**Syrian Arab Republic**
**Ministry of Higher Education**
**Manara University**
**Engineering Faculty**
**Robotics and Intelligent Systems Section**

# Front View Obstacles Detection for Autonomous Cars Based on Stereo Vision

## Prepared By:

Alhadi Zidan
Jawa Habib

## Supervised By:

Dr. Hassan Alahmad
Dr. Samer Suleiman
Eng. Wassim Ahmad

**Date:**
**25 . July . 2023**

# Table of Contents

# Table of Figures

# Table of Tables

# Abstract

Obstacle avoidance is a crucial functional requirement in autonomous vehicles (AV). Its performance is strictly related to accurate detection of obstacles in the vehicle's surroundings. Through literature, different approaches for obstacle detection have been discussed depending on the method used for environment perception. This project presents a front view obstacle**s** detection approach for autonomous vehicles based on stereo vision. The experiments were dedicated into two core sections of the proposed algorithm: stereo matching and road removal. Different methods for stereo matching have been implemented, advantages and disadvantages of each method were presented, and a discussion was held in order to identify the appropriate method in terms of reliability and computational cost. A state-of-the-art road removal approach was proposed in order to isolate the obstacles in each stereo pair. The proposed algorithm was implemented on DrivingStereo dataset Cloudy Weather subset, which provides 500 stereo images taken by a stereo camera mounted on a car driving in a Chinese city. **The proposed methodology was able to distinguish between the free drivable space and the obstacles in front of the car.**

# Introduction and Related Works

## 1.1 Introduction

Obstacle Avoidance is a crucial functional requirement of autonomous cars because it is directly related to safety constraints and path planning algorithms. Obstacle Avoidance process can be introduced as in Figure1.1.

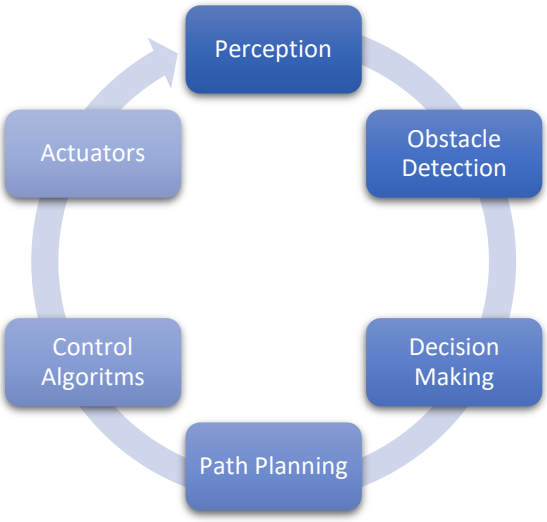



*Figure 1.1 Obstacle Avoidance Process*

1. Perception: represents data acquisition and reading the surroundings. Common sensors used are Cameras, LIDAR, RADAR, Ultrasonic, …

2. Obstacle Detection: represents the core of the process which stimulates the following steps. It detects objects from sensors output and classifies them into obstacles and non-obstacles.

3. Decision Making: An important step in the process. This step is responsible for determining the attitude of the car depending on the traffic and the users' situation e.g., modes of driving such as moderate, aggressive, and calm driving, or are we taking over the obstacle, slowing down, or stopping because of him.

4. Path Planning: This step is responsible for setting the path that the car must follow to overtake or avoid the obstacle. This includes the speed, steering, and global path following.

5. Control: Given the desired steering and speed, a Control Algorithm is used to keep the car on the desired path.

6.Actuators: this step is the final step. It takes the output of the control system as an input. It is no more than hardware getting orders from the microcontroller or microprocessor used.

Each step is an individual process, the consecution and repetition of those steps resemble the overall obstacle avoidance algorithm.

In order to achieve a correct obstacle avoidance, the obstacles' positions must be identified accurately, which means a reliable obstacle detection algorithm is needed. This work is dedicated to distinguishing between the free drivable space and the obstacles in front of the car in cloudy weather using a stereo camera mounted on the car.

## 1.2 Related Works

Obstacle detection approaches can be categorized into three groups depending on the method used for environment perception. The three categories are: sensor-based, image-based, and hybrid.

### 1.2.1 Sensor-based approach

This approach relies on using active sensors such as ultrasonic sensors, Radio Detection and Ranging (Radar) sensors, and Light Detection and Ranging (LiDAR) sensors to retrieve data from the car's surroundings. This approach is common and widely used in literature. Many researchers have studied active sensors implementation for obstacle detection.

In [1], an obstacle detection algorithm based on ultrasonic sensors for autonomous land vehicle has been developed. The algorithm was able to solve the common problem of the ultrasonic sensor's traditional obstacle detection mode, in which only one obstacle can be detected in one time cycle since the smallest cycle time of sonar return is 60ms in 10m range. Additionally, the results show that the algorithm dramatically reduced the specular reflection, cross-talk, and environment noise, at the same time of measuring the distance to obstacle with high accuracy. In contrast, the proposed algorithm considers only the objects lying in the overlapped area of the two sonar's aperture angle ranges. Moreover, obstacles weren't presented in real shape and size, rather in the form of circles and lines with no reference to their state whether they are static or moving.

Despite the cheap cost of ultrasonic sensor compared to other sensors, its performance is greatly affected by the changes of environmental conditions such as temperature and humidity [2]. Additionally, its waves are affected by both constructive and destructive interference of ultrasonic reflections from multiple environmental obstacles [3]. Hence, other range sensors such as RADAR and LIDAR have been more commonly used in this context.

RADAR has been commonly discussed by researchers for obstacle detection due to its long-range detection capabilities, resilience to environmental conditions, and reasonable cost [4] [5]. Nevertheless, considering it as a reliable method is still facing crucial challenges such as mutual-interference mitigation, limited resolution, false detection of metal objects around the perceived surroundings like road signs or guardrails, and the challenges of distinguishing static, stationary objects [6] [7].

LIDAR has been a popular choice for obstacle detection due to its fast response, high accuracy, and ability to give point cloud data [8] [9]. In contrast, its high cost, and inability to detect overhanging obstacles due to its constant predefined position and angle, motivated the researchers to find other solutions.

### 1.2.2 Image-based approach

Image-based obstacle detection methods can be divided into monocular and stereo. While monocular methods rely on a single image taken by a single camera mounted in front of or around the car, stereo methods use two images of the same scene taken from two different views using two synchronized cameras [10].

Most monocular methods are based on using deep learning models for objects detection. These methods classify the objects presented in the image according to the classes presented in their training datasets. Based on the class assigned to each object, they differentiate between obstacles and free drivable space.

Despite the large amount of research published to enhance the accuracy and reliability of these models, their main challenges are still present [11] [12]. Some of the main problems are not providing 3D structure and not getting overhanging obstacles. The reason behind is setting fixed 2D rectangles, while the world is 3D. Additionally, the models' performance mainly depends on their dataset. When an object that is not a part of the dataset occurs in the car's surroundings, ontology cracks may happen [13].

On the other hand, stereo vision has the benefit of extracting depth information and 3D geometry from passive 2D images without the need for dedicated active light-emitting range measurement devices like LIDARs. In [14], a methodology to employ two 360◦ cameras vertically to perceive obstacles all around the autonomous vehicle using stereo vision is presented. On the other hand, cameras' performance is affected by complex illumination, complex shadows, and bad weather [9].

### 1.2.3 Hybrid approach

This approach combines the two previous ones to solve their drawbacks. In [15], a LIDAR and monocular based overhanging obstacle detection was proposed. While 2D LIDAR measures the precise distance to the object, it cannot detect low objects and overhanging obstacles due to its predefined, constant, scanning height and angle. On the other hand, monocular cameras provide 2D scenery information with relatively poor distance information. To solve these problems, a sensor fusion approach was implemented.

# Theories and Formulations

This chapter is aimed at giving an introduction to the theories and mathematical backgrounds used through the work. First, the used sensing system is defined properly. Some assumptions are adopted for simplifying the solution with a small relative error. Then, several keywords, which are the main outputs of the algorithm, are defined and explained to the needed abstraction with previewing the most used algorithms in the field. After that, some *Segmentation* algorithms are previewed and explained. Finally, some *Filtration and Smoothening* operators, which are needed for pre-processing and post-processing steps, are shown briefly.

## 2.1 Sensing System

The sensing system needed for the proposed method is a Stereo Camera System. This section introduces the mentioned system with the adopted assumptions, but first, some theories and models are presented briefly.

### 2.1.1 Camera Models

Camera models are used to represent the cameras in a mathematical form. The purpose of this representation is to make new improvements and uses of the system or to model the system attitude and simulate the process the system is working on. In other words, camera modeling is to approximate the camera as a certain mechanism and write the mathematical equations describing it. The most popular camera model is the pinhole model.

The *pinhole model* is the basic camera model used in computer vision. Its name stems from the concept of pinhole camera (also related to the camera obscura): usually, a closed box into which a single tiny hole is made with a pin, through which light may enter and hit a photosensitive surface inside the box (cf. Fig. 2.1). The pinhole camera model mimics the geometrical projection carried out by a pinhole camera, as follows (see also Fig. 2.1). The entire optics and aperture of a camera are reduced to a single point – the *optical center* or *center of projection*. The photosensitive surface is assumed to be planar and is, geometrically, represented by the so-called image plane. To determine where a 3D point is depicted in the image, it suffices to construct a straight line from that point and going through the optical center (one may consider this as a light ray). This line's intersection with the image plane gives the desired image point of the 3D point. A key property of this model is that points that are collinear in 3D get imaged to image points that are also collinear. [16]

Before diving into the mathematical equations, some notations must be defined to use them throughout the work. The *Focal Length f* is the distance between the center of projection and the image plane. The *Optical Axis is* the line passing through the optical center and that is orthogonal to the image plane. The *Principal Point* is the intersection point of the optical axis and the image plane. In addition to those notations, one needs to take into account the *layout of pixels* in the image plane. The *layout of pixels* describes how the pixels are arranged in the image. One simple layout is the rectangular grid, where pixels are arranged into rows and columns. The *Aspect Ratio* is defined as the value of $k_u/k_v$, where $k_u$ , $k_v$ are the column-wise and row-wise density of pixels, respectively. [16]

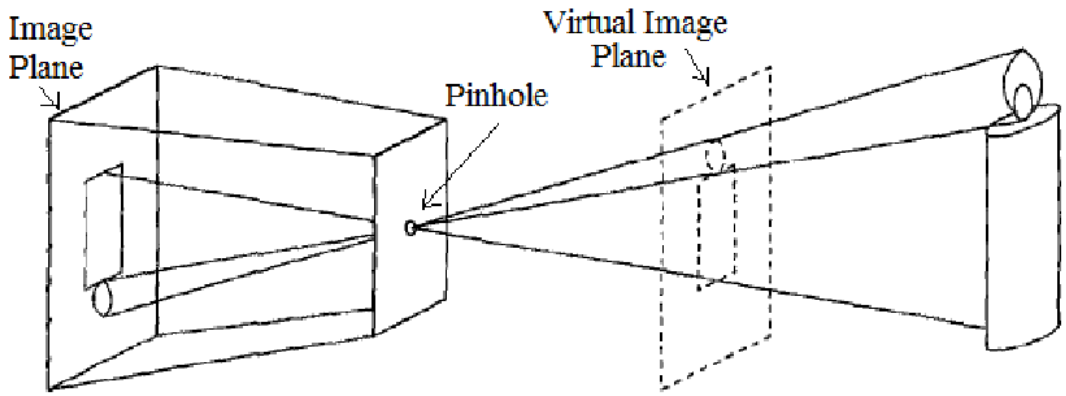



*Figure 2.1 Pinhole Camera planes*

### 2.1.1 Model Equations

To start with the mathematical representation of the model, coordinate frames must be defined and transformation matrices have to be well calibrated. In general, the modelling of a camera needs to define 4 coordinate frames: *World Coordinate frame, Camera Coordinate frame, Image Coordinate frame,* and *Pixel Coordinate frame.* Usually, the world frame is fixed somewhere while the others are carried on the camera. The camera frame center is carried on the optical center, the image frame center is on the principal point on image plane, and the pixel frame center is in the corner of the image plane. the z-axis is considered to be coincident with principal axis for both camera and image frames. The purpose of pixel frame is to represent the 3D points using pixel units not metric units as in other frames. [17] [16]

To project a point in 3D onto the image using pixel frame, transformations from each frame to the next must be defined. The point is first in the world coordinates. The transformation from world coordinates to camera coordinates is by linear transformation or *Affine transformation*. The transformation matrix is constructed from a translation then a rotation as the following,

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R^T} & -\boldsymbol{R^T t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} . \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \qquad (2.1)$$

where $R$ is the orientation of the camera frame, $t$ is the coordinates of the camera frame origin, $(X_w, Y_w, Z_w)$ and $(X_c, Y_c, Z_c)$ are the coordinates of a point in world coordinates and camera coordinates respectively.

