Coursework Declaration and Feedback Form

The Student should complete and sign this part

Student Number: 2943675S	Student Name: ANUGRAH SEBASTIAN
Programme of Study (e.g. MSc in Electronics and Electrical Engineering): MSc in Computer Systems Engineering	
Course Code: ENG5059P	Course Name: MSc Project
Name of **First** Supervisor: Dr. Gilles Bailet	Name of **Second** Supervisor: Dr. Massimo Vassalli
Title of Project: Revolutionizing Space Exploration: 3D Scanning for Defect- Free 3D Printed Components	
Declaration of Originality and Submission Information	
I affirm that this submission is all my own work in accordance with the University of Glasgow Regulations and the School of Engineering requirements Signed (Student) : Anugrah	ENG5059P
Date of Submission : 09/08/2024	

Feedback from Lecturer to Student – to be completed by Lecturer or Demonstrator	
Grade Awarded: Feedback (as appropriate to the coursework which was assessed):	
Lecturer/Demonstrator:	Date returned to the Teaching Office:

Revolutionizing Space Exploration: 3D Scanning for Defect- Free 3D Printed Components

By: ANUGRAH SEBASTIAN

GUID: 2943675S

SUPERVISOR:

Dr. Gilles Bailet

Dr. Massimo Vassalli

August 2024

A thesis submitted in partial fulfilment of the requirements for the degree of
MASTER OF SCIENCE IN COMPUTER SYSTEMS ENGINEERING

Abstract

Integrating 3D printing technology in space missions represents a transformative advancement, enabling in-site manufacturing and repair capabilities. However, the reliability of 3D-printed components is challenged by the occurrence of errors during the printing process. This study addresses the critical need to mitigate such errors to ensure the structural integrity and functionality of printed parts. In this study we progressed towards developing a 3D scanning system capable of working with 3D printers the errors are within acceptable ranges / preserving its usability. We experimented with different data acquisition methods such as video capture and snapshot at measured steps for Laser Triangulation alongside developing algorithms to preserve printed material's usability. Our results show promising potential in integrating the scanning procedures within the printing process extending to industrial manufacturing for real-time production monitoring and quality control.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisors, Dr. Gilles Bailet, for their unwavering support, guidance, and encouragement throughout my MSc research. His insightful criticism, thorough critique, and constructive suggestions were critical to the production of this study. I am also grateful to Mr. Satyam Bhatti for his ongoing help and guidance in strengthening our scientific approach to real-world situations. In addition, I appreciate Dr. Massimo Vassalli's work in assessing this paper.

I am grateful to Muhammed Sadik, my research partner, for participating and contributing to this study. I would also like to thank my classmate, Sohail Abdul Jaffer, for his lively talks, meaningful comments, and encouragement. I am grateful to the academic and administrative staff who helped me with different administrative and logistical issues during my MSc. Their help has been highly appreciated.

Finally, I would like to express my gratitude to my family and friends for their everlasting support, encouragement, and love. Their unwavering support in me has been a source of strength and inspiration during my MSc studies.

Contents

Abstract 2

Acknowledgements 3

1. Introduction 6

  1.1 Background 6

  1.2 3D Digitisation 7

  1.3 Aims and Objectives 8

  1.4 Outline 8

2. Literature Review 10

  2.1 Introduction 10

  2.2 3D Scanning 10

3. Methodology 13

  3.1 Computer vision 13

    3.1.1 Passive stereo vision 13

    3.1.2. Active stereo vision 13

  3.2 Scanning height in laser triangulation 14

  3.3 Laser triangulation geometry for height 15

  3.4 Pre-processing 16

    3.4.1 RGB to Greyscale Conversions 16

    3.4.2 Conversion Process 17

  3.5 Camera calibration 18

    3.5.1 Introduction to Camera Calibration 18

  3.6 Edge detection 20

    3.6.1 Laplacian gradient 21

    3.6.2 Sobel gradient 21

  3.7 Interpolation 22

    3.7.1 Linear Interpolation 22

    3.7.2 Polynomial Interpolation 22

    3.7.3 Cubic Spline Interpolation 23

    3.7.4 Nearest Neighbour Interpolation 24

4. Experiment 26

  4.1 Hardware setup 26

  4.2 Software Setup 27

    4.2.1 Laser Line Tracing 27

    - 4.2.3 Data acquisition 31
  - 4.3 Camera Calibration 35
- 5. Results 36
  - 5.1 Data Acquisition: Video capture 37
  - 5.2 Data Acquisition: Snapshots at measured steps 38
- 6. Discussion 46
  - 6.1 Accuracy and Error Analysis 46
  - 6.2 Methodology Performance 46
  - 6.3 Calibration and Pixel Displacement 46
  - 6.4 Integration and Reconstruction 47
  - 6.5 Potential Improvements 47
  - 6.6 Conclusion 47
- 7. Future scope 48
- References 49
- Appendix 51
  - 1. GANTT Chart 51
  - 2. Project code 51
    - 2.1. Imports 51
    - 2.2. Experiment Logic Functions 51
    - 2.3. Utility Functions 55
    - 2.4. Experiment Code 58

1. Introduction

1.1 Background

The establishment of colonies on the Moon and Mars has been hindered by the reliance on substantial resources for transportation and the extreme living conditions, which necessitate suitable habitats for human survival [1]. Throughout the history of space exploration, there have been constant limitations on what can be constructed for space missions. Every spacecraft and piece of equipment sent into orbit has faced restrictions on size, weight, and structural durability due to the demands of launch. Furthermore, long-duration missions necessitate either stockpiling replacement parts or sending them from Earth, resulting in significant delays. However, if equipment and spacecraft could be manufactured in space's zero-gravity environment, these limitations would be erased. This capability would allow for the fabrication of structures of practically infinite size and complexity, which are not feasible to launch from Earth and, in certain situations, impossible to fabricate on Earth [2]. Long-term missions could decrease their reliance on spare parts and reduce dependency on Earth resupply. Furthermore, space exploration fraught with constant accident which require equipment part to be sent from Earth to the designated spacecraft. There has been a lot of speculation and debate about all the possible accidents, additionally developing solutions which minimise these accidents to the extent that it does not significantly hinder space exploration. Rather than attempting to predict and prepare for every potential machine failure, accident, or other challenge that may occur during space missions, it appears more rational to utilize the versatility of 3-D printing for "in-space manufacturing" (ISM) [3].

In recent years Additive manufacturing has emerged as the creative solution to traditional manufacturing processes which struggled with production time, production cost, precision manufacturing, manufacturing speed and difficulty of customisability. Seen to be the next step in the evolution of manufacturing, Additive manufacturing is not without its flaws and challenges. The typical process of Additive manufacturing, commonly known as 3D printing, starts with a Stereolithography (STL) file which tries to capture the boundary information of a given model. The STL data is subsequently utilized by Computer-aided design (CAD) software to generate a surface representation of a given model, employing planar triangles through the tessellation process. Although several alternatives to the STL format have been presented over the years, most computers continue to use STL. The CAD design is then materialised into its physical form through an additive manufacturing process, wherein materials are sequentially deposited layer-by-layer into a three-dimensional space until the object is fully constructed. This transformative method begins with an initially non-existent entity and progressively builds the final structure [4].

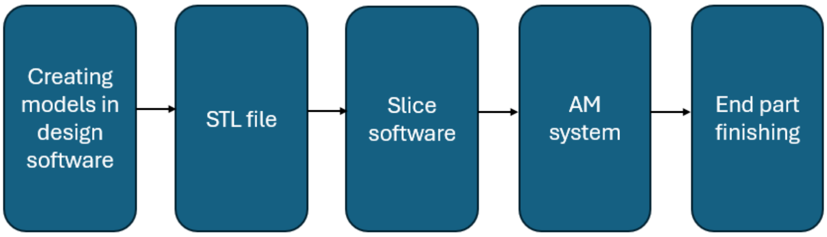


Figure 1: Additive manufacturing process

The conversion of a 3D model into a physical object presents its own set of challenges, often resulting in printed models that significantly deviate from the anticipated design. Fortunately, these design challenges can be addressed by focusing on any of the following avenues:

- Model Shape optimization
- Design process optimization
- Pre & Post process practices
- Printing methodologies
- Error control
- multi-material printing

Errors occur in all industrial processes, including 3D printing. Due to a lack of quality control methods, the machines that are now on the market might not always be the most dependable. Three types of faults can be found in 3D printing: material errors, process errors (errors during printing) and data preparation problems (error before printing).

With its unmatched creative freedom and capacity for customized production, 3D printing is leading the way in a manufacturing revolution. However, the technology's development is hindered by material limitations, speed and scale issues, cost, technical expertise requirements, intellectual property concerns, and quality control challenges. Addressing these challenges through research, innovation, and collaboration is crucial for the continued advancement and widespread adoption of 3D printing technology. As these hurdles are overcome, 3D printing will undoubtedly play an increasingly integral role in shaping the future of manufacturing and beyond.

1.2 3D Digitisation

3D digitisation is a transformative technology that captures the three-dimensional attributes of objects and environments, converting them into digital formats. This process involves creating digital replicas that can be manipulated, analysed, and reproduced, offering immense possibilities across various industries. From preserving historical artefacts to enhancing manufacturing processes, 3D digitisation is reshaping our interaction with the physical world. Central to this technology are several sophisticated 3D scanning techniques, each with its unique methodologies and applications.

The Importance of 3D Digitisation

3D digitisation has revolutionised numerous fields by providing precise digital models of physical objects. In cultural heritage, it allows for the preservation and restoration of artefacts and monuments, ensuring that history is conserved digitally for future generations. In healthcare, 3D digitisation facilitates the creation of custom prosthetics and accurate surgical models, enhancing patient outcomes. Technology also plays a crucial role in manufacturing and engineering, enabling quality control, reverse engineering, and rapid prototyping.

Furthermore, 3D digitisation is pivotal in the entertainment and gaming industries, where it brings characters and environments to life with unprecedented realism. The ability to digitise real-world objects and integrate them into digital spaces has also advanced augmented reality (AR) and virtual reality (VR) applications, creating immersive and interactive experiences.

1.3 Aims and Objectives

The main objective of this project is to create, develop, and evaluate a 3D scanning hardware system for monitoring a 3D printer. This system will track every printed layer throughout the printing process, employing laser scanning and computer vision methods to reconstruct the geometry of the printed object using laser triangulation techniques.

Keys Objectives:

- Replicate the hardware setup for the laser triangulation with reverse geometry
- Design a data acquisition methodology which gives the best results for laser triangulation with reverse geometry
- Design algorithms which give high accuracy for print object dimensions
- Make the entire system replicable for future work

1.4 Outline

The remainder of this report is structured as follows:

i. **Chapter 2** reviews existing research on 3D scanning technologies and their integration with additive manufacturing. It highlights key advancements, discusses various scanning techniques, and addresses the challenges faced in the field. The review includes an analysis of different methodologies and technologies, emphasizing their contributions and limitations.
ii. **Chapter 3** details the research methodology, outlining the techniques and procedures used to investigate the study's objectives. It covers the application of computer vision, including passive and active stereo vision, and discusses methods for object height calculation using laser triangulation with reverse geometry. The chapter also describes pre-processing steps, camera calibration, edge detection methods, and interpolation techniques, providing a thorough explanation of the methods used in the research.

iii. **Chapter 4** details the experimental setup and procedures used to test the methodologies outlined in the previous chapters. It includes descriptions of hardware and software configurations, data acquisition processes, and camera calibration procedures.

iv. **Chapter 5** presents the results of the experiments, including data acquisition and analysis. It discusses the findings and their implications for the study.

v. **Chapter 6** analyses the results, discussing their implications for the research. It includes an evaluation of accuracy, error analysis, and methodology performance, as well as considerations for calibration, pixel displacement, integration, reconstruction, and potential improvements.

vi. **Chapter 7** outlines potential areas for future research, including new technologies, methodologies, and applications for 3D scanning and additive manufacturing. It discusses how the study's findings could be expanded upon and applied in different contexts.

2. Literature Review

2.1 Introduction

The scholarly community has recently given additive manufacturing (AM), commonly referred to as 3D printing, a great deal of attention and examination. This chapter reviews previously published papers on AM, focusing on the use of scanning techniques to evaluate their limitations and contributions to the literature.

The journal article by (Wong, 2012) [5], offers a comprehensive overview of the field, discussing various AM technologies, including their principles, applications, and advancements. The review highlights the potential of AM to produce complex geometries directly from digital models and addresses the implications of AM on manufacturing efficiency, customization, and material usage, noting both advantages and challenges associated with these technologies.

The paper (Oropallo, 2016) [6] offers a comprehensive overview of the critical challenges facing 3D printing technology, highlighting ten key issues that need to be addressed for its advancement. These challenges encompass technical, operational, and strategic aspects, including material limitations that restrict application range and performance, printer capabilities that affect precision and complexity, and the need for extensive post-processing to achieve desired results. Other challenges include inadequate software and design tools, lack of standardization leading to variability in quality, high economic costs compared to traditional manufacturing, intellectual property protection issues, unclear environmental impacts, insufficient education and training, and evolving regulatory frameworks with safety and quality uncertainties. The authors stress the importance of interdisciplinary collaboration and innovation in materials science, engineering, software development, and policy-making to overcome these obstacles.

Monitoring the additive manufacturing process could help avoid errors during production. A system could be developed to perform real-time scanning of the printed object, creating a digital twin that can be compared to the STL or CAD input to detect deviations.

2.2 3D Scanning

To explore the potential scanning systems, further literature was reviewed. In the paper (Besl, 1988) [7] investigated the design and use of active optical range imaging sensors for machine vision. The paper provided a comprehensive overview of these sensors, designed to capture three-dimensional information about objects and environments by measuring the distance to surfaces using optical methods. The authors also introduced the concept of range imaging sensors and discussed various active optical range imaging techniques, including triangulation, time-of-flight, and interferometry, detailing their operational mechanics, advantages, and limitations.

(Breier, 2015) [8] presented a novel method for accurately assessing the height profile of printed circuit boards (PCBs) using laser triangulation. The study focuses on improving the precision of height measurements by employing a perpendicular camera setup, significantly enhancing measurement accuracy compared to traditional angled setups. The research highlighted practical applications in the quality control and inspection of PCBs in industrial settings, demonstrating the importance of accurate height profiling in detecting manufacturing defects and ensuring reliability.

The (Wang, 2021) [9] study explored the integration of a linear laser scanner with a camera for effective 3D reconstruction. The proposed system uses laser triangulation technology combined with image processing to capture precise 3D data. Experiments demonstrated the system's ability to achieve high-resolution 3D reconstructions with significant accuracy, highlighting improvements in detail capture and measurement precision over traditional techniques.

In (Yao, 2023) [10] proposed an improved grayscale centre of gravity technique for laser centre line extraction used in the three-dimensional reconstruction of faults in battery films. The method enhanced the precision of laser centre line extraction, crucial for accurate 3D reconstruction. The study concluded that this method offers a robust and precise tool for laser centre line extraction, with potential applications in other manufacturing and quality assurance areas.

The paper (Liu, 2023) [11] provided a comprehensive study on advancements in 3D reconstruction using laser measurement techniques. The research focused on enhancing the precision and efficiency of 3D reconstruction, exploring technologies such as laser triangulation and algorithms like Structure from Motion (SFM). The paper highlights the development of new algorithms to reduce noise and improve resolution, addressing common challenges in laser-based 3D reconstruction, such as handling complex geometries and varying surface textures.

With such evidence, laser-based triangulation systems can work for the scanning of a wide range of objects. There was a need to clearly define the required mathematical calculations that work well for the reverse geometry laser triangulation system that needed to be developed. Thanks to the paper (Muñoz-Rodríguez, 2003) [12] demonstrated the required mathematics of calculating heights from laser line displacement in the captured images. The authors explored the precision and effectiveness of utilizing light line displacement for object shape detection in their study published in the Journal of Modern Optics. The research investigated the methodology for detecting and evaluating the displacement of light lines projected onto objects, which is crucial for accurately determining the object's shape. The study employed a systematic approach to evaluate the displacement of light lines. The authors described the experimental setup, including the laser source, projection system, and detection equipment. They outlined the mathematical models used to analyse the displacement data and correlate it with the object's shape. The methodology focused on minimizing errors and improving the accuracy of shape detection. The results section presented the findings from the experimental evaluations. The authors demonstrated that their approach effectively captured the shape of various objects with high accuracy. They provided detailed data on the displacement measurements and discussed the implications of their findings for industrial applications. They highlighted the advantages of using light line displacement, such as higher precision and reduced computational complexity.

3. Methodology

3.1 Computer vision

Computer vision is a branch of artificial intelligence (AI) and computer science that aims to give computers the same level of comprehension and interpretation of visual data as people. This requires the automatic extraction, analysis, and comprehension of information from digital images or videos. Computer vision is a rapidly advancing field with significant impacts across various industries. By mimicking human visual perception, it enables machines to understand and interact with the visual world, leading to innovations that enhance efficiency, safety, and user experiences.

3.1.1 Passive stereo vision

Passive stereo vision is a technique in computer vision used to extract three-dimensional information from a scene using two or more cameras. The same picture is photographed from slightly different angles by two cameras that are usually at a set distance apart (baseline). These cameras replicate how depth is perceived by human eyes. Passive stereo vision is a powerful technique for deriving depth information from visual data, leveraging the natural light and the geometric relationship between multiple camera views. Its applications span across various fields, enhancing machine perception and enabling advanced functionalities in robotics, autonomous systems, and 3D modelling.

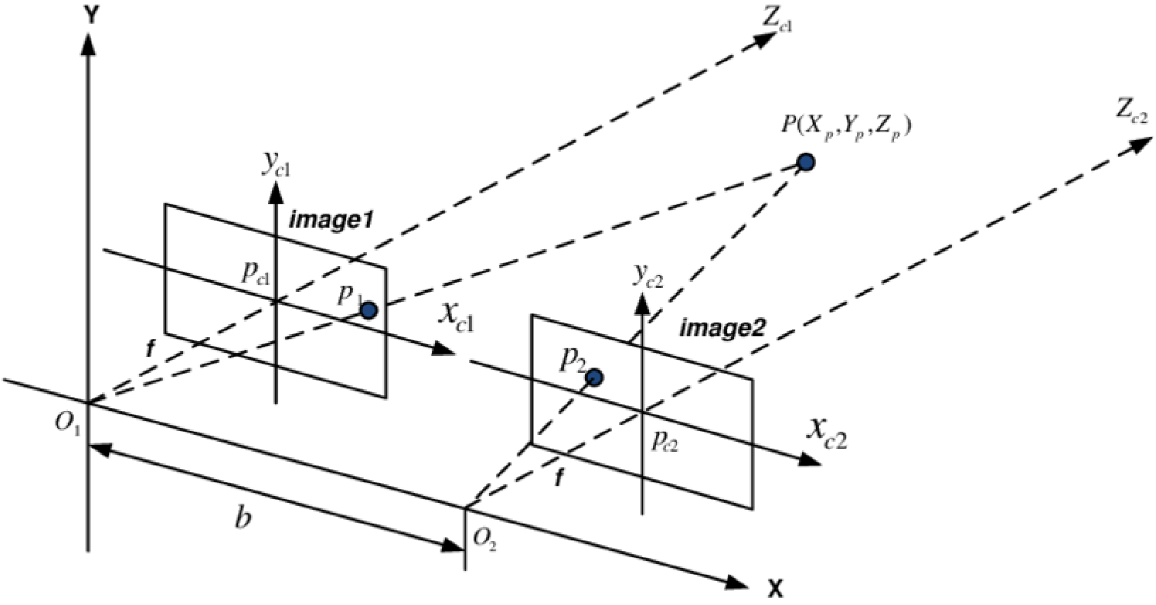


Figure 2: Configuration scheme of passive stereo vision

3.1.2. Active stereo vision

Active stereo vision is a technique in computer vision that enhances the process of depth perception by projecting a known pattern of light onto the scene. This method overcomes some of the limitations of passive stereo vision, particularly in environments with low texture or poor lighting conditions. A known light pattern (e.g., dots, stripes, or grids) is projected onto the scene using a projector or structured light source. Active stereo vision enhances depth perception by projecting a known light pattern onto the scene, making it easier to identify corresponding points between camera images. This technique provides more accurate and reliable depth information, especially in challenging environments with low texture or poor lighting and finds applications in various fields such as industrial inspection, 3D scanning, robotics, and medical imaging.

