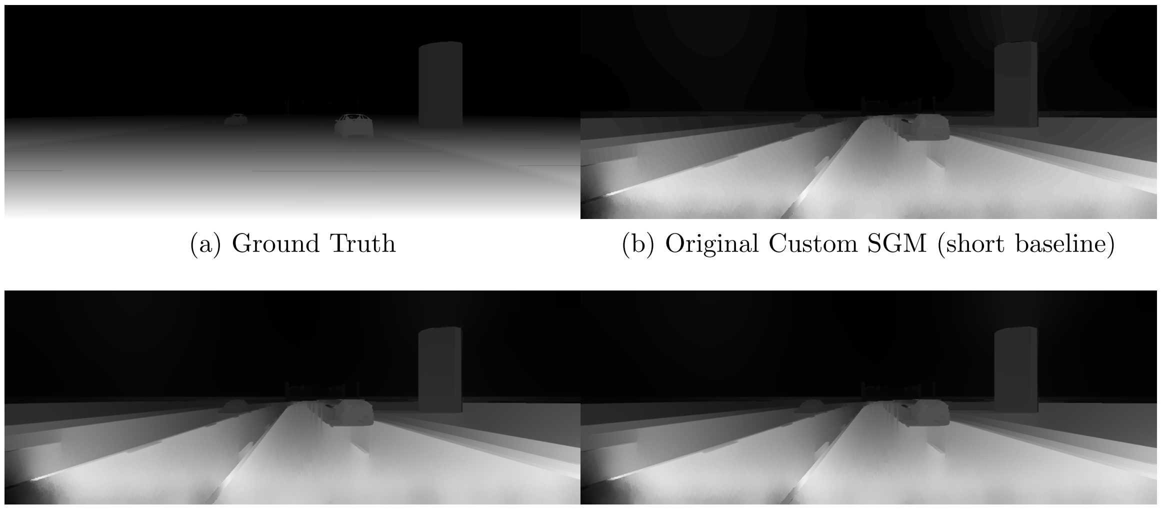
(d) Disparity Fusion

**Figure 5.23: Comparison of point-clouds generated from traditional stereo algorithms and disparity fusion**

Fusing CNN-based algorithms with their non-CNN counterpart does seem to provide a benefit to the overall disparity map, by combining the smoothness of textureless regions of the CNN algorithms with the sharp edges around objects of the non-CNN algorithms.

### 5.2.2 Cost Fusion

The final method tested for multi-baseline depth estimation was cost fusion. Since cost fusion is an approach designed specifically for algorithms that use costs to determine disparities, only the custom SGM algorithm can be modified to support cost fusion. Additionally, because cost fusion can be performed at two different points in the SGM algorithm, both methods of cost fusion are examined.



(a) Ground Truth (b) Original Custom SGM (short baseline)

(c) Cost Fusion before Cost Aggregation (d) Cost Fusion after Cost Aggregation

**Figure 5.24: Disparity maps generated using cost fusion in a multi-baseline camera configuration**

The grayscale representations of the cost fusion maps are very similar to the original SGM algorithm disparity map. This makes sense because the basic structure of the algorithm remains the same except that additional information is integrated during the cost fusion step.

Looking at Figure 5.25, both cost fusion approaches closely follow the lowest mean absolute difference across all distances which contrasts with disparity fusion which just followed the average between the short and wide baseline configurations.

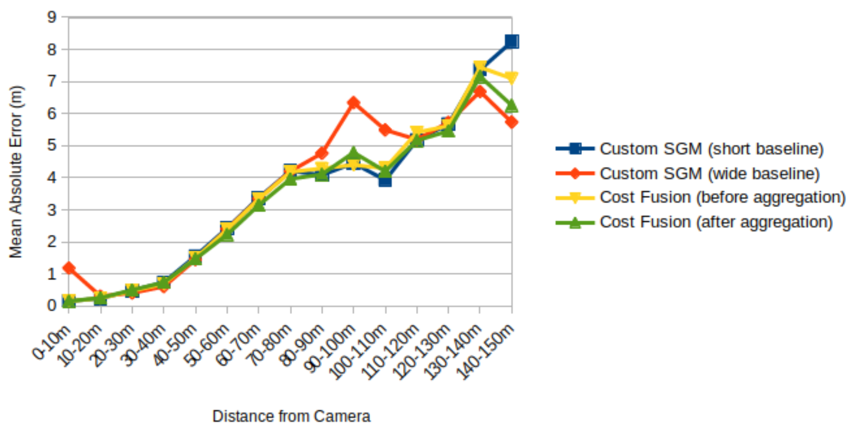


**Figure 5.25: Comparison of MAE between cost fusion and traditional stereo configurations for the algorithm Custom SGM at various 10m intervals**

Using disparity fusion, BMP was higher than both the short and wide baseline stereo configurations. With cost fusion, both fusion before aggregation and fusion after aggregation algorithms had BMP values for the two vehicles the same or lower than the best performing stereo configuration.

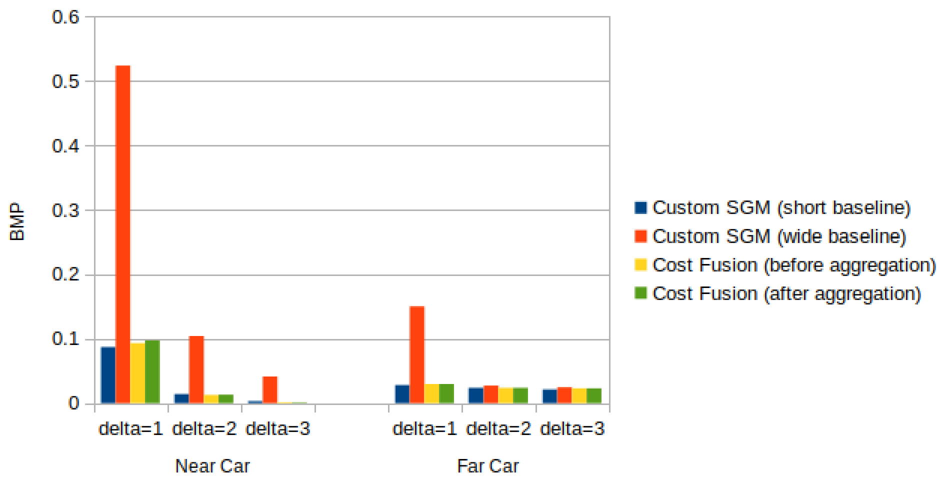


**Figure 5.26: Comparison of BMP between cost fusion and traditional stereo configurations for the algorithm Custom SGM of the near and far vehicles**

Very similar results can be seen in the BMPRE plot for the two vehicles as well. Cost fusion for a multi-baseline configuration had a slight performance improvement over the best performing individual stereo configurations.

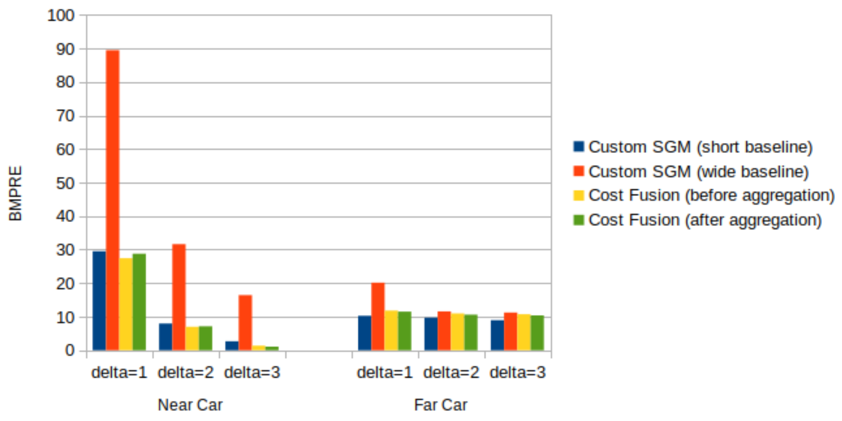
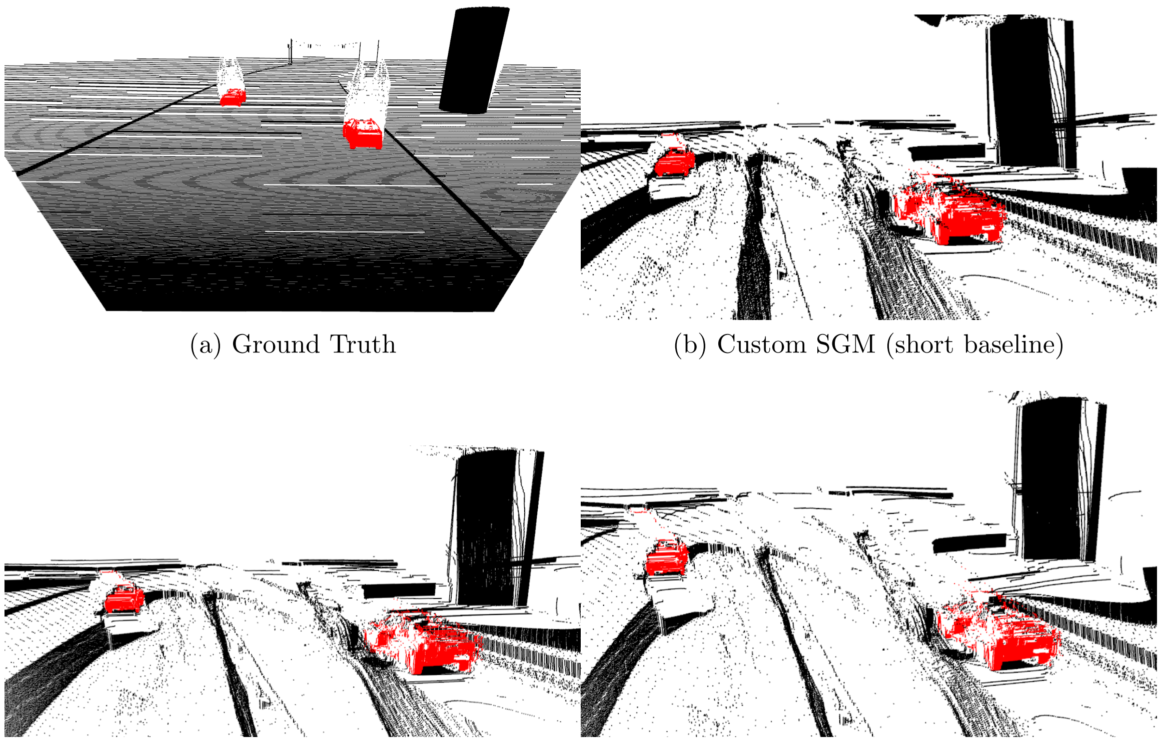


**Figure 5.27: Comparison of BMPRE between cost fusion and traditional stereo configurations for the algorithm Custom SGM of the near and far vehicles**

The point-cloud representations of the cost fusion depth maps do not differ significantly from the original Custom SGM algorithm in the short baseline configuration. Cost fusion still struggles with regions having little texture like the road.



(a) Ground Truth (b) Custom SGM (short baseline)

(c) Cost Fusion (Before Aggregation) (d) Cost Fusion (After Aggregation)

**Figure 5.28: Comparison of point-clouds generated using cost fusion for a multi-baseline camera configuration**

In comparison to both traditional stereo configurations and disparity fusion for the algorithm Custom SGM, cost fusion performs on par or slightly better in terms of the accuracy of the resulting depth and disparity maps. Looking at the MAE plot in Figure 5.29, both cost fusion before and after aggregation are the best performing algorithms over the entire depth range.

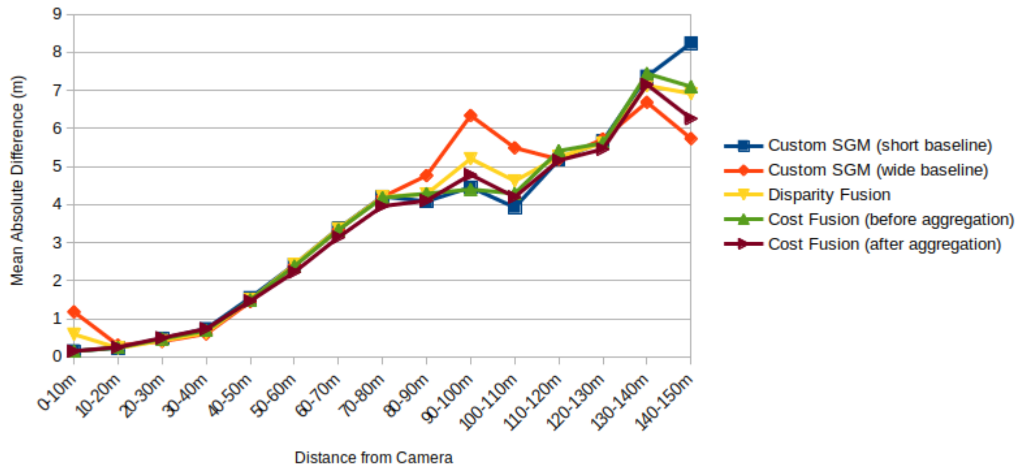


**Figure 5.29: Comparison of MAE between fusion methods and traditional stereo in multi-baseline camera configurations at various 10m intervals**

Specifically looking at the vehicles in the scene, the MAE for either cost fusion method was at least 13% lower for the near vehicle, and at least 5% better for the far vehicle than the other methods.

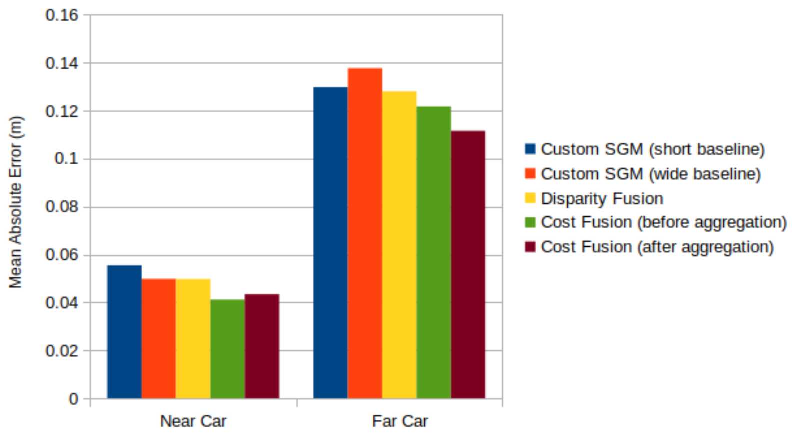


**Figure 5.30: Comparison of MAE between fusion methods and traditional stereo in multi-baseline camera configurations of the near and far vehicles**

## 5.3 Multi-axis Stereo

The final configuration investigated was the multi-axis stereo configuration. In Figure 4.2 this corresponds to configuration (c) which uses images captured from a reference camera, a camera positioned horizontally (center camera), and another camera positioned vertically (top camera) from the perspective of the reference camera. This section analyses if the combination of a vertical and horizontal stereo system into a triple-camera system improves performance over traditional stereo configurations and the multi-baseline configuration. Like the multi-baseline configuration, both disparity fusion and cost fusion methods are examined.

