ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Image acquisition, 3D reconstruction and a visual interactive digital heritage system

---

Team #9

CHUANRUI CHEN
(cc86@illinois.edu)
DENGLIN CHENG
(denglin3@illinois.edu)
QIANYAN SHEN
(qianyan2@illinois.edu)
ZIYING LI
(ziyingl4@illinois.edu)

Sponsors: Prof. Shurun Tan

May 31, 2024

Abstract

Through 3D reconstruction technology, real-world scenes are transformed into digital models, enabling intuitive interaction and analysis. In the field of cultural heritage preservation, this technology assists in preserving historical information, restoration, and protection, while providing researchers with visualization tools. Given the high cost and complexity of traditional techniques, it is crucial to develop simplified and cost-effective modeling methods. This project proposes a cultural heritage scanning system, consisting of scanning hardware, point cloud reconstruction software, and a database interaction platform, with the aim of preserving cultural artifacts in fine detail and promoting cultural exchange.

Contents

1 Introduction

1.1 Purpose

Cultural artifacts possess significant historical, cultural, and artistic value. However, due to the passage of time and the impact of natural deterioration, many artifacts face risks of damage, loss, or decay. During the course of study, exhibition, and inheritance, artifacts may also sustain deterioration and damage to cultural heritage [1]. Additionally, for history enthusiasts and researchers worldwide, detailed information about specific artifacts is not readily accessible. Furthermore, traditional photographs often fail to capture the intricate details of artifacts, hampering comprehensive research and preservation efforts. Therefore, the conservation and exhibition methods of cultural relics have been a common concern of scholars around the world.Therefore, our team aims to develop a system that can generate realistic 3D models of cultural artifacts and provide users with a user-friendly interactive interface for immersive exploration.

We plan to design a system that can capture the detailed geometric shapes of artifacts using advanced scanning and 3D reconstruction techniques, and create 3D models. Additionally, we need to establish a database to store the collected artifact information and design a user-friendly interface that allows users to easily browse and interact. This will enable us to accurately capture and preserve the features of artifacts and provide a platform for enthusiasts to interact with artifacts up close.

Based on the analysis above, our first requirement is an accurate and efficient system for collecting the visual information of artifacts. We need to address how to handle the positional relationship between artifacts and cameras, as well as how to convert RGBD data into 3D models. Therefore, we need a mechanical device that can control the position and angle of the camera relative to the artifacts. This will help us obtain accurate RGBD data. Secondly, we need an efficient and accurate system to convert RGBD images into 3D models. To address this issue, we have employed a point cloud reconstruction method. Firstly, we obtain point clouds from the RGBD images. After applying filtering and registering operations to the point clouds, we obtain a complete point cloud. Subsequently, we do reconstruction on the point cloud, resulting in a meshed model of the artifact that includes color information.

Nevertheless, considering our goal of better preserving, studying, and disseminating traditional cultural heritage, having only the 3D model data of artifacts is not enough. We also need to build a platform to store and render the models and provide interactive possibilities for users. We will export the reconstructed 3D models and load them into the database subsystem. Users can search for artifacts of interest in the database, and the rendered model data from the search results will be displayed in the interactive interface. Users can then appreciate and study the details of the artifacts up close through operations such as zooming and rotating.

By digitizing cultural heritage and sectors like education and tourism, our technology can generate greater economic benefits and serve as a strong foundation for the digital display, dissemination, and preservation of historical relics [2].

1.2 Visual Aid

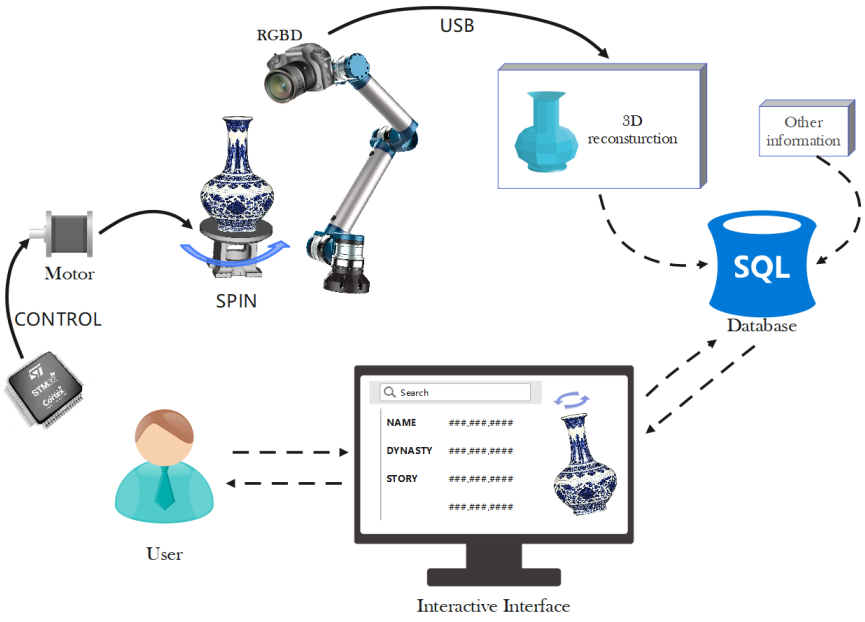


Figure 1: Visual Aid for the whole project

1.3 Functionality

- **Overall Outcomes:**

  The system is capable of reconstructing objects with dimensions (including length, width, and height) ranging from 5cm to 20cm.

  The dimensional accuracy of the obtained 3D models is within 5% error precision.

  Our system has the capacity to store information on 2500 artifacts.

- **Modeling Outcomes and User Experience:**

  The database securely stores information of cultural artifacts and their corresponding 3D models.

  The system responds to searches within 3 seconds, displaying the 3D models and artifact information.

  Users can search for specific heritage items using keywords on the website.

- **Hardware Level:**

  The minimum response time for real-time control is less than 200 milliseconds.

  The rotation accuracy is within ±1 degree.

  The rotation speed can reach up to 15 degrees per second.

1.4 Subsystem Overview

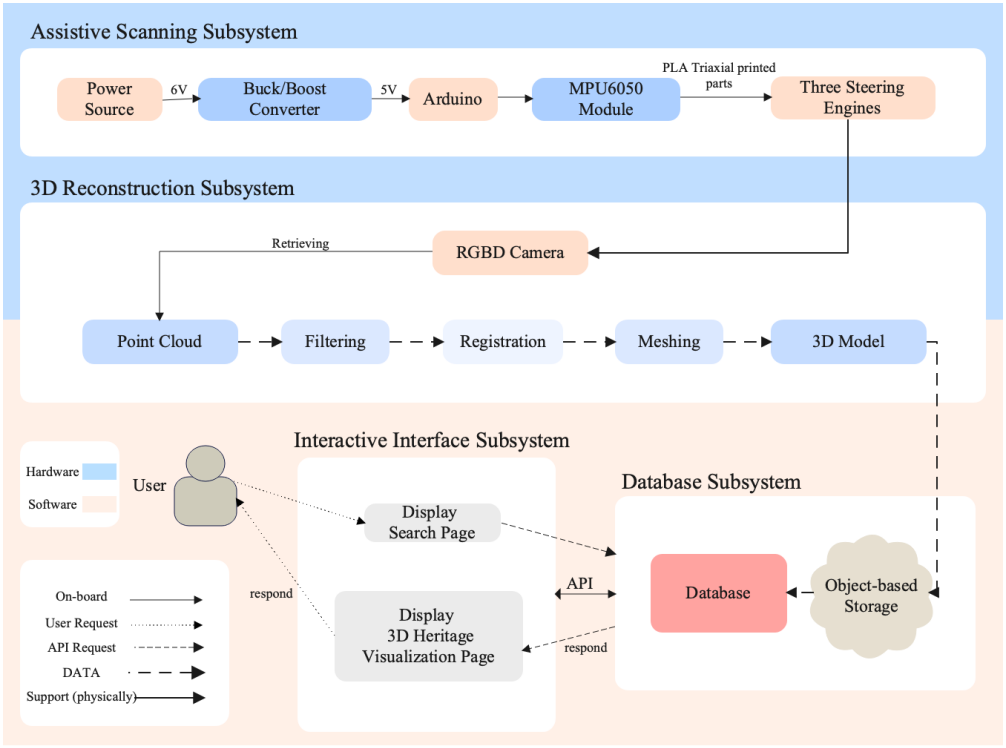


Figure 2: Block Diagram

Our system consists of four subsystems: the Assistive Scanning subsystem, based on hardware, designed with a mechanical gimbal to support the camera for easy positioning of the camera and artifacts. The 3D Reconstruction subsystem obtains data from RGBD cameras, generates point clouds, and completes the reconstruction process. The Database subsystem stores 3D models and artifact information, providing basic functionalities such as adding, deleting, querying, and modifying data. The Interactive Interface subsystem handles webpage design, model rendering, and basic interactions.

