Towards the Development of Multi-View Reconstruction System for Paleontological Fossils

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Yichen Xie

January 2024

Towards the Development of Multi-View Reconstruction System for Paleontological Fossils

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Yichen Xie

January 2024

Approved:

Department Chair Signature

Date

University ID: N13905325
Net ID: yx2606

Approved by the Guidance Committee:

Major: Computer Science

Claudio T. Silva
Institute Professor
NYU Tandon School of Engineering

Date

Daniele Panozzo
Associate Professor
NYU Courant Institute of Mathematical Sciences

Date

Qi Sun
Assistant Professor
NYU Tandon School of Engineering

Date

Vita

Yichen Xie was born on the 19th of June 1997, in Tianjin, China. He got his Bachelor's degree in both Artificial Intelligence Science and Economics at Peking University in 2019 and enrolled in a Master's Program in Computer Science at New York University in 2021. Throughout his undergraduate and graduate school, he interned as a Research Engineer at Megvii Research and worked as a Research Assistant at Peking University.

This work was completed at the Visualization Imaging and Data Analysis Center (VIDA) Lab at New York University in January 2024.

Acknowledgements

During my master's studies at New York University, I had the great pleasure of collaborating with and learning from so many wonderful people. First, I would like to thank my supervisors, Prof. Claudio Silva, Prof. Qi Sun, and Prof. Daniele Panozzo, who provided me with exceptional guidance and support throughout the whole PaleoScan project, and also Dr. Yurii Piadyk and Mr. Joao Rulff, who give me detailed instructions and help me overcome difficulties. Besides, I also would like to thank my girlfriend Xiaozheng, who gave me hope and showed me the light during the lowest point of my life. I also would like to thank my parents and my family, who supported me on the way to studying aboard. I would also like to express my gratitude to all the friends around me, who cured my loneliness. To all of them, I dedicate this thesis. I hope this thesis will be a farewell to the past and a new start for the future.

This thesis is based upon work funded by the National Science Foundation CNS-1828576.

Yichen Xie

January 2024

ABSTRACT

Towards the Development of Multi-View Reconstruction System for Paleontological Fossils

by

Yichen Xie

Advisor: Prof. Claudio T. Silva, Ph.D.

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (Computer Science)

January 2024

Digitization is an innovative and invaluable technique for preserving and distributing ancient fossils. It alleviates the financial burden on museums while providing a flexible means for fossil analysis and exhibition. Generally, digitization involves extracting physical information from fossils, organizing multi-modal data, and constructing interactive visualization systems. Among these steps, the reconstruction of fossils plays a pivotal role as it provides paleontology researchers with realistic visual information. Although significant progress has been made in achieving accurate fossil reconstruction over the past decades, areas remain for improvement, such as accuracy, robustness, and cost-effectiveness.

This thesis focuses on the task of multi-view reconstruction of fossil objects and presents the development of PaleoScan, a robust reconstruction algorithm working on a single camera-based scanner. The proposed pipeline performs corner detection, camera calibration, point cloud generation, and surface recovery on the acquired image dataset, exhibiting robustness and efficiency in various scenarios. We present all the details in our system settings and conduct multiple visualized experiments to demonstrate its effectiveness: fast and robust detection and calibration process is performed, remarkable accuracy in the multi-view matching and depth estimation is achieved (within 1 mm), and models with rich details (meshes with more than three million triangles) that resemble real objects are generated. PaleoScan is a comprehensive end-to-end framework that could be broadly applicable to paleontology studies, potentially enhancing the quality and reliability of on-site digitization for all kinds of fossils.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Three-dimensional reconstruction is one of the most critical techniques in fossil digitization. A well-designed reconstruction pipeline can generate digital records with sensors, extract valuable visual information, create detailed three-dimensional models, and provide visualization interfaces for paleontologists and museum visitors. This revolutionary approach has transformed paleontology studies by enabling researchers to gain insightful information from fossils without causing physical damage and allowing for modifications and annotations in a virtual space.

Various technologies have been introduced and developed to reconstruct fossils, including Serial Grinding, Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Laser Scanning, etc. These methods can produce high-resolution maps of fossil objects but have notable limitations. Grinding-based methods result in irreversible physical damage to the fossils, while radiation-based methods require expensive and non-portable equipment that is difficult to deploy near excavation sites. Therefore, paleontology digitization faces the problem of high transportation and preservation costs.

Fortunately, with the advancements in Computer Vision, image-based reconstruction methods have gained popularity in fossil digitization. These methods offer significant advantages. Firstly, images can be easily captured using commonly available cameras, making the data acquisition process more flexible. Secondly, they can leverage existing efficient image processing algorithms. However, images obtained through cameras often suffer from lower quality than those obtained using complex radiation scanners due to uncontrollable ambient lighting and lens distortions. Consequently, computer scientists face the challenge of developing robust and efficient algorithms to overcome these shortcomings and recover as many physical details of the fossils as possible.

This challenge motivated us to construct a low-cost scanner utilizing a single-lens camera. We captured multiple images of a simulated fossil object from various viewpoints and developed an algorithm to generate point clouds and meshes for the object. We aim to design an automated reconstruction pipeline that exhibits robustness and accuracy that researchers can quickly learn and utilize in paleontology studies.

1.2 Related Works

In recent years, various methods have been proposed for digitizing fossils. One early method by Sollas et al. [1], introduced in 1904, is physical tomography, which involves obtaining images of thin serial slices of a fossil. However, physical tomography is time-consuming and destructive, as it alters the original state of the fossils.

To address these shortcomings, non-destructive methods such as radiography scanning and camera image integration have gained attention. X-ray computed tomography (CT) (e.g., Donoghue et al. [43], Wind et al. [11]) works by capturing multiple projection images at different angles to detect the inner structure of the specimen using a penetrating beam. These images, arranged in a predetermined pattern perpendicular to the rotation axis, map X-ray attenuation through the specimen [9]. Neutron radiography by Winkler et al. [45] has also been used as an alternative to classical X-ray imaging. Schiffbauer et al. [37] introduce focused ion beam electron microscopy (FIB-EM). Magnetic resonance imaging (MRI) by Ziegler et al. [47] offers another approach to acquiring tomography maps, providing the ability to depict soft tissue in zoological specimens. Additionally, Bates et al. [3] uses light detection and range (LiDAR) images to generate 3D models of dinosaur fossil trackways. These radiography imaging methods provide non-destructive means for fossil reconstruction and analysis, particularly valuable for unique and rare fossil materials. However, they all require expensive and non-portable scanning facilities.

Reconstruction using ordinary camera images has become an attractive option to address the need for cost reduction and adaptability to complex environments. Advanced image processing algorithms can extract details such as size and texture from various materials, enabling the generation of simulated models using graphical techniques. For instance, Falkingham et al. [15] acquired images with a megapixel camera, determined camera positions with *Bundler* [40], generated sparse and dense point clouds using multi-view stereo (MVS) software, and produced three-dimensional meshes using *MeshLab* [14]. COLMAP by Schonberger et al. [38, 39] provides a general-purpose end-to-end pipeline for scene and object reconstruction from unstructured imagery. Jaiswal et al. [23] utilized an RGB-D (Kinect-style) camera for 3D object modeling. Compared with the methods above, our pipeline focuses explicitly on small-scale fossil object reconstruction and does not use any sensors to acquire depth information directly.

1.3 Outline

The following chapters are organized based on the structure of the processing pipeline in our system. In Chapter 2, we provide an overview of the design and settings of our camera-based scanner and the management of the acquired image dataset. Chapter 3 focuses on the ChArUco board and the detection algorithm, a fundamental component for the subsequent calibration stage.

Moving forward, Chapter 4 delves into the calibration methods, covering intrinsic and extrinsic parameters calibration processes. Chapter 5 presents the multi-view matching approach and the generation of dense point clouds for our fossil objects. Chapter 6 introduces the surface-recovering algorithm and showcases relevant experiments demonstrating its effectiveness in producing realistic and refined surfaces. We also provide the results of experiments on real fossil objects in Appendix Chapter A and discuss the problem of adaptation from artificial to real fossil models.

Finally, in Chapter 7, we conclude the thesis by summarizing the key findings and contributions of our research. Additionally, we provide an outlook on potential challenges and areas for future work, highlighting opportunities for further improvement and development in the field of fossil digitization.

Chapter 2

Camera Scanner

This chapter introduces the design and settings of our camera-based scanner[1], and the pre-processing and arrangement of the obtained image dataset.

2.1 Camera-based Scanner

In most reconstruction tasks, capturing images using a camera is the initial step. However, there are variations in the categories and settings of cameras and scanners. For example, [15] use an Olympus E-500 megapixel camera to acquire images at 15-degree intervals, with approximately 50% overlap between consecutive images.

In our system, we have designed a scanner consisting of a cubic iron frame with a track that can be shifted along an axis perpendicular to itself. The camera is attached to this track and can move along the track (See Fig 2.1). Consequently, we can precisely control the camera's position by controlling the track and the camera's pedestal. It is worth noting that since the track only moves within a

[1]The scanner in our project is designed and built by Dr. Yurii Piadyk at New York University and Mr. Otávio José De Freitas Gomes at the Federal University of Ceará.

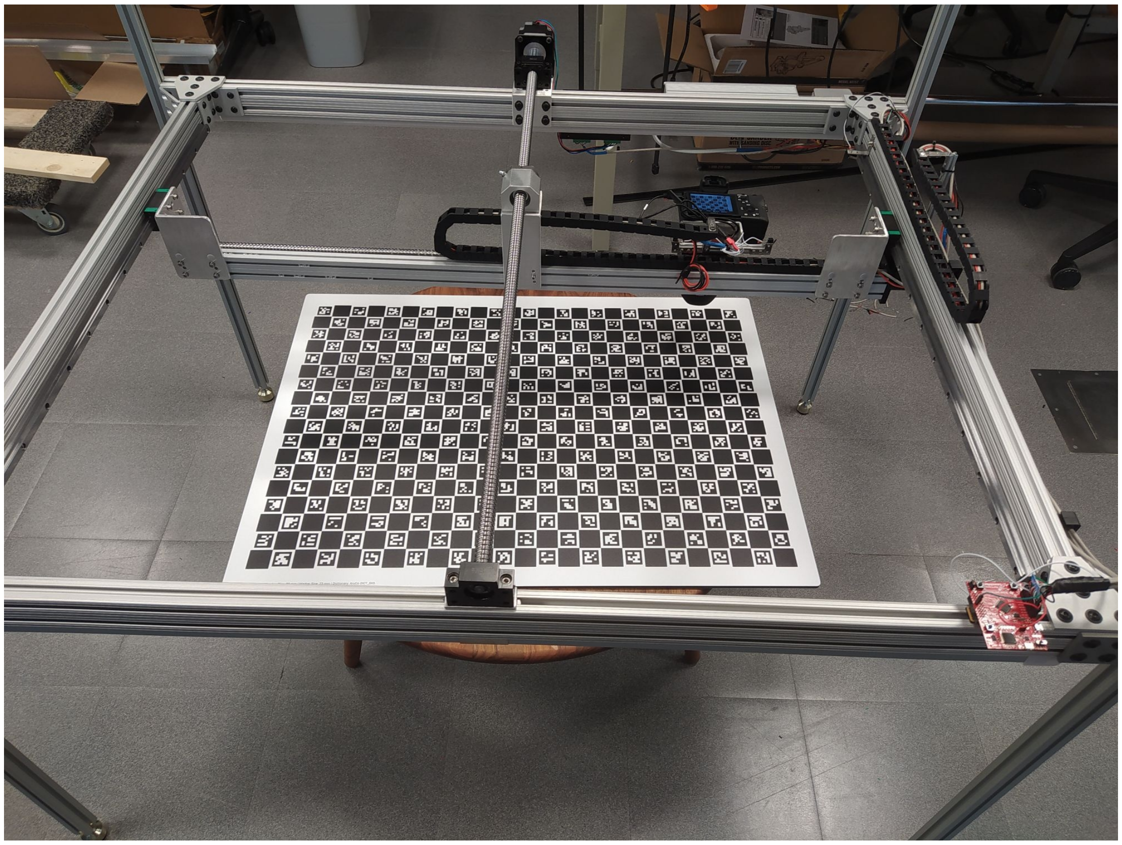


Figure 2.1: The framework of our camera scanner. The two perpendicular tracks provide two degrees of freedom for the camera movement. The object is placed within the steel frame.

horizontal plane, we can ensure that the camera position remains restricted to a fixed height.

Our setup employed a SONY megapixel camera, whose shutter is connected to a microcontroller (See Fig 2.2). The microcontroller runs a controlling software with a user-friendly interface, allowing users to interact with the camera and specify the desired image capture pattern. The camera can capture images in either RAW (uncompressed) or JPG (compressed) format. Our experiments utilized RAW format images to prioritize accuracy and convenience.

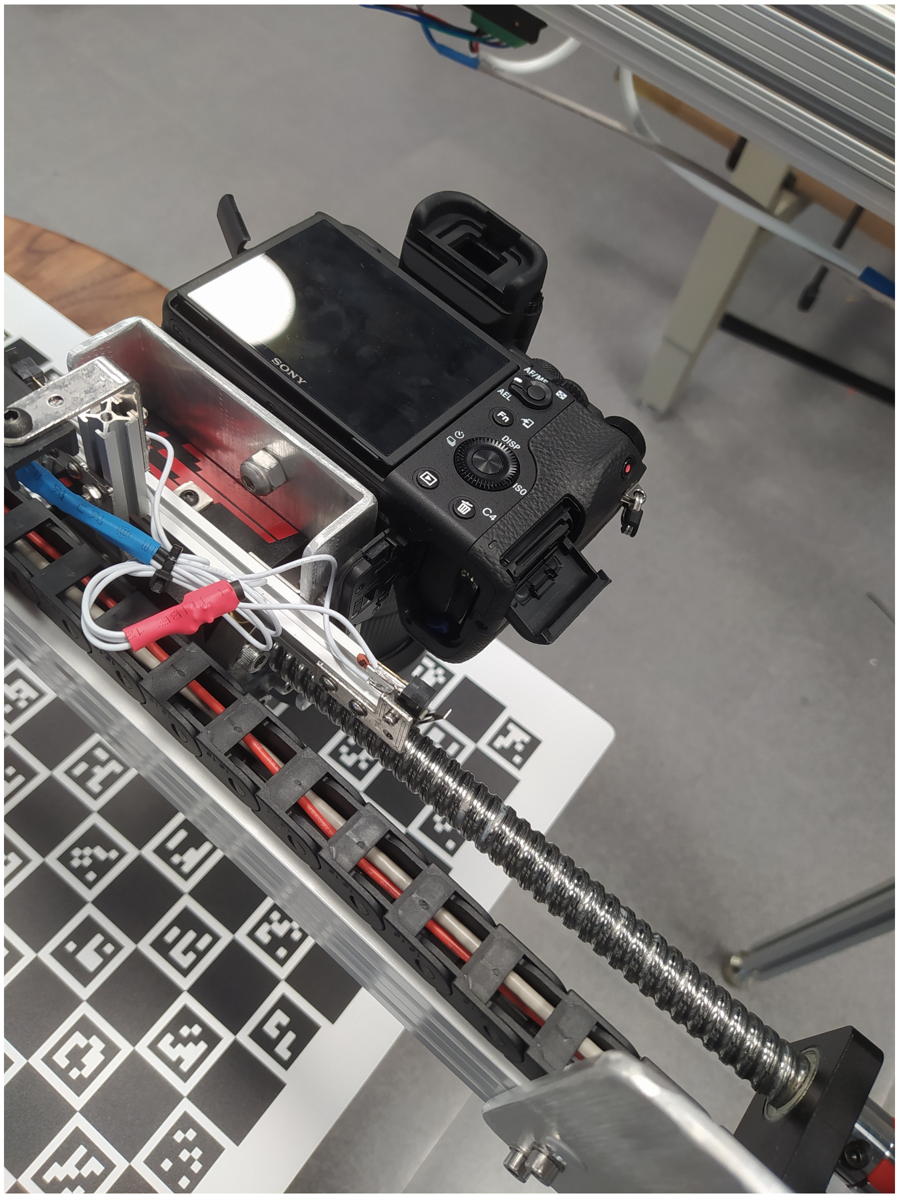


Figure 2.2: The single-lens reflex camera in our scanner. The camera is connected to a micro-controller with signal wires and takes photos automatically in designed patterns. It moves along predetermined paths in our experiments.

2.2 Dataset Arrangement

To obtain a structured image dataset that can be quickly processed in subsequent stages, we devised a specific movement pattern for the camera, maintaining fixed vertical and horizontal distances between images. The camera initiates its movement from the origin of the scanner coordinate system, which is situated at the bottom right corner, then follows a serpentine path until it reaches the top left or right corner. The final stop position is determined by the vertical movement being even or odd. Additionally, we incorporate an offset from the scanning range's edge to concentrate the captured images on the scanned object (See Fig 2.3).

After capturing images along the predefined path, we obtain a series of images organized based on the camera's movement sequence. However, this unstructured dataset is not ideal for the following calibration and multi-view matching tasks, as it does not directly provide information about the adjacency between images. It is quite inefficient to locate an image above or below another image. Therefore, we reorganized the images into a new hashing data structure called Image Grid to solve this problem: Each image is assigned a pair of indices in this structure, and a dictionary employs the two-dimensional indices as keys to access the corresponding images. The Image Grid greatly simplifies code implementation for other tasks and reduces the time required to search adjacent images in a large dataset.

Furthermore, we translated from RAW images (Bayer Pattern) to JPG format images and applied white balancing to ensure color calibration for all the images. This process can be time-consuming, particularly for large datasets, due to the I/O bottleneck of the disk. To mitigate this problem, we implemented the translation algorithm once at the initial stage of our experiment and then stored the results as a compressed cache file in our system. Thus we can avoid repetitive translations.

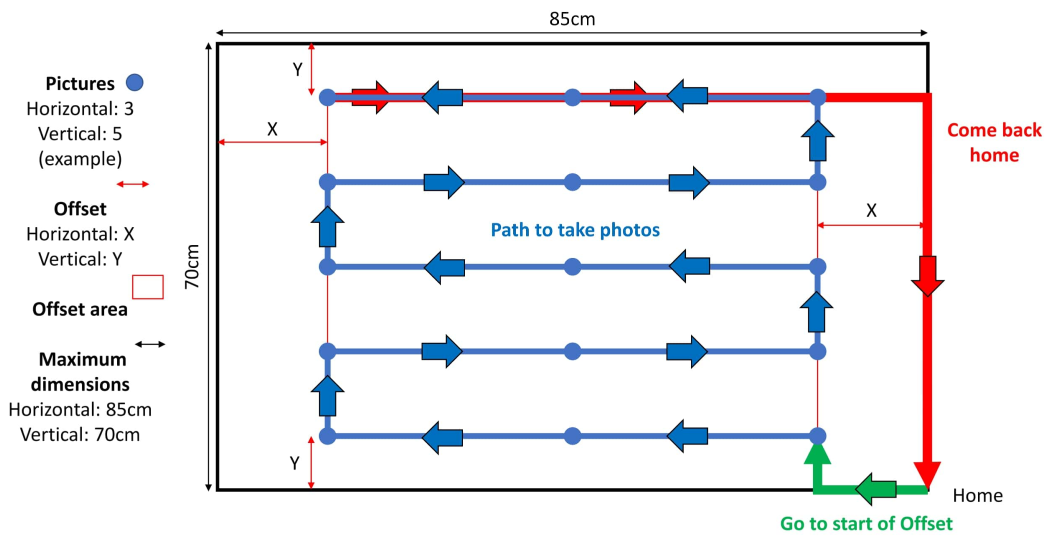


Figure 2.3: Visualization of camera movement. The camera always starts its movement from the bottom left corner. We limit the movement range of the camera by setting offsets (X and Y), allowing it to capture object information more effectively. We restrict the camera to moving only a short distance each step along either the horizontal or vertical direction.

In our experimental setup, we generated five distinct image datasets, namely Flat-1, Flat-2, Irregular-1, Irregular-2, and Box. Each dataset represents a different object configuration.

Flat-1 and Flat-2 correspond to the image sets capturing an object with a perfectly flat surface. These datasets focus on objects devoid of any irregularities or folds. Irregular-1 and Irregular-2 are image sets that feature objects with folds and deformations. These datasets highlight the challenges posed by objects with non-uniform surfaces. Lastly, the Box dataset comprises a series of images capturing a brown rectangular box. This dataset is designed to provide a natural, standardized, and consistent object configuration for experimentation.

Chapter 3

Feature Detection

This chapter covers the concepts and algorithms of image feature detection in our project. First, we briefly discuss the categories of image features and introduce ChArUco Board, a useful tool for generating rich image features. Then we present our corner detection algorithm and evaluate our experiment results.

3.1 Image Features

Image features encompass specific properties of an image, such as patterns, textures, positions, etc., which can be used to distinguish differences and find matching parts between images. These features include points, lines, regions, and even encoded codes derived from the original features. In the context of reconstruction tasks, our primary focus lies on extracting point features from images. Point features are essential as they provide location information about the camera and are the basic units for point cloud generation.

A wide range of methods have been proposed to extract point features from images. Harris Corner Detector [18] is one of the most well-known algorithms in Computer Vision. It is simple and efficient but susceptible to noise. To improve robustness, various alternative methods have been introduced, including SIFT [30], SURF [4], BRIEF [22], FAST [42], ORB [35]. These methods aim to maintain invariance to rotation and translation transformations but often come with increased time consumption. Consequently, researchers often face trade-offs between accuracy and efficiency when selecting the most suitable method based on their project requirements.

Another approach to introducing image features is adding fiducial markers into the image. A fiducial marker is an object deliberately placed within the field of view of an imaging system to serve as a reference or a measurement tool. Fiducial markers like ARToolKit, ARTag, AprilTag, and ArUco (See Fig 3.1) can be easily detected using simple detectors and template matching algorithms. Compared to extracting features directly from objects or scenes, this approach requires less time and can even achieve higher accuracy, particularly in texture-less conditions.

Our project aims to extract point features efficiently and accurately for camera calibration. To reach this goal, we utilize the ChArUco Board, which enriches the background with image features, allowing us to achieve comparable accuracy with simple detectors. The details of the ChArUco Board will be presented in the subsequent section.

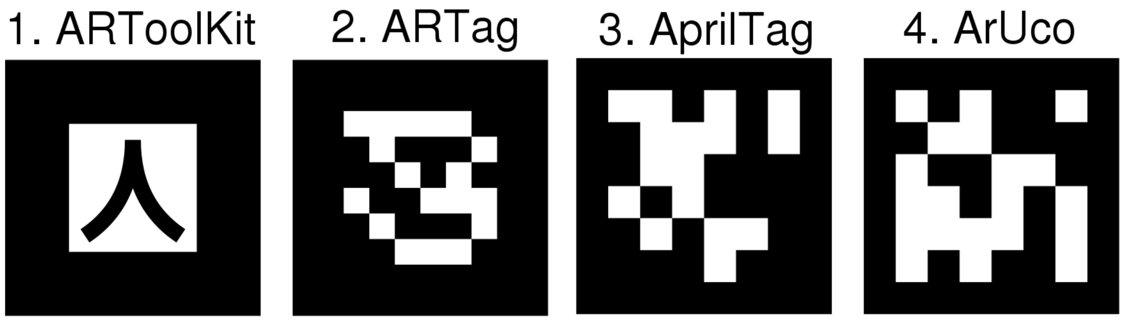


Figure 3.1: Example of fiducial markers. Each kind of markers has its unique coding to achieve differentiation within the group.

3.2 ChArUco Board

The ChArUco Board combines the characteristics of a chessboard (or checkerboard) and fiducial markers (ArUco markers) (See Fig 3.2). Chessboard corners are highly suitable as point features for detection due to the distinct contrast between the black and white grid squares. They have been widely employed in camera calibration tasks. However, chessboard corners become indistinguishable when the board rotates by 90 degrees or upside-down.

To address this limitation, the ChArUco Board incorporates ArUco markers into the chessboard. Each white grid in the chessboard is assigned a unique ArUco code, introducing differences between them. The presence of ArUco markers provides valuable information regarding the rotation of the chessboard, thus ensuring the accuracy and correctness of corner matching between images and enhancing the robustness of the overall feature detection process.

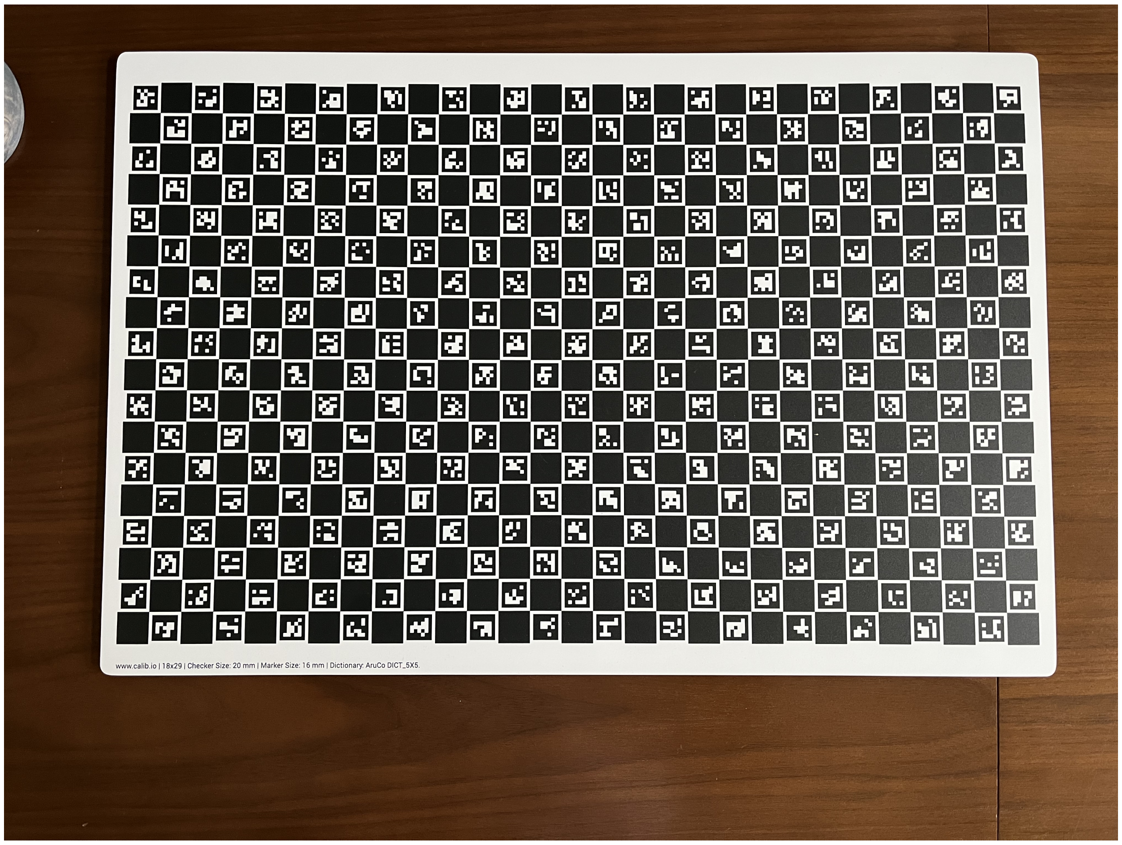

www.calib.io | 18x29 | Checker Size: 20 mm | Marker Size: 16 mm | Dictionary: AruCo DICT_5X5

Figure 3.2: An example ChArUco Board. Every white square in the chessboard is assigned with an ArUco marker, thus encoded with a different index.

3.3 Algorithm

Our detection method is developed based on the ChArUco detection package in OpenCV, taking algorithm implementations from [27] and [17] as references. The detection algorithm generates a matching template for the assigned board and marker sizes, subsequently identifying and matching markers within the image.

Initially, the algorithm employs an edge detection algorithm to extract the contours of the markers. The inner region of these contours is then analyzed to extract the internal code of each marker. After that, marker identification and error correction are performed to ensure accurate detection.

As an improvement to the original detection algorithm, we introduce a re-scale coefficient that allows for the adjustment of input image resolution. This adjustment helps reduce noise and time consumption. Additionally, we have imported the *joblib* [24] package to enhance the detection algorithm's speed, enabling parallel execution of multiple detection functions.

3.4 Evaluation

We evaluated our algorithm's performance by generating images with color markers placed on the detected corners. We then calculated the average time cost for each image set.

In Figure 3.3, we provide an example of corner detection visualization. The green lines represent the detected contours of the ArUco markers, while the blue markers indicate the corresponding codes of these markers. Additionally, the corners of the white grids, which serve as the feature points used in the calibration process, are also detected and marked. This visualization allows us to assess the correctness and accuracy of the detected corners in the images.

However, relying solely on visual evaluation may be partially convincing. Therefore, we also conducted quantitative validation after the calibration process. The validation results will be presented in Section 4.6, providing a more comprehensive assessment of the algorithm's performance.

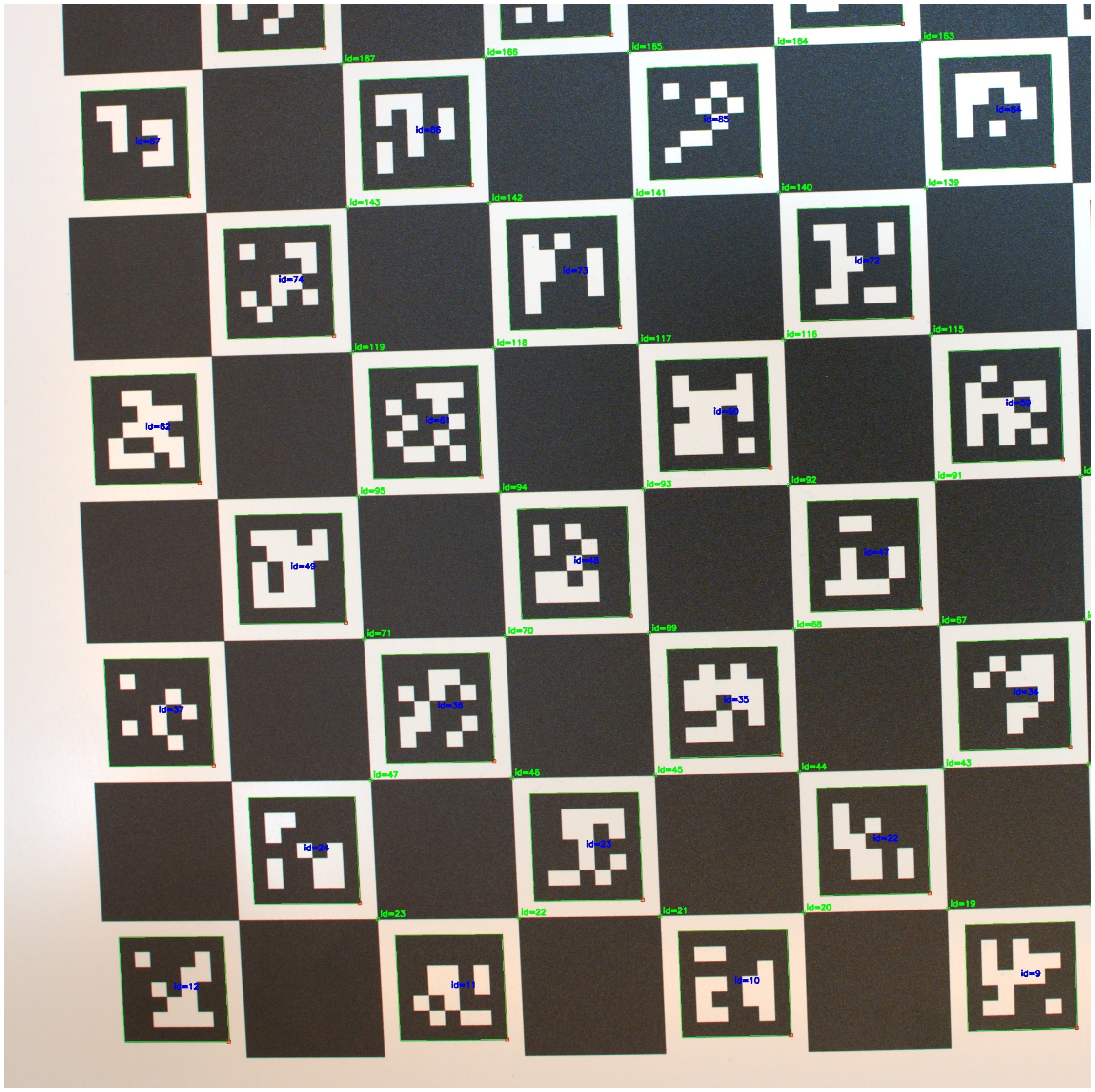


Figure 3.3: Visualization of detected markers. Blue Words: ID of ArUco marker. Green Lines: Edges of Aruco markers. Green Point and Words: Chessboard corners and IDs.

Table 3.1: Time cost of corner detection for different image sets. The detection algorithm operates at a speed of approximately 1.5s per image.

Image Set	Image Quantity (Object + Calibration)	Time (second)
Flat-1	30 + 35	46.3 + 50.1
Flat-2	30 + 35	47.6 + 51.2
Irregular-1	30 + 35	46.1 + 52.1
Irregular-2	30 + 35	45.7 + 48.2
Box	200 + 35	272.1 + 48.0

Table 3.1 presents the time cost of the detection process for each image set. On average, our detection algorithm operates at a speed of 1.5 seconds per image in small image sets (consisting of less than 35 images). Notably, the time cost is not significantly influenced by the number of detected corners. Moreover, in larger image sets, the algorithm runs even faster, with a speed of 1.3 seconds per image. This difference in performance can be attributed to the communication cost between threads, as parallelization proves to be more effective in handling larger datasets.

Chapter 4

Calibration

In this chapter, we first introduce the principles of camera calibration, including intrinsic and extrinsic parameters calibration. Then we present our *self-adaptive calibration algorithm* and have a discussion on distortion coefficients. Lastly, we provide both visual and quantitative evaluations of our methods.

4.1 Principles

Camera calibration is a necessary process that involves determining the intrinsic and extrinsic parameters of the camera. The intrinsic parameters include the intrinsic matrix and distortion coefficients, which remain constant regardless of the camera's pose and position. On the other hand, the extrinsic parameters consist of rotation and translation vectors, which describe the camera's pose and position in world coordinates.

In the commonly used pinhole camera model, the image-to-world transformation can be represented as:

$$x \simeq PX = K[R \quad T]X = K[R \quad -R^T C]X \tag{4.1}$$

Here, P is called the projection matrix. K is the intrinsic calibration matrix, R is the rotation matrix, and T is the translation vector. However, it is worth noting that sometimes the rotation matrix R is represented as a vector instead of a matrix. The transformation between the matrix and vector forms of R is known as the Rodriguez Transformation.

The intrinsic calibration matrix K is typically represented as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Here, f_x and f_y denote the camera's focal lengths along the x-axis and y-axis, respectively. c_x and c_y represent the offsets of the principal point. The intrinsic calibration matrix K is an upper triangular matrix.

To estimate the camera parameters, given 2D image points as observations x and the corresponding 3D world points in the world coordinate X, their relationship can be expressed as:

$$x \simeq PX \tag{4.3}$$

The projection matrix P can then be decomposed into intrinsic and extrinsic components using RQ-decomposition, with the normalization of the Q matrix. The R matrix (an upper triangular matrix) represents the calibration matrix K, while the Q matrix (an orthogonal matrix) represents the rotation matrix R [20].

4.2 Dataset Preparation

The goal of calibration is to obtain accurate estimations of both intrinsic and extrinsic parameters. To achieve this, we divide the calibration process into two parts: one for intrinsic parameters and the other for extrinsic parameters. For each part, we utilize different image sets.

The first image set is named as **calibration dataset**, which consists of images of the ChArUco board in various poses. The variation in board pose provides distinct visual cues, making it easier to estimate accurate intrinsic parameters.

The second image set is named as **object dataset**, which includes images of fossil objects and the ChArUco board. This set serves a dual purpose, as it is used in the calibration process and employed in the multi-view matching stage. By generating the object image set immediately after the calibration image set, we ensure a similar environment for both sets. Therefore, it guarantees the validity of using the same intrinsic parameters in the subsequent extrinsic calibration.

As Section 2.2 mentioned, we have prepared five object datasets for later experiments. We have also created three calibration datasets (Flat, Irregular, and Box) to facilitate the calibration process. Each object dataset (Flat-1, Flat-2, Irregular-1, Irregular-2, and Box) corresponds to a specific calibration image dataset: Flat for Flat-1 and Flat-2, Irregular for Irregular-1 and Irregular-2, and Box for the object dataset Box. This arrangement allows us to associate the appropriate calibration images with their corresponding object datasets, ensuring accurate calibration and subsequent analysis.

4.3 Self-Adaptive Calibration

We have implemented a self-adaptive calibration algorithm based on ChArUco Board corners, drawing inspiration from the OpenCV Charuco package and the approach described in [2]. Our motivation for developing this algorithm stems from the observation that performing calibration only once does not meet our requirements due to the influence of image noise on the accuracy of detected corners. Therefore, we needed a method to remove outliers repetitively during the calibration process.

The core idea of our algorithm is to iteratively analyze the input data statistics, determine an appropriate threshold for outlier removal, and then perform calibration again on the refined dataset. We adopted the 3-σ principle, which is widely used for outlier detection, to identify outliers. In a normal distribution, approximately 99.7% of the data lies within three standard deviations from the mean. Assuming the error of corner observation E follows a normal distribution with a standard deviation of σ:

$$E \sim N(0, \sigma) \tag{4.4}$$

The observed corner X' should also follow a normal distribution:

$$X' = X + E \sim N(\mu, \sigma) \tag{4.5}$$

Here X represents the distribution of the ground truth corners, and μ is the mean of X. Therefore, we can employ the 3-σ principle to filter out outliers in X'.

During our experiments, we discovered that almost all the corners fell within the 3-σ range, while the accuracy could not meet our expectations. Consequently, we modified our algorithm to utilize the 2-σ principle, encompassing 95% of the data on average. In each iteration, we filter out corners outside the 2-σ range, calibrate, and calculate the changes in projection errors between two consecutive iterations. We utilize the change in projection errors as the stopping criterion for the iteration loop. The loop terminates when no corners are removed, indicating that the remaining corners are concentrated enough to generate an accurate calibration result.

Below is the pseudo-code for our self-adaptive calibration algorithm which we apply in the intrinsic calibration process of our system:

Algorithm 1 Self-Adaptive Calibration

```
$intrinsicParams \leftarrow Calibrate(corners)$
$prevError \leftarrow ProjectionError(intrinsicParams)$
while $\Delta error < ErrorThreshold$ do
    $intrinsicParams \leftarrow Calibrate(corners)$
    $error \leftarrow ProjectionError(intrinsicParams)$
    $\mu, \sigma \leftarrow error$
    $corners \leftarrow corners \in (\mu - 2\sigma, \mu + 2\sigma)$
    $\Delta error \leftarrow error - prevError$
    $prevError \leftarrow error$
end while
$Return \quad intrinsicParams$
```

4.4 Distortion Coefficients

The pinhole camera model serves as an ideal representation in camera geometry theory. However, in practical applications, cameras have lenses that introduce unwanted distortion to captured images. This distortion comprises radial and tangential distortion, which can be parameterized using various distortion coefficients. There are a total of 14 distortion coefficients available:

$$dist = (k_1, k_2, p_1, p_2, [, k_3[, k_4, k_5, k_6[, s_1, s_2, s_3, s_4[, \tau_x, \tau_y]]]]) \tag{4.6}$$

However, in many tasks, not all of these coefficients are necessary. The first four coefficients are already enough to meet the requirement. Including additional coefficients can improve the accuracy of calibration (See Table 4.1), but it can also introduce undesirable artifacts, such as image corner cracks, during the undistortion process (See Fig 4.1).

Through experiments conducted in our system, we observed that cracks are only introduced when coefficients beyond k_4 are used. Therefore, we choose to include distortion coefficients k_1 to k_4 in our calibration process. This selection balances improving calibration accuracy and avoiding introducing noticeable artifacts in the undistorted images.

Table 4.1: Average projection error when distortion coefficient number changes. The smallest error appears when there are six coefficients, but it introduces unwanted artifacts. Thus we choose four coefficients with the second smallest error.

Distortion Coefficients	Average Projection Error
$k_1 \sim k_3$	0.91618156
$k_1 \sim k_4$	0.914913
$k_1 \sim k_5$	0.91551137
$k_1 \sim k_6$	0.9145506

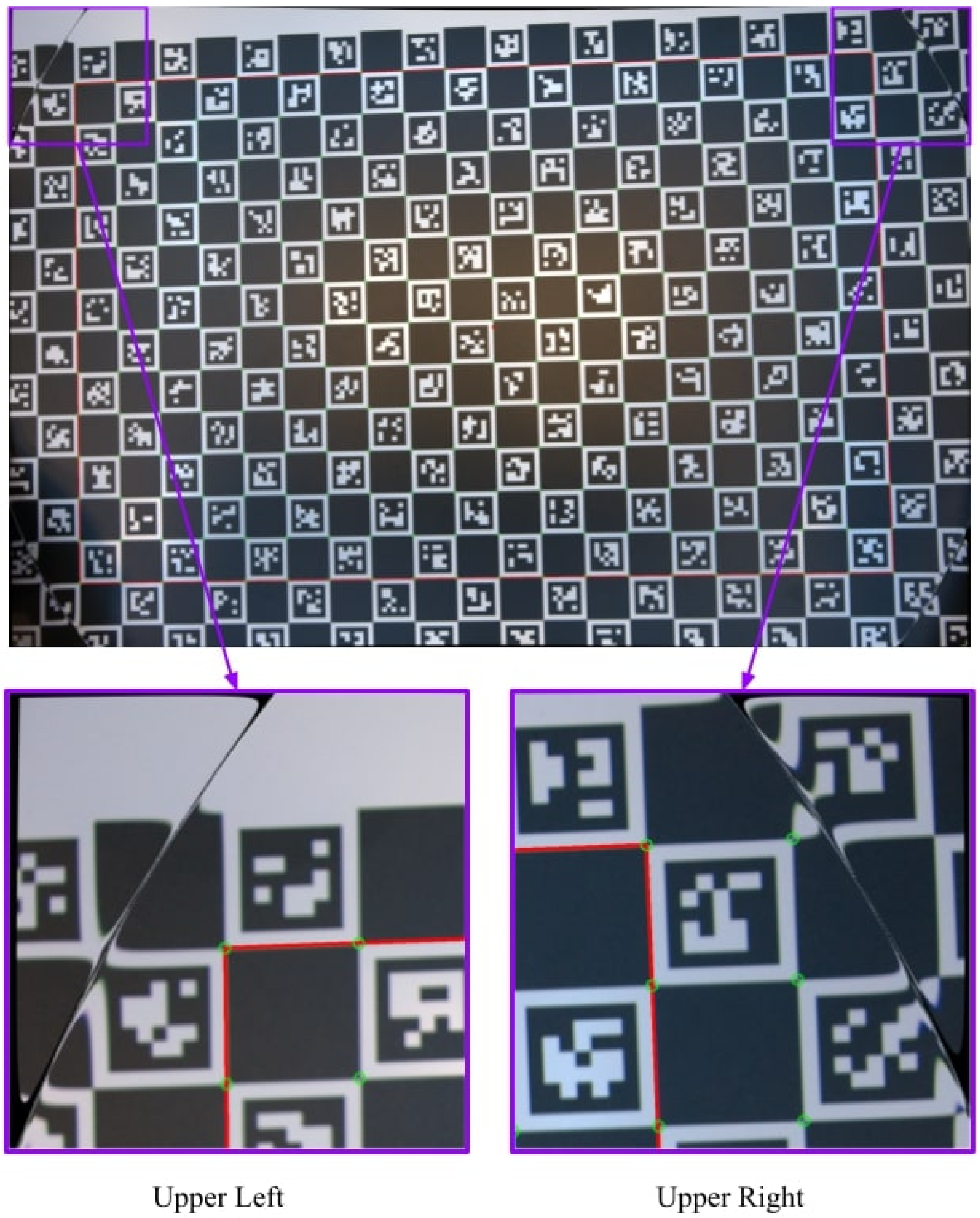


Figure 4.1: Visualization for unwanted artifacts in images. Top: Artifacts appear at four corners of the image. Bottom: Artifacts in detail. The artifacts look like cracks along the diagonal.

4.5 Perspective-n-Points

To estimate the extrinsic parameters of the camera in our object image set, which provide information about camera pose and position for each object image, we employ the Perspective-n-Points (PnP) technique. This method utilizes the coordinates of detected points in both 3D world space and 2D image space. Our implementation solves the PnP problem using a non-linear Levenberg-Marquardt minimization scheme [12, 31] and leverages the APIs provided by OpenCV.

Before solving the PnP problem, we perform image undistortion on the images and adjust the intrinsic matrix using the undistortion coefficients. This step helps to improve the accuracy of the calibration. We then visualize the cameras' positions in both 2D and 3D space. The results of this process can be found in Section 4.6, where we present a comprehensive evaluation of the calibration results.

4.6 Evaluation

4.6.1 Visual Evaluation

Intrinsic Calibration

To visually evaluate the accuracy of our algorithm, we generate images with red circle markers placed on the detected points and green circle markers on the projected points. The projected points are obtained by projecting the ChArUco corners from the template onto the image using the estimated calibration parameters. By comparing the overlap between the two sets of circles, we can assess the accuracy of both the detection and calibration processes (See Fig 4.2).

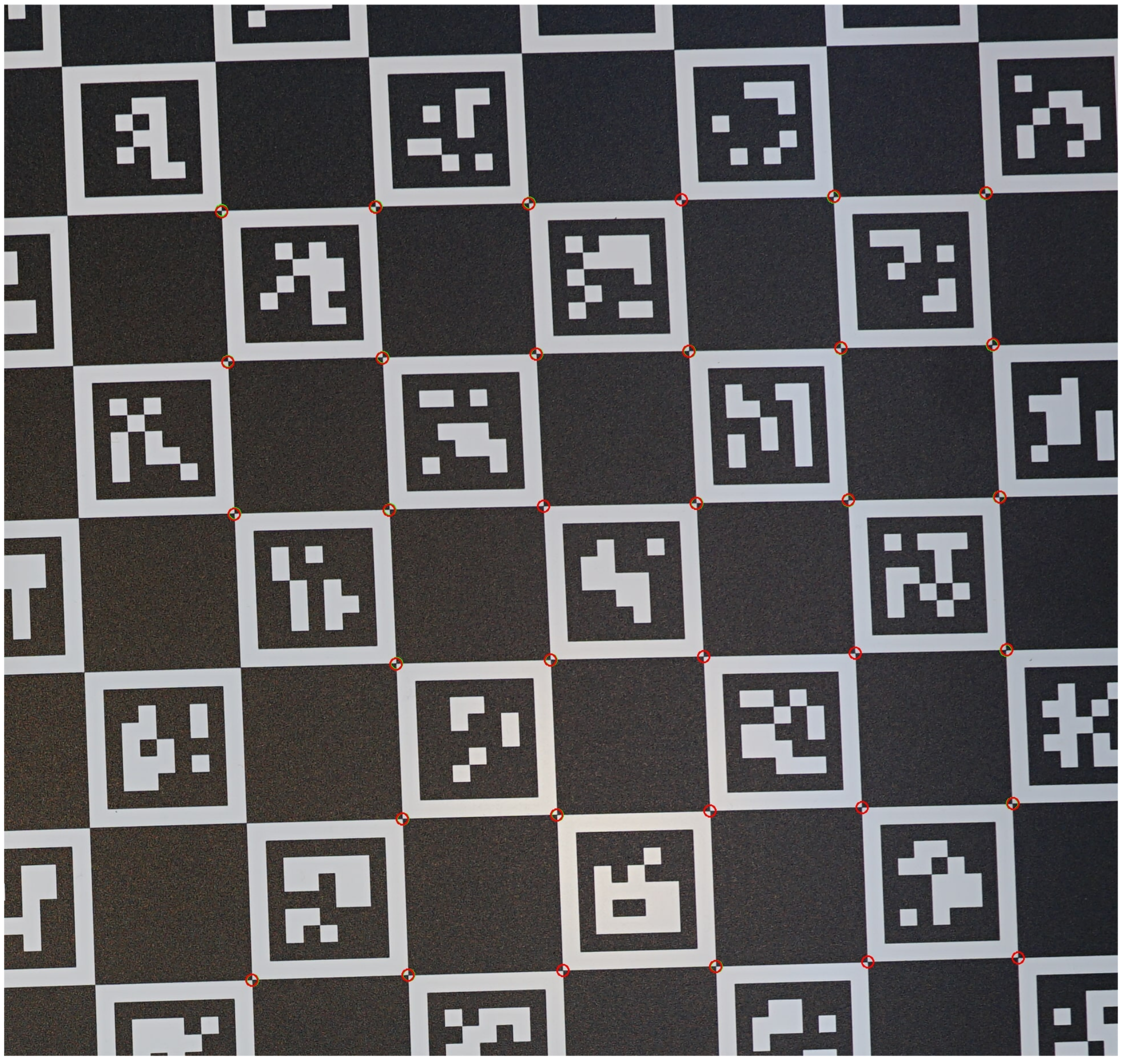


Figure 4.2: Visualization of reprojection. Red circle: Corners from the detection algorithm. Green circle: Corners from reprojection. Circles of the same corner overlap in this image, indicating that the parameters obtained by the calibration algorithm are accurate.

To further assess the effectiveness of our undistortion process, we draw a rectangle on the ChArUco Board and examine the result. (See Fig 4.3) In the original image, the edges of the rectangle do not align accurately with the edge of the grids on the ChArUco Board. However, after applying undistortion, the edges of the rectangle align perfectly, indicating the successful removal of distortion effects.

Extrinsic Calibration

We also provide visualizations of camera position estimation in both 2D and 3D space. In Figure 4.4, the camera positions are projected onto a parallel plane to the ChArUco Board. The scale used in the visualization is the same as the real-world length, allowing us to compare the mean distance of camera movement with scanner settings in vertical and horizontal directions. The indices near the points indicate the capturing order of the images, and the points are connected by blue lines to depict the path of camera movement.

In Figure 4.5, the camera positions are represented as cones, with different colors indicating the capturing order of images. The apex of a cone shows the pose of a camera. This 3D visualization provides a comprehensive view of the camera positions and poses.

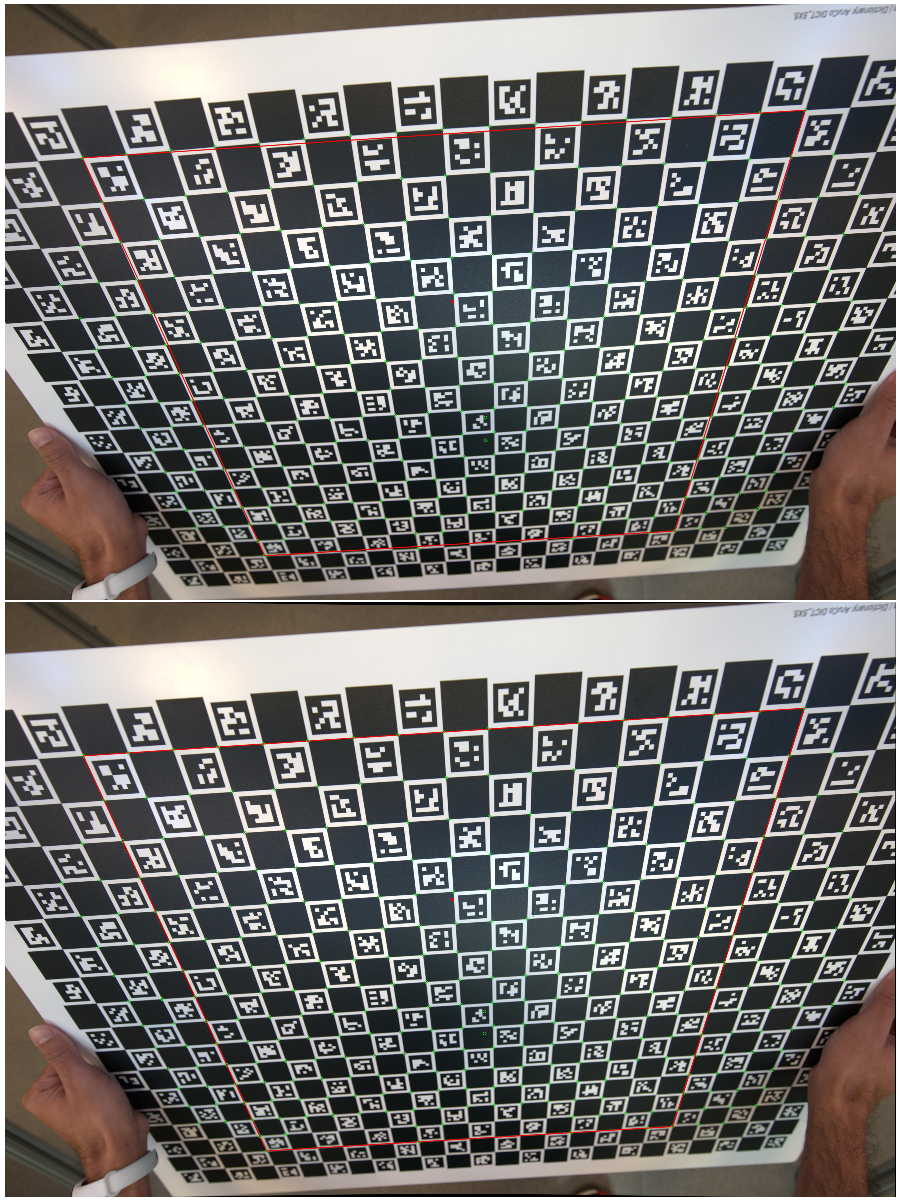


Figure 4.3: Comparison between the original image (top) and the undistorted image (bottom). In the original image, the red lines do not align with the edges of the checkerboard cells, while in the undistorted image, the red lines align much better.

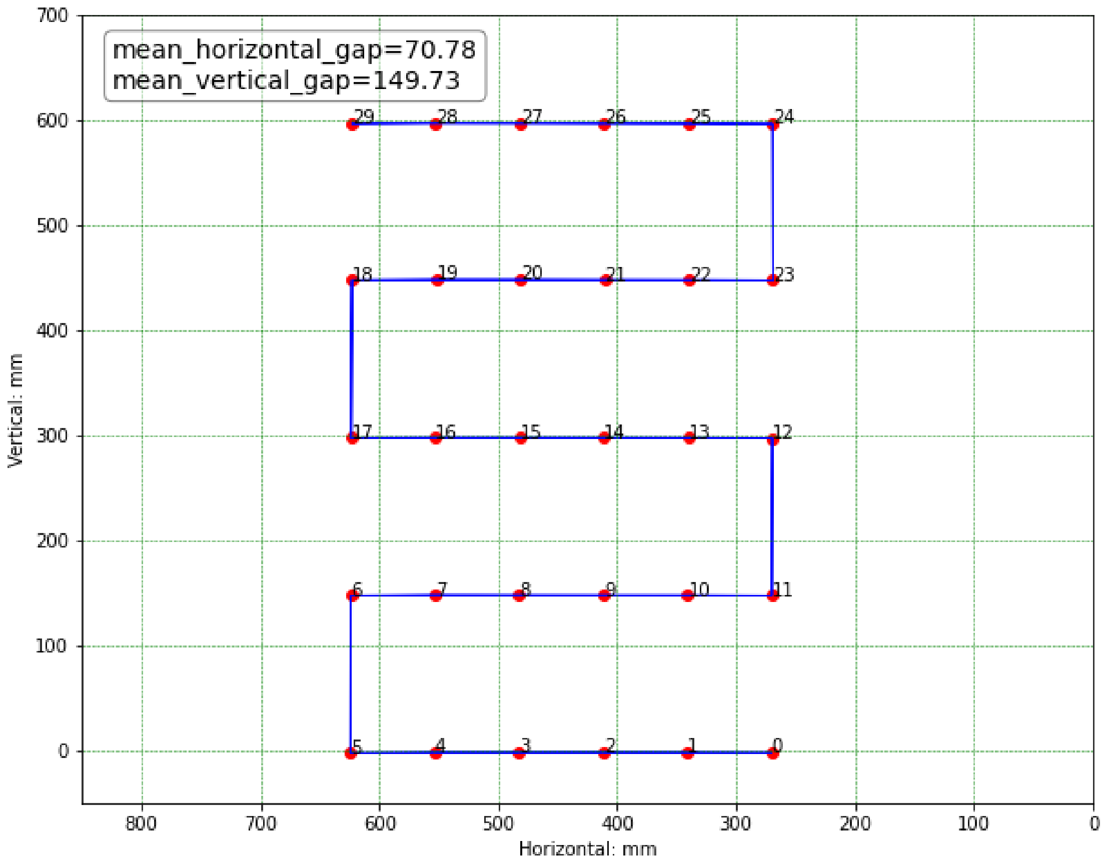


Figure 4.4: Visualization for camera position in 2D space. Red dots represent the camera positions at the time of capturing each image. The indices nearby indicate the order of images. Blue lines show the track of camera movement. The camera positions calculated using the calibration parameters match the designed pattern of the scanner.

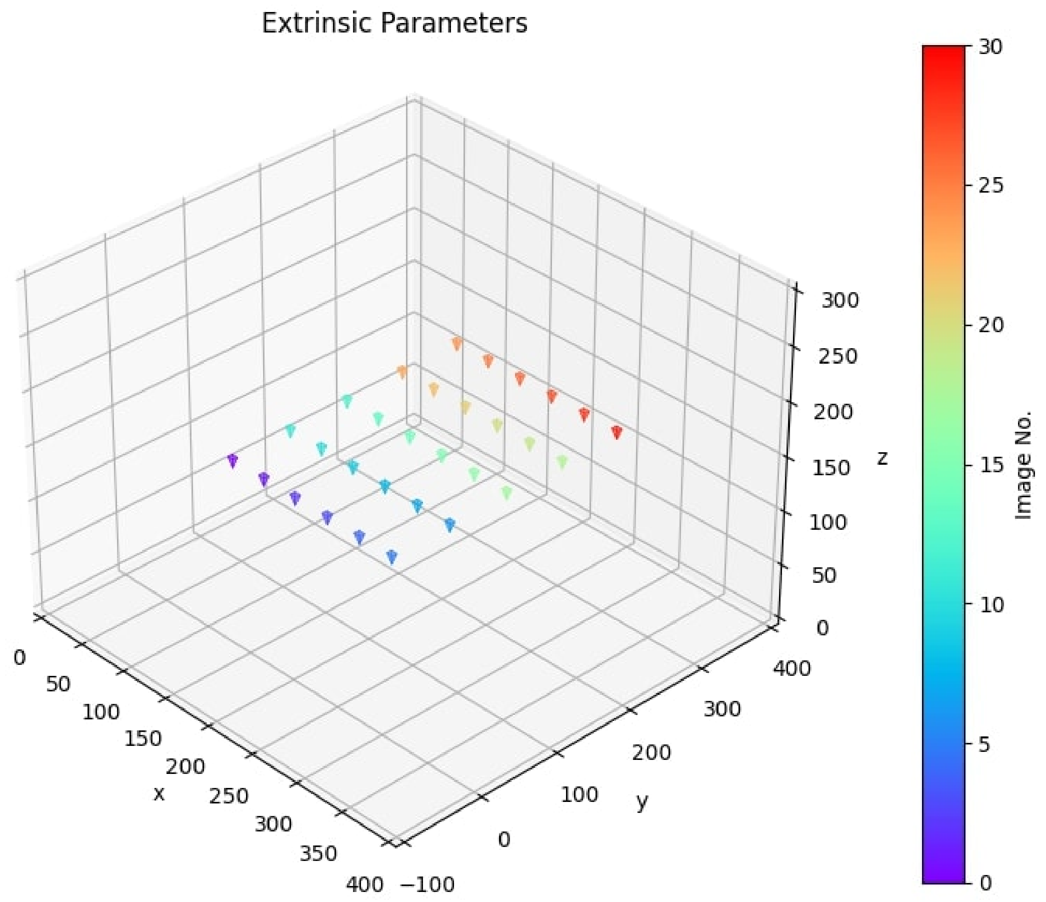


Figure 4.5: Visualization for camera position in 3D space. The small colored cones represent the camera's position in three-dimensional space, with the apex of the cones representing the camera's orientation. The color of the cone represents the order of the corresponding camera position, starting from purple and ending with red. The camera positions are distributed on a horizontal plane in three-dimensional space.

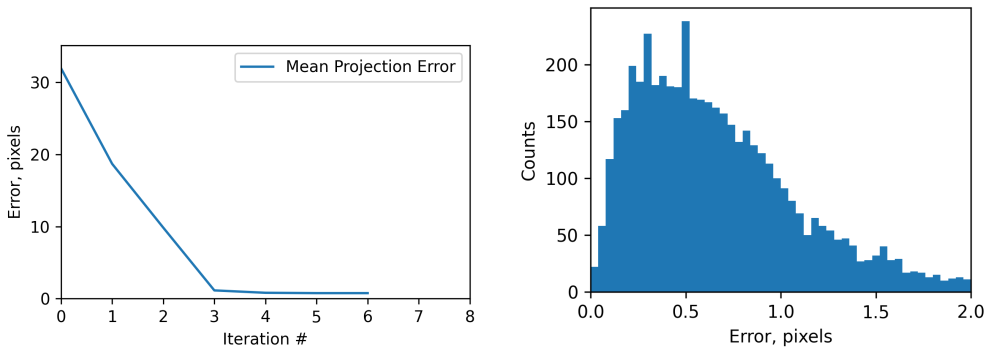
4.6.2 Quantitative Evaluation

Self-Adaptive Calibration

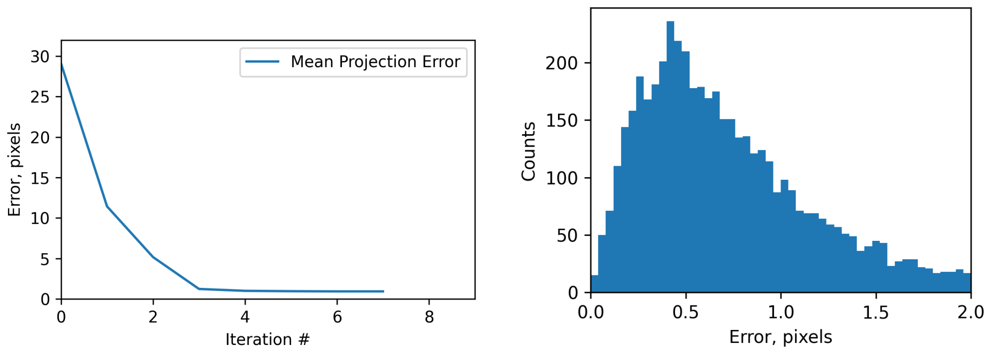
We evaluate the performance of our adaptive calibration method by calculating the mean projection error in each iteration and presenting the results as curve plots (See Fig 4.6). Our algorithm consistently converged within six iterations across all three calibration image sets. In the Flat and Irregular datasets, we observed that the initial projection error was around 30 pixels, rapidly decreasing to less than 1 pixel after three iterations. In the Box dataset, the projection error was reduced from 1.2 to 1.0. This outcome serves as solid evidence supporting the effectiveness of our adaptive calibration method.

Intrinsic Calibration

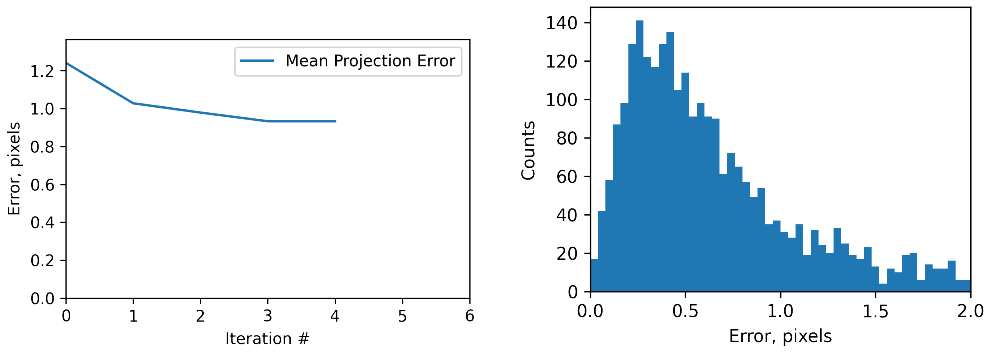
To assess the accuracy of intrinsic calibration, we projected all points within each calibration image set with estimated intrinsic parameters and computed the projection error. Subsequently, we created histograms to visualize the distribution of projection errors (See Fig 4.6). Across all three calibration image sets, we observed that most points exhibited an error within the range of 1 pixel. This result indicates that our intrinsic calibration approach yields precise results, with most points accurately projected onto their corresponding image positions.



(a) Flat



(b) Irregular



(c) Box

Figure 4.6: Results of Self-Adaptive Calibration. Average reprojection error in each iteration (left) and distribution of projection error of the final output (right). Calibration datasets from top to bottom are Flat, Irregular, and Box.

Extrinsic Calibration

We computed the mean distance between camera positions to evaluate the extrinsic calibration and compared it to the camera movement specified in the scanner settings (See Table 4.2). We found that most differences between the movement distance and scanner settings were within 1 mm. It indicates that our extrinsic calibration provides accurate estimates of the camera positions, aligning closely with the intended movements set by the scanner.

Table 4.2: Contrast between calculated camera movement distance and scanner settings (in mm). The distances are calculated with calibration parameters and are similar to the designed movements of the camera scanner. The differences in most cases are within 1.0 mm.

Image Set	Vert Dist	Vert Settings	Horizon Dist	Horizon Settings
Flat-1	149.73	150	70.78	70
Flat-2	99.32	100	150.86	150
Irregular-1	149.49	150	70.69	70
Irregular-2	98.96	100	150.85	150
Box	43.56	44.40	23.31	23.70

Chapter 5

Multi-View Matching

This chapter serves as a comprehensive guide to multi-view matching and depth estimation. It begins by discussing the main principles and geometry involved in multi-view matching, providing a solid foundation for the subsequent sections. Next, the chapter delves into the Sweeping Plane Algorithm, which is a crucial reference for our algorithm. Then we present a detailed explanation of our Re-projection Matching Algorithm, showing its specific methodology and techniques. Finally, we thoroughly evaluate and discuss our methods, allowing for a comprehensive analysis of their performance.

5.1 Principles

Multi-view matching is a fundamental and vital technique in 3D reconstruction. It plays an important role in estimating the depth of all the pixels within 2D images, which enables the generation of either sparse or dense point clouds in 3D space. While numerous approaches exist to accomplish this objective, this section reviews two prevalent methods: the disparity map and the re-projection method. Both techniques offer distinct strategies for matching corresponding points across multiple views and inferring depth information, contributing to the reconstruction process.

Disparity Map

The disparity map identifies corresponding pixels between images captured from different viewpoints (See Fig.5.1). We measure the disparity d between corresponding pixels in left and right (or top and bottom) images. The depth z of a pixel can be calculated by:

$$z = BF\frac{1}{d} \tag{5.1}$$

Here, B and F represent the baseline and focal length, respectively. Given a fixed focal length, this equation demonstrates that the baseline B is a magnification factor for obtaining depth z. It has been observed that increasing the distance between camera positions can lead to more precise depth estimations [32].

However, as the baseline between two images increases, finding accurate matches between pixels becomes more challenging. It can result in more false matches and reduce the overall accuracy of disparity estimation. Therefore, there is always a trade-off between precision and accuracy when determining the appropriate baseline. Furthermore, the information contained in a disparity map is closely tied to the specific pair of source images used and cannot be directly fused when dealing with multiple views simultaneously.

Re-projection

Apart from the disparity map, depth estimation can be achieved using the re-projection method. This approach involves finding corresponding pixels between

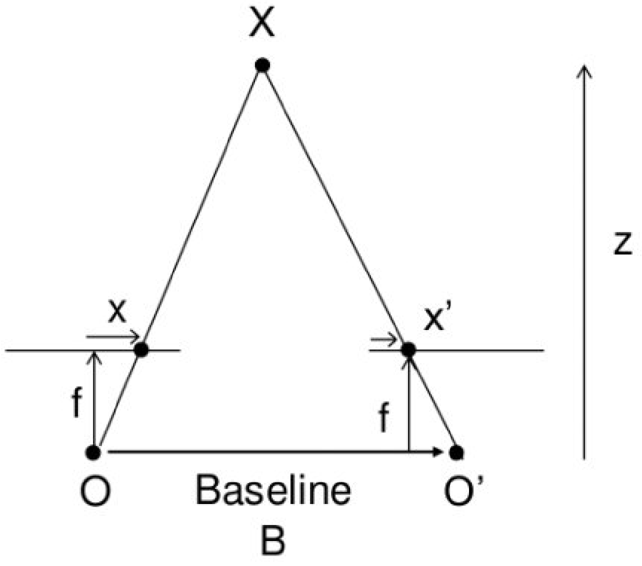


Figure 5.1: Geometry of multi-view disparity. Pixel depth z can be estimated by baseline B, focal length f, and pixel disparity $x' - x$. [2]

two images and utilizing intrinsic and extrinsic parameters to estimate depth. Given matched image pixels x_1 and x_2, the relationship can be expressed as follows:

$$x_2 = K_2(R_{12} - T_{12}^T n/depth)K_1^{-1}x_1 \tag{5.2}$$

Here, K_1 and K_2 represent the intrinsic matrices, while R_{12} and T_{12} are transformation vectors that can be calculated using the extrinsic parameters. The accuracy of the re-projection method heavily relies on the precision of the calibration process and the matching criteria employed, and the computational cost of re-projection is influenced by the searching strategy used during the matching process.

We choose the re-projection method to estimate pixel depth in our system. This decision was made due to the high accuracy of our calibration process. We can have a good estimation of depth with such precise intrinsic and extrinsic parameters.

[2]Reference from https://docs.opencv.org/3.4/dd/d53/tutorial_py_depthmap.html

Additionally, we aim to improve depth estimation accuracy by fusing information from multiple-view images. The details of our re-projection method are presented in Section 5.3.

5.2 Space-Sweeping Plane Algorithm

In 1996, Collins et al. proposed a space-sweep approach to address the challenge of multi-image matching [7]. This method was further enhanced by Gallup et al. in their work [16]. The core idea of this approach is to generate multiple planes at varying depths and select the one that minimizes target loss (See Fig. 5.2).

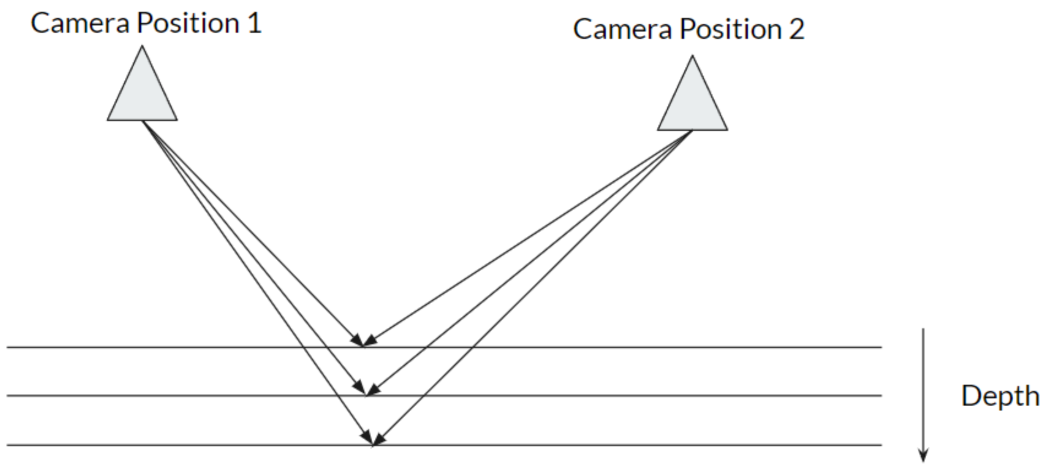


Figure 5.2: Visualization of the sweeping plane algorithm. Pixels from cameras are projected onto multiple planes with different depths. The plane with the minimum loss will be selected as the optimal depth.

The sweeping plane algorithm belongs to the re-projection category as it estimates the depth of objects through the projection process. It fulfills the requirements of true multi-image matching as outlined in [7], which include:

1. Generalization to any number of images greater than 2.
2. Complexity of $O(n)$ where n is the number of images.
3. All images are treated equally.

However, in our system, we have modified the sweeping plane algorithm. First, since multiple images are captured from different viewpoints, we leverage the sweeping plane technique to satisfy the first two properties. However, we choose the image in the middle as the reference image and only re-project pixels from the reference image to other images. Second, instead of performing the sweeping plane search to find the best match for each pixel, we directly search for the best match of reference image pixels in the target images. These modifications allow us to adapt the sweeping plane algorithm to our specific system requirements and improve its efficiency and accuracy.

5.3 Re-projection Matching

In this section, we provide a detailed explanation of our re-projection algorithm. The first step is to select the image in the middle of the image set as the reference image (see Figure 5.3). We focus on estimating the depth of all pixels in the reference image while treating the other images as target images, which provide matching targets for the pixels in the reference image.

There are two reasons for choosing a reference image in our system. First, we aim to generate a dense point cloud of the fossil object. Therefore, we select the image that contains the most pixels of the object, which is always the image in the middle. Second, by choosing a reference image, we reduce our system's time and

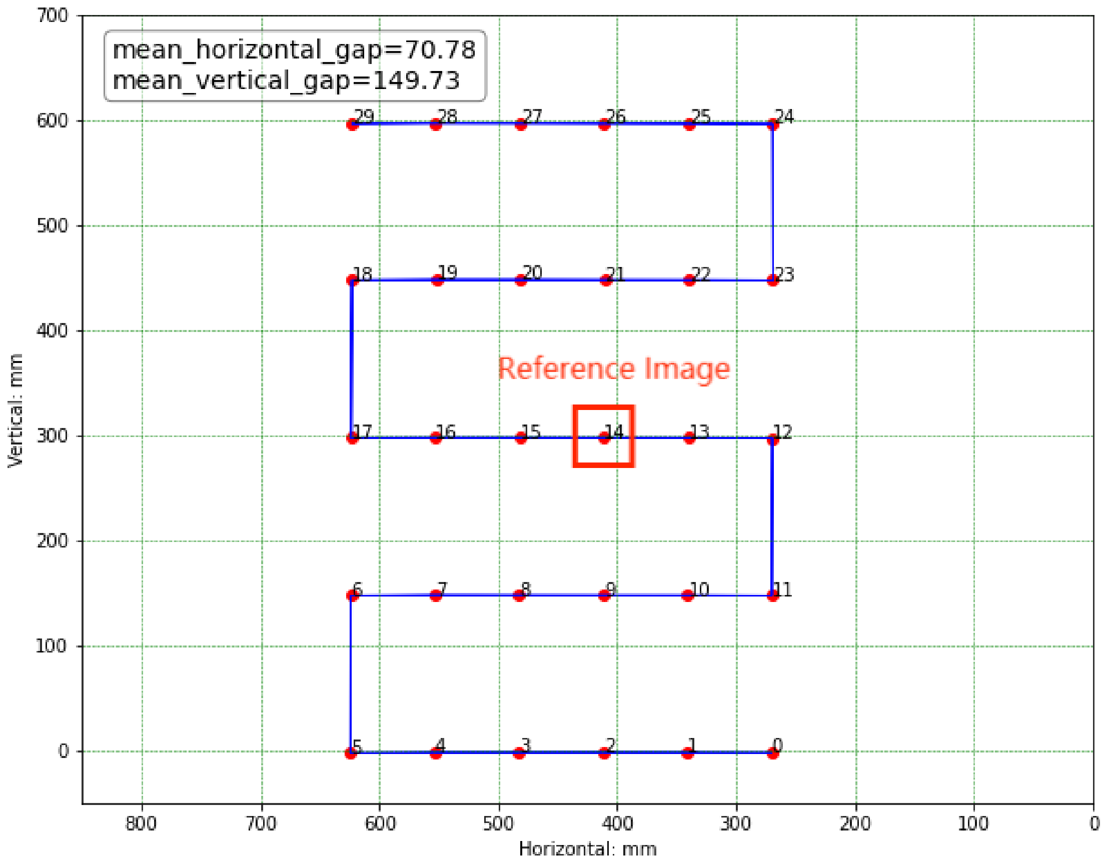


Figure 5.3: Example of the reference image in the image grid. The reference image is closest to the center in horizontal and vertical directions. When two images are in the center, we choose the one closer to the camera origin.

memory usage. Instead of performing matching between every pair of images, we only match the reference image with the other images. This approach improves the efficiency of the matching process, especially when some image pairs have few pixels in common.

Then we select a region of interest (ROI) in the reference image and project all the pixels within this region into 3D world coordinates (see Figure 5.4). This region selection process helps our algorithm focus on the fossil object and remove unnecessary background, saving much time and reducing the final output size.

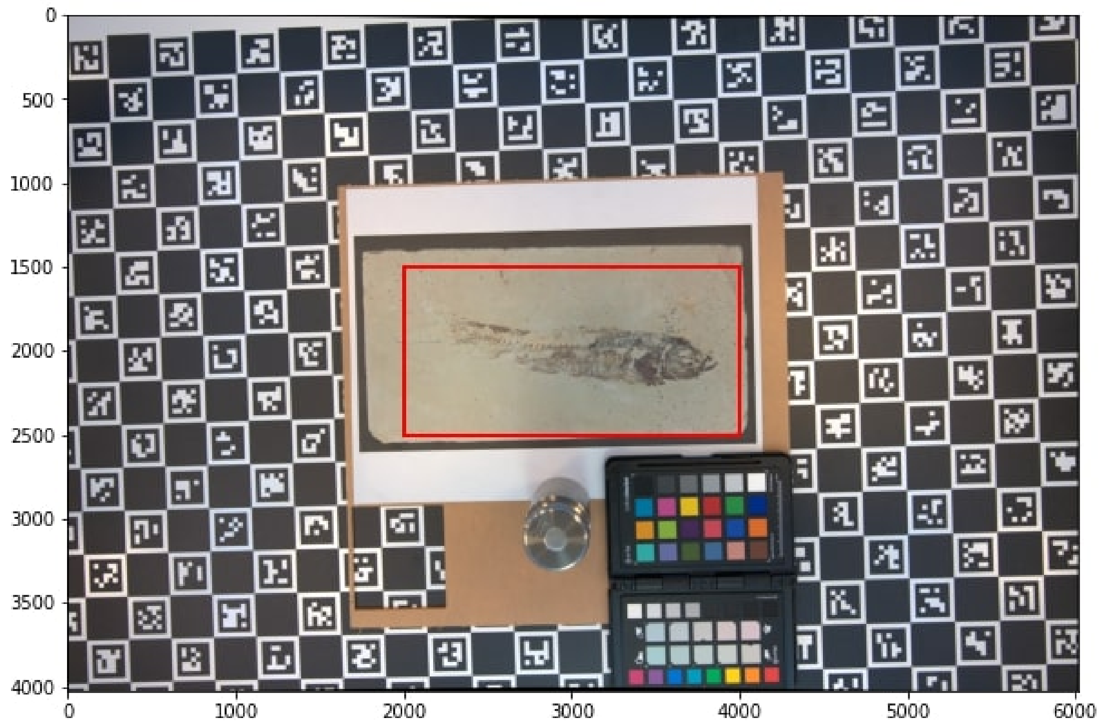


Figure 5.4: Visualization of region selection. The red rectangle is the edge of the region of interest (ROI). We try to include the fossils within the ROI and exclude unrelated textures such as ChArUco board and color corrector.

To perform the projection from image pixels to 3D world points, we set a depth range and determine the number of depth values to be searched, which we refer to as the *depth resolution*. The depth resolution determines the precision of our depth estimation. Assuming the depth range is d and the depth resolution is r, the minimal depth difference we can distinguish is given by:

$$\Delta d = \frac{d}{r} \tag{5.3}$$

For each imaginary plane with a specific depth value, we project the pixels within the ROI onto that plane. We implement this process by generating a ray from the camera's position and tracking the ray to find the intersection of the ray with the plane. To make our algorithm more efficient, we only perform the projection process onto two planes corresponding to the depth interval's lower and upper bounds. Then we search for optimal depth in later steps. We use ray-tracing techniques to avoid projecting pixels directly with calibration parameters, thus preventing harm from inaccuracy results in the calibration process.

Next, we reverse the projection process to unproject 3D world points from the two planes onto each target image, obtaining two corresponding pixels. These pixels are then connected with a line segment, and we search along the line with a resolution equal to the depth resolution r (See Fig. 5.5). If the resolution is set too high, the length of every step would be less than 1 pixel, causing a lot of redundant computation. Thus we use cache variables to optimize computation and update them only when encountering a new pixel, avoiding redundant calculations.

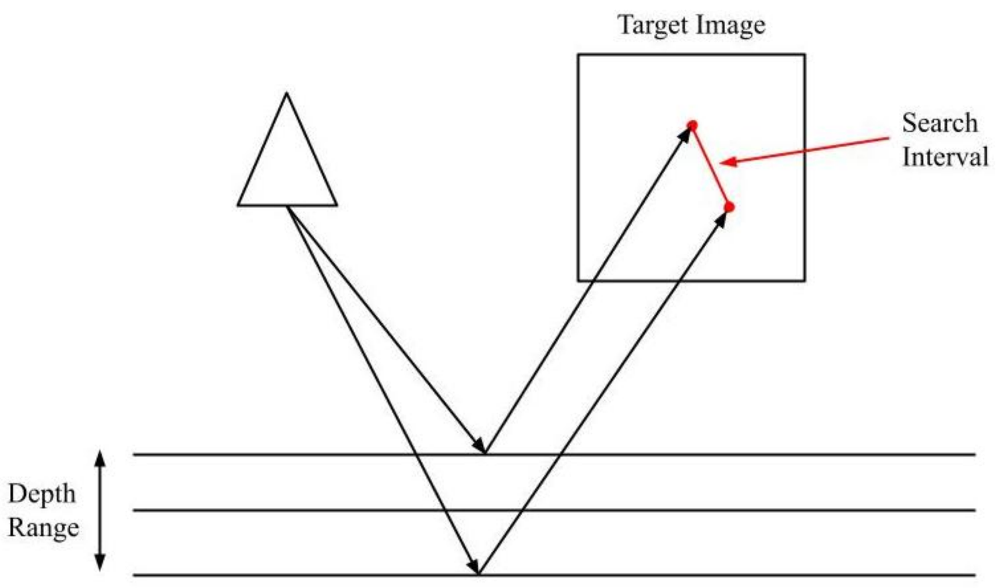


Figure 5.5: Visualization of re-projection process and searching interval. Pixels from the left camera are projected onto planes with different depths. The projected pixels with the most and least depth are unprojected as endpoints of the search interval.

We use the absolute projection matching error as the criterion to find the optimal match for a given pixel. The error is calculated as the sum of absolute differences between the reference pixel value and the corresponding target pixel values within a matching window in the reference image:

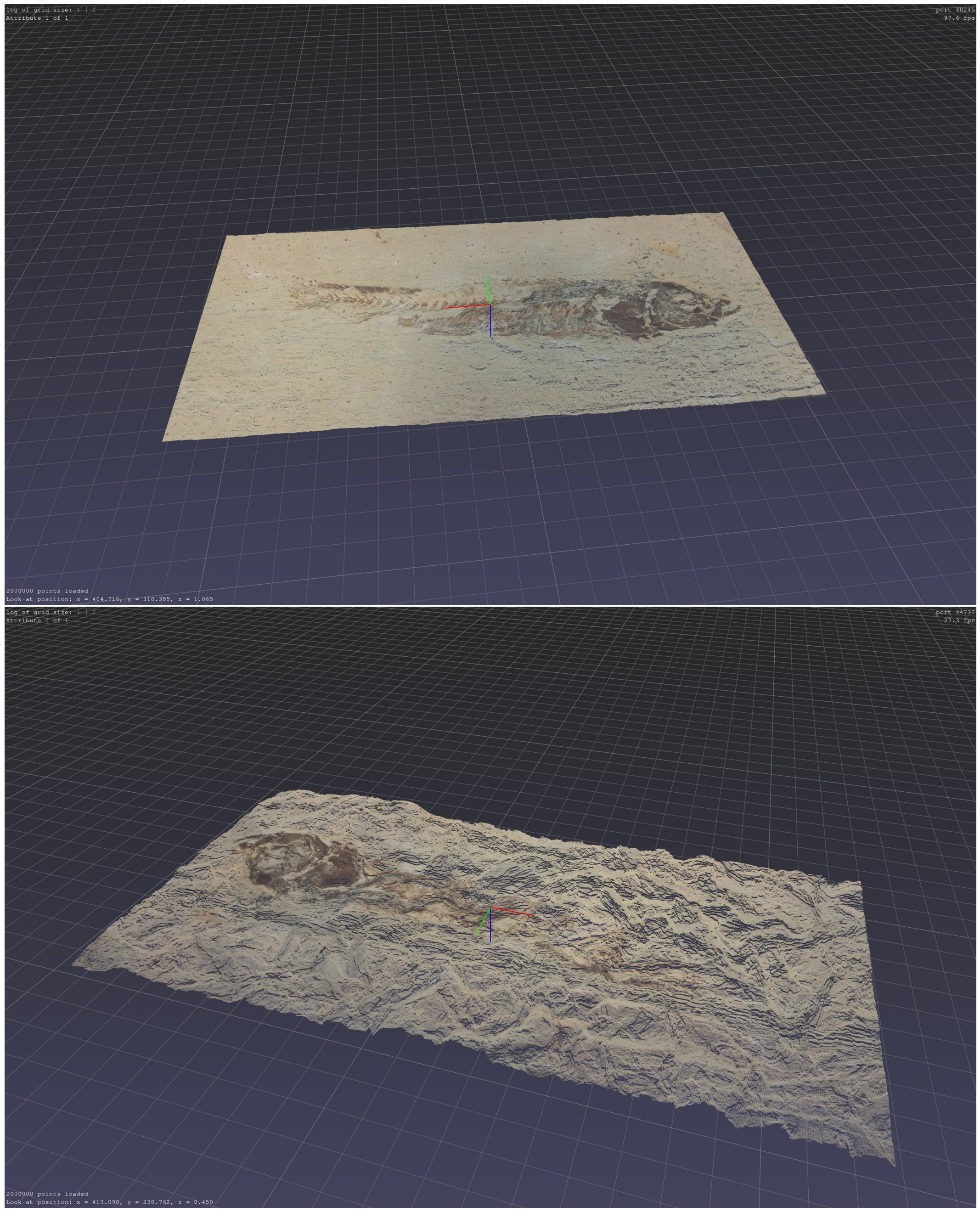
$$error = \sum_{[i,j]\in W} |ref[i,j] - target[i',j']| \tag{5.4}$$

Here W is the pixel set within a matching window in the reference image, i, j and i', j' are corresponding pixel coordinates in the two images. We calculate the matching error along the search interval and select the depth with minimal error as the optimal estimation.

After performing depth estimation for each image pair in the image set, we obtain $N - 1$ depth maps, where N is the number of images. We integrate the information from these depth maps by averaging the depth estimations for every pixel. Before averaging, we apply a 2σ criterion to filter out unreliable results. Additionally, there may be invalid estimations for pixels that do not appear in the target image. We set these estimations to NaN and ignore them during the averaging process. Finally, we project the image pixels in the reference image to 3D world coordinates using the estimated depth map to generate a dense point cloud of the fossil object.

5.4 Evaluation

We visualize the generated dense point cloud using 3D visualization software such as pptk or Open3D (See Fig. 5.6). The dense point cloud consists of approximately two million points, and each point is colored according to its corresponding pixel color in the reference image.



```
log of grid size: 1 | 2                                port 40245
Attribute 1 of 1                                         97.6 fps

2000000 points loaded
Look-at position: x = 404.716, y = 310.385, z = 1.065
```

```
log of grid size: 1 | 2                                port 44737
Attribute 1 of 1                                         27.3 fps

2000000 points loaded
Look-at position: x = 413.090, y = 230.762, z = 0.450
```

Figure 5.6: Dense Point Cloud Visualization with PPTK package. The one on the top is the Flat object, while the one below is the Irregular object. Each point is colored with its corresponding pixel color in the original image.

To assess the accuracy of our multi-view matching method, we employ Principle Component Analysis (PCA) to project all the points in the dense point cloud onto the dimension with the smallest eigenvalue. This projection helps extract the depth information from the dense point cloud, as most surface points are distributed around a plane. We generate a histogram for the PCA output of the Flat image set (See Fig. 5.7). The mean of our depth estimation is 1.79 mm, with a standard deviation of 0.237 mm.

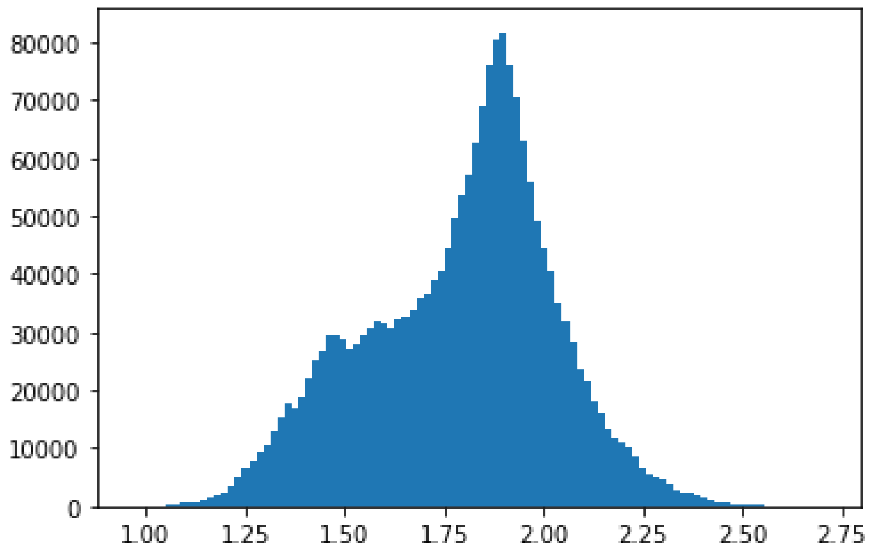


Figure 5.7: Histogram of PCA projection for the Flat object. The depth of the reconstructed point cloud approximately follows a normal distribution, with a mean value of 1.79 mm.

We also designed and performed PCA analysis on different sizes of ROIs to demonstrate the robustness of our algorithm. Table 5.1 shows that various local areas have similar mean values and standard deviations. Considering the 3σ criterion, almost all the depth estimation is within ± 1 mm around the mean value.

Table 5.1: PCA projection results with multiple ROI sizes on the same Flat object. The regions are selected in the middle of the reference image. The mean values and standard deviations are similar. And almost all the estimations lie within ± 1 around the mean value.

Region of Interest	Mean Value	Standard Deviations
500×500	1.971	0.314
1500×350	1.918	0.361
2000×1000	1.790	0.237

Chapter 6

Surface Recovery

This chapter focuses on the final step of our reconstruction pipeline: surface recovery. This step involves analyzing the geometric structure of a dense point cloud and generating a mesh grid along with normal vectors, which are essential inputs for a rendering engine. In this chapter, we present two popular surface reconstruction methods: the Ball Pivoting algorithm and the Poisson Surface Reconstruction method. Furthermore, in the last section, we evaluate these two methods to assess their performance and effectiveness.

6.1 Ball Pivoting

The Ball Pivoting Algorithm (BPA) is a polygonal mesh construction technique from a point cloud in 3D space [6]. BPA operates by generating 3D balls with a given radius. If a ball hits any three points and does not fall through, it creates a triangle. Then, the algorithm starts pivoting from the edges of the existing triangles, and every time it hits 3 points where the ball does not fall through, we create another triangle.

One key aspect of achieving successful results with BPA in our reconstruction system is finding an appropriate radius for the balls. We employ a method that proves effective in most scenarios: we compute the average distance from a point to its nearest neighbor, denoted as δ, and utilize 1.5δ and 3δ as the radii for the balls. This approach helps ensure that the generated mesh accurately captures the geometric details of the point cloud.

6.2 Poisson Reconstruction

Poisson Reconstruction is another widely used technique in 3D surface reconstruction [25, 26]. This algorithm solves a Poisson equation to generate a surface representing all points in the point cloud with the same scalar value in a designed scalar field. While we won't delve into the intricacies of how the algorithm works, we will focus on adapting it to meet the requirements of our surface reconstruction system.

Poisson reconstruction employs an adaptive octree data structure in implementation to compute the indicator function and solve the Poisson system [25]. Therefore, the choice of octree depth plays a crucial role in the algorithm. A deeper octree can store more information, resulting in a more detailed mesh, but it also increases the computational cost required for the algorithm. We conducted several experiments with different octree depths to find the optimal setting for our system. The results of these experiments are presented in Table 6.3 and 6.4.

Considering the resolution of the region of interest in our experiments is $2000 \times 1000 = 2,000,000$ pixels, we determined that an 11-level depth octree (with more than 2,000,000 pixels) is sufficient to recover an adequate amount of detail from the fossil object. This depth setting balances capturing fine details and reasonably processing time for our reconstruction pipeline.

Table 6.1: Poisson reconstruction on Flat object with different octree depth. As the depth increases, the number of generated points and meshes will grow exponentially.

Octree Depth	Point #	TriangleMesh #	Time Cost (s)
8	34,709	69,238	1.68
9	134,189	268,196	3.78
10	560,154	1,120,118	11.68
11	2,359,395	4,718,572	47.30

Table 6.2: Poisson reconstruction on the Irregular object with different octree depth. Like Flat objects, the number of generated points and meshes increases significantly with octree depth. At the same depth, irregular objects generate more details than flat objects.

Octree Depth	Point #	TriangleMesh #	Time Cost (s)
8	45,788	91,407	2.21
9	191,900	383,628	5.54
10	837,276	1,674,225	15.07
11	3,607,915	7,214,490	57.11

6.3 Evaluation

6.3.1 Visual evaluation

We evaluate the two surface-recovering methods both visually and quantitatively. We generate normal vectors for every vertex and visualize meshes with the help of Open3D Toolkits (See Fig. 6.1 and 6.2). As mentioned in the previous section, we set the radius of BPA to 1.5δ and 3δ, and the depth of Poisson Reconstruction to 11 during the evaluation process.

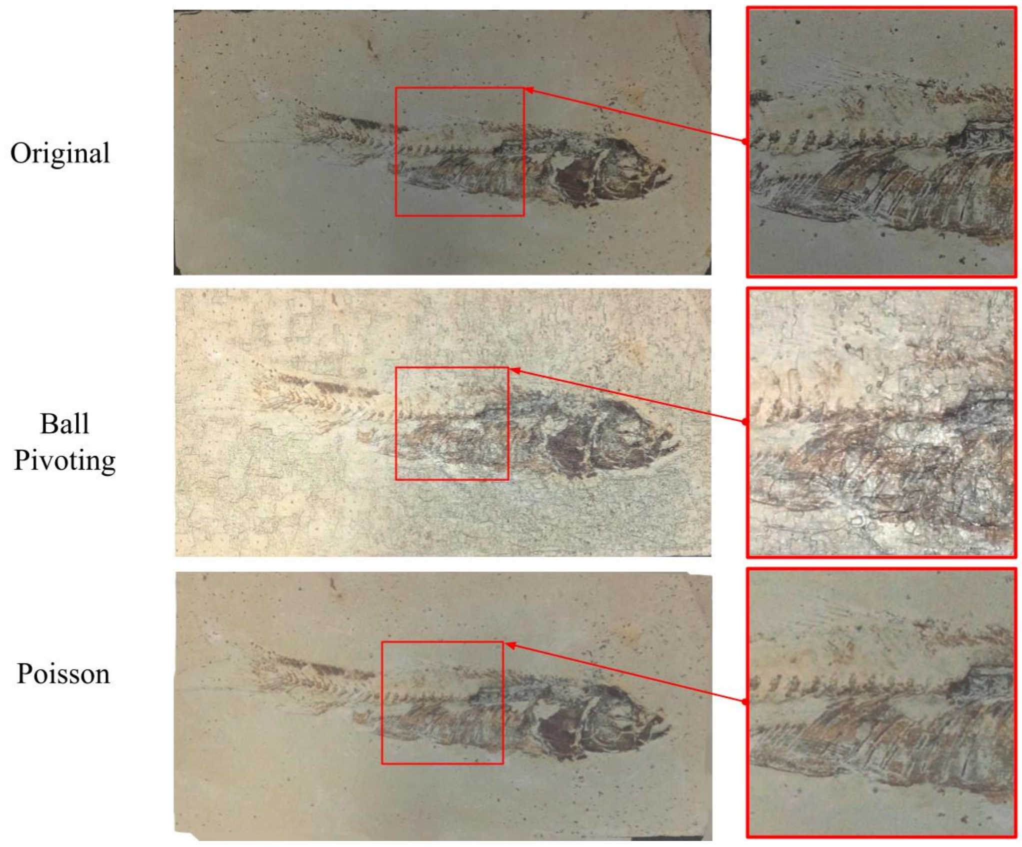


Figure 6.1: Reconstruction visualization of Flat object. Images from top to bottom are the original image, BPA output, and Poisson Reconstruction output. The generated surface is similar to the original image. BPA introduces more noise, while the surface generated by Poisson is smoother.

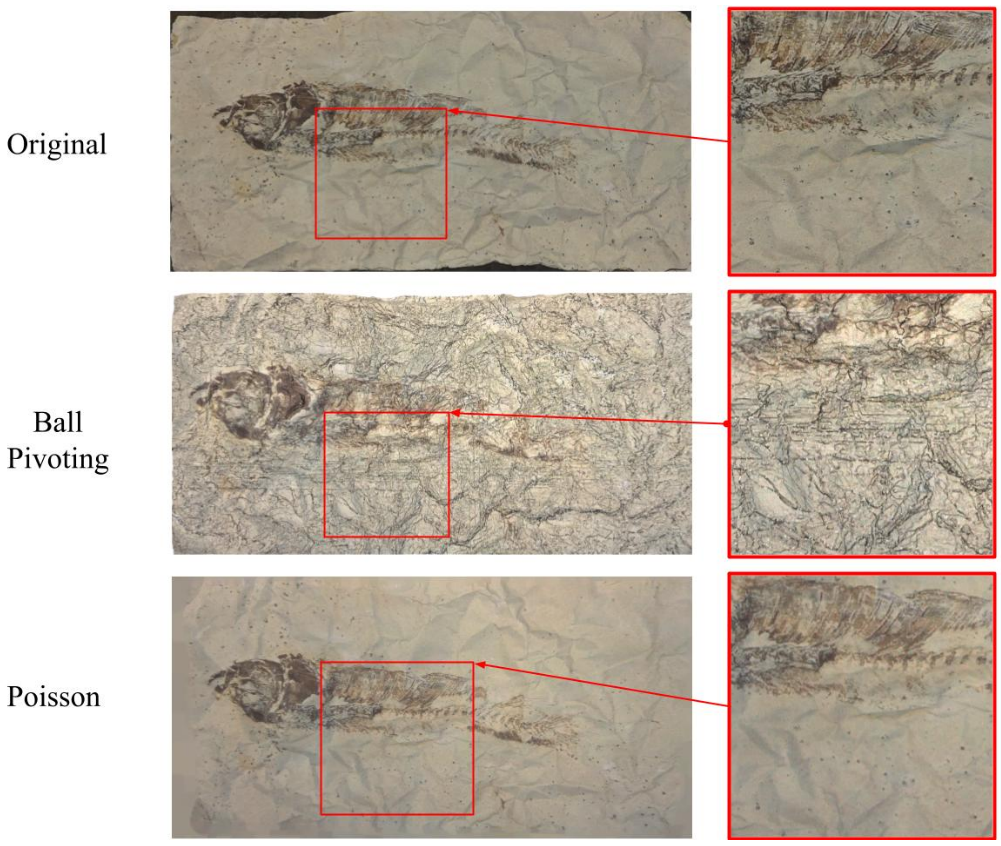


Figure 6.2: Reconstruction visualization of Irregular object. Images from top to bottom are the original image, BPA output and Poisson Reconstruction output. The generated surface matches the texture of the irregular object. The surface generated by Poisson exhibits more continuity compared to the one generated by BPA.

From the visual inspection of the meshes in Figure 6.1 and 6.2, we observe that the generated meshes closely resemble the original images. This indicates the correctness and effectiveness of our reconstruction system. Upon comparing the outputs of the two methods, we notice that BPA is more noise-sensitive, leading to more irregular textures in the resulting mesh. On the other hand, Poisson Reconstruction introduces some unexpected extensions around the surface. Fortunately, these can be easily removed during the post-processing stage by cropping.

6.3.2 Quantitative evaluation

To provide a quantitative assessment, we present the results of our comparison experiment in Table 6.3 and 6.4. We applied both methods to two different image sets and generated meshes with approximately the same number of points. It is worth noting that BPA produces meshes with fewer triangles compared to Poisson Reconstruction.

Table 6.3: Comparison between BPA and Poisson reconstruction on Flat dataset. When the level of detail is approximately the same, Poisson will generate more points and triangles in mesh and cost more time.

Method	Number of Points	Number of Triangles	Time Cost (s)
BPA	2,000,000	3,567,180	56.70
Poisson	2,359,397	4,718,576	81.15

Table 6.4: Comparison between BPA and Poisson reconstruction on Irregular dataset. For complex object surfaces, Poisson will generate more points and triangles to capture the details while still being computationally efficient.

Method	Number of Points	Number of Triangles	Time Cost (s)
BPA	2,000,000	3,280,136	65.00
Poisson	3,607,915	7,214,490	57.11

Overall, the evaluation confirms that both methods can generate plausible surface reconstructions. However, BPA tends to be more noise-sensitive, while Poisson Reconstruction may introduce unwanted extensions. The choice between the two methods ultimately depends on the specific requirements of the reconstruction task and the desired trade-off between noise sensitivity and mesh complexity.

Chapter 7

Conclusions

7.1 Thesis Summary

In this thesis, we have taken a comprehensive approach to the entire pipeline of an image-based reconstruction system tailored for fossils in paleontology. Drawing upon advanced three-dimensional multi-view reconstruction techniques developed in recent decades, we have applied them specifically to this unique case. The primary objective of this thesis is to showcase the design and effectiveness of the reconstruction system we have developed.

Throughout this work, we have presented a detailed account of the various stages involved in the pipeline and thorough testing and evaluation of these processes. In Chapter 1, we introduced the background and principles of fossil reconstruction, providing the necessary context for understanding the significance of our project. In Chapter 2, we focus on the hardware settings utilized in our experiments and the arrangement of the acquired image dataset, a prerequisite to understanding the subsequent steps in the reconstruction pipeline. Next, in Chapters 3, 4, 5, and 6, we looked into the core steps of the reconstruction pipeline. At each step, we have identified and analyzed the challenges encountered and have provided practical solutions to overcome them.

We have demonstrated that our reconstruction pipeline operates effectively within the scope of paleontological reconstruction. The thorough examination of each component and the rigorous evaluation of the system indicate its performance. By presenting this comprehensive exploration of the reconstruction pipeline, we contribute to the body of knowledge in paleontology, showcasing the potential of image-based reconstruction techniques in fossil analysis. The insights gained from this thesis will benefit the study of fossils and serve as a foundation for future research and development in the broader domain of image-based reconstruction systems.

7.2 Future Work

Many open challenges and problems remain in fossil reconstruction, intending to recover all the information about the specimens. In this section, we discuss some of the problems and the potential approaches to overcome them.

Efficiency and Robustness

Pursuing algorithms' robustness and high efficiency is an endless task in computer science research. Though our pipeline works quite well on our dataset, many complex fossil datasets exist in various forms worldwide. More efficient algorithms are still necessary to process these massive amounts of data. Meanwhile, robustness is also essential to meet the requirement of various scenarios and inputs. Continuously updating and optimizing our algorithms to cope with new challenges is the key factor to this problem.

Data Acquisition

Data acquisition is another important aspect that can pose significant expenses and time constraints in real-world applications. While building our camera-based scanner helped reduce costs, there is still room for improvement in this area. Developing algorithms that can handle inputs with fewer images or low-resolution images, including those captured by common smartphones, would significantly enhance the accessibility and affordability of fossil reconstruction.

Visualization

While obtaining a realistic model is essential, the ultimate goal of reconstruction is to provide visual information about fossils to those who may not have physical access to them. Therefore, developing user-friendly visualization systems that effectively present the reconstructed models is essential. A sound visualization system should not only deliver realistic representations but also can integrate multiple types of information. Emphasizing the development of such visualization systems will significantly enhance the impact and usability of the reconstruction pipeline.

Appendix A

Application on Real Fossil Object

We shall introduce the reconstruction results on a real fossil object in this chapter. The real object images are acquired via our camera scanner at the excavation site in Brazil. A.1 shows an example image of this real fossil object.

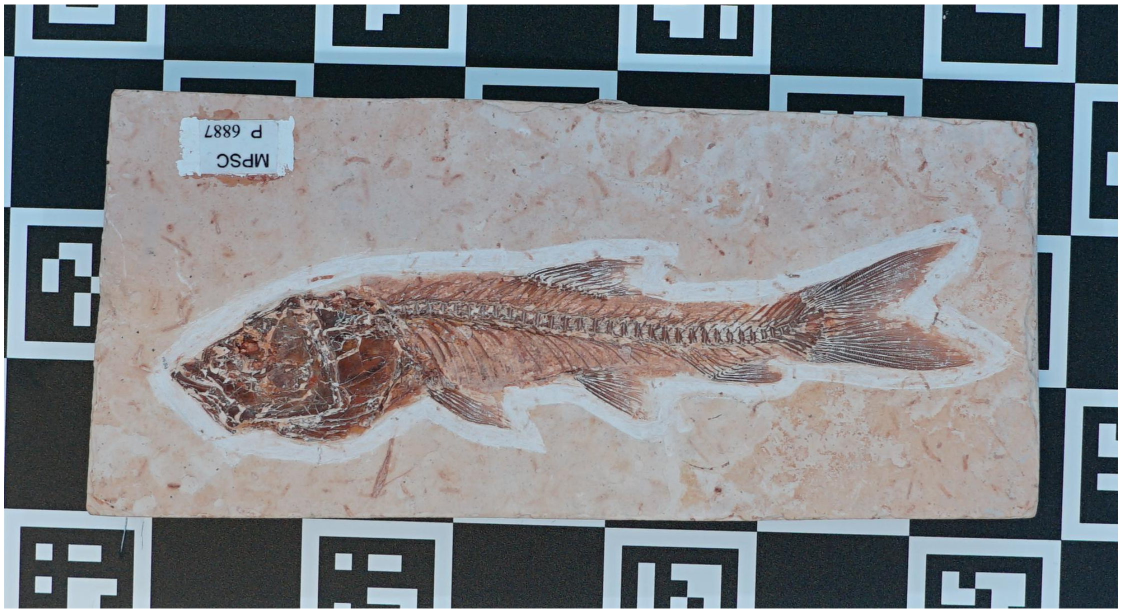


Figure A.1: An example image of the real fossil object. A ChArUco Board is placed underneath the fossil as background.

A.1 Corner Detection and Calibration

We used our corner detection algorithm in these images of a real fossil object. The corners and markers detected are shown in A.2. The detected corners lie accurately on the corners of chessboard grids.

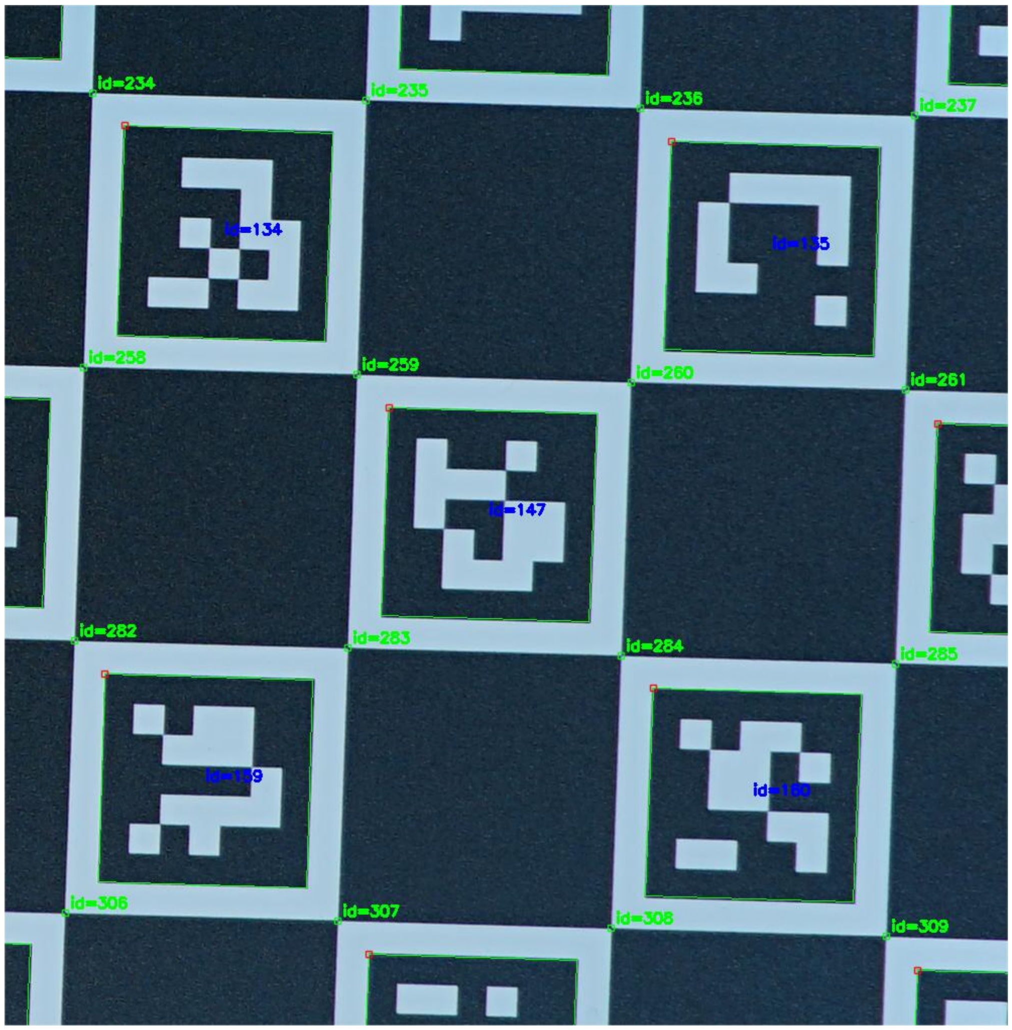


Figure A.2: The corner detection in real fossil reconstruction. Green text: Chessboard corners and IDs. Blue text: IDs of ChArUco markers. Green lines: Edges of ChArUco markers.

We used our calibration system to estimate the intrinsic and extrinsic parameters of the reconstruction system. We generate a histogram for the re-projection error of detected corners to evaluate our calibration result of intrinsic parameters (See Fig.A.3). Most of the errors lie around one pixel.

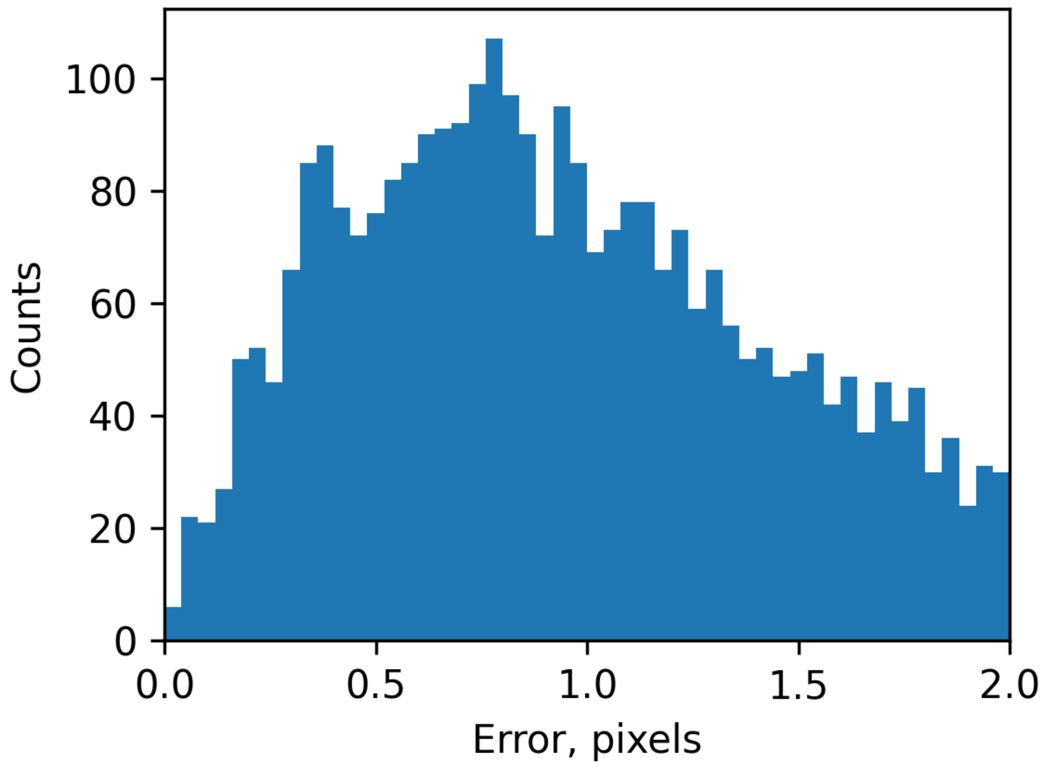


Figure A.3: Histogram for corner re-projection errors. Most errors lie within two pixels. The mean error is 0.618766, with a standard deviation of 0.192278.

We also generate a visualization of the camera positions calculated by estimated extrinsic parameters (See Fig.A.4).

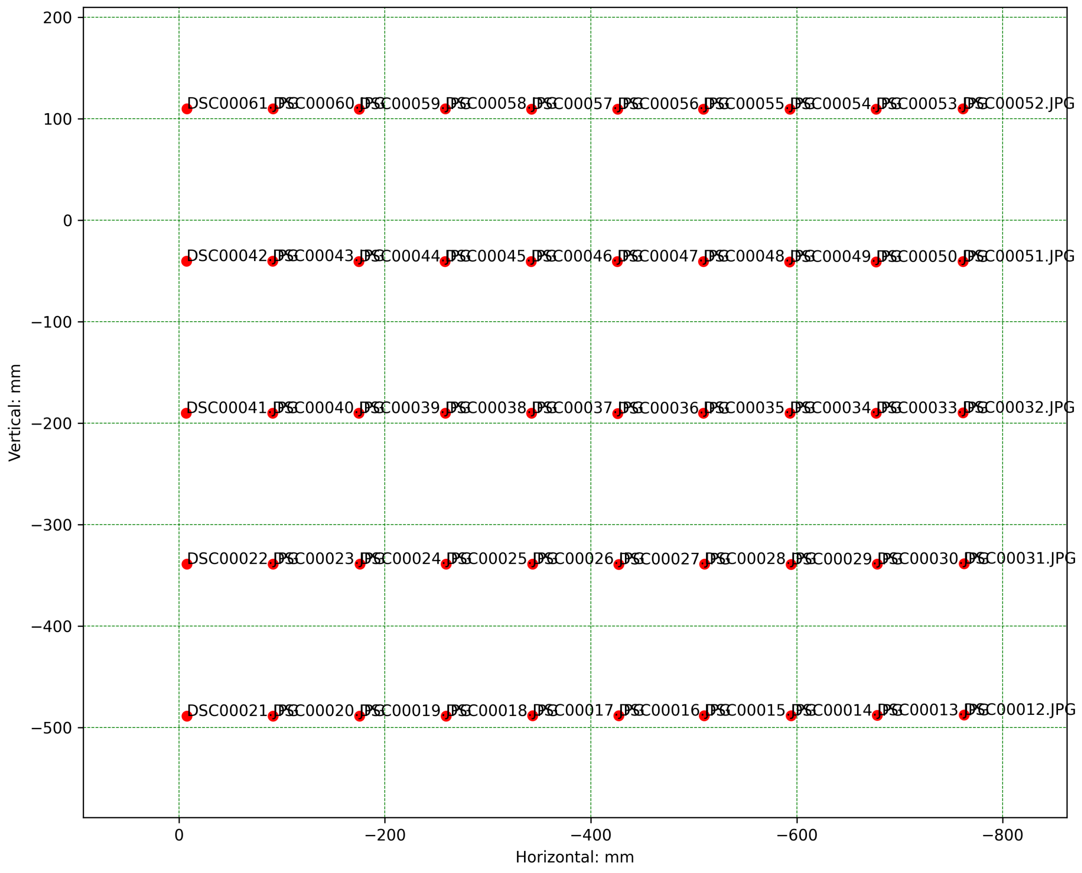


Figure A.4: The camera positions calculated by estimated extrinsic parameters in 2D space. The red points show the camera positions. The text next to the red dots refers to the names of the images taken at the corresponding positions.

A.2 Point Cloud and Surface Recovery

We performed a multi-view re-projection matching algorithm on the image dataset and obtained the point cloud of the real fossil object surface. We selected a region of interest (ROI) in the reference image that includes key textures on the surface of the fossil (See Fig.A.5).

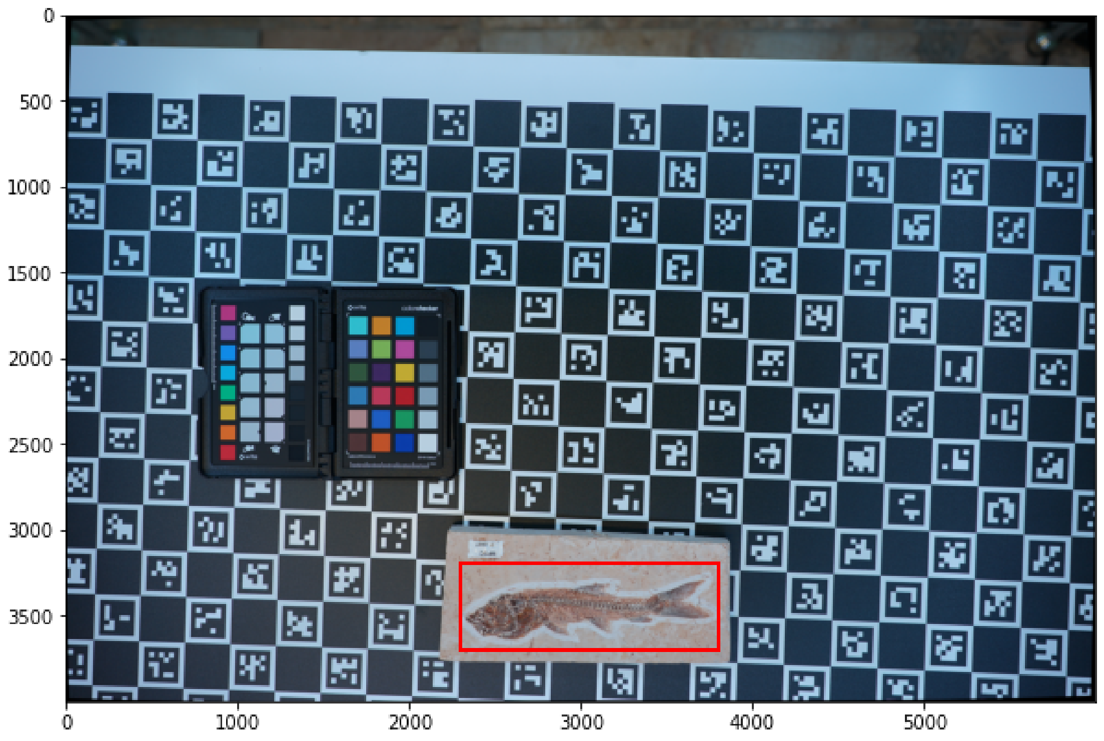
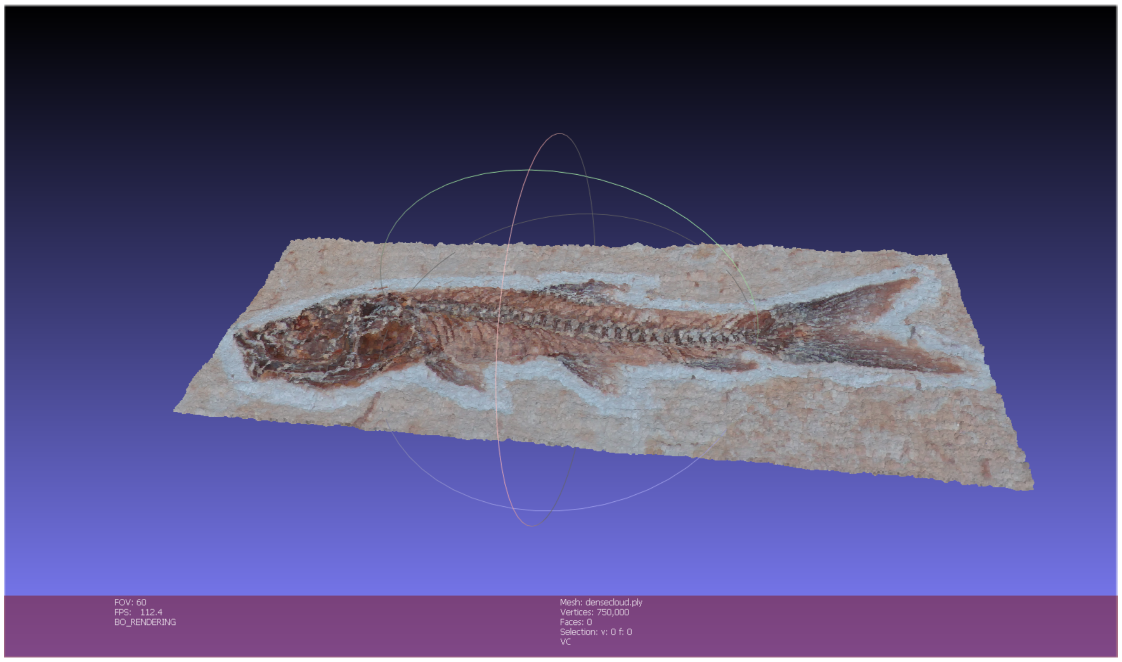


Figure A.5: The red rectangle shows the region of interest in the reference image. The unimportant background is excluded.

Then we re-project all the pixels within this ROI to other images to estimate the depth of these pixels and find their corresponding positions in 3D space. We visualize the generated point cloud via MeshLab at Fig.A.6.

Once we obtain the point cloud, we apply both the Ball Pivoting Algorithm and Poisson Reconstruction Algorithms to the point cloud to recover the surface of



```
FOV: 60
FPS:   112.4
BO_RENDERING
```

```
Mesh: densecloud.ply
Vertices: 750,000
Faces: 0
Selection: v: 0 f: 0
VC
```

Figure A.6: The dense point cloud for real fossil object. It has textures similar to those found on real fossils.

the fossil object (See Fig.A.7). We measure the number of points and triangles of the surface generated, as well as the time cost of both algorithms. We set octree depth as ten for Poisson Reconstruction. The results are shown in Table A.1.

Table A.1: Comparison between BPA and Poisson reconstruction on real fossil object. At the same order of magnitude of vertices, the Poisson method generates more triangle surfaces and takes less time.

Method	Number of Points	Number of Triangles	Time Cost (s)
BPA	750,000	536,937	168.1
Poisson	524,944	1,049,707	9.7

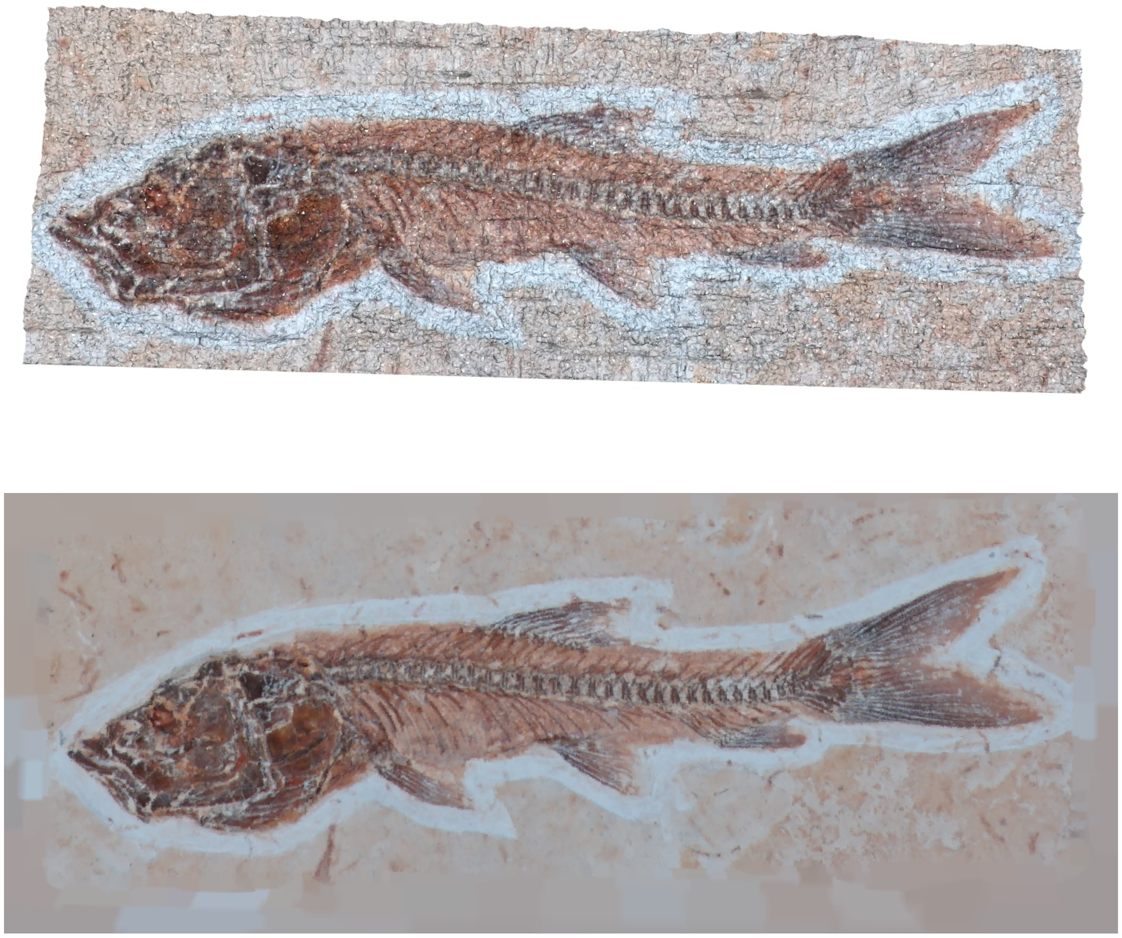


Figure A.7: The results of surface recovery for the real fossil object. Top: The result of the Ball Pivoting Algorithm. Bottom: The result of Poisson Reconstruction Algorithm.

A.3 Discussion

Based on the results of the experiments above, we find that our algorithm does not perform well on real fossil objects. Here, we shall discuss the problem of adaptation from artificial to real objects. Upon observation of the texture in detail, we find that the raised portion of the fishbone is not well reconstructed. Thus, we look into each step of our pipeline and raise several possible deficiencies below.

First, the metric we choose in the matching process heavily affects the depth estimation accuracy. In our pipeline, we choose pure L2 loss to find the best matching with the least square error. L2 loss is sensitive to outliers and extremely harmful to dense point cloud generation. It would be a good idea to use regularization terms in this part to enhance the robustness of our matching algorithm.

Second, the depth fusion method we used also affected the final results. In our experiments, we adopted the average fusion method after filtering outliers. It works well in flat and irregular objects but cannot perform equally well on real fossils since they contain more small-scale textures. Applying a more complex strategy to fuse the depth we estimate from image pairs would be a valuable enhancement to our system.

Third, the size of the sliding window is too small during the matching process. A small sliding window can easily produce incorrect matches when very similar texture patterns are present, leading to incorrect depth estimation. To improve this, multiple sliding window scales should be used for pixel matching in future experiments.

In conclusion, we have realized a complete fossil reconstruction system from end to end and provided an effective and extensible framework. Improving the accuracy and robustness of each step in our pipeline will be important tasks in our future work.

Bibliography

[1] A method for the investigation of fossils by serial sections. *Proceedings of the Royal Society of London*, 72(477-486):98–98, Jan. 1904.

[2] G. An, S. Lee, M.-W. Seo, K. Yun, W.-S. Cheong, and S.-J. Kang. Charuco Board-Based Omnidirectional Camera Calibration Method. *Electronics*, 7(12):421, Dec. 2018.

[3] K. T. Bates, P. L. Manning, B. Vila, and D. Hodgetts. THREE-DIMENSIONAL MODELLING AND ANALYSIS OF DINOSAUR TRACKWAYS. *Palaeontology*, 51(4):999–1010, July 2008.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. Series Title: Lecture Notes in Computer Science.

[5] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics*, 32(2):1–17, Apr. 2013.

[6] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The

ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, Oct. 1999.

[7] R. Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–363, 1996.

[8] G. C. Conroy and M. W. Vannier. Noninvasive Three-Dimensional Computer Imaging of Matrix-Filled Fossil Skulls by High-Resolution Computed Tomography. *Science*, 226(4673):456–458, Oct. 1984.

[9] J. A. Cunningham, I. A. Rahman, S. Lautenschlager, E. J. Rayfield, and P. C. Donoghue. A virtual world of paleontology. *Trends in Ecology & Evolution*, 29(6):347–357, June 2014.

[10] P. Davis, M. Horn, L. Schrementi, F. Block, B. Phillips, E. M. Evans, J. Diamond, and C. Shen. Going Deep: Supporting Collaborative Exploration of Evolution in Natural History Museums. page 8.

[11] P. C. J. Donoghue, S. Bengtson, X.-p. Dong, N. J. Gostling, T. Huldtgren, J. A. Cunningham, C. Yin, Z. Yue, F. Peng, and M. Stampanoni. Synchrotron X-ray tomographic microscopy of fossil embryos. *Nature*, 442(7103):680–683, Aug. 2006.

[12] E. Eade. Gauss-newton/levenberg-marquardt optimization. *Tech. Rep.*, 2013.

[13] C. et. al. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[14] P. C. et. al. MeshLab.

[15] P. Falkingham. Acquisition of high resolution three-dimensional models using free, open-source, photogrammetric software. *Palaeontologia Electronica*, 2012.

[16] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[17] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, June 2014.

[18] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6, Manchester, 1988. Alvey Vision Club.

[19] R. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[20] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[21] H. Hirschmuller and D. Scharstein. Evaluation of Cost Functions for Stereo Matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, USA, June 2007. IEEE.

[22] D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. Calonder, V. Lepetit,

C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314, pages 778–792. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. Series Title: Lecture Notes in Computer Science.

[23] M. Jaiswal, J. Xie, and M.-T. Sun. 3D object modeling with a Kinect camera. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–5, Chiang Mai, Thailand, Dec. 2014. IEEE.

[24] Joblib Development Team. Joblib: running python functions as pipeline jobs, 2020.

[25] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson Surface Reconstruction. *Symposium on Geometry Processing*, page 10 pages, 2006. Artwork Size: 10 pages ISBN: 9783905673241 Publisher: The Eurographics Association.

[26] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, June 2013.

[27] S. Koch, Y. Piadyk, M. Worchel, M. Alexa, C. Silva, D. Zorin, and D. Panozzo. Hardware design and accurate simulation of structured-light scanning for benchmarking of 3d reconstruction algorithms. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[28] M.-H. Le, B.-S. Woo, and K.-H. Jo. A comparison of sift and harris conner features for correspondence points matching. In *2011 17th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, pages 1–4, 2011.

[29] Y. G. Leclerc, Q.-T. Luong, and P. Fua. Measuring the Self-Consistency of Stereo Algorithms. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Computer Vision - ECCV 2000*, volume 1842, pages 282–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. Series Title: Lecture Notes in Computer Science.

[30] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[31] K. Madsen, H. Nielsen, and O. Tingleff. *Methods for Non-Linear Least Squares Problems (2nd ed.).* 2004.

[32] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, Apr. 1993.

[33] L. Puggelli, R. Furferi, and L. Governi. Low Cost Device to Perform 3D Acquisitions Based on ChAruCo Markers. In C. Rizzi, A. O. Andrisano, F. Leali, F. Gherardini, F. Pini, and A. Vergnano, editors, *Design Tools and Methods in Industrial Engineering*, pages 189–200. Springer International Publishing, Cham, 2020. Series Title: Lecture Notes in Mechanical Engineering.

[34] R. Raguram, J.-M. Frahm, and M. Pollefeys. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5303, pages 500–513. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. Series Title: Lecture Notes in Computer Science.

[35] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient

alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, Barcelona, Spain, Nov. 2011. IEEE.

[36] K. V. Ryabinin and M. A. Kolesnik. Adaptive Scientific Visualization Tools for a Smart Paleontological Museum. *Programming and Computer Software*, 45(4):180–186, July 2019.

[37] J. D. Schiffbauer and S. Xiao. Paleobiological Applications of Focused Ion Beam Electron Microscopy (FIB-EM): An Ultrastructural Approach to the (Micro)Fossil Record. In M. Laflamme, J. D. Schiffbauer, and S. Q. Dornbos, editors, *Quantifying the Evolution of Early Life*, volume 36, pages 321–354. Springer Netherlands, Dordrecht, 2011. Series Title: Topics in Geobiology.

[38] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[40] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, July 2006.

[41] P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, pages 432–437, Fort Collins, CO, USA, 1999. IEEE Comput. Soc.

[42] D. Viswanathan. Features from accelerated segment test (fast). 2011.

[43] J. Wind. Computerized X-ray tomography of fossil hominid skulls. *American Journal of Physical Anthropology*, 63(3):265–282, Mar. 1984.

[44] F. Windhager, P. Federico, G. Schreder, K. Glinka, M. Dork, S. Miksch, and E. Mayr. Visualization of Cultural Heritage Collection Data: State of the Art and Future Challenges. *IEEE Transactions on Visualization and Computer Graphics*, 25(6):2311–2330, June 2019.

[45] B. Winkler. Applications of Neutron Radiography and Neutron Tomography. *Reviews in Mineralogy and Geochemistry*, 63(1):459–471, Jan. 2006.

[46] F. Zhou, Y. Cui, Y. Wang, L. Liu, and H. Gao. Accurate and robust estimation of camera parameters using RANSAC. *Optics and Lasers in Engineering*, 51(3):197–212, Mar. 2013.

[47] A. Ziegler, M. Kunth, S. Mueller, C. Bock, R. Pohmann, L. Schröder, C. Faber, and G. Giribet. Application of magnetic resonance imaging in zoology. *Zoomorphology*, 130(4):227–254, Dec. 2011.