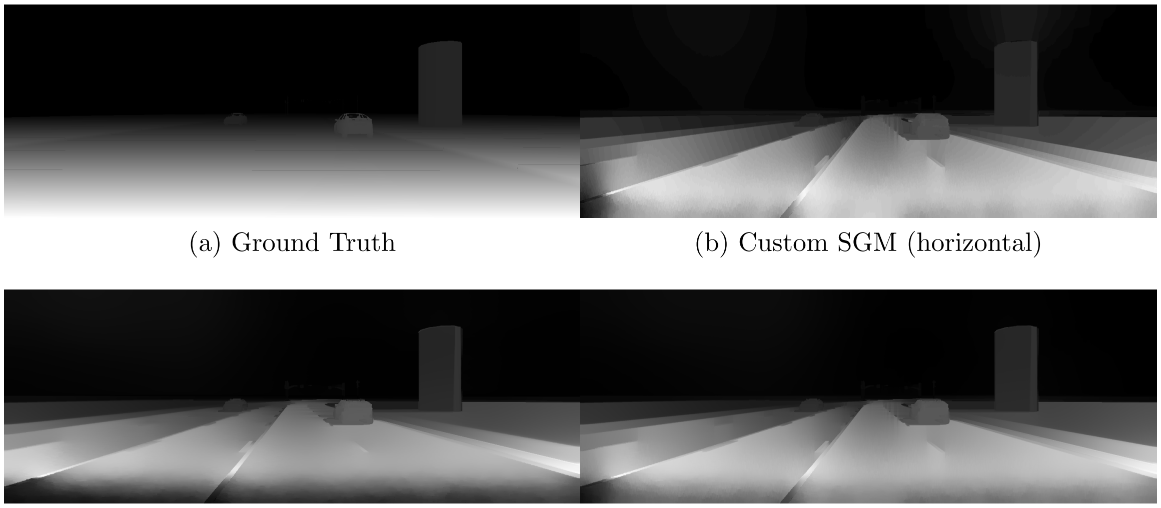
### 5.3.1 Disparity Fusion

Since the horizontal and vertical baselines are the same length of 0.5m, the disparity fusion equation becomes a sum of disparities.

$$D_{fused}(x, y) = D_1(x, y) + D_2(x, y) \tag{5.2}$$

$D_1$ and $D_2$ are the calculated disparities from the horizontal baseline systems and vertical baseline systems respectively. Note that this equation makes the disparities in the fused disparity map in reference to a system with a baseline of 1m rather than the 0.5m baseline of the two input disparity maps.

The grayscale representation of the traditional configurations and multi-axis disparity fused disparity maps do not show significant differences. The main thing to note is that along the bottom of the vertical disparity map is a dark region which does not match the ground truth grayscale representation. This region is not visible in the vertical camera pair which means that it is impossible to match pixels to the reference


(a) Ground Truth (b) Custom SGM (horizontal)

(c) Custom SGM (vertical) (d) Disparity Fusion

**Figure 5.31: Comparison between traditional stereo configuration disparity maps and the multi-axis disparity fusion disparity map**

image. The actual values of the disparities in this region are best guesses but should be regarded as invalid because it is impossible to determine correct disparities. This same problem is apparent in the horizontal disparity map to the left although the region is not as noticeable. The Custom SGM matching algorithm could be implemented in such a way that these regions are invalidated, like in the BM and SGBM algorithms. These regions are only ignored during the calculation of performance metrics. Invalid regions from both the horizontal and vertical maps appear in the disparity fusion map.

Similar to the other disparity fusion maps, the performance of the multi-axis disparity fusion map falls in-between the performance of the two individual disparity maps when looking at the MAE over various 10m intervals (Figure 5.32). One interesting aspect is that the vertical baseline performs better as the distance from the camera increases in comparison to the horizontal baseline system even though the actual baseline distances are equal. This is likely because the vertical camera has a higher perspective than the cameras along the horizontal axis.

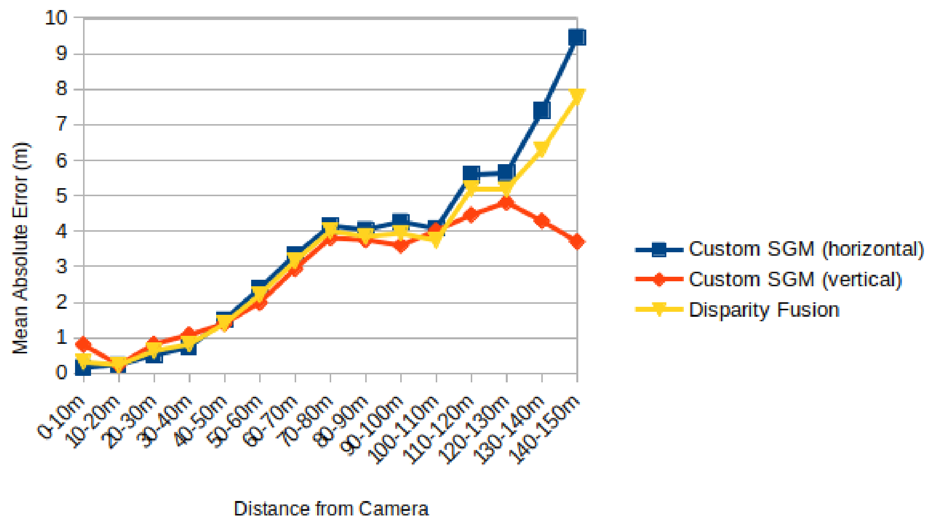


**Figure 5.32: Comparison of MAE between disparity fusion and traditional stereo in a multi-axis configuration at various 10m intervals**

The same trend occurs for the disparity fusion map around the regions of the two vehicles as well (Figure 5.33). The vertical stereo configuration has a significantly higher BMPRE than the horizontal axis configuration for the two vehicles.

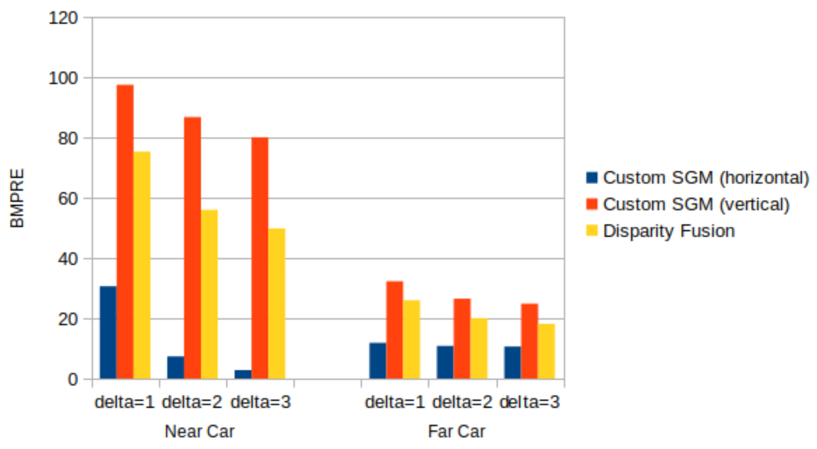


**Figure 5.33: Comparison of BMPRE between disparity fusion and traditional stereo in a multi-axis configuration of the near and far vehicles**

Even though the baselines of the vertical and horizontal camera configurations are equal, there is a difference in the ability to accurately perceive the depth of objects. This was expected because different orientations of the stereo camera systems were expected to detect structures in images differently. It is interesting that a horizontal stereo configuration is more accurate than a vertical configuration for detecting depth of vehicles.

### 5.3.2 Cost Fusion

Using the cost fusion method of fusion, the grayscale representations of the disparity maps show little difference to the traditional stereo configurations (shown in Appendix A). The plot of MAE over a range of distances shows a similar story to cost fusion for multi-baseline systems, where the cost fusion methods follow or have an error lower than the best performing traditional stereo configuration.

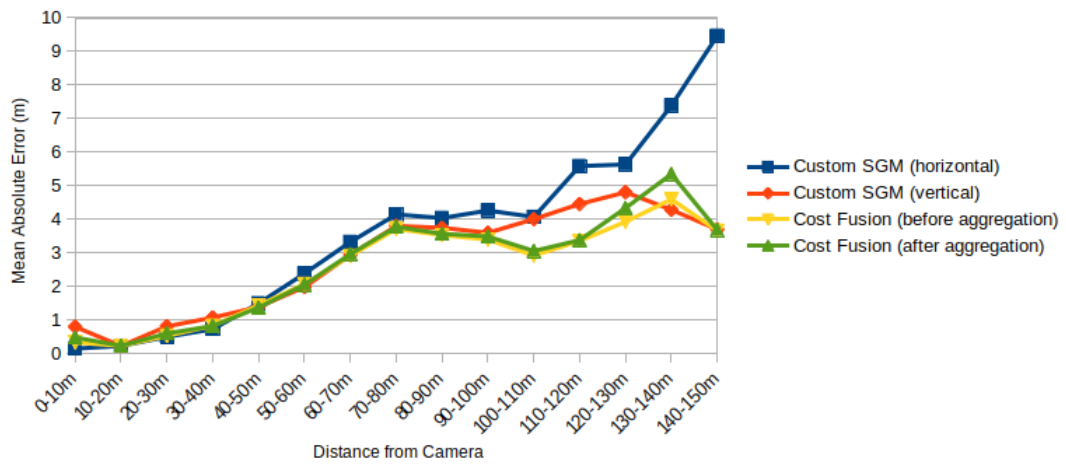


**Figure 5.34: Comparison of MAE between cost fusion and traditional stereo in a multi-axis configuration at various 10m intervals**

Looking at the two vehicles, rather than cost fusion having a lower BMPRE than the best traditional stereo configuration in the multi-baseline configuration, the BMPRE plot for multi-axis cost fusion looks more similar to disparity fusion. The cost fusion methods fall at about the midpoint between the two traditional configurations for each distance. This indicates that multi-axis cost fusion may not be an effective method for triple-camera depth estimation. Additionally, the error for performing fusion after

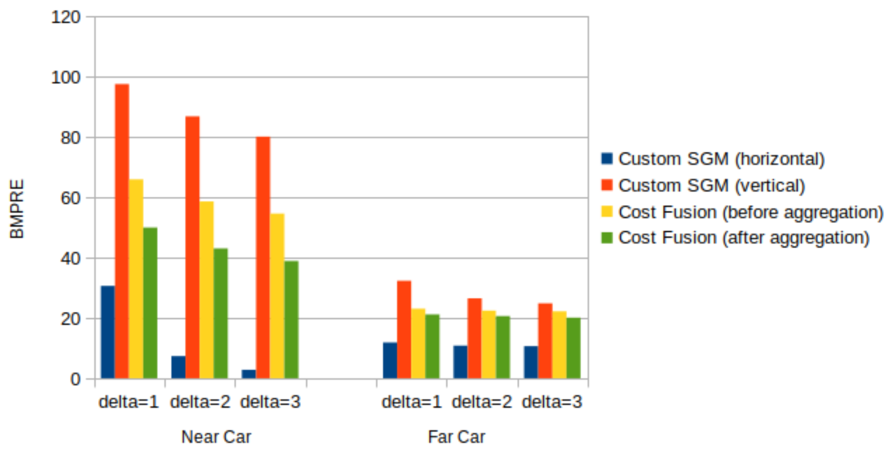


**Figure 5.35: Comparison of BMPRE between cost fusion and traditional stereo in a multi-axis configuration of the near and far vehicles**

cost aggregation is lower than when performing fusion before cost aggregation. This large of a difference between cost fusion before and after aggregation was not seen in multi-baseline cost fusion.

Comparing tradition stereo to the fusion methods for the multi-axis configuration, overall, both cost fusion methods perform better across all distances. The decreased

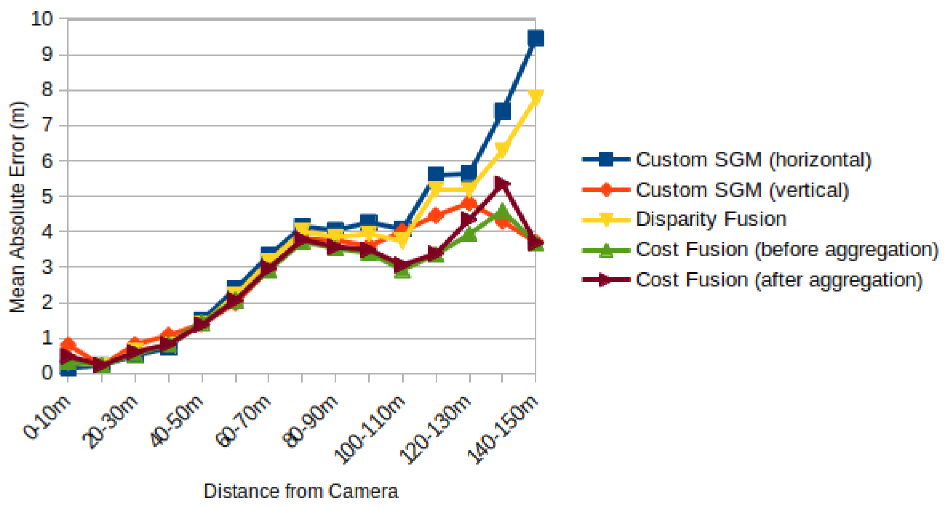


**Figure 5.36: Comparison of MAE between fusion methods and traditional stereo in multi-axis camera configurations at various 10m intervals**

error of the traditional vertical camera system over the horizontal system at large distances seems to have contributed to improved accuracy for the fusion methods.

In contrast, examining the areas in the depth maps where the vehicles are located, the cost fusion methods have a much higher BMPRE than the Custom SGM algorithm in the horizontal configuration. This seems to be caused by a high BMPRE from the Custom SGM algorithm in the vertical stereo configuration.