2 Design

2.1 Subsystem Overview

2.1.1 Assistive Scanning Subsystem

In the evolving field of dynamic imaging technology, precision and stability are paramount. The design and implementation of a self-stabilized gimbal represent a sophisticated blend of mechanical design, electronic control, and software engineering aimed at achieving high-quality motion compensation for cameras. This paper explores the intricate mathematical principles and control mechanisms that form the backbone of a gimbal's ability to maintain camera stability even when its base is in motion.

At the heart of the gimbal's control system lies the MPU6050 sensor module, which serves as the pivotal element for orientation detection. This sensor is adept at measuring both angular velocity and linear acceleration via its built-in gyroscope and accelerometer. The data obtained from the MPU6050 are critical in computing the necessary adjustments required to maintain the camera's orientation relative to a reference plane. The gimbal structure comprises three servo motors aligned along the Z, Y, and X axes. These motors are responsible for the precise manipulation of the camera's positioning, ensuring that the camera plane remains parallel to the base plane at all times. The alignment and operation of these servos are crucial for counteracting any abrupt or gradual movements of the base, thereby stabilizing the visual field of the camera. The effectiveness of a gimbal in stabilizing images relies heavily on the real-time processing capabilities of its control system. This system calculates the deviations from the desired orientation by continuously monitoring the sensor outputs. An advanced algorithm interprets these deviations and converts them into control signals that adjust the servos on the Z, Y, and X axes.

The Arduino code forms the operational core of the gimbal, interfacing directly with the hardware to execute the control strategies developed. Initialization routines prepare the system by setting up communication protocols via the I2C interface, configuring the MPU6050 sensor, and calibrating the servo motors to their neutral positions. The main operational loop of the code listens for data from the MPU6050. When new data is detected, it processes this information to determine the current orientation of the camera. Depending on how this real-world orientation deviates from the target, the code calculates the necessary adjustments and commands the servo motors to realign the camera appropriately. The practical implications of these mathematical and control strategies are profound in applications requiring high precision and stability in image capture, such as in aerial photography, filmmaking, and robotic vision systems.

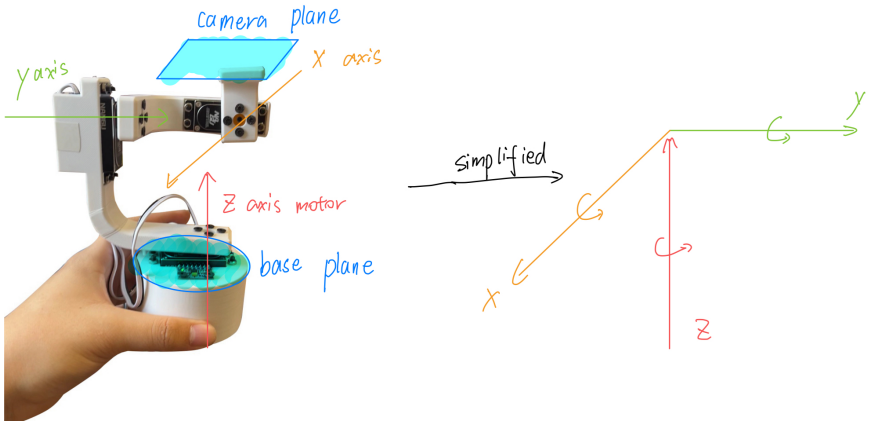


Figure 3: Schematic Representation of the Gimbal Mechanics and Sensor Placement

Figure 4: Structural Failure Analysis and Reinforcement

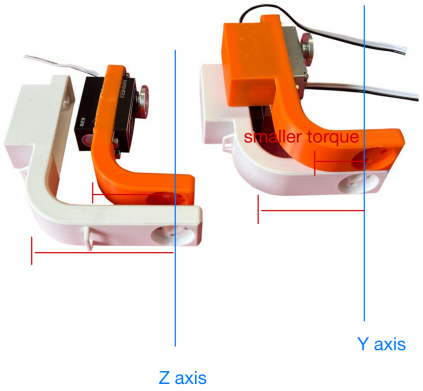


Figure 5: Initial and Upgraded 3D Printed Components

The design and fabrication of the 3D-printed parts for this self-stabilized gimbal illustrate the iterative nature of engineering prototypes, highlighting the blend of practical testing and theoretical design. This section provides a comprehensive overview of the 3D printed components, the challenges encountered, and the iterations made to enhance the gimbal's performance and durability. The gimbal's structure comprises primarily white and yellow parts, all of which were meticulously designed and printed using a 3D printer. These components include the arms and mounts for the servo motors, as well as the chassis that houses the electronic components and provides a stable base for the gimbal mechanism.

One of the critical challenges faced during the testing phase was the breakage of a servo motor arm on one of the white parts. This incident occurred during a test run where the battery's failure led to a sudden loss of power, resulting in insufficient torque being delivered by the servo motor. Consequently, the entire assembly, including the gimbal and the camera, crashed to the ground. This event prompted a reevaluation of the structural integrity of the gimbal. In response to the breakage, the thickness at the fracture point was increased. This modification was aimed at providing additional strength where the servo motors are mounted with M3 screws and nuts, ensuring that the parts could withstand greater forces during operation. This adjustment is highlighted in the images of the white and yellow parts, which show the areas that were thickened to prevent future failures. The incident also led to considerations regarding the operational torque of each servo motor. To achieve a more stable operation and minimize the risk of mechanical failures, the torque settings of the servo motors were adjusted.

The circuit diagram provided illustrates the complete electronic configuration for controlling the self-stabilized gimbal, featuring a dual 3.7V Li-ion battery setup connected in series for enhanced voltage suitable for powering the servos and the control logic. Integrated into this system is a TP4056 Type-C charging module for efficient battery management and recharging capabilities. Voltage regulation is achieved via a dedicated con-

verter, ensuring that both the Arduino Nano and the servos operate at optimal voltage levels. The Arduino Nano serves as the brain of the operation, processing input from the MPU6050 sensor, which tracks orientation and motion. Based on this data, the Arduino dynamically adjusts the positions of the servos across the X, Y, and Z axes to stabilize the camera effectively against movements.