The projection of the point in the camera frame onto the image frame is derived using similar triangles as the following,

$$\frac{x}{f} = \frac{X_c}{Z_c} \quad \Rightarrow \quad x = f.\frac{X_c}{Z_c} \qquad (2.2)$$

$$\frac{y}{f} = \frac{Y_c}{Z_c} \quad \Rightarrow \quad y = f.\frac{Y_c}{Z_c} \qquad (2.3)$$

Now, the transition from the image coordinates to the pixel coordinates,

$$u = k_u(x + x_0) = k_u.f.\frac{X_c}{Z_c} + k_u x_0 \qquad (2.4)$$

$$v = k_v(y + y_0) = k_v.f.\frac{Y_c}{Z_c} + k_v y_0 \qquad (2.5)$$

Overall, the mapping of a 3D point in world coordinates into the pixel coordinates is described as follows, [17] [16]

$$K\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u.f & 0 & k_u x_0 & 0 \\ 0 & k_v.f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} \boldsymbol{R^T} & \boldsymbol{-R^T t} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} . \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (2.6)$$

where $K$ is a constant put to refer to the fact that the right side is related to the other one by a constant. The first matrix is called the *Intrinsic matrix*, which has the *intrinsic parameters* of the camera. And the second matrix is called the Extrinsic parameters, which are the parameters derived from Affine Transformation.

The derived equation is the simplest form of a camera model. The pinhole model is simple and neglects many aspects of true cameras. For instance, apertures are finite and thus most 3D points are not imaged in a unique image point but within a finite area of the image plane. Likewise, pixels in digital cameras gather incoming light across finite areas. Thus, the pinhole model does not tell anything about blur, point spread functions, quantization, or other effects such as vignetting that occur in true cameras. [17] Another important aspect which is not included in the model is the *lens distortion*. This kind of distortion is added by the optic lens of the camera. The lenses contribute the radial lens distortion, familiar as, barrel distortion or pincushion distortion, as in Fig. 2.2. This distortion is modelled by the following equations,

$$x_u = c_x + (x_d - c_x)(1 + k_1 r_d^2 + k_2 r_d^4 + e_x) \quad (2.7)$$

$$y_u = c_y + (y_d - c_y)(1 + k_1 r_d^2 + k_2 r_d^4 + e_y) \quad (2.8)$$

for the principal point $(c_x, c_y)$ and $r_d = \sqrt{(x_d - c_x)^2 + (y_d - c_y)^2}$, given the lens-distorted image point $(x_d, y_d)$ to obtain the undistorted image point $(x_u, y_u)$.

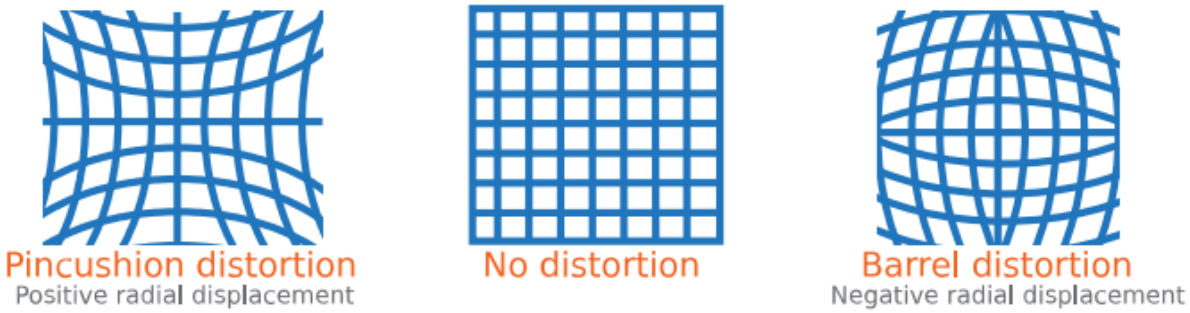



*Figure 2.2 Radial Distortion [26]*

In general, pinhole model can provide a good approximation in many applications. Other improvements will include adding some of the neglected effects. This will increase the complexity of the problem by decreases the error of the process.

### 2.1.2 Calibration

When using a camera or a system of cameras, a *calibration* operation must be performed before the system starts working. *Camera calibration* in the context of three-dimensional machine vision is the process of determining the internal camera geometric and optical

characteristics (intrinsic parameters) and/or the 3-D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters). [18]

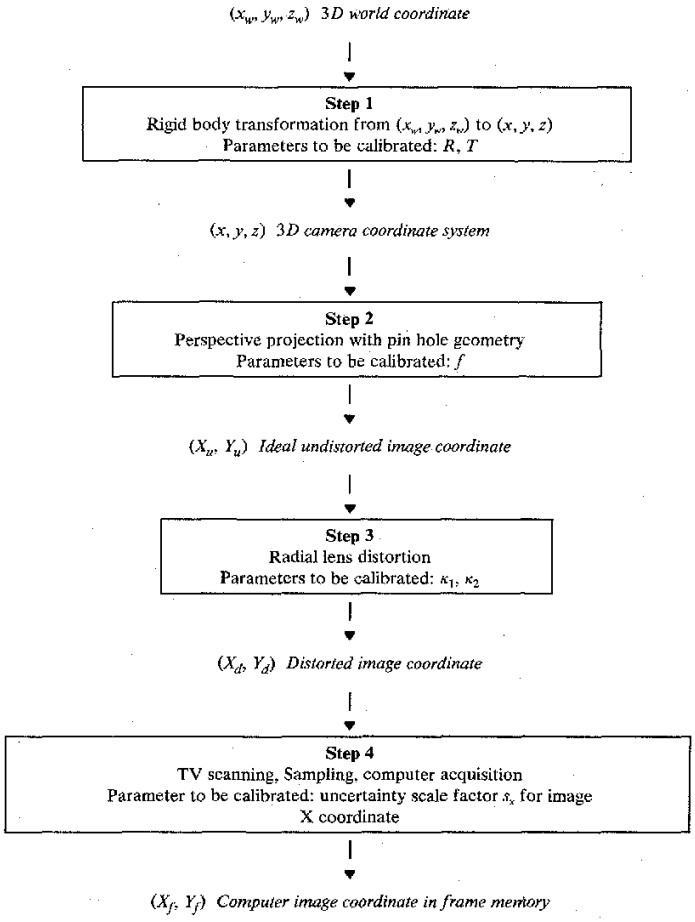



*Figure 2.3 Calibration Algorithm diagram [18]*

Intrinsic or internal parameters are the (effective) focal length, dimensions of the sensor matrix, sensor cell size or aspect ratio of sensor height to width, radial distortion parameters, coordinates of the principal point, or the scaling factor. Extrinsic parameters are those of the applied affine transforms for identifying poses (i.e., location and direction) of cameras in a world coordinate system. [16] Fig. 2.3 illustrates the process of 3D point mapping to 2D image frame and the parameters that need to be calibrated at each step.

As the calibration process is not the point of interest in this work, details will not be covered in the context. Anyways, the general calibration procedure includes taking some number of known pints in world coordinates and their corresponding points in image planes. The number of those points are way bigger than the number of parameters to be calibrated, which means that the problem reduces to solving a system of non-linear equations. [18] [17]

### 2.1.4 Two camera systems

In many applications and projects, understanding or representing the 3D scenes is a crucial step throughout the algorithm. Multiple cameras must be used in order to get the 3D coordinates of the scenes. For this reason, it is important to mention camera systems in this context. In this work, two cameras will be used, so only 2-camera systems will be covered.

**Stereo Camera System:** systems that use more than a camera are called *stereo systems*. Those multiple cameras can be either configured in a specific geometry, or can be randomly configured, according to the application purposes. Stereo camera systems need to be calibrated before starting. This calibration is necessary to know the exact configuration of the cameras

(Extrinsic Parameters), as well as the intrinsic parameters of each camera. It is preferred that the cameras are as identical as possible. However, some errors in configuration or in internal measurements of camera parameters will be found. Calibration process removes that error and it can be considered as identical cameras in a known configuration.

Fig. 2.4 shows the configuration of the two cameras used in this work, where each camera is described by the pinhole model. This configuration is called *Canonical Stereo Geometry*. This geometry is characterized by having an identical copy of the camera on the left translated by the distance b along the $X_s$-axis of the $X_sY_sZ_s$ camera coordinate system of the left camera. The projection center of the left camera is at $(0, 0, 0)$ and the projection center of the cloned right camera is at $(b, 0, 0)$. In other words, we have:

1. two coplanar images of identical size $N_{cols} \times N_{rows}$,
2. parallel optic axes,
3. an identical effective focal length $f$ , and
4. collinear image rows (i.e., row y in one image is collinear with row y in the second image).

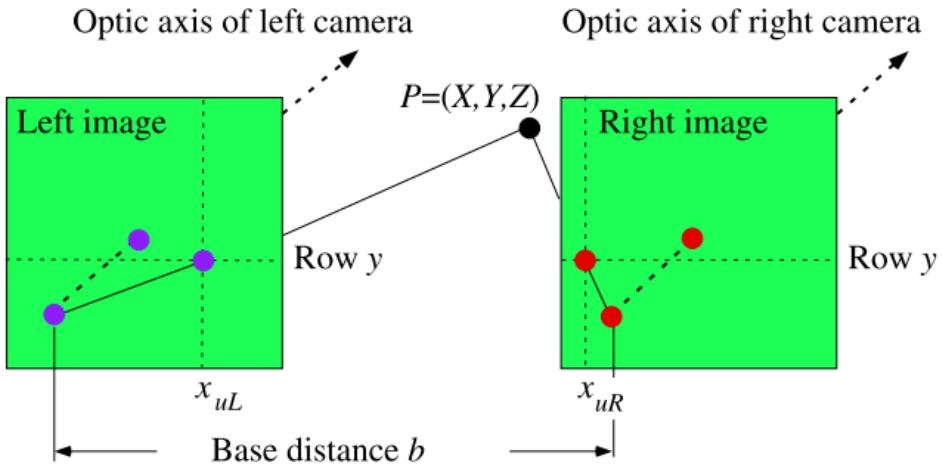



*Figure 2.4 Stereo Camera Configuration*

By applying the central projection equations of (2.2) and (2.3) for both cameras, a 3D point $P = (X_s, Y_s, Z_s)$ in the $X_sY_sZ_s$ coordinate system of the left camera is mapped into undistorted image points.

$$P_L = \left(\frac{f.X_s}{Z_s}, \frac{f.Y_s}{Z_s}\right)$$

$$P_R = \left(\frac{f.(X_s - b)}{Z_s}, \frac{f.(Y_s - b)}{Z_s}\right)$$

### 2.1.3 Rectification

Rectification is the process projecting the image planes of the camera onto a mutual plane. the aim of this process is to make the epipolar lines, which will be described later, horizontal and colinear so the matching process can be more straightforward and less computational complex.

## 2.2 Disparity Map

### 2.2.1 Proof of Validity

In this section, a mathematical proof of the possibility of knowing the depth of a point in 3D using its pixel representations in two camera is discussed. First, using the pinhole camera model discussed earlier, the transformations from each camera's coordinates frames to the pixel coordinates frame of each image are described by the following equations, [19]

$$K\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \begin{bmatrix} k_u.f & 0 & k_u x_0 & 0 \\ 0 & k_v.f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} \quad (2.9)$$

$$K\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} k_u.f & 0 & k_u x_0 & 0 \\ 0 & k_v.f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix} \quad (2.10)$$

And the equation of the Affine transformation between the camera coordinates is the following,

$$\begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} . \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} \quad (2.11)$$

Using the equations 2.11 through 2.13, the following equations are derived, which find for a point in 3D its pixel in both cameras.

$$K\begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \begin{bmatrix} k_u.f & 0 & k_u x_0 & 0 \\ 0 & k_v.f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} . \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} \quad (2.12)$$

$$K\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} k_u.f & 0 & k_u x_0 & 0 \\ 0 & k_v.f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} . \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} \quad (2.13)$$

If the pixel coordinates in the two cameras of a 3D point are known, the problem turn into solving 4 linear equations with 4 variables, which means that a 3D coordinates of a point can be calculated using its corresponding pixels in the two cameras.

### 2.2.2 Correspondence

As mentioned before, equations 2.14 and 2.15 relates the point's coordinates with its pixels in both images. Here an important concept appears, the *correspondence of points*. Correspondent points in two images are the two pixels that refers to the same 3D point. [16]

Finding the correspondent points is very essential to get the 3D coordinates of the point. A trivial solution to this problem is the linear search on the correspondent point from the second image. However, this is not efficient and inaccurate due to multiple similarity possibilities. Common algorithms for this problem are the block matching algorithms.

**Block Matching algorithms:** The idea behind the block matching is to search for the best match of a point and its neighbors together as a block. Depending on a specific criterion, which is called cost function, the best match, or the correspondent block, relates to the centers of the blocks.