3.2 Scanning height in laser triangulation

In laser triangulation, a laser emits a beam of light that strikes the surface at a point. The light from this point is reflected back and captured by a camera, which is positioned at a known distance from the laser and at a known angle relative to the laser beam. The basic setup can be visualized as a right-angled triangle.

- d is the horizontal distance between the laser and the camera.
- θ is the angle between the laser beam and the line parallel to the surface (also known as the triangulation angle).
- D is the height or distance from the surface to the laser/camera setup.

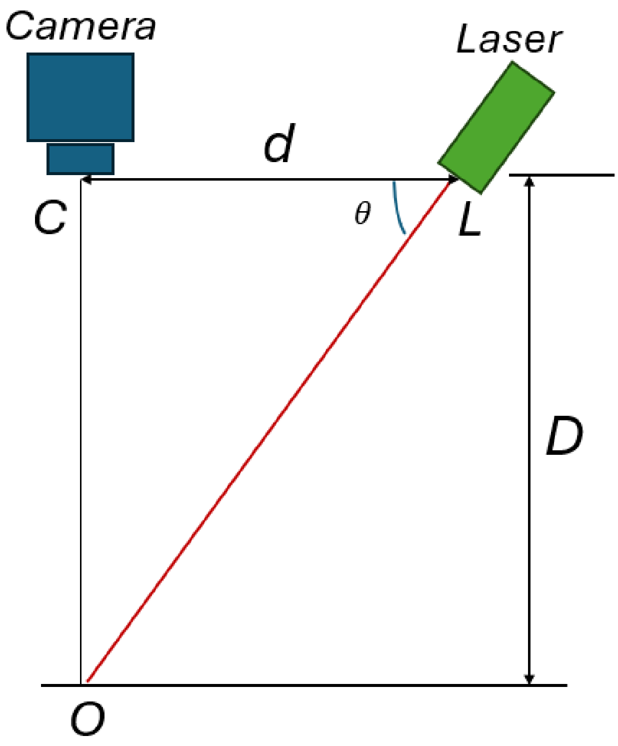


Figure 3: Scanning height calculation in laser triangulation scanning

Consider the right triangle formed by the surface point (O), the point where the laser hits the surface (L), and the position of the camera (C):

Using basic trigonometric relationships in a right-angled triangle:

$$D = d \cdot \tan\theta \qquad (1)$$

3.3 Laser triangulation geometry for height

Laser triangulation is an implementation of active stereo vision, where a laser beam is projected onto an object to be scanned. A camera is used to capture the displacement of the laser line caused by the object from a different point of view. Displacement can be then used to calculate the geometry of the scanned object such as height. A collection of multiple scans can then be aggregated to reconstruct a point cloud of the scanned object.

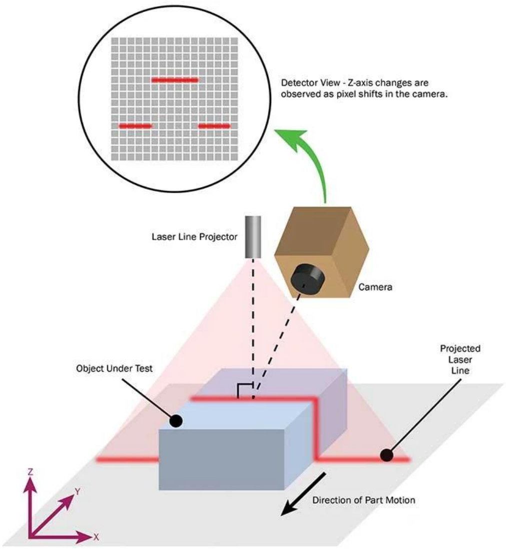


Figure 4: Laser line displacement in laser triangulation. Source [13]

Figure 5 shows the transverse section of the set-up and describes the geometric relationship between a line displacement $(s(x, y)$ and the object height data. D is the distance between the CCD camera and the reference plane. The laser diode and the CCD camera are separated by d. The x- and y-axes are situated on the reference plane. $h(x, y)$ indicates the height data between the object's surface and the reference plane. Additionally, $s(x, y)$ represents pixel displacement. The light line on the reference plane moves from A to B when a light line is projected onto the target. This geometry was shown by Muñoz-Rodrígue.

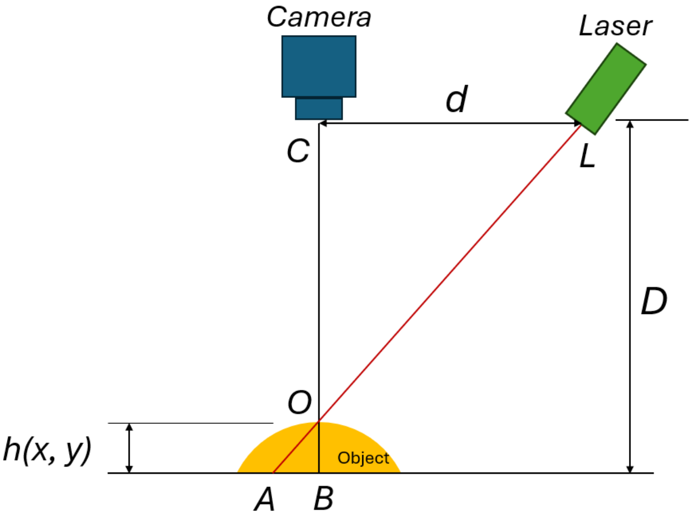


Figure 5: Optical geometry of the laser triangulation with reverse geometry during the scanning process

$$s(x,y) = \overline{AB} = x_A - x_B \tag{2}$$

If the k pixel/mm is known, then $s(x,y)$ can be written

$$s(x,y) = \Delta pk \tag{3}$$

where

$$\Delta p = p_A - p_B \tag{4}$$

Thus, height can be given by

$$h(x,y) = \frac{D \cdot s(x,y)}{d + s(x,y)} \tag{5}$$

3.4 Pre-processing

3.4.1 RGB to Greyscale Conversions

The conversion of a colour image from RGB (Red, Green, Blue) to greyscale is a common image processing task. The three-channel RGB colour image is converted throughout this procedure into a single-channel greyscale image, where each pixel denotes an intensity value. Here's the theory behind the RGB to greyscale conversion:

RGB Colour Model

In the RGB colour model, each pixel in an image is represented by three components:

- R (Red): Intensity of the colour red.
- G (Green): Intensity of the colour green.
- B (Blue): Intensity of the colour blue.

Each of these components typically ranges from 0 to 255 in 8-bit images.

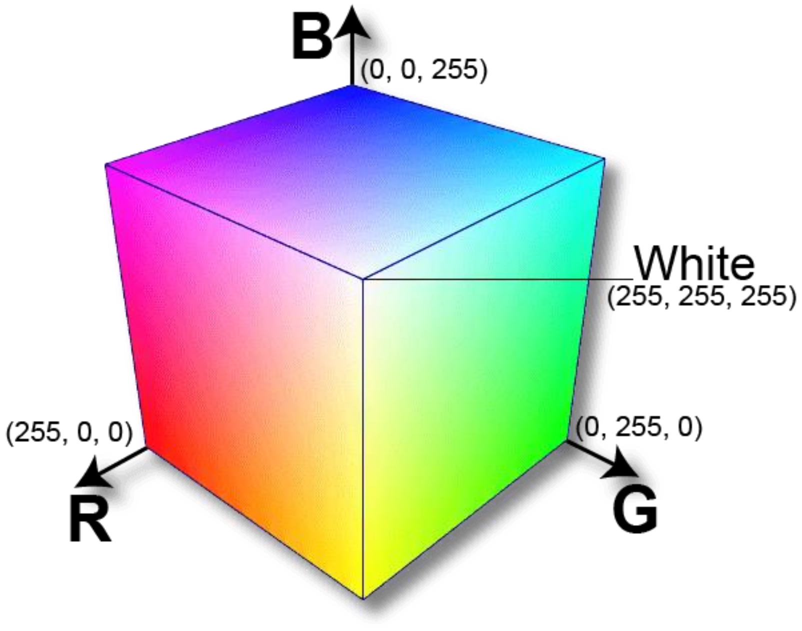


Figure 6: RGB coordinate system with all the different colours that the model can describe [14]

Greyscale Image

A greyscale image, on the other hand, contains only shades of grey. In an 8-bit image, 255 represents white, and 0 represents black in a greyscale image. Each pixel in an image represents an intensity value.

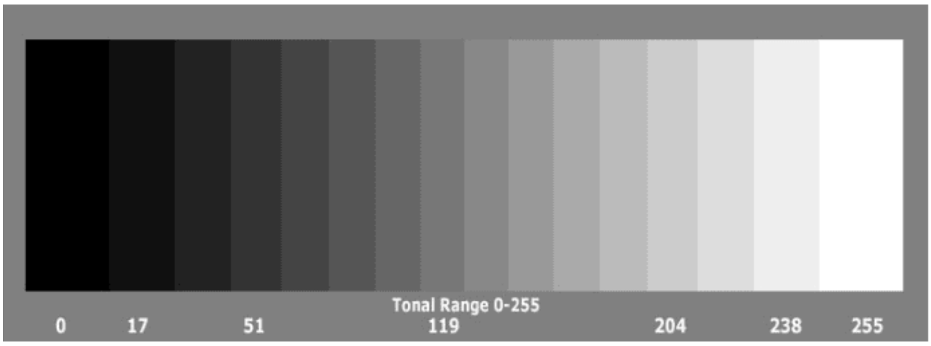


Figure 7: Full range of colours that the grayscale model can describe. Source [14]

3.4.2 Conversion Process

The conversion from RGB to greyscale is based on a weighted sum of the RGB components. The aim is to compute a single intensity value that represents the perceived brightness of the colour. The human eye perceives brightness differently for each colour, with green being the most sensitive, followed by red and blue. Hence, the contribution of each colour channel to the greyscale intensity is weighted.

Conversion Methods [14]**:**

3.4.2.1 Average Method:

The simplest method to convert RGB to greyscale is to take the average of the three components:

$$\mathrm{Grey} = \frac{\mathrm{R+G+B}}{3} \tag{6}$$

This method treats each colour channel equally but does not account for human perception.

3.4.2.2 Luminosity Method:

A more accurate method is the luminosity method, which weights the channels according to human perception. The formula typically used is:

$$\mathrm{Grey} = 0.299 \cdot \mathrm{R} + 0.587 \cdot \mathrm{G} + 0.114 \cdot \mathrm{B} \tag{7}$$

These coefficients are derived from the luminance perception characteristics of the human eye, as the green channel has the most significant impact on perceived brightness, followed by red, and then blue.

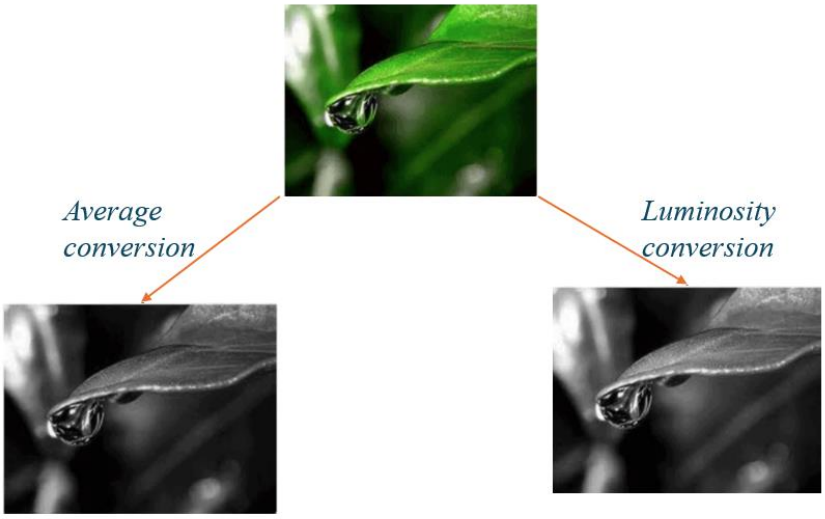


Figure 8: RGB to Greyscale image conversion comparison of Average and Luminosity conversions. Source [14]

3.5 Camera calibration

Since camera calibration makes it possible to extract metric information from 2D images, it is crucial for many computer vision applications. Because Zhang's method is robust and easy to apply, it is a commonly used technique for camera calibration. The theoretical foundation for Zhang's approach is as follows:

3.5.1 Introduction to Camera Calibration

The objective of camera calibration is to ascertain the camera's intrinsic and extrinsic parameters:

Intrinsic Parameters: These details highlight the camera's internal characteristics, such as:

- Focal length $\left((f_x, f_y)\right)$
- Principal point $\left((c_x, c_y)\right)$
- Skew coefficient (s) which is typically zero for most cameras.

Extrinsic Parameters: These define the camera's location and orientation in relation to the global coordinate system, including:

- **Rotation matrix** $((R))$
- **Translation vector** $((T))$

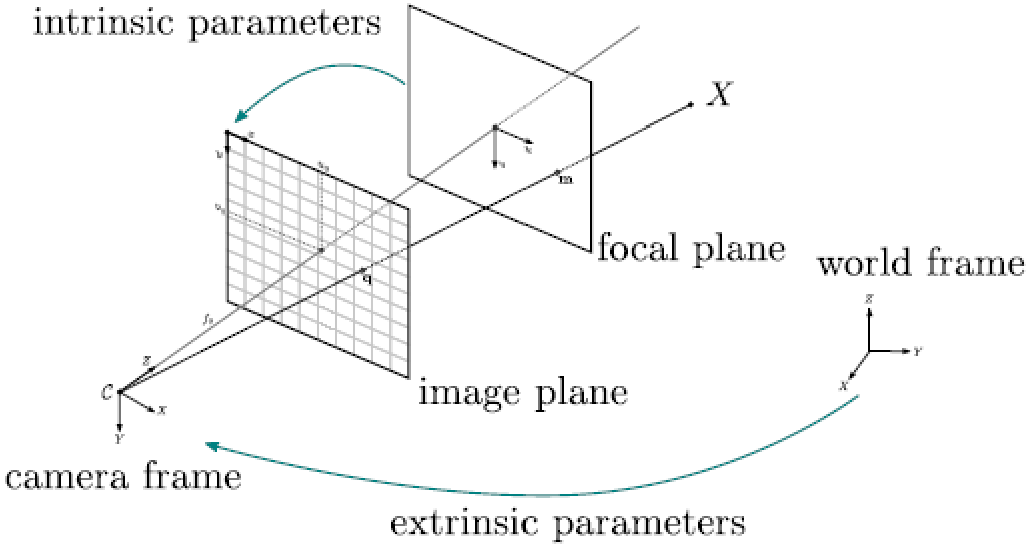


Figure 9: The pinhole camera model. An oriented central projective camera. Source [15]

3.5.2 Zhang's Method Theory

Zhang's method for camera calibration is a robust and practical approach that uses multiple images of a planar calibration pattern (e.g., a chessboard) taken from different angles. The method involves the following key steps [16]:

1. **Corner Detection:**
    - The algorithm detects the corners of the calibration pattern in the image.
    - These detected corners provide the 2D image points, which correspond to known 3D points in the world coordinates (the calibration pattern).
2. **Homography Estimation:**
    - For each image, a homography matrix is computed that relates the 2D image points to the 3D points on the plane.
    - The homography is a projective transformation that maps points from the 2D plane in the world to the image plane.
3. **Initial Parameter Estimation:**
    - From the homographies, the intrinsic camera parameters are estimated. The intrinsic parameters include the focal lengths, principal point, and skew coefficient.
    - The intrinsic parameters are used to decompose each homography into the rotation and translation vectors, which represent the camera's pose relative to the calibration pattern.
4. **Non-linear Optimization:**
    - The initial estimates are refined by minimising the re-projection error using non-linear optimisation techniques. The re-projection error is the difference between the observed image points and the projected 3D points using the estimated parameters.

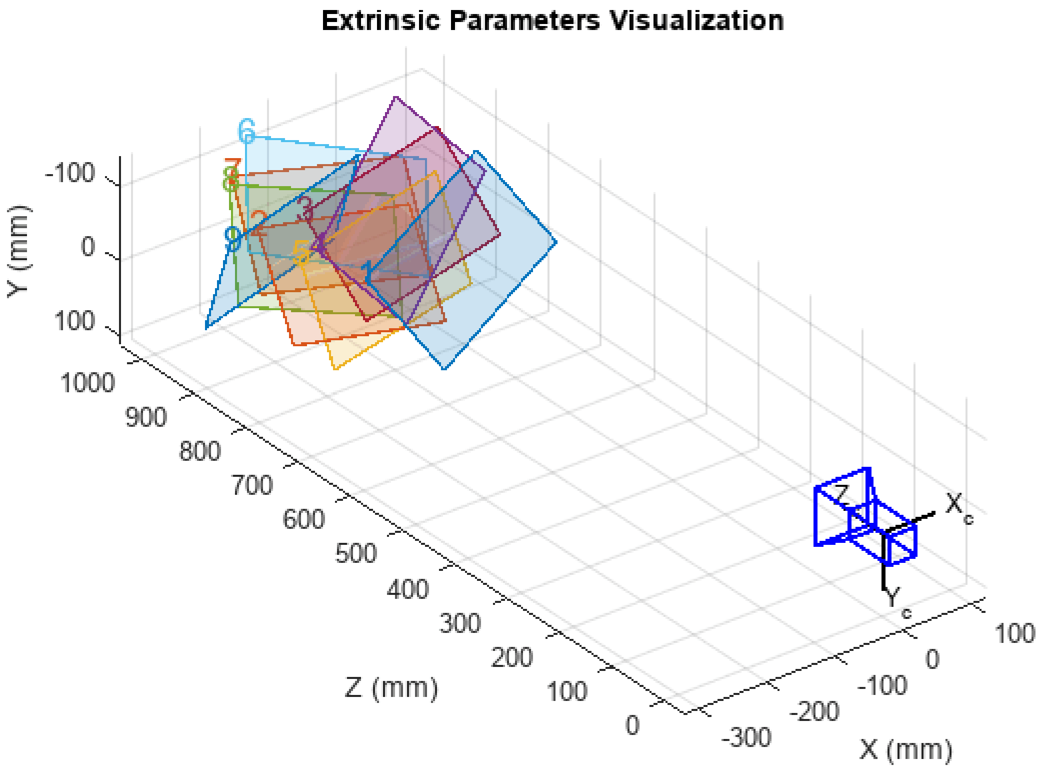


Figure 10: Extrinsic parameter visualisation using Zhang's Method. Source [17]

3.6 Edge detection

Edge detection is a fundamental task in image processing and computer vision, aimed at identifying points in an image where the image brightness changes sharply. These points often correspond to object boundaries or other significant features in the image. Edge detection using gradients involves calculating the gradient of the image intensity function. Several operators, such as the Sobel and Laplacian, are used to compute these gradients.

The gradient of an image is a vector that has both magnitude and direction. It points in the direction of the maximum rate of change of intensity and its magnitude represents the rate of change. The gradient is computed as the partial derivatives of the image intensity function with respect to spatial coordinates.

Given an image $(I(x, y))$:

- The gradient in the x-direction $((G_x))$ is the partial derivative of (I) with respect to (x)
- The gradient in the y-direction $\left((G_y)\right)$ is the partial derivative of (I) with respect to (y).
- The magnitude of the gradient at each pixel can be computed as:

$$G = \sqrt{G_x^2 + G_y^2}] \qquad (8)$$

- The direction (or orientation) of the gradient is given by:

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{9}$$

3.6.1 Laplacian gradient

The Laplacian operator is a second-order derivative operator that computes the rate at which the gradient changes. It is used to find regions of rapid intensity change and is often used in combination with Gaussian smoothing to reduce noise sensitivity.

The 2D Laplacian of an image is defined as:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{10}$$

- A commonly used discrete approximation of the Laplacian is:

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Another popular variant includes diagonals:

$$L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

3.6.2 Sobel gradient

The Sobel operator is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. It uses convolution with two 3x3 kernels, one for the x-direction and one for the y-direction.

- The Sobel kernel for the x-direction S_y is:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- The Sobel kernel for the y-direction S_y is:

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The gradient approximations are then computed by convolving these kernels with the image:

$$G_x = I * S_x \tag{11}$$

$$G_y = I * S_y \tag{12}$$

3.7 Interpolation

Interpolation is a method used to estimate unknown values that fall between known values. It is widely used in numerical analysis, data analysis, and computer graphics for reconstructing missing data, smoothing data, and resampling images. Different interpolation methods vary in complexity and application. Here we discuss four common types: linear interpolation, polynomial interpolation, cubic spline interpolation, and nearest neighbour interpolation.

3.7.1 Linear Interpolation

Linear interpolation is the simplest form of interpolation. It assumes that the change between two points can be approximated by a straight line.

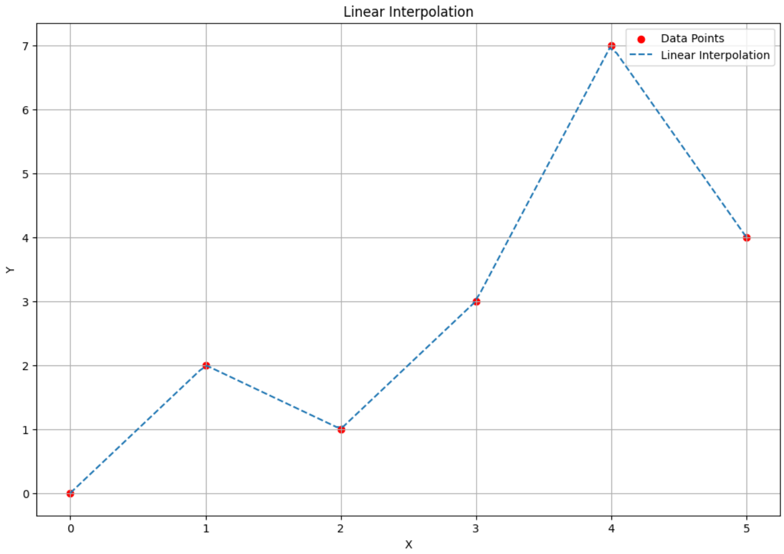


Figure 11: Graph showing the linear interpolation

For two points $((x_0, y_0))$ and $((x_1, y_1))$, the linear interpolation formula for a point (x) between (x_0) and (x_1) is:

$$y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0} \tag{13}$$

Advantages:

- **Straight Line**: Connects two known points with a straight line.
- **Ease of Use**: Simple to implement and computationally efficient.
- **Applications**: Commonly used in real-time applications due to its speed and simplicity.

3.7.2 Polynomial Interpolation

Polynomial interpolation uses polynomials to interpolate a set of data points. The goal is to find a single polynomial of degree $n - 1$ that passes through (n) data points.

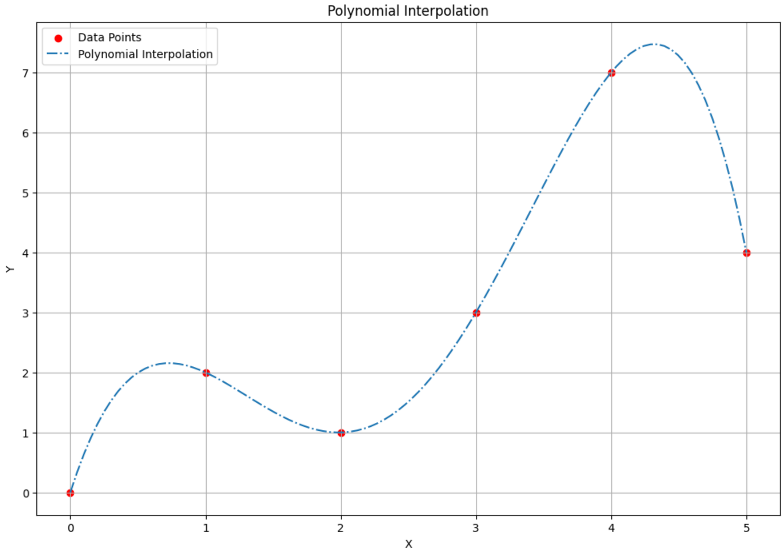


Figure 12: Graph showing the polynomial interpolation

The interpolating polynomial in the Lagrange form is:

$$P(x) = \sum_{i=0}^{n} y_i \underset{0\leq j\leq n j\neq i}{\square} \prod \frac{x-x_j}{x_i-x_j} \tag{14}$$

Advantages:

- **Flexibility**: Higher-degree polynomials can fit more complex data.
- **Oscillation Problem**: High-degree polynomials can exhibit oscillations between points (Runge's phenomenon).
- **Applications**: Useful when a smooth and continuous curve fitting multiple points is required.

3.7.3 Cubic Spline Interpolation

Cubic spline interpolation is a type of spline interpolation where the interpolant is a piecewise cubic polynomial that passes through a set of data points and has continuous first and second derivatives.

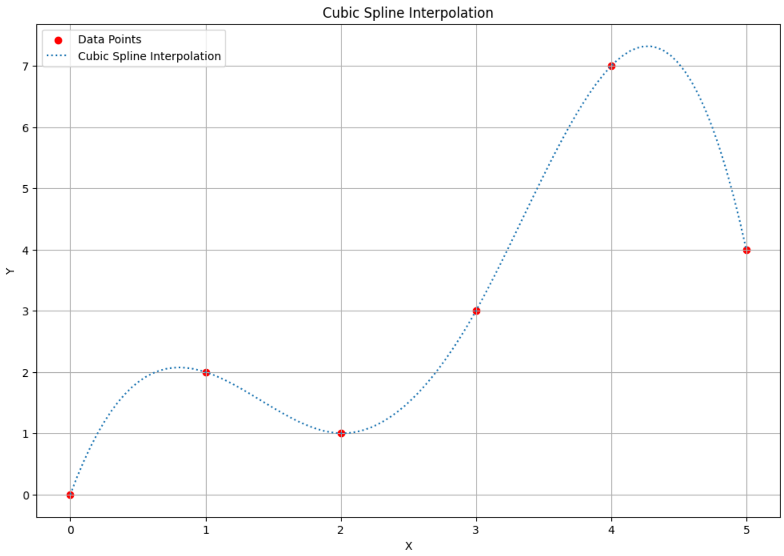


Figure 13: Graph showing the cubic spline interpolation

For $(n + 1)$ points $((x_0, y_0), (x_1, y_1), \dots, (x_n, y_n))$, cubic spline interpolation constructs (n) cubic polynomials $S_i(x)$ for $i = 0,1, \dots, n - 1$ such that:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{15}$$

Subject to the conditions:

- $S_i(x_i) = y_i$
- $S_i(x_{i+1}) = y_{i+1}$
- $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$
- $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$

Advantages:

- **Smoothness**: Ensures smoothness and continuity in the first and second derivatives.
- **Stability**: Avoids the oscillation problem seen in polynomial interpolation.
- **Applications**: Frequently used in computer graphics, data fitting, and animation.

3.7.4 Nearest Neighbour Interpolation

Nearest neighbour interpolation is the simplest method for interpolating between data points. It assigns the value of the nearest data point to the unknown point.

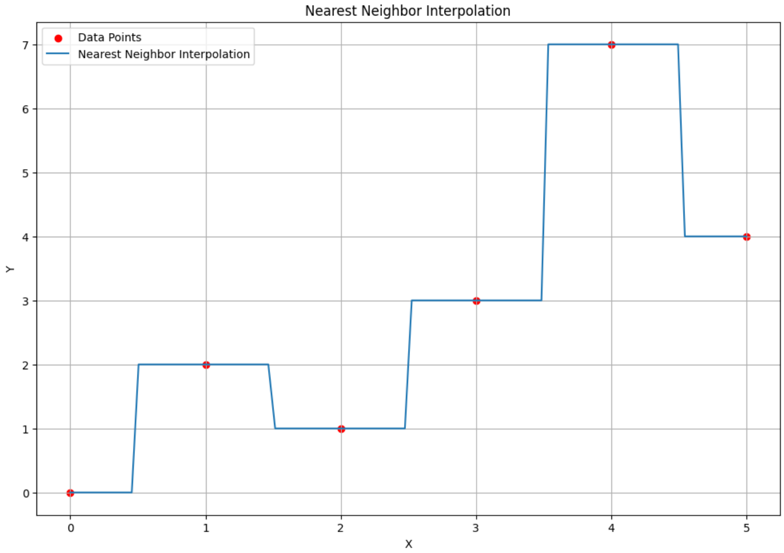


Figure 14: Graph showing nearest neighbour interpolation

For an unknown point(x), find the nearest known point (x_i) . The interpolated value (y) is:

$$y = y_i \tag{16}$$

Advantages:

- **Simplicity**: Very simple and fast to compute.
- **Discontinuity**: Results in a piecewise constant function which can have discontinuities.
- **Applications**: Suitable for categorical data or situations where simplicity and speed are critical.

4. Experiment

4.1 Hardware setup

The experiment involved developing a setup capable of performing 3D scanning simultaneously with 3D printing. The 3D printer employed was the Creality Ender 3. We opted for laser triangulation with reverse geometry, necessitating the 3D printing of an attachment to accommodate a USB nozzle camera and a red line laser diode with a power output of 5mW.

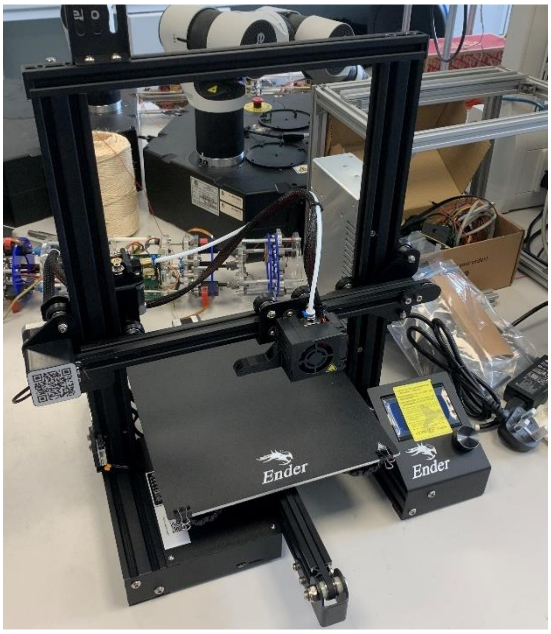


Figure 15: Creality Ender 3 (3D printer used for the experiments)

Determining the appropriate scanning height was a critical step. Through intuitive analysis, we identified that a scanning height of 80mm was optimal for our purposes, as it enabled the capture of small to medium-sized objects without compromising the resolution of the laser line displacement in the captured images. The scanning height calculation was done Mr. Muhammed Sadik and determined to be 80 mm. With a scanning height of 80 mm and an angle of 65, the distance between the camera and the laser was calculated to be 37.3 mm.

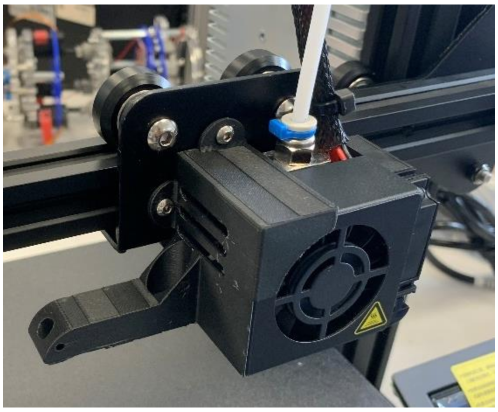


Figure 16: Printer attachment to accommodate the laser and camera

4.2 Software Setup

4.2.1 Laser Line Tracing

The primary objective of the experiment was to accurately determine the precise position of the laser line within the grid of the captured image. To facilitate subsequent calculations, the captured images were converted from RGB to greyscale. This conversion provided a manageable 480x640 2D matrix, as opposed to a more complex 480x640x3 3D matrix.

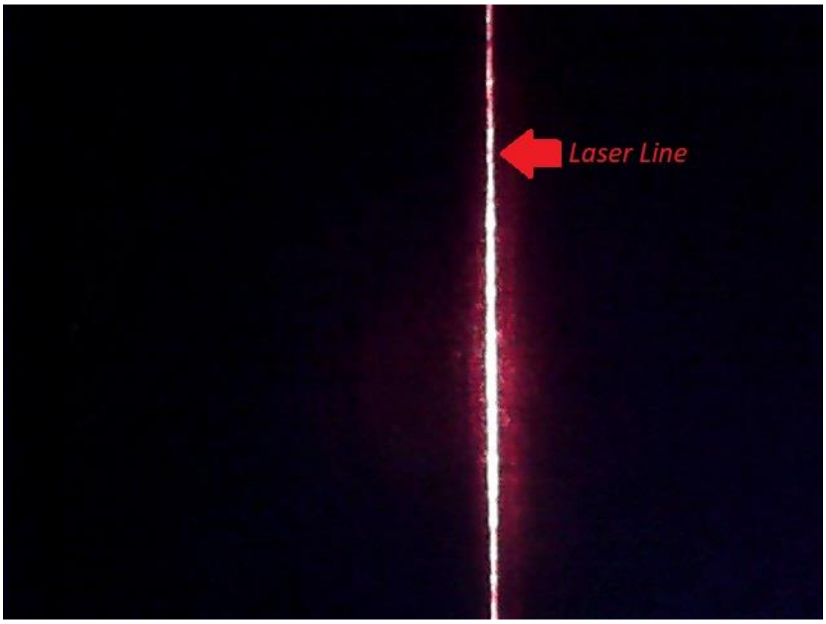


Figure 17: Laser line in RGB

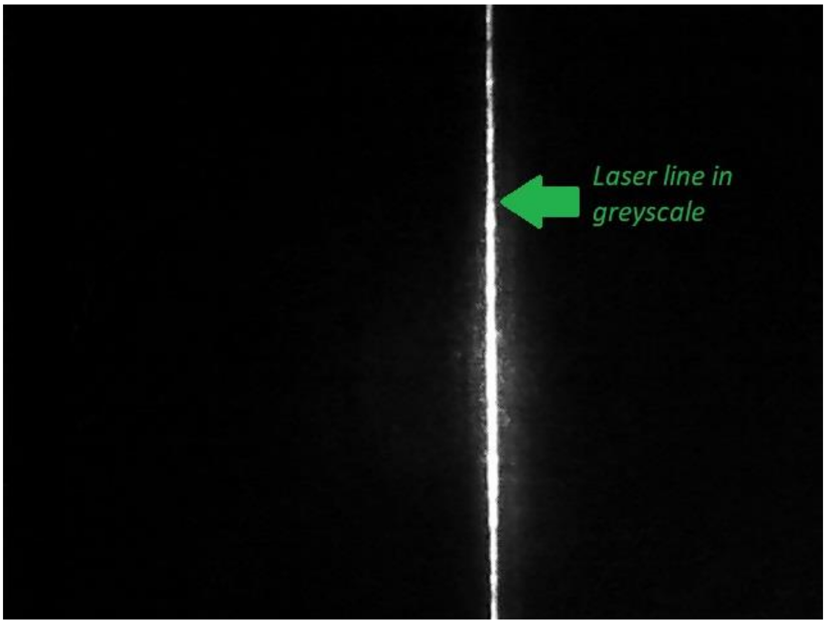


Figure 18: Laser line after RGB to Greyscale conversion using Luminosity method

Precisely determining the location of the laser line within the 2D grid proved challenging. In the captured image, the laser line appears as a cluster of varying intensities across multiple pixels rather than a single pixel.

The initial approach, inspired by the methodology outlined in paper by *Muñoz-Rodríguez*, posited that the laser line should correspond to the pixel with the highest intensity along the X-axis. Consequently, the pixels with the highest intensity values were identified row by row to trace the laser line.

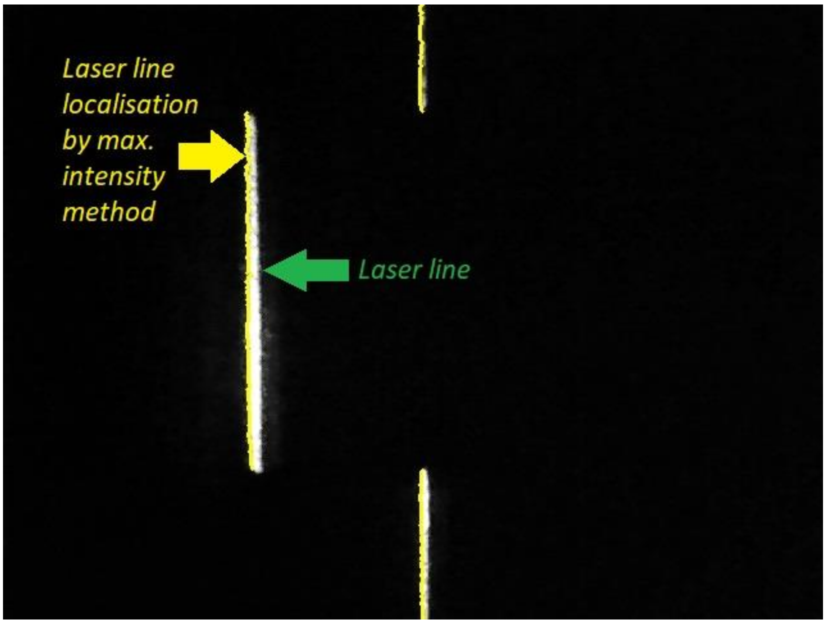


Figure 19: Laser line localisation using Maximum intensity pixel algorithm

The obtained results are suboptimal due to the constraints imposed by limited camera resolution and interference from the laser itself, leading to the inadequacy of using the pixel with maximum intensity to trace the laser line accurately.

To address this issue, a more in-depth analysis of the pixel intensity patterns induced by the laser was conducted. It was observed that pixel intensities in the captured images initially increase and then decrease, resembling a Gaussian bell curve with values concentrated around a mean. This observation enabled us to model the pixel intensity of each row using a bell curve fitting approach, as also discussed in the paper by *Muñoz-Rodríguez*. Implementing this method significantly improved the accuracy of tracing the laser line.

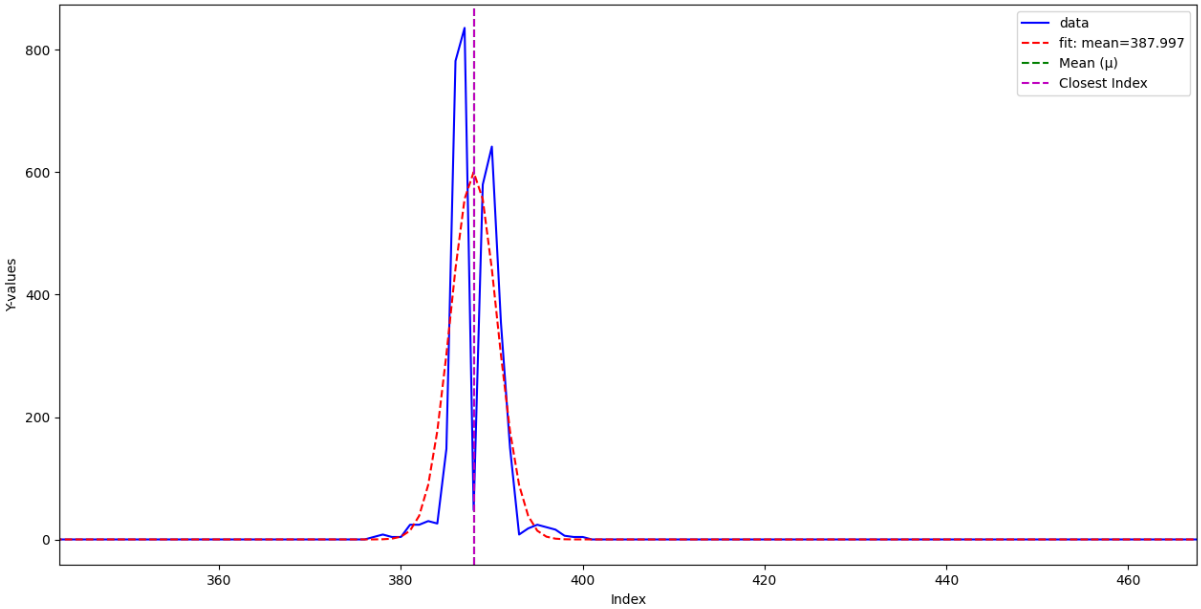


Figure 20: Gaussian curve fitting for localisation of the laser line for pixel intensities

However, this method also presents challenges, as it functions optimally only in highly controlled environments where both the laser and camera remain completely stationary. In our use case, the repeated movements of the scanner head introduce interference and diffraction in the laser line, which prevents the pixel values from conforming properly to a normal distribution.

Consequently, we opted to trace the edge of the laser line as captured by the camera. Edge tracing was implemented using various gradient methods, including the Laplacian gradient and Sobel. The results indicated that employing the Sobel gradient along the x-axis produced the most accurate outcomes.

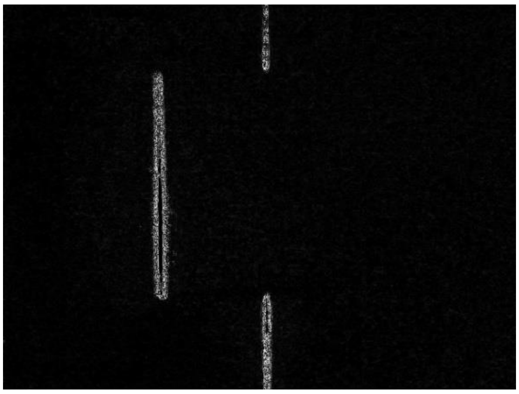


Figure 21: Laser line edge tracing using Laplacian operator

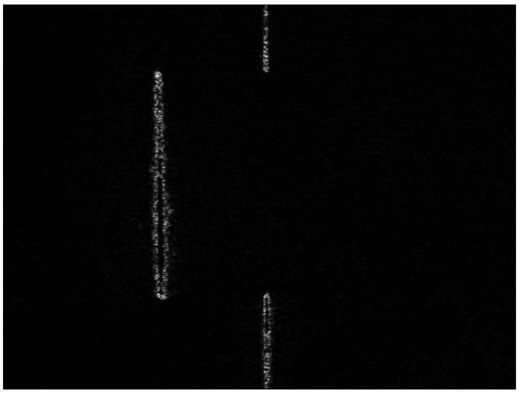


Figure 22: Laser line edge tracing using Sobel Y operator

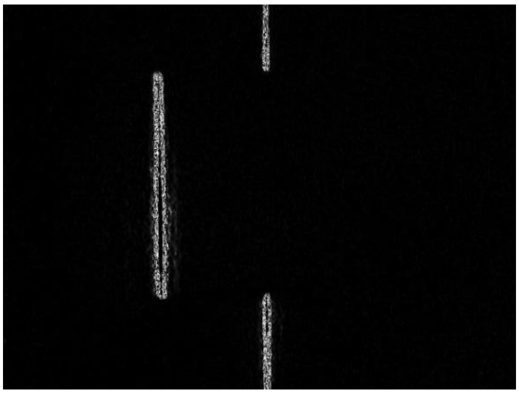


Figure 23: Laser line edge tracing using Sobel X operator

An analysis of the values obtained using the Sobel X gradient showed the existence of both positive and negative values. Fitting a Gaussian bell curve across these values gave a good trace of the outer edge of the laser line, whereas fitting the bell curve of the mirror of these values gave a trace of the inner edge of the laser line. Taking the mean of these outer and inner edge values gives a very good approximation of the laser line trace in the image.

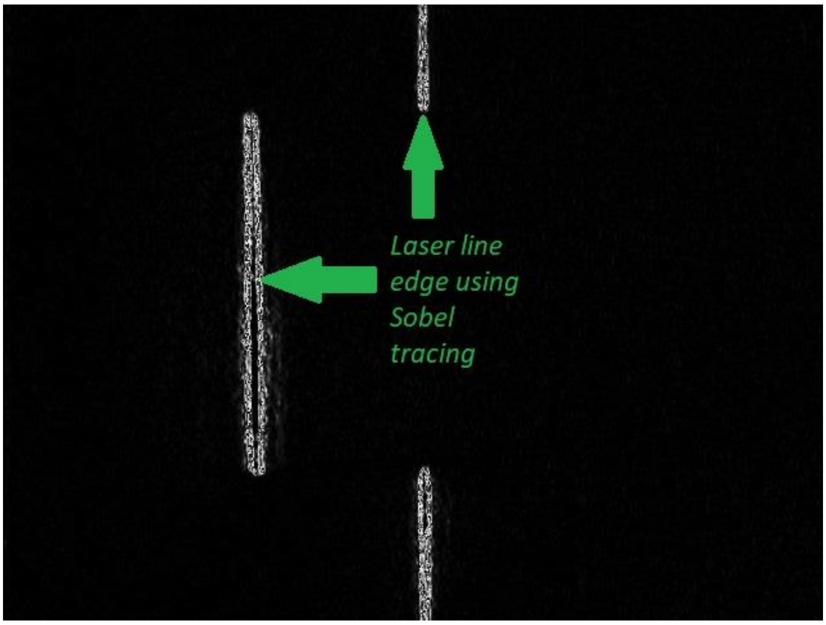


Figure 24: Laser line edge tracing using Sobel X operator

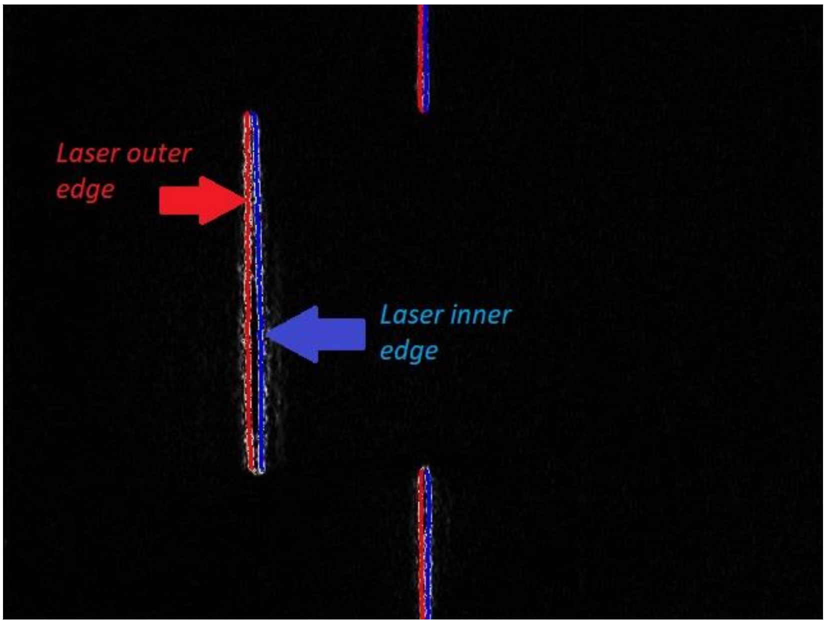


Figure 25: Inner and Outer edge tracing of the laser line

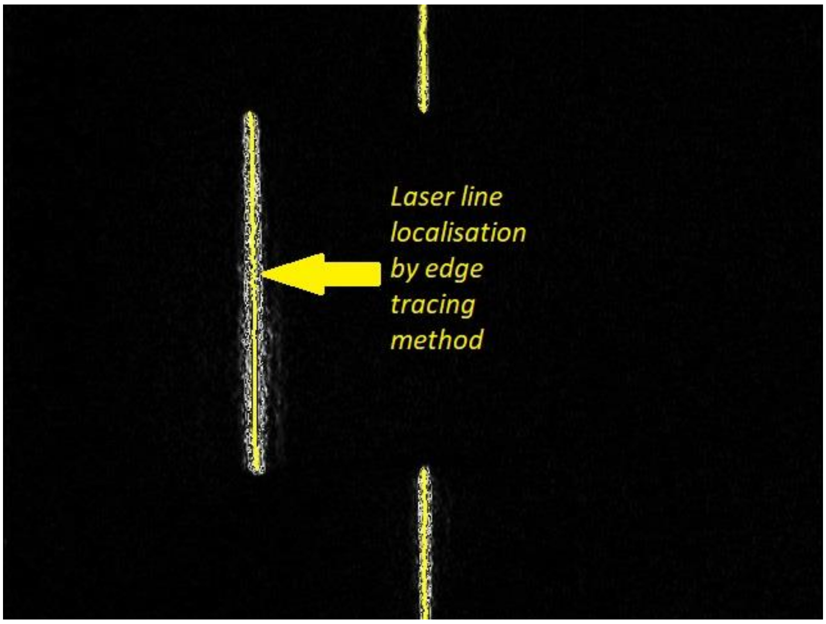


Figure 26: Laser line localisation using the inner and outer edge

4.2.3 Data acquisition

4.2.3.1 Video capture

The initial methodology involved documenting the scanning process via a straightforward video capture. An initial scan of the printer bed was conducted without any objects present. The laser line recorded in this scan was intended to serve as the reference line for calculating pixel displacement caused by the presence of any objects. However, this approach proved inadequate because the movement of the scanner head resulted in the reference line shifting at the start and end of the scan due to vibrations. These vibrations caused significant shifts, leading to incorrect calculations of pixel displacement if the reference line was assumed to remain stationary during the scanning process.

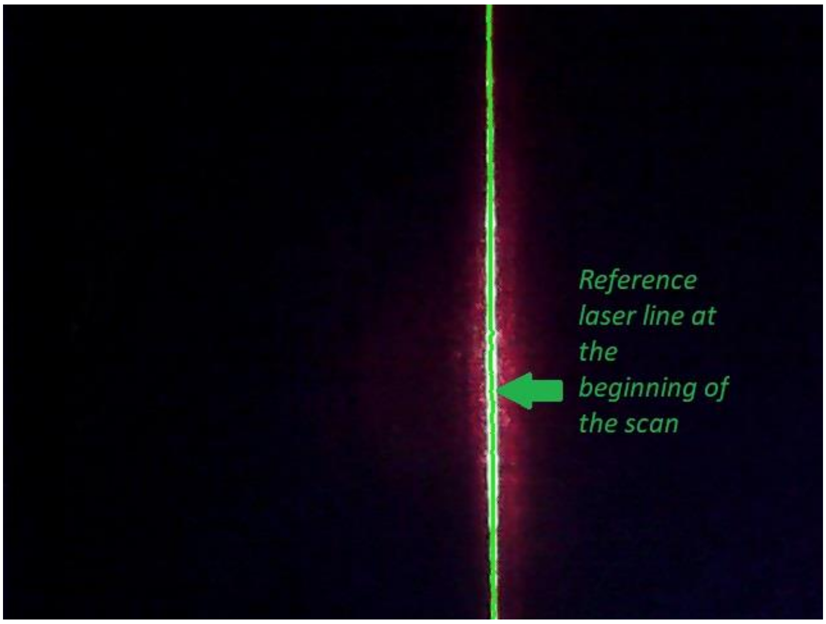


Figure 27: Laser line in the first frame of the scanning process

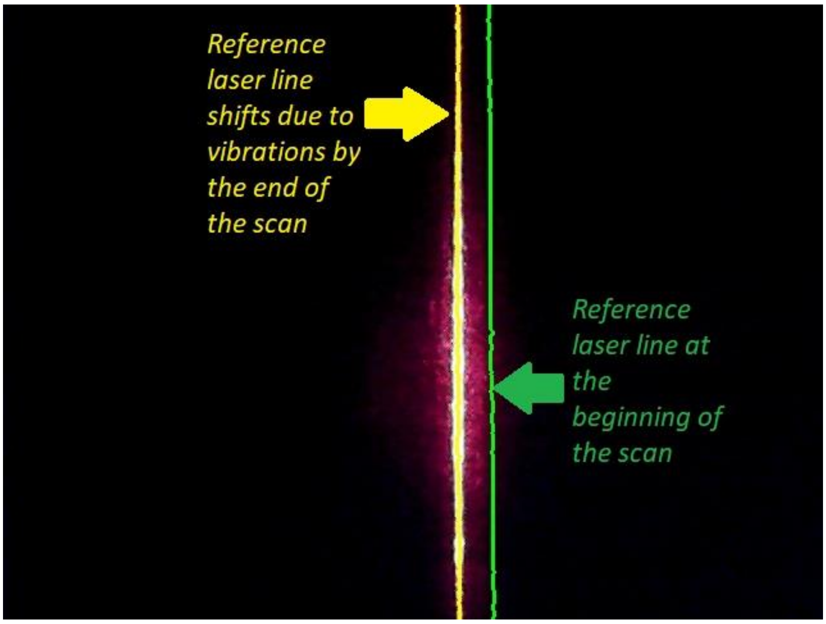

Reference laser line shifts due to vibrations by the end of the scan
Reference laser line at the beginning of the scan

Figure 28: Laser line appears shifted in the last frame of the scanning process

To address the issue of reference line movement during the scanning process, it was determined that the reference line should be recreated for each frame captured during the scanning sequence. The presence of an object causes displacement of some laser line pixels. In contrast, in areas devoid of an object, the laser line pixels remain undisplaced. These pixel values along the X-axis can be utilized to interpolate the laser line values along the Y-axis of the image frame, thereby reconstructing the laser line. Various interpolation methodologies were then employed to calculate the pixel values for the laser line on the Y-axis, and their variance was subsequently calculated. The results are presented below:

Interpolation Method	**Variance (px)**
Linear Interpolation	22.7511
Polynomial Interpolation (deg=2)	23.7864
Polynomial Interpolation (deg=3)	21.5395
Cubic Spline Interpolation	33.7757
Nearest Neighbour Interpolation	22.7511

Table 1: Interpolation methods and corresponding variance

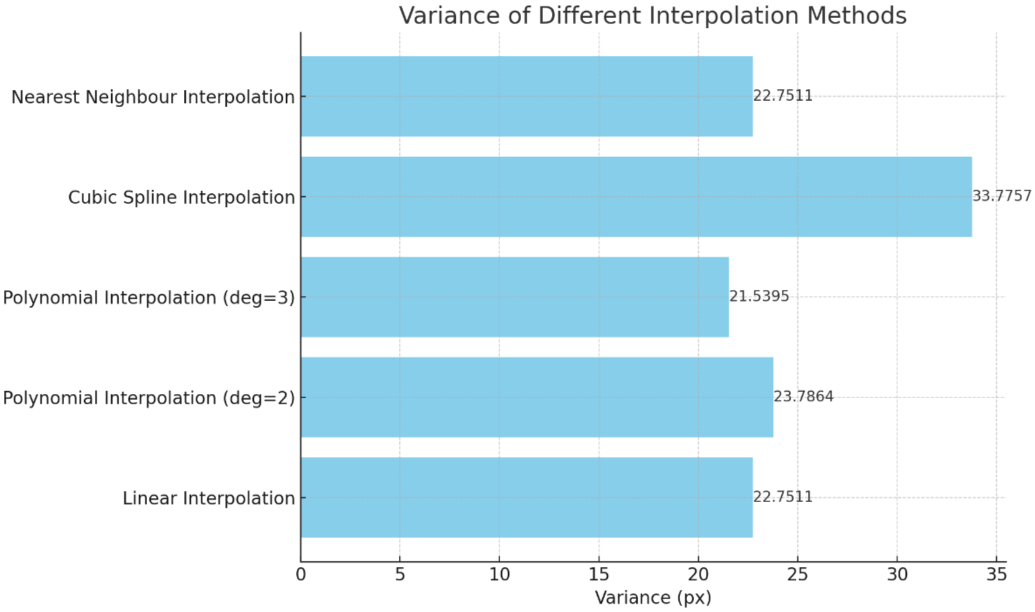


Figure 29: Interpolation comparisons for recreating reference laser line

It was determined that polynomial interpolation of the third degree yields the most accurate results, and therefore, it was employed to reconstruct the reference for the laser line. This dynamic reconstruction of the reference line effectively mitigated the shift of the reference line caused by vibrations at the beginning and end of the scanning process.

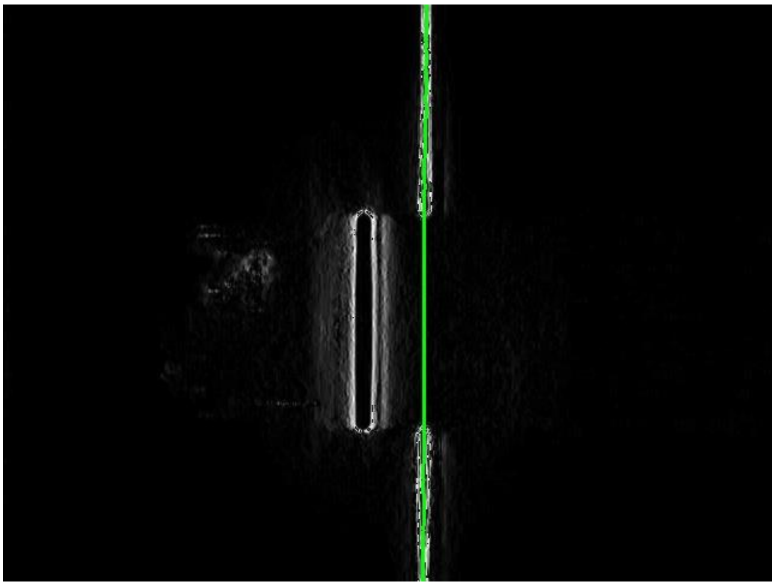


Figure 30: Reference line laser recreating using polynomial interpolation

4.2.3.2 Snapshots at regular intervals

Despite addressing the issue by dynamically recreating the reference line, the pixel displacement cannot be reliably calculated. This unreliability stems from the fact that interpolated values may not accurately represent the actual pixel values of the laser line. The problem is further exacerbated when scanning larger values (exceeding 25 mm), where pixel displacement occurs across the entire frame, rendering the recreation of the reference line infeasible. Consequently, this issue imposes a stringent constraint on the height range of the scanned object.

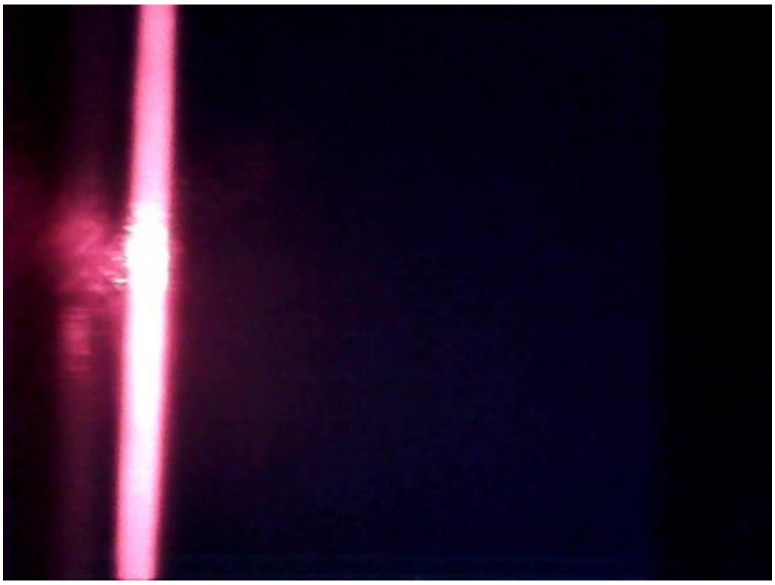


Figure 31: Laser line reference disappears for 30 mm cube making reference line recreating unfeasible

Fortunately, this issue was addressed through the advancement of a new hardware apparatus that facilitated the capture of snapshots of the printer bed at regular intervals. Consequently, prior to the commencement of the printing process, the entire printer bed was scanned at 1mm intervals. This ensured that the reference line for each position of the scanning head was precisely defined, thereby eliminating any potential shifts caused by the movement of the scanning head or distortions introduced by the presence of objects.

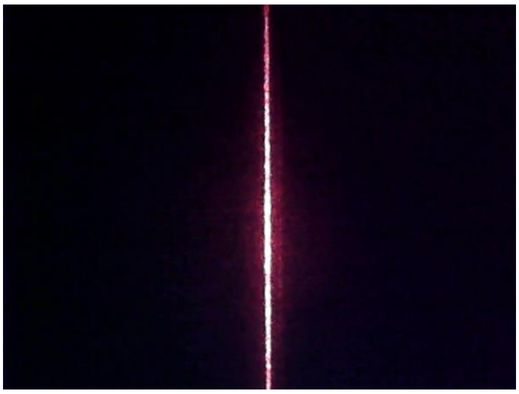


Figure 32: Laser line reference sample for 0th index

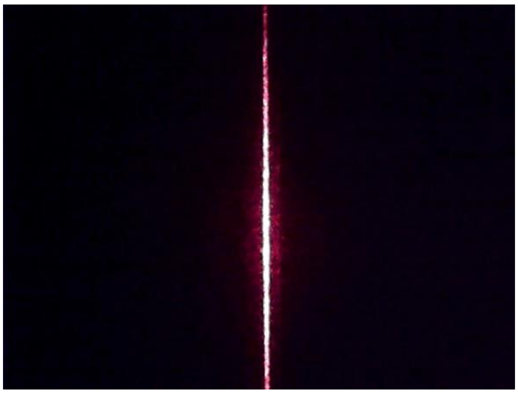


Figure 33: Laser line reference sample for Nth index

4.3 Camera Calibration

Camera calibration was conducted using OpenCV's implementation of Zhang's method. This procedure yielded the intrinsic matrix, as well as the rotation and translation parameters. Additionally, the calibration process established the relationship between pixels in the captured image and their corresponding real-world measurements, determined by calculating the average pixel distance in relation to the known dimensions of the chessboard squares.

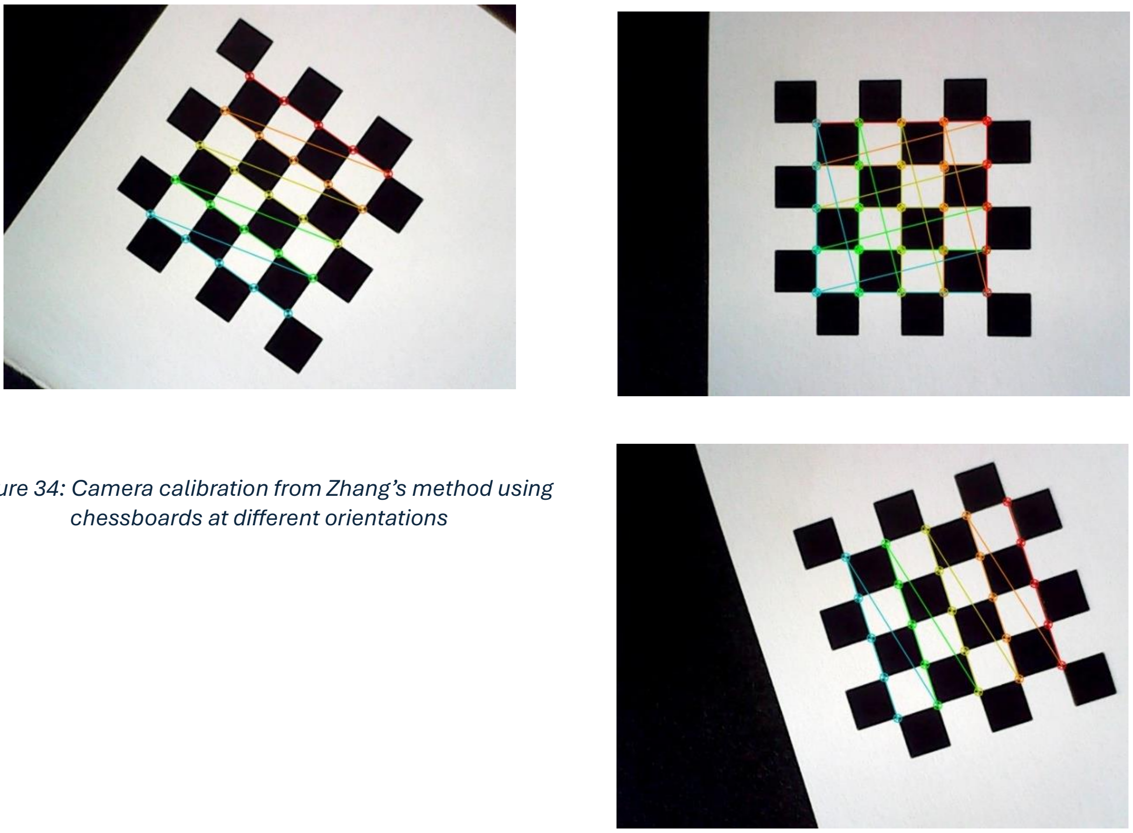


Figure 34: Camera calibration from Zhang's method using chessboards at different orientations

5. Results

Figures 35 and 36 illustrate the objects designated for scanning. Prior to the scanning process, camera calibration was conducted, which provided the pixel-to-millimetre conversion factor (k), intrinsic matrix (I), rotation vector (R), and translation vectors (T). The scanning was executed by projecting a vertical laser line onto the surface of the printer bed at a height (D) of 80 mm. The distance (d) between the camera and the laser diode was 37.3 mm. The camera captured RGB frames with a resolution of 480x640 pixels, which were subsequently converted to grayscale images of the same resolution. Data acquisition was performed using various methodologies, as detailed in Sections 5.1 and 5.2. The presence of scanning objects resulted in pixel displacement of the laser line in some frames. Frames were processed to extract the laser line edges utilizing the Sobel X gradient. The edge traces were further analysed through Gaussian curve fitting with a numeric mirror to determine both the outer and inner edges of the captured laser line. The methodologies for calculating pixel displacement are also outlined in Sections 5.1 and 5.2. The pixel displacement was then divided by the (k) pixel/mm (computed during camera calibration) to compute $s(x,y)$. The $s(x,y)$ values were subsequently incorporated into Formula (5) to determine the corresponding heights for each segment of the laser line. These heights were stored in depth maps for each frame.

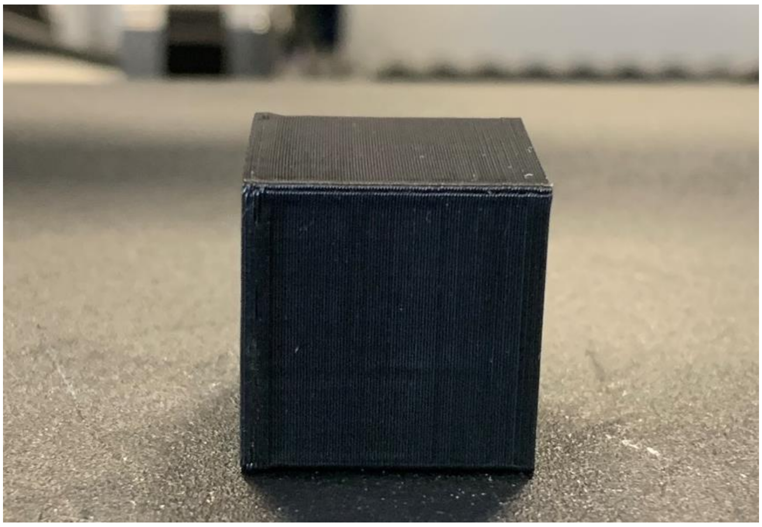


Figure 35: Object to be scanned (Black cube)

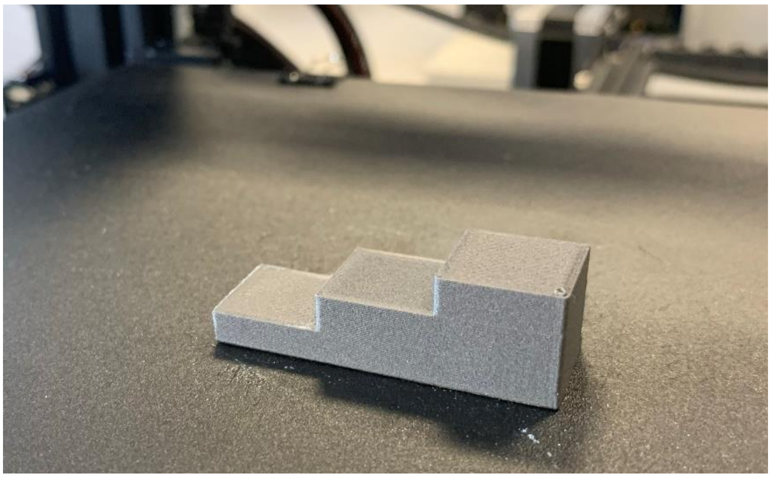


Figure 36: Object to be scanned (Grey staircase)

5.1 Data Acquisition: Video capture

During the scanning process, the scanner head was traversed from one end of the printer bed to the other along the X-axis. Concurrently, the camera recorded video footage of the scanning process. The presence of scanning objects led to the displacement of some laser line pixels. The reference line was reconstructed using the method described in Section 4.2.3.1. Depth maps were generated for each video frame, with the measured object heights and the average height values detailed in Table 2.

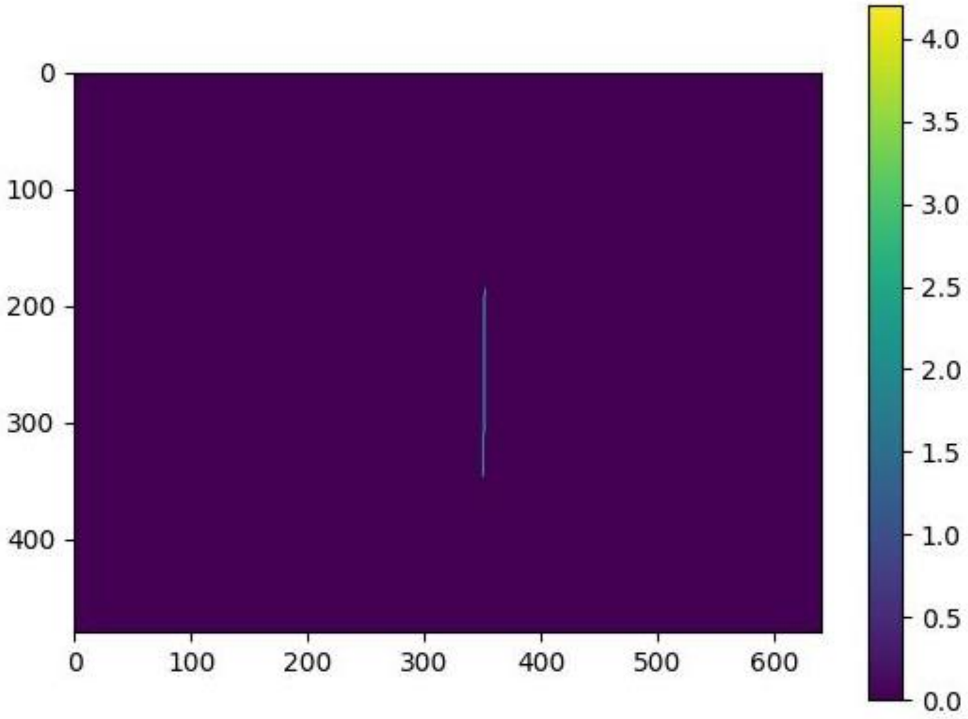


Figure 37: Depth map sample for data acquisition using video capture

Measured Height (mm)	Calculated Height (mm)	Error (mm)	% Error
3.68	3.98	0.3	8.152173913
8.94	8.95	0.01	0.111856823
14	14.03	0.03	0.214285714

Table 2: The measured heights, calculated heights, and the corresponding errors from data acquisition using video captured

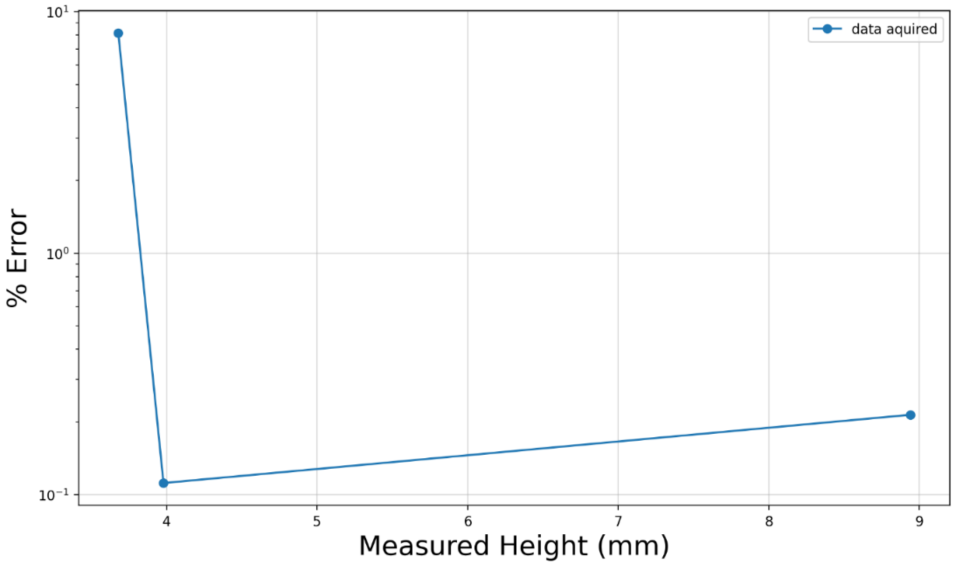


Figure 38: Relationship between % error and measured height for data acquisition using video capture

5.2 Data Acquisition: Snapshots at measured steps

The scanner head was traversed across a range of 64 mm, with images captured at 1 mm intervals. Initially, a scan was performed in the absence of any objects to record the position of the laser line at each sample location. These reference line positions were documented for subsequent calculations. After positioning the scanning objects on the printer bed, the scanning process was restarted. The outer and inner edge traces of the objects were utilised to ascertain the laser line's position within the images. Pixel displacement measurements were then compared to the previously computed reference lines to precisely quantify the pixel displacement for each image.

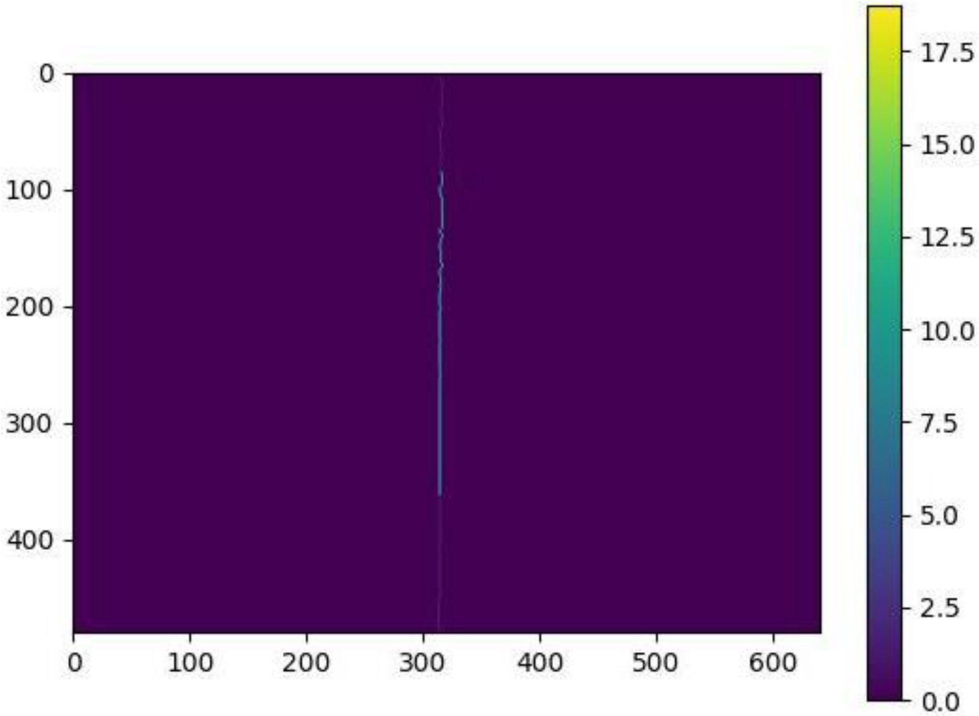


Figure 39: Depth map sample for data acquisition using snapshot at measured steps

The depth maps generated throughout the scanning process were integrated to reconstruct the object in three dimensions, as illustrated in Figure 39. The measured height of the object and the average height obtained are detailed in Table 3. The height data were used to generate a mesh grid, which was employed to reconstruct the geometric configuration of the objects, as depicted in Figures 41, 42 and 43.

Measured Height (mm)	**Calculated Height (mm)**	**Error (mm)**	**Error %**
2.35	2.30	0.05	2.22
4.69	4.59	0.10	2.18
9.63	9.45	0.18	1.85
14.55	14.31	0.24	1.66
20.10	19.93	0.17	0.83
24.82	24.84	0.02	0.08

Table 3: The measured heights, calculated heights, and the corresponding errors from data acquisition using snapshots at measured steps

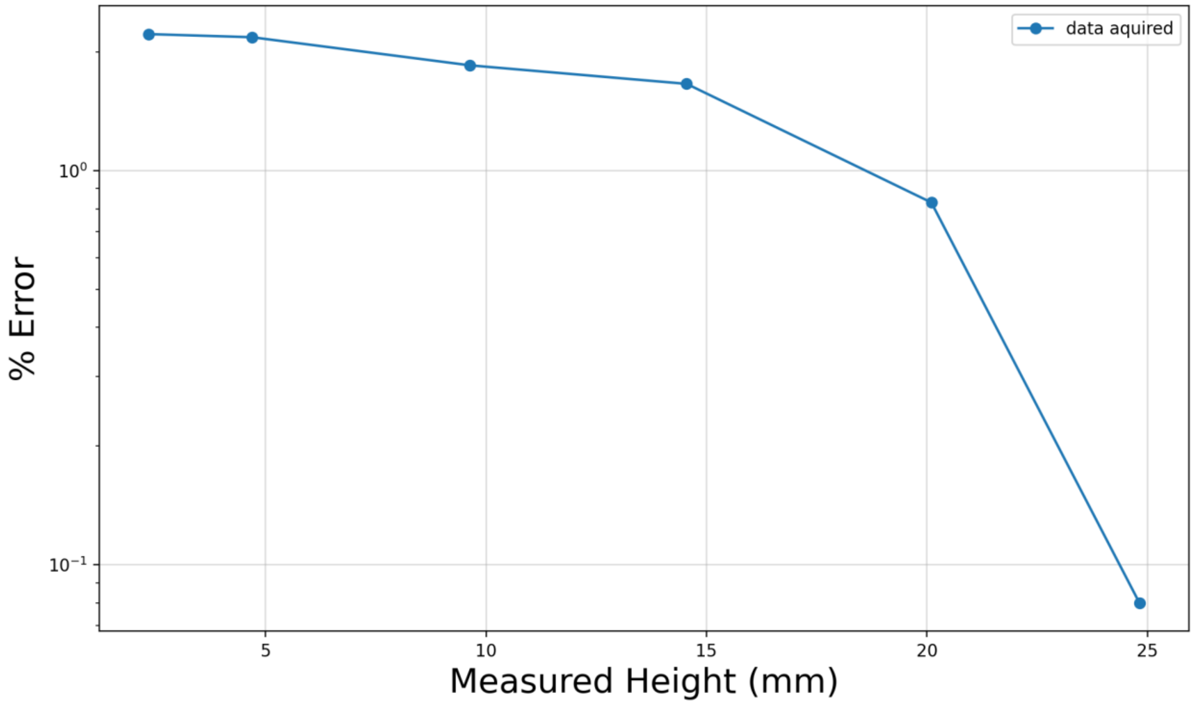


Figure 40: Relationship between % error and measured height for data acquisition using snapshots at measured steps

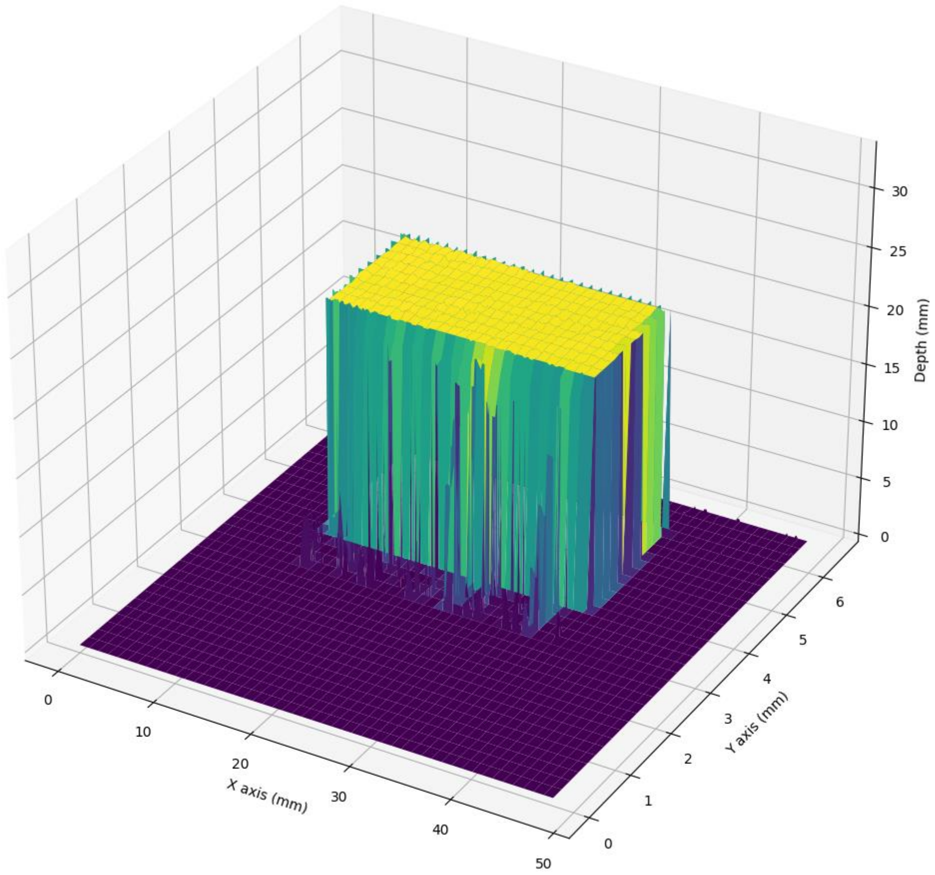


Figure 41 (b): 3D reconstruction of a 20mm cube with snapshots at measured steps from anterior perspective

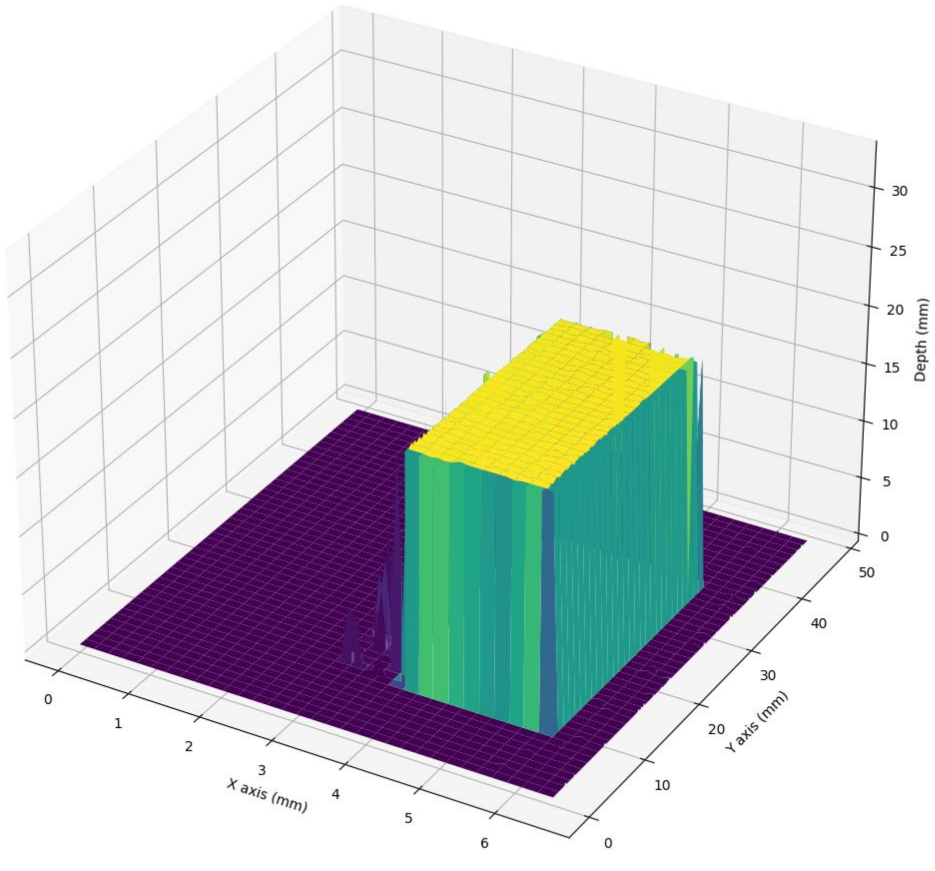


Figure 41 (b): 3D reconstruction of a 20mm cube with snapshots at measured steps from posterior perspective

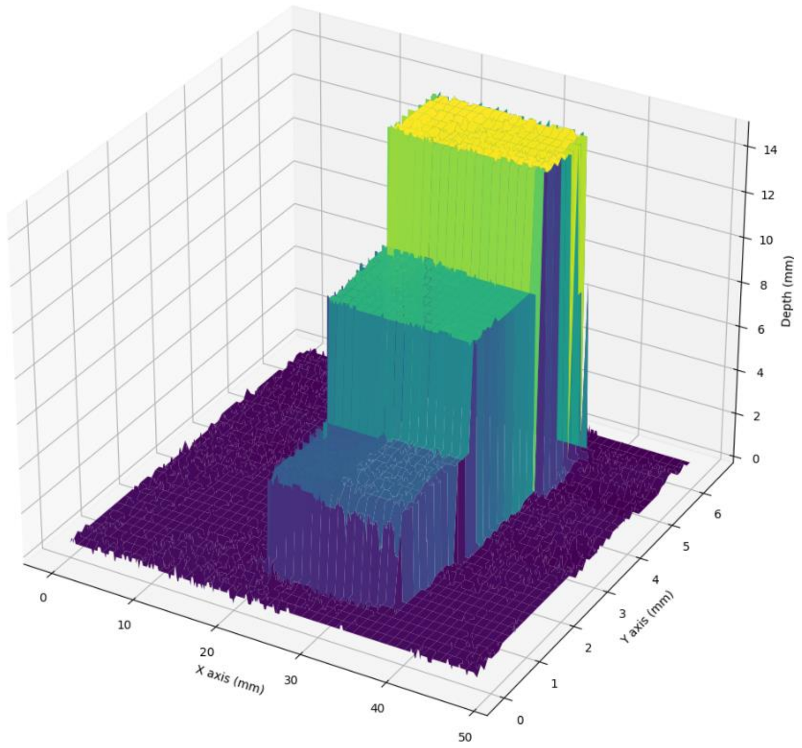


Figure 42 (a): 3D reconstruction of a staircase with snapshots at measured steps from anterior perspective

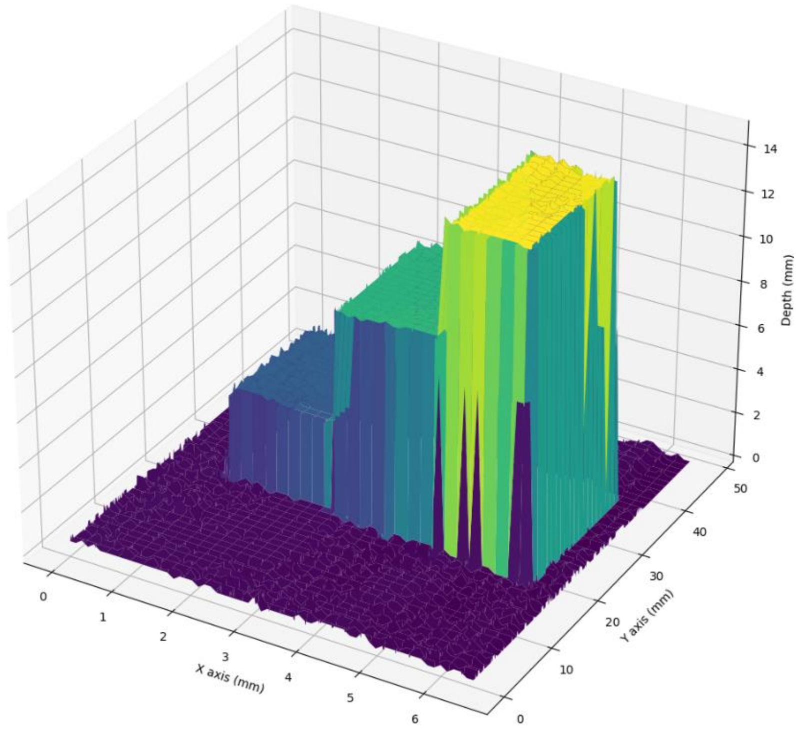


Figure 42 (b): 3D reconstruction of a staircase with snapshots at measured steps from posterior perspective

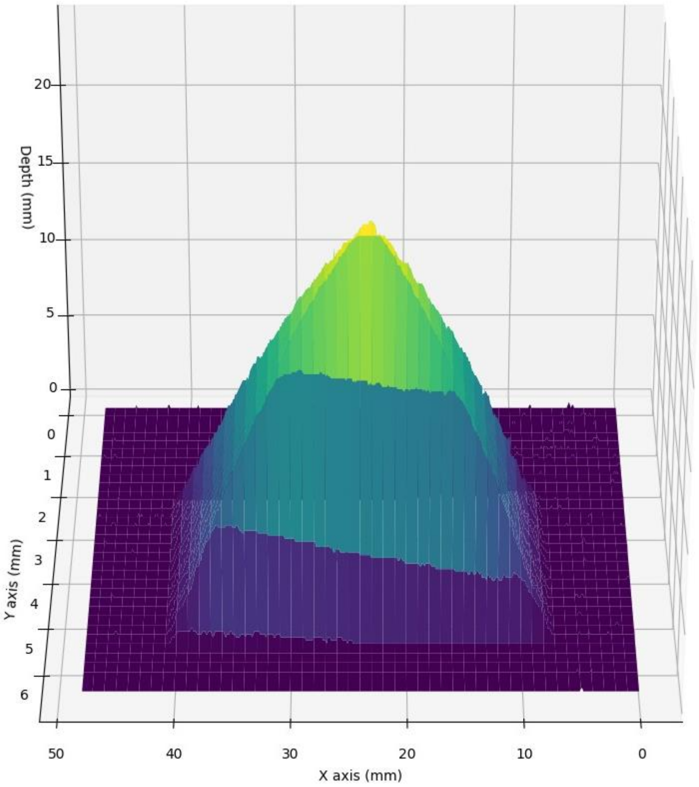


Figure 43 (a): 3D reconstruction of pyramid with snapshots at measured steps from anterior perspective

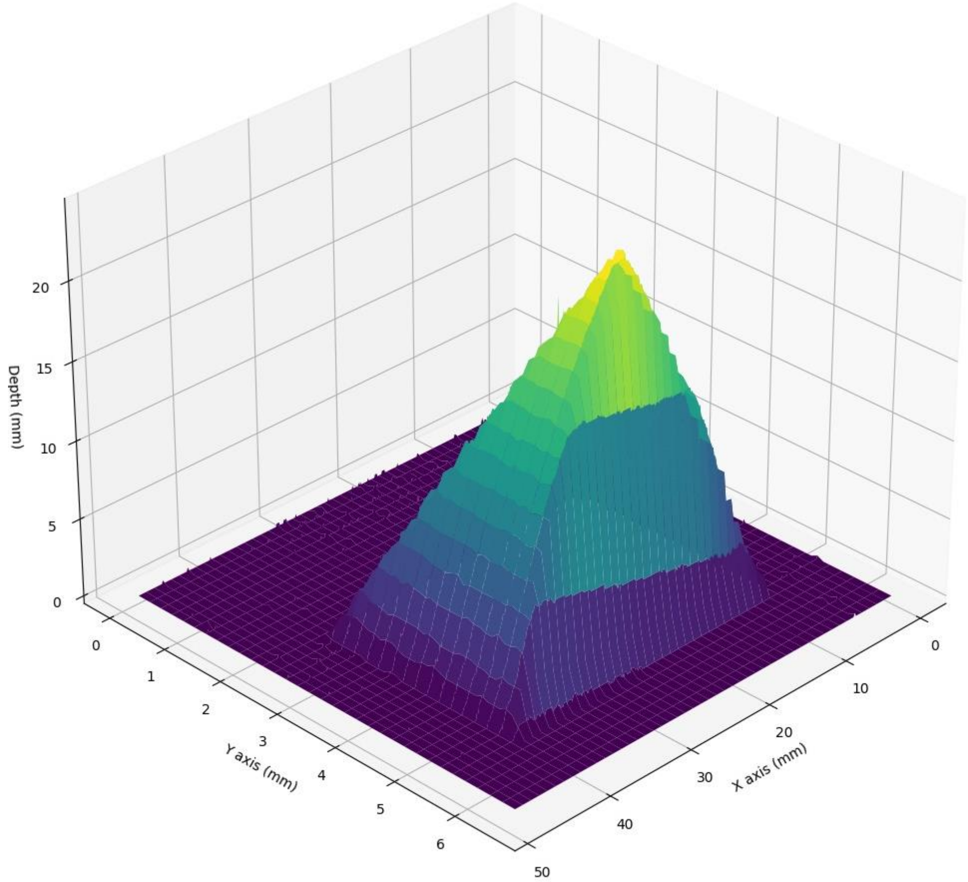


Figure 43 (b): 3D reconstruction of pyramid with snapshots at measured steps from posterior perspective

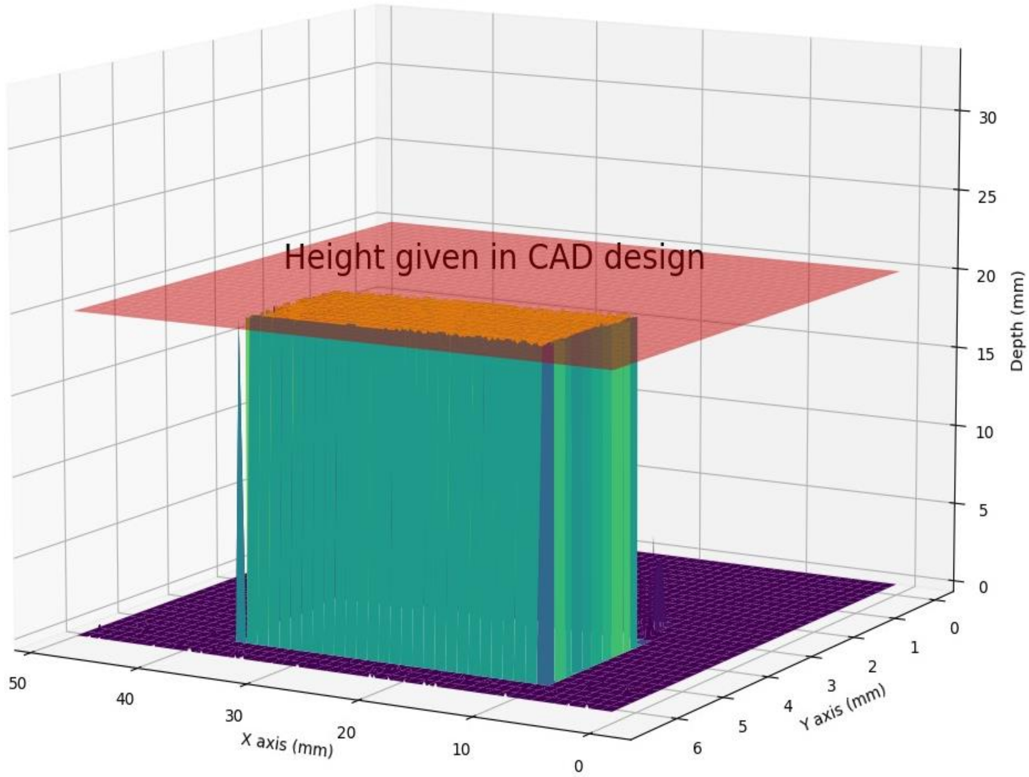


Figure 44 (a): Height comparison between given CAD design and scanned cube

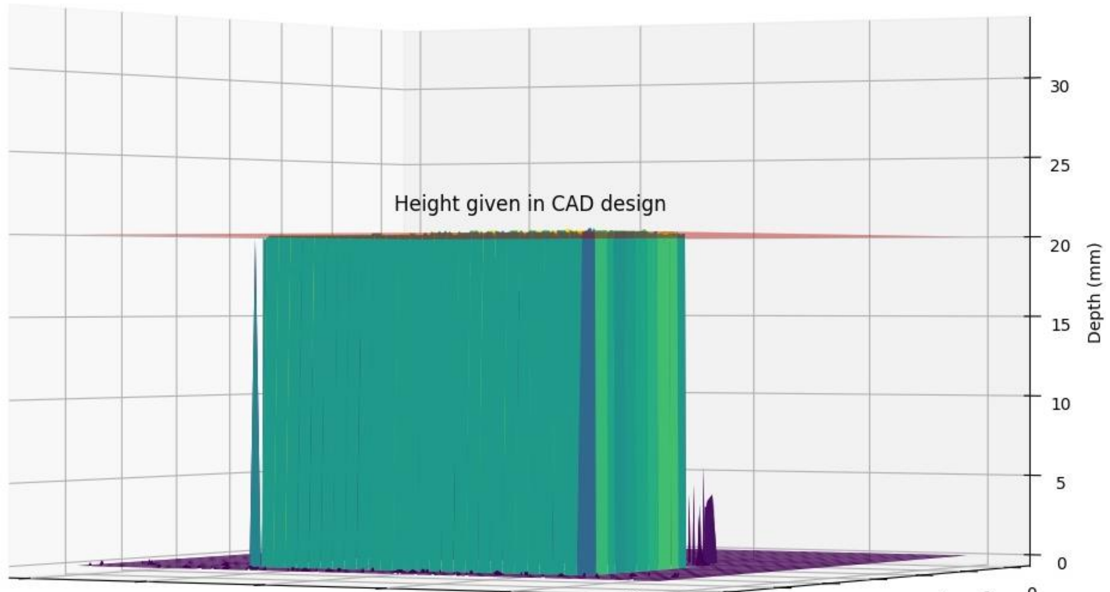


Figure 44 (b): Height comparison between given CAD design and scanned cube

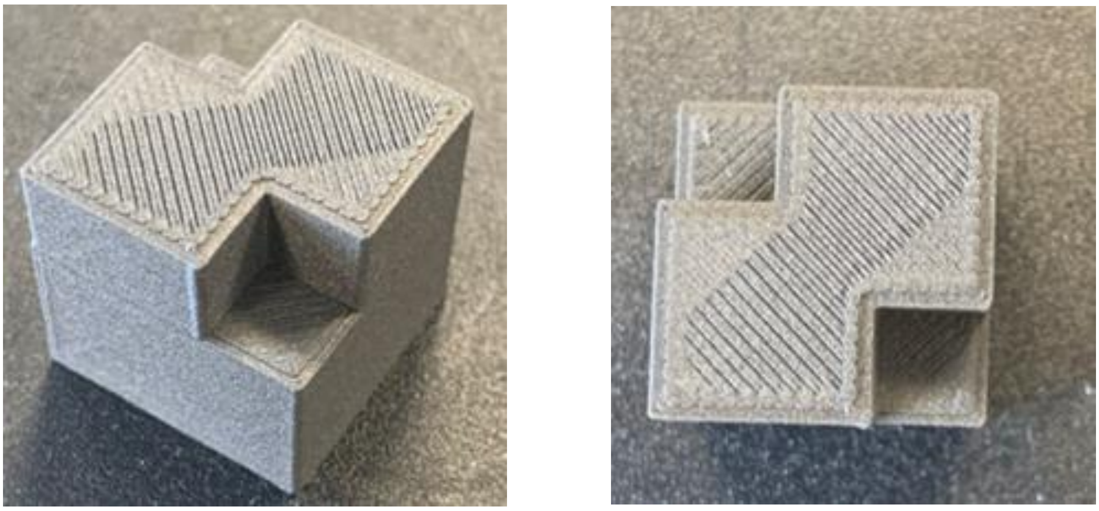


Figure 45: 20 mm cube with explicitly added defects

In Figure 44, the reconstructed geometry of the scanned objects can be compared to the dimensions specified in the CAD design. The promising results of the geometry reconstruction highlight the potential application of the scanning system in detecting defects within printed objects by comparing their physical attributes to the CAD models. To evaluate this capability, deliberate defects were incorporated into the CAD design of a 20 mm cube. The printed object, featuring these intentional defects, was subsequently scanned using the system, as depicted in Figure 45. The scanning process employed the snapshots at measure steps data acquisition method, ensuring a detailed capture of the object's geometry. Following the established reconstruction steps, the geometry of the test object was generated.

The reconstructed geometry distinctly reveals the presence of defects in the test objects. These defects are characterized by areas where the detected height is significantly smaller than the height outlined in the CAD design, indicating discrepancies between the printed object and its digital blueprint. This disparity in height measurements serves as a critical metric for identifying defects. The ability to detect such variations underscores the efficacy of the scanning system in quality control processes, where ensuring the fidelity of printed objects to their original designs is paramount. The results suggest that this scanning technique could be instrumental in maintaining high standards of accuracy and reliability in manufacturing and production environments.

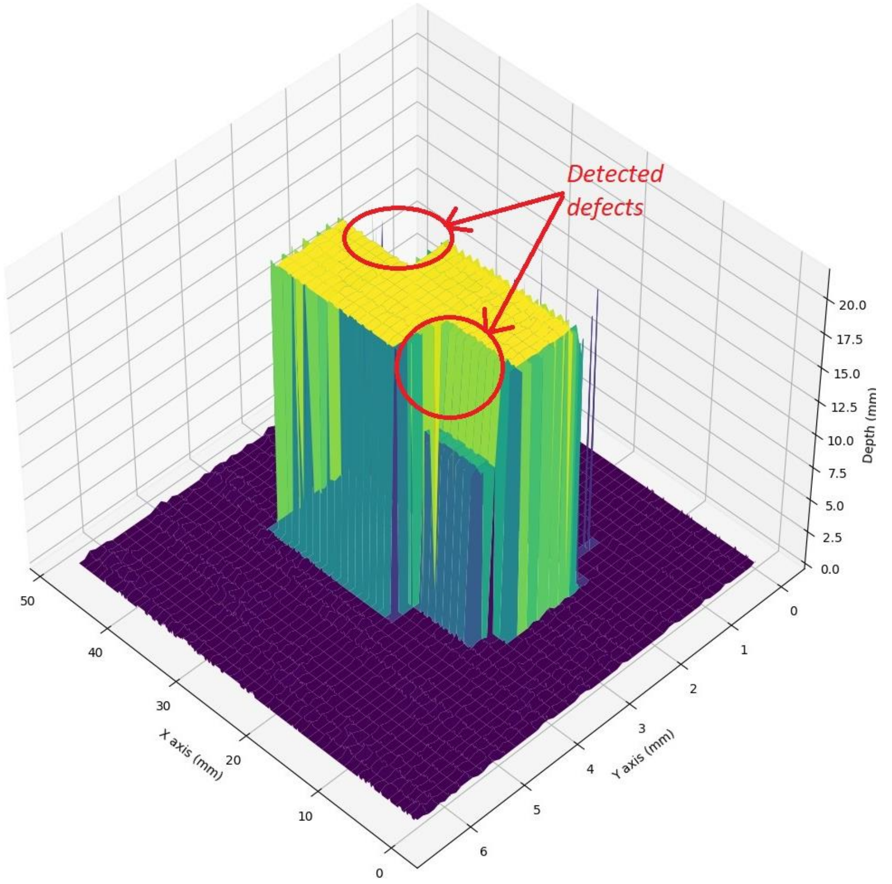


Figure 46 (a): 3D reconstruction of defect test object from anterior perspective

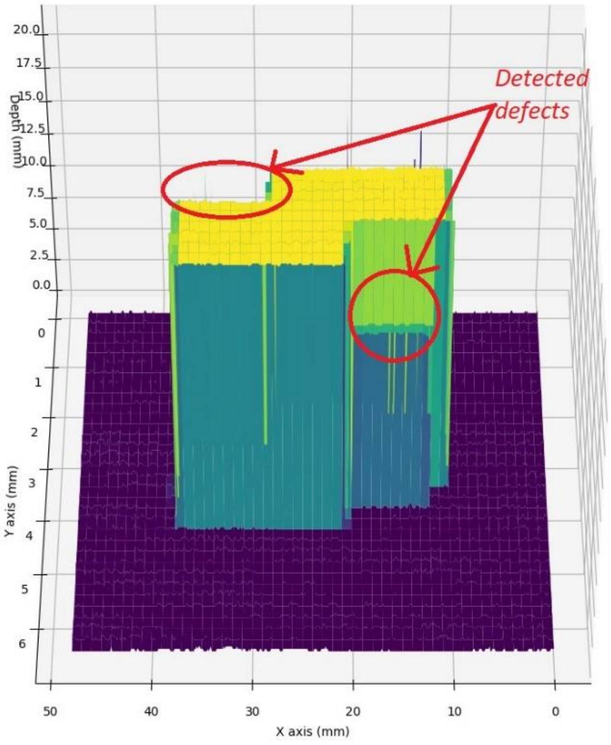


Figure 46 (b): 3D reconstruction of defect test object from posterior perspective

6. Discussion

The results from the scanning process illustrate the effectiveness and accuracy of the developed 3D scanning system. The analysis of pixel displacement and subsequent height calculations yielded promising outcomes, with measured and calculated heights showing minimal errors. This chapter will discuss the implications of these findings, the performance of the methodologies used, and potential areas for improvement.

6.1 Accuracy and Error Analysis

The accuracy of the system was evaluated through the comparison of measured and calculated heights of various objects. The errors for video capture data acquisition were relatively low, with the highest error percentage being 8.15% for the smallest measured height of 3.68 mm. This discrepancy can be attributed to several factors, such as the resolution of the camera, the precision of the Sobel X gradient edge detection, and Gaussian curve fitting methods. Smaller objects are more susceptible to such errors due to the limited number of pixels representing them, leading to a higher impact of any pixel-level inaccuracies.

For snapshot data acquisition, the errors were minimal, with the highest error percentage observed at 2.22% for a height of 2.35 mm. Notably, the error percentage decreased progressively as the height of the scanned objects increased. This trend may be due to the camera providing better resolution for objects within certain height ranges. The relatively higher error at smaller heights can be attributed to the lower resolution for smaller objects and the discrete nature of the data acquisition process, where images were captured at 1 mm intervals. The precision of this method is inherently limited by the step size and the stability of the scanning apparatus during each step.

6.2 Methodology Performance

Both data acquisition methods demonstrated strong performance in terms of generating accurate depth maps and reconstructing 3D objects. Video capture is easier to implement is lower hardware setup requirements. The continuous nature of the data acquisition allowed for a more comprehensive analysis of pixel displacement across the entire scanning process, since video can be sliced to isolate desired frames that should be analysed. However, it also introduced challenges related to repeatability, data volume and processing time.

On the other hand, the snapshot method, with its stepwise data acquisition, proved to be efficient in terms of data management and processing. The discrete approach facilitated easier reference line computation and comparison, contributing to precise height measurements. The primary limitation of this method was the potential for missed details between steps, which could affect the accuracy of the reconstruction for objects with complex geometries.

6.3 Calibration and Pixel Displacement

The calibration process, which determined the pixel-to-millimetre conversion factor, intrinsic matrix, rotation vector, and translation vectors, was crucial for the accuracy of the height calculations. Any calibration errors would propagate through the entire measurement process, emphasizing the need for meticulous calibration procedures. The Sobel X gradient method for edge detection, combined with Gaussian curve fitting, provided a robust mechanism for identifying the laser line edges. The accuracy of this approach is evident from the low error percentages observed in both data acquisition methods.

6.4 Integration and Reconstruction

The integration of depth maps and the reconstruction of 3D objects were successfully demonstrated, with Figures 41, 42 and 43 illustrating accurate representations of the scanned objects. The mesh grid generated from the height data provided a detailed geometric configuration, showcasing the system's capability to model the physical dimensions of objects accurately. The consistency between measured and calculated heights further validates the effectiveness of the integration and reconstruction process.

6.5 Potential Improvements

While the results are promising, there are areas where improvements can be made:

1. **Camera Resolution:** Increasing the camera resolution could enhance the accuracy of pixel displacement measurements, especially for smaller objects.
2. **Edge Detection Algorithms:** Exploring advanced edge detection algorithms could improve the precision of laser line edge identification, reducing errors further.
3. **Step Size in Snapshot Method:** Reducing the step size in the snapshot method could capture more details and improve the accuracy of object reconstruction.
4. **Real-Time Processing:** Enhancing the system's capability for real-time processing could make the scanning process more efficient and practical for various applications.

6.6 Conclusion

The results demonstrate that the developed 3D scanning system is capable of producing accurate height measurements and detailed 3D reconstructions with minimal errors. The methodologies employed, including camera calibration, pixel displacement measurement, and depth map integration, proved effective in achieving the desired outcomes. The system's performance could be further enhanced by future improvements in resolution, edge detection, and processing efficiency, making it a useful tool for a range of industrial applications.

7. Future scope

The study highlights the potential of integrating 3D scanning with 3D printing for space missions and industrial applications. Future research can enhance the system's capabilities and broaden its applicability in several key areas:

1. **Advanced Camera Technologies:**
    a. **Higher Resolution:** Using higher resolution cameras to improve accuracy in capturing finer details.
    b. **Multispectral Imaging:** Incorporating cameras that capture a wider range of wavelengths for better edge detection and material differentiation.
2. **Enhanced Data Acquisition:**
    a. **Dynamic Step Size:** Implementing dynamic step sizes in the snapshot method for detailed capture in complex areas.
    b. **Hybrid Methods:** Combining continuous video capture with intermittent high-resolution snapshots for comprehensive data coverage.
3. **Real-Time Processing:**
    a. Machine Learning: Developing algorithms for real-time edge detection and error correction to enhance efficiency and accuracy.
    b. Parallel Processing: Using parallel processing to speed up real-time analysis and reduce computational load.
4. **Improved Calibration:**
    a. Automated Calibration: Creating automated procedures for consistent and precise calibration to minimize errors.
    b. Environmental Adaptation: Developing techniques to adapt calibration to environmental changes like temperature and vibrations.
5. **Better Integration with Additive Manufacturing**:
    a. In-Situ Monitoring: Integrating the scanning system with 3D printers for real-time monitoring and adjustments during printing.
    b. Feedback Loops: Establishing feedback mechanisms for on-the-fly error correction during manufacturing.
6. **Application-Specific Adaptations:**
    a. Space Mission Customization: Tailoring the system for specific space mission environments and conditions.
    b. Industrial Use Cases: Exploring the system's application in various industries like automotive, aerospace, and biomedical fields.

By pursuing these directions, the 3D scanning system can become a more accurate and versatile tool, enhancing its use in space missions and a wide range of industrial applications.

References

[1] J. J. H. D. N. K. A. M. E. A. Z. S. M. W. W. B. &. B. B. R. Dunn, “3D printing in space: enabling new markets and accelerating the growth of orbital infrastructure,” *Proc. Space Manufacturing,* vol. 14, pp. 29-31, 2010.

[2] R. &. S. V. Hedayati, “3D printing for space habitats: Requirements, challenges, and recent advances,” *Aerospace,* vol. 10, no. 7, p. 653, 2023.

[3] A. W. J. W. A. G. J. W. M. P. N. C. .. &. T. X. Zocca, “Challenges in the technology development for additive manufacturing in space,” *Chinese Journal of Mechanical Engineering: Additive Manufacturing Frontiers,* vol. 1, no. 1, p. 100018, 2022.

[4] M. &. S. A. Kamran, “A comprehensive study on 3D printing technology,” *MIT International Journal of Mechanical Engineering,* vol. 6, no. 2, pp. 63-69, 2016.

[5] K. V. a. A. H. Wong, “A review of additive manufacturing,” *International Scholarly Research Notices,* p. 208760, 2012.

[6] W. &. P. L. A. Oropallo, “Ten challenges in 3D printing,” *Engineering with Computers,* pp. 135-148, 2016.

[7] P. J. Besl, “Active, optical range imaging sensors,” *Machine Vision and Applications,* pp. 127-152, 1988.

[8] M. M. P. L. W. B. M. P. T. &. M. D. Breier, “Accurate laser triangulation using a perpendicular camera setup to assess the height profile of PCBs,” in *2015 IEEE International Conference on Industrial Technology (ICIT)*, 2015.

[9] R. Wang, “3D Reconstruction Using a Linear Laser Scanner and A Camera,” in *2021 2nd International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, 2021.

[10] R. W. B. H. M. H. D. W. L. L. H. &. L. X. Yao, “A method for extracting a laser center line based on an improved grayscale center of gravity method: application on the 3D reconstruction of battery film defects,” *Applied Sciences,* p. 9831, 2023.

[11] L. C. H. T. M. L. D. C. Y. &. Y. W. Liu, “Research on 3D reconstruction technology based on laser measurement,” *Journal of the Brazilian Society of Mechanical Sciences and Engineering,* p. 297, 2023.

[12] J. A. &. R.-V. R. Muñoz-Rodríguez, “Evaluation of the light line displacement location for object shape detection,” *Journal of Modern Optics,* pp. 137-154, 2003.

[13] C. I. DAN CALLEN, “Configuring a 3D Triangulation Vision System,” Photonics Spectra, May 2017. [Online]. Available: https://www.photonics.com/Articles/Configuring_a_3D_Triangulation_Vision_System/a62061. [Accessed 5 July 2024].

[14] Panagiotis Antoniadis, “How to Convert an RGB Image to a Grayscale,” Baeldung, 18 March 2024. [Online]. Available: https://www.baeldung.com/cs/convert-rgb-to-grayscale. [Accessed 5 July 2024].

[15] O. authors, “Pinhole camera model,” openMVG , [Online]. Available: https://openmvg.readthedocs.io/en/latest/openMVG/cameras/cameras/. [Accessed 5 August 2024].

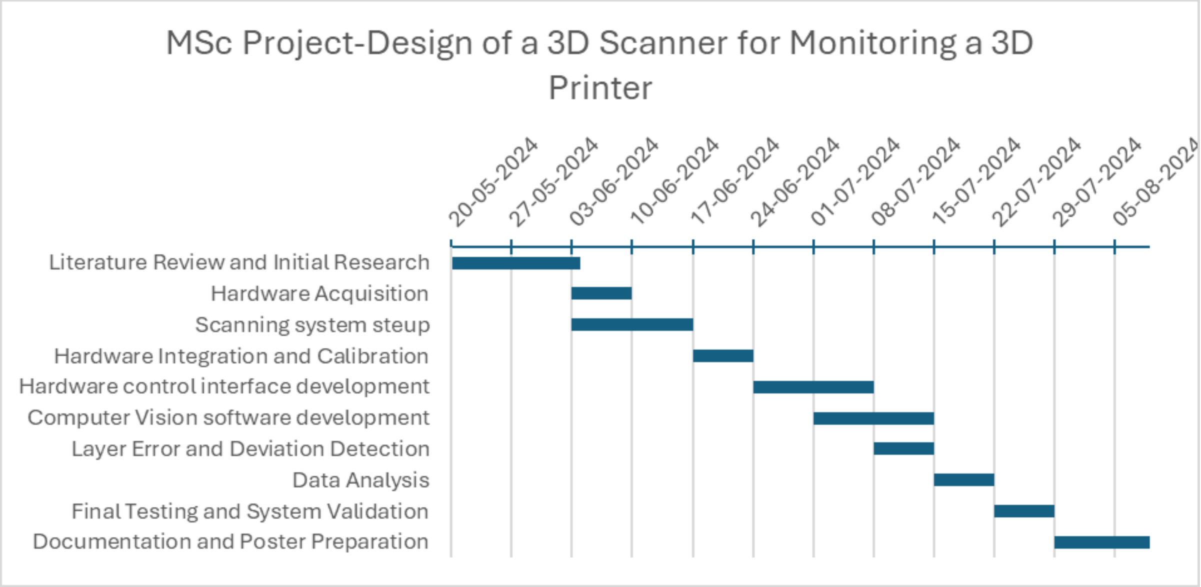
[16] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 11, pp. 1330-1334, 2000.

[17] “Evaluating the Accuracy of Single Camera Calibration,” MathWorks Inc., [Online]. Available: https://in.mathworks.com/help/vision/ug/evaluating-the-accuracy-of-single-camera-calibration.html. [Accessed 5 July 2024].

Appendix

1. GANTT Chart

The GANTT chart below depicts the projected timetable during the project's early planning stage. The main tasks and corresponding timeframes from the project study are shown.



2. Project code

The complete project code, along with the experimental data and results, can be accessed in the following GitHub repository [Link].

2.1. Imports

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import open3d as o3d
from scipy.optimize import curve_fit
from camera_calibration import getCameraCalibrationValues
import os
import glob
import warnings
```

2.2. Experiment Logic Functions

2.2.1 Function to calculate Sobel X gradient

```
def getSobelXGradient(frame):
    sobelX = cv2.Sobel(frame, cv2.CV_64F, 1, 0)
    return sobelX
```

2.2.2 Function to calculate laser outer edge

```
def getIntensityIndexOuterEdge(y_values):
    x_values = np.arange(len(y_values))

    # Define the Gaussian function
    def gaussian(x, amplitude, mean, stddev):
        return amplitude * np.exp(-((x - mean) ** 2) / (2 * stddev ** 2))

    # Initial guess for the parameters: amplitude, mean, stddev
    initial_guess = [max(y_values), np.argmax(y_values), 1]

    # Perform the curve fitting
    try:
        params, covariance = curve_fit(gaussian, x_values, y_values,
p0=initial_guess, maxfev=10000)
    except RuntimeError as e:
        print(f"{Colors.FAIL}Curve fitting failed: {e}{Colors.ENDC}")
        return np.argmax(y_values)

    # Extract the fitted parameters
    fitted_amplitude, fitted_mean, fitted_stddev = params

    # Find the index closest to the mean
    closest_index = np.argmin(np.round(np.abs(x_values - fitted_mean)))

    return closest_index
```

2.2.3 Function to calculate laser inner edge

```
def getIntensityIndexInnerEdge(y_values):
    y_values = -(y_values)
    x_values = np.arange(len(y_values))

    # Define the Gaussian function
    def gaussian(x, amplitude, mean, stddev):
        return amplitude * np.exp(-((x - mean) ** 2) / (2 * stddev ** 2))

    # Initial guess for the parameters: amplitude, mean, stddev
    initial_guess = [max(y_values), np.argmax(y_values), 1]

    # Perform the curve fitting
    try:
        params, covariance = curve_fit(gaussian, x_values, y_values, 
p0=initial_guess, maxfev=10000)
    except RuntimeError as e:
        print(f"{Colors.FAIL}Curve fitting failed: {e}{Colors.ENDC}")
        return np.argmax(y_values)

    # Extract the fitted parameters
    fitted_amplitude, fitted_mean, fitted_stddev = params

    # Find the index closest to the mean
    closest_index = np.argmin(np.round(np.abs(x_values - fitted_mean)))

    return closest_index
```

2.2.4 Function to calculate depth map for a single frame

```
def getDepthMap(frame, referenceLine, heightValues):
    depthMap = np.zeros(frame.shape)
    for i in range(frame.shape[0]):
        # print(f"{Colors.OKBLUE}({i}, {referenceLine[i]}, 
{heightValues[i]})")
        depthMap[i][int(referenceLine[i])] = heightValues[i]
    return depthMap
```

2.2.5 Function to calculate camera parameters

```
def getCameraCalibrationValues():
    ################ FIND CHESSBOARD CORNERS - OBJECT POINTS AND IMAGE POINTS 
#############################

    chessboardSize = (5, 5)
    frameSize = (640,480)

    folder = './ChessBoard/calibration2'


    # termination criteria
    criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)


    # prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
    objp = np.zeros((chessboardSize[0] * chessboardSize[1], 3), np.float32)
    objp[:,:2] = np.mgrid[0:chessboardSize[0],0:chessboardSize[1]].T.reshape(-
1,2)

    size_of_chessboard_squares_mm = 5
    objp = objp * size_of_chessboard_squares_mm
    # Arrays to store object points and image points from all the images.
    objpoints = [] # 3d point in real world space
    imgpoints = [] # 2d points in image plane.


    images = glob.glob(folder + '/*.jpg')
    pixels_per_mm = []

    count = 0
    for image in images:
        img = cv.imread(image)
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

        # Find the chess board corners
        ret, corners = cv.findChessboardCorners(gray, chessboardSize, None)

        # If found, add object points, image points (after refining them)
        if ret == True:

            objpoints.append(objp)
            corners2 = cv.cornerSubPix(gray, corners, (11,11), (-1,-1),
criteria)
            imgpoints.append(corners)

        # Draw and display the corners
        cv.drawChessboardCorners(img, chessboardSize, corners2, ret)
        cv.imshow('img', img)
        # cv.imwrite(folder + '/test' + str(count) + '.jpg', img)
        cv.waitKey(1000)

        # Calculate the average distance between adjacent corners in pixels
        total_distance = 0
        for i in range(chessboardSize[1]):
            for j in range(chessboardSize[0] - 1):
                total_distance += np.linalg.norm(corners2[i *
chessboardSize[0] + j] - corners2[i * chessboardSize[0] + j + 1])

        avg_distance_in_pixels = total_distance / (chessboardSize[1] *
(chessboardSize[0] - 1))

        # Calculate pixels per mm
        pixels_per_mm.append( avg_distance_in_pixels /
size_of_chessboard_squares_mm)
        count += 1

    else:
        print("Chessboard not found")
    cv.destroyAllWindows()
    pixels_per_mm = np.mean(pixels_per_mm)
    print(f"1 mm = {pixels_per_mm} pixels")



    ############## CALIBRATION
#######################################################

    ret, cameraMatrix, dist, rvecs, tvecs = cv.calibrateCamera(objpoints,
imgpoints, frameSize, None, None)

    print("Camera calibrated : \n", ret)
    print("Camera Matrix : \n", cameraMatrix)
    print("dist : \n", dist)
    print("Rotation Vectors : \n", rvecs)
    print("Translation Vectors : \n", tvecs)
    # Save the camera calibration result for later use (we won't worry about
rvecs / tvecs)
    pickle.dump((cameraMatrix, dist), open( "calibration.pkl", "wb" ))
    pickle.dump(cameraMatrix, open( "cameraMatrix.pkl", "wb" ))
    pickle.dump(dist, open( "dist.pkl", "wb" ))

    return pixels_per_mm, cameraMatrix, dist, rvecs, tvecs
```

2.3. Utility Functions

2.3.1 Function to draw frame edge trace

```
def drawEdgeTrace(objectName, edgeTrace, intensityInnerEdge,
intensityOuterEdge, meanIndex, referenceLine, count):
    # Convert the Sobel gradient image to absolute values to fit into the 8-
bit range
    edgeTraceAbs = np.abs(edgeTrace).astype(np.uint8)

    # Convert the grayscale edgeTrace to BGR color image
    edgeTraceColor = cv2.cvtColor(edgeTraceAbs, cv2.COLOR_GRAY2BGR)

    # Define the output folder and create it if it doesn't exist
    output_folder = f'./edge_trace/{objectName}/'
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)


    # Plot the intensity points on the edgeTraceColor image
    if intensityInnerEdge is not None:
        for idx, val in enumerate(intensityInnerEdge):
            cv2.circle(edgeTraceColor, (int(val), int(idx)), 1, (255, 0, 0), -
1)  # Blue dot with radius 1

    if intensityOuterEdge is not None:
        for idx, val in enumerate(intensityOuterEdge):
            cv2.circle(edgeTraceColor, (int(val), int(idx)), 1, (0, 0, 255), -
1)  # Red dot with radius 1
    if meanIndex is not None:
        for idx, val in enumerate(meanIndex):
            cv2.circle(edgeTraceColor, (int(val), int(idx)), 1, (0, 255, 255),
-1)  # Yellow dot with radius 1
    if referenceLine is not None:
        for idx, val in enumerate(referenceLine):
            cv2.circle(edgeTraceColor, (int(val), int(idx)), 1, (0, 255, 0), -
1)  # Green dot with radius 1

    # Save the image with the plotted points
    cv2.imwrite(f"./edge_trace/{objectName}/frame{count}.jpg", edgeTraceColor)
```

2.3.2 Function to visualise combined depth maps

```
def visualize_depth_map_3d(depth_map):
    fig = plt.figure()
    fig.set_size_inches(10, 10)
    ax = fig.add_subplot(111, projection='3d')

    x = np.linspace(0, depth_map.shape[1] / 10, depth_map.shape[1])
    y = np.linspace(0, depth_map.shape[0] / 10, depth_map.shape[0])
    x, y = np.meshgrid(x, y)

    ax.plot_surface(x, y, depth_map, cmap='viridis')

    ax.set_xlabel('X axis (mm)', fontsize=20)
    ax.set_ylabel('Y axis (mm)', fontsize=20)
    ax.set_zlabel('Depth (mm)', fontsize=20)

    # Increase font size of axis ticks
    ax.tick_params(axis='both', which='major', labelsize=16)
    ax.tick_params(axis='z', which='major', labelsize=16)

    plt.tight_layout()
    plt.show()
```

2.3.3 Function to visualise combined depth maps with elevation and azimuth angle

```
def visualize_depth_map_3d_angle(depth_map, elev=30, azim=30):
    fig = plt.figure()
    fig.set_size_inches(10, 10)
    ax = fig.add_subplot(111, projection='3d')
    x = np.linspace(0, depth_map.shape[1] / 10, depth_map.shape[1])
    y = np.linspace(0, depth_map.shape[0] / 10, depth_map.shape[0])
    x, y = np.meshgrid(x, y)

    ax.plot_surface(x, y, depth_map, cmap='viridis')

    # Create a plane at z = 20 mm
    z_plane = np.full_like(depth_map, 20)
    ax.plot_surface(x, y, z_plane, color='red', alpha=0.5)

    # Add a label on the red plane
    mid_x = (x.max() + x.min()) / 2
    mid_y = (y.max() + y.min()) / 2
    ax.text(mid_x, mid_y, 22, "Height given in CAD design", color='black',
fontsize=20, ha='center', va='center')

    ax.set_xlabel('X axis (mm)', fontsize=20)
    ax.set_ylabel('Y axis (mm)', fontsize=20)
    ax.set_zlabel('Depth (mm)', fontsize=20)

    # Increase font size of axis ticks
    ax.tick_params(axis='both', which='major', labelsize=16)
    ax.tick_params(axis='z', which='major', labelsize=16)

    # Set the viewing angle
    ax.view_init(elev=elev, azim=azim)

    plt.tight_layout()
    plt.show()
```

2.3.4 Function to draw depth maps

```
def drawDepthMapFrame(depthMapFrame, objectName, count):
    output_folder = f'./depth_maps/{objectName}/'
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Create a figure and axis
    fig, ax = plt.subplots()
    # Display the depth map
    cax = ax.imshow(depthMapFrame)
    # Add a color bar
    fig.colorbar(cax, ax=ax)
    # Save the figure
    plt.savefig(f"{output_folder}/frame{count}.jpg")
    # Close the figure to free up memory
    plt.close(fig)
```

2.4. Experiment Code

2.4.1 Localising laser reference line before print job

```
# Define the object name and extract frames
objectName = "05.08.24.Reference_Scan"
# objectName = "report"
# objectFrames = extract_frames(objectName, ".mp4")

meanPixelDisplacement = []
count = 0

heightList = []

# Define the translation distance between frames in mm
translation_distance = 0.75

objectFrames = glob.glob(f"./measured_steps/{objectName}/*.jpg")


referenceLineList = []
# Loop through the frames
for image in objectFrames:
    print(f"Processing frame {image}...")
    frame = cv2.imread(image, cv2.IMREAD_GRAYSCALE)

    # Get the Sobel X gradient
    edgeTrace = getSobelXGradient(frame)
    # Calculate intensity index along the laser edge trace
    intensityInnerEdge = np.round(np.array([getIntensityIndexInnerEdge(i) for i in edgeTrace]))
    intensityOuterEdge = np.round(np.array([getIntensityIndexOuterEdge(i) for i in edgeTrace]))
    meanIndex = np.round((intensityInnerEdge + intensityOuterEdge) / 2)

    # maxIntensity = np.argmax(edgeTrace, axis=1)


    referenceLine = meanIndex
    # referenceLine = maxIntensity


    drawEdgeTrace(objectName, edgeTrace, intensityInnerEdge, intensityOuterEdge, meanIndex, referenceLine, count)
    # drawEdgeTrace(objectName, edgeTrace, None, None, None, None, count)

    referenceLineList.append(referenceLine)

    count += 1
    # break  # Remove this break to process all frames
```

```
referenceLineList = np.array(referenceLineList)
referenceLineList.shape
```

2.4.2 Print object scanning procedure

```
# Define the object name and extract frames
objectName = "test1"

meanPixelDisplacement = []
count = 0

heightList = []
all_point_clouds = []

objectFrames = glob.glob(f"./measured_steps/{objectName}/*.jpg")

depthMap = []

# Loop through the frames
for image in objectFrames:
    print(f"Processing frame {image}...")
    frame = cv2.imread(image, cv2.IMREAD_GRAYSCALE)
    cv2.imwrite(f"./edge_trace/{objectName}/gray_{count}.jpg", frame)

    # Get the Sobel X gradient
    edgeTrace = getSobelXGradient(frame)

    edgeMax = np.max(edgeTrace, axis=1)

    c = pixels_per_mm
    referenceLine = referenceLineList[count]

    if np.any(edgeMax < 30):
        print(f"{Colors.FAIL}No edge detected in frame {count}{Colors.ENDC}")
        heights = depthMap[-1]
        depthMap.append(heights)
        drawEdgeTrace(objectName, edgeTrace, intensityInnerEdge=None,
intensityOuterEdge=None, meanIndex=None, referenceLine=None, count=count)
        count += 1
        continue

    # Calculate intensity index along the laser edge trace
    intensityInnerEdge = np.round(np.array([getIntensityIndexInnerEdge(i) for
i in edgeTrace]))
    intensityOuterEdge = np.round(np.array([getIntensityIndexOuterEdge(i) for
i in edgeTrace]))
    meanIndex = np.zeros(len(intensityInnerEdge))
    for i in range(len(intensityInnerEdge)):
        if abs(intensityInnerEdge[i] - intensityOuterEdge[i]) > 30:
            print(f"{Colors.FAIL}No edge detected in frame
{count}{Colors.ENDC}")
            meanIndex[i] = referenceLine[i]
        else:
            meanIndex[i] = np.round((intensityInnerEdge[i] +
intensityOuterEdge[i]) / 2)
            if meanIndex[i] > referenceLine[i]:
                meanIndex[i] = referenceLine[i]


    drawEdgeTrace(objectName, edgeTrace, intensityInnerEdge,
intensityOuterEdge, meanIndex, referenceLine, count)

    if referenceLine is None:
        print(f"{Colors.FAIL}Reference line not found{Colors.ENDC}")
        heights = depthMap[-1]
        referenceLine = referenceLineList[-1]
    else:
        pixelDisplacement = abs(meanIndex - referenceLine)
        c = pixels_per_mm
        s = pixelDisplacement / c
        heights = 80 * s / (37.3 + s)

    depthMap.append(heights)

    # Depth map for individual frames
    depthMapFrame = getDepthMap(frame, referenceLine, heights)
    drawDepthMapFrame(depthMapFrame, objectName, count)

    count += 1
    # break  # Remove this break to process all frames
```

2.4.3 Visualise print object 3D geometry

```
depthMapCopy = np.array(depthMap)
depthMapCopy.shape
np.savetxt("output.csv", depthMapCopy, delimiter=",")
visualize_depth_map_3d(depthMapCopy.T)
visualize_depth_map_3d_angle(depthMapCopy, elev=0, azim=120)  # Change viewing
angle
```