TECHNICAL UNIVERSITY OF CRETE

DIPLOMA THESIS

Design and Implementation of an Embedded System for Stereo Vision from a Single Camera

---

Author:
DIMITRIOS NTOUNETAS

Thesis Committee:
PROF. APOSTOLOS DOLLAS
ASSOC. PROF. SOTIRIOS IOANNIDIS
PROF. IOANNIS PAPAEFSTATHIOU (AUTH)

A thesis submitted in fulfillment of the requirements for the diploma of Electrical and Computer Engineer

in the

School of Electrical and Computer Engineering
Microprocessor and Hardware Laboratory

February 13, 2024

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

Design and Implementation of an Embedded System for Stereo Vision from a Single Camera

by DIMITRIOS NTOUNETAS

3D Vision has been a topic of investigation for numerous decades, with researchers striving to overcome the challenge of losing the third dimension when using images. Over the years, various techniques have been devised to capture depth from images, including stereo vision, fringe projection, laser scanning, or a combination of these methods. While these techniques have demonstrated impressive results, they may not be suitable for certain applications due to factors such as high cost. Consequently, there is a pressing need to develop a 3D vision system that is affordable, fast, and capable of producing high-resolution output. This thesis implements an innovative approach to estimate depth, initially studied by G. Rematska in her Master Thesis. Depth is calculated by using a single camera in conjunction with two spectrally distinct light sources. The light sources consist of two sets of LED arrays, and depth information can be extracted by analyzing the different reflections captured by the camera from these two light sources. The key aspect of depth estimation lies in the blue to red ratio, which can be correlated to depth through appropriate calibration and processing. In this thesis, the goal was to evolve a system which was verified under laboratory conditions (controlled lighting, no camera vibrations, completely fixed distance of camera to target surface) into one suitable for field use, in which the conditions (lighting, camera vibrations, distance to target surface) are widely varied. The methodology was verified using MATLAB and subsequently implemented as a real time embedded system. The resulting system was thoroughly tested in the field and operates in real time and is fully optimized to minimize hardware resource usage while maintaining a high frequency of operation for processing high-resolution images.

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

Design and Implementation of an Embedded System for Stereo Vision from a Single Camera

by DIMITRIOS NTOUNETAS

Η τρισδιάστατη όραση αποτελεί αντικείμενο έρευνας εδώ και πολλές δεκαετίες, με τους ερευνητές να προσπαθούν να ξεπεράσουν την πρόκληση της απώλειας της τρίτης διάστασης κατά τη χρήση εικόνων. Με την πάροδο των ετών, έχουν επινοηθεί διάφορες τεχνικές για τη σύλληψη του βάθους από εικόνες, συμπεριλαμβανομένης της στερεοσκοπικής όρασης, της προβολής κροσσών, της σάρωσης με λέιζερ ή ενός συνδυασμού αυτών των μεθόδων. Ενώ οι τεχνικές αυτές έχουν επιδείξει εντυπωσιακά αποτελέσματα, ενδέχεται να μην είναι κατάλληλες για ορισμένες εφαρμογές λόγω παραγόντων όπως το υψηλό κόστος. Κατά συνέπεια, υπάρχει επιτακτική ανάγκη να αναπτυχθεί ένα σύστημα τρισδιάστατης όρασης που να είναι προσιτό, γρήγορο και ικανό να παράγει αποτελέσματα υψηλής ανάλυσης. Η παρούσα διατριβή εφαρμόζει μια καινοτόμο προσέγγιση για την εκτίμηση του βάθους, η οποία μελετήθηκε αρχικά από την Γ. Ρεματσκα στη μεταπτυχιακή της διατριβή. Το βάθος υπολογίζεται με τη χρήση μιας μόνο κάμερας σε συνδυασμό με δύο φασματικά διαφορετικές πηγές φωτός. Οι πηγές φωτός αποτελούνται από δύο σύνολα συστοιχιών LED και η πληροφορία βάθους μπορεί να εξαχθεί αναλύοντας τις διαφορετικές ανακλάσεις που καταγράφονται από την κάμερα από αυτές τις δύο πηγές φωτός. Η βασική πτυχή της εκτίμησης του βάθους έγκειται στην αναλογία μπλε προς κόκκινο, η οποία μπορεί να συσχετιστεί με το βάθος μέσω κατάλληλης βαθμονόμησης και επεξεργασίας. Στην παρούσα διατριβή, ο στόχος ήταν να εξελιχθεί ένα σύστημα που επαληθεύτηκε σε εργαστηριακές συνθήκες (ελεγχόμενος φωτισμός, μηδενικές δονήσεις της κάμερας, εντελώς σταθερή απόσταση της κάμερας από την επιφάνεια του στόχου) σε ένα σύστημα κατάλληλο για χρήση στο πεδίο, όπου οι συνθήκες (φωτισμός, δονήσεις της κάμερας, απόσταση από την επιφάνεια του στόχου) ποικίλλουν ευρέως. Αρχικά, η μεθοδολογία επαληθεύτηκε με τη χρήση MATLAB και στη συνέχεια υλοποιήθηκε ως ενσωματωμένο σύστημα πραγματικού χρόνου. Το σύστημα που προέκυψε δοκιμάστηκε διεξοδικά και λειτουργεί σε πραγματικό χρόνο και είναι πλήρως βελτιστοποιημένο για την ελαχιστοποίηση της χρήσης πόρων υλικού, διατηρώντας παράλληλα υψηλή συχνότητα λειτουργίας για την επεξεργασία εικόνων υψηλής ανάλυσης.

Acknowledgements

I would like to express my heartfelt gratitude to all those who have contributed to the completion of this Diploma Thesis. This journey has been a collaborative effort, and I am thankful for the support and encouragement received along the way.

First and foremost, I extend my deepest appreciation to my thesis advisor Professor Apostolos Dollas for his invaluable guidance, insightful feedback, and unwavering support throughout the entire process. His expertise and mentorship have been instrumental in shaping the outcome of this work.

I am also indebted to my colleagues and peers who provided a stimulating academic environment, fostering meaningful discussions and the exchange of ideas. I especially want to thank my colleague Christos Spyridakis for the support he showed and the help he provided to overcome the challenges that arose in the system implementation. The collective efforts of the academic community have greatly enriched the quality of this thesis.

Furthermore, I extend my gratitude to my friends and family for their patience, understanding, and encouragement during the challenging moments of this academic endeavor. Their unwavering support has been a constant source of motivation.

This thesis stands as a testament to the collaborative spirit that defines academic pursuits, and I am sincerely grateful for the contributions of all those who have played a role, big or small, in its completion...

Contents

List of Figures

List of Tables

List of Abbreviations

3D	**3** **D**imensional
ARM	**A**dvanced **R**ISC **M**achine
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
CPU	**C**entral **P**rocessor **U**nit
CRI	**C**olor **R**endering **I**ndex
CS	**C**omputer **S**cience
DC	**D**irect **C**urrent
DDR4	**D**ouble **D**ata **R**ate type **4** memory
DRAM	**D**ynamic **R**andom **A**ccess **M**emory
DSLR	**D**igital **S**ingle **L**ens **R**eflex
ENIAC	**E**lectronic **N**umerical **I**ntegrator **A**nd **C**omputer
FHD	**F**ull **H**igh **D**efinition
FFMPEG	**F**ast **F**orward **M**oving **P**icture **E**xperts **G**roup
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
FPS	**F**rames **P**er **S**econd
FPU	**F**loating-**P**oint **U**nit
GDDR6	**G**raphics **D**ouble **D**ata **R**ate type **6** memory
GIL	**G**lobal **I**nterpeter **L**ock
GPIO	**G**eneral **P**urpose **I**nput / **O**utput
GPU	**G**raphic **P**rocessor **U**nit
HBM	**H**igh **B**andwidth **M**emory
HD	**H**igh **D**efinition
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
I2C	**I**nter-**I**ntegrated **C**ircuit
IO	**I**nput **O**utput
IoT	**I**nternet **o**f **T**hing
LED	**L**ight **E**mitting **D**iode
LPDDR4	**L**ow **P**ower **D**ouble **D**ata **R**ate type **4**
LUT	**L**ook **U**p **T**able
MPSoC	**M**ulti **P**rocessor **S**ystem **o**n **C**hip

OpenCV	**Open** Source **C**omputer **V**ision
OpenGL	**Open** **G**raphics **L**ibrary
OS	**O**perating **S**ystem
PL	**P**rogrammable **L**ogic
PS	**P**rocessing **S**ystem
RAM	**R**andom **A**ccess **M**emory
RGB	**R**ed **G**reen **B**lue Colorspace
SBC	**S**ingle-**B**oard **C**omputer
SDK	**S**oftware **D**evelopment **K**it
SoC	**S**ystem **o**n **C**hip
SPI	**S**erial **P**eripheral **I**nterface
SMD	**S**urface **M**ounted **D**evice
SSD	**S**olid **S**tate **D**rive
SSH	**S**ecure **Sh**ell
SVGA	**S**uper **V**ideo **G**raphics **A**rray
TDP	**T**hermal **D**esign **P**ower
TOF	**T**ime **O**f **F**light
UART	**U**niversal **A**synchronous **R**eceiver/**T**ransmitter
UV	**U**ltra **V**iolet
UNIVAC	**Univ**ersal **A**utomatic **C**omputer
USD	**U**nited **S**tates **D**ollar
USB	**U**niversal **S**erial **B**us
UVC	**U**sb **V**ideo **C**lass
V4L2	**V**ideo**4L**inux**2**
WUXGA	**W**idescreen **U**ltra E**x**tended **G**raphics **A**rray

To those who've shaped me into who I am today,
thank you for your guidance and support…

Chapter 1

Introduction

1.1 Problems Addressed In The Thesis

Upon reviewing various depth estimation methodologies for our specific focus on road surfaces, we encountered challenges in adapting existing approaches to address our needs. In this chapter we will delve into the analysis of different methods explored. Stereoscopic vision techniques are anticipated to face limitations, particularly in scenarios involving flat, monochromatic surfaces like roads, where the lack of distinct structures complicates the identification of corresponding pixels, leading to inaccurate disparity estimation. Furthermore, stereoscopic vision is hindered when used on a moving platform because the relative distance of the two cameras may change due to vibrations, creating many problems, because a stable relative distance and no relative rotation of the cameras is a basic requirement of disparity-based stereoscopic vision.

While laser scanners and range finders have been employed in 3D road surface modeling, their widespread use is hindered by high costs and limitations in providing accurate depth information without complicated software development. Fringe projection techniques show promise; however, their application on a moving platform introduces errors due to vibrations induced by the road.

In this thesis a real-time system is presented from the theoretical method initially studied in the Master's Thesis of G. Rematska in 2015[1]. The system is a real representation of this low cost method while trying to solve many practical problems associated with it and moving this concept from the lab to real road conditions.

Although our approach is still in the developmental stages, it offers a cost-effective solution with enough accuracy in depth estimation results, aiming to overcome the drawbacks and the high cost associated with other methodologies.

1.2 Basic Ideas

The basic idea of this thesis is to create a cost-effective real-time embedded system for road surface scanning by using an innovative technique. The system consists of three subsystems inside a single casing. The camera subsystem is a single RGB camera module responsible for depicting the road surface underneath the casing. The illumination subsystem consists of two monochromatic LED arrays, one red and one blue, in order to achieve the best spectrum isolation. The processing subsystem is a small microprocessor that can be easily found on the market. Finally all the subsystems are shielded with a baffle shield, so much from the noise created from external light source but also from the physical elements.

After processing the Red and Blue channels of the RGB frame acquired from the camera we can extract depth information from the examined surface by considering the color ratio of the two colors as indication of the respected slope and then recreating the 3D surface. The basic idea is that slopes facing each LED array have higher color ratio with the respective color of the array being dominant. By finding the correct analogy of color ratio and slope values we can calculate the real distance both fast and with great accuracy especially for a low cost system.

1.3 Scientific Contribution

The system described in this thesis provides the ground idea for creating a low-budget real-time depth estimation system in order to extract information from images or video. It uses a single camera as opposed to some of the existing systems using methodologies where two or more cameras are used. Furthermore, comparing it to expensive laser scanning range finders this system is extremely low cost and easy to implement. Results have shown that the proposed methodology can detect depth of one millimeter or more [1].

1.4 Thesis Outline

- **Chapter 2 - Theoretical Background:** This chapter has a brief reference to what is an embedded system, the history of embedded systems, techniques and systems already in use according to the literature used for depth and distance estimation.
- **Chapter 3 - Proposed System Modeling:** This chapter has the system model, the technique verification process, and implementation code speed issues.
- **Chapter 4 - System Design:** This chapter has the design process of the embedded system of this thesis.
- **Chapter 5 - Results:** This chapter presents the necessary testing that was carried out for the verification of the correct operation, and performance of the system is presented.
- **Chapter 6 - Conclusions and Future Work:** This chapter presents the final conclusions which were drawn from the entire thesis, as well as some of the possible future developments of the system.

Chapter 2

Theoretical Background

Projects we have completed
demonstrate what we know.
Future projects decide what we
will learn.

Dr. Mohsin Tiwana

2.1 Embedded Systems

An embedded system is a specialized computing system that is dedicated to performing specific functions within a larger system or product. Unlike general-purpose computers, which are designed to handle a wide range of tasks and applications, embedded systems are tailored to execute predefined tasks or functions. These systems are embedded as integral components within larger devices or systems, and they often operate in real-time with specific constraints[2].

2.1.1 History and Types of embedded systems

The history of embedded systems is intricately tied to the evolution of computing and technological advancements. Embedded systems, as we understand them today, have a rich history that spans several decades. Here is an overview of the key milestones in the development of embedded systems[3]:

1. **Early Computing Machines (1940s-1950s)**: ENIAC and UNIVAC marked the beginning, laying the groundwork for integrating computing into diverse applications.

2. **Embedded Control Systems (1960s-1970s)**: The emergence of embedded control systems for industrial, aerospace, and automotive applications, using intergated circuits with logic gates and later on microprocessors like the Intel 4004 and 8008.
3. **Microcontroller Revolution (1980s)**: Widespread adoption of microcontrollers, combining a processor, memory, and peripherals on a single chip. The Intel 8048 and even more so the Intel 8051 microcontroller (1980) became a prominent platform.
4. **Real-Time Operating Systems (1990s)**: Growing importance of real-time operating systems, enhancing responsiveness in consumer electronics, telecommunications, and automotive applications.
5. **Proliferation in Consumer Electronics (2000s-2010)**: Integration of embedded systems into smartphones, smart appliances, and Internet of Things (IoT) devices, driven by advances in semiconductor technology, low-power design, and connectivity.
6. **Internet of Things (IoT) Era (2010s-Present)**: Integration of embedded systems into IoT, forming a network of intelligent systems for autonomous data exchange and task execution.

Throughout their history, embedded systems have evolved from specialized control systems to integral components in a wide array of applications. The ongoing advancements in hardware, software, and connectivity continue to shape the landscape of embedded systems, making them an essential part of the modern technological ecosystem [4].

2.2 Stereoscopic Vision

Stereo vision in the realm of machine vision refers to the use of two or more cameras to perceive and interpret the three-dimensional structure of a scene. Inspired by human binocular vision, stereo vision systems in machine vision leverage the disparities between images captured by multiple cameras to estimate depth and reconstruct a detailed, spatial representation of the environment. The typical method of stereo vision technique is shown on figure 2.1.

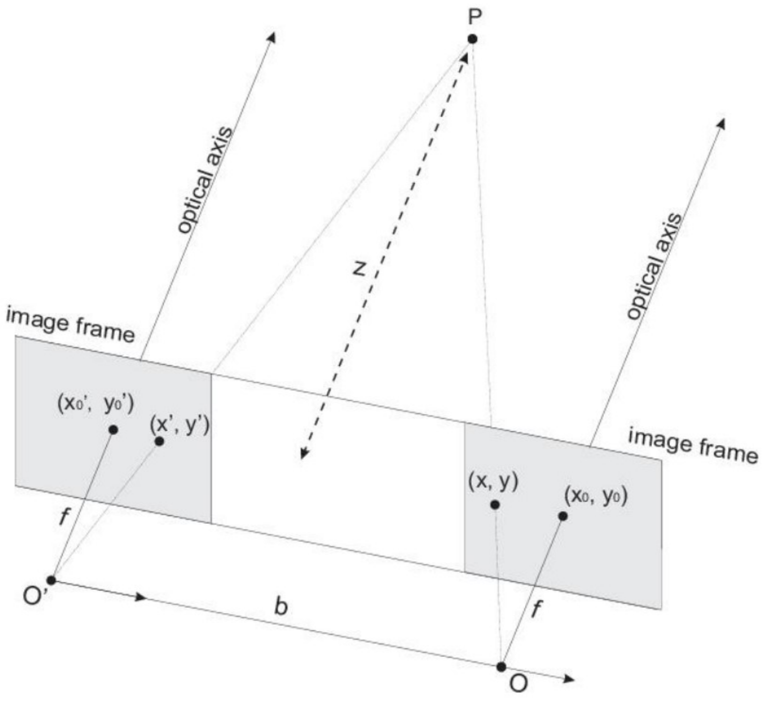


FIGURE 2.1: Stereo geometry of camera system. Source: [5]

The fundamental principle involves the triangulation of corresponding points in the left and right camera images to calculate the distance to objects in the scene. These disparities, caused by the different viewing angles of the cameras, provide valuable depth information, allowing machines to infer the relative distances and positions of objects[5]. In general the depth calculation process using Stereo cameras and stereo pairs can be summarized on figure 2.2.

For each point (x, y), depth is calculated using equation 2.1 where d is the disparity calculated from the stereovision algorithm, f is the focal length of the camera b is baseline of the stereoscopic camera and DPhor is the pixel horizontal dot pitch 2.2

$$Depth = \frac{b * f}{DP_{hor}} \tag{2.1}$$

$$DP_{hor} = \frac{\text{Active pixel array area}}{\text{Image Width}} \tag{2.2}$$

2.2.1 Stereoscopic Vision Key Applications

Stereo vision is widely utilized in various industrial and technological applications, contributing to the advancement of robotics, automation, and computer vision [6]. Key applications include[7]:

1. **3D Reconstruction**: Stereo vision enables the creation of detailed 3D models of objects or environments by mapping the spatial relationships between points captured by the cameras.
2. **Object Recognition and Localization**: By perceiving depth, stereo vision systems enhance object recognition capabilities and facilitate accurate localization of objects in a given space.
3. **Robotics and Automation**: Stereo vision is integral to robotics for tasks such as navigation, obstacle avoidance, and grasping objects. It provides robots with the ability to understand and interact with their surroundings in three dimensions.
4. **Quality Control and Inspection**: In manufacturing, stereo vision is employed for quality control, defect detection, and precision inspection, ensuring the accuracy and consistency of manufactured products.
5. **Autonomous Vehicles**: Stereo vision plays a vital role in the development of autonomous vehicles by helping them navigate and perceive the 3D structure of the road and surroundings.
6. **Augmented Reality**: Stereo vision contributes to augmented reality applications by enhancing the realism of virtual objects and their interaction with the physical environment.

Utilizing Stereo vision techniques to recover surface height through the correspondence process, while often effective, faces several limitations. Real-world challenges such as reflections, occlusions, and texture-less areas complicate the identification of corresponding pixels. This difficulty becomes particularly pronounced when dealing with flat, single-color surfaces. In such cases, where numerous pixels share identical values and the image lacks sufficient structure, accurate estimation of disparity becomes challenging. These factors contribute to the failure of Stereo vision algorithms in accurately recovering surface height in certain scenarios.

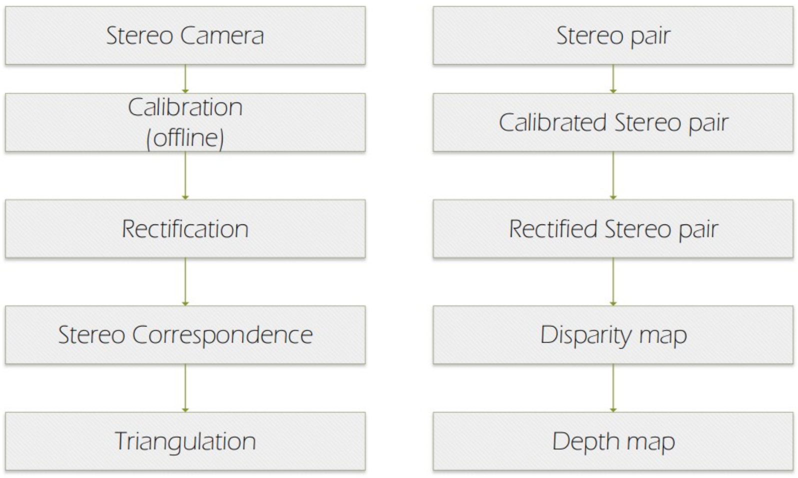


FIGURE 2.2: General description of depth calculation process
Source: [1]

As technology advances, stereo vision algorithms and systems continue to improve, offering higher accuracy, faster processing, and broader applicability. The integration of stereo vision in machine vision applications underscores its significance in enhancing the visual perception and decision-making capabilities of machines, making it a pivotal tool in the field of computer vision and automation.

2.3 3D Scanners

Another method of depth estimation is Laser Scanning. In this technique, different types of light, usually in the UV range, are being emitted. This light is reflected from the scanned object and with different types of sensors it is being captured in order to calculate the depth or the distance from the object. Laser Scanners can be broadly categorized in to Time-Of-Flight (TOF) scanners and triangulation scanners[8].

2.3.1 Time Of Flight

The Time-Of-Flight 3D laser scanner is an active scanner that uses laser light to probe a subject. [9]. The laser range finder finds the distance of a surface by timing the round-trip time of a pulse of light, visualised on figure 2.3. They use a light source, a very precise clock and a detector for the reflected back light. Since the speed of light **c** is known, the round-trip time determines the travel distance of the light, which is twice the distance between the scanner and the surface. If **t** is the round-trip time, then distance is equal to **c * t / 2** [10]. The accuracy of a time-of-flight 3D laser scanner depends on how precisely we can measure the **t** time: 3.3 picoseconds (approx.) is the time taken for light to travel 1 millimetre[6].