Cost functions are the criteria that are needed to consider the block as a best match. The most used cost functions are the following: [16]

- **SAD (Sum of Absolute Differences)** $\sum_1^n \sum_1^m |\text{img1}(i,j) - \text{img2}(i,j)|$
- **SSD (Sum of Squared Differences)** $\sum_1^n \sum_1^m (\text{img1}(i,j) - \text{img2}(i,j))^2$
- **NCC (Normalized Cross Correlation)**

### 2.2.3 Epipolar Plane and Lines

The *Epipolar Plane* is the plane defined by the focus of the two cameras and the 3D point. The intersection of this plane with each image plane is two lines called *Epipolar Line.* By geometry, it is found that the correspondent point of any point in the first image laying on its Epipolar line is a specific point on the Epipolar line of the other image. Figure 2.5 depicts the situation clearly.

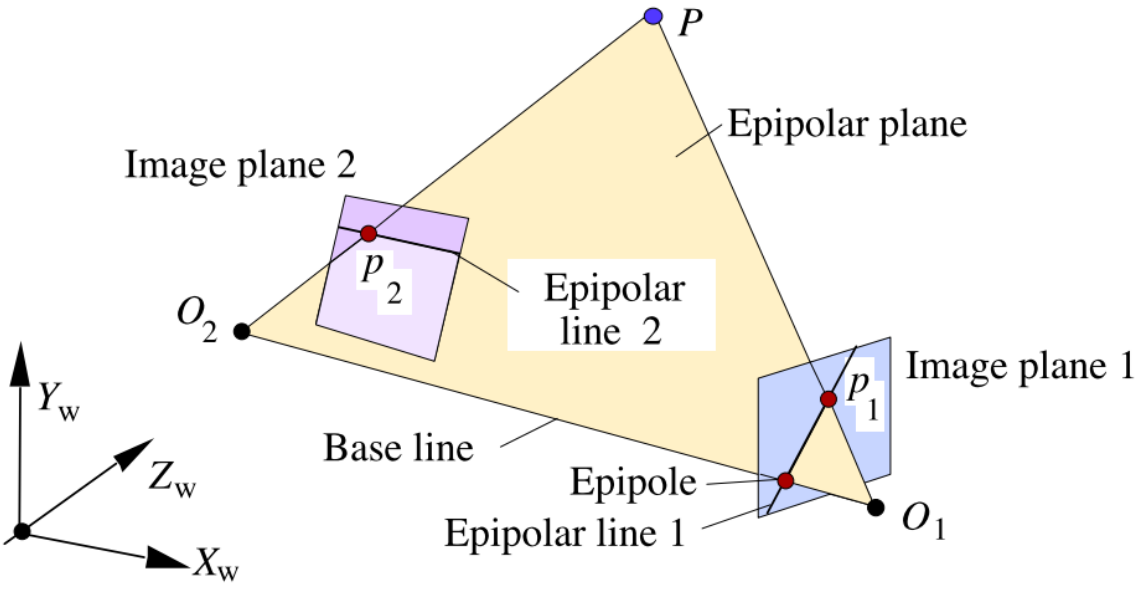



*Figure 2.5 Epipolar Plane and Lines [16]*

This important result improves the accuracy and complexity of matching algorithms by only searching that specific line for correspondence.

### 2.2.4 Disparity Map

*Disparity* of a point is the difference of its correspondent pixel points of the cameras. Figure 2.5 shows the Epipolar lines and plane of a random camera configuration. In canonical stereo camera, the Epipolar lines are horizontal and colinear, except for some inaccuracy in measurements and manufacturing which are corrected by the rectification process. These properties are responsible for making the disparity of a point only related to the shift on the x-axis. The following equation represents the disparity of the canonical stereo camera,

$$Disparity = u_x^l - u_x^r \quad (2.14)$$

## 2.3 Depth Map

Depending on equations 2.10 through 2.16, by solving them using Least Square Method in solving linear equations, the estimation of 3D coordinates of a point is found to be as the following,

$$X_p = \frac{bu_x^l}{disparity} \quad (2.15)$$

$$Y_p = \frac{bu_y^l}{disparity} \quad (2.16)$$

$$Z_p = \frac{bf}{disparity} \text{ (2.17)}$$

where b is the baseline and f is the focal length. Equation 2.19 calculates the depth of a point using its disparity, this is the theoretical side behind the practical work of this project [17] .

## 2.4 Filtering

Filtering is a fundamental operation in image processing that plays a crucial role in enhancing image quality, reducing noise, and extracting features. In this context, filtering refers to the process of modifying an image by applying a mathematical function to each pixel or a small neighborhood of pixels. Some common types of filters used in image processing median filter, Gaussian filter, mean filter, and bilateral filter. The choice of filter depends on the specific application and the desired outcome.

### 2.4.1 Median Filter

Median filtering is a non-linear smoothing method that reduces the blurring of edges by replacing the current pixel in the image with the median of the pixels in its neighborhood. The median in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulse noise quite well. Further, as median filtering does not blur edges much, it can be applied iteratively [20]. One downside of the median filter, in addition to its moderate computational cost, is that since it selects only one input pixel value to replace each output pixel, it is not as efficient at averaging away regular Gaussian noise [19].

### 2.4.2 Gaussian filter

This filter is based on the gaussian distribution. Hence, it is used to remove gaussian noise. The Gaussian filter is given by:

$$G(x, y) = e^{-(x^2+y^2)/2\sigma^2} \text{ (2.18)}$$

where x, y are the image co-ordinates and σ is a standard deviation of the associated probability distribution. The standard deviation σ is the only parameter of the Gaussian filter—it is proportional to the size of the neighborhood on which the filter operates [20].

# Equipment and Tools

## 3.1 Python

Python is a high-level, interpreted programming language that has gained popularity in the computer vision field due to its simplicity, readability, and versatility. Python's extensive libraries and frameworks provide powerful tools for image and video analysis, processing, and manipulation, making it an ideal choice for developing computer vision applications. Python's NumPy and SciPy libraries provide efficient numerical and scientific computing capabilities. NumPy library has been used in this project to deal with arrays and modify them.

Moreover, Other popular libraries for computer vision in Python include OpenCV, Pillow, and Matplotlib, which provide a range of functions for image processing, visualization, and manipulation. OpenCV and Matplotlib libraries have been used in this project to analyze images and modify them [21].

## 3.2 MATLAB

MATLAB is a programming and numeric computing platform used by millions of engineers and scientists to analyze data, develop algorithms, and create models. MATLAB is a widely used software tool for calibration and rectification of stereo cameras. Stereo camera calibration is a crucial step in stereo vision systems, as it ensures that the images from the two cameras are accurately aligned, and the distance between objects in the scene can be accurately measured. Rectification is another important step in the stereo vision pipeline, where the images are transformed to eliminate the skew caused by the camera's position and orientation. MATLAB provides a comprehensive toolbox for stereo calibration and rectification, including functions for camera calibration, stereo parameter estimation, and image rectification. The toolbox also includes a graphical user interface that simplifies the calibration and rectification process. [22] This toolbox will be used in the calibration and rectification of the stereo camera once the project’s hardware is available.

## 3.3 DrivingStereo

DrivingStereo is a stereo images dataset taken from a stereo camera on a car driving in a Chinese city [23]. In this work, it is used as input frames due to the unavailability of the hardware needed. The dataset consists of a total of 182188 stereo images with a resolution of $1762 \times 800$ pixels. The dataset is divided into subsets according to different criteria mentioned in the dataset documentation. One of those criteria is the weather. To eliminate the effects of weather in the images, e.g., sun, fog, rain…, Cloudy Weather subset has been used. This subset consists of 500 stereo images containing many types of road obstacles, such as pedestrians, cars, trucks, motorcycles, walls, trees, sidewalks, … The baseline of the stereo camera is 54cm and the FOV of the cameras is 50 deg. Fig 3.1 shows an example of a stereo frame.

*Figure 3.1 Left is the left image from the frame, Right is the right image of the pair*

# Methods and Experiments

This chapter presents the practical side of the work. It discusses the algorithms, the methodologies, and the steps tested and used throughout the work. First, the general algorithm of the work is presented as a block diagram. Then, each step and its different algorithms are discussed separately.

## 4.1 Work Algorithm

The proposed algorithm consists of two main procedures, *Initialization*, and *Detection Loop*. The *initialization* process is done only once, at the startup of the system. After the initialization is done, the system starts the *Detection Loop* and repeats it each instant. After each loop, the system is supposed to have an objects list as an output of the algorithm.

### 4.1.1 Initialization

As mentioned above, the initialization process takes place only once at the startup. It represents the pre-calculations related to the configuration of the system. In short, the initialization process is the calculation of calibration, rectification, and undistortion maps.

**Calibration:** MATLAB provides a powerful toolbox for calibrating both monocular cameras and stereo cameras. For stereo calibration, a set of checkerboard stereo images are needed. The set must be at least 20 stereo images. The toolbox is called by writing stereoCameraCalibrator instruction in MATLAB's command window. Fig. 4.1 shows the toolbox in MATLAB.

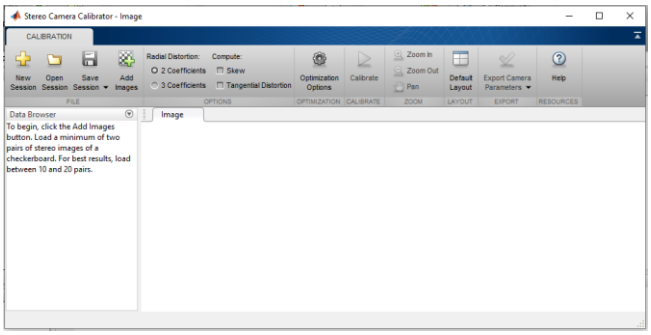



*Figure 4.1 Calibration Toolbox in MATLAB*

The first step to make is to input the calibration image set to the calibrator. This is done by clicking the Add Images icon then choosing the folders of the right and left calibration images and giving the real size of checkerboard square. This is depicted in Fig. 4.2

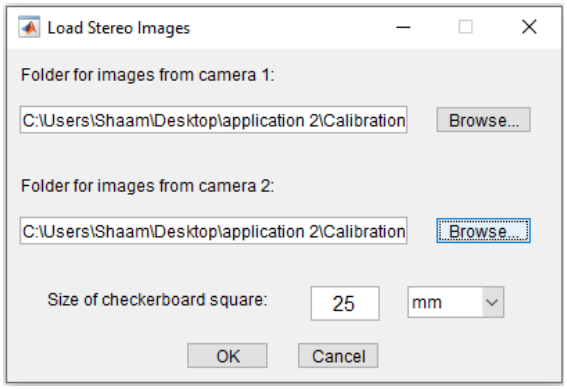



*Figure 4.2 Adding images to MATLAB calibrator*

The calibrator searches for the origin of the checkerboard and the squares' corners in each stereo frame. The calibrator gives the options to choose which parameters we are calibrating for in the

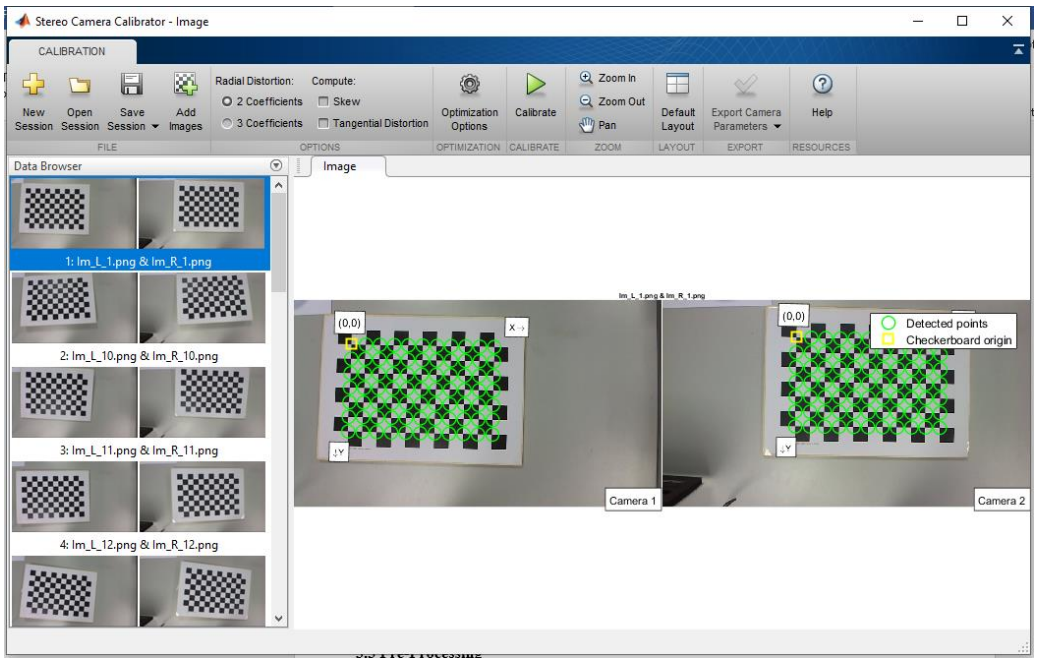



*Figure 4.3 Calibration dataset added to MATLAB Calibrator*

Option Field. In our case, we have only radial distortion with 3 parameters and the skew parameter is needed. To start calibrating, there is the Calibrate button. The outputs of the calibrator are the matrices and parameters of each camera, the extrinsic matrices of both cameras and the configuration of the system. The output can be exported to the workspace to use it.