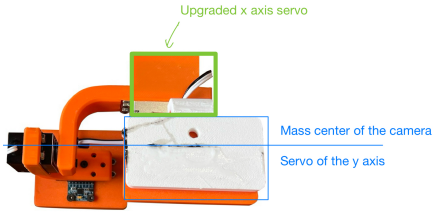


Figure 6: Mass center of the camera

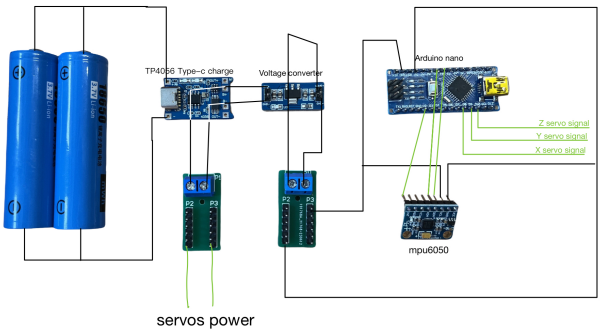


Figure 7: circuits with physical PCB

2.1.2 3D Reconstruction Subsystem

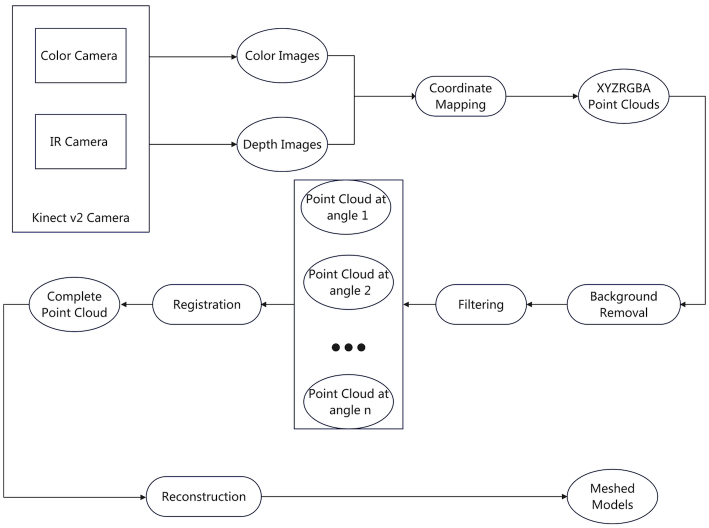


Figure 8: Block Diagram for 3D Reconstruction Subsystem

This subsystem aims to acquire point clouds through RGBD images and utilize them for 3D reconstruction.

The system takes depth images and colour images as input from the RGBD camera supported by the Assistive Scanning Subsystem. In order to generate 3D models with lower

errors, the system not only needs to acquire point clouds and perform 3D reconstruction on them but also employs various algorithms for denoising and smoothing the point clouds.

We choose Kinect v2 as our RGBD camera. We need to obtain the depth and colour frame sequences of the rotating objects using the Kinect SDK, and then map the pixels on each depth image to 3D space through coordinate transformation to obtain the initial point cloud. Since we also need the artifact texture information, we also need to perform a mapping between the depth image and the colour image to obtain the colour information for each 3D point. Then we store the spatial position and colour information of each point in the point cloud created by PCL or Open3D.

(a) 1920 * 1080 input Color Image

(b) 512 * 424 input Depth Image

(c) Retriving Point Cloud

Figure 9: Chair Example for Point Cloud Retriving

After obtaining the initial point cloud, we first need to separate the point cloud of the desired object from the background. In order to do this, we need to simultaneously acquire a background point cloud without the target object A and a point cloud with the target object B. Then we need to register the point cloud A and B. The result is two point clouds with a uniform coordinate system. We make a difference between the two point clouds to get a different region [3]. Then a clustering operation is performed on this point cloud difference to get the point cloud cluster with the closest distance to the center of the object and the highest number of points, i.e., the point cloud data of the target object.

Figure 10: Point clouds of chair from different viewpoints

After segmentation the point cloud is first denoised. Statistical methods are used to find the Outlier. Specifically a statistical analysis of the neighborhood of the points is needed

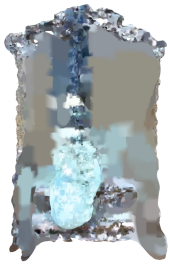
to eliminate the points that do not meet specific criteria [4]. We calculate the average distance from each point to all its neighbors. The resulting distance distribution is assumed to follow a Gaussian distribution with a mean and a standard deviation. Any point that exceeds the interval defined by the standard deviation can be recognized as an outlier and subsequently removed.

Since the scans are performed from different viewpoints, the obtained point clouds often have separate local coordinate systems and need to be converted to uniform global coordinates, a process known as point cloud registration [5]. We perform Coarse registration first and Fine registration later. The core idea is to iterate the nearest point method to make two point clouds close to each other, and finally minimize the distance error between two point clouds. However, the problem of this operation is that it is easy to fall into the local optimization and lack of overlapping region, which leads to poor alignment accuracy. Therefore, it is more suitable for objects with distinctive structural features. For objects with regular shapes (e.g. vases and water bottles), we need to select overlapping regions with clear features for registration, and then apply the Transformation matrix obtained from the registration to the source point cloud.

Figure 11: Registered Point clouds of chair from different viewpoints

We use the Ball Pivoting Algorithm (BPA) and the Poisson Surface Reconstruction Algorithm for reconstruction part. The BPA algorithm [6] starts from a seed triangle and defines a rolling ball with radius r. The ball is rolled along the edges, and it is determined whether there are other points inside the ball to form a new triangle. repeated until all points are included in the triangle mesh. The BPA algorithm works well for uniformly distributed point clouds, but it becomes challenging to choose an appropriate rolling ball radius for non-uniform point clouds, which affects the quality of the 3D model.

(a) Reconstruction Result of BPA



(b) Reconstruction Result of Poisson

(c) Reconstruction Result of Poisson removing low densities part

(d) Real Color image of This Ornament (mirrored)

Figure 12: Example for Reconstruction

Poisson Surface Reconstruction [7] is a commonly used 3D reconstruction method for generating smooth surface models from point cloud data. It is based on the numerical solution of Poisson's equation, which infers the interior of the surface by considering the point cloud data as boundary conditions of the surface. The advantage is the ability to handle irregularly distributed point cloud data and generate smooth surface models. The disadvantages are but slower processing speeds and the inference of non-existent regions in low-density point cloud regions, which need to be additionally removed.

Existing commercial depth sensors assume that target objects have Lambertian reflective surfaces. These depth sensors cannot work properly when facing objects that produce strong specular reflections or transparent objects. This is because the surfaces of these objects cannot provide uniform diffuse reflection, resulting in a failure of depth measurement [8]. Therefore, the major defect of this subsystem, and the main defect of existing depth sensors, is the inability to work properly on non-Lambertian reflective surfaces, which limits the system's application in a wider range of scenarios. To address this defect, we can propose an alternative approach using artificial intelligence methods, such as Neural Radiance Fields (NeRF) [9]. By training on RGB images of objects taken from different viewpoints and camera poses, NeRF learns a radiance field function that represents the color value and volume density corresponding to any spatial position and viewing direction. During the optimization process of minimizing the error between the radiance integral projected onto each pixel and the input RGB images, the algorithm can effectively capture any visual phenomena, including non-Lambertian surfaces.

2.1.3 Database Subsystem

The database subsystem is responsible for two main things: first, it aims to store the basic information of the artifacts, including countries, historical backgrounds, etc., and at the same time save the generated complex 3D model data; and, it deploys the back-end of the website so that the front-end can retrieve information from the database.

The data generated by the 3D Reconstruction Subsystem will be loaded into the Database Subsystem. With this database, users can search and view artifacts from exotic countries. Based on this requirement, we opted for **Relational Cloud Database RDS MySQL (RDS)**. However, it is primarily designed for storing structured data and is generally not suitable for directly storing large files or unstructured raw file content. Therefore, we also adopted **Object Storage Service (OSS)** to address this need. OSS can store files of any type, including OBJ format files generated by 3D modeling software. OSS provides data upload, download, management, and distribution services based on HTTP RESTful API, allowing OBJ format files to be directly uploaded to OSS buckets [10]. RDS can store URL links pointing to OBJ files in OSS. This allows RDS to record the storage location information of OBJ files in OSS, achieving data association between the relational database and object storage. Therefore, in our Database Subsystem, RDS stores metadata of the model (such as model ID, name, historical background, etc.) and the URL link, while OSS stores the model files themselves. By recording the URL or other reference information of OSS objects in RDS, it enables the database to locate and retrieve the corresponding OBJ model files during queries.