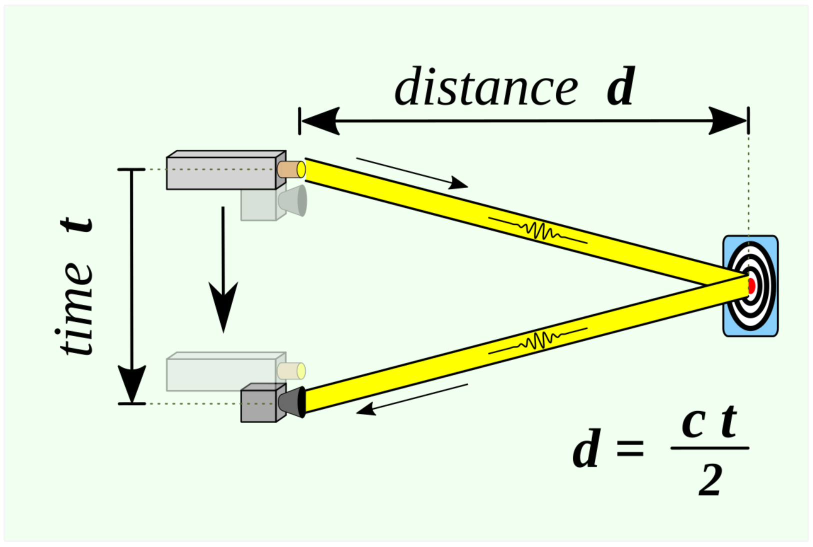


FIGURE 2.3: Time Of Flight 3D Scanner Principle. Source: [10]

The laser range finder only detects the distance of one point in its direction of view. Thus, the scanner scans its entire field of view one point at a time by changing the range finder's direction of view to scan different points. The view direction of the laser range finder can be changed either by rotating the range finder itself, or by using a system of rotating mirrors. The latter method is commonly used because mirrors are much lighter and can thus be rotated much faster and with greater accuracy. Typical time-of-flight 3D laser scanners can measure the distance of 10,000 100,000 points every second.

Additionally, road inspection, pavement evaluation, and road 3D surface modeling may be performed with 3D scanning techniques [11]. Moreover, can offer a real-time inspection system that uses structured light triangulation to find road distress characteristics including rutting and shove. Also provide a framework for recognizing structures from mobile laser scanning data [12].

2.3.2 Triangulation

Triangulation based 3D laser scanners are also active scanners that use laser light to probe the environment. Alike time-of-flight 3D laser scanner, the triangulation laser scanner shines a laser on the subject and exploits a camera to look for the location of the laser dot. Depending on how far away the laser strikes a surface, the laser dot appears at different places in the camera's field of view. This technique is called triangulation because the laser dot, the camera and the laser emitter form a triangle as shown on figure 2.4. The length of one side of the triangle, the distance between the camera and the laser emitter is known. The angle of the laser emitter corner is also known. The angle of the camera corner can be determined by looking at the location of the laser dot in the camera's field of view. These three pieces of information fully determine the shape and size of the triangle and give the location of the laser dot corner of the triangle. In most cases a laser stripe, instead of a single laser dot, is swept across the object to speed up the acquisition process [13].

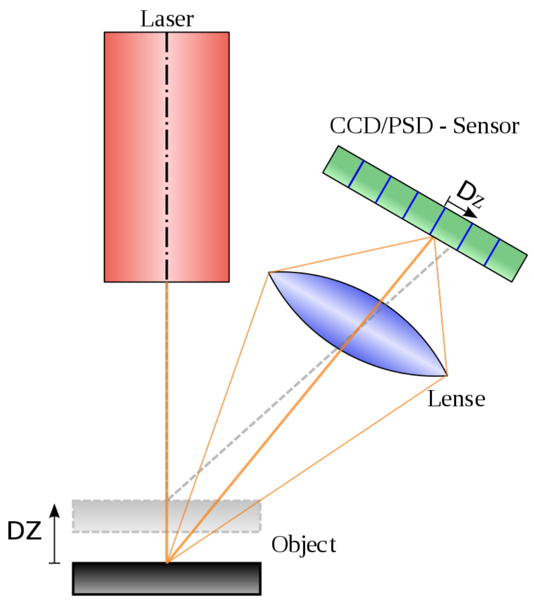


FIGURE 2.4: Triangulation 3D Scanner Principle. Source:[13]

2.3.3 Costs of 3D Scanners

1. Professional Desktop 3D Scanners:
    - Structured Light Scanners: These scanners use patterns of light to capture 3D data.
        - Price Range: $5,000 to $20,000
        - Features: Medium to high precision, suitable for capturing detailed geometry. Some models offer color capture.
    - Laser Scanners (Triangulation-based): Laser scanners use laser beams for precise distance measurements.
        - Price Range: $10,000 to $30,000
        - Features: High precision, faster scanning capabilities, suitable for intricate industrial applications.
2. High-precision Industrial 3D Scanners:
    - Laser Scanners (Advanced Industrial Grade): These scanners are designed for demanding applications such as aerospace and automotive industries.
        - Price Range: $30,000 to $100,000+
        - Features: Extremely high precision, large scanning volumes, fast data acquisition, and often integrated with industrial automation systems.
    - Computed Tomography (CT) Scanners: For highly detailed internal and external scans of objects.
        - Price Range: $100,000 to $500,000+
        - Features: Extremely high precision for inspecting internal structures, commonly used in industries like medical, aerospace, and electronics.
3. Handheld 3D Scanners: Professional Handheld Scanners: Portable scanners for capturing complex shapes and hard-to-reach areas.
    - Price Range: $5,000 to $15,000
    - Features: Good precision, flexibility in scanning different object sizes and shapes.

2.4 Monocular Methods

Different kinds of monocular methods for estimating depth are being developed, as shown on 2.5. Most of them are using expensive computers and new technology like machine learning in order to train models to understand depth from many different environmental factors[14] [15]. The main disadvantage of these methods is that it is based on a very time consuming and prone to error procedure of training the depth estimation model. Most of the time the training is supervised from real humans and other expensive systems, so the results are very much dependent on the quality and quantity of the training data. Furthermore, these systems are used in conjunction with other estimation system like Lidars in order to confirm their resulting data. Finally, monocular methods can fail to predict depth for objects that were often observed to be in motion during training e.g. moving cars – including methods which explicitly model motion.

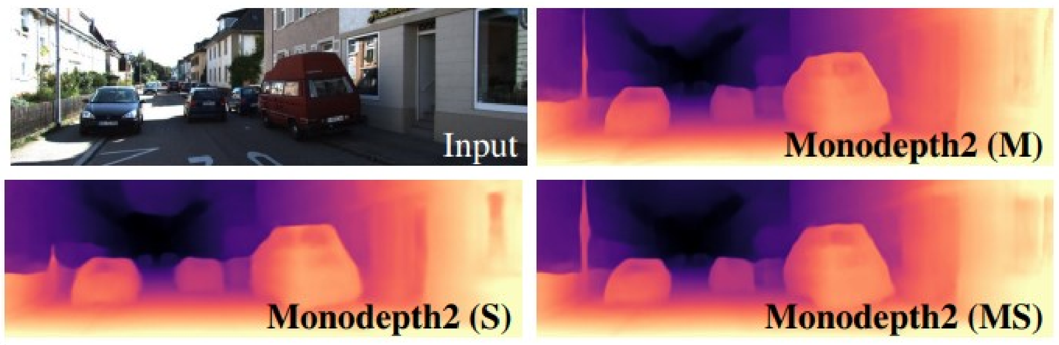


FIGURE 2.5: Monocular Method Results. Source: [15]

2.4.1 Fringe Projection

Fringe projection's main idea is projecting fringe images to an object using a projector and then recording the reflection with a camera in order to come to a conclusion. Fringe images are structured patterns composed by sinusoidal stripes [16]. A typical Fringe projection system is shown on figure 2.6. Straight fringe stripes are projected onto the object by a projector, either vertically or horizontally. The recorded image exhibits phase shifts due to the objects surface configuration. Analyzing the image we can calculate the phase modulation. Knowledge of the correspondence between the camera and the projector is required to retrieve the 3-D information through triangulation [17]. The phase modulation is calculated using a variety of fringe projections techniques. These techniques can be broadly classified as spatial and temporal [18]. For temporal methods it is necessary to acquire a number of images by projecting phase-shifted image fringe patterns, while spatial methods require a sufficiently high frequency spatial carrier [19]. The two most common methods for analyzing fringe data are phase-shifting and Fourier transform [20][21].

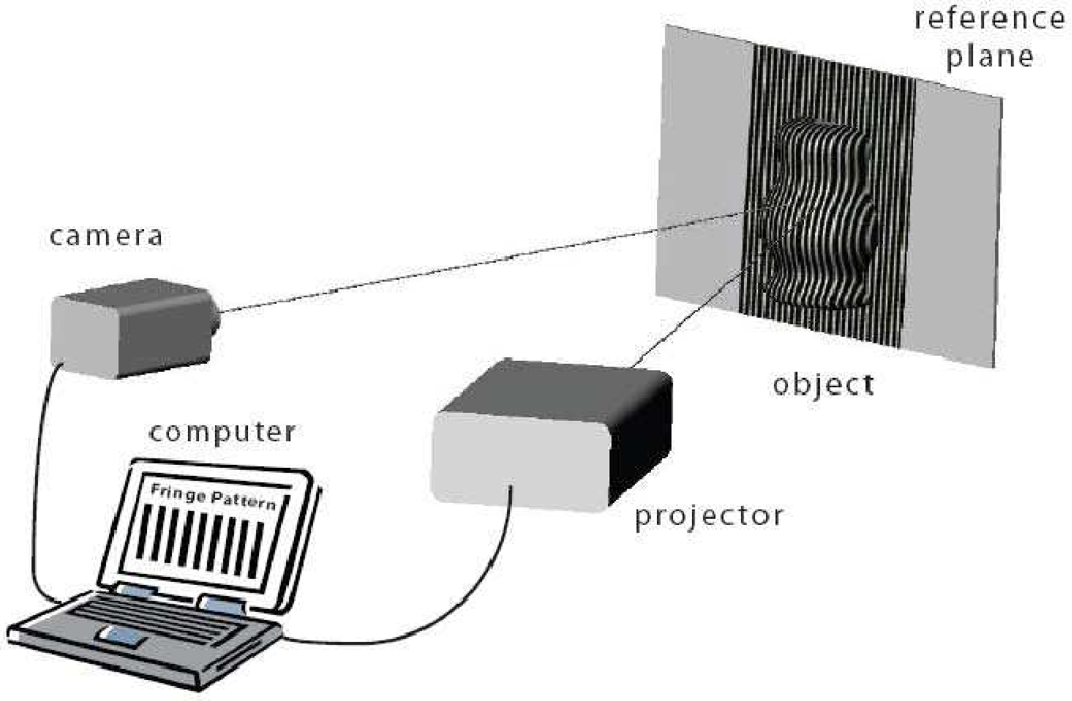


FIGURE 2.6: Fringe-Projection-System

Chapter 3

Proposed System Modeling

3.1 Basic illustration of operation

A very simple illustration in order to understand the way the system is organised is depicted below on the figure 3.1. In this 3D model the shielding and the processing unit are not depicted in order to understand the basic principle of the system. A RGB camera is placed on top, facing vertically the surface that is being examined. On the left and right of the camera 2 distinct LED array light sources are placed tilted by 45° on the horizontal axis. This ensures better illumination of the scene. In addition to this tilt, each LED array has a light diffuser fitted to make the light produced softer and the light lines as uniformly distributed as possible. This is very important because hotspots need to be avoided in order to keep the error factor low. The basic principle of operation is simple. Slopes facing each colored LED array have a color ratio with the specific color being prominent on the light that reflect back. How prominent is the color is the factor that tells the system how much is the angle of the slope that is being calculated. Areas without prominent color are being characterised as flat. This system is great at measuring dept in non-anisotropic surfaces, meaning that the direction of the reflected light has to be a linear function of the angle of incidence. In practice this works, because the vast majority of road surface that this system is targeted at, falls into this category.

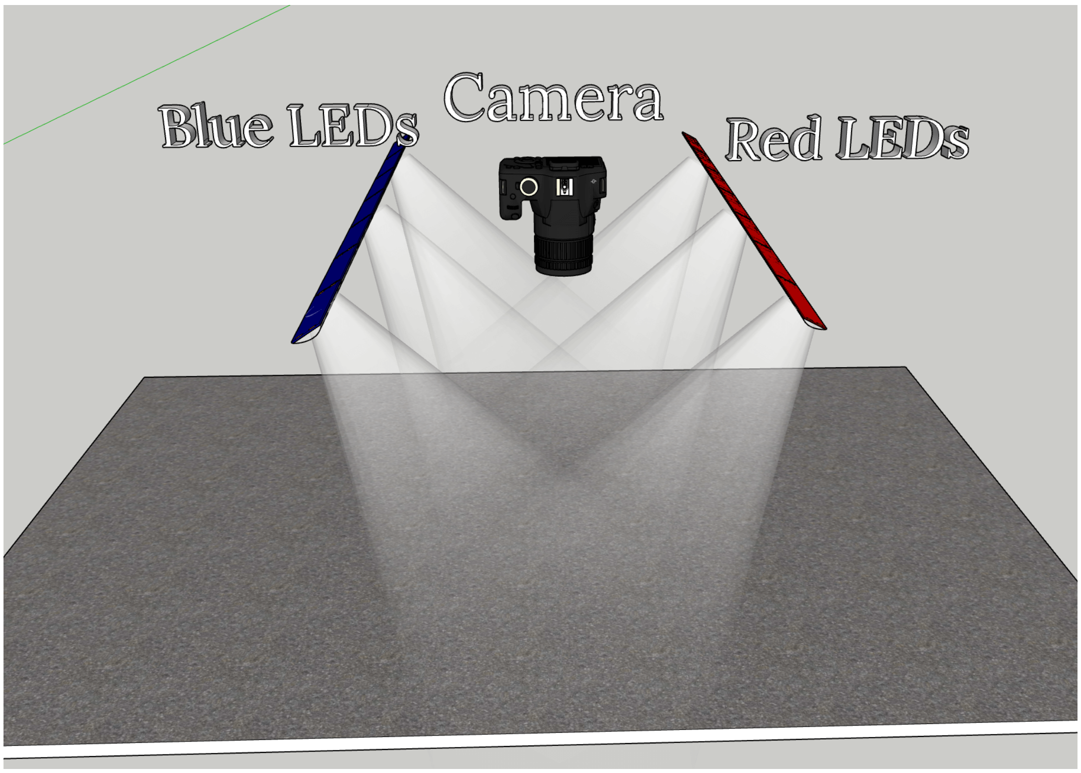


FIGURE 3.1: Basic Model of the System

The whole system, as implemented in the present thesis, is covered by a plastic cover in order to shield it for the sunlight and also the elements of nature like rain and dust. Minor noise created from sunlight can be tolerated as it adds an almost uniform light level to both colors so the system can be calibrated with it.

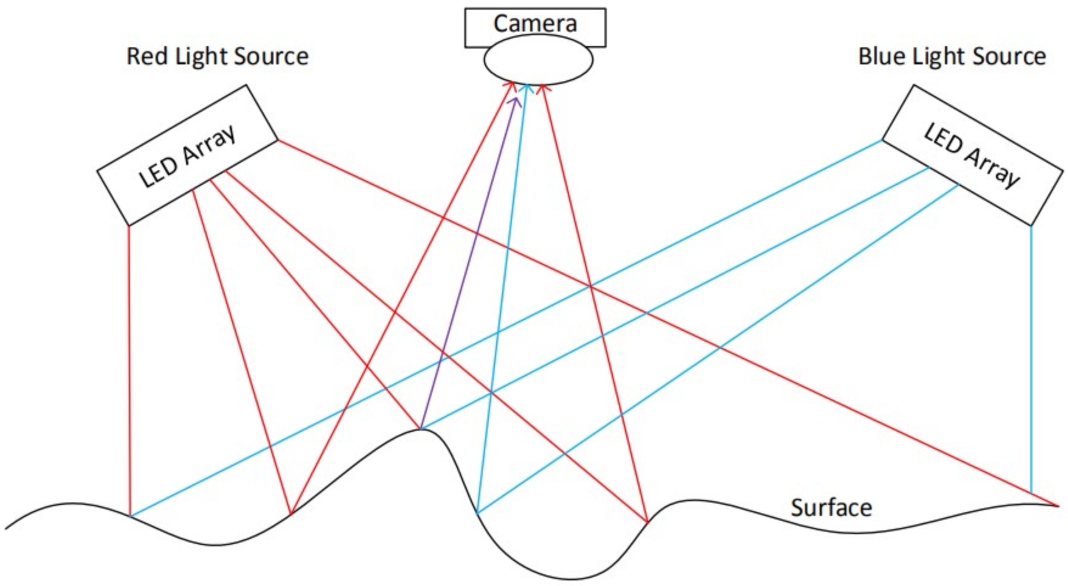


FIGURE 3.2: Simple Representation of Operation

3.1.1 Calculation model

In this section the basic principles of the the calculation procedure are presented. A simple algorithm flowchart is created in order to show as simply as possible the idea of this depth calculating mechanism. More shown also in the figure 3.3.

Upon activation of the system, the two RGB LED arrays are turned on. The examined surface is illuminated by both these light sources. The camera module receives the reflected light from both these light sources. A calibration step is required in order to receive the correct the same level of reflected light intensity from both light sources, this can be visualized on figure 3.2. The calibration step is explained below. After the system is properly calibrated, it begins constantly capturing frames as a video sequence. Each frame is individually processed. First, all the pixels of each frame are split between the three RGB channels, Red, Green and Blue. The green channel is not used as it conveniently exists between the red and blue color on the visible spectrum so it acts as a buffer between the two. The next step is to save the pixel values of the red and blue intensities of each pixel with their specific positions on two matrices, one for each color. To account for the attenuation of light for each light source a normalization is required. Two normalization matrices $I_{rn}(x,y)$ and $I_{bn}(x,y)$, one for each color, are created during the calibration step, holding the information of how is the light being reflected on the empty surface without any object. By multiplying each frame with these normalization factors a normalized and uniform light distribution model is achieved and the system is ready to receive new information.The final formula is expressed below:

$$I_1(x,y) = I_R(x,y) \cdot I_{rn}(x,y) \tag{3.1}$$

$$I_2(x,y) = I_B(x,y) \cdot I_{bn}(x,y) \tag{3.2}$$

The following process is obtaining the intensity ratio of each pixel in the frame by dividing the two previous matrices $I_1(x,y)$ and $I_2(x,y)$ we realistically can extract each LED sources contribution to each pixel. Simply stated, the information gained for this procedure is that a ratio greater than one means that in this specific pixel shows a slope that faces the LED of the numerator. So if, $I_1(x,y)/I_2(x,y)$ is greater than one the slope on this pixel is towards the red LED source while if it is lower that one it is towards the blue LED source. With this process a matrix of the gradients of each frame is created. Furthermore, from the difference of the numerator from the denominator we can create a quantitative approach of the angle of the slope. A simple flowchart is shown below 3.3.

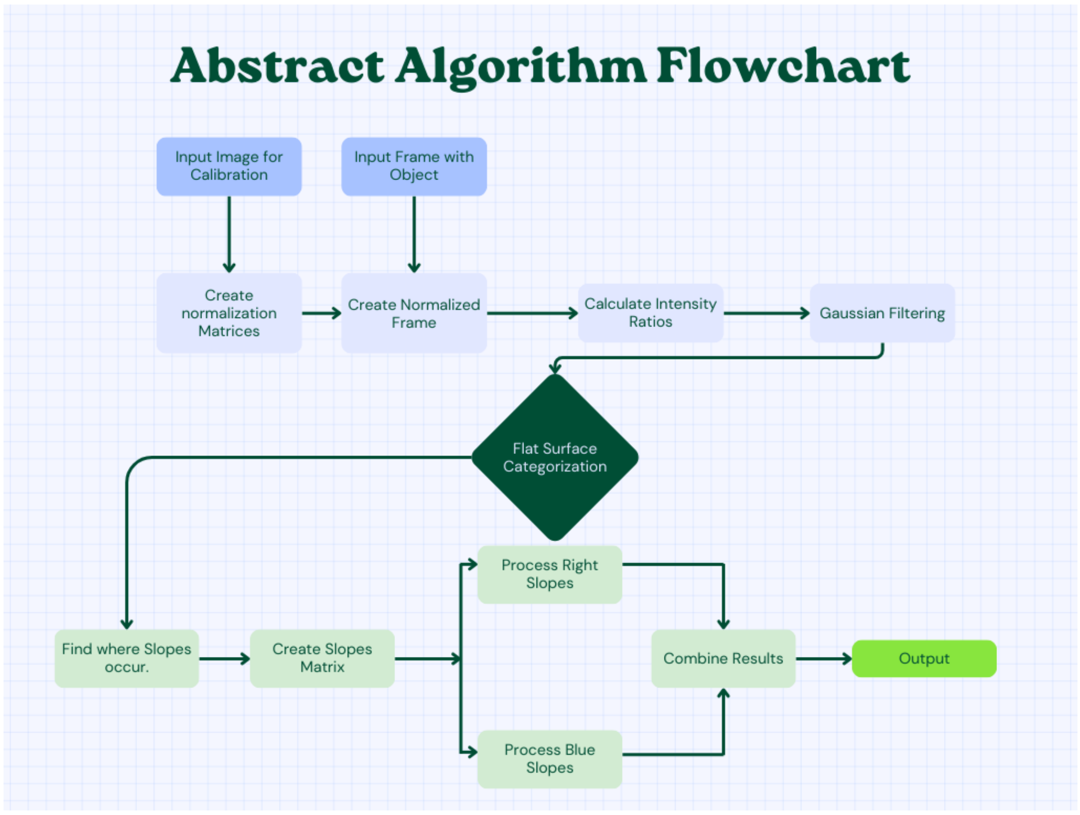


FIGURE 3.3: Abstract Algorithm Flowchart

3.1.2 Flat surface identification linear procedure

When the ratio is one, it is safe to assume that the slope in this spot is zero (the surface is flat) as the pixel is reflecting the same amount of light from both light sources equally. While this is true, in reality it is not possible for light to be reflected mathematically the same in a pixel so this ratio is close to one but almost never exactly one. In order to bypass this problem a specific method is developed in order to find these flat areas as they are very important for the slope calculation of the object as they act as the starting point for every calculation since the slope is calculated as the absolute difference between the flat areas and the ratio where slope occurs.