**Rectification and Undistrtion:** After calibration, the intrinsic and extrinsic matrices are known but the stereo images of the system are not ready to use before rectification and removing the distortion. OpenCV provides functions to rectify and undistort the stereo images. First, cv2.stereoRectify is used to calculate the new camera matrices after rectification, then cv2.initUndistortRectifyMap is used whose output is a rectificating-undistorting map for each camera. Those maps are used to remap the stereo image frames to rectified and undistorted frames, that can be used in stereo matching.

### 4.1.2 Detection Loop

After the initialization and calculation of the rectification and undistortion maps, the detection algorithm starts looping. Fig. 4.4 shows the block diagram of the general detection algorithm. A brief summary of each step is first introduced, then each one is discussed separately.

The algorithm steps are the following:

1. **Stereo Capture:** Capturing the frame at this current moment. The frame here is defined as two images of the same scene from two different camera.
2. **Pre-Processing:** Represents camera distortion removal and rectification of the frame, in addition to filtration and noise removal.

3. **Stereo Matching:** Matching points from the left image with their correspondence from the right one. The output of this step is the disparity map.
4. **Post-Processing:** Filtering, smoothening, and morphological operations on the disparity map preparing it to be processed.
5. **Depth Calculation:** Calculating the depth of each pixel depending on the disparity map and the camera's parameters. The output of this step is the Depth Map.
6. **Segmentation:** the process of extracting areas with similar features from the image. In this case, extracting areas with similar disparity and considering suitable constraints to classify them as objects. The output of this step is the Obstacles List.

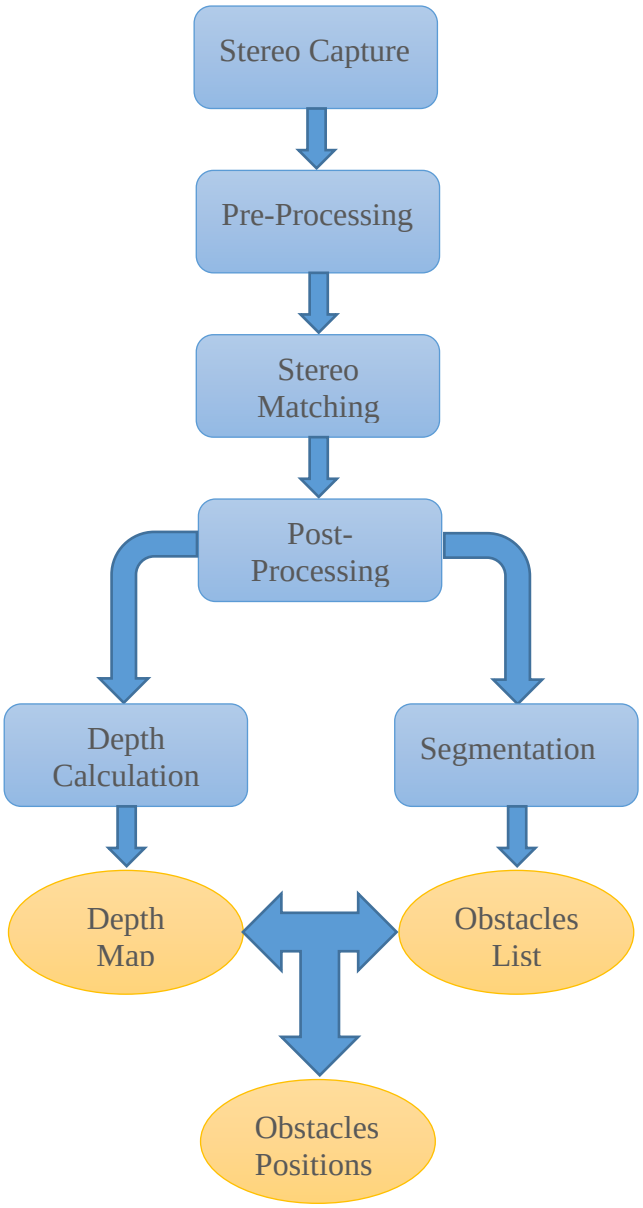



*Figure 4.4 General Detection Algorithm Block Diagram*

## 4.2 Obtaining Stereo Frames

First thing to do at each loop is to capture the stereo frames from the stereo camera. It is important to have a capture at the same instant to ensure that the images are of the same scene. The algorithm is supposed to capture the images from two cameras, but due to technical reasons, the hardware of the system is not available at the time of writing. An alternative to the camera hardware is to use a stereo images dataset. In this work, a stereo images dataset, called DrivingStereo, taken from a stereo camera on a car driving in a Chinese city is used as input frames.

## 4.3 Pre-Processing

This step aims to prepare the raw frames to be processed later. Pre-processing includes remapping the frame images using the rectification and undistortion map calculated in the initialization phase, as well as some filtration operations on the new images.

**Remapping the frames:** remapping is a crucial step in the process. The two images from the stereo camera can be not horizontally correspondent so corresponding points cannot be found. OpenCV provides a function to remap the original frame to the new one that improves the previously mentioned property of correspondent points. The function is cv2.remap() which takes the original image and a map as input and outputs the new image. In the case of study, the function must be used twice, one for each image, and each image has a different map.

The DrivingStereo dataset includes also the intrinsic and extrinsic matrices as well as the rectified and undistorted matrices. And fortunately, the images are processed before publishing to be undistorted and rectified from the creator. This has reduced the need of this step.

**Filtration:** Several filters were tested on the images; examples of the tested filters are:

- Bilateral Filter

- Gaussian Filter
- Median Filter
- Mean Filter

After looking for the best parameters of each filter and their kernel, it was noticed that the filtration was useless for the next step. For this reason, no filter was used.

## 4.4 Stereo Matching

In this step, four methods have been tested:

- Block Matching BM
- Block Matching SGBM
- Stereo BM with Weighted Least Squares filter
- Stereo SGBM with Weighted Least Squares filter

In this section, descriptions of the parameters used for each method are provided first. Then, discussion of the results is held.

### 4.4.1 Stereo BM

Stereo BM algorithm with parameters values presented in table 1 has been applied to dataset's images using OpenCV library in python. The code is presented in Appendix A.

*Table 4.1*

| Parameter | Value |
|---|---|
| NumDisparities | 10*16 |
| BlockSize | 21 |
| UniquenessRatio | 10 |
| SpeckleWindowSize | 50 |
| Disp12MaxDiff | 2 |
| SpeckleRange | 2 |
| TextureThreshold | 2000 |

Where according to OpenCV documentation [24] [25]:

- **NumDisparities:** is the disparity search range, which can be calculated as maximum disparity minus minimum disparity. The value is always greater than zero. In the current implementation, this parameter must be divisible by 16. For each pixel, algorithm will find the best disparity from 0 (default minimum disparity) to NumDisparities. The search range can be shifted by changing the value of the minimum disparity.
- **BlockSize:** is the linear size of the blocks compared by the algorithm. The size should be odd (as the block is centered at the current pixel). Larger block size implies smoother, though less accurate disparity map. Smaller block size gives more detailed disparity map, but there is higher chance for algorithm to find a wrong correspondence.
- **Disp12MaxDiff:** is the maximum allowed difference (in integer pixel units) in the left-right disparity check.

- **UniquenessRatio:** is margin in percentage by which the best (minimum) computed cost function value should win the second-best value to consider the found match correct. Normally, a value within the 5-15 range is good enough.
- **SpeckleWindowSize:** is the maximum size of smooth disparity regions to consider their noise speckles and invalidate.
- **SpeckleRange:** is the maximum disparity variation within each connected component. Setting this parameter to a positive value will apply speckle filtering. Its value will be implicitly multiplied by 16. Normally, 1 or 2 is good enough.
- **TextureThreshold:** Filters out areas that do not have enough texture information for reliable matching.

The parameters values are tuned manually to get the suitable output.

### 4.4.2 Stereo SGBM

Stereo SGBM algorithm with parameters values presented in table 2 has been applied to dataset's images using OpenCV library in python. The code is presented in appendix A.

*Table 4.2*

| Parameter | Value |
|---|---|
| MinDisparity | 0 |
| NumDisparities | 11*16 |
| BlockSize | 7 |
| UniquenessRatio | 5 |
| SpeckleWindowSize | 200 |
| Disp12MaxDiff | 0 |
| SpeckleRange | 2 |
| PreFilterCap | 63 |
| P1 | 8*1*7*7 |
| P2 | 32*1*7*7 |
| Mode | cv2.STEREO_SGBM_MODE_SGBM_3WAY |

Where according to OpenCV documentation [25]:

- **P1:** is the first parameter controlling the disparity smoothness. It represents the penalty on the disparity change by plus or minus 1 between neighbor pixels. The larger its value is, the smoother the disparity is.

  In this project, the value used for P1 can be described as:

  $$P1 = 8 * number\ of\ channels * blocksize\text{^}2$$

- **P2:** is the second parameter controlling the disparity smoothness. It represents the penalty on the disparity change by more than 1 between neighbor pixels. The larger its value is, the smoother the disparity is.

In this project, the value used for P2 can be described as:

$$P2 = 8 * number\ of\ channels * blocksize\text{^}2$$

- **PreFilterCap:** Truncation value for the prefiltered image pixels. The algorithm first computes x-derivative at each pixel and clips its value by [-preFilterCap, preFilterCap] interval. The result values are passed to the Birchfield-Tomasi pixel cost function.
- **MinDisparity:** is the minimum possible disparity value. Normally, it is zero but sometimes rectification algorithms can shift images, so this parameter needs to be adjusted accordingly.

The parameters values are tuned manually to get the suitable output.

### 4.4.3 Disparity Weighted Least Squares filter (WLS)

It is a disparity map filter based on Weighted Least Squares filter (in form of Fast Global Smoother that is a lot faster than traditional Weighted Least Squares filter implementations) and optional use of left-right-consistency-based confidence to refine the results in half-occlusions and uniform areas. According to OpenCV documentation, it has two parameters which are:

- **Lambda:** is a parameter defining the amount of regularization during filtering. Larger values force filtered disparity map edges to adhere more to source image edges. The typical value for this parameter is 8000.
- **SigmaColor:** is a parameter defining how sensitive the filtering process is to source image edges. Large values can lead to disparity leakage through low-contrast edges. Small values can make the filter too sensitive to noise and textures in the source image. Typical values of it range from 0.8 to 2.0.

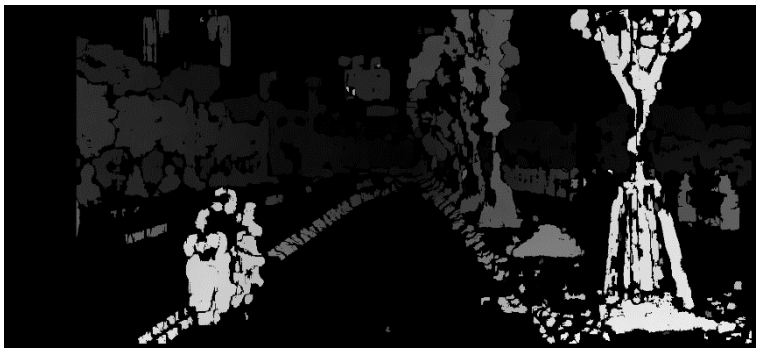
WLS filter was applied to both matching algorithms mentioned above. The parameters values were tuned manually to get the most appropriate output. The chosen parameters are presented in *Table 3.3*

*Table 4.3*

| Parameter | Value |
|---|---|
| Lambda | 8000 |
| SigmaColor | 1.5 |

The results of applying the mentioned method are presented below with the original image taken by the left camera. The black pixels in the disparity maps represent invalid disparities.