Our database design contains one table, ”Artifacts”, which has ”ID” as the primary key. The attribute ”Link” represents the link to the corresponding OBJ files returned by the object-based storage. ”Name”, ”Year”, ”Country” and ”Historical background” are vital information about artifacts. The relationship is shown in the ER diagram below:

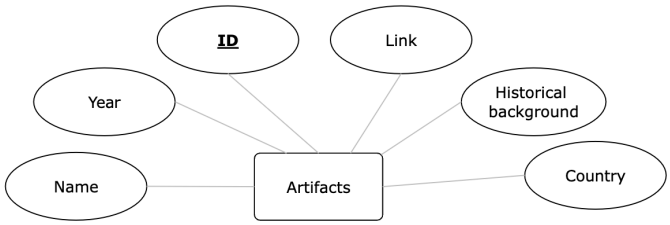


Figure 13: Entity-Relationship Diagram

In addition, for the creation and management of running virtual server instances, we also utilized **Elastic Compute Service (ECS)**. With 1M external bandwidth, it can meet the concurrent access of 10-15 people online at the same time. These servers not only support concurrent user accesses but are also highly flexible, accommodating various operating systems and application software for easy scalability. They can seamlessly adapt to changes in resource demands due to business growth [11]. We have selected the **BT Panel image** for ECS, which is a server management software supporting both Windows and Linux systems. It allows for easy server management through a web interface, enhancing operational efficiency. To enable internal network communication between ECS instances and RDS instances, the network type of the RDS instance is fixed to a Virtual Private Cloud (VPC) and configured to be the same VPC as the ECS instance. Moreover, the private IP address of the ECS instance must be added to the IP whitelist of the RDS

instance to allow the ECS to access the RDS instance properly [12].

To deploy our website, I have chosen **PM2**, an open-source application process manager based on Node.js [13]. PM2 includes a comprehensive set of features such as daemonization, monitoring, and logging. If a Node.js application crashes or stops for any reason, PM2 can automatically restart the application, ensuring continuous availability of the service. PM2 allows developers to update Node.js applications to new versions without stopping the current service, which is crucial for services that need to run 24/7. Additionally, PM2 provides log management functionality, making it more convenient to track and debug applications in a production environment. Once the ports configured in the ECS security group are added to PM2, our website will be accessible normally.

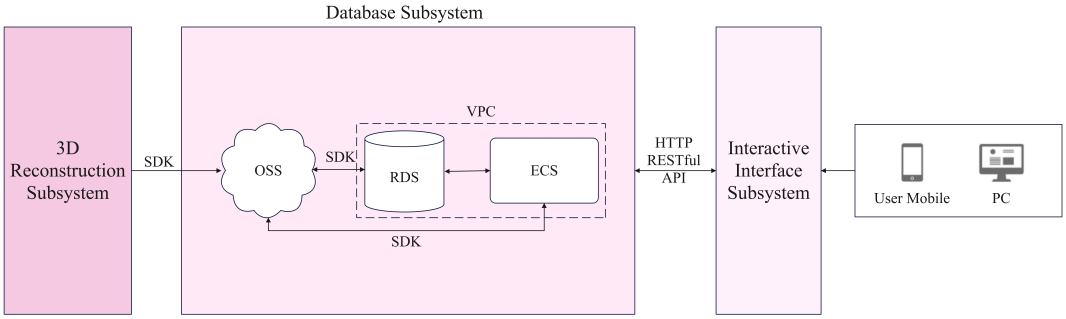
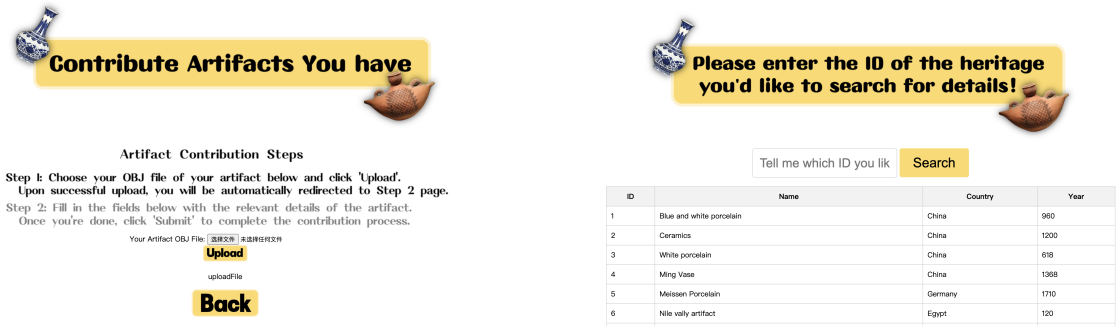


Figure 14: Overview of Database Subsystem

In order to provide a more detailed explanation of the interaction process between the Database Subsystem and other subsystems, I will proceed to separately explain the interaction between the Database Subsystem, the 3D Reconstruction Subsystem, and the Interaction Interface Subsystem.

Regarding the interaction process between the Database Subsystem and the 3D Reconstruction Subsystem, this process is realised through the corresponding page on the website (as shown in figure 15a below), where Obj files from 3D Reconstruction Subsystem uploaded through the upload button on the website are uploaded to the node.js server via the **Multer storage engine**, and then uploaded to the OSS via the Alibaba Cloud ali-oss SDK. After that, OSS generates a URL signature which is then stored in RDS with SDK.



(a) Web page for showing information on all artefacts from the database

(b) Web page for showing information on all artefacts from the database

Figure 15: Database-related web pages.

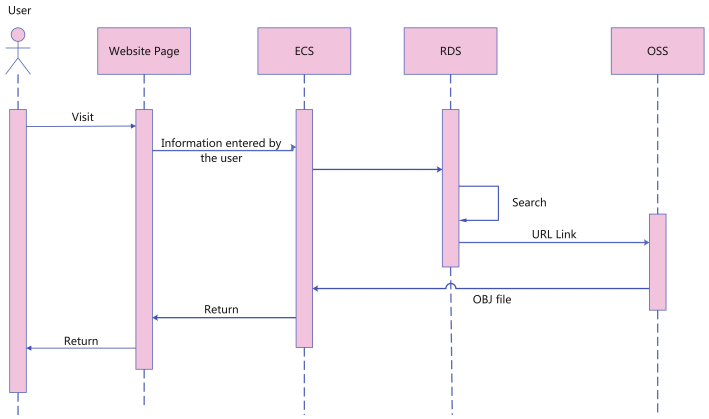


Figure 16: User search sequence diagram

For interaction between Database Subsystem and the Interaction Interface Subsystem, the detailed process is shown in figure 16. Users first input the ID of the artifact they want to search for on our website. Then, ECS forwards this information to RDS, which retrieves and finds the corresponding artifact information from its own database. If the retrieval fails, an error is returned to the webpage. If successful, the corresponding URL link is used to retrive the OBJ file. The website takes all the information about the artefacts from the RDS and displays it in a tabular form on the web page, and the corresponding page is shown according to figure 15b.

2.1.4 Interactive Interface Subsystem

The Interactive Interface Subsystem is responsible for facilitating user interaction with the visual heritage system. It provides a graphical user interface (GUI) through which users can search for artifacts, view relevant information, and explore 3D models. This subsystem interprets user requests, retrieves data from the Database Subsystem, and renders 3D models along with accompanying information. It also incorporates interactive control

functions to enhance the user experience.In the design of the Interactive Interface Subsystem, we opted to utilize the Express framework combined with WebGL technology to achieve graphical rendering and web development.

The workflow of the Interactive Interface Subsystem involves two crucial stages. Firstly, user interaction design ensures an intuitive interface, allowing easy browsing of artifacts and interaction with 3D models. Upon user requests, the subsystem communicates with the Database Subsystem to retrieve artifact information, including OBJ format 3D model files. We use Express framework here. The Express framework is a JavaScript library that supports a component-based development approach. This method simplifies the construction of user interfaces, making them more modular and enhancing application performance and maintainability.