At this point of the calculation model each frame is represented as a matrix of ratio values. Flat areas are identified by individually accessing and processing each row of this matrix. The main idea is that the mean value of the whole row is the value representing the flat part of the surface, but since slopes can be present from only one side of the object this is not always true, so a more sophisticated methods had to be implemented. To solve this problem, large variations of the ratio had to be excluded from the mean value calculation. In order to achieve that an arbitrary value of a threshold is selected and named τ_1. This value was experimentally selected but it can differ depending what surface, lighting and camera conditions each system faces. In this specific system the value of 0.1 was selected. Only values that lie between these values as the below equation 3.3 expresses are included in the mean value calculation.

$$mean(\lambda(i)) - \tau_1 < \lambda(i) < mean(\lambda(i)) + \tau_1 \tag{3.3}$$

After having excluded all the values with great variation from the mean value from row $\lambda(i)$ a subset of the $\lambda(i)$ row is created that is named $\lambda_{temp}(i)$. A temporary mean value of the line is saved as shown in figure 3.4. One value is created the same way that τ_1 was created. This value is called l and it is experimentally set to 0.6. This value is used in order to calculate the final mean value of the row by filtering all the values from the original row that are not inside the bounds presented by this equation 3.5.

$$tempMean = mean(\lambda_{temp}(i)) \tag{3.4}$$

$$tempMean - l < \lambda(i) < tempMean + l \tag{3.5}$$

After filtering all the values outside these bounds (seen upwards) a new subset of $\lambda(i)$ is created that is named $\lambda_{final}(i)$. The final mean value of the row is calculated from this subset with this equation 3.6.

$$FinalMean = mean(\lambda_{final}(i)) \tag{3.6}$$

This whole process is done two times for each frame. One from each light source to the other. From blue light source to red in order to capture the slopes and flat areas from the blue light's perspective and then the same for the red light source as presented on 3.4. After this process has finished the data can be compared in order to approximately find the error factor and the trust factor for the generated results.

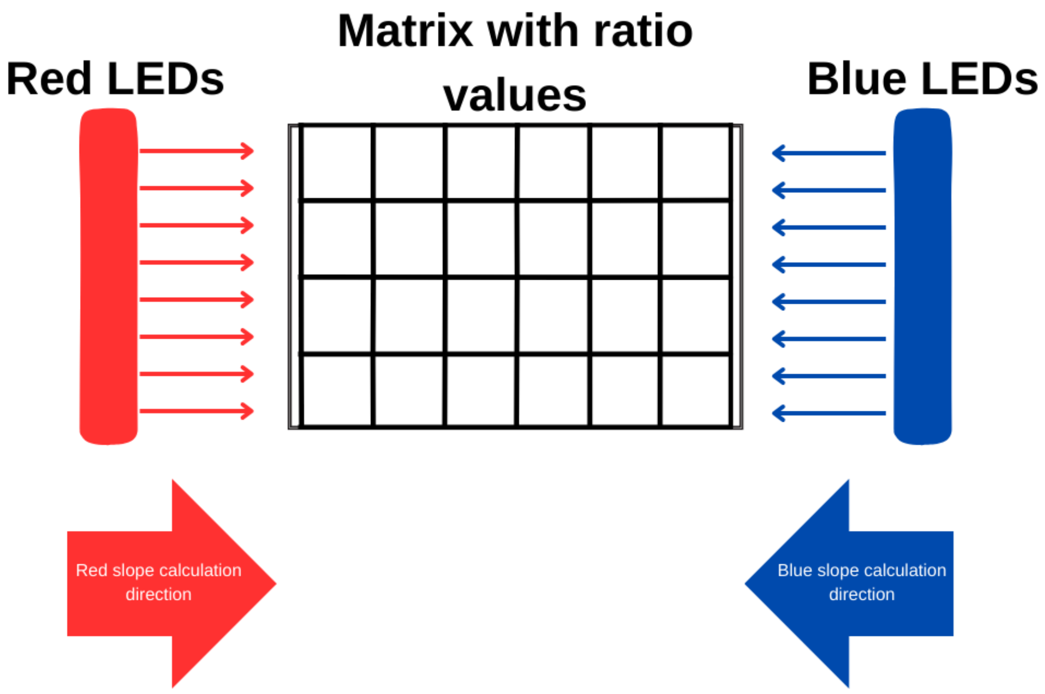

Red LEDs
Matrix with ratio values
Blue LEDs
Red slope calculation direction
Blue slope calculation direction

FIGURE 3.4: Calculation directions for each light source

3.1.3 Threshold Value Selection

Threshold values can be chosen based on different factor and can be very different for each system implementation. The smaller the threshold values, the greater the accuracy that the system can achieve. This of course does not mean that by choosing really small threshold values automatically the system can achieve is peak accuracy. The main factors from which the threshold value selection is dependent on are the surface that we intent to scan and its reflective properties, the lighting conditions, how much noise is arriving to the system and the quality of the used camera. In fully controlled conditions and in lab conditions, really small threshold values can be achieved and the accuracy of the system is at its peak, while for real road conditions the threshold values need to be a bit larger in order to compensate for the noise that is impossible to avoid.

Using a lower than optimal threshold value, noise from the camera or small surface abnormalities can be visible as shown in figure 3.5. This noise can create artifacts and can intervene with the normal operation of the system creating anomalies to the height or depth of the measured objects.

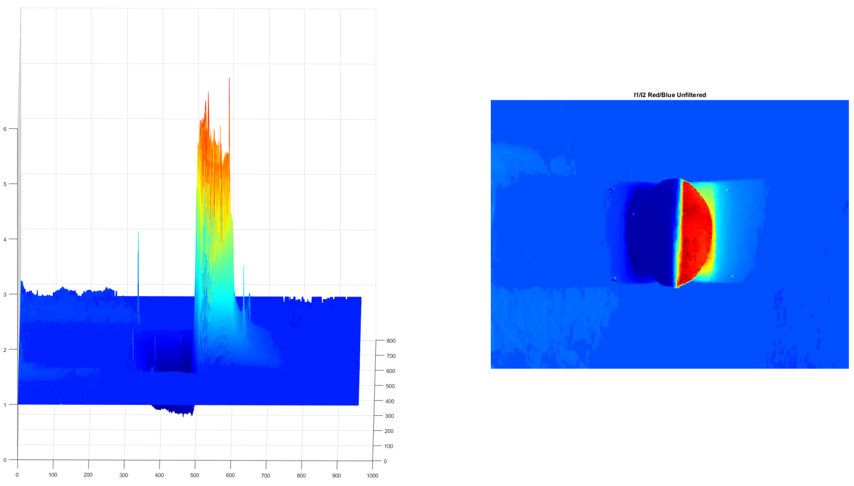


FIGURE 3.5: Red/blue ratio with low threshold value

Using a higher threshold value 3.6 it is possible to eliminate the noise and create a much clearer image with only the objects that are wanted in the scene. This create a far better environment for the depth estimation algorithm with false positives.

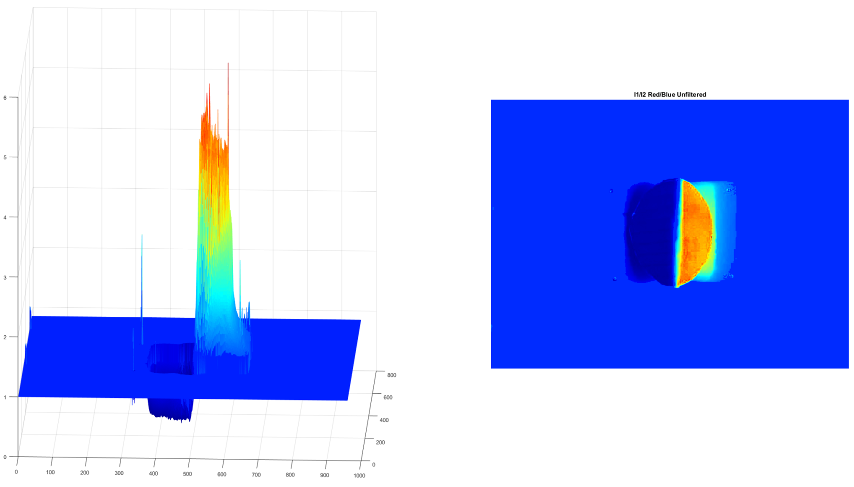


FIGURE 3.6: Red/blue ratio with higher threshold value

3.1.4 Calibration Procedure

Calibrating the system in order to work correctly is a very simple yet important process that needs to take place always before starting the system. The main goal of calibration is the equalization of the LED array's intensities. Describing the process is simple, a white surface is laid under the camera system in the 0 height point so that the blue and the red light hits this surface uniformly. There are two dimmers, one on each LED array; these dimmers are used in order to bring the histogram of the red and the blue colors as near spectrally as possible. The finished result is shown in the figure 3.7 below. After this first step, this procedure is repeated on the surface that is about to be scanned in order to check for abnormalities related to the specific surface, some surfaces may require minor adjustments.

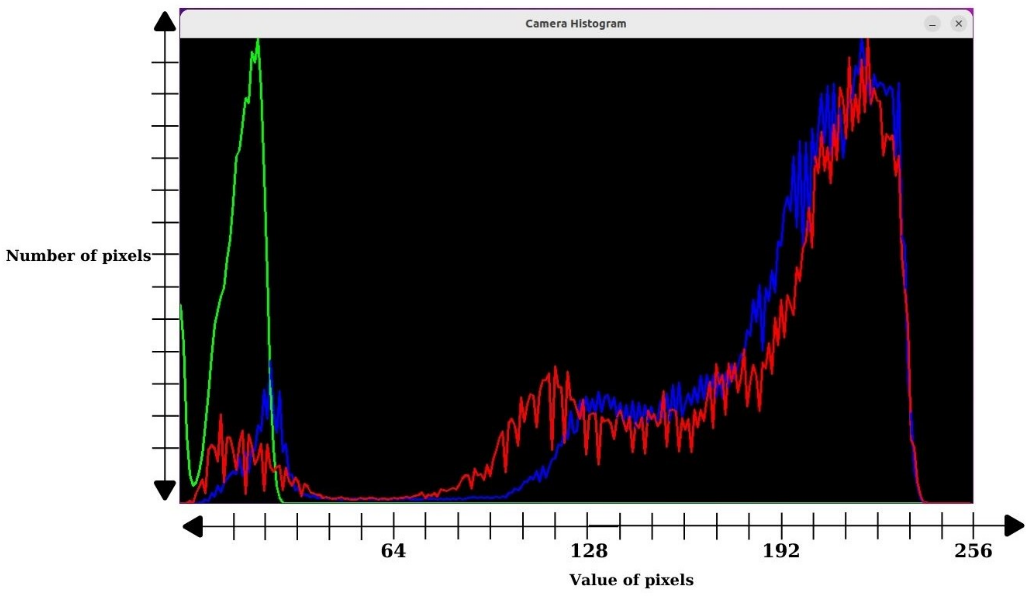


FIGURE 3.7: Calibrated Histogram

3.2 Verification of the Technique

The first step of implementing a real prototype for this system was verifying that the technique described by G. Rematska was correct and could indeed be recreated in different conditions. The first stage of this verification was done in Matlab and did confirm the thesis results. An image of the illuminated scene is added 3.8.

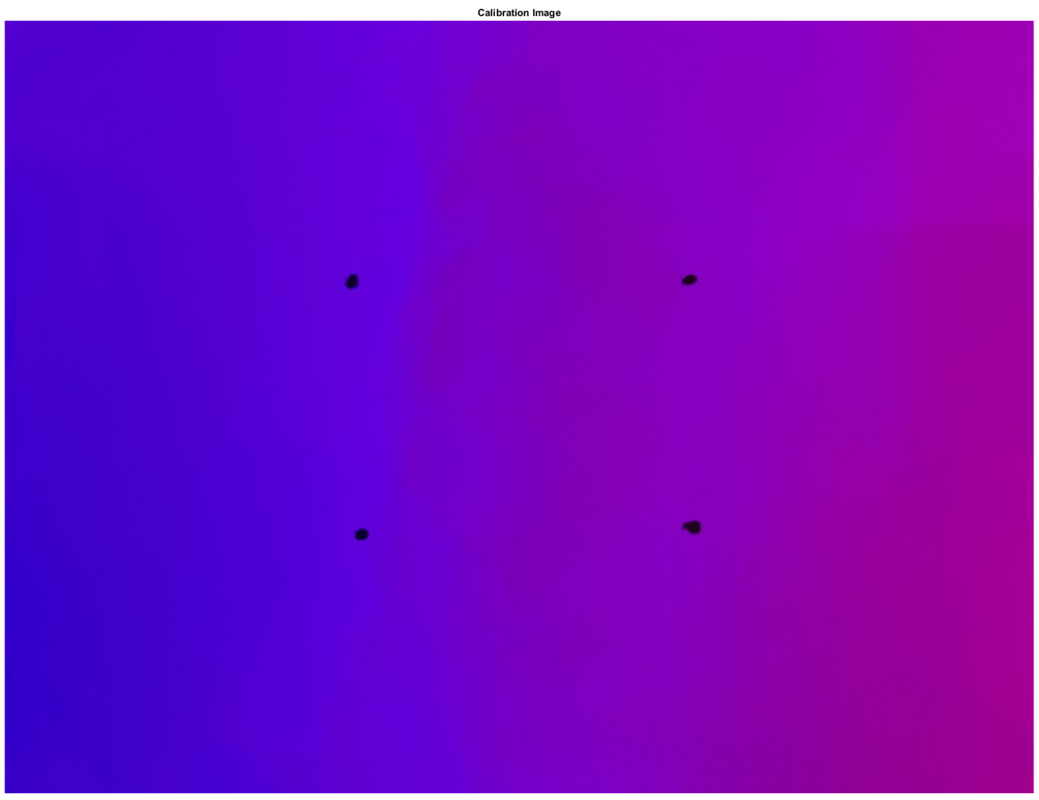


FIGURE 3.8: Calibration Scene Image

This image is used for calibration purposes and most importantly it provides the system with very important information about the surface illumination when no object exists in the scene. With these data the normalization factor matrix is created and the flat surfaces can be established. After successfully importing the calibration scene image, the next image imported is the object image 3.9. This image can be a still frame from a video sequence as used in this system or can be just a picture of the object. A very important parameter in order for this method to work is the position of the camera relative to the light sources, as any change can create unwanted results. The camera has to be completely still inside the system or many artifacts or miscalculations can occur. In our experiment the still frame depicts an object with measured height of 5cm.

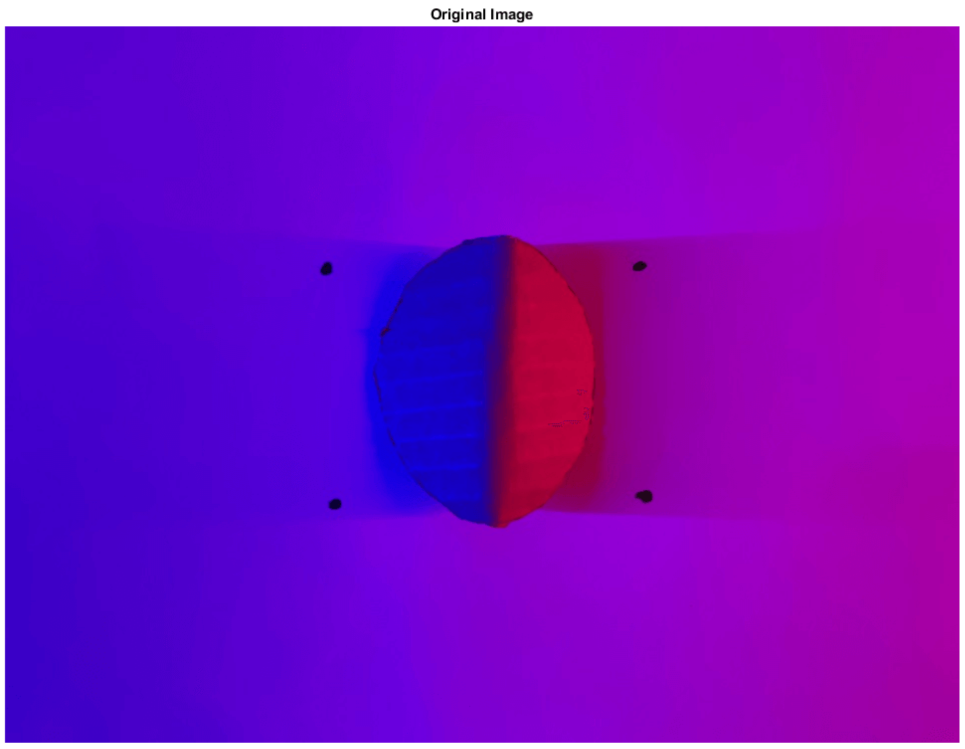


FIGURE 3.9: Object scene Image

After importing the image, the next step is to isolate the blue and red intensities as shown in 3.10 of the LEDs in order to be able to calculate their contributions in the slopes.

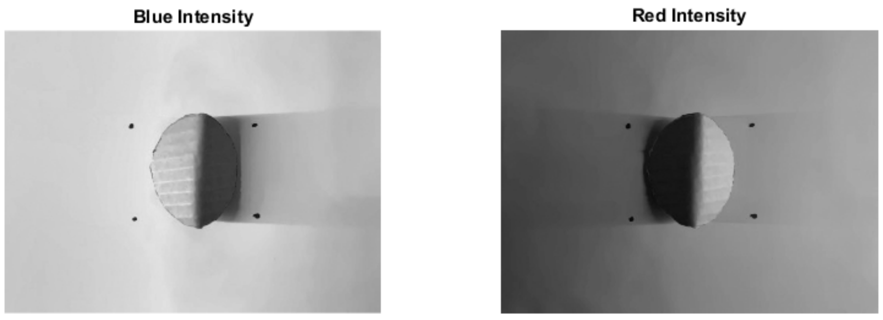


FIGURE 3.10: Color Contributions without normalization

In the next figure 3.11 the importance of normalizing the scene with the calibration image 3.8 is shown. The main advantage is eliminating the flat surface contribution from the light variation because of the distance of the light sources on the horizontal axis.

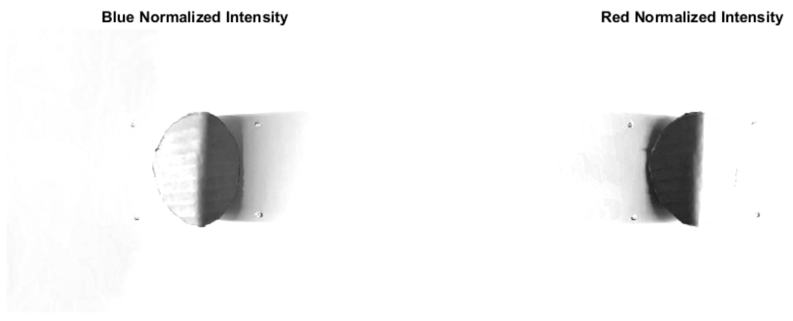


FIGURE 3.11: Contributions with normalization

Blue/*red* and *red*/*blue* ratios after normalization are very important data as explained before. In the figure 3.12, below, it is very easy to visualize the slopes that occur from the two different color contributions. Slopes facing the blue LEDs are calculated from *blue*/*red* ratio while slopes facing the red LEDs are calculated from *red*/*blue* ratio.

At this stage G. Rematska proposes in her thesis a blurring filter like a 5X5 Gaussian filter. In the testing of this methodology and all the replicated results these kind of filters show a good effect of lowering the deviation of the data in nearby pixels and help in the accuracy of the system. Higher order filters have no effect on the observed values of the ratio, therefore they were discarded. The explanation for this observation is the higher resolution of images used does not need that high order of filter because the extra resolution is lost. The lowest resolution used is high definition (HD) which means a resolution of 1280x720. In this resolution the smoothing filters of 5X5 could provide the most improvement on the data collected, so it safe to reach a conclusion that the processing load added from higher order filtering can be avoided.

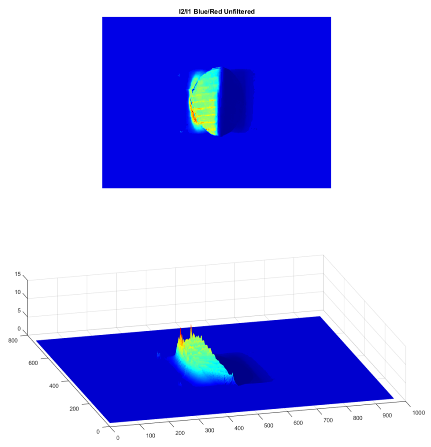


FIGURE 3.12: Color Ratios unfiltered

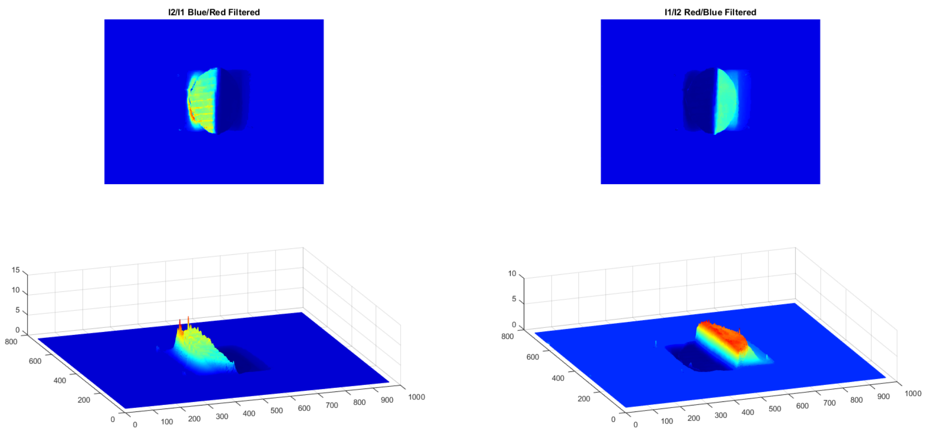


FIGURE 3.13: Color Ratios filtered with a 5x5 Gaussian Filter

Finally, to estimate the whole depth we process the image in the two directions left to right and right to left always keeping in mind that the directions for processing are starting from the one LED's light source to the other. When we process the image left to right we examine the slopes that occur when they are facing the blue LEDs and when we process the image right to left we examine the slopes that occur when they are facing the red LEDs. The depth at each pixel is calculated by adding the depth in the current pixel with the depth already calculated in the previous pixel. Equation 3.9 describes the value of the distance z of a line at pixel i, on figure 3.14 a simple visualization is shown. The parameter a is the position of the first occurrence of the slope of a specific object meaning the point where the flat surface transitions to an object.

$$\text{object distance from camera} = \frac{\text{object Size} * \text{focal length}}{\text{Object Size in Image}} \tag{3.7}$$

$$\text{object size in image} = \frac{\text{object width (pixels)} * \text{sensor width (mm)}}{\text{image width (pixels)}} \tag{3.8}$$

Sensor Specifications	
Focal Length	1.53 mm
Sensor Width	22.0 mm
Width in pixels	1920 px
Pixel Size	5.2 µm

TABLE 3.1: Camera sensor Specifications

$$\sum_{k=\alpha}^{i-1} z_k = z_i \tag{3.9}$$

$$z_k = \text{depth in k pixel} = \text{slope} * \text{objectSize of one Pixel} \tag{3.10}$$

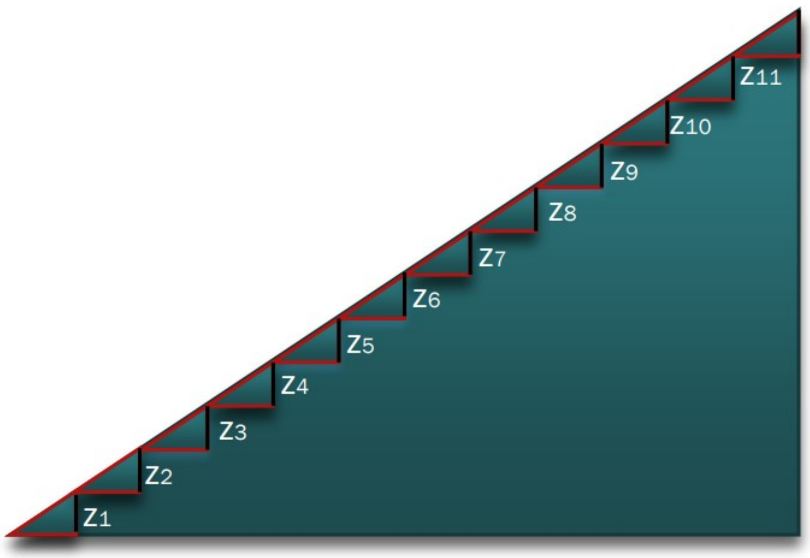


FIGURE 3.14: Representation of equation 3.9

The final object reconstruction is shown below on figure 3.15 and the height of the reconstructed object matches the height of the real object. The measured height from the system is 4.9 cm approximately while the real height of the object is 5 cm.

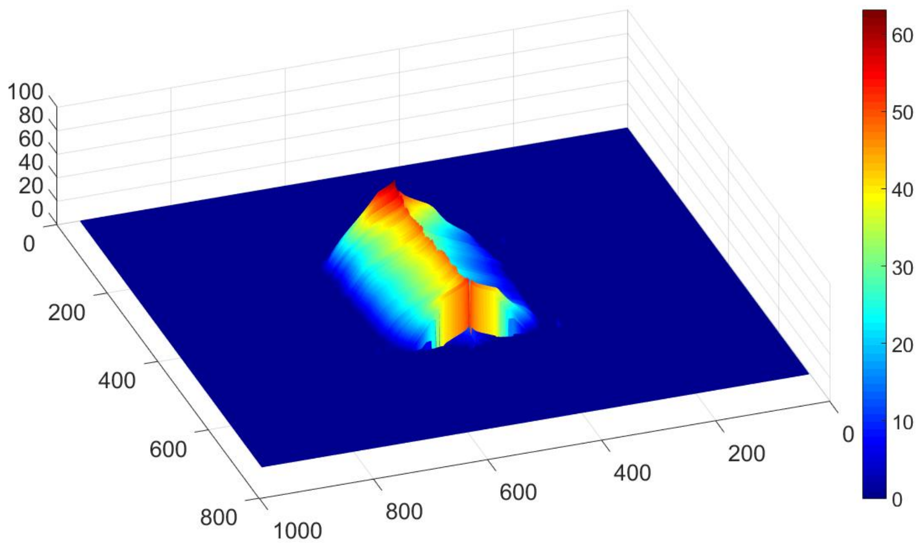


FIGURE 3.15: Final Object Reconstruction

A final observation can be made on figure 3.15. As it can be observed on the two edges of the object it is created a "tail". This is an important observation and it is caused from the shadows created by the object. These "tails" do not interfere with the object detection and depth estimation but maybe they can be eliminated in a future version of this system.

3.3 A Comparative Analysis of Computational Speed: Python vs. C++

The performance disparity between Python and C++ has been a subject of extensive investigation, considering their distinct design philosophies and use cases. In this implementation this disparity was shown as the system was first implemented in Python and then was later re-implemented using C++ for more performance and greater accuracy in transforming the data. Some of the reasons behind the speed differences observed in code written in Python and C++, their inherent characteristics and how they impact computational efficiency will be presented in this section. This comparative analysis is important because of the use of OpenCV library in order to implement the code for the depth calculation. OpenCV is mostly used with Python or C++ code. As a first step the implementation of the algorithm with Python was looking as wise step because of the abstraction of python making the coding of the project faster. After the project was finished the performance analysis showed that Python was lacking on performance and optimising the code gave little speedup for the system. Because of this, the decision to re-implement the system in C++ was made giving a lot more freedom on manipulating the memory and making the essentially the program from single-threaded to multi-threaded which gave a very good speedup as will be presented below.

3.3.1 OpenCV library

OpenCV is an open-source computer vision library, it has emerged as a cornerstone in the field of image processing and computer vision. Renowned for its versatility and extensive suite of tools, OpenCV facilitates the development of robust applications for image and video analysis, object detection, machine learning, and beyond. The library is structured into modules, each addressing specific computer vision tasks, ranging from basic image processing to advanced machine learning algorithms. OpenCV is primarily implemented in C and C++, leveraging the efficiency and performance advantages of these languages. However, its flexibility extends to bindings for Python, Java, and other languages.

3.3.2 Python

Python, renowned for its readability and simplicity, employs dynamic typing and features a high-level abstraction, offering ease of use and quick development. However, these conveniences come at the expense of execution speed. Python is an interpreted language, and its dynamic nature introduces runtime overhead in variable handling and type checking. Furthermore, Python's Global Interpreter Lock (GIL) poses limitations on concurrent execution, hindering parallel processing capabilities. This Global Interpreter Lock was an especially big disadvantage for the microprocessor used to implement the project as it is an ARM Cortex-A72 with 4 cores and a relatively slow clock at 1.8 Ghz. Using Python it was very difficult to use the parallelism of the four cores in order to speed up the calculation's executions which resulted to bad performance.

3.3.3 C++

C++, a statically-typed language, emphasizes performance and control over abstraction. Compiled directly into machine code, C++ programs exhibit enhanced execution speed. The absence of a GIL facilitates efficient multithreading, allowing for concurrent processing and improved parallelism. Additionally, C++ provides low-level memory manipulation, enabling developers to optimize code for specific hardware architectures. All these upsides lead to faster program execution as shown on table 3.2.

Python vs C++ on ARM Cortex-A72		
Language used	Average Frame time	Frames per Second (FPS)
Python	3.64s	0.274
C++	0.464s	2.155

TABLE 3.2: Python vs C++ Speed Comparison ARM Cortex-A72

$$S_{latency} = \frac{L_{slow}}{L_{fast}} = \frac{3.64s}{0.464s} = 7.84 \tag{3.11}$$

As shown on equation 3.11 the latency of the system was greatly increased when the language was changed from Python to C++. The output of shown to the user managed to become almost 8 times faster than the Python code. This shows how significant is each language's architecture and how important it is, to choose the correct language for each project.

Chapter 4

System Design

This chapter describes the design and implementation process of the depth estimation system of this thesis. A high-level block diagram separates the system into four distinct subsystems as shown on figure 4.1. First, the lighting subsystem responsible for the correct light intensity, light smoothness and color correctness of the illuminated scene. Second, the optical camera subsystem, whose responsibility is detection, and data collection of the color differences of every pixel in the RGB colorspace of the scanned surfaces and objects in front of the camera. Third, receiving the information from the camera sensor in order to approximate the dept of every object, the processing subsystem, responsible for running the depth estimation algorithm and correctly presenting the results to the end user with every crucial information necessary. Finally, completing the system is the forth subsystem, the Shield subsystem, having the role of the skeleton of the whole system and preventing noise from external light sources to interfere with the depth processing of the above parts.

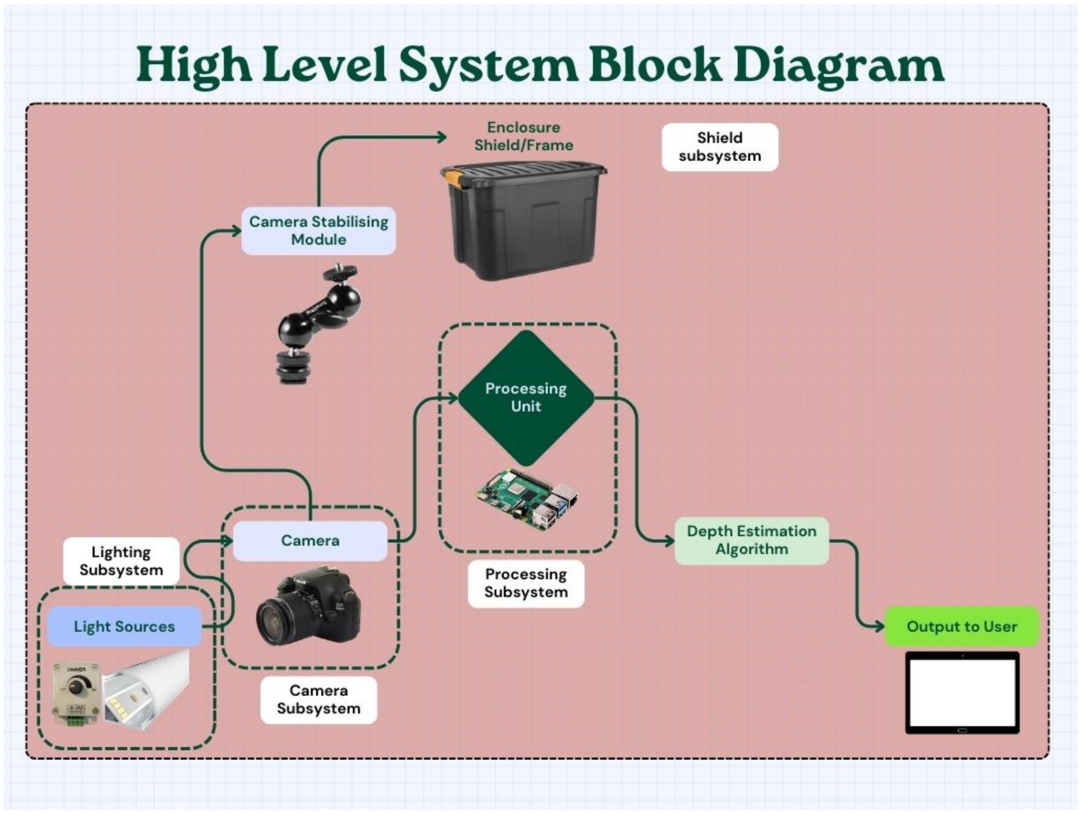


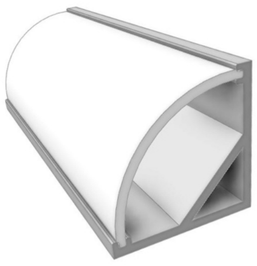
FIGURE 4.1: High Level System Block Diagram

4.1 Lighting subsystem

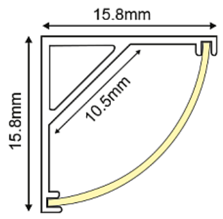
4.1.1 Lighting Requirements

- 2 colors Monochromatic
- High illumination intensity
- Uniform Illumination
- Low cost
- Individual intensity adjustment
- Robustness
- Readily available commercially
- Low noise

Light illumination is critical in this system and for this reason a system needed to be designed in order to meet all the above requirements. The lighting subsystem consists of two monochromatic LED arrays mounted inside two angled aluminium profiles equipped with light diffusers, presented on figure 4.2. The profiles help protect the LEDs and act as a cooling mechanism to keep them from overheating. The LED type selected is SMD 5050, this type is commercially available low cost and has the biggest size compared to other types producing the greatest illumination for its price.



(A) Model of used aluminium profile



(B) Layout of aluminium profile with diffuser

FIGURE 4.2: Aluminium Profile with diffuser

High density LED strips are also selected which can achieve the uniform illumination needed. High density LEDs have 60 LED SMD chips per meter providing higher illumination intensity and more uniform light distribution as shown in 4.3.

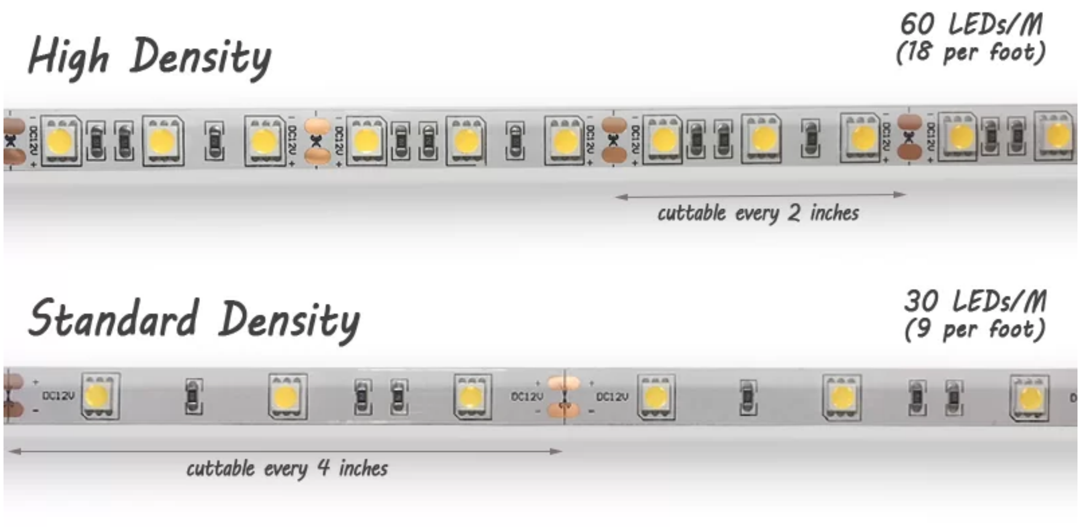


FIGURE 4.3: High Density Led Strip vs Low Density Led Strip

The selected LED strips have low noise, which means that the light produced is very close spectrally to what is promised. This is very important because the system depends on this color difference.

To check the accuracy of the color picked up by the camera a histogram test was conducted. A white scene was illuminated from each monochromatic source at a time and the histogram was calculated. In figure 4.4 the blue noise created from the red LED can be extracted.

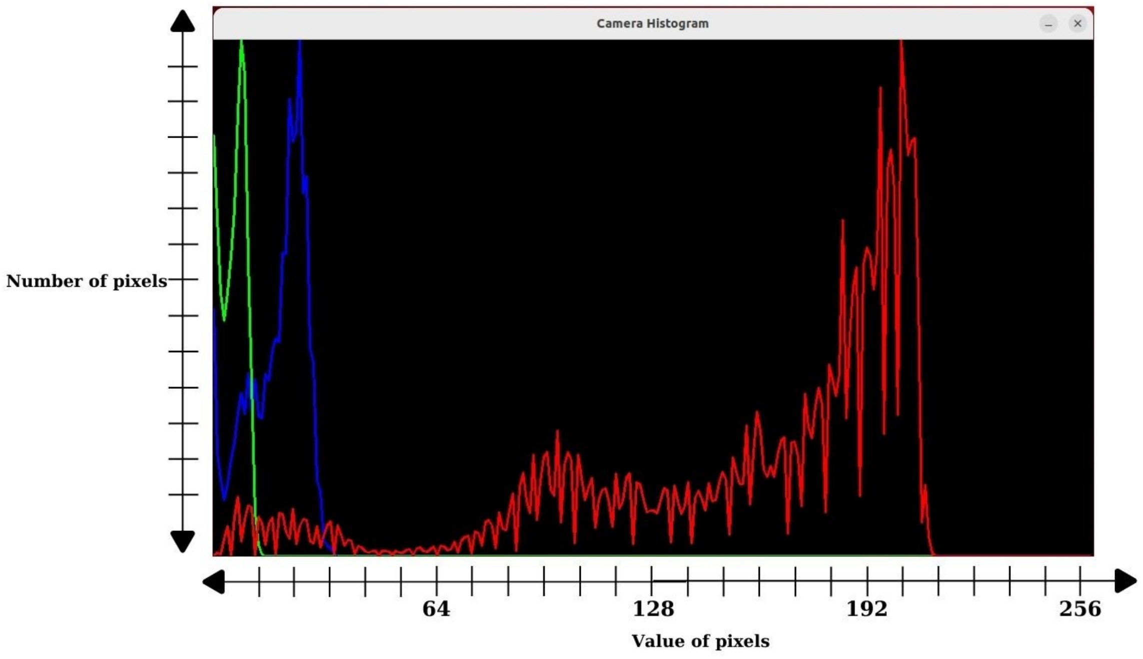


FIGURE 4.4: Blue light noise when only red LED on

As shown in the above figure the noise is insignificant and primarily caused by the camera's sensor which makes it unavoidable.

In figure 4.5 slightly more red noise is observed in the blue LED only scene. This might not be favorable and is again cause by the camera's sensor, this time exhibiting higher sensitivity in the red color's spectrum as it is expected from a CMOS sensor [22] but it is tolerable and does not produce any problems in the reliable operation of the system. This is further explained below in section 4.2.3.

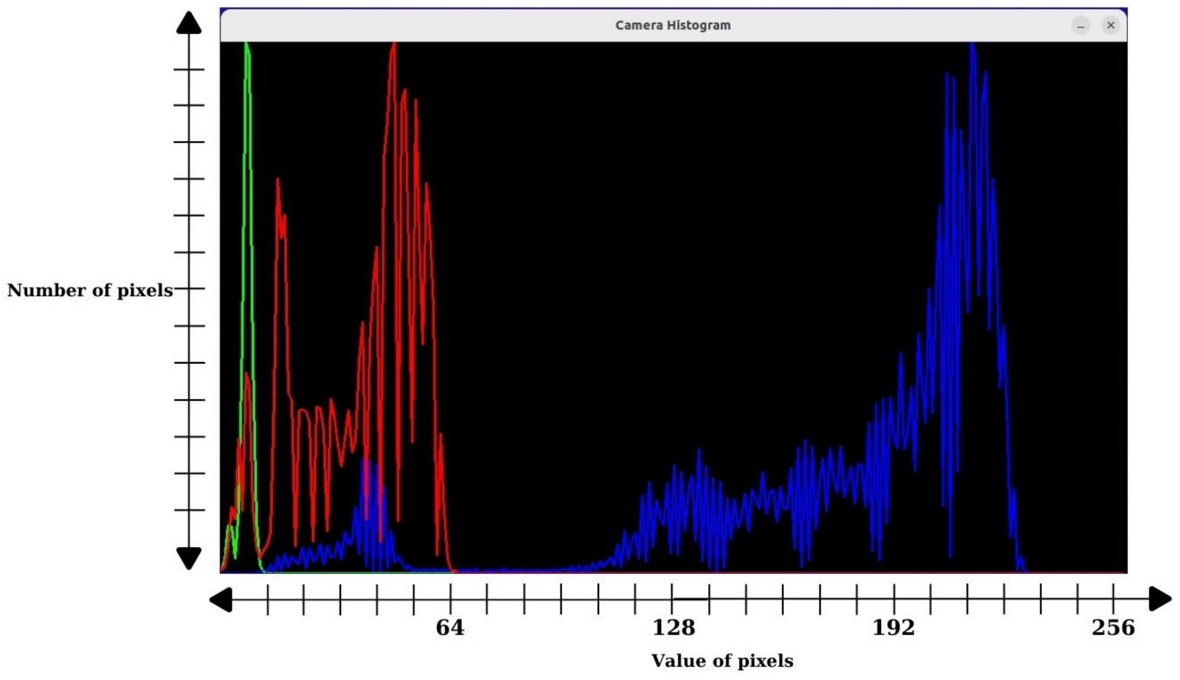


FIGURE 4.5: Red light noise when only blue LED on

4.2 Camera subsystem

4.2.1 Imaging requirements

- High definition image
- Ability to connect with the processing subsystem
- Adequate frame rate $FPS \geq 20$
- Broad dynamic range
- Accurate color depiction
- Near focusing Lenses
- Low cost
- Ability to capture Video
- Manual configuration without automatic adjustments

4.2.2 Camera selection

Taking into account the above requirements the selected camera was specified to be the Creative Live! 1080 v2. This camera checked all the above requirements in theory and was purchased in order to be tested in the system prototyping process. While this camera, being a USB webcam was easy to use and has great High Definition (HD) image at 1080p resolution with a very low cost under 30€, has a big downside which is creates a problem on the depth estimation process. The downside is that the camera itself has an automatic calibration system inside changing the white balance, ISO and exposure time of the sensor all the time which great for the purpose its manufacturer intended it to be used but not for this specific process.

In order to solve the camera problem a Canon EOS 1100D DSLR camera is used that was available at the time. This camera is solving the problem of the automatic parameter changing and gives greater image data with many more optional parameters like zooming manual focus and bigger sensor size. The only downsides are the bigger size, more weight and cost. Having in mind that this camera was already available and that this is a prototype system and not a final product, the downsides can be omitted. Both the camera modules are presented below on 4.6.

(A) Starting Web cam used
Creative Live! 1080 v2

(B) Final prototype camera
Canon EOS 1100D

FIGURE 4.6: Cameras used for the prototyping process

4.2.3 Spectral sensitivity of Camera Sensors

While calibrating the intensity of the two monochromatic LED sources in order to achieve a similar lever of intensity for each color a slight abnormality was observed. The red color needed to have a higher intensity value compared to the blue intensity value in order to achieve the same intensities in the histogram. This is the result of the CMOS sensor's spectral sensitivity. Color sensors usually have three different types of photosites that are sensitive to light in different bands of wavelengths. CMOS sensors have a different sensitivity according to the wavelength of light they face [23]. The spectral sensitivity of a CMOS sensor is on figure 4.7.