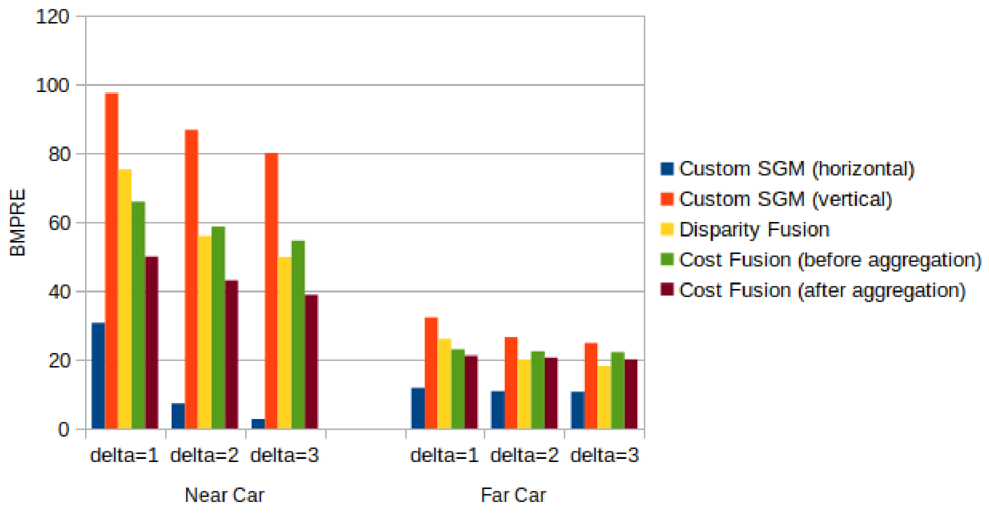


**Figure 5.37: Comparison of BMPRE between disparity fusion, cost fusion, and traditional stereo in multi-axis camera configurations of the near and far vehicles**

Overall, a multi-axis stereo configuration does benefit from the unique perspective of the vertical camera at long ranges, but does not seem to improve the depth accuracy for shorter distances where important objects in the environment are located.

## 5.4 Multi-Baseline vs Multi-Axis

Comparing the multi-baseline and multi-axis triple camera configurations shows similar relationships to what was discovered in earlier sections. Since both multi-baseline and multi-axis configurations share a stereo camera pair, most of the performance difference between the two configurations depends on the performance of the other stereo pair. This is especially evident with the disparity fusion method whose performance was essentially the mean of the performance of the two individual stereo pairs.

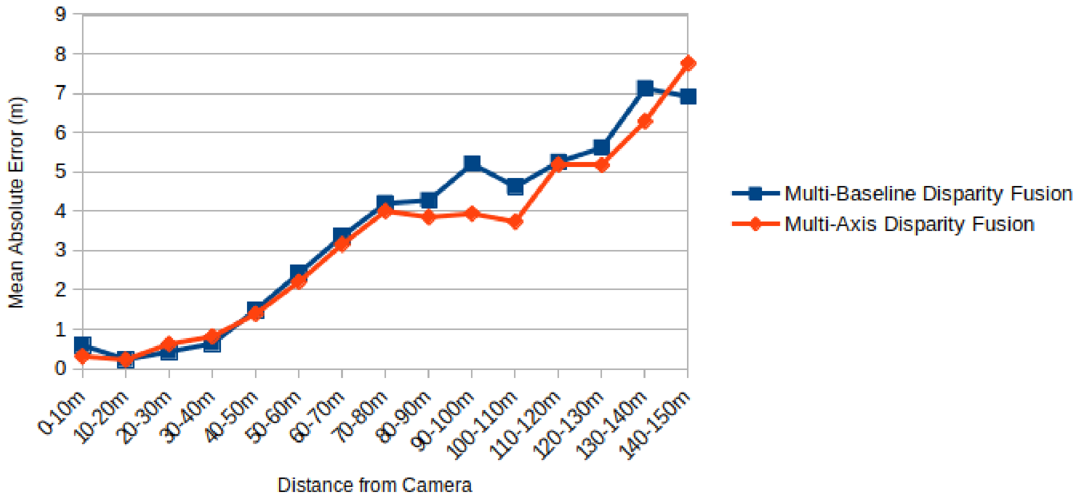


**Figure 5.38: Comparison of MAE for disparity fusion disparity maps between multi-axis and multi-axis camera configurations at various 10m intervals**

Multi-axis disparity fusion has a lower MAE at distances greater than about 70m which corresponds to the improved performance of the vertical stereo pair in comparison to the wide baseline stereo pair at the larger distances.

Similarly, when examining the vehicles in the scene, the vertical stereo configuration had a higher BMPRE than the wide baseline configuration which resulted in a larger BMPRE for the multi-axis disparity fusion than the multi-baseline disparity fusion

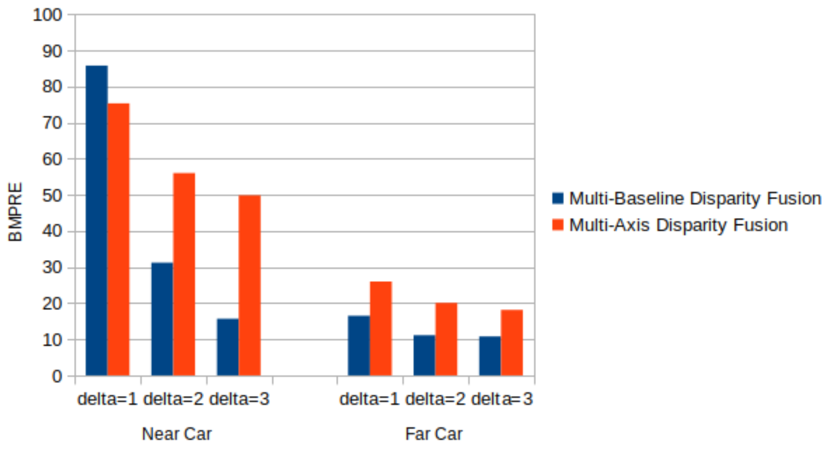


**Figure 5.39: Comparison of BMPRE for disparity fusion disparity maps between multi-axis and multi-axis camera configurations of the near and far vehicles**

(Figure 5.39). The areas where the individual disparity maps had low error, the resulting disparity fused maps also had low error.

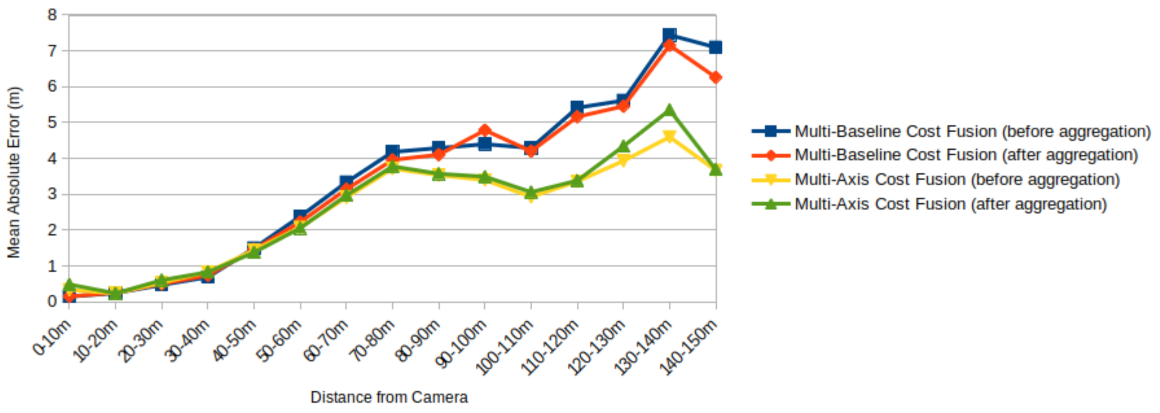


**Figure 5.40: Comparison of MAE for cost fusion between multi-axis and multi-axis camera configurations at various 10m intervals**

This same relationship also occurs even in the cost fusion disparity maps. The multi-axis and multi-baseline cost fusion methods have relatively the same MAE for distances up until about 70m, when the multi-axis cost fusion begins to outperform the multi-baseline cost fusion.

Examining the vehicles, multi-baseline cost fusion has significantly lower BMPRE than the multi-axis cost fusion. Even though cost fusion overall has higher accuracy

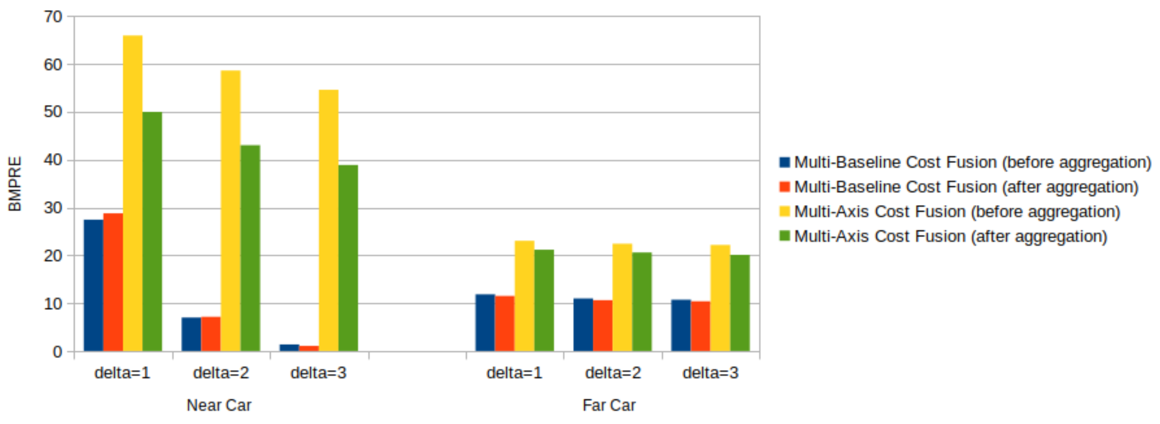


**Figure 5.41: Comparison of BMPRE for cost fusion disparity maps between multi-axis and multi-axis camera configurations of the near and far vehicles**

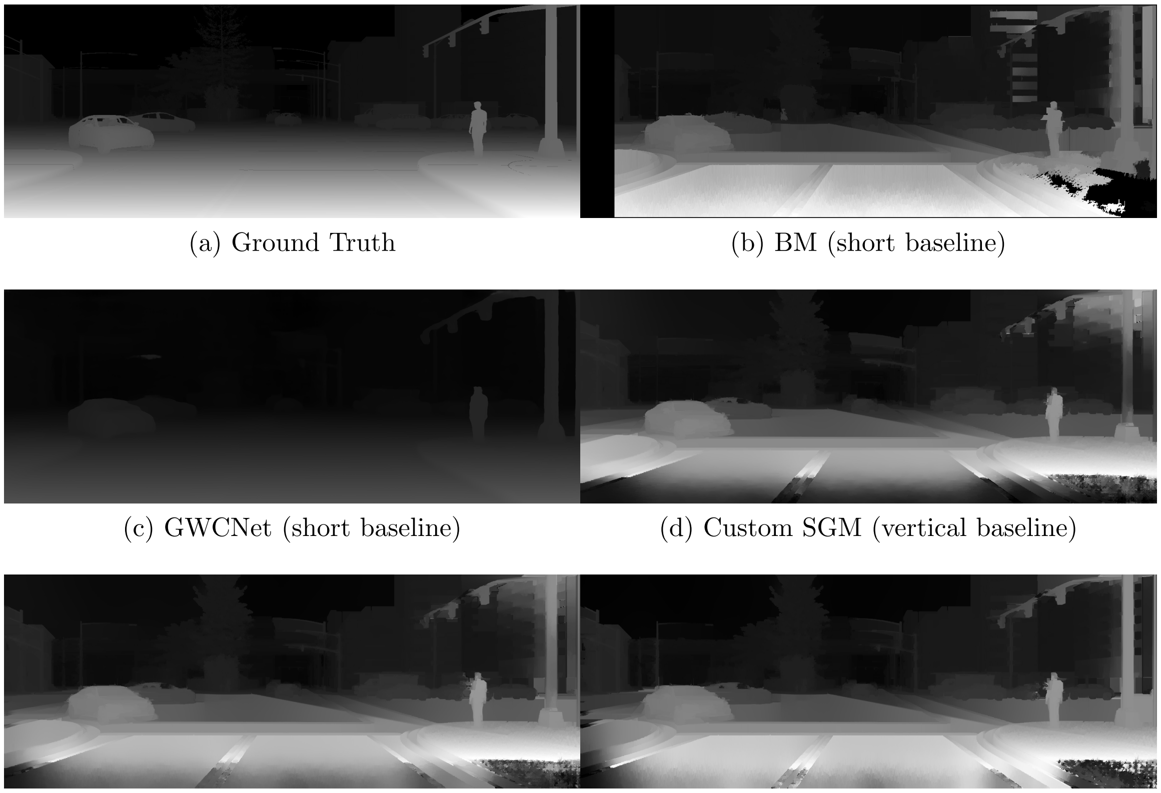
than disparity fusion when comparing algorithms in a single configuration, there still is a relationship between the accuracy of the individual stereo pair and the accuracy of cost fusion.

Overall, the fusion methods in the multi-axis camera configuration have lower error across most of the depth range than fusion methods in the multi-baseline camera configuration. Looking at just the vehicles in the scene, the opposite is true.

## 5.5 City Environment Dataset

In the next comparison between all the algorithms and camera configurations, metrics were computed for the depth and disparity maps (excluding invalid regions) generated by each algorithm and camera configuration using images from the CARLA city environment. Additionally, a selection of the grayscale representations of disparity maps generated by the algorithms of the CARLA city scene are shown in Figure 5.42.

The characteristics of these grayscale disparity maps are like those displayed in previous sections. Even though there are significantly more objects and texture in this



(a) Ground Truth (b) BM (short baseline)

(c) GWCNet (short baseline) (d) Custom SGM (vertical baseline)

(e) Disparity Fusion (multi-axis) (f) Cost Fusion after aggregation (multi-baseline)

**Figure 5.42: Disparity maps generated by a selection of algorithms and camera configurations of the CARLA city environment**

environment, the smoothness of the GWCNet algorithm over the traditional algorithm is still obvious. For the closest distances in the disparity maps, specifically the bottom right corner, it is evident that the traditional algorithms and fusion algorithms struggle to match pixels. At the same time, the structures of the objects and features in the background of the GWCNet are not as clear as in the other algorithms. This likely means that the trend of the CNN-based algorithms performing better at closer distances but worse at longer distances remains true in the new environment.

Examining the BMP metrics across the entire disparity map for all algorithms in Table 5.3, a couple things are noticeable. Across the short, wide, and vertical baseline stereo configurations the algorithm GWCNet has by far the lowest BMP. The BMP metrics for GWCNet in the short baseline configuration are 68% ($\delta = 3px$), 66%

($\delta = 2px$), and 30% ($\delta = 1px$) lower than the next best algorithm in the short baseline configuration and 33% ($\delta = 3px$), 58% ($\delta = 2px$), and 40% ($\delta = 1px$) lower than the next best algorithm in the wide baseline configuration. As one of the top algorithms on the KITTI leaderboards this result was unsurprising.

**Table 5.3: BMP values of disparity maps generated by different algorithms using images of the CARLA City Environment**

| Algorithm | Camera Configuration | BMP | | |
|---|---|---|---|---|
| | | $\delta$=3px | $\delta$=2px | $\delta$=1px |
| BM | short baseline | 0.289 | 0.353 | 0.461 |
| SGBM | short baseline | 0.206 | 0.271 | 0.376 |
| PSMNet | short baseline | 0.142 | 0.380 | 0.777 |
| GWCNet | short baseline | **0.045** | **0.092** | **0.264** |
| Custom SGM | short baseline | 0.233 | 0.302 | 0.422 |
| BM | wide baseline | 0.466 | 0.507 | 0.565 |
| SGBM | wide baseline | 0.421 | 0.472 | 0.541 |
| PSMNet | wide baseline | 0.145 | 0.341 | 0.728 |
| GWCNet | wide baseline | **0.096** | **0.143** | **0.322** |
| Custom SGM | wide baseline | 0.498 | 0.551 | 0.631 |
| Custom SGM | vertical baseline | 0.253 | 0.314 | 0.410 |
| Disparity Fusion (Custom SGM) | multi-baseline | 0.512 | 0.565 | 0.635 |
| Cost Fusion (before aggregation) | multi-baseline | 0.219 | 0.287 | **0.406** |
| Cost Fusion (after aggregation) | multi-baseline | **0.216** | **0.287** | 0.407 |
| Disparity Fusion (Custom SGM) | multi-axis | 0.375 | 0.447 | 0.555 |
| Cost Fusion (before aggregation) | multi-axis | **0.203** | **0.262** | **0.368** |
| Cost Fusion (after aggregation) | multi-axis | 0.206 | 0.266 | 0.373 |
| Disparity Fusion (GWCNet and Custom SGM) | multi-baseline | 0.484 | 0.556 | 0.690 |

Examining the performance of the fusion methods for the metric BMP, cost fusion in a triple-camera configuration is an improvement over all two-camera configurations for the Custom SGM algorithm. Cost fusion after cost aggregation in a multi-baseline camera configuration had BMP metrics 7.3% ($\delta = 3px$), 4.9% ($\delta = 2px$), and 3.6% ($\delta = 1px$) lower than the Custom SGM algorithm in the short and wide baseline camera configurations. Similarly, cost fusion before cost aggregation in a multi-axis camera configuration had BMP metrics 13% ($\delta = 3px$), 13% ($\delta = 2px$), and 10%

($\delta = 1px$) lower than the Custom SGM algorithm in the short and vertical baseline camera configurations.

In contrast, the disparity fusion methods do not show any decrease in BMP over the Custom SGM algorithm in any of the traditional stereo configurations even when a Custom SGM depth map was combined with the best performing depth map (GWC-Net in a short baseline configuration).

**Table 5.4: BMPRE values of disparity maps generated by different algorithms using images of the CARLA City Environment**

| Algorithm | Camera Configuration | BMPRE | | |
|---|---|---|---|---|
| | | $\delta$=3px | $\delta$=2px | $\delta$=1px |
| BM | short baseline | 3,106,980 | 3,112,623 | 3,118,804 |
| SGBM | short baseline | 2,427,069 | 2,432,517 | 2,438,839 |
| PSMNet | short baseline | 85,898 | 127,559 | 154,422 |
| GWCNet | short baseline | **31,746** | **44,901** | **64,607** |
| Custom SGM | short baseline | 66,837 | 73,234 | 81,411 |
| BM | wide baseline | 1,745,200 | 1,747,728 | 1,749,994 |
| SGBM | wide baseline | 1,311,484 | 1,314,070 | 1,316,818 |
| PSMNet | wide baseline | 51,566 | 68,178 | 86,588 |
| GWCNet | wide baseline | **33,734** | **36,793** | **43,569** |
| Custom SGM | wide baseline | 134,390 | 138,367 | 142,551 |
| Custom SGM | vertical baseline | 87,760 | 93,340 | 98,447 |
| Disparity Fusion (Custom SGM) | multi-baseline | 111,755 | 115,752 | 119,196 |
| Cost Fusion (before aggregation) | multi-baseline | 48,707 | 55,413 | 63,324 |
| Cost Fusion (after aggregation) | multi-baseline | **47,154** | **54,059** | **62,176** |
| Disparity Fusion (Custom SGM) | multi-axis | 82,914 | 87,335 | 91,363 |
| Cost Fusion (before aggregation) | multi-axis | **38,946** | **43,934** | **50,179** |
| Cost Fusion (after aggregation) | multi-axis | 41,349 | 46,390 | 52,797 |
| Disparity Fusion (GWCNet and Custom SGM) | multi-baseline | 100,914 | 107,209 | 115,572 |

Examining the BMPRE metric confirms that GWCNet is the best performing algorithm in both the short and wide baseline configurations. The BMPRE metrics for GWCNet in the short baseline configuration are 53% ($\delta = 3px$), 39% ($\delta = 2px$), and 21% ($\delta = 1px$) lower than the next best algorithm in the short baseline configuration and 35% ($\delta = 3px$), 46% ($\delta = 2px$), and 50% ($\delta = 1px$) lower than the next best algorithm in the wide baseline configuration. Additionally, contrary to the relatively low BMP metrics of the BM and SGBM algorithms, the BMPRE metrics for these algorithms in the short and wide baseline configurations are at minimum an order of magnitude larger than the other algorithms. This means that when the BM and SGBM algorithms incorrectly match a pixel, they miss by a large amount in comparison to the other algorithms.

The cost fusion algorithms have the next lowest BMPRE metrics across the entire disparity maps. These algorithms show improvement in lowering BMPRE over the short, wide, and vertical baseline Custom SGM algorithms. Cost fusion after cost aggregation in a multi-baseline camera configuration had BMPRE metrics 29% ($\delta = 3px$), 26% ($\delta = 2px$), and 24% ($\delta = 1px$) lower than the Custom SGM algorithm in the short and wide baseline camera configurations. Similarly, cost fusion before cost aggregation in a multi-axis camera configuration had BMP metrics 42% ($\delta = 3px$), 40% ($\delta = 2px$), and 38% ($\delta = 1px$) lower than the Custom SGM algorithm in the short and vertical baseline camera configurations. There are greater improvements in the BMPRE metric for cost fusion than the BMP metric because a decrease in BMP directly causes a decrease in BMPRE while not all BMPRE decreases cause decreases in BMP.

Similarly to BMP, disparity fusion methods do not show any decrease in BMPRE over the Custom SGM algorithm in any of the traditional stereo configurations. The disparity fusion methods have about double the BMPRE than the cost fusion methods.

**Table 5.5: MAE and MSE values of depth maps generated by different algorithms using images of the CARLA City Environment**

| Algorithm | Camera Configuration | MAE (m) | MSE (m) |
|---|---|---|---|
| BM | short baseline | 5.804 | 12.378 |
| SGBM | short baseline | 4.740 | 15.800 |
| PSMNet | short baseline | 10.783 | 18.778 |
| GWCNet | short baseline | 6.395 | 13.533 |
| Custom SGM | short baseline | **4.863** | **10.391** |
| BM | wide baseline | 6.753 | 13.210 |
| SGBM | wide baseline | 6.718 | 18.291 |
| PSMNet | wide baseline | 6.495 | 12.197 |
| GWCNet | wide baseline | **3.835** | **9.296** |
| Custom SGM | wide baseline | 9.671 | 16.736 |
| Custom SGM | vertical baseline | 5.037 | 10.580 |
| Disparity Fusion (Custom SGM) | multi-baseline | 5.992 | 10.113 |
| Cost Fusion (before aggregation) | multi-baseline | 4.364 | 9.515 |
| Cost Fusion (after aggregation) | multi-baseline | **4.148** | **9.233** |
| Disparity Fusion (Custom SGM) | multi-axis | 4.943 | 10.070 |
| Cost Fusion (before aggregation) | multi-axis | 3.764 | 7.975 |
| Cost Fusion (after aggregation) | multi-axis | **3.735** | **7.966** |
| Disparity Fusion (GWCNet and Custom SGM) | multi-baseline | 6.494 | 10.433 |

Looking across the depth maps generated for each algorithm, the cost fusion method has lower depth error metrics than most of the algorithms in any configurations. This was not the case with the disparity error metrics where the GWCNet and PSMNet significantly outperformed the other algorithms.

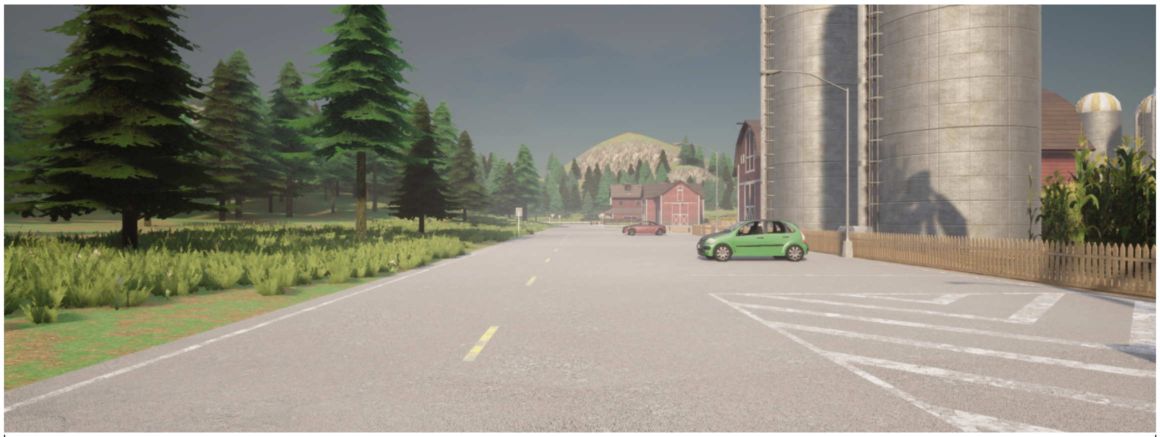
The GWCNet algorithm in the wide baseline configuration still has a low MAE and MSE in comparison to the other algorithms, but the multi-axis cost fusion after aggregation algorithm has the lowest depth error overall. The multi-axis cost fusion after aggregation is only slightly lower than GWCNet and has a MAE 23% lower and MSE 23% lower than the Custom SGM algorithm in the short baseline configuration. The GWCNet algorithm not performing as well with depth metrics is likely because GWCNet struggles with matching in areas furthest away from the camera and a pixel difference in a disparity estimation of a faraway object has a much larger impact on the depth error than a pixel difference in a disparity estimation of a close object (because of the inverse relationship between depth and disparity). Since all the algorithms perform better than GWCNet at longer distances, the difference in error between GWCNet and the other algorithms was decreased.

In summary, the GWCNet algorithm overall had the lowest error across all the error metrics for the CARLA city environment. Cost fusion methods performed comparably to GWCNet, and saw a decrease in error over the Custom SGM algorithm in all stereo configurations. The disparity fusion methods, similar to previous analysis, had error rates fall between the performance of the two input disparity maps.

## 5.6 Eight Additional Datasets

In order to validate the depth and disparity map accuracy improvements of the fusion methods across a wider range of data, the Custom SGM in a short baseline configuration, Multi-axis Disparity Fusion, Multi-axis Cost Fusion before aggregation, and Multi-baseline Cost Fusion before aggregation algorithms were run on an additional 8 datasets generated from the CARLA simulator. Depth metrics MAE and MSE, and disparity metrics BMP ($\delta$=1px) and BMPRE ($\delta$=1px) were computed from the generated depth maps and disparity maps respectively. These metrics and a sample image from each dataset are shown below in Figures 5.43-5.46.

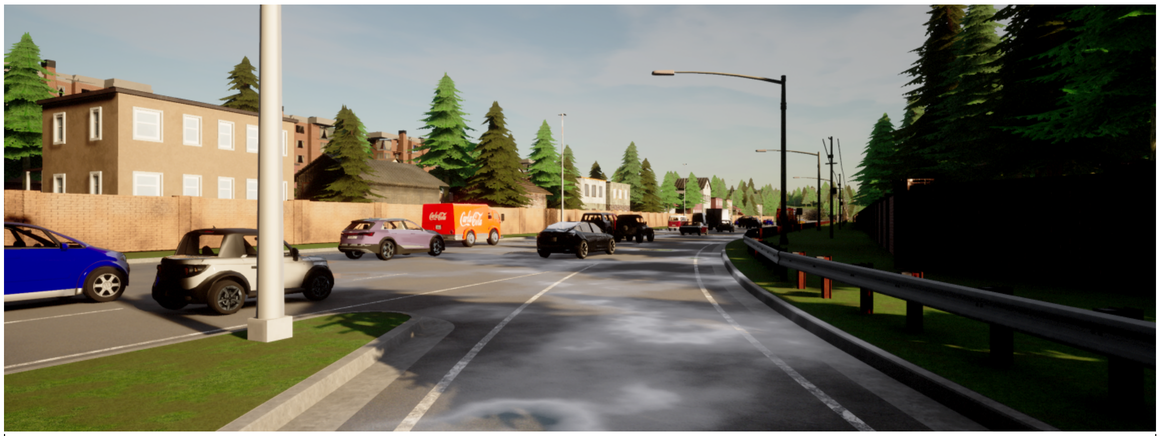
| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 3.36 | 11.10 | 0.397 | 49,609 |
| Multi-axis Disparity Fusion | 3.05 | 10.51 | 0.395 | 44,433 |
| Multi-axis Cost Fusion | 2.77 | 10.13 | 0.355 | 37,867 |
| Multi-baseline Cost Fusion | 3.27 | 10.29 | 0.360 | 49,000 |



| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 6.20 | 21.19 | 0.487 | 88,852 |
| Multi-axis Disparity Fusion | 5.93 | 20.08 | 0.484 | 89,342 |
| Multi-axis Cost Fusion | 5.54 | 18.87 | 0.461 | 84,182 |
| Multi-baseline Cost Fusion | 5.78 | 19.21 | 0.482 | 86,317 |

**Figure 5.43: Error metrics of depth and disparity maps generated by fusion algorithms of a CARLA City Street Environment and a CARLA Rural Farm Environment**

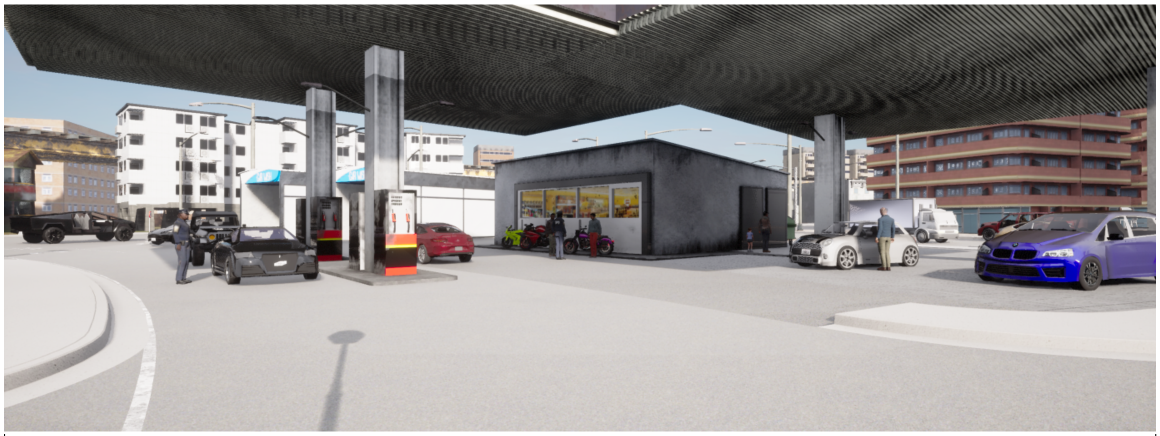
| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 5.93 | 18.35 | 0.429 | 94,625 |
| Multi-axis Disparity Fusion | 5.74 | 17.74 | 0.421 | 89,643 |
| Multi-axis Cost Fusion | 5.68 | 17.70 | 0.411 | 88,581 |
| Multi-baseline Cost Fusion | 5.94 | 18.08 | 0.420 | 90,018 |



| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 4.18 | 15.34 | 0.574 | 70,612 |
| Multi-axis Disparity Fusion | 3.99 | 14.79 | 0.570 | 66,575 |
| Multi-axis Cost Fusion | 3.49 | 14.26 | 0.500 | 53,528 |
| Multi-baseline Cost Fusion | 3.79 | 14.80 | 0.555 | 58,720 |

**Figure 5.44: Error metrics of depth and disparity maps generated by fusion algorithms of a CARLA Forest Road Environment and a CARLA Freeway Environment**

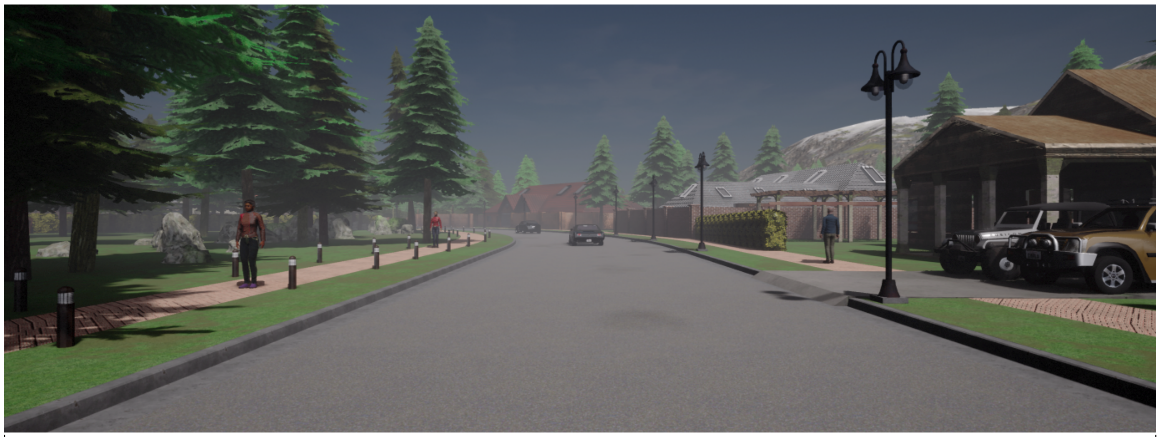
| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 6.51 | 23.85 | 0.628 | 122,785 |
| Multi-axis Disparity Fusion | 6.37 | 23.75 | 0.614 | 115,547 |
| Multi-axis Cost Fusion | 6.05 | 23.22 | 0.576 | 108,574 |
| Multi-baseline Cost Fusion | 6.41 | 23.74 | 0.626 | 120,899 |



| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 5.04 | 13.51 | 0.563 | 98,953 |
| Multi-axis Disparity Fusion | 4.97 | 12.19 | 0.620 | 108,839 |
| Multi-axis Cost Fusion | 3.54 | 11.29 | 0.506 | 79,715 |
| Multi-baseline Cost Fusion | 5.22 | 13.80 | 0.555 | 90,844 |

**Figure 5.45: Error metrics of depth and disparity maps generated by fusion algorithms of a CARLA Rain Environment and a CARLA Gas Station Environment**

| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 7.95 | 20.66 | 0.489 | 64,158 |
| Multi-axis Disparity Fusion | 7.82 | 19.19 | 0.488 | 63,138 |
| Multi-axis Cost Fusion | 7.88 | 18.89 | 0.458 | 58,674 |
| Multi-baseline Cost Fusion | 7.96 | 19.20 | 0.487 | 63.858 |



| Algorithm | MAE | MSE | BMP | BMPRE |
|---|---|---|---|---|
| Custom SGM | 4.86 | 17.35 | 0.589 | 86,002 |
| Multi-axis Disparity Fusion | 4.83 | 17.30 | 0.591 | 89,088 |
| Multi-axis Cost Fusion | 4.69 | 17.10 | 0.572 | 85,322 |
| Multi-baseline Cost Fusion | 4.81 | 17.32 | 0.587 | 86,185 |

**Figure 5.46: Error metrics of depth and disparity maps generated by fusion algorithms of a CARLA Parking Lot Environment and a CARLA Neighborhood Environment**

Note that for all these 8 datsets, both Disparity Fusion and Cost Fusion algorithms in any camera configuration showed depth and disparity estimation improvements over just the traditional Custom SGM algorithm in a short baseline camera configuration. Additionally, Multi-axis Cost Fusion had the lowest error metrics among the fusion algorithms.

## 5.7 Computational Performance

Table 5.6 shows the runtime and peak memory usage of various depth estimation algorithms during the processing of a single set of images. These algorithms were implemented in Python 3.8 and ran on an Intel Core i7 8565U processor with 16GB of RAM.

**Table 5.6: Computational Performance Comparison of the Custom SGM algorithm and fusion algorithms**

| Algorithm | Runtime (sec) | Peak Memory Usage (MB) |
|---|---|---|
| Custom SGM | 391.78 | 3,818.4 |
| Disparity Fusion | 1,286.37 | 9,519.6 |
| Cost Fusion (before aggregation) | 640.75 | 4,844.6 |
| Cost Fusion (after aggregation) | 1,022.19 | 8,472.3 |

Reviewing the runtime statistics, as expected, all fusion algorithms had longer runtimes than the Custom SGM algorithm due to the added processing needed for the additional image. The longest runtime was the disparity fusion algorithm, which ran for 3.3 times longer than the Custom SGM algorithm. The runtime being over twice as long was expected because disparity fusion computes the Custom SGM algorithm twice and then calculates a weighted mean between corresponding elements in the produced disparity maps. In contrast, the cost fusion algorithms, which fuse information from each perspective much earlier in the process and avoid computing steps in the algorithm more than once, have runtimes that are lower than disparity fusion. Cost Fusion before aggregation and Cost Fusion after aggregation had runtimes 1.6 and 2.6 times longer than the Custom SGM algorithm respectively.

Cost Fusion methods only differ in where cost fusion is performed (before and after cost aggregation). The almost 400 second longer runtime of cost fusion after cost aggregation than cost fusion before cost aggregation is caused by needing to compute cost aggregation an additional time than cost fusion before cost aggregation.

Peak memory usage of each algorithm follows a similar pattern with disparity fusion peaking at a value 2.5 times that of the Custom SGM algorithm while cost fusion before aggregation and cost fusion after aggregation peaked at values 1.3 and 2.2 times larger than Custom SGM respectively.

In summary, disparity fusion performs the greatest number of duplicate steps which caused the algorithm to have the highest runtime and largest peak memory usage. Cost fusion before aggregation had the least number of duplicate steps to the Custom SGM algorithm which resulted in the lowest runtime and smallest peak memory usage in comparison to the other fusion algorithms. Additionally, the large difference in runtime and peak memory usage between the cost fusion algorithms demonstrates that cost aggregation is the most performance and memory intensive step of the Custom SGM algorithm.

Chapter 6

# CONCLUSION

The goal of this thesis was to determine if a triple-camera configuration would be able to improve the accuracy of image-based depth estimation. Since there were no existing datasets that fulfilled all the requirements for this research, a dataset was built from images and ground truth data captured of environments created in the open-source simulator CARLA. The images in the dataset were fed into different stereo matching algorithms which produced estimated depth maps of the environment. Using ground truth data, each depth map was evaluated for accuracy by computing four metrics, BMP, BMPRE, MAE, and MSE, over the entire depth map and over specific regions in the depth map. These metrics were used to compare the accuracy of the depth maps to each other.

## 6.1 Findings

This thesis examined the accuracy of the algorithms BM, SGBM, GWCNet, PSMNet, and Custom SGM in short and wide baseline camera configurations to compare to the accuracy of the disparity fusion and cost fusion methods in multi-axis and multi-baseline camera configurations. The following are some of the discoveries from this research.

The accuracy of the depth maps in a wide baseline configuration was lower than the accuracy of depth maps in a short baseline configuration, across all algorithms. The purpose of the wider baseline was to improve the accuracy of depth estimation for objects located far away because baseline length has an inverse relationship with depth error. There is an accuracy improvement for objects located more than 100m away, but a majority of the important objects are located closer than 100m, which means that overall, the depth maps generated from the wide baseline configuration have worse accuracy. As noted previously, a larger baseline can have more occlusions and discontinuities in the resulting images which makes matching of pixels more difficult. The wide baseline length chosen in this thesis was 1 meter which may have been too large for this application. A deeper investigation into how varying baseline length impacts accuracy, especially for the multi-baseline and multi-axis camera configurations, is an area I see future research opportunities in.

GWCNet was the algorithm with the highest depth map accuracy overall, especially for the BMP and BMPRE metrics. Both CNN-based algorithms (PSMNet and GWC-Net) performed significantly better around non-textured regions than the traditional algorithms which was most obvious in the visual representations of the depth maps. This was why the CNN-based algorithms had a higher accuracy than the traditional methods overall. These algorithms seem to understand general features that are contained in images captured from the perspective of a vehicle driving (like the road) and use this knowledge to match pixels that may not be very similar because of their global understanding. Because GWCNet was trained on real images from the KITTI dataset, I would expect the accuracy to be even higher if trained using images from the CARLA environment.

Disparity fusion performed as expected, with the accuracy of the fused depth map falling at about the average of the two input depth maps. The idea with disparity fusion was that high accuracy areas in one depth map could be combined with high accuracy areas of another depth map. However, since there is no way to determine a highly accurate region (without ground truth) and the combination is a weighted mean, disparity fusion also combines the low accuracy regions into the resulting depth map. Across the entire depth map, the accuracy may be more consistent, but there seems to be no improvement in accuracy overall with disparity fusion.

Both cost fusion in the multi-axis and multi-baseline camera configurations had improved accuracy over the Custom SGM algorithm in the traditional stereo configurations. This is the most significant result of this thesis because it shows that accuracy can be improved by adding an additional camera. Both the multi-axis and multi-baseline configurations saw an accuracy improvement which indicates that it is more than just a specific configuration that caused the performance improvement. However, the accuracy metrics of cost fusion still fall slightly behind the performance of the GWCNet algorithm. Qualitatively, the visual representations of the GWCNet depth maps also appear much more like the real world than the cost fusion depth maps. So, although the triple camera cost fusion Custom SGM algorithm shows an improvement in accuracy, it is still behind the cutting-edge depth estimators (Which are mostly CNN algorithms). Since there are promising results by adding an additional camera to traditional stereo matching algorithms, it would be interesting to see if an additional camera would improve the accuracy of a CNN-based depth estimation algorithm.

The final discovery was that the difference in accuracy by performing cost fusion before or after cost aggregation was very small. This is important because cost fusion after cost aggregation must compute cost aggregation twice while cost fusion before cost aggregation only computes cost aggregation once. Cost aggregation is the most performance intensive part of the Custom SGM algorithm and in order to make triple camera depth estimation algorithms feasible in an autonomous vehicle application their performance must be as good or better than existing algorithms. The current implementation of Custom SGM (including cost fusion) is not optimized for performance and further work is necessary to determine if multi-axis or multi-baseline cost fusion can be optimized enough to run on real-time video.

## 6.2 Future Work

Since the idea for this research stemmed from papers that were examining methods to improve the performance of the 3D object detection and localization algorithms, the next step is plugging the depth estimation algorithms into the 3D object detection pipelines to evaluate and compare their performance to each other. In order to have ground truth data to evaluate 3D object detection performance, the type and bounding-box location of objects in the environment will also need to be captured when generating images, a feature that is already available in the CARLA simulator.

Another algorithm/camera configuration that could be analyzed is the combination of the multi-axis and multi-baseline systems into a 4-camera system. A modification of the cost fusion algorithm that could fuse costs from three sources rather than two would be relatively simple. Additionally, all the necessary images to perform an analysis already exist within the current synthetic dataset.

However, I believe that the most promising research opportunity is designing and training a machine learning depth estimator that uses three images from either a multi-axis or multi-baseline camera configuration to generate a depth map. Most of the current state-of-the-art depth estimators are machine learning algorithms, and this trend does not seem to be diminishing. At this time, I do not know of any research into this area. The biggest issue is that there are no datasets large enough to train and test a triple-camera machine learning depth estimation algorithm. This paper demonstrates that the simulator CARLA could be useful for generating a large set of life-like images at a low cost.

# BIBLIOGRAPHY

[1] 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, 2013.

[2] A. Januzi, "Triple-camera setups for image-based depth estimation," 2020.

[3] R. Fan, L. Wang, M. J. Bocus, and I. Pitas, "Computer stereo vision for autonomous driving," *arXiv preprint arXiv:2012.03194*, 2020.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[5] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer, "Vision and navigation for the carnegie-mellon navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, 1988.

[6] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2018.

[7] D. Coffin, S. Oliver, and J. VerWey, "Building vehicle autonomy: Sensors, semiconductors, software and us competitiveness," 2019.

[8] V. De Silva, J. Roche, and A. Kondoz, "Robust fusion of lidar and wide-angle camera data for autonomous mobile robots," *Sensors*, vol. 18, no. 8, p. 2730, 2018.

[9] H. Luo, C. Pape, and E. Reithmeier, “Scale-aware multi-view reconstruction using an active triple-camera system,” in *Sensors (Basel) 2020 Dec; Volume 20(23).* NCIB, 2020.

[10] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, “Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8445–8453.

[11] J. Lambert, A. Carballo, A. M. Cano, P. Narksri, D. Wong, E. Takeuchi, and K. Takeda, “Performance analysis of 10 models of 3d lidars for automated driving,” *IEEE Access*, vol. 8, pp. 131 699–131 722, 2020.

[12] S. Royo and M. Ballesta-Garcia, “An overview of lidar imaging systems for autonomous vehicles,” *Applied sciences*, vol. 9, no. 19, p. 4093, 2019.

[13] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE conference on computer vision and pattern recognition.* IEEE, 2012, pp. 3354–3361.

[14] “Kitti vision benchmark suite,” http://www.cvlibs.net/datasets/kitti/index.php.

[15] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[16] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, “ST3D: self-training for unsupervised domain adaptation on 3d object detection,” *CoRR*, 2021. [Online]. Available: https://arxiv.org/abs/2103.05346

[17] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, “Se-ssd: Self-ensembling single-stage object detector from point cloud,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 494–14 503.

[18] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, “Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving,” *arXiv preprint arXiv:1906.06310*, 2019.

[19] T. Ewbank, “Efficient and precise stereoscopic vision for humanoid robots,” Ph.D. dissertation, Master’s thesis, University of LIÈGE, 2017.

[20] C. Ricolfe-Viala and A.-J. Sanchez-Salmeron, “Lens distortion models evaluation,” *Applied optics*, vol. 49, no. 30, pp. 5914–5928, 2010.

[21] “Matlab camera calibration,” https://www.mathworks.com/help/vision/camera-calibration.html.

[22] “Opencv camera calibration,” https://docs.opencv.org/4.5.2/dc/dbb/tutorial_py_calibration.html.

[23] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[24] P. Munro and A. P. Gerdelan, “Stereo vision computer depth perception,” *Country United States City University Park Country Code US Post code*, vol. 16802, 2009.

[25] G. Xu and Z. Zhang, *Epipolar geometry in stereo, motion and object recognition: a unified approach.* Springer Science & Business Media, 2013, vol. 6.

[26] M. G. Mozerov and J. Van De Weijer, “Accurate stereo matching by two-step energy minimization,” *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 1153–1163, 2015.

[27] A. C. Bovik, *Handbook of image and video processing.* Academic press, 2010.

[28] H. Hirschmuller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2007.

[29] J.-R. Chang and Y.-S. Chen, “Pyramid stereo matching network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[30] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li, “Group-wise correlation stereo network,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3273–3282.

[31] K. Shen, “Effect of baseline on stereo vision systems,” 2018.

[32] W. Boonsuk, “Investigating the effects of stereo camera baseline on the accuracy of 3d projection for industrial robotic applications,” *International Journal of Engineering Research and Innovation*, vol. 8, no. 2, 2016.

[33] J. Kallwies, T. Engler, B. Forkel, and H.-J. Wuensche, “Triple-sgm: Stereo processing using semi-global matching with cost fusion,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 192–200.

[34] D. G. Bailey, “Sub-pixel estimation of local extrema,” in *Proceeding of Image and Vision Computing New Zealand*, 2003, pp. 414–419.

[35] T. Yan, Y. Gan, Z. Xia, and Q. Zhao, “Segment-based disparity refinement with occlusion handling for stereo matching,” *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 3885–3897, 2019.

[36] W. Wang and C. Zhang, “Local disparity refinement with disparity inheritance,” in *2012 Symposium on Photonics and Optoelectronics*, 2012, pp. 1–4.

[37] J. Kallwies and H.-J. Wuensche, “Effective combination of vertical and horizontal stereo vision,” in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1992–2000.

[38] S. Merrouche, M. Andrić, B. Bondžulić, and D. Bujaković, “Objective image quality measures for disparity maps evaluation,” *Electronics*, vol. 9, no. 10, p. 1625, 2020.

[39] I. Cabezas, V. Padilla, and M. Trujillo, “Bmpre: An error measure for evaluating disparity maps,” in *2012 IEEE 11th International Conference on Signal Processing*, vol. 2. IEEE, 2012, pp. 1051–1055.

[40] Y. Hel-Or and S. Edelman, “A new approach to qualitative stereo,” in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1. IEEE, 1994, pp. 316–320.

[41] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[42] K. Konolige, “Small vision systems: Hardware and implementation,” in *Robotics research*. Springer, 1998, pp. 203–212.

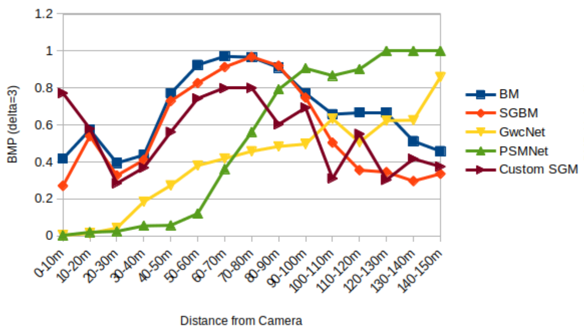
# APPENDICES

# Appendix A

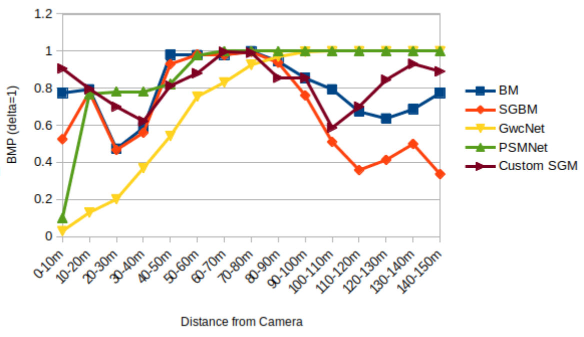
# ADDITIONAL RESULTS

Figures for some additional quantitative and qualitative metrics not shown in the Results chapter can be found in this appendix.

## A.1 Traditional Stereo

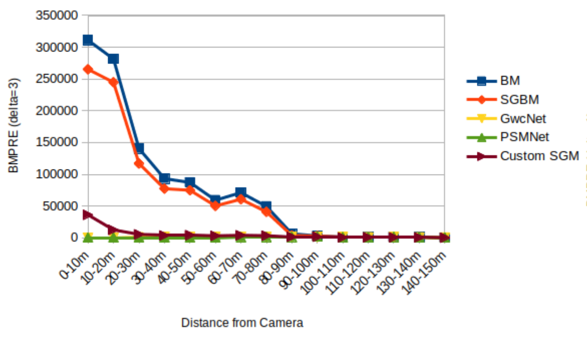


(a) $\delta$=3px

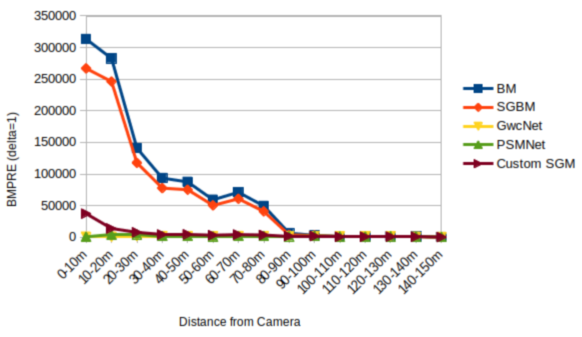


(b) $\delta$=1px

**Figure A.1: BMP values from disparity maps generated by different algorithms at various 10m intervals**

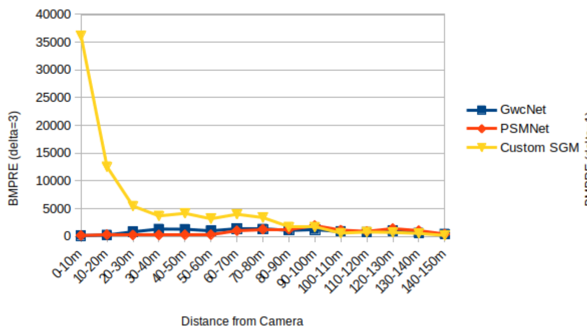


(a) $\delta$=3px

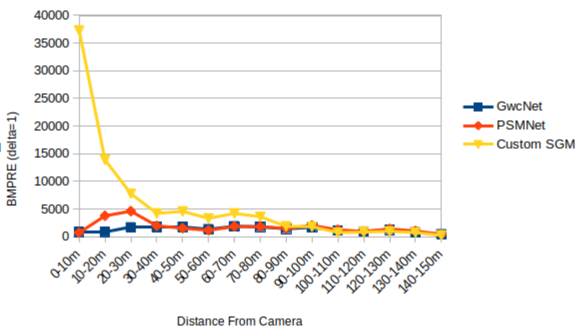


(b) $\delta$=1px

**Figure A.2: BMPRE values from disparity maps generated by different algorithms at various 10m intervals**



(a) $\delta$=3px



(b) $\delta$=1px

**Figure A.3: Comparison of BMPRE values from disparity maps generated by different algorithms in a short baseline camera configuration at various 10m intervals**

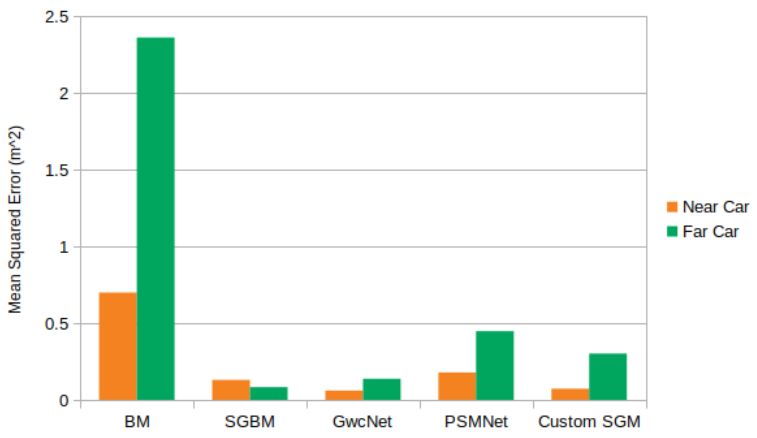


**Figure A.4: Comparison of MSE values from depth maps generated by different algorithms in a short baseline camera configuration of the near and far vehicles**

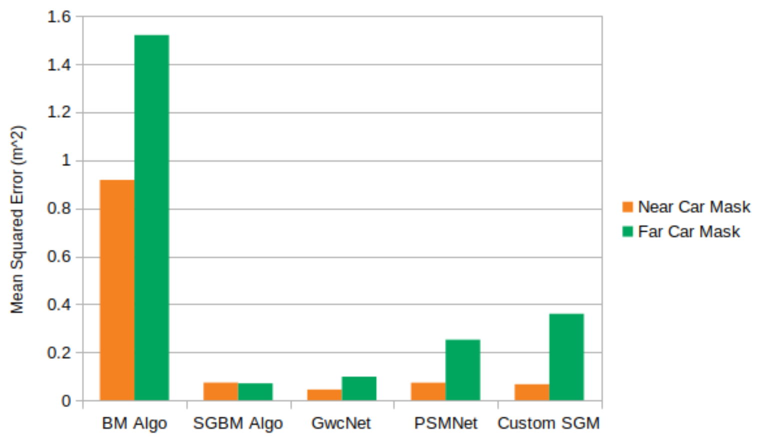


**Figure A.5: Comparison of MSE values from depth maps generated by different algorithms in a wide baseline camera configuration of the near and far vehicles**

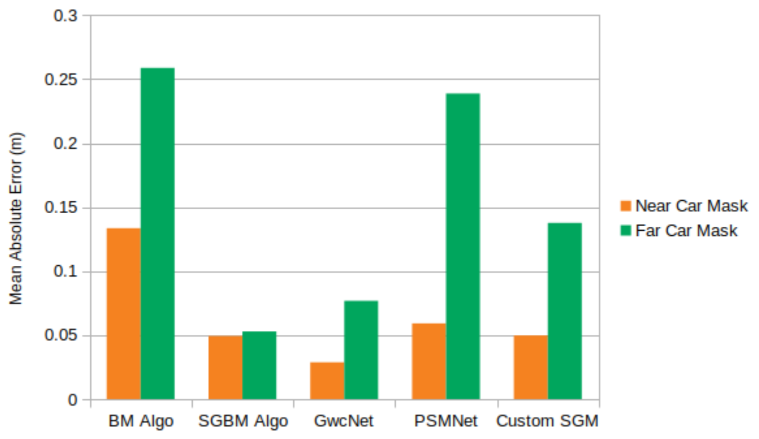


**Figure A.6: Comparison of MAE values from depth maps generated by different algorithms in a wide baseline camera configuration of the near and far vehicles**

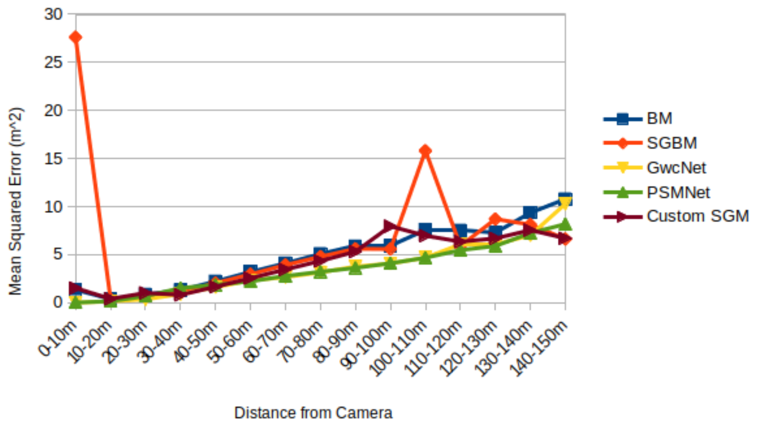


Figure A.7: Comparison of MSE values from depth maps generated by different algorithms in a wide baseline camera configuration at various 10m intervals

## A.2 Multi-Baseline Stereo

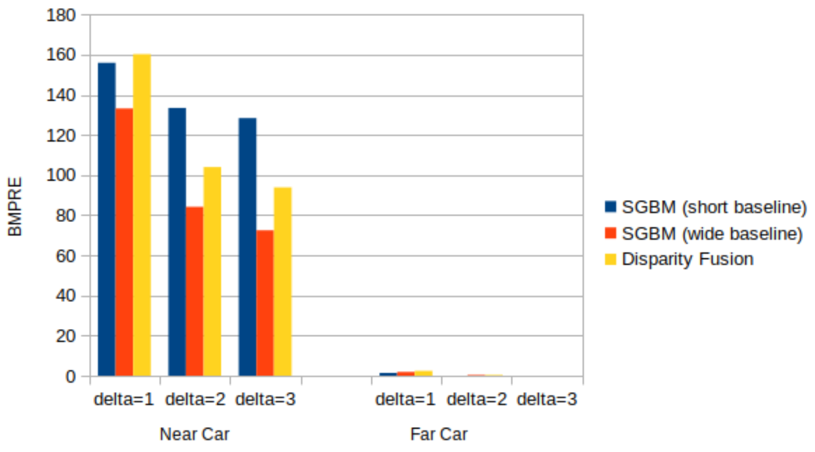
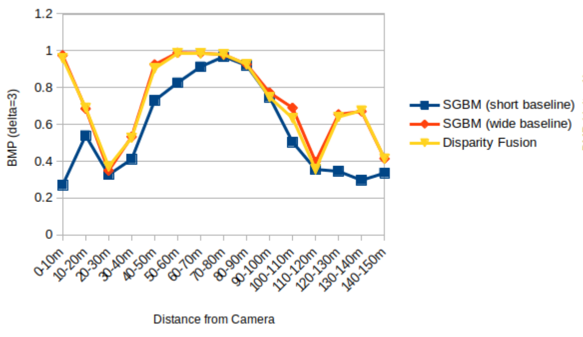
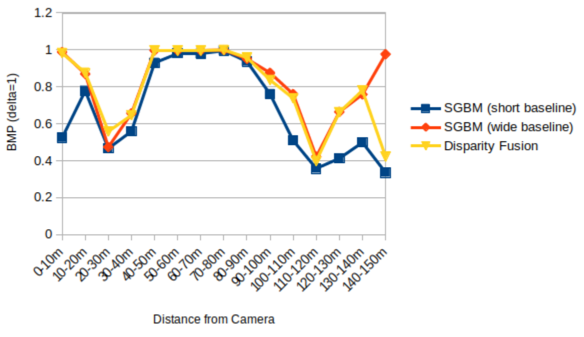


Figure A.8: Comparison of BMPRE between disparity fusion in a multi-baseline configuration and traditional stereo for the algorithm SGBM of the near and far vehicles



(a) $\delta$=3px



(b) $\delta$=1px

**Figure A.9: Comparison of BMP between disparity fusion in a multi-baseline configuration and traditional stereo for the algorithm SGBM at various 10m intervals**

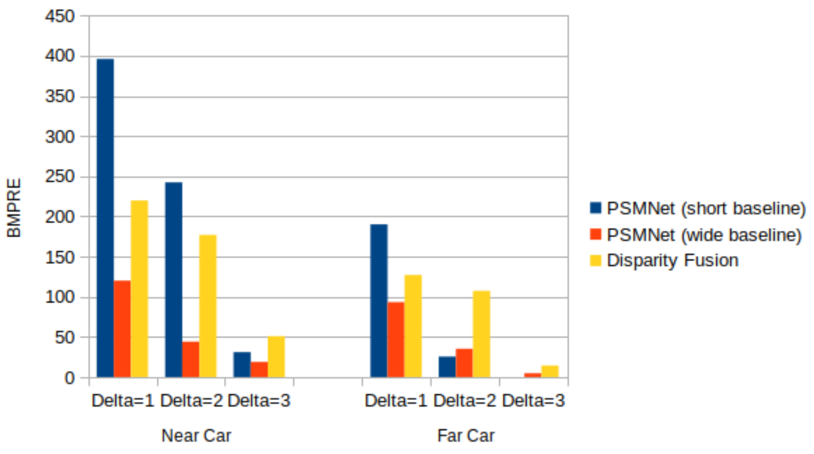
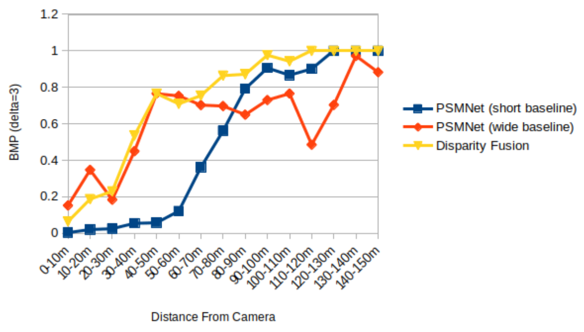
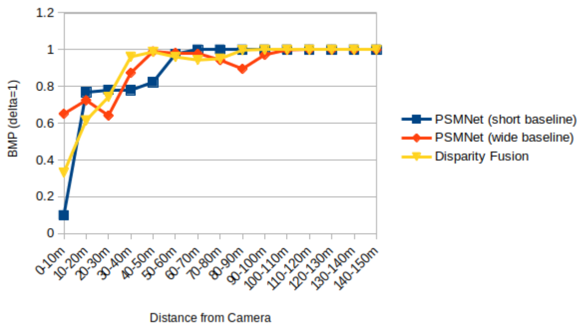


**Figure A.10: Comparison of BMPRE between disparity fusion in a multi-baseline configuration and traditional stereo of the algorithm PSMNet of the near and far vehicles**



(a) $\delta$=3px



(b) $\delta$=1px

**Figure A.11: Comparison of BMP between disparity fusion in a multi-baseline configuration and traditional stereo for the algorithm PSMNet at various 10m intervals**

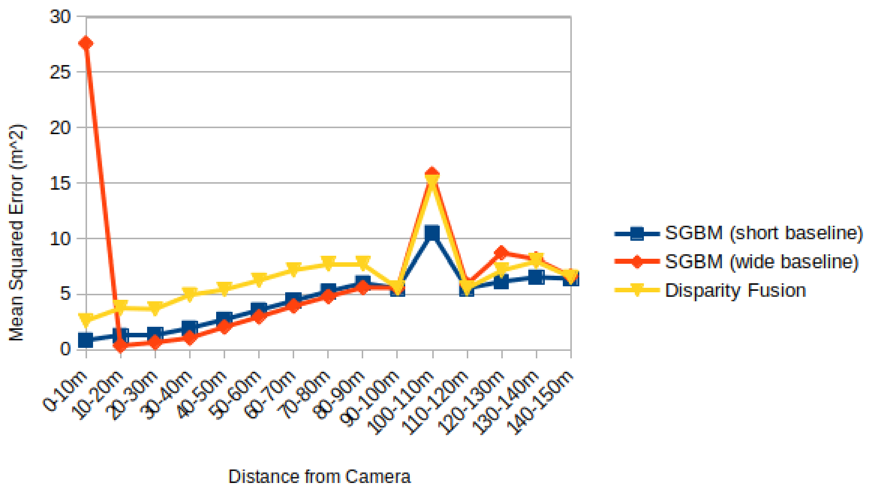


**Figure A.12: Comparison of MSE between disparity fusion and traditional stereo configurations for the algorithm SGBM at various 10m intervals**

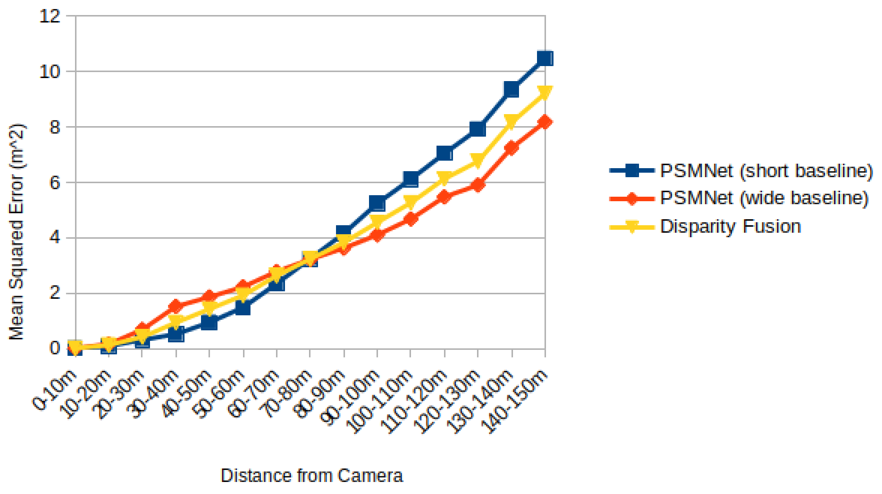


**Figure A.13: Comparison of MSE between disparity fusion and traditional stereo configurations for the algorithm PSMNet at various 10m intervals**

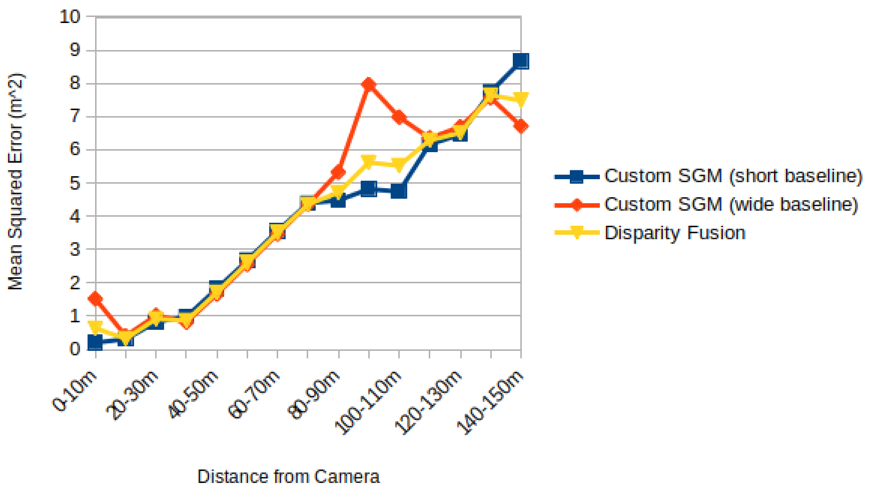


**Figure A.14: Comparison of MSE between disparity fusion and traditional stereo configurations for the algorithm Custom SGM at various 10m intervals**

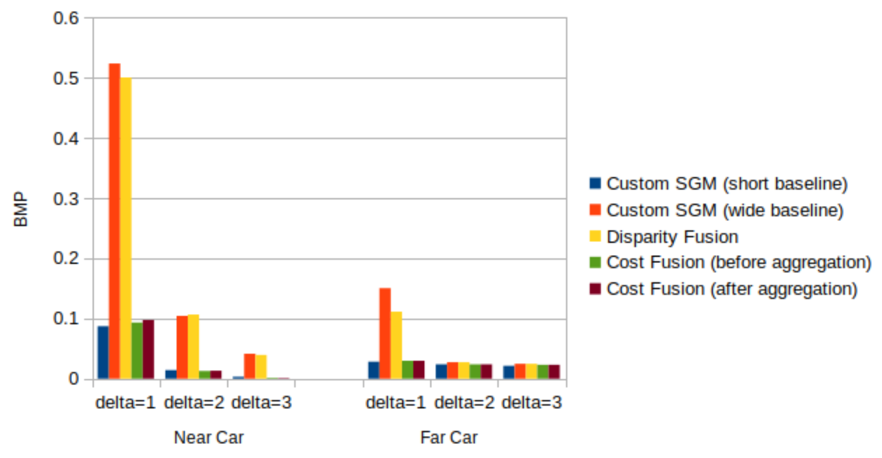


**Figure A.15: Comparison of BMP between fusion methods and traditional stereo in a multi-baseline configuration of the near and far vehicles**

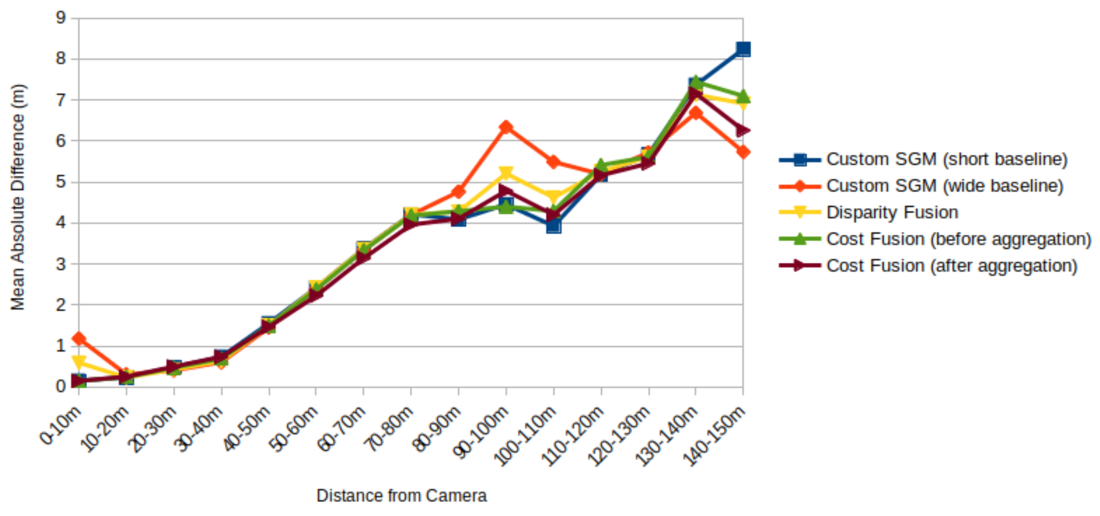
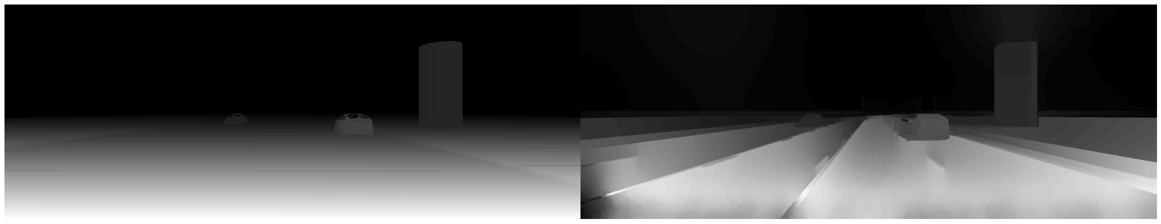
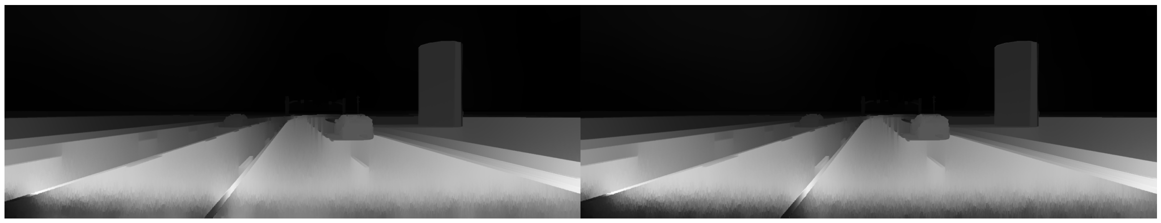


**Figure A.16: Comparison of MAE between fusion methods in a multi-baseline configuration and traditional stereo for the algorithm Custom SGM at various 10m intervals**

## A.3 Multi-axis Stereo



(a) Ground Truth (b) Custom SGM (horizontal)



(c) Cost Fusion (before aggregation) (d) Cost Fusion (after aggregation)

**Figure A.17: Comparison between traditional stereo configuration disparity maps and the multi-axis cost fusion disparity map**

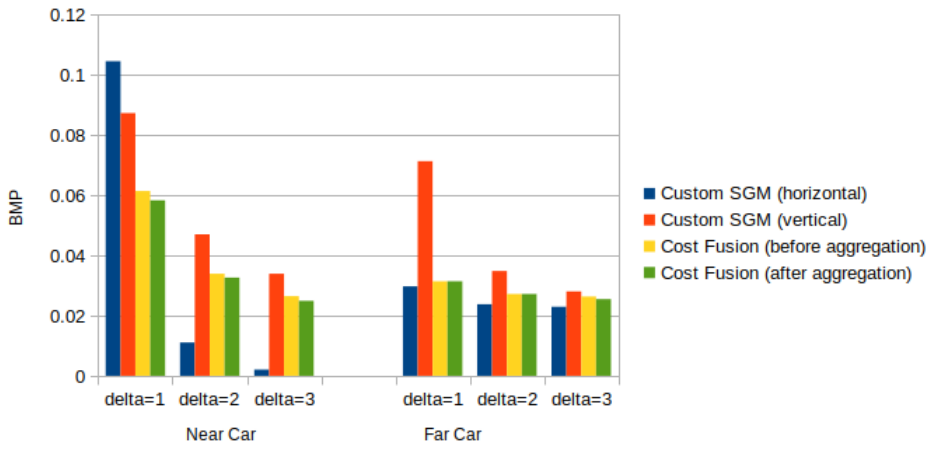


**Figure A.18: Comparison of BMP between fusion methods and traditional stereo in a multi-axis configuration of the near and far vehicles**

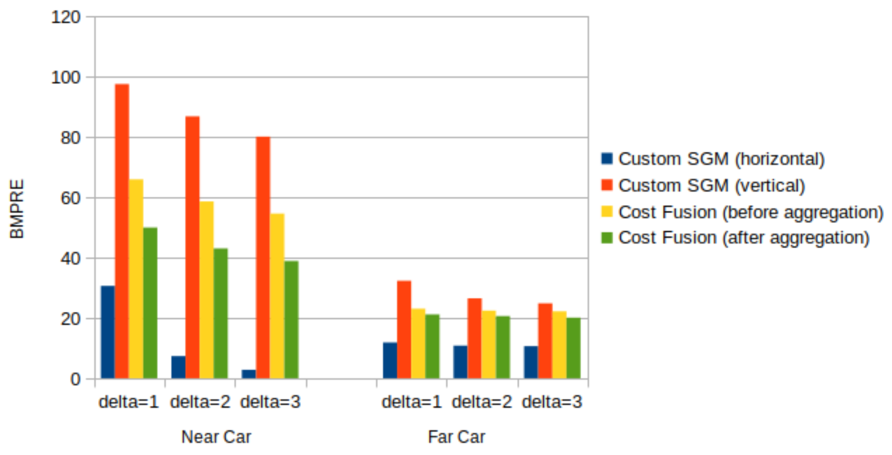


**Figure A.19: Comparison of BMPRE between fusion methods and traditional stereo in a multi-axis configuration of the near and far vehicles**

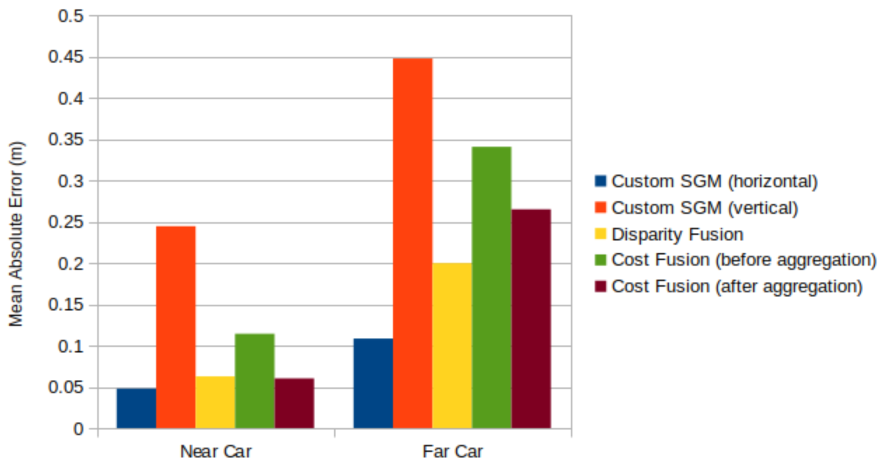


**Figure A.20: Comparison of MAE between fusion methods and traditional stereo in a multi-axis camera configurations of the near and far vehicles**

## A.4 Multi-Baseline vs Multi-Axis

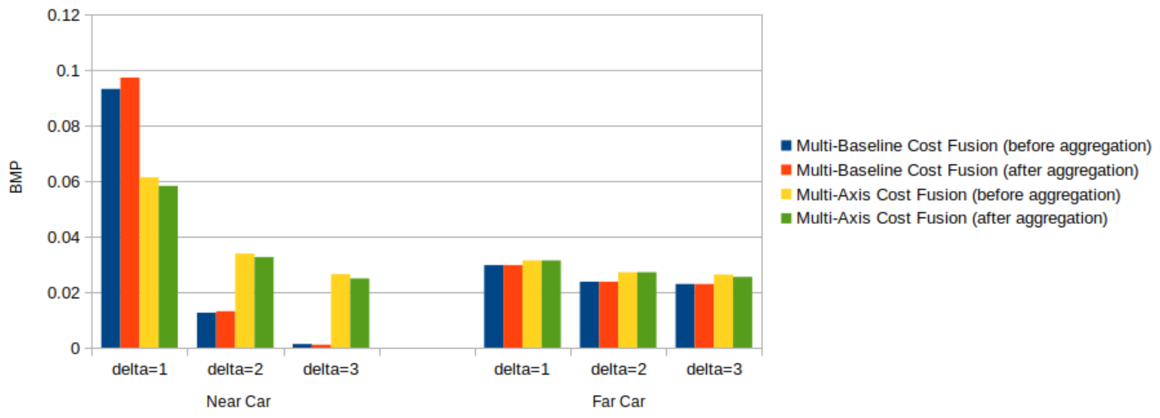
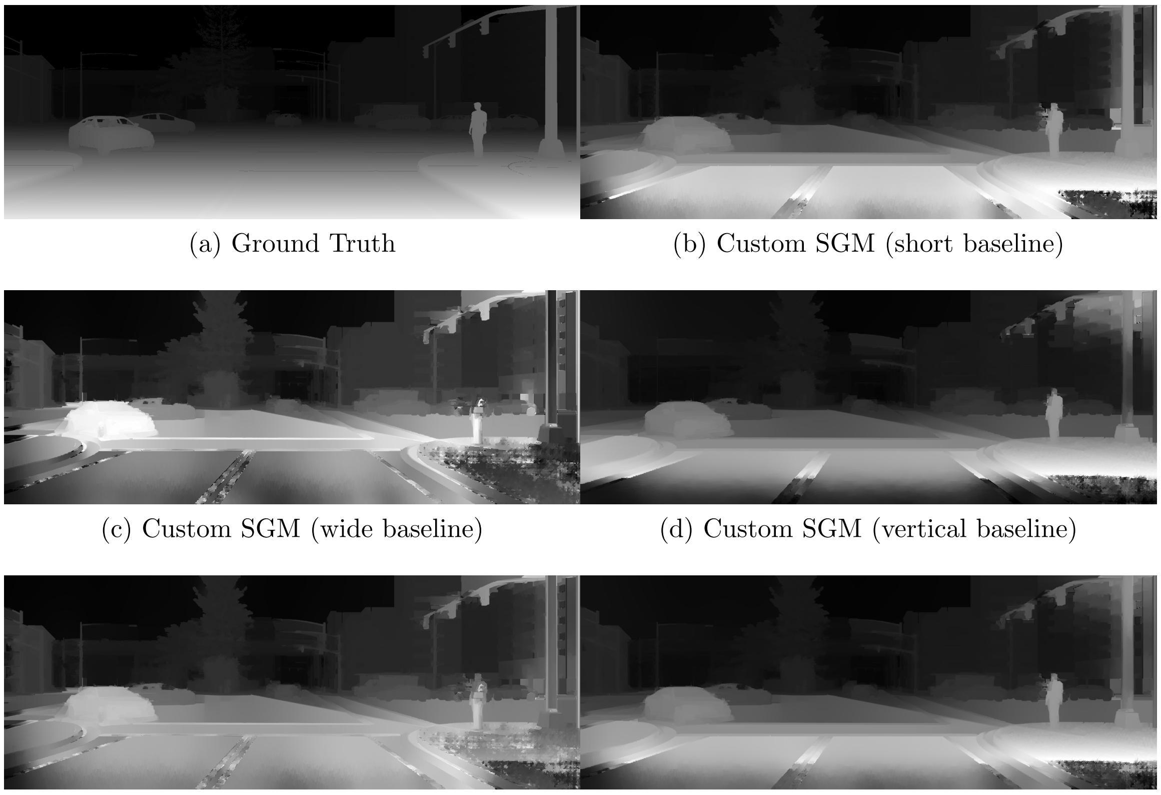


**Figure A.21: Comparison of BMP for cost fusion disparity maps between multi-axis and multi-axis camera configurations of the near and far vehicles**


(a) Ground Truth

(b) Custom SGM (short baseline)

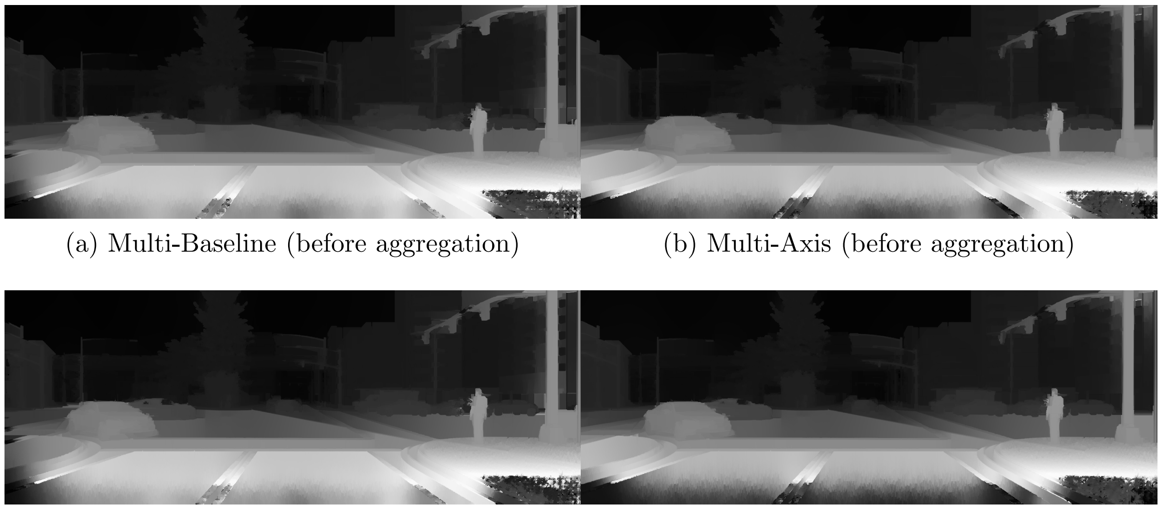
(c) Custom SGM (wide baseline)

(d) Custom SGM (vertical baseline)

(e) Disparity Fusion (multi-baseline)

(f) Disparity Fusion (multi-axis)

**Figure A.22: Comparison between disparity maps generated by Custom SGM in traditional camera configurations and disparity maps generated by disparity fusion of the CARLA city environment**


(a) Multi-Baseline (before aggregation) (b) Multi-Axis (before aggregation)

(c) Multi-Baseline (after aggregation) (d) Multi-Axis (after aggregation)

**Figure A.23: Comparison of cost fusion methods in different camera configurations of the CARLA city environment**