Secondly, rendering 3D model files onto the user's screen is critical. This involves handling vertex, texture, and normal information from OBJ files and applying shaders for visualization. Here we choose to use WebGL as the API. This web-based graphics rendering technology allows the creation of 3D and 2D graphics using JavaScript and the OpenGL API within a web browser. As WebGL operates on the underlying GPU, we use GLSL language to implement paired vertex and fragment shaders. The vertex shader calculates the position of vertices, while the fragment shader computes the color value of each pixel in the currently drawn primitive. WebGL then rasterizes primitives and then shaders automatically perform viewport mapping to convert to Screen Space.

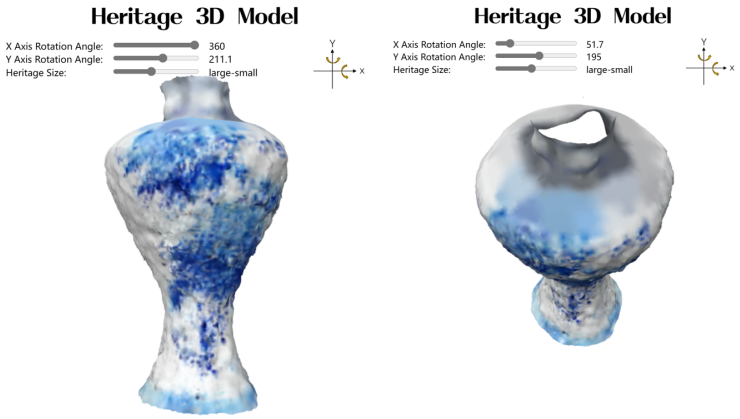


Figure 17: Website 3D Model Example

The following words show technology details of this subsystem. OBJ file is a standard format for representing 3D geometry. Parsing OBJ files involves extracting essential data like vertex positions, texture coordinates, normals, and colors. The process systematically handles each line of the OBJ file, identifying keywords (including v, vn, f, etc.) and their corresponding arguments, and organizes the parsed data into suitable data structures for utilization in WebGL. For each line in the OBJ file, the line is parsed to extract the keyword and arguments. If the keyword is a known OBJ keyword, the corresponding handler function is called with the arguments; otherwise, the line is skipped. After processing all lines, empty arrays are removed from the geometries, and the material libraries and

geometries are returned. These information will be stored in the buffer if needed and GLSL shader will operate them.

Handling and rendering OBJ files on the webpage involve several detailed steps. JavaScript parses the obj file as previously mentioned. WebGL is prepared by creating a canvas element and acquiring context, storing parsed data in buffers for 3D model rendering. The camera's position, orientation, and projection matrix are established for scene projection. Interactive features are implemented, allowing users to dynamically manipulate the scene via sliders, controlling rotation angles and camera position. There are three sliders on the screen, including X-axis rotation, Y-axis rotation, and model size. The model can rotate along the x-axis and y-axis both for 360 degrees, with a step of 0.1, and the size slider is used to zoom in and zoom out. Subsequently, WebGL draws the scene by passing vertex data, shader programs, and camera projection information to the GPU. Continuous updates ensure real-time responsiveness to user interactions by computing the new projection (based on new model size slider input) and view matrices (based on new x and y rotation angle slider inputs). Figure 17 is a 3D model example on the webpage.

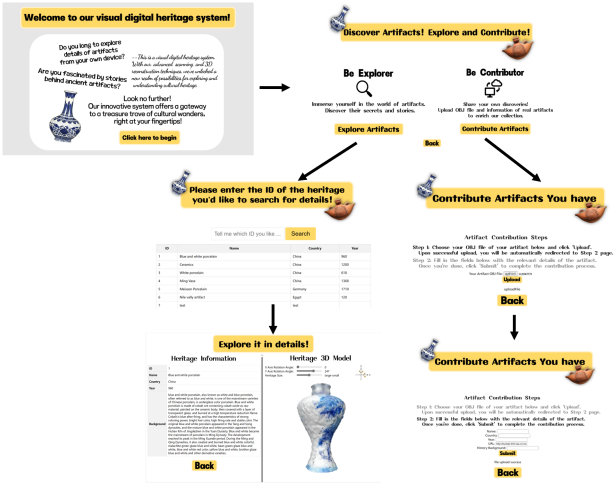


Figure 18: Website UI Structure

Finally, the design of website structure offers a seamless journey. The structure is shown in figure 18. Users start at the welcoming page, then proceed to the mode choice page, offering options to explore existing artifacts or contribute new ones. Exploring artifacts leads to a search page with a database of artifacts. Users can locate artifacts quickly by ID, accessing detailed information including ID, name, country, year, background, and interactive 3D models for exploration. Alternatively, users can contribute artifacts by uploading OBJ files and relevant information on a dedicated page. Submitted artifacts enrich the platform's collection, aiding in the preservation of cultural heritage. Consistent navigation via a back button ensures an intuitive experience, allowing users to easily navigate back through their exploration or contribution journey. This streamlined structure ensures a seamless and user-friendly experience, facilitating both exploration and contribution efforts.

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Parts

Table 1: Cost Analysis table listing all parts

Description with Manufacturer and Part #	**Quantity**	**Unit Cost**	**Total Cost**
#GM2804 Brushless Motor with AS5600 Encoder	3	¥103.9	¥311.7
42 Stepper Motor #S42H40D20	3	¥53	¥159
Arduino Nano Development Board	2	¥29.5	¥59
Servo #MG996R	4	¥18	¥72
#MPU6050 Module	2	¥35.7	¥71.4
Dupont Cables	1	¥16	¥16
Microsoft Xbox Kinect Sensor and Adapter	1	¥770	¥770
Alibaba Cloud Elastic Compute Service	1	¥162	¥162
Alibaba Cloud RDS MySQL	1	¥99	¥99
Alibaba Cloud Object Storage Service	1	¥4.98	¥4.98
Website Domain Name Registration	1	¥12	¥12
Total			**¥1737.08**

3.1.2 Labor

According to the 2023 China Undergraduate Employment Report[14], the average monthly salary for undergraduates is 5990 CNY, or 34.56 CNY per hour. four of us in our group work an average of 2 hours a day, 5 days a week. Our weekly salary is $2 \times 5 \times 5 \times 34.56 = 1382.4$ CNY. For a total of nine weeks of work, our salary is $9 \times 1382.4 = 12441.6$ CNY.

3.1.3 Total

Based on the above analysis, we can get the total cost of the whole project is 14178.68 CNY.

3.2 Schedule

Table 2: Schedule

Date	Chuanrui Chen	Denglin Cheng	Qianyan Shen	Ziying Li
Week 1 (3.11 – 3.17)	Decide to use Vue and Visual Studio Code to design UI system.	Select and purchase modules, motors, and other system components.	Buy Kinect and connect it to your computer, install and learn to use the SDK	Discuss with professors and settle the structure of Database Subsystem.
Week 2 (3.18 – 3.24)	Learn based Html, CSS, JavaScript language and build the basic page of website	Design subsystem architecture and planned for future motor upgrades.	Get colour and depth information from Kinect and visualising it	Decide to use RDS, OSS, and ECS servers from Alibaba Cloud; ensure the environment deployment of a MySQL database and the object-based storage.
Week 3 (3.25 – 3.31)	Finish building the html page and try to build the front end structure.	Begin MPU6050 integration and code development for motion tracking.	Get the point cloud from the camera and save it locally	Ensure the functionality of RDS MySQL server and OSS server.
Week 4 (4.1 – 4.7)	Finish building the the front end structure.	Implement MG996R servo motor control and conducted initial tests.	Simple processing of point clouds and exploring registrations	Ensure the interaction between MySQL database instance and backend using cloud server.
Week 5 (4.8 – 4.14)	Make sure the webpage can communicate with database through API.	Calibrate sensors and fine-tune servo responses for accuracy.	Registration and post-processing of point clouds	Ensure the interaction between frontend and backend (stage 1).

Continued on next page

Table 2 – continued from previous page

Date	Chuanrui Chen	Denglin Cheng	Qianyan Shen	Ziying Li
Week 6 (4.15 – 4.21)	Make sure 3D model can be displayed on the webpage smoothly.	Assemble prototype; test MPU6050 and servo interaction.	Explore reconstruction algorithms to reconstruct point clouds	Ensure the interaction between frontend and backend (stage 2).
Week 7 (4.22 – 4.28)	Decorate the webpage and add more functions.	Analyze, optimize performance, and begin stress testing.	Interface with Assistive Scanning Subsystem to organise the whole process	Make sure the functionality of the whole subsystem.
Week 8 (4.29 – 5.5)	Interface with 3D Reconstruction Subsystem and Database Subsystem for the whole process.	Initiate stepper or brushless motor integration and testing.	Interface with Database Subsystem and Interactive Interface Subsystem.	Interface with 3D Reconstruction Subsystem and Interactive Interface Subsystem.
Week 9 (5.6 – 5.12)	Organize and prepare the Final Demo and Report.	Conduct comprehensive tests and prepare the Final Demo and Report.	Organize and prepare the Final Demo and Report.	Organize and prepare the Final Demo and Report.

4 Requirements and Verification

4.1 Assistive Scanning Subsystem

Table 3: Requirements and Verifications table for Assistive Scanning Subsystem

	Requirement Description	Verification Procedure
1	Arduino Nano must receive a stable 5V supply, to maintain operational stability and prevent damage to the board	Verify voltage post-buck converter using a multimeter
2	MPU6050 must accurately capture motion data within manufacturer specifications, ensuring precise control and feedback for motor movement	Conduct calibration tests using Arduino software to verify accuracy of MPU6050 readings against known motion patterns.
3	Servos must respond accurately to control signals within their operational range, to achieve the desired angular positioning for the scanning functionality.	Test servos using a 270 degrees of motion simulation and measure actual angles with a protractor or similar precision instrument.
4	Subsequent versions may use better servos, subject to power and performance requirements.	Compare servo specifications, such as torque and speed, to ensure they meet system requirements for future upgrades

The system meet rigorous performance specifications and ensure robust functionality across its multiple components. Central to its design is the Arduino Nano, which receives a steady 5V supply from a voltage converter, safeguarding the system's stability and the board's integrity. The MPU6050 sensor plays a critical role in motion detection, with its outputs finely tuned to match manufacturer standards through precise calibration processes. This ensures the servos receive accurate data for real-time adjustment of the camera's orientation, allowing the gimbal to respond swiftly and accurately to changes in movement across its operational range.

In enhancing the performance and reliability of system, we explored different servo motor options, each offering varied capabilities and costs, tailored to the specific demands of the system. The initial configuration used the MG996R servo motor, a cost-effective choice at 18 CNY each. The MG996R operates within a voltage range of 4.8V to 6V, delivering a torque of 9.4 kg-cm at 4.8V and 11 kg-cm at 6V, with a speed of 0.17 seconds per 60 degrees rotation at 4.8V and 0.14 seconds at 6V. Despite its affordability, this motor provided the basic functionality required for early testing and development phases. For more demanding applications, we upgraded to the N6020KG servo, priced at 148 CNY each. This servo offers enhanced performance characteristics, suitable for applications requiring more precision and higher torque. It operates efficiently across a voltage range of 5V to 8.4V, achieving a torque of up to 25 kg at 8.4V, with a speed improvement notice-

able as the voltage increases from 5V to 8.4V. The pinnacle of our servo selection is the T80 brushless motor, which costs 205 CNY each and is designed for top-tier applications. This motor delivers exceptional torque and speed, crucial for the gimbal's high-end performance. Operating at voltages between 6V and 8.4V, the T80 provides a torque range from 63 kg at 6V to 78 kg at 8.4V, with a consistent decrease in speed from 0.16 to 0.14 seconds per 60 degrees as voltage increases, ensuring both power and precision in the stabilization process. These upgrades reflect our commitment to optimizing the gimbal's performance across various operational contexts.

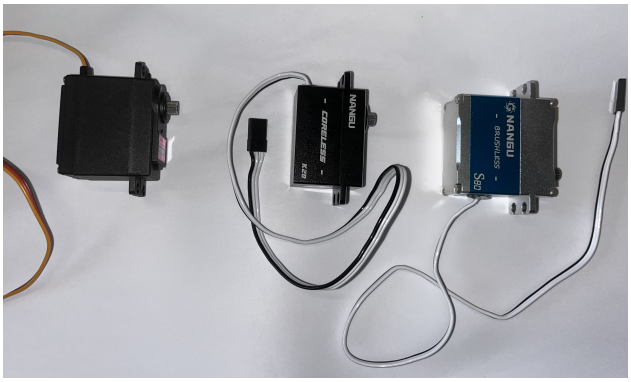


Figure 19: Two upgrades on servos

4.2 3D Reconstruction Subsystem

Table 4: Requirements and Verifications table for 3D Reconstruction Subsystem

	Requirement Description	**Verification Procedure**
1	The subsystem can convert $\mathbf{1920 * 1080}$ colour images and $\mathbf{512 * 424}$ depth images into point clouds.	For the subsystem, it is necessary to show the acquired colour and depth images, save and display the acquired point cloud.
2	The subsystem converts the point cloud into a smooth as possible 3D model.	The subsystem needs to visualize the reconstruction results. Manual checking of the model for continuity and smoothness.
3	The subsystem is capable of converting point clouds with **10,000** points or more.	Input large-scale point cloud data to observe the processing power and efficiency of the subsystem.
4	The reconstructed model can be saved as **OBJ** and other formats for storage in a database and visualisation in the interactive interface subsystem.	The saved model can be imported into professional software for visualisation in **OBJ** format.

The 3D reconstruction subsystem should be able to convert point clouds with more than 10,000 points into OBJ format. Figure 5 is the vertex counts of each reconstructed OBJ file, which represents the number of points in the original point cloud before reconstruction.

Table 5: The number of points in meshed point clouds

Stone Ornament	Chair	Cup	Pen Container	Vase
69132	178630	15171	13498	96944

Below is the dimensional error analysis for the four models including length width and height, the average error for all three dimensions is 3.84% which is within the high-level requirements target range. The result shows in Figure 6.

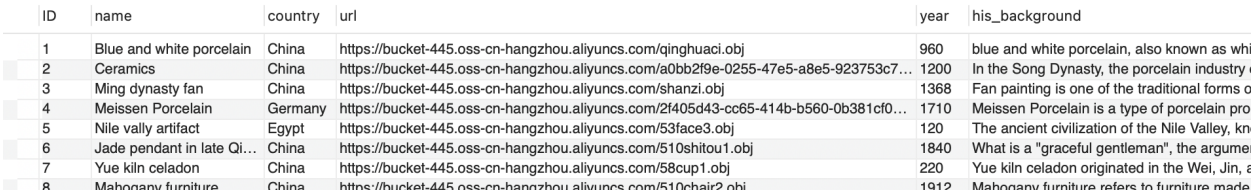
Table 6: Error analysis of the Models

Model Name	Height			Width			Length		
	Measured(cm)	Real(cm)	Errors (%)	Measured(cm)	Real(cm)	Errors (%)	Measured(cm)	Real(cm)	Errors (%)
Stone Ornament	23.86	24.90	4.17	8.10	7.60	6.58	16.16	15.70	2.55
Cup	10.16	10.10	0.59	8.37	8.50	1.53	12.93	12.20	5.98
Pen Container	9.50	9.70	2.06	10.80	10.50	2.86	10.30	9.70	6.19
Vase	26.21	24.80	5.60	14.60	13.90	5.04	14.19	13.90	2.09
Chair	81.43	80.00	1.79	59.31	56.50	4.97	54.29	56.50	3.91
Fan	32.82	32.50	0.92	10.45	9.60	8.85	19.14	18.50	3.46
Mean Errors (%)	2.52			4.97			4.03		

4.3 Database Subsystem

Table 7: Requirements and Verifications table for Database Subsystem

	Requirement Description	Verification Procedure
1	The subsystem should be able to store OBJ files in Object Storage Service (OSS).	Manually check if the OSS stores the OBJ files successfully.
2	The subsystem should be able to store the metadata and the URL links that returned by OSS.	Manually check if RDS stores the metadata and the URL links that returned by OSS.
3	The subsystem should be able to retrieve artifacts based on user search queries.	Manually check if the database correctly retrieve artifacts based on user search queries.
4	The subsystem should be able to establish a interaction between the database subsystem and the Interactive Interface Subsystem.	Manually check if the subsystem can provide data to the Interactive Interface Subsystem for artifact viewing.