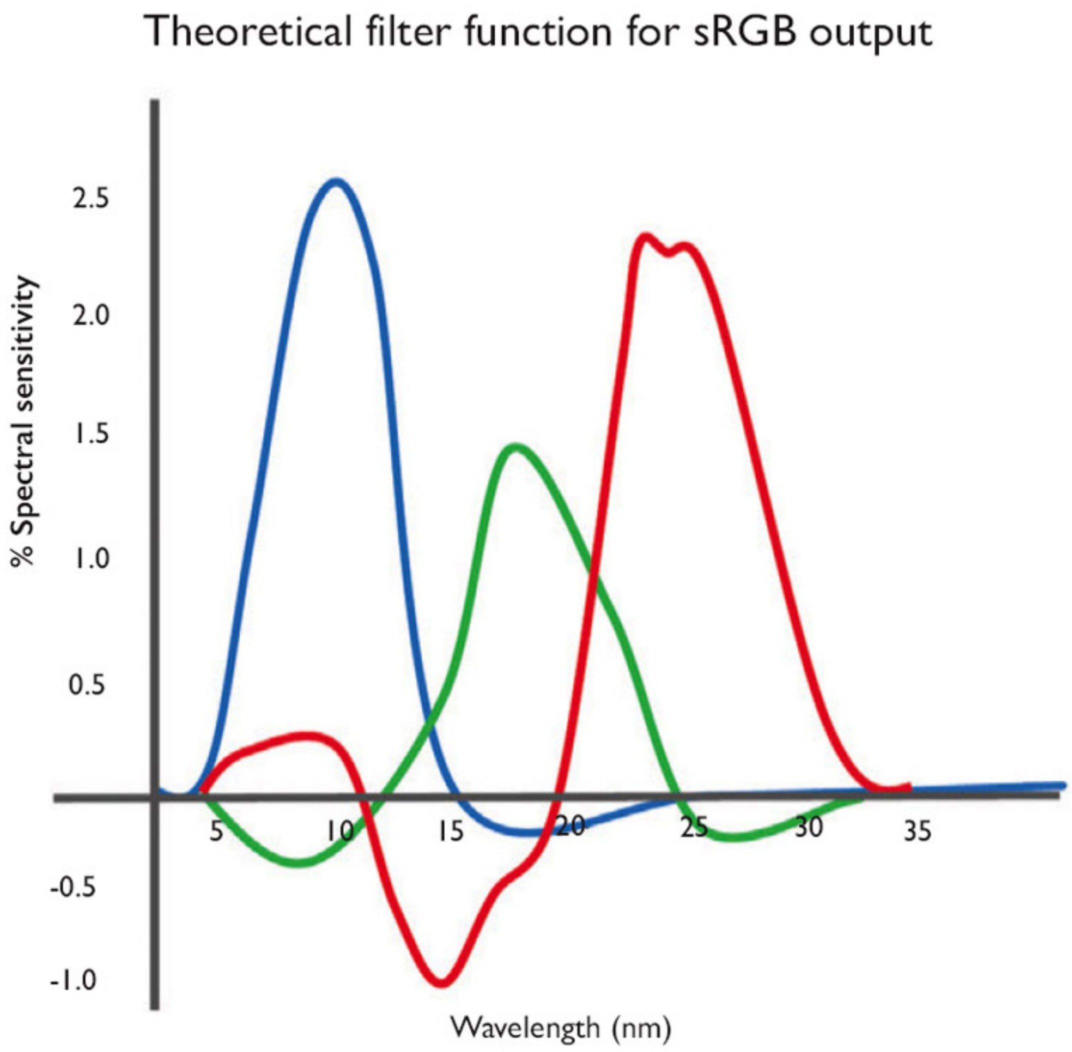


FIGURE 4.7: Spectral Response of CMOS sensor. Source: [23]

This difference in spectral sensitivity explains the result recreated in the camera experiment. The blue LED source is configured to have a lower intensity than the red source. The cameras using CMOS sensors are more sensitive to lower wavelengths in the visible spectrum [24]. Those wavelength are mainly responsible for blue light. All types of sensors have different spectral response to different light wavelengths resulting in different ratios of illumination in order to achieve a similar intensity level. This is a very important conclusion if the system needs to change scale, because if the implementation of the system is on a large scale the intensity of the red source will need to be bigger that the blue one or in very small scale this spectral response difference may result in alteration of the results.

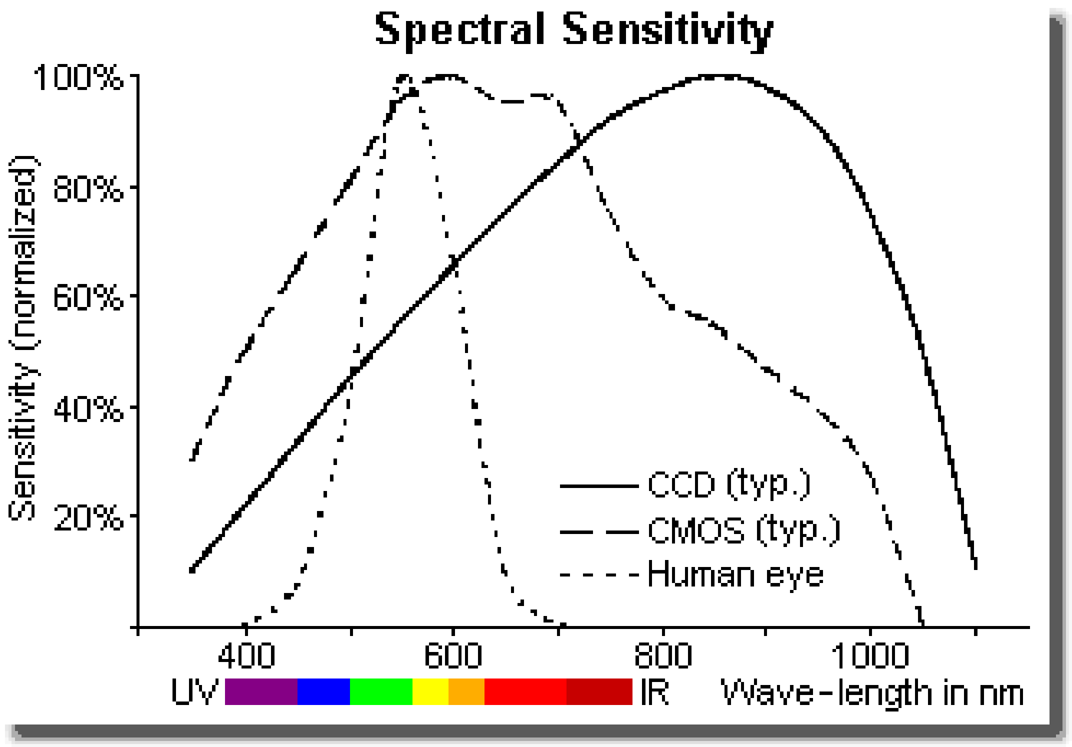


FIGURE 4.8: Spectral sensitivity of common sensors.
Source: [24]

4.2.4 Connecting DSLR to Linux

The difference between a common commercial USB webcam and a professional DSLR camera is that the first one is intended to be used with a computer and be able to communicate efficiently. This means that a webcam is usually UVC compliant making the process of transmitting data to a computer easier. This is not the case for a DSLR camera. For this system to work correctly a bridge for communication between the DSRL camera and the Raspberry Pi 4 used need to be constructed. A series of open source packages is used in order to achieve this goal.

- gphoto2 [25]

- ffmpeg [26]
- v4l2loopback Kernel Module [27]

Gphoto2 is a camera capturing program responsible for communicating with the camera and capturing the imaging data. It can be configure in order to take an image or capture a video.

FFmpeg is a multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play any media. It is used in order to transcode the video from the camera in to the Linux system.

V4l2loopback (Video4Linux2) is a module that allows you to create "virtual video devices". Normal (v4l2) applications will read these devices as if they were ordinary video devices, but the video will not be read from e.g. a capture card but instead it is generated by another application.

The final code used is bellow.

```
gphoto2 --stdout --set-config liveviewsize=0 --capture-movie
| ffmpeg -i - -vcodec rawvideo -pix_fmt yuv420p -threads 0 -f
v4l2 -s:v 1920x1280 -r 25 /dev/video0)
```

The provided command is a combination of the two utilities, `gphoto2` and `ffmpeg`, and is used for capturing live video from a camera and streaming it to a virtual video device on `/dev/video0`. Let's break down the command:

```
gphoto2 --stdout --set-config liveviewsize=0 --capture-movie
```

This part of the command uses `gphoto2`, a command-line interface for digital cameras, to capture live video. The options used are:

- `--stdout`: Directs the output to the standard output.
- `--set-config liveviewsize=0`: Sets the live view size to 0 which is the highest resolution.
- `--capture-movie`: Initiates the capture of a video.

The output of this command is then piped (`|`) to the next part:

```
ffmpeg -i - -vcodec rawvideo -pix\_fmt yuv420p -threads 0 -f v4l2
-s:v 1920x1280 -r 25 /dev/video0
```

This part uses `ffmpeg` to process the video stream received from `gphoto2`. The options used are:

- `-i -`: Specifies the input source as the standard input.

- `-vcodec rawvideo -pix_fmt yuv420p`: Sets the video codec to raw video and the pixel format to YUV420p.
- `-threads 0`: Uses the optimal number of threads.
- `-f v4l2`: Specifies the output format as Video4Linux2.
- `-s:v 1920x1280`: Sets the video size to 1920x1280 pixels.
- `-r 25`: Sets the frame rate to 25 frames per second.
- `/dev/video0`: Specifies the output device as `/dev/video0`.

In summary, this command captures live video from a camera using `gphoto2` and then processes and streams it using `ffmpeg` to a virtual video device (`/dev/video0`) with specific settings for video size and frame rate.

Using this command it easy to import the virtual video device in the C++ code as an input and continue with the computation of the depth estimation.

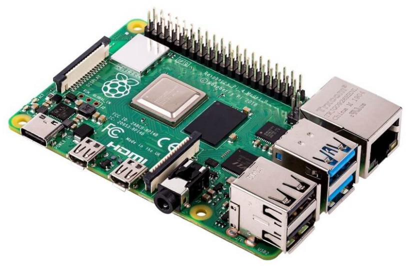
4.3 Processing subsystem

4.3.1 Processing Requirements

- Small form factor
- Low energy consumption
- Low cost
- Multiple Processing Cores
- Adequate DRAM ($DRAM \geq 4GB$)
- USB connectivity
- Graphical Interface for Visualizing Live Results
- Ability to run Linux

4.3.2 Candidate Embedded Linux Systems for implementation

Two very popular options in the field of embedded systems - as central processing units - are the Raspberry Pi boards manufactured by the Raspberry Pi Foundation in collaboration with Broadcom, and Nvidia's Jetson boards. Figure 4.9 shows as an example an indicative version from each family, while in Table 4.1 and Table 4.2 the technical characteristics of each board are presented as a starting point.


(A) Raspberry Pi 4

(B) Nvidia Jetson Nano

FIGURE 4.9: Candidate Embedded Linux Systems for implementation

Raspberry Pi 4 Specifications	
CPU	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
SDRAM	8GB LPDDR4-3200
Connectivity	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet
USB	2 USB 3.0 ports 2 USB 2.0 ports
Peripherals	GPIO, I2C, SPI, UART
Visual Outputs	2 × micro-HDMI® ports (up to 4kp60 supported)
Graphics Systems	H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode) OpenGL ES 3.1, Vulkan 1.0
Storage	Micro-SD card slot for loading operating system and data storage
Power	5V DC via USB-C connector (minimum 3A)
Price	116€

TABLE 4.1: Raspberry Pi 4 Specifications

Nvidia Jetson Nano Developer Kit Specifications	
CPU	Quad-core ARM Cortex-A57 MPCore processor
SDRAM	4 GB 64-bit LPDDR4; 25.6 gigabytes/second
Connectivity	Gigabit Ethernet
USB	4x USB 3.0 Type-A connectors 1x USB 2.0 Micro-B connector
Peripherals	GPIO, I2C, SPI, UART
Visual Outputs	1x HDMI 2.0, 1x DP 1.2
Graphics Systems	1x 4K30 | 2x 1080p60 | 4x 1080p30 | 9x 720p30 (H.264/H.265) 1x 4K60 | 2x 4K30 | 4x 1080p60 | 8x 1080p30 | 18x 720p30 (H.264/H.265)
Storage	Micro-SD card slot for loading operating system and data storage
Power	5V DC via USB-C connector (minimum 3A)
Price	220€

TABLE 4.2: Raspberry Pi 4 Specifications

4.3.3 Processing System selection

According to the above requirements a small **Raspberry Pi** 4.10 single-board computer (SBC) is selected.

This single-board computer has good enough specifications to create a prototype system with enough computing power in order to run the depth estimation algorithm. Also it has enough IO ports in order to process all the data coming from the camera and create an output in a screen for the final user to understand in great detail the differences in the depth of the scanned surface.

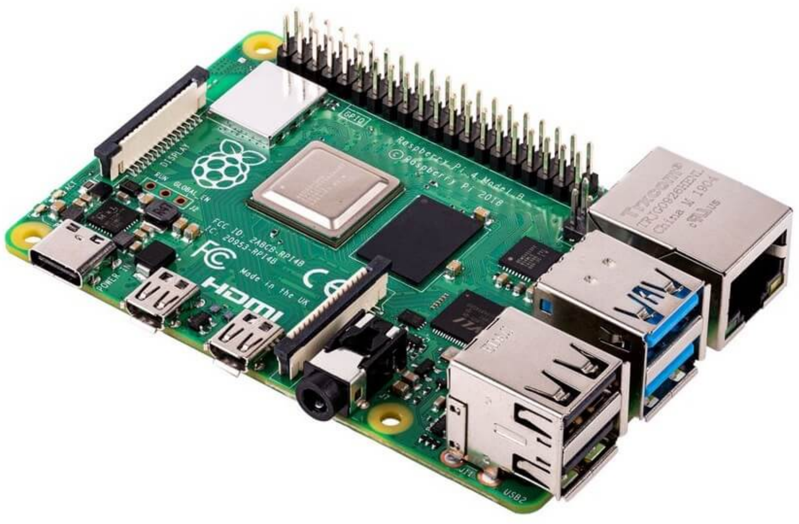


FIGURE 4.10: Raspberry Pi 4 8 GB

The most important reason that led in the selection of this specific single-board computer is the Cortex-A72 CPU chip. As the successor of the ARM Cortex-A57 used in the Jetson Nano, it designed to use 20% less power and offer 90% greater performance which is very crucial because the processing for the depth estimation is using the CPU and it is not optimized for GPU acceleration. Furthermore, very important is the implementation of **VFPv4** Floating Point Unit onboard (per core). VFPv4 has 32 64-bit FPU registers as standard, adds both half-precision support as a storage format and fused multiply-accumulate instructions to the features of VFPv3. This is a big advantage especially in relation to most of the microcontrollers or microprocessors in this price range of $\leq$ 100€ which is improving the efficiency of division floating point division calculations that are needed for the depth estimation of every frame processed.

Maximum input power is also a very important aspect for an embedded system designed to work outside where there is no power from the grid. This single-board computer uses a maximum of 15 watts of power, creating a system that can easily work from a portable system like a battery or some type of renewable energy.

Encoding and decoding subsystems are also very important and create an efficient and fast system that is able to compute large amounts of data in a smaller time frame using lower amounts of energy. In this specific application H264 encoder, decoder and OpenGL are necessary because video uses H264 encryption and after the depth estimation the output is shown to the final user using OpenGL Graphic standard.

Finally the price difference with all these extra advantages shows that the logical decision is the Raspberry Pi 4 costing half the amount of Jetson Nano while having a 90% performance advantage. The only reason for the selection of the Jetson Nano would be the implementation of some type of GPU acceleration in which case the GPU of the Jetson Nano would make it the perfect choice.

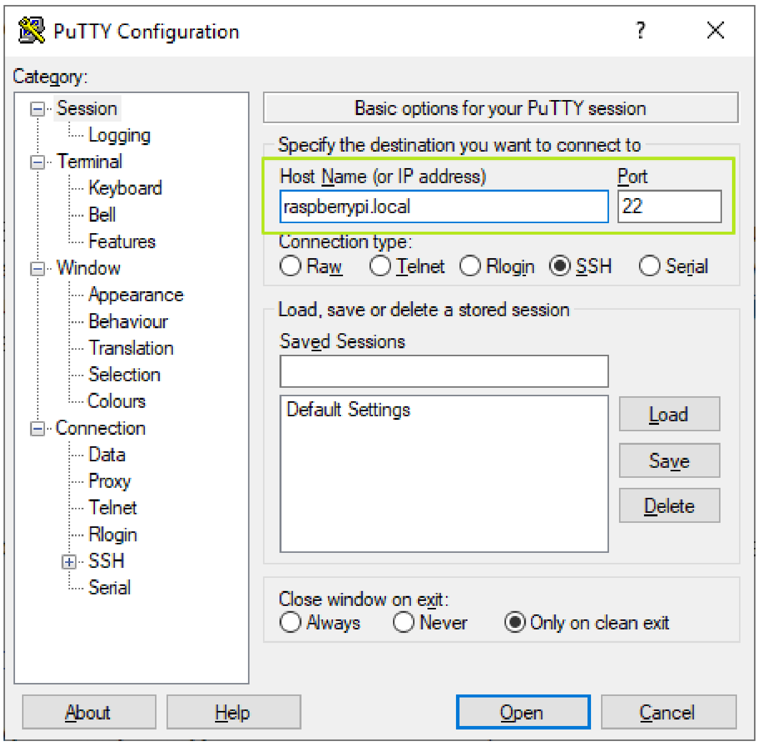
4.3.4 Operating System Selection

Since the development of the program used for the depth estimation is done with the c++ language and the OpenCV Library it was only logical to combine it with Ubuntu Linux. A specially designed version of Ubuntu Linux was installed on the Raspberry Pi 22.04.3 64bit version for ARM [28] architecture. Essentially all functions of the created application are controlled from this OS which is an Embedded General Purpose Operating System, which means that it will be mainly responsible for the operation of the scheduling, file-system abstraction, networking, etc.

4.3.5 Local computer communication with the Embedded System

Running the system while being indoors has many benefits. One of them is always having a stable internet connection with a router able to control the communication between a local computer and the embedded system in development. For the outdoors testing and final prototyping of the system the problem of communication between a local computer and the final embedded system had to be solved. In order to solve this problem an easy and stable way of communication between a local laptop and the Raspberry Pi4 was found. Many different ways of communication were tested, like using a phone to act as the router and create a small network but this was not stable enough. The final and most stable was SSH communication with the Raspberry Pi 4 over a single Ethernet cable connection [29]. Raspberry Pi

4's network module is smart enough that a regular Ethernet cable suffices in order to create a connection. After enabling SSH on the Raspberry Pi 4 simply connect the local computer with the remote Embedded with the Ethernet cable.



Specify the destination you want to connect to

Host Name (or IP address): raspberrypi.local
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions:

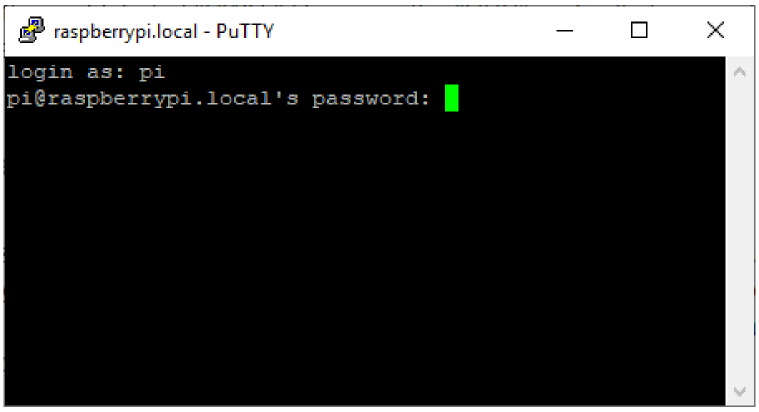
Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

FIGURE 4.11: Putty Configuration

Using a program like Putty or Bitvise SSH client create a connection using the Host Name raspberry.local on port 22 as shown on 4.11.

The next step is to input the Raspberry's username and password and the connection is ready as shown on 4.12.



```
login as: pi
pi@raspberrypi.local's password:
```

FIGURE 4.12: Password Insertion

4.4 Shield subsystem and completed System

In order to complete the system the need for some auxiliary components exist. First and foremost the whole system is covered from an opaque shield. The main focus of the shield to keep light from other light sources from reaching the scanned surface in order to maintain the greatest quality of the results. The second use of the shield is as a support platform for all the other subsystem to be installed correctly. Finally the shield protects the system from splashes and general pollution that can possibly interfere with the correct operation of the system.

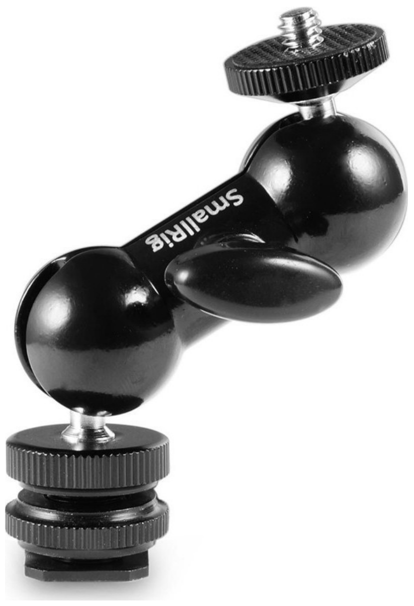
A simple and low cost solution was found for the shield of the system. A black opaque 60 liter organisation box was purchased and used inverted presented on figure 4.13. This created a great hollow space where the lighting and the camera subsystem were able to be installed while blocking all light from outside sources.

FIGURE 4.13: Plastic Box used as the shield

4.5 Auxiliary components

4.5.1 Camera Stabilisation System

A very important problem that needed to be addressed was the camera stabilisation in way that the camera and the other components of the system would stay stable on each other while the camera needed to hang from the ceiling of the shield frame without falling. A brilliant solution to the problem was a double end ball Camera articulating arm system equipped with 1/4" screws in both ends. One end is mounted on the plastic shield with some rubber rings for sock absorption purposes and the other one is mounted to the camera. This system as shown in figure 4.14 gives the ability to the system for easy and fast camera mounting and unmounting while it is possible to fine tune the scene that the camera is pointing to. It is possible to tilt and move the camera horizontally and vertically to some degree of freedom in order achieve the best camera position.



(A) Double End Ball Camera Stabilisation Arm

(B) Rubber Rings for Shock absorption

FIGURE 4.14: Camera Stabilisation and Shock absorption

4.5.2 Wheels and tires

The system needs to be a moving platform in order to read road surfaces. For this to be accomplished a type a wheel should be implemented in order to help the system move on the road. For this reason a stable type of manufacturing wheel, as shown on figure 4.15, was chosen. These wheels have a diameter of 12.5 cm and total clearance of 15 cm. This was important because larger diameter will ensure more vibration reduction and 15 cm clearance means that the system will be able to detect objects as high as 15 cm. Furthermore, a rubber tire variation was chosen for added shock absorption.

FIGURE 4.15: Stable Wheel

4.5.3 LED Dimmers

Last but not least auxiliary components of the system are the LED dimmers, like the one shown on figure 4.16. These devices are crucial for the correct intensity configuration of the colored LED light sources. With the LED dimmers it is easy to set the light intensity of each LED strip independently in order to achieve the exact same light level to both light sources. When applying the depth estimation algorithm in different types of surfaces it is crucial to be able to change easily and fast the light output of the LED sources because different materials and colored surfaces have different color reflections.

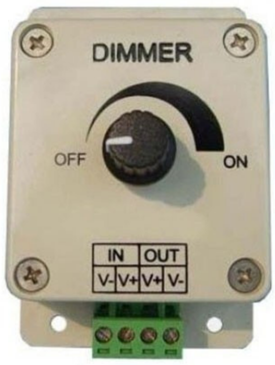


FIGURE 4.16: LED Dimmer

4.6 Finished System

4.6.1 Inside View of the shield

On the image 4.17 below an overview of the final inside look of the system can be observed. As shown the camera is mounted on the top roof of the inside of the plastic shield and the two light sources are mounted on the sides.

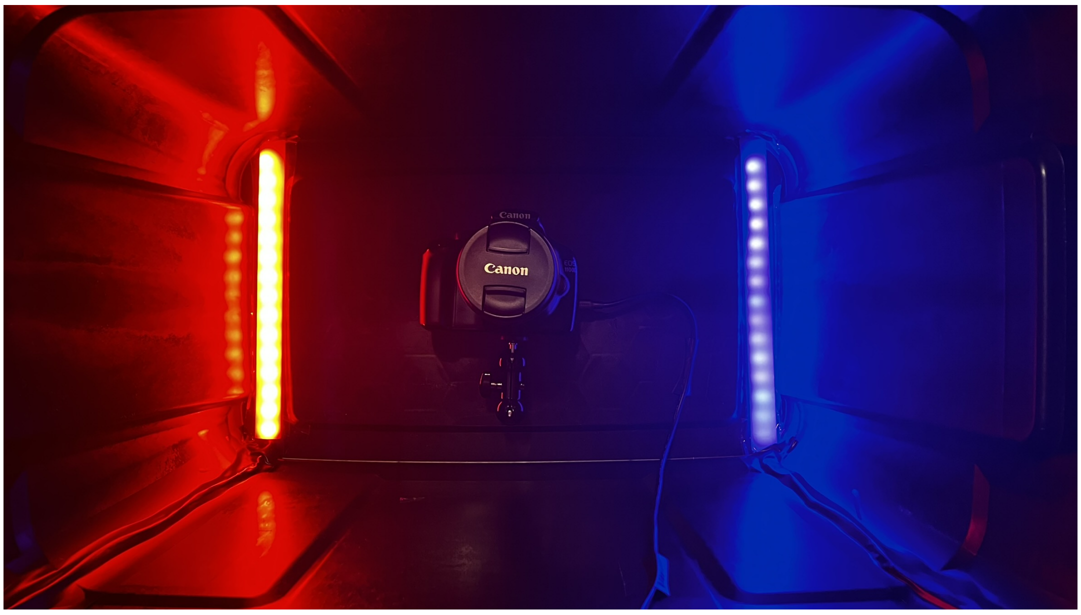


FIGURE 4.17: Inside View of the system

4.6.2 Outside look

On the below image 4.18 a final overview of the outside of the completed working system is shown. On this image it easy to understand the basic mechanics of the resulting system. As the light sources are enabled and the white ball is placed as an object ready to be scanned.

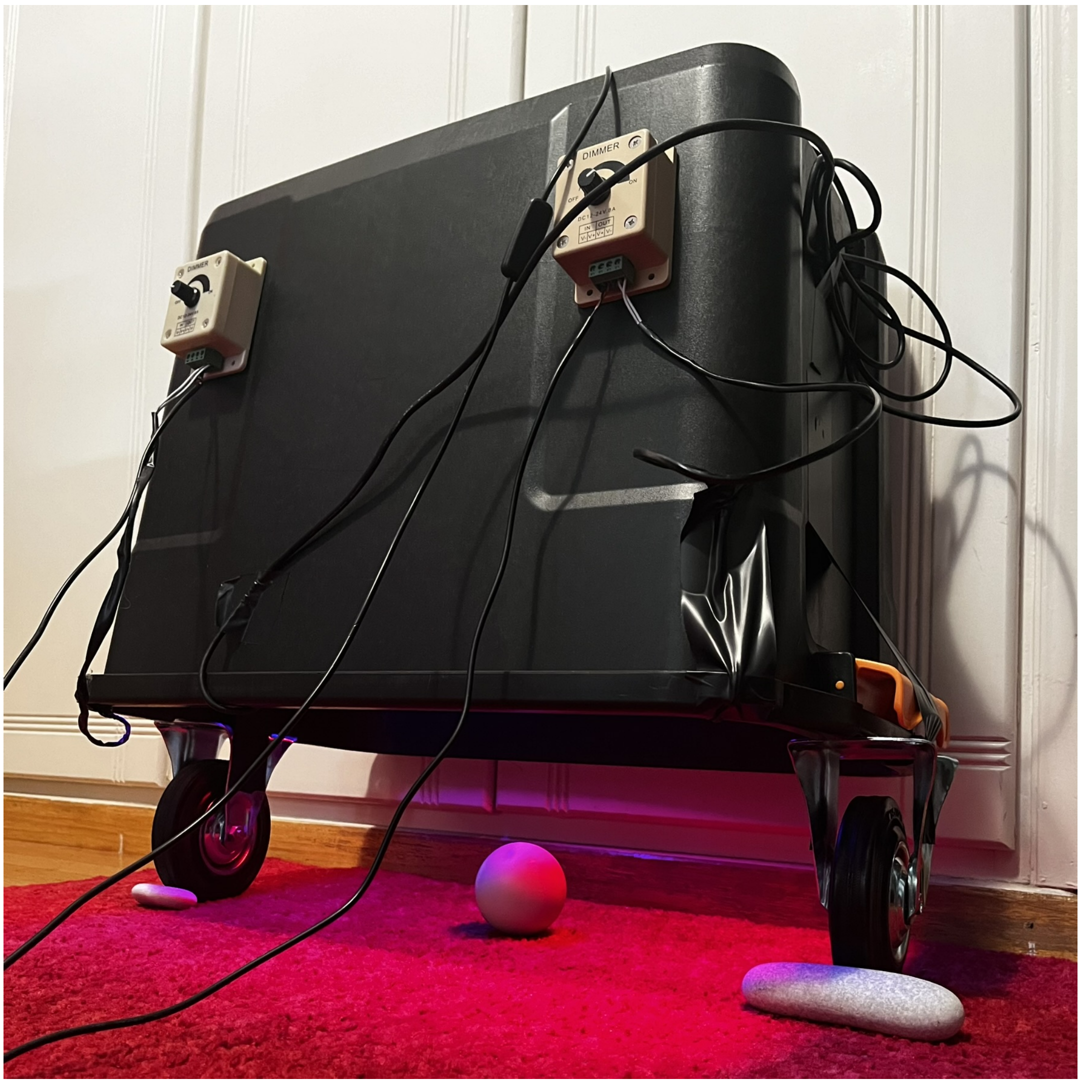


FIGURE 4.18: Outside look of the system

Chapter 5

System Verification and Performance Evaluation

This chapter describes the actions taken in order to to verify the performance of the system. The testing procedure was divided into two stages. The first, which was carried out in an indoor environment at the same time as the implementation of the system in order to verify its ability to recognize the height of objects. The second stage, which was carried outdoors with all the equipment being tested in a "real life" and more challenging environment with all the possible problems this can create. In this section we will focus on the second stage of the testing procedure which was mostly the verification stage of the Thesis. In this stage the purpose is to compare the results extracted with the expected results and discover which parts of the system perform as expected and which parts form bottlenecks that reduce the effectiveness of the system.

5.1 Testing environment of the data collection

The environment of the data collection needed to be outside in a realistic environment with enough places to acquire data and low traffic in order to enable the data acquisition process without problems.

5.2 Data acquisition process

The process of data acquisition is simple. The system was set up and configured. First, the calibration process took place. The camera was adjusted for the scene using the Stabilisation Arm so that the camera can face all the scanned surface correctly. The next step was the LED sources calibration. The LED sources were switched on and with the data received in real time from the camera the color intensity histogram was created. The light intensities of the two light sources were equalized using the LED dimmers according to the outputted histogram. After the calibration process was finished the scanning of different surfaces started. First the system is above a part of the surface were no anomalies are detected and it known from the user beforehand. This way the system can acquire the first frame of the flat surface which will help normalize all the next frames. Next the system is moved to another position where it is constantly outputting to its monitor the depth estimation of the currently scanned region.

5.3 System Verification Process

In this part of the Thesis a single Frame from the outdoor testing experiment will be presented from the beginning as input to the end as output, explaining all the inner phases it passes in order for the algorithm to calculate the depth of the objects shown in this frame.

First part in the process is obtaining the normalization frame 5.1 . This frame is obtained when the depth estimation is beginning and is mostly used for the system to calibrate the way the light is reflected in the specific color and material that is about to be scanned. This frame is responsible to help the system understand if in the next frames a pixel's specific value will be due to a slope existing or due to the proximity of this pixel to light source compared to other pixels. This way even though in the picture it is possible to detect where the hot spots of the LED colored sources are these will not create any problems on the system's calculations.

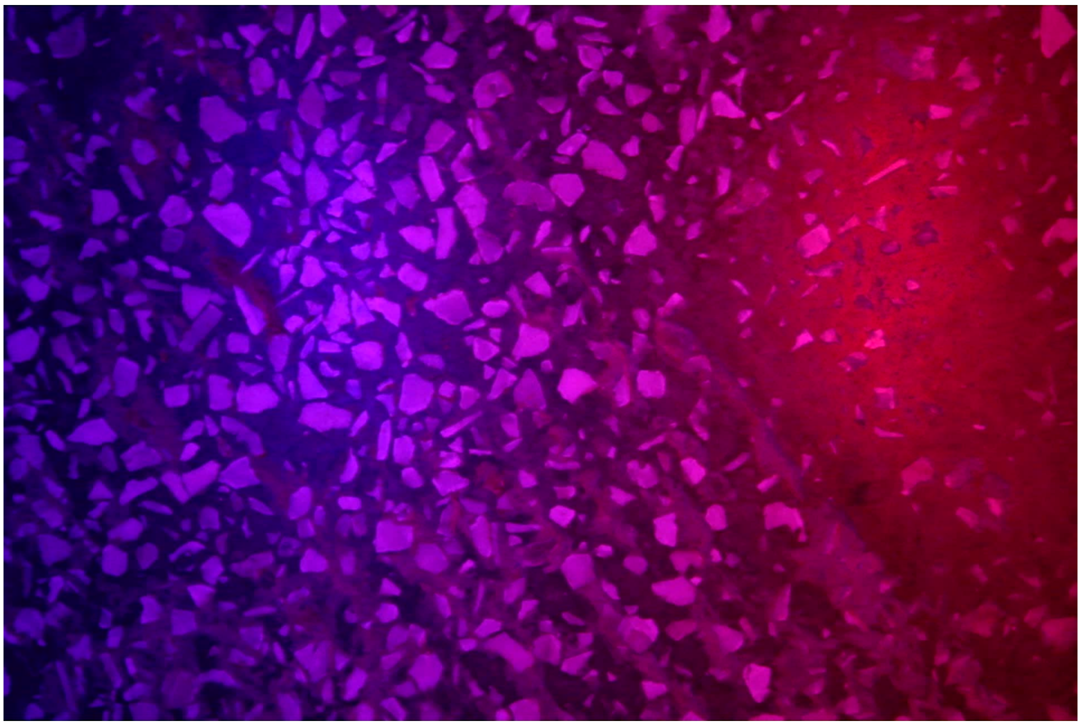


FIGURE 5.1: Normalization Frame

Next part is a random frame in this whole process. This is the frame that the system is processing at the moment as shown on 5.2. This frame is the part of a gutter with some fallen leaves and a very small rock the depth of the gutter is measured at about 2.6 to 2.9 cm at its different parts.

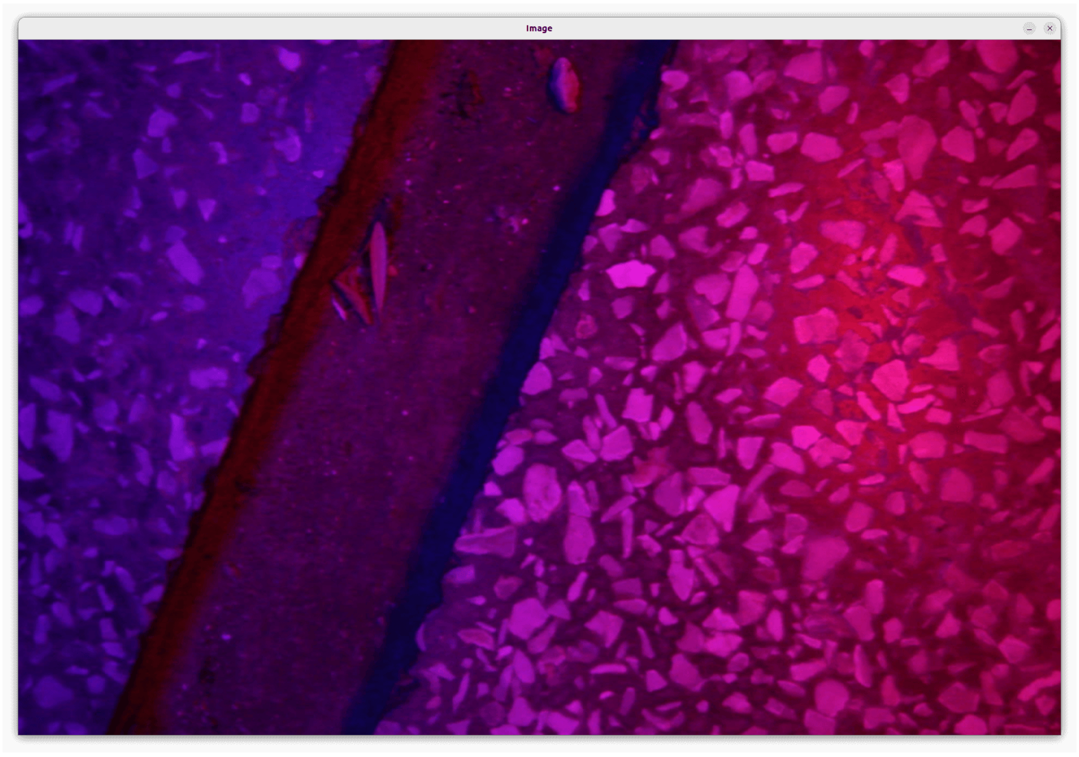


FIGURE 5.2: Input Frame

After the first part of the processing the color ratio matrices are being created. On figure 5.3 a representation of the *Blue*/*Red* matrix is shown. In this matrix it is useful to point that the blue light is coming from the left the frame. The parts illuminated more from the blue light appear a lot more white than the parts illuminated more from the red light, while shadows of the blue light appear darker and almost black.

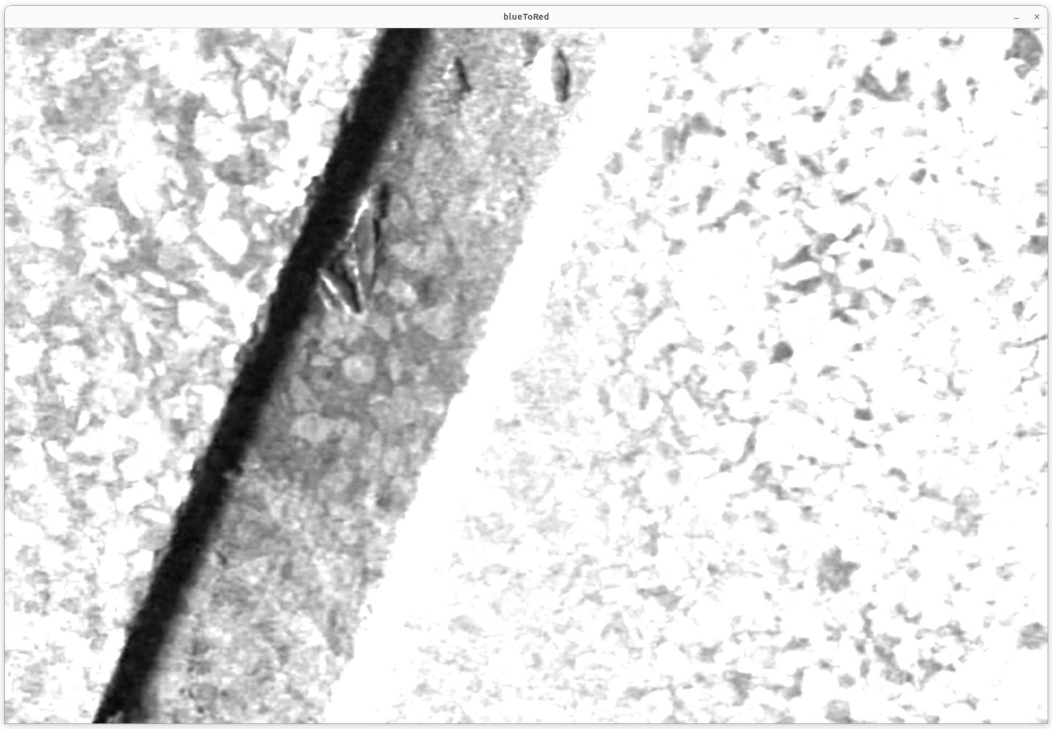


FIGURE 5.3: Blue to Red Ratio Frame

The same way as the previous photo mentioned on figure 5.4 the representation of *Red*/*Blue* matrix is shown. In this matrix parts illuminated more from the red light appear a lot more white and also shadows of the red light appear darker and almost black.

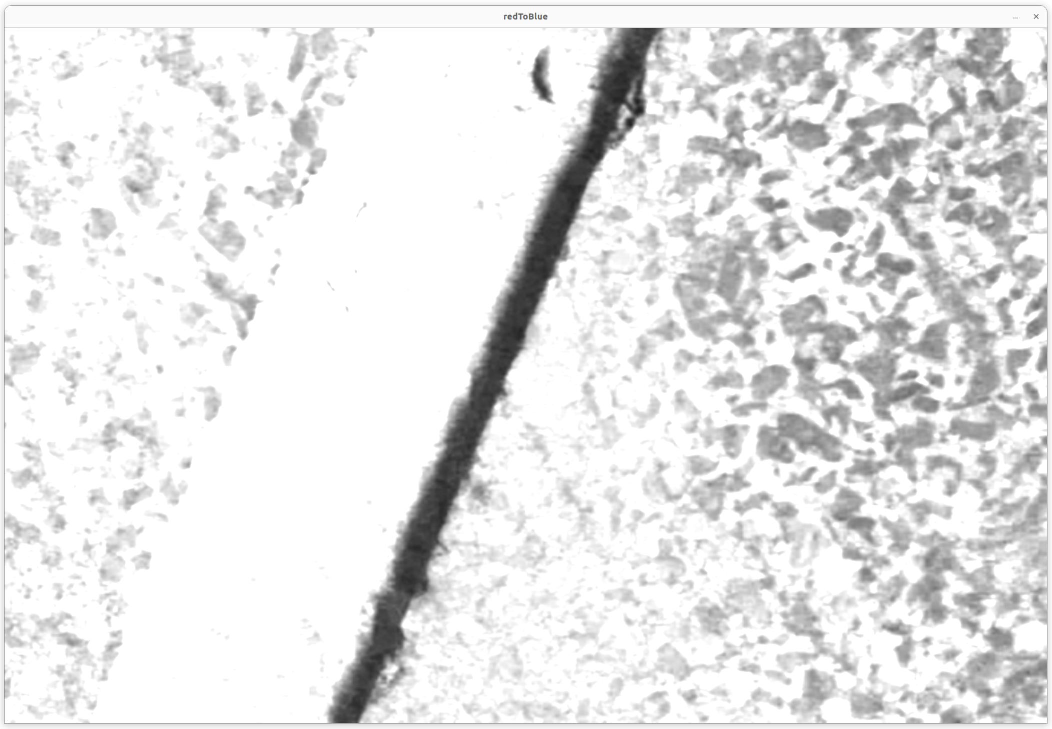


FIGURE 5.4: Red to Blue Ratio Frame

Combining the information given from the previous two matrices, another matrix is created that represents all the parts of the frame that a slopes occurs as shown on figure 5.5. This is very important information because those are the parts that will be further processed in order to obtain the final output for the final user. As shown on the previous representations white is indicating the parts of the frame that some kind of slope is occurring. Currently there is no information about how steep is the slopes or of it is deeper or higher of the current level. Also while virtually all the frame has slopes, most of them are very small because the surface is coarse.

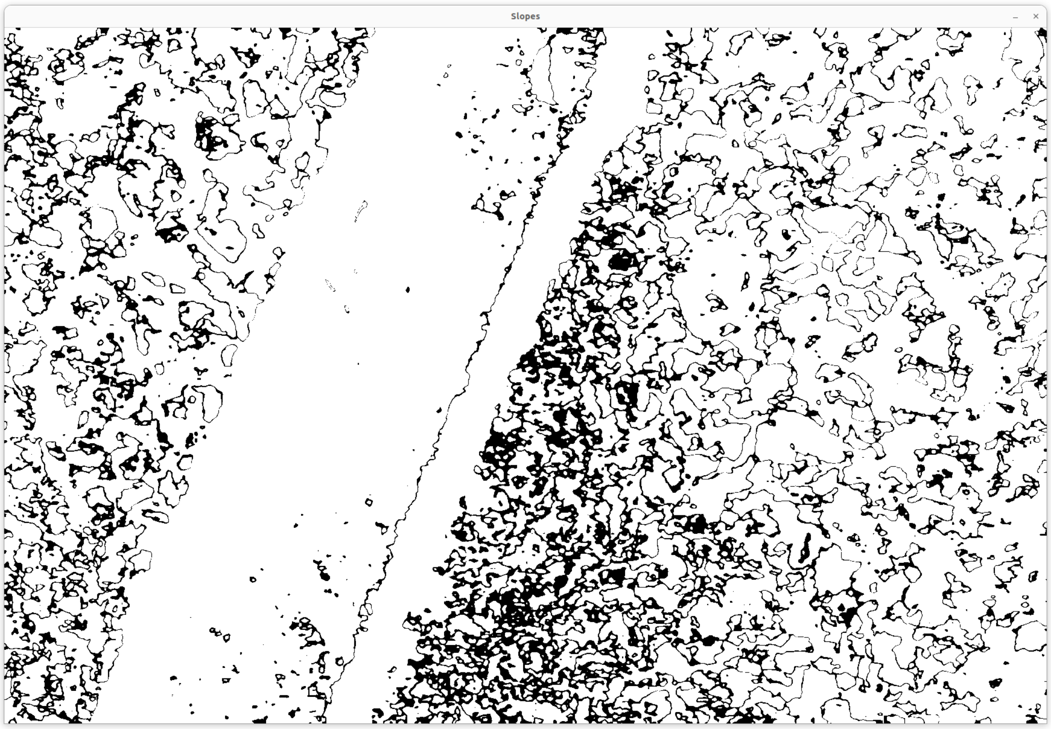


FIGURE 5.5: Spotted Slopes Frame

Finally, using all the matrices a final more user friendly representation of the result is constructed as shown on figure 5.6. In this representation, green represents the normal level while bluer parts of the image indicate the part of the surface that is under the normal level that the system started during the calibration. Using this simple color code it easy for the final user to understand and estimate the level change of the surface in real time. In this final image all the objects in the input image are easily identifiable with the added information about their relative distance from the normal level. Even the leaves and the rock are easily discernible while also providing the information that are lower that the normal level of the system which is on the upper lip of the gutter.

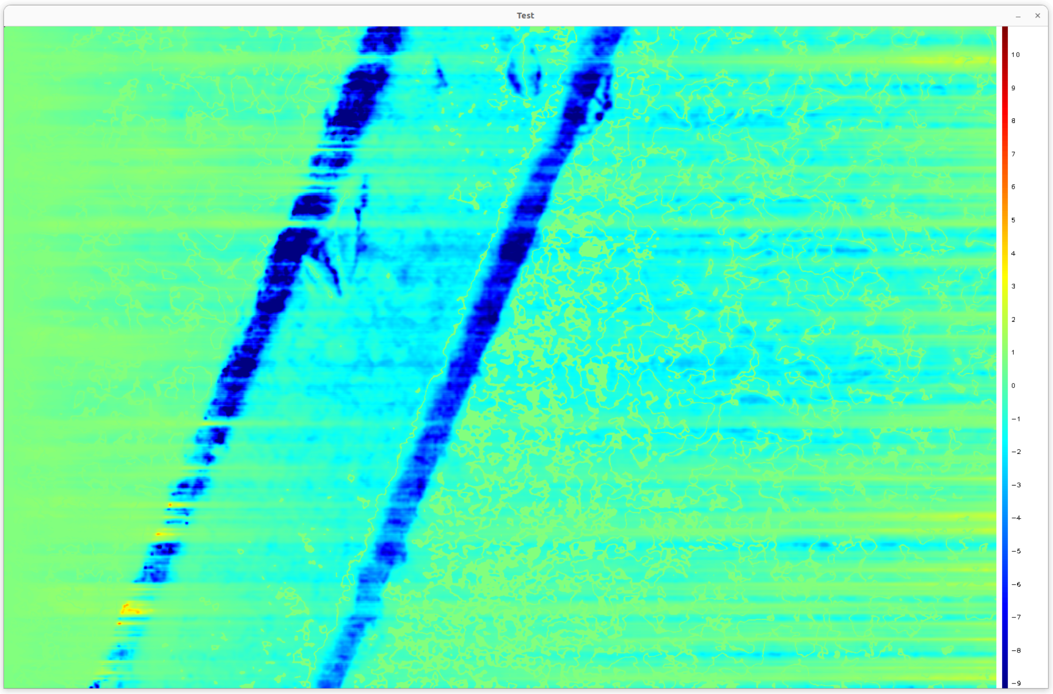


FIGURE 5.6: Output to Final User Frame

From the final output of depth estimation system the 3D mesh below is easily created showing a 3D representation of the information that the system was able to deduct from the input frame using the depth estimation algorithm presented in this Thesis. On figure 5.7 the 3D representation of the final result is presented from above the normal level as it created from the real time system. This image can easily verify the correct result. The measured depth of the gutter is about 2.7cm which is the same as the real measured depth of the gutter. Each 1cm is represented by 1000 points on the represented scale.

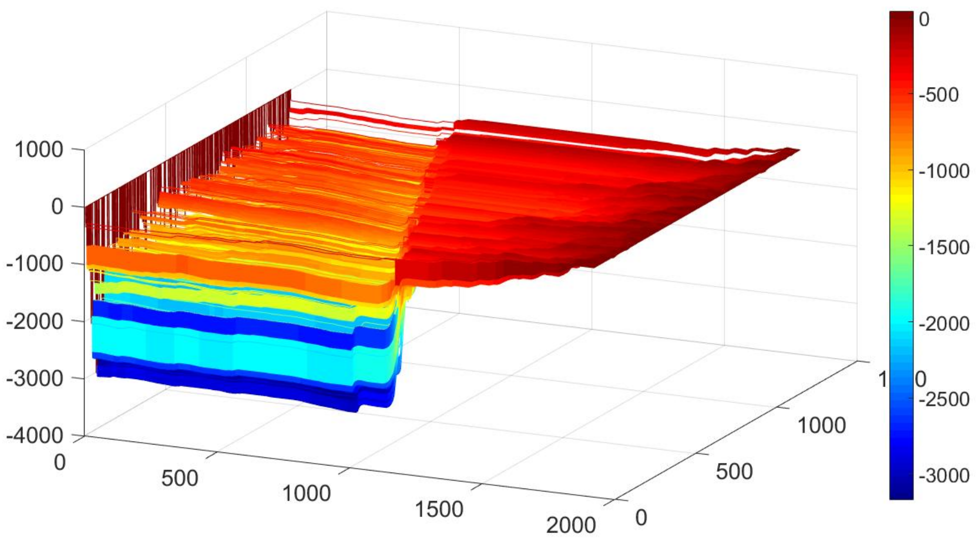


FIGURE 5.7: 3D Mesh of the image from Above

On figure 5.8 the same results are shown from a different angle underneath the tested surface. This shows how well the system was able to read the scanned surface and understand its height level differences.

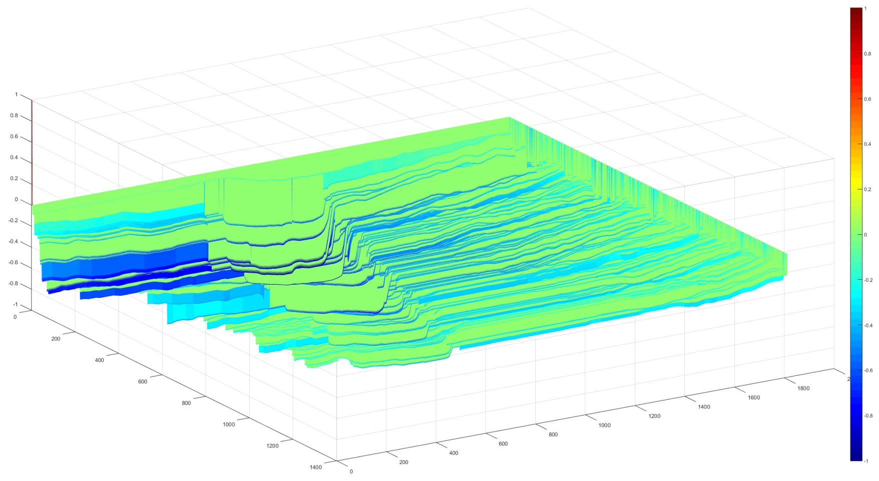


FIGURE 5.8: 3D Mesh of the image from Underneath

5.3.1 Testing small surface potholes

In the same manner as the previous testing procedure, one more challenging test for the system took place. In this scanned surface there are small holes and cracks on the surface. This test is presenting the ability of the system to detect very small defects on a road surface.

Again a normalization frame 5.9 was captured and the normalization factor were created.

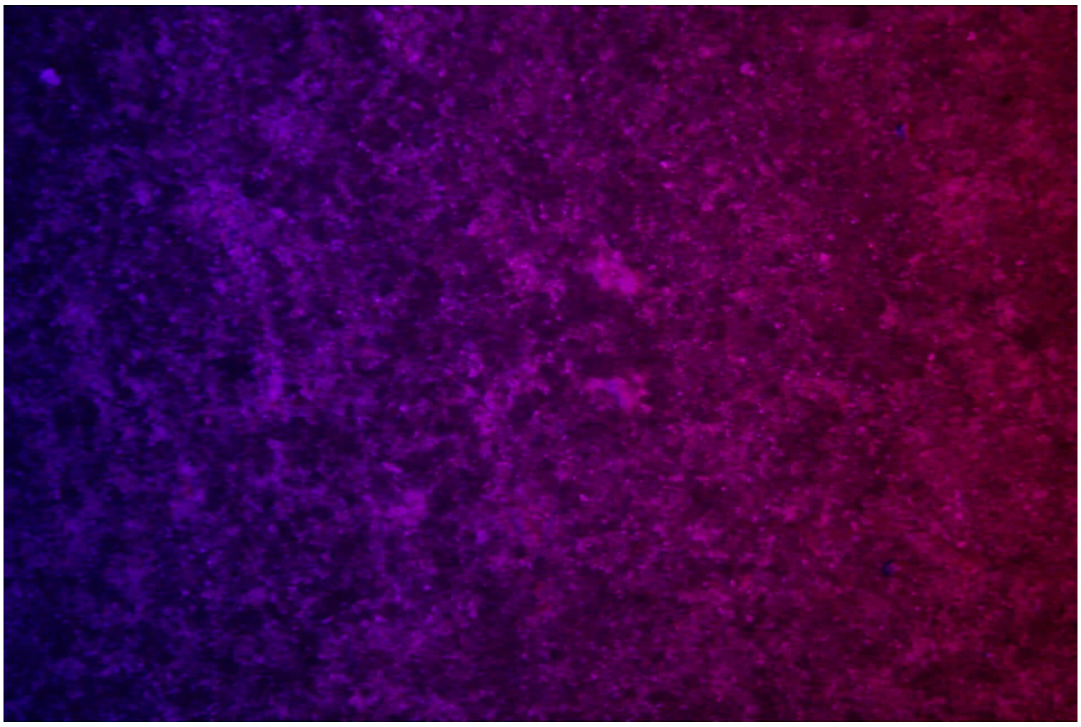


FIGURE 5.9: Normalization Image

Then as the system was hovering over the scanned area the frame shown on figure 5.10 was captured.

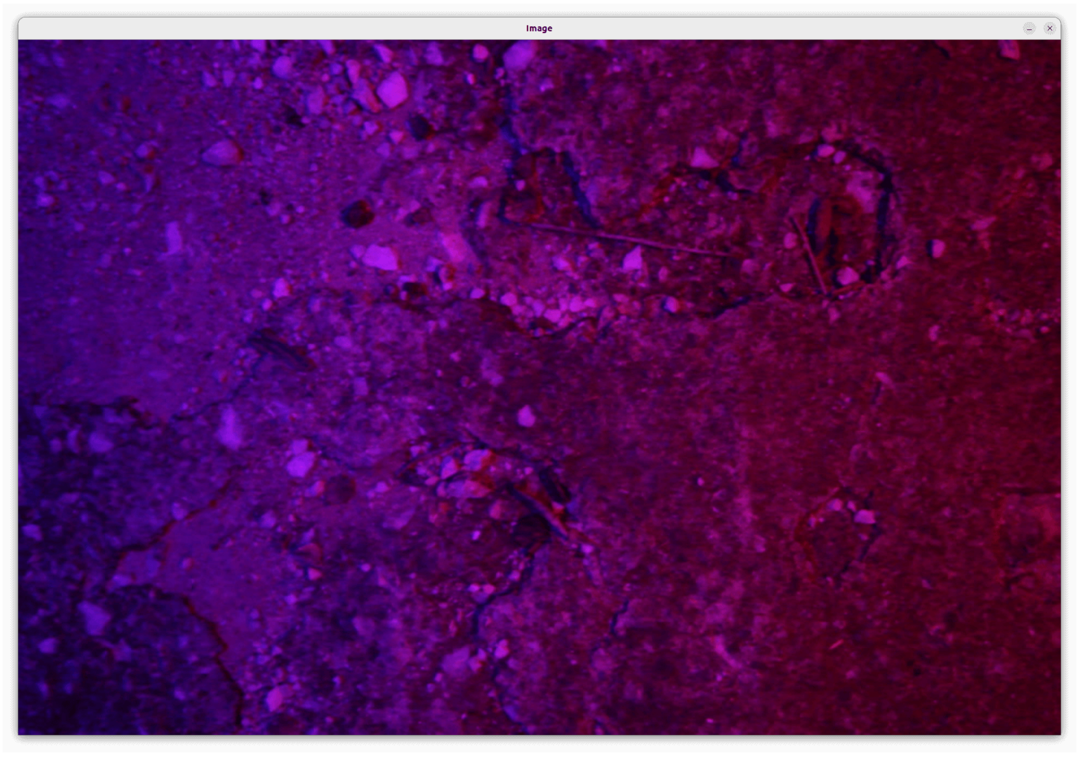


FIGURE 5.10: Input Image

Figure 5.11 shows the *Blue*/*Red* ratio of the scanned surface.

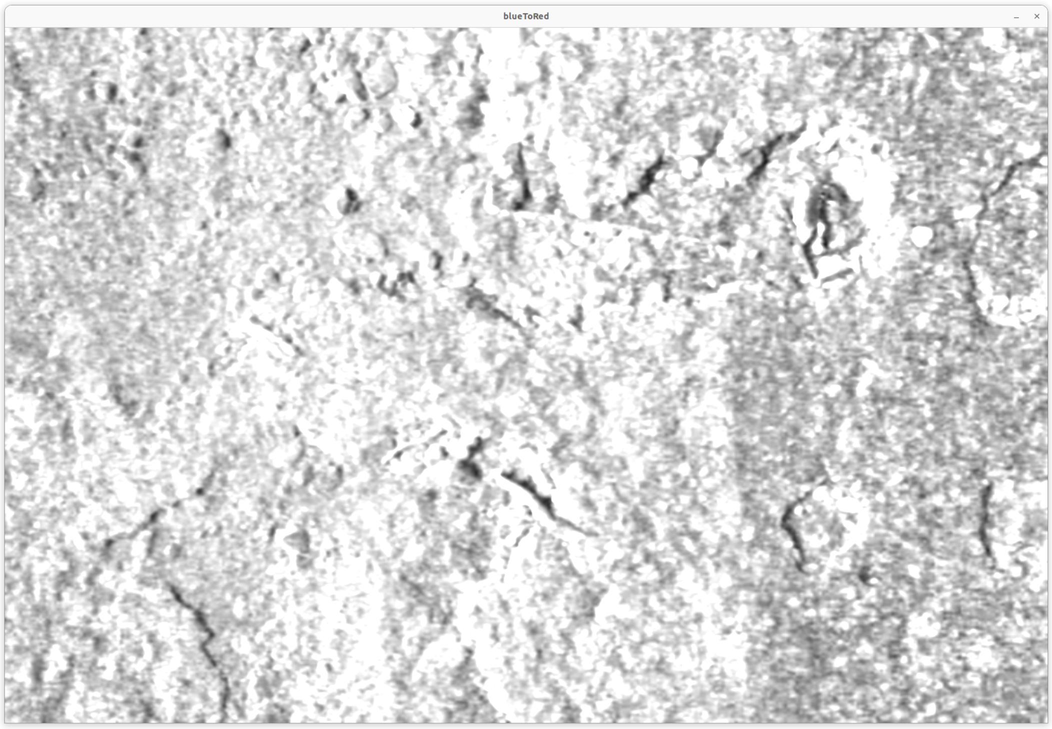


FIGURE 5.11: Blue to Red ratio image

In the same manner, figure 5.12 shows the *Red*/*Blue* ratio.

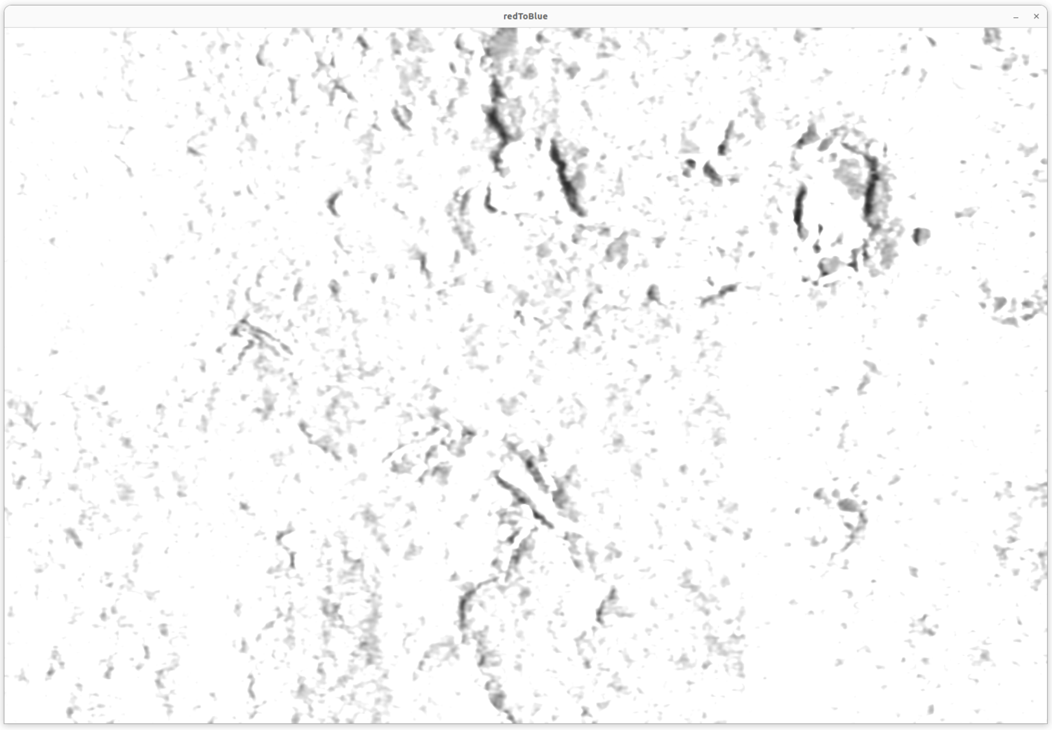


FIGURE 5.12: Red to Blue ratio image

All the detected slopes can be observed on figure 5.13.

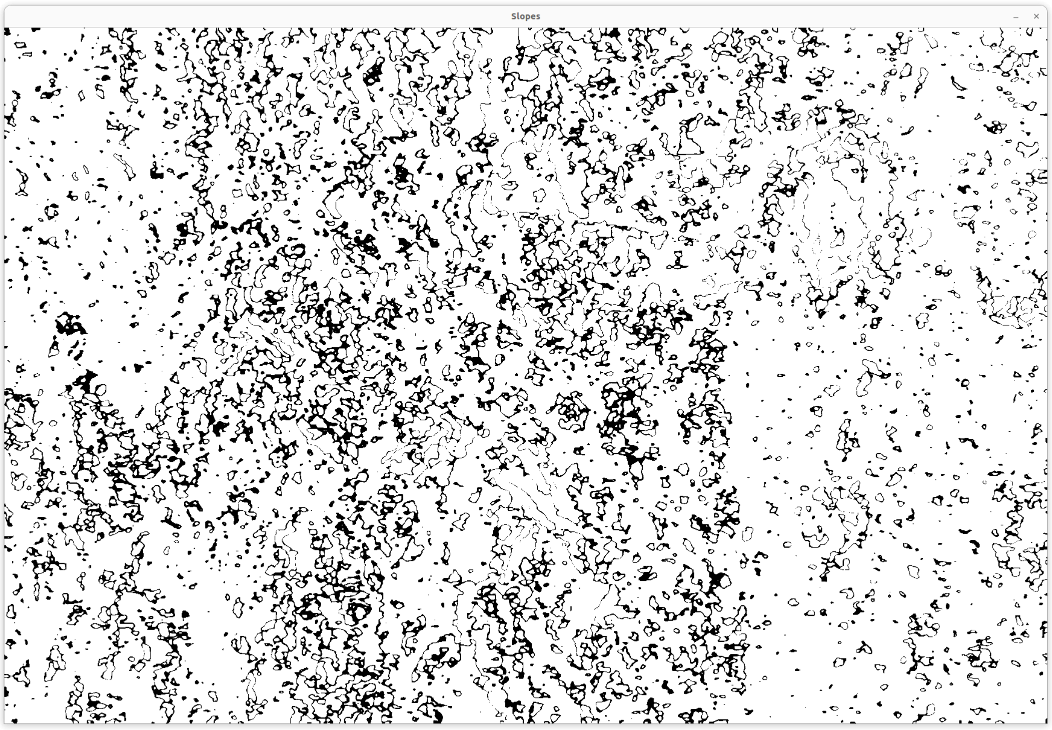


FIGURE 5.13: Slopes Image

And finally the output image for the user was as shown below on figure 5.14. The cracks and holes can be distinguished for the final user as needed. This show the ability of the system to detect even smaller cracks and holes on a road surface. An important observation is that on the most left part of the image that the surface has no or very small defects the output image fails to reproduce them. This can happen as the system was designed to work outside on the field and detect slightly larger defects so there can be more room for an increase in accuracy in future versions.

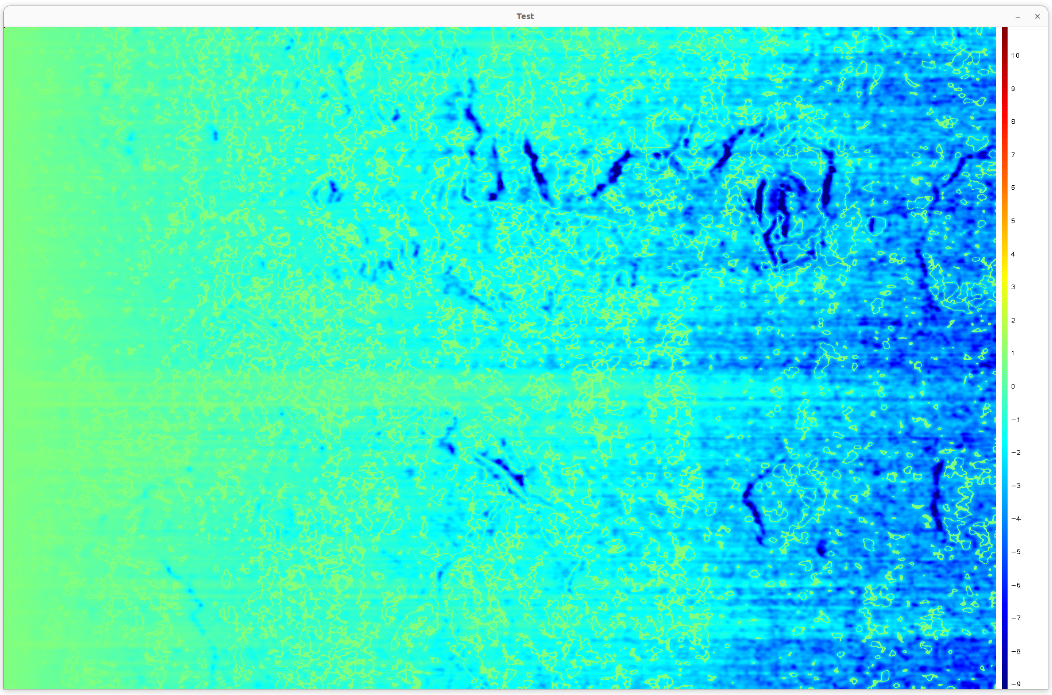


FIGURE 5.14: Output Image

5.3.2 Testing small rock

Another simple test was carried out, with a small rock as the test object. This test is a simple representation of a known object and it's output.

First the height of the rock is measured as shown on image 5.15. As shown the rock is measured as 1.9cm approximately.

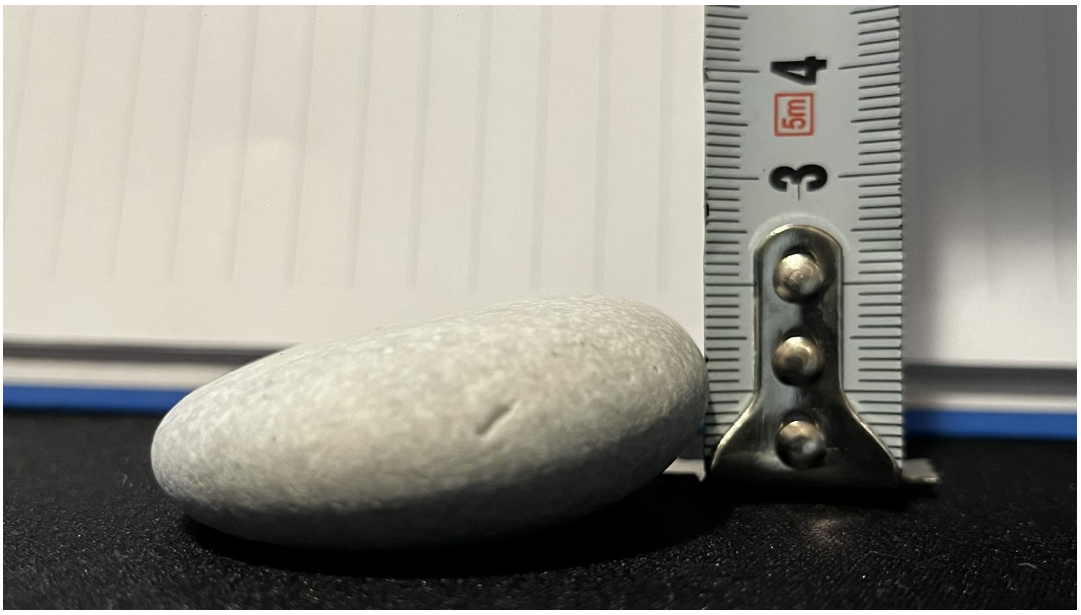


FIGURE 5.15: Height of the Rock

Next in image 5.16 the rock is shown from the lenses of the system.

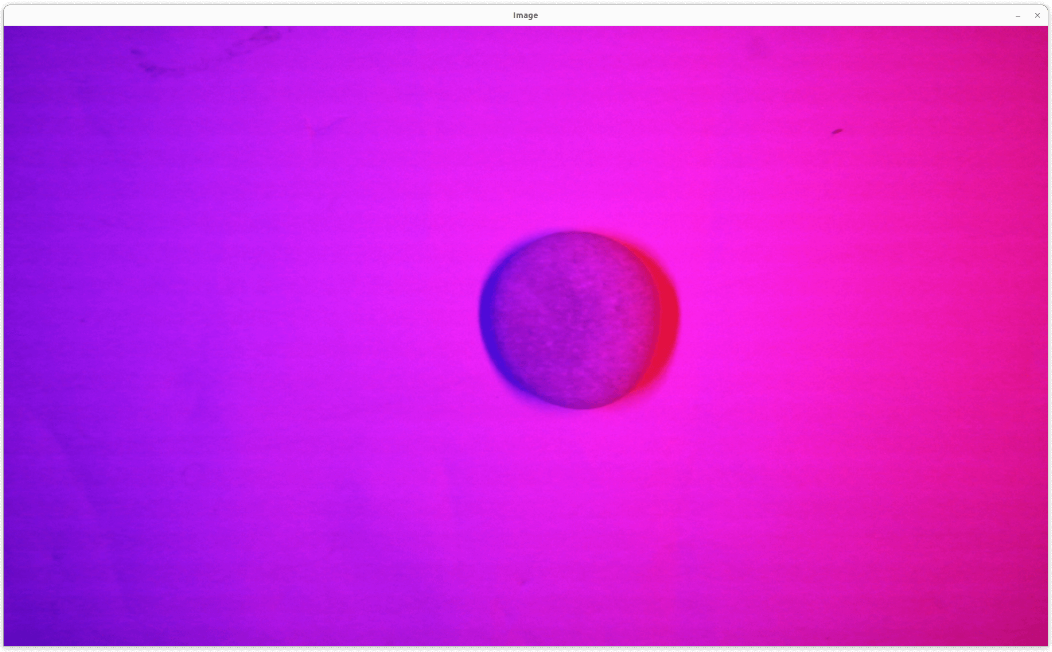


FIGURE 5.16: Input Image of Rock

Last image is showing the result from the height estimation of the system. The system is estimating the rock as approximately 2 - 3cm which is really close.

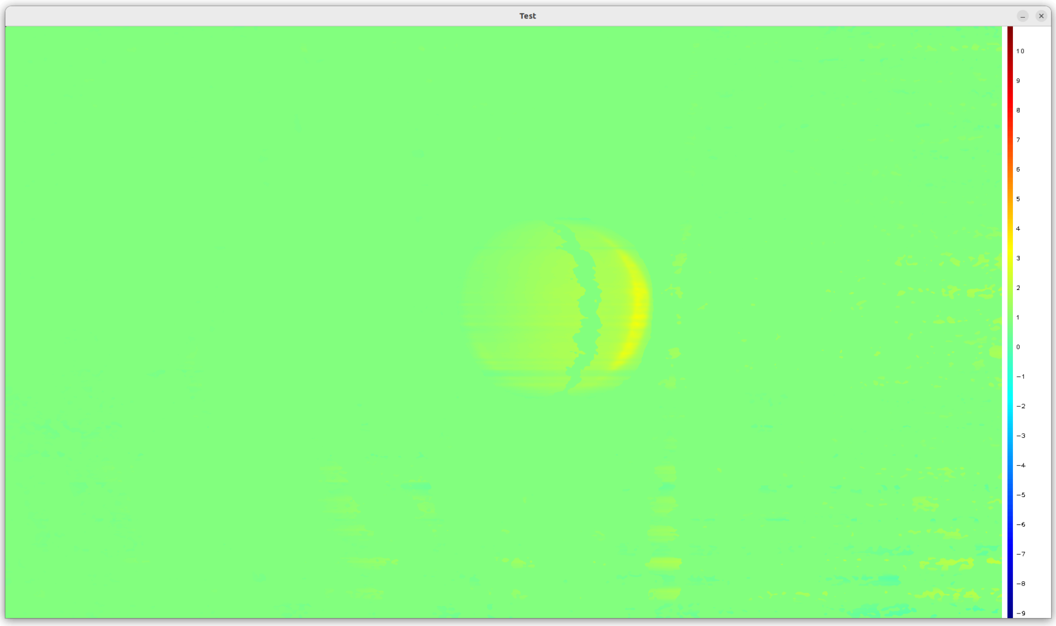


FIGURE 5.17: Output Image of Rock

5.3.3 Challenging conditions for the system operation

There are condition where the correct operation of the system was found to be hindered by mainly environmental factors like reflections created for surfaces or water.

More specifically as, discussed previously, when the the scanned surface is anisotropic meaning that, the direction of the reflections is not a linear function of the angle of incidence, the behavior of the system can be unpredictable. The main operation of the depth estimation algorithm is based on the isotropic properties of the scenes which is scanned in real time so when the surface is wet or has reflective materials the resulted outcomes are most of the time false as the system is not designed to understand the different angles the light is refracted from the anistropic surfaces. An example of these surfaces is show below on figure 5.18

FIGURE 5.18: Isotropic vs Anisotropic reflections

Another challenging condition is when the potholes being measured are being filled with water. In this condition the water is reflecting the light for the LED light sources creating unexpected results and does not allow the system to collect the correct data of the bottom of the pothole in order to measure it's distance.

Last but not least the color of the scanned surface is a matter of great essence. Surfaces containing red or blue color may trick the system into trying to measure slopes that may not be there. This can create many artifacts to the final result and does not product the correct results.

5.3.4 Final Assessment of the Verification

From all the above processing and the experiment of a random tested frame the system passed the verification process and ensured that it's proof of concept works as it is intended to. Furthermore it can be a good low budget alternative for depth estimation applications. The operation of the system is great under certain conditions. The scanned surface should be dry without reflective surfaces on it and preferably monochromatic in order to achieve the best results. Objects on the scanned scene should not contain red or blue colors in order to avoid the obstruction of the correct slope detection of the objects.

5.4 Performance Evaluation

Performance is an important part of a real time system. The performance of the system of this thesis can be easily evaluated because it can be broken down to two main components. Speed of produced results and accuracy of the results produced.

5.4.1 System throughput

The calculation speed of this system can be measured in many different ways and can be dependent on different factors. The two main metrics are the maximum input frames per second and the maximum output frames per second. Both of these factor contribute in order to calculate the system throughput.

The maximum input frames per second are mainly bottlenecked from the camera's ability to capture frames. The limit for the selected camera is 30 fps (frames per second). In order to raise this limit a new camera would need to replace the currently selected.

On the other hand the maximum output frames per second is metric that depends on the processing power of the CPU used and the quality of the underlined code. A series of testing has taken place in order to find the maximum output frames for the current configuration. The same dataset with different resolutions was given to the system resulting to different Frames per Second averages.

As shown on the table 5.1 the higher the resolution the slower the system becomes.

Resolution	Frames Per Second (FPS)(Average)
800X600 (SVGA)	11.87
1360X768 (HD)	6.24
1920X1080 (FHD)	3.64
1920X1200 (WUXGA)	1.76

TABLE 5.1: Output FPS relative to resolution

From the above table it is easy to deduct the conclusion that when a meter wide area is scanned with the system on a moving platform, the maximum scanning speed is as follows on table 5.2 . The calculations on this table consider that the maximum speed possible for the moving platform is the speed that each frame is different from the previous by half a frame.

Resolution	Maximum moving speed (Km/h)
800X600 (SVGA)	21.36
1360X768 (HD)	11.23
1920X1080 (FHD)	6.55
1920X1200 (WUXGA)	3.16

TABLE 5.2: Maximum moving speed for 2 frames per meter

5.4.2 Calculation Accuracy

A series of testing has taken place in order to decide the maximum calculation accuracy of the system for different resolutions of the current configuration. The same dataset with different resolutions was given to the system. Next the same object was measured again in order to calculate the mean error of 3 different tests for each resolution. This test cannot have the best results because the results can be different and they depend extremely from the conditions in the time of testing and from the calibration process beforehand. The highest possible effort was made in order to keep the testing conditions the same and give the system the best fighting chance as possible.

Resolution	Mean error cm (from real)
800X600 (SVGA)	1.63 cm
1360X768 (HD)	1.48 cm
1920X1080 (FHD)	0.82 cm
1920X1200 (WUXGA)	0.81 cm

TABLE 5.3: Mean error in cm

As shown from the above table 5.3 the accuracy of the system is dependent from the resolution and especially from the width of the frame, as the depth calculation is done on the axis of the width, meaning from left to right and from right to left. This observation can also verify the results of the 1920X1200 and 1920X1080 resolutions which show the same mean error even though the height resolution is different.

Chapter 6

Conclusions and Future Work

This chapter sums up the thesis and suggests some ways it could be expanded in a way that can benefit both the scientific research domain and the industry standard depth estimating systems.

6.1 Conclusions

In conclusion, there is a big need for a low budget simple, yet efficient system that uses just one camera to estimate depth for surface object detection and road scanning for security and protection purposes. The system is very promising for monochromatic surfaces with isotropic reflecting materials. The single camera characteristic is very helpful when used on a moving platform because it eliminates the problem created from the two cameras having their distance change which is a basic condition for all the stereoscopic systems.

The currently created system was tested meticulously and proved that is a viable alternative to traditional depth estimation system with lower cost. The system is capable of achieving great accuracy when properly calibrated and when the surface conditions are in favor of it's required standards. Wet or reflective materials may create unpredictable results and therefore it is not recommended for use on these materials. The throughput performance with the simple microcomputer used is in an acceptable range for a slow moving platform, that can enable the system to work in a real time environment permitting it's user to understand the depth of the scanned scene in real time. Furthermore, the system is easily scalable and the end user is able to make the necessary changes in order to upgrade the processing part until it can reach the required throughput for higher requirement applications.

6.2 Future Work

In the future the system should be reconfigured with more computing power which can enable the testing of more frames each second resulting in greater user experience. Some possible solution may be the use of a small form factor multicore computer with a slight larger cost. Also the use of even brighter light source would allow the transfer of the camera to a bigger distance from the surface for a bigger scanned scene with each frame, which would allow the use of the system in a bigger scale. Moreover, it would be very useful for an automatic calibration process to be developed that read the histogram of the produced output and automatically calibrated the light sources in order to make sure the quality of the final result is always optimal. One more upgrade for the system could be a better camera if the required depth estimation accuracy is smaller than 1cm. Last but not least, it is possible to create applications with this system as the beginning platform, with a better user interface and the addition of useful information for the end user like software object detection from the post-processing of the produced frames in order to give the user plenty of information to make decisions.

All things considered, this is a very hopeful endeavour in order to create a system that is solving a very basic but meaningful problem for computer vision, with many future upgrades and applications possible.

References

[1] G. Rematska. "Depth perception from a single camera and multiple light sources". In: *Master Thesis, School of ECE, Technical University of Crete* (July 2015). URL: http://purl.tuc.gr/dl/dias/A0BABCD5-2160-4799-85D7-3EC099D249BA.

[2] J. A. Stankovic. "Real-Time and Embedded Systems". In: *ACM Computing Surveys* 28.1 (Mar. 1996). URL: https://dl.acm.org/doi/pdf/10.1145/234313.234400.

[4] R.lavanya R. S. Peter R. R. Rajeshwari. "Embedded Systems". In: *International Journal Of Engineering Research and Technology (IJERT) IFET-2014 Conference Proceedings* (Jan. 2014), pp. 66–67. URL: https://www.ijert.org/research/embedded-systems-IJERTCONV2IS01010.pdf.

[5] A.Anderson S.Souza R.Maia L.M.G.Gonçalves. "3D Probabilistic Occupancy Grid to Robotic Mapping with Stereo Vision". In: *Current Advancements in Stereo Vision* (July 2012). URL: https://www.intechopen.com/chapters/37778.

[6] F.Blais. "Review of 20 Years of Range Sensor Development". In: *Journal of Electronic Imaging* (Jan. 2004), pp. 231–240. URL: https://www.cs.columbia.edu/~allen/PHOTOPAPERS/Blais_JEI04_Review20YearsRangeSensor.pdf.

[7] P.J.Besl. "Active, optical range imaging sensors". In: *Machine Vision and Applications* (June 1988), pp. 127–152. URL: https://doi.org/10.1007/BF01212277.

[9] F.Duchoň M.Dekan L.Jurišica A.Vitko. "Some Applications of Laser Rangefinder in Mobile Robotics". In: *Journal of Control Engineering and applied informatics* 14 (June 2012), pp. 50–57. URL: https://scholar.google.com.vn/citations?view_op=view_citation&hl=en&user=ltCOPzkAAAAJ&citation_for_view=ltCOPzkAAAAJ:UebtZRa9Y70C.

[11] Q.Li M.Yao X.Yao B.Xu. "A real-time 3D scanning system for pavement distortion inspection". In: *Measurement Science and Technology* 21 (Sept. 2009). URL: https://iopscience.iop.org/article/10.1088/0957-0233/21/1/015702/pdf.

[12] S.Pu M.Rutzinger G.Vosselman S.O.Elberink. "Recognizing basic structures from mobile laser scanning data for road inventory studies". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 21 (Dec. 2011), p. 11. URL: https://doi.org/10.1016/j.isprsjprs.2011.08.006.

[13] A.Abdelhafiz. "Integrating Digital Photogrammetry and Terrestrial Laser Scanning". In: *Assuit University PhD Thesis* (Feb. 2009). URL: https://www.researchgate.net/publication/278157315_Integrating_Digital_Photogrammetry_and_Terrestrial_Laser_Scanning.

[14] C.Godard O.M.Aodha G.J.Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency". In: *University College London* (Jan. 2017). URL: https://arxiv.org/pdf/1609.03677v3.pdf.

[15] C.Godard O.M.Aodha G.J.Brostow. "Digging Into Self-Supervised Monocular Depth Estimation". In: *UCL, Caltech, Niantic* (June 2018). URL: https://arxiv.org/pdf/1806.01260v4.pdf.

[16] S. Zhang. "Recent progresses on real-time 3D shape measurement using digital fringe projection techniques". In: *Optics and Lasers in Engineering* 1 (Feb. 2010). URL: https://www.sciencedirect.com/science/article/pii/S0143816609000529?via%3Dihub.

[17] P.Kühmstedt I.Schmidt M.Heinze G.Notni C.Bräuer-Burchardt A.Breitbarth. "Fringe projection based high-speed 3D sensor for real-time measurements". In: *Optical Measurement Systems for Industrial Inspection VII* 8082 (May 2011). URL: https://doi.org/10.1117/12.889459.

[18] B.Li Y.Wang S.Zhang P.Ou. "Flexible real-time natural 2D color and 3D shape measurement". In: *Optics Express* 21 (2013), pp. 19743–19754. URL: https://doi.org/10.1364/OE.21.016736.

[19] Y.Gong S.Zhang. "Ultrafast 3-D shape measurement with an off-the-shelf DLP projector". In: *Optics Express* 18 (May 2010), pp. 19743–19754. URL: https://doi.org/10.1364/OE.18.019743.

[20] M.Takeda K.Mutoh. "Fourier transform profilometry for the automatic measurement of 3-D object shapes". In: *Applied Optics* 22 (May 1983), pp. 3977–3982. URL: https://doi.org/10.1364/AO.22.003977.

[21] X.Su W.Chen. "Fourier transform profilometry:: a review". In: *Optics and Lasers in Engineering* 35 (May 2001), pp. 263–284. URL: https://doi.org/10.1016/S0143-8166(01)00023-9.

[22] J.Deglint F.Kazemzadeh D.S.Cho D.A.Clausi A. Wong. "Numerical Demultiplexing of Color Image Sensor Measurements via Non-linear Random Forest Modeling". In: *Vision and Image Processing Research Group, University of Waterloo, Waterloo, Ontario, Canada* (Dec. 2015). URL: https:

//www.researchgate.net/publication/287249644_Numerical_Demultiplexing_of_Color_Image_Sensor_Measurements_via_Non-linear_Random_Forest_Modeling.

[24] R.Bandara. “A Music Keyboard with Gesture Controlled Effects Based on Computer Vision”. In: *Thesis for BSc. University of Sri Jayewardenepura* (Jan. 2011). URL: https://www.researchgate.net/publication/216533561_A_Music_Keyboard_with_Gesture_Controlled_Effects_Based_on_Computer_Vision.

External Links

[3] “Wikipedia Embedded System”. In: (). URL: https://en.wikipedia.org/wiki/Embedded_system.

[8] “Wikipedia 3D scanning”. In: (). URL: https://en.wikipedia.org/wiki/3D_scanning.

[10] “Wikipedia Time of flight”. In: (). URL: https://en.wikipedia.org/wiki/Time_of_flight.

[23] “Spectral Sensitivity Curves of Cameras”. In: (). URL: https://www.techbriefs.com/component/content/article/36142-spectral-sensitivity-curves-of-cameras.

[25] “Gphoto Official Website”. In: (). URL: http://www.gphoto.org.

[26] “FFMPEG Official Website”. In: (). URL: https://ffmpeg.org.

[27] “Gphoto Official Website”. In: (). URL: https://github.com/umlaeute/v4l2loopback.

[28] “Ubuntu 22.04.3 for Raspberry pi”. In: (). URL: https://ubuntu.com/download/raspberry-pi/thank-you?version=22.04.3&architecture=desktop-arm64+raspi.

[29] “Local Raspberry Pi connection over Ethernet”. In: (). URL: https://blackdevice.com/how-to-connect-to-raspberry-pi/.