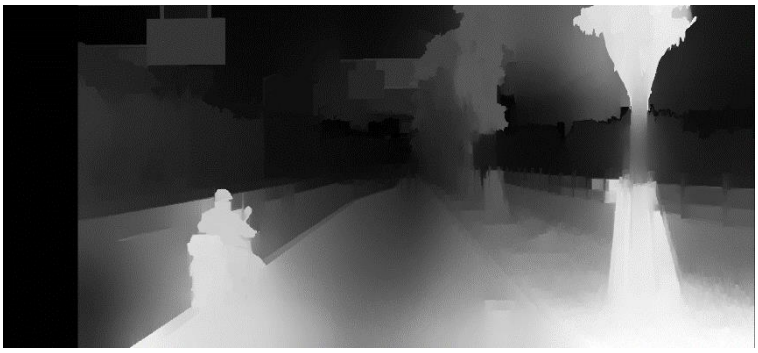
*(a)*



*(b)*

*(c)*

*(d)*



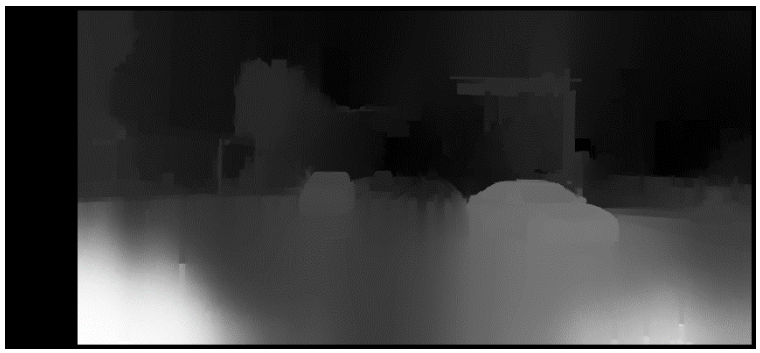
*(e)*

*Figure 4.5 (a) Original scene 1 (b) Disparity map 1 produced by BM (c) Disparity map 1 produced by SGBM (d) Disparity map 1 produced by BM with WLS (e) Disparity map 1 produced by SGBM with WLS*

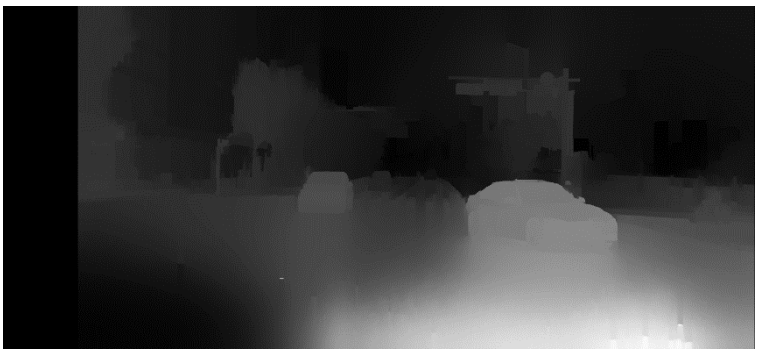
*(a)*

*(b)*

*(c)*



*(d)*

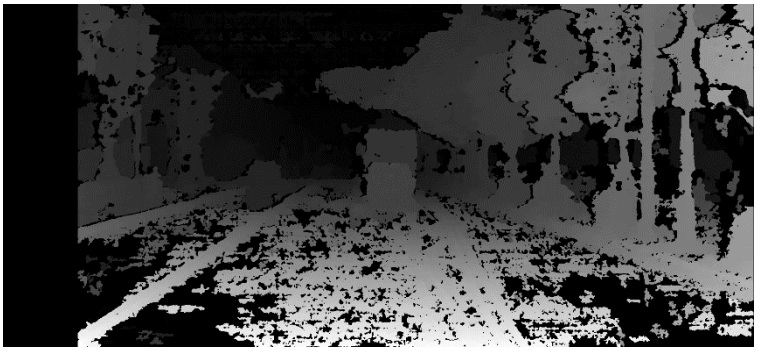


*(e)*

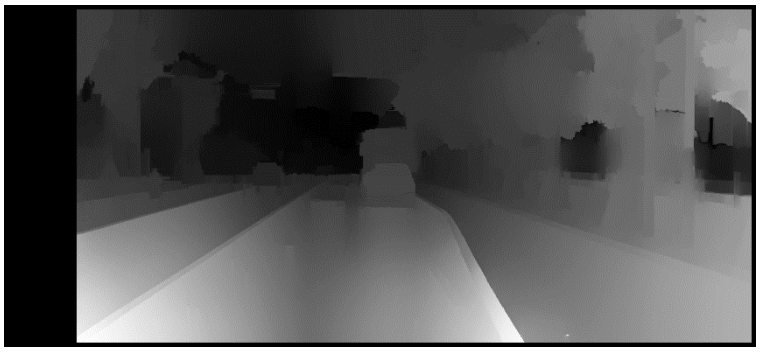
*Figure 4.6 (a) Original scene 2 (b) Disparity map 2 produced by BM (c) Disparity map 2 produced by SGBM (d) Disparity map 2 produced by BM with WLS (e) Disparity map 2 produced by SGBM with WLS*

*(a)*

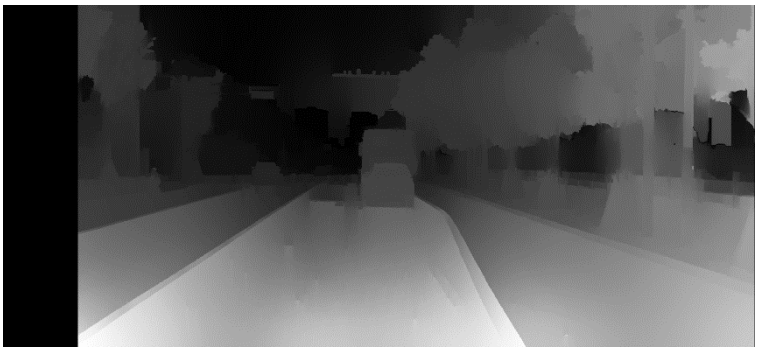
*(b)*



*(c)*

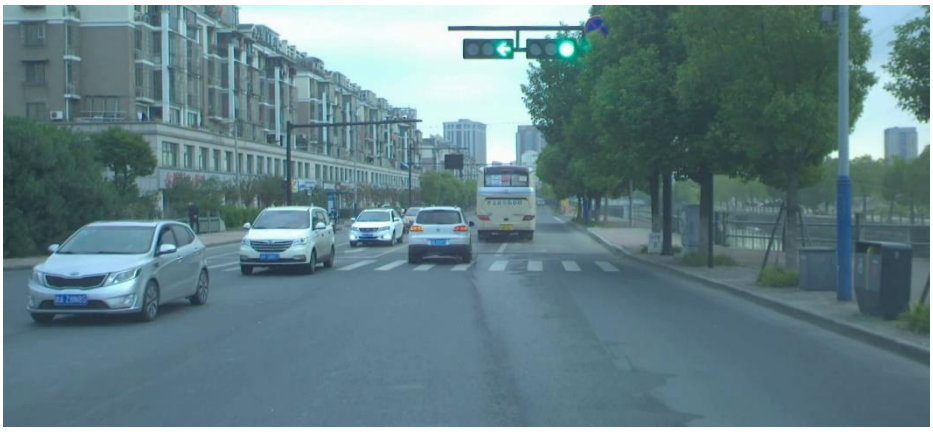


*(d)*

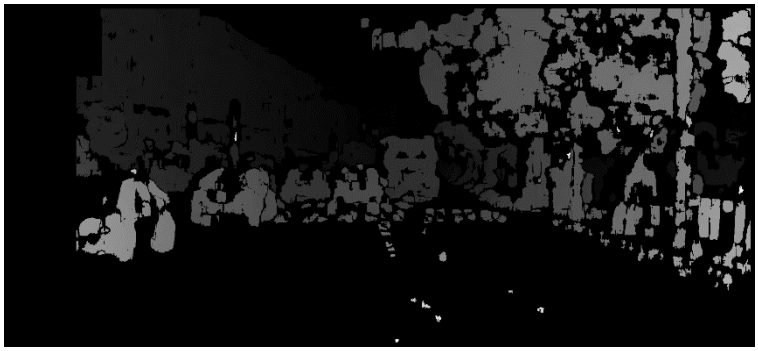


*(e)*

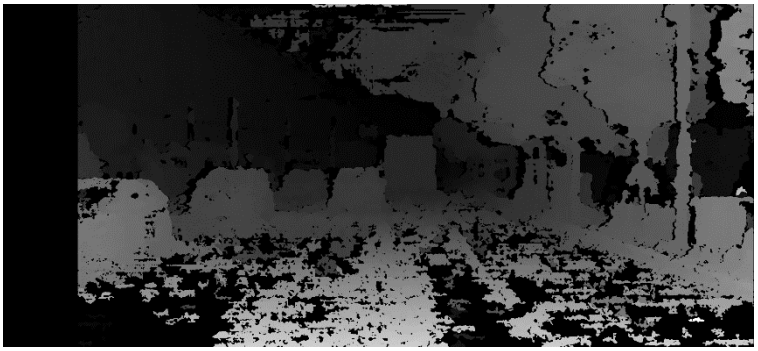
*Figure 4.7(a) Original scene 3 (b) Disparity map 3 produced by BM (c) Disparity map 3 produced by SGBM (d) Disparity map 3 produced by BM with WLS (e) Disparity map 3 produced by SGBM with WLS*



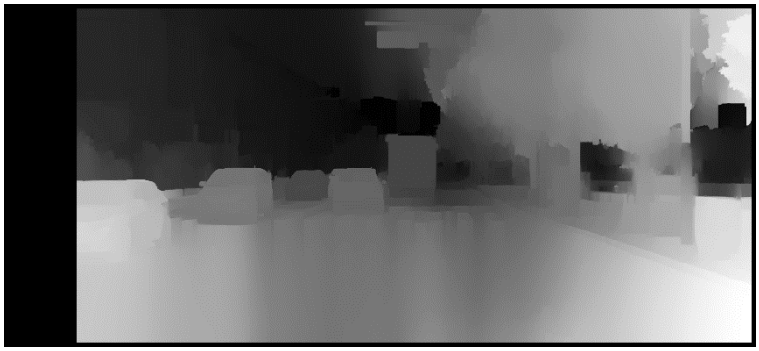
*(a)*



*(b)*



*(c)*



*(d)*

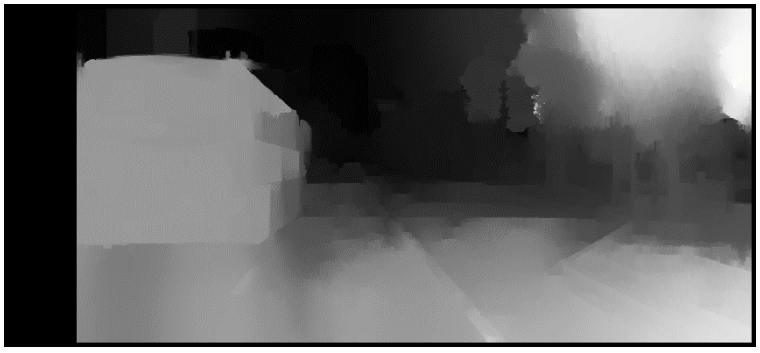
*(e)*

*Figure 4.8 (a) Original scene 4 (b) Disparity map 4 produced by BM (c) Disparity map 4 produced by SGBM (d) Disparity map 4 produced by BM with WLS (e) Disparity map 4 produced by SGBM with WLS*

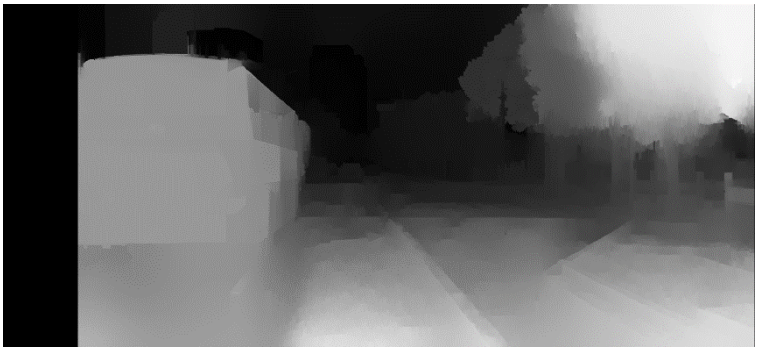

*(a)*

*(b)*

*(c)*

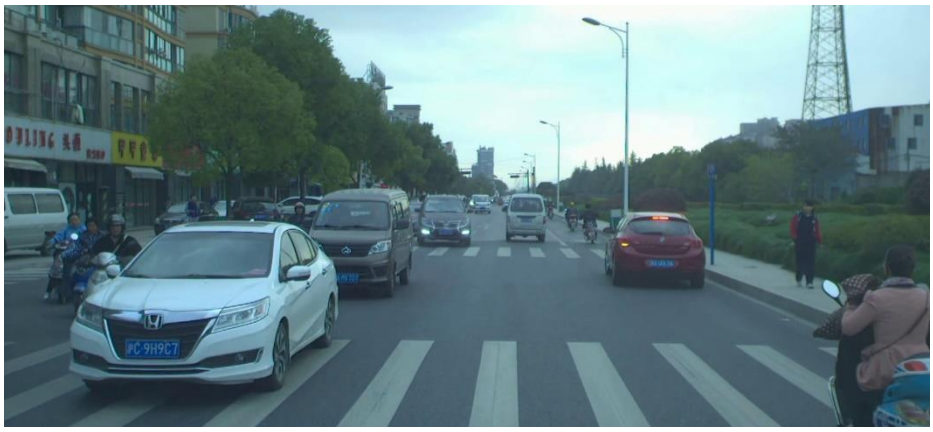


*(d)*



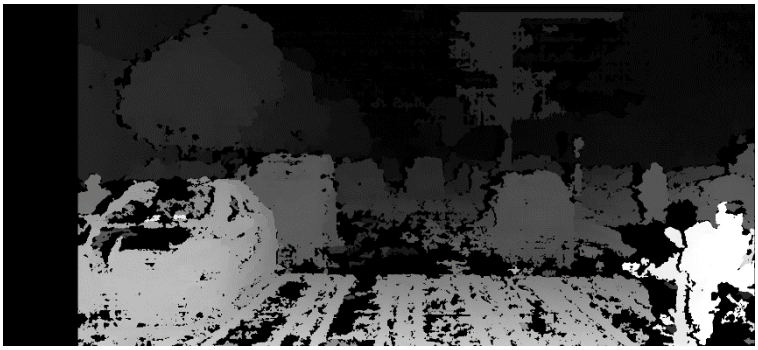
*(e)*

*Figure 4.9 (a) Original scene 5 (b) Disparity map 5 produced by BM (c) Disparity map 5 produced by SGBM (d) Disparity map 5 produced by BM with WLS (e) Disparity map 5 produced by SGBM with WLS*

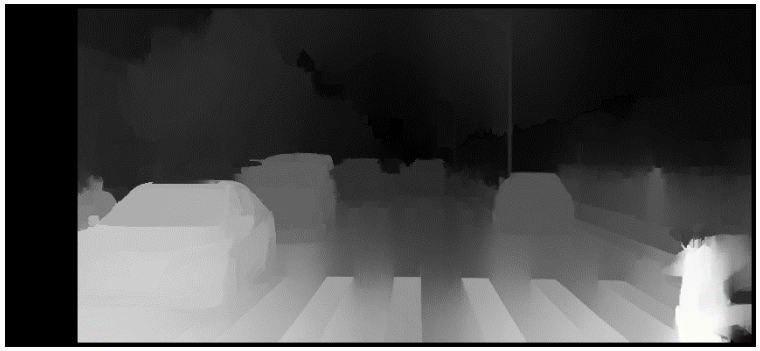


*(a)*

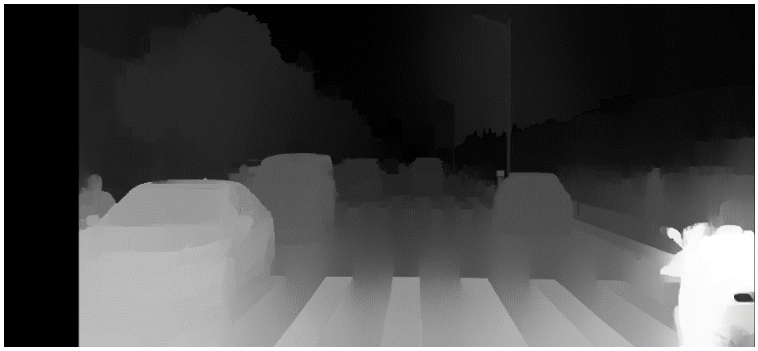
*(b)*



*(c)*



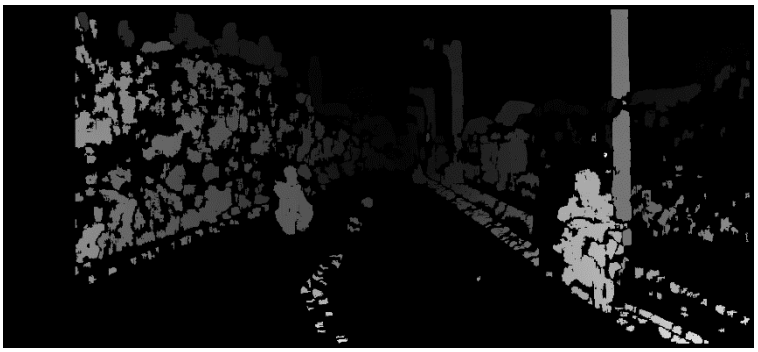
*(d)*



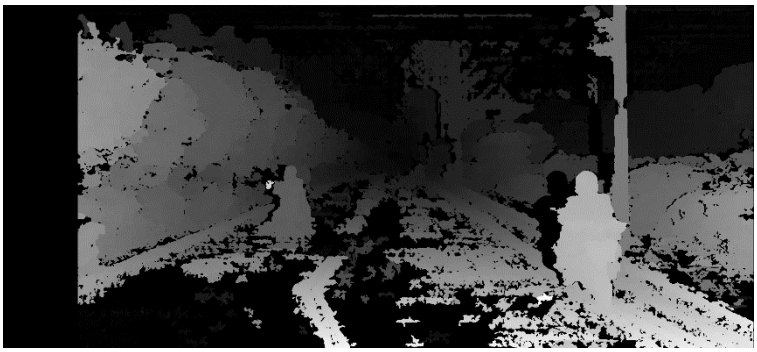
*(e)*

*Figure 4.10 (a) Original scene 6 (b) Disparity map 6 produced by BM (c) Disparity map 6 produced by SGBM (d) Disparity map 6 produced by BM with WLS (e) Disparity map 6 produced by SGBM with WLS*

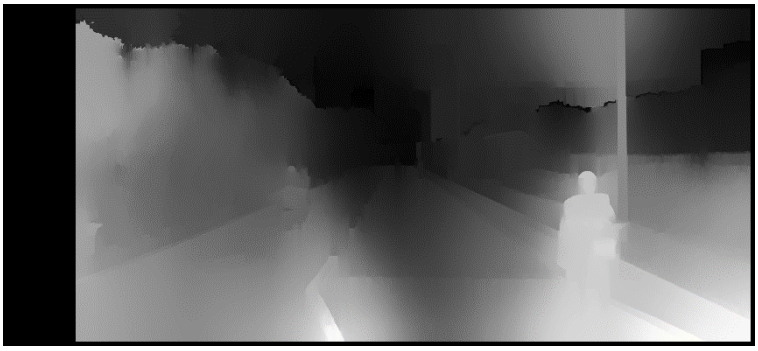
*(a)*



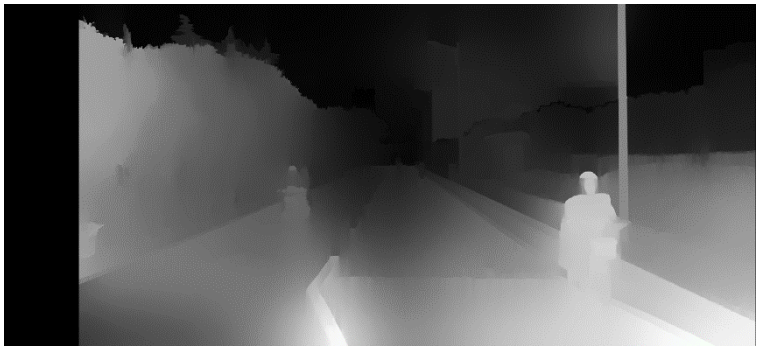
*(b)*



*(c)*



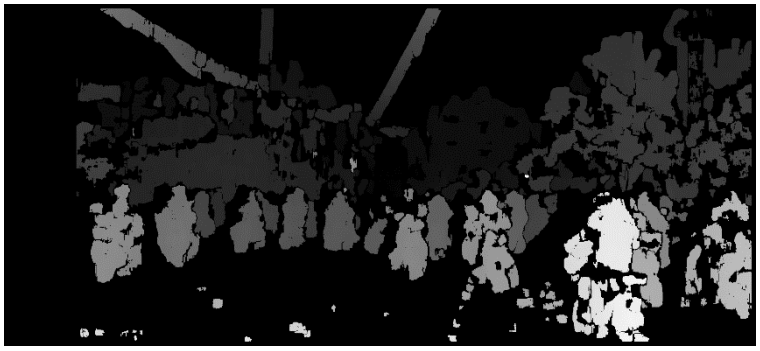
*(d)*



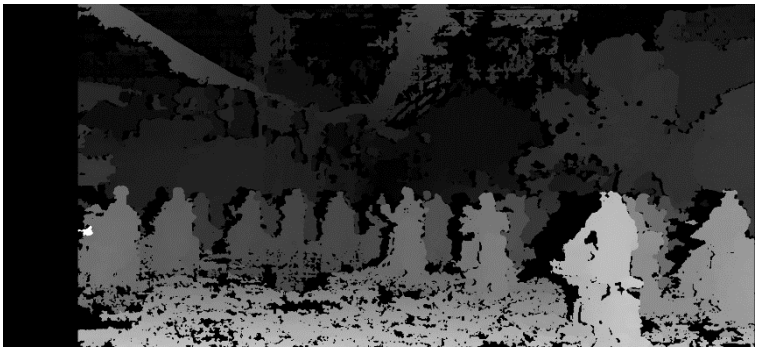
*(e)*

*Figure 4.11 (a) Original scene 7 (b) Disparity map 7 produced by BM (c) Disparity map 7 produced by SGBM (d) Disparity map 7 produced by BM with WLS (e) Disparity map 7 produced by SGBM with WLS*

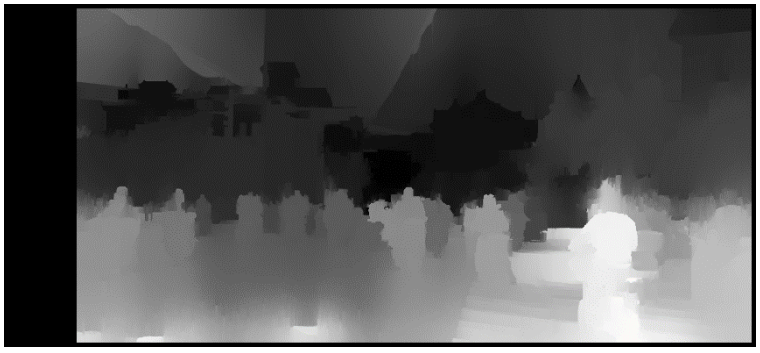
*(a)*



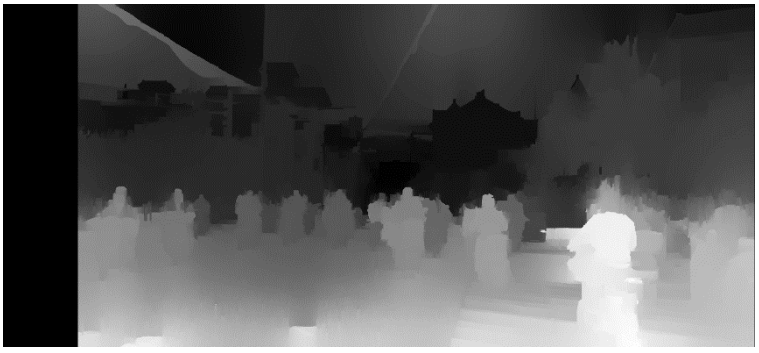
*(b)*



*(c)*



*(d)*



*(e)*

*Figure 4.12 (a) Original scene 8 (b) Disparity map 8 produced by BM (c) Disparity map 8 produced by SGBM (d) Disparity map 8 produced by BM with WLS (e) Disparity map 8 produced by SGBM with WLS*

According to the presented images, BM algorithm didn’t detect texture less areas. Hence, road pixels that didn’t contain lane lines were black. Additionally, parts of the same body weren’t connected. SGBM algorithm was better in matching road pixels compared to BM.

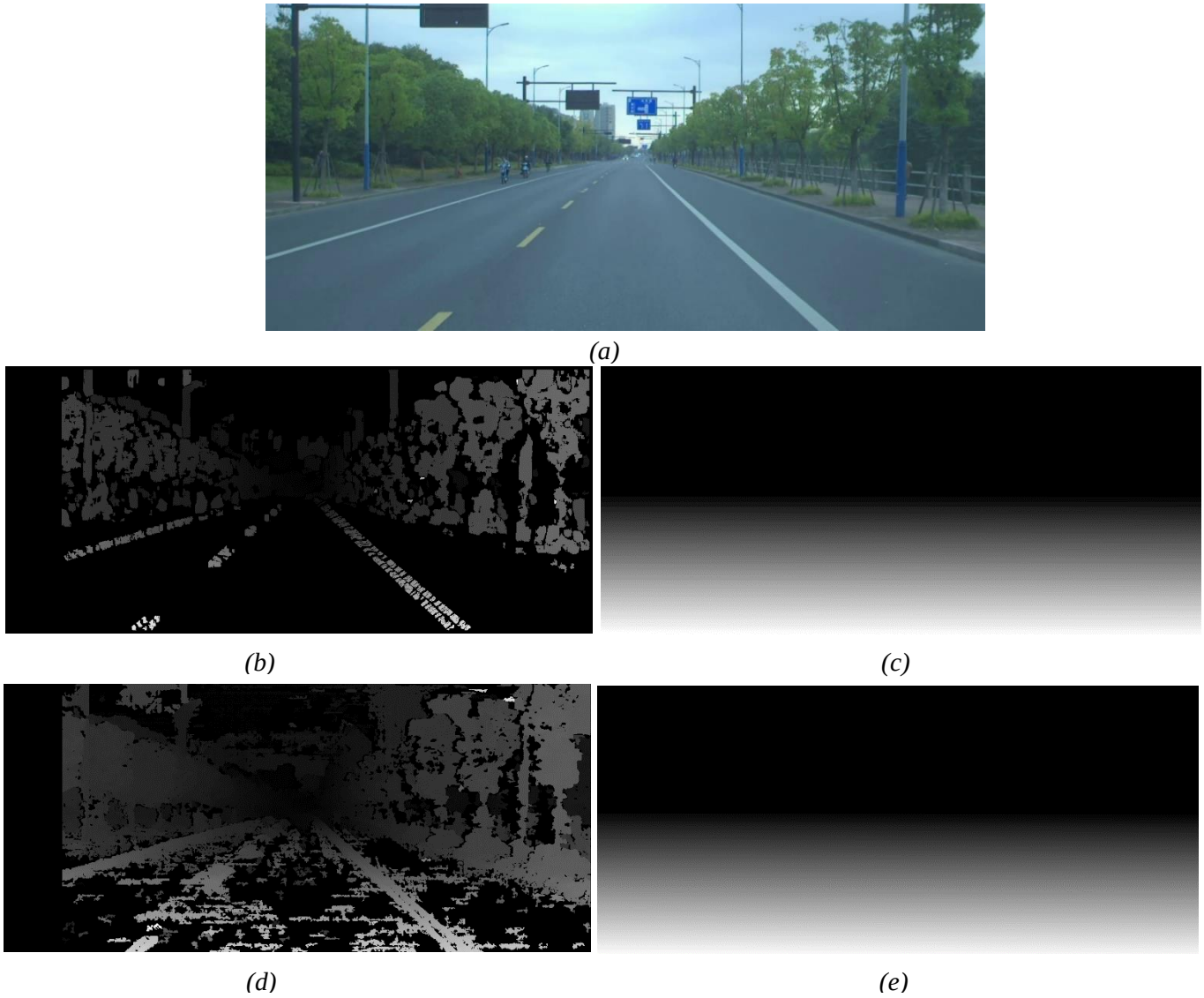
The purpose of applying WLS was to fill the gaps presented in the disparity maps and to get a smooth image. Nevertheless, the disparity values weren’t distributed gradually with respect to distance from camera, which means that the depth values won’t be correct.

## 4.5 Post-Processing

### 4.5.1 Road Removal

This step aims to isolate the obstacles in the image. This can be done by defining the free space and then subtracting it from the image. To do so, a disparity map of the ground plane is needed. The ground plane can be defined by using the lane equation, which is supposed to be provided from a lane detection algorithm. Since lane detection isn't the main scope of this work, the ground is assumed to be horizontal flat, and the lane equation has been calculated manually from the dataset.

The calculation of lane equation was done by taking two points' coordinates of the lane line in the disparity map generated by each matching algorithm. The ground disparity map is generated according to the disparity values of points belonging to the lane line with taking into consideration radial distortion effect.



*(a)* *(b)* *(c)* *(d)* *(e)*

*Figure 4.13 (a) Original scene (b) Disparity map produced by BM (c) Disparity map of the ground plane produced by BM (d) Disparity map produced by SGBM (e) Disparity map of the ground plane produced by SGBM*

To isolate the obstacles in each image, the generated ground disparity map was subtracted from the image disparity map resulted from the previous step. All the experiments were performed on 12th Gen Intel(R) Core (TM) i7-1255U CPU, 1.70 GHz, 16 GB RAM. The resulting images are shown below.

*(a)*

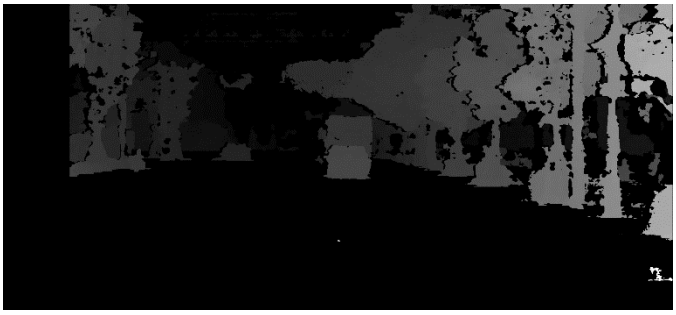
*(b)*

*(c)*

*Figure 4.14 (a) Original scene (b) Road removal on disparity map produced by BM (c) Road removal on disparity map produced by SGBM*

*(a)*

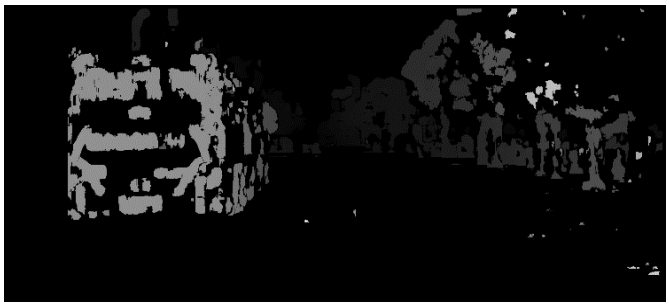
*(b)*



*(c)*

*Figure 4.16 (a) Original scene (b) Road removal on disparity map produced by BM (c) Road removal on disparity map produced by SGBM*

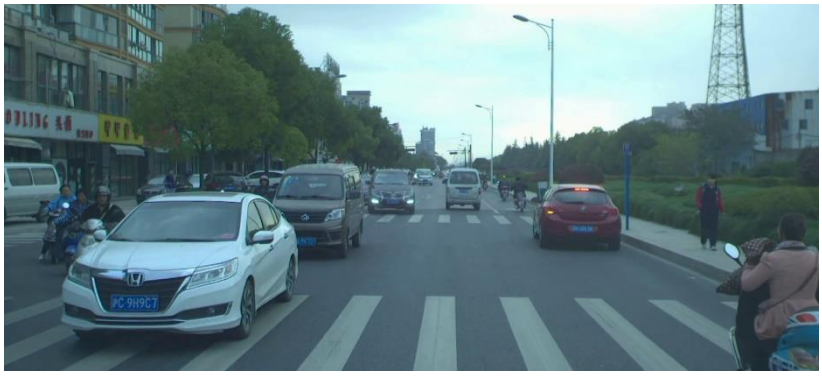

*(a)*



*(b)*

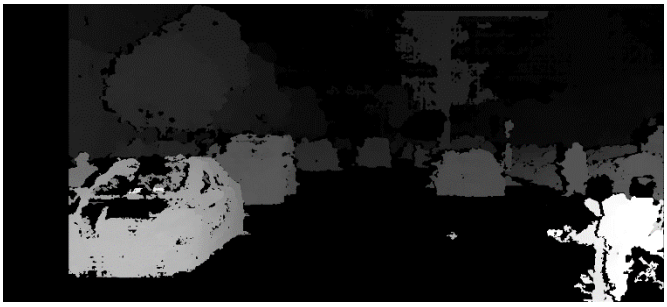
*(c)*

*Figure 4.15 (a) Original scene (b) Road removal on disparity map produced by BM (c) Road removal on disparity map produced by SGBM*



*(a)*

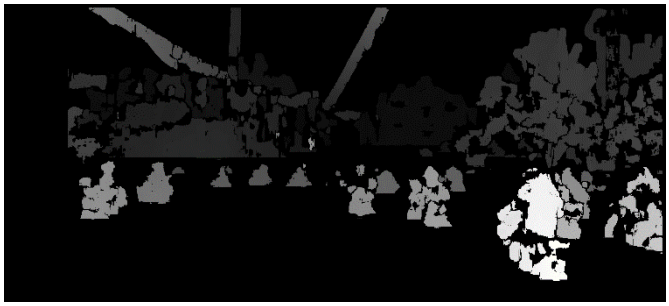
*(b)*



*(c)*

*Figure 4.17 (a) Original scene (b) Road removal on disparity map produced by BM (c) Road removal on disparity map produced by SGBM*

*(a)*



*(b)*

*(c)*

*Figure 4.18 (a) Original scene (b) Road removal on disparity map produced by BM (c) Road removal on disparity map produced by SGBM*

According to the presented images:

SGBM has shown better performance with respect to connecting the parts of the same body, the number of invalid disparities, and preserving obstacles' boundaries. On the other hand, some random shaped disparity values are still presented on the road.

With respect to the computational cost, that is assessed by measuring for each method the execution time needed to process a stereo pair on the same used machine, BM has shown faster computational time as presented in *Figure 4.19*.

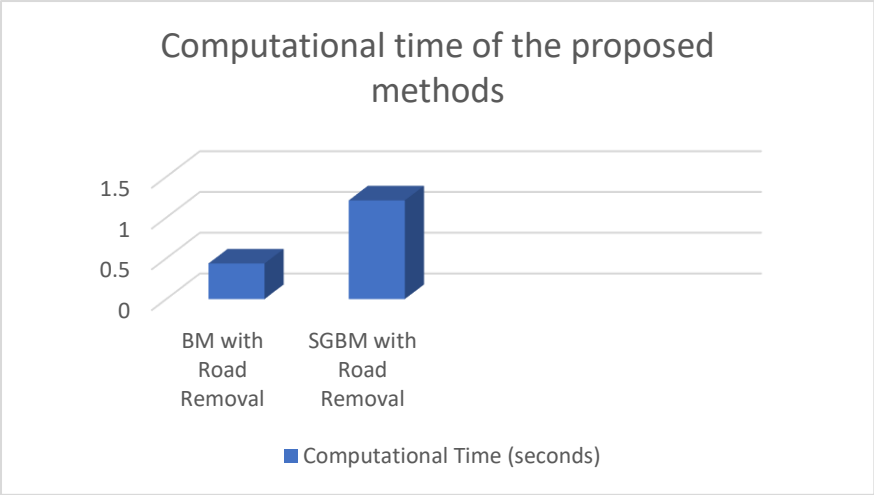



*Figure 4.19 Computational time of the proposed methods*

### 4.5.2 Filtering and smoothening

Morphological operations; opening and closing, were applied on the disparity map after applying road removal in order to:

- remove the noise presented in the disparity map, especially for enhancing the road removal quality.
- Fill the holes presented in the bodies.

## 4.6 Depth Calculation

The depth value of each pixel is related to its disparity, the focal length, and the baseline. While the baseline of the used dataset was provided, the focal length wasn't available. When a stereo camera is available, the focal length can be calculated from the camera's matrices.

# Results

This chapter is for presenting the results of the practical side of this project. The results are shown while applying SGBM algorithm with the previously mentioned parameters, followed by the proposed road removal approach on the same original scenes presented in the previous chapter. The output of the proposed methodology is used to isolate the obstacles from the free space by using OpenCV functions (See Appendix A).

**It is important to notice that no Lane Detection algorithm was implemented, this causes detecting obstacles on the sidewalks or sometimes detecting sidewalks as a free space.** Implementing a Lane Detection algorithm will specify a Region of Interest (ROI) to consider. The ROI is the road in between the Road Lines, which means that it excludes the sidewalks from the searching area so that they are not considered as a free space.

The following images shows the results of the proposed method in obstacle detection.

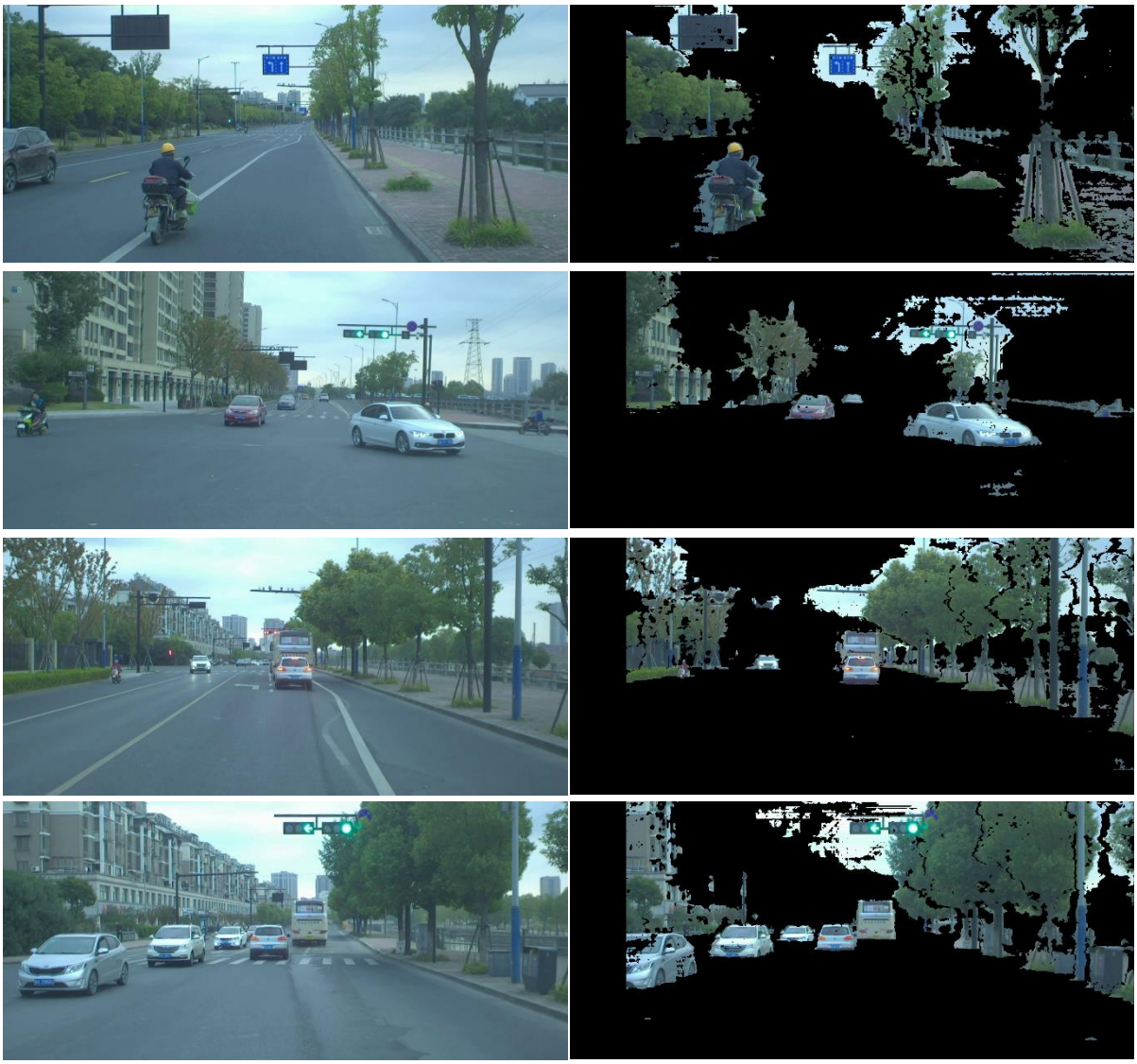


*Figure 5.1 Isolated Obstacles 1*

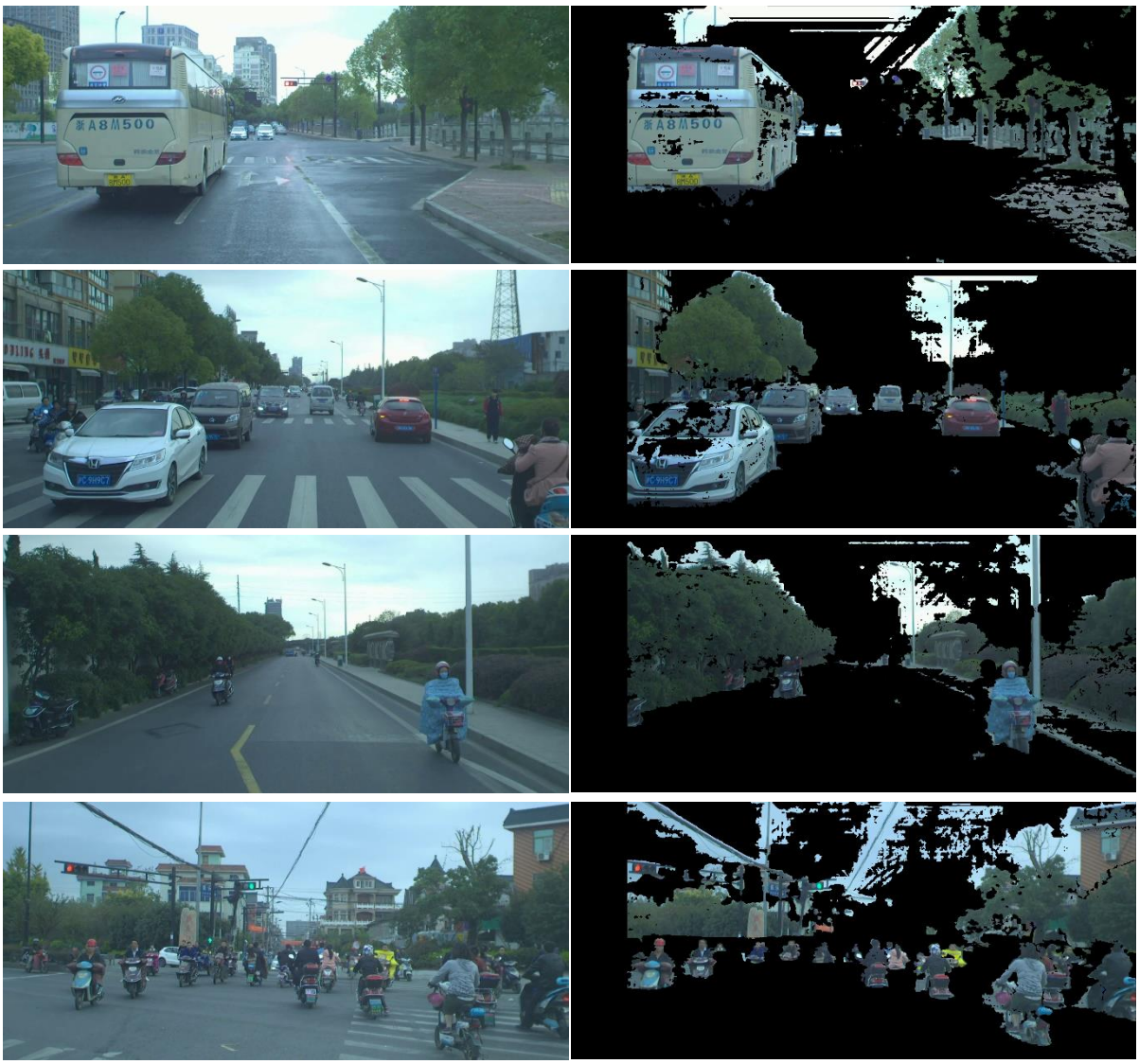


*Figure 5.2 Isolated Obstacles 2*

# Conclusion and Future Works

This project has proposed a methodology for distinguishing between the free drivable space and the obstacles in front of the autonomous car based on a stereo vision. The work was implemented on DrivingStereo dataset that provides a subset of images taken by a stereo camera mounted on the car in a cloudy weather. The aim of the project was achieved through two main processes: Stereo Matching and a state-of-the-art Road Removal.

In order to improve the output of the project, the following approaches can be done in the future:

- Implementing multiple stereo cameras together to provide a 360 view of the car's surroundings and applying the proposed algorithm on it.
- Implementing a sky removal approach.
- When the project's hardware is available, ground truth disparity map will be calculated.
- Applying a deep learning approach for segmentation, which will give a list of obstacles IDs combined with their (x, y, z) coordinates.

# References


[1] H. Cui, Y. Li and J. Liu, "An Obstacle Detection Algorithm Based on Ultrasonic Sensors for Autonomous Land Vehicle," in *International Conference on Mechatronics, Control and Automation Engineering (MCAE 2016)*, 2016.

[2] S. e. a. Campbell, "Sensor technology in autonomous vehicles: a review," in *2018 29th Irish Signals and Systems Conference (ISSC)*, 2018.

[3] E. Tondin Ferreira Dias, H. Vieira Neto and F. Schneider, "A Compressed Sensing Approach for Multiple Obstacle Localisation Using Sonar Sensors in Air," 2020.

[4] C. Blanc, R. Aufrere, L. Malaterre, J. Gallice and J. Alizon, "OBSTACLE DETECTION AND TRACKING BY MILLIMETER WAVE RADAR," in *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, Lisboa, 2004.

[5] J. Dickmann, J. Klappstein, M. Hahn, N. Appenrodt, H.-L. Bloecher, K. Werber and A. Sailer, "Automotive Radar the Key Technology For Autonomous Driving: From Detection and Ranging to Environmental Understanding," in *2016 IEEE Radar Conference (RadarConf)*, 2016.

[6] S. Sun, A. P. Petropulu and H. V. Poor, "MIMO Radar for Advanced Driver-Assistance Systems and Autonomous Driving: Advantages and challenges," *IEEE SIGNAL PROCESSING MAGAZINE*, pp. 110-112, 2020.

[7] D. Yeong, G. Velasco-Hernandez, J. Barry and J. Walsh, "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review," *Sensors*, pp. 11-13, 2021.

[8] P. Wang, "Research on Comparison of LiDAR and Camera in Autonomous Driving," *Journal of Physics: Conference Series*, vol. 2093, no. 2021 International Conference on Mechanical Automation and Electronic Information Engineering (MAEIE 2021), p. 012032, 2021.

[9] J. Han, D. Kim, M. Lee and M. Sunwoo, "Enhanced Road Boundary and Obstacle Detection Using a Downward-Looking LIDAR Sensor," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 971-985, 2012.

[10] S. Badrloo, M. Varshosaz, S. Pirasteh and J. Li, "Image-Based Obstacle Detection Methods for the Safe Navigation of Unmanned Vehicles: A Review," *Remote Sensing*, vol. 14, no. 15, p. 3824, 2022.

[11] J. Woo, J.-H. Baek, S.-H. Jo, S. Kim and J.-H. Jeong, "A Study on Object Detection Performance of YOLOv4 for Autonomous Driving of Tram," *Sensors*, vol. 22, p. 9026, 2022.

[12] B. Li, Y. Chen, H. Xu and F. Zhong, "Fast vehicle detection algorithm based on lightweight YOLO7-tiny," *arXiv [2304.06002]*, p. , 2023.

[13] J. Cohen, "A Look at Tesla's Occupancy Networks," Think Autonomous, 12 September 2022. [Online]. Available:

https://www.thinkautonomous.ai/blog/occupancy-networks/. [Accessed 24 July 2023].

[14] N. Appiah and N. Bandaru, "Obstacle detection using stereo vision for self-driving cars," Stanford University, 2015.

[15] J. Young and M. Simic, "LIDAR and monocular based overhanging obstacle detection," *Procedia Computer Science*, vol. 60, p. 1423–1432, 2015.

[16] R. Klette, Concise Computer Vision: An Introduction into Theory and Algorithms, London: Springer, 2014.

[17] K. Ikeuchi, Computer Vision, a Reference Guide, New York: Springer, 2014.

[18] R. Y. TSAI, "A Versatile Camera Calibration Techniaue for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses," *IEEE JOURNAL OF ROBOTICS AND AUTOMATION*, vol. 4, pp. 323-344, AUGUST 1987.

[19] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010.

[20] M. Sonka, V. Hlavac and R. Boyle, Image Processing, Analysis, and Machine Vision, Stamford: Cengage Learning, 2014.

[21] G. v. Rossum, "Python Tutorial Release 3.7.0," Python Software Foundation, 2018.

[22] MathWorks, "MATLAB," The MathWorks, Inc., [Online]. Available: https://www.mathworks.com/products/matlab.html?s_tid=hp_ff_p_matlab. [Accessed 24 July 2023].

[23] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi and B. Zhou, "DrivingStereo: A Large-Scale Dataset for Stereo Matching in Autonomous Driving Scenarios," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, 2019.

[24] OpenCV, "OpenCV," OpenCV, 28 June 2023. [Online]. Available: https://docs.opencv.org/4.8.0/d9/dba/classcv_1_1StereoBM.html. [Accessed 20 July 2023].

[25] "OpenCV," 19 July 2023. [Online]. Available: https://docs.opencv.org/4.x/d2/d85/classcv_1_1StereoSGBM.html. [Accessed 20 July 2023].

[26] Mathworks, "What is Camera Calibration?," Mathworks, 2022. [Online]. Available: https://nl.mathworks.com/help/vision/ug/camera-calibration.html. [Accessed 29 May 2023].

# Appendix A

Project's Documentation

https://drive.google.com/drive/folders/1CHHz7Jjb3wFQbPmAs624oWr52Zhk9m6n?usp=drive_link

Project's Link

https://drive.google.com/drive/folders/13hvzKUffD9hN9XLu0OwuiyH0cudlHleD?usp=sharing

Stereo Matching with Road Removal

https://drive.google.com/drive/folders/1F7JuCDoVHQEmRIdnUPZqwJpVD58L-u1u?usp=drive_link

Generating Ground Disparity

https://drive.google.com/drive/folders/1gqOyi2mmHBLutQja66xcRDJiw2tbmzWF?usp=drive_link

Obstacles Isolation

https://drive.google.com/drive/folders/1YRqzmfDr7uBWshhFwkA04NUBR2-5-9w3?usp=drive_link