ID	name	country	url	year	his_background
1	Blue and white porcelain	China	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/qinghuaci.obj	960	blue and white porcelain, also known as whi
2	Ceramics	China	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/a0bb2f9e-0255-47e5-a8e5-923753c7…	1200	In the Song Dynasty, the porcelain industry
3	Ming dynasty fan	China	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/shanzi.obj	1368	Fan painting is one of the traditional forms o
4	Meissen Porcelain	Germany	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/2f405d43-cc65-414b-b560-0b381cf0…	1710	Meissen Porcelain is a type of porcelain pro
5	Nile vally artifact	Egypt	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/53face3.obj	120	The ancient civilization of the Nile Valley, kn
6	Jade pendant in late Qi…	China	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/510shitou1.obj	1840	What is a "graceful gentleman", the argume
7	Yue kiln celadon	China	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/58cup1.obj	220	Yue kiln celadon originated in the Wei, Jin, a
8	Mahogany furniture	China	https://bucket-445.oss-cn-hangzhou.aliyuncs.com/510chair2.obj	1912	Mahogany furniture refers to furniture made

Figure 20: Verification for requirement 1

Figure 21: Verification for requirement 2

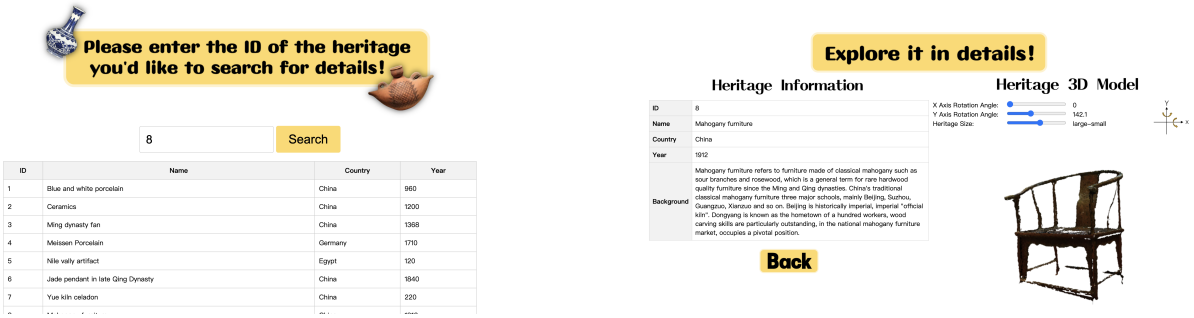


Figure 22: Verification for requirement 3

```
2024-05-18T20:22:20: SELECT * FROM artifacts WHERE ID = '8'
2024-05-18T20:22:20:  [0mPOST /search/getobj  [32m200  [0m10.516 ms - 23935 [0m
2024-05-18T20:22:20:  [0mGET /search/3Dshow.css  [33m404  [0m2.262 ms - 1333 [0m
```

Figure 23: Verification for requirement 4

4.4 Interactive Interface Subsystem

Table 8: Requirements and Verifications table for Interactive Interface Subsystem

	Requirement Description	Verification Procedure
1	The subsystem shall provide a search bar for users to search artifacts by keywords.	Verify that the search bar is visibly present on the website's interface and that users can input keywords to initiate a search.
2	The subsystem shall display 3D models and relevant information within 3 seconds of initiating a search.	Conduct performance testing to measure the time it takes for the website to load search results and verify that it meets the specified requirement.
3	Users shall be able to manipulate and explore 3D models using interactive controls such as zooming and rotating.	Perform user testing to ensure that interactive controls for zooming, rotating, and panning 3D models are functional and intuitive to use.
4	The subsystem shall allow users to navigate back from the result page to the search page.	Manually test the website's navigation functionality to confirm that users can return to the search page from the result page using provided controls or browser features.

Pictures below are verifications for requirement 1,3 and 4. For requirement 2, it can be verified by entering our website and counting time.

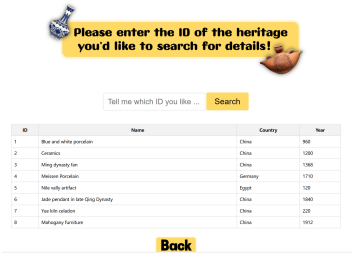


Figure 24: Verification for requirement 1

Figure 25: Verification for requirement 3

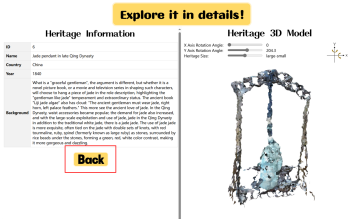


Figure 26: Verification for requirement 4

5 Conclusion

5.1 Accomplishment

Our project has designed and realized a complete set of facilities, including scanning instruments, reconstruction software, a complete process with a database and a front-end ui web page. It provides a cheaper and more personalized option for individual enthusiasts and museums. This enables the recording and sharing of information about artifacts around the world.

Scanning head can realize real-time control in 200 milliseconds, precise control of the platform's rotary movement, the rotation angle changes in the range of ± 1 degree, the speed can reach 15 degrees per second, in order to fully cover the cultural relics. Equipped with RGBD camera, the accuracy is in mm level and the depth range is 0.5-4m, which can meet the general interactive and recording requirements. The provided point cloud reconstruction software can restore the original color point cloud from the depth map and color images provided by the camera, and provides users with an easy-to-use and personalized processing flow.

In addition to ensuring the record and protection of cultural relics, we have also built a supporting front and back end for display and dissemination. This includes a database that can support the storage of more than 2500 models, and the ability to view artifact profiles and realize user interactions such as rotation and zoom. The database and web-page also support searching for specific artifacts and uploading one's own models and background information. Through these features, we are able to provide a comprehensive display and dissemination platform that allows users to conveniently browse, interact and share information about cultural relics, and promotes the wide dissemination of cultural relics and the transmission of culture.

5.2 Uncertanties

We are aware of the issues with the current gimbal system, primarily concerning its sturdiness and durability. The 3D-printed components are prone to breakage and are unable to meet the requirements for long-term stable operation. Additionally, the motors lack sufficient power and the support for the camera is inadequate, resulting in an unstable center of gravity for the gimbal.

The reconstruction is faster and more effective when doing large, more distinctive structural features, simple colors, and less reflective objects. Such as antique furniture, large sculptures, etc. For objects with fine textures, such as colored glaze vases, details need to be obtained through a lot of repeated sampling. For objects with detailed texture engraving, the detail requirements cannot be met. For objects that may cause strong specular reflection, such as ceramic mugs, holes are created. For objects that produce weak specular reflection, color distortion is produced. Also the color will react to changes in lighting and will not accurately get the true color of the object.

5.3 Future Work

To enhance the gimbal system, it is necessary to use stronger and more durable materials, significantly improving the structural integrity of the gimbal. This will ensure its ability to withstand demanding usage conditions and maintain stability over extended periods. Additionally, improvements in power supply management and motor control are required to optimize the overall functionality of the system, ensuring efficient power utilization and precise control.

For the 3D reconstruction component, real-time capability and reducing manual parameter matching should be considered in the future. Introducing a motorized rotating turntable can be beneficial, ensuring that the turntable's rotation speed matches the data acquisition speed of the scanning algorithm, enabling automated scanning and reconstruction. Development of adaptive algorithms can be pursued to utilize different parameters for point clouds with varying sparsity. Additionally, incorporating image feature recognition can assist in selecting parts with significant matching features as alignment targets, reducing the need for manual selection and alignment.

For databases, establishing a connection between the local and cloud environments is also crucial. If it is possible to achieve direct communication between the reconstruction software and the backend database, it will make the entire process more coherent and complete. Due to the large memory footprint of stored models, optimizing network configurations to improve access time is necessary.

5.4 Ethic Consideration

5.4.1 Ethics

In accordance with term 3 of the IEEE Code of Ethics, we pledge to "avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties

when they do exist" [15]. This commitment underscores our commitment to cultural sensitivity and awareness, guaranteeing that our scent cues are designed to be culturally respectful and appropriate in diverse contexts.

Furthermore, we consider the risk of unintended negative consequences arising from the misuse of emerging technologies. Hence, we follow term 6 of the IEEE Code of Ethics, which requires us to "enhance our technical competence and accept technological assignments only when qualified by training or experience, or after full disclosure of pertinent limitations" [15]. This ensures that our technology is exclusively provided to certified organizations and companies, minimizing the potential for misuse.

Meanwhile, in line with ACM and IEEE ethics, we will maintain transparency in our operations and communications. As stipulated by the ACM Ethics guidelines, term 3, we will " Be honest and trustworthy," [16] providing full disclosure of system capabilities and limitations. Similarly, the IEEE Ethics guidelines, term 5, demands that we " seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors" [15] .

Finally, by following both the IEEE Ethics guidelines [15] and ACM Ethics guidelines [16] , we aim to ensure that our technology not only advances in its technical capabilities but also contributes positively to society and operates within the highest standards of ethical conduct.

5.4.2 Safety

- Electrical Safety: Given our system's reliance on electronic components, we implement measures to prevent electrical shocks and hazards, ensuring all components are properly insulated and comply with relevant safety standards.
- Mechanical Safety: Our system includes moving parts; thus, we ensure these components are securely enclosed to prevent accidental injuries during operation.
- Environmental Safety: In project design and implementation, we adhere to the guidelines of ACM and IEEE regarding environmental sustainability [15][16]. We carefully use materials such as lithium batteries to prevent environmental hazards. We also select Polylactic Acid materials for 3D printed components, which are biodegradable and recyclable, to promote ecological friendliness and sustainable development.
- Data Security: To protect sensitive information collected during scanning, we employ robust security measures to prevent data breaches and unauthorized access.
- User Training: Comprehensive training for all users is essential to safely operate the 3D scanner, emphasizing awareness of potential hazards and adherence to established safety protocols.

References

[1] W. Li, "Application of virtual reality technology in the inheritance of cultural heritage," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1087, 2018, p. 062 057.

[2] J. Sun and H. Kim, "Digital display design of historical relics—using artistic projection of historical relics as an example," *TECHART: Journal of Arts and Imaging Science*, vol. 9, no. 1, pp. 35–48, 2022.

[3] *Open3d pull request #1884 commits*, Accessed on 2024-05-21. [Online]. Available: https://github.com/isl-org/Open3D/pull/1884/commits.

[4] Point Clouds. "Statistical Outlier Removal." (), [Online]. Available: https://pcl.readthedocs.io/projects/tutorials/en/latest/statistical_outlier.html (visited on 03/27/2024).

[5] Unknown, *02-introduction to point cloud registration - heima robotics — pcl-3d point cloud*, Online, [Accessed on: Insert Date], Unknown. [Online]. Available: https://robot.czxy.com/docs/pcl/chapter03/registration_intro/.

[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*, vol. 5, no. 4, pp. 349–359, 1999.

[7] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[8] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige, "Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 602–11 610.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[10] Alibaba Cloud. "What is OSS?" (2024), [Online]. Available: https://www.alibabacloud.com/help/en/oss/product-overview/what-is-oss (visited on 04/15/2024).

[11] Alibaba Cloud. "What is ECS?" (2024), [Online]. Available: https://www.alibabacloud.com/help/en/ecs/product-overview/what-is-ecs?spm=a2c63.p38356.0.0.45912ed2adH7yr (visited on 04/15/2024).

[12] Alibaba Cloud. "Configure an ip address whitelist for an apsaradb rds for postgresql instance." (2024), [Online]. Available: https://help.aliyun.com/zh/rds/apsaradb-rds-for-postgresql/configure-an-ip-address-whitelist-for-an-apsaradb-rds-for-postgresql-instance?spm=a2c4g.11186623.0.i12 (visited on 05/26/2024).

[13] PM2. "Managing applications states." (2024), [Online]. Available: https://pm2.keymetrics.io/docs/usage/process-management/ (visited on 05/26/2024).

[14] Wang Boqing, Wang Mengping, *2023 China Undergraduate Employment Report*. Beijing: Social Sciences Academic Press, 2023.

[15] IEEE. "IEEE Code of Ethics." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/27/2024).

[16] ACM. "ACM Code of Ethics and Professional Conduct." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 03/27/2024).

Appendix A Requirements

Configuration satisfied with Kinect v2 camera:

- Processor (CPU): AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx 2.00GHz
- Memory (RAM): 8.00GB
- Processor Architecture: 64-bit (x64)
- USB: USB 3.0
- Operating System: Windows 10 Professional Edition

Software environment required for C++ code:

- Programming Language: C++
- Integrated Development Environment (IDE): Visual Studio 2019
- Libraries: PCL version 1.12.1; OpenCV version 4.5.5; Kinect SDK 2

Software environment required for Python code:

- Programming Language: Python
- Integrated Development Environment (IDE): Pycharm
- Libraries: open3d 0.11.2; opencv 3.4.2; pyqt5 5.15.4;pykinect2; pcl-py 0.2.11; numpy 1.19.2; pygame 2.5.2; matplotlib 3.3.4

Appendix B Algorithms

- Obtain point cloud from Kinect.

Algorithm 1 Obtaining point cloud from Kinect

Input: Kinect device
Output: Point cloud data
Initialize Kinect device
Create point cloud container, $PointCloud$
while *Capturing point cloud* **do**
  Get depth image, DepthImage, and color image, ColorImage
  Get depth value, $depth$, and color value, $color$, from DepthImage and ColorImage
  for *each pixel* (x, y) **do**
    Convert pixel coordinates (x, y) to 3D coordinates (X, Y, Z)
    if $depth$ *is valid* **then**
      Create a point, $Point$, and assign (X, Y, Z) and $color$ to Point
      Add $Point$ to $PointCloud$
    end
  end
end
return *PointCloud*

- Point Cloud Registration.

Algorithm 2 Iterative Closest Point (ICP) algorithm

Input: Source point cloud A, Target point cloud B, an initial transformation matrix
Output: Transformation matrix
$T \longleftarrow T_0$
while $not\ converged$ **do**
  for $i \longleftarrow 1$ *to* N **do**
    $m_i \longleftarrow findClosestPointInA(T \cdot b_i)$
    $w_i \leftarrow 0$
    if $\|m_i - T \cdot b_i\| \leq d_{\max}$ **then**
      $w_i \leftarrow 1$
    end
  end
  $T \leftarrow \underset{T}{\operatorname{argmin}} \left\{ \sum_i w_i \|T \cdot b_i - m_i\|^2 \right\}$
end
return T

- Configure the OSS.

Algorithm 3 Configure the OSS.

$yourEndpoint \longleftarrow$ the Endpoint corresponding to the region where the Bucket is located. For example, for East China 1 (Hangzhou), the Endpoint is filled in as https://oss-cn-hangzhou.aliyuncs.com.
$BucketName \longleftarrow$ the Bucket name, for example, *examplebucket*.
$ObjectName \longleftarrow$ the complete object path, the complete path cannot contain the Bucket name, for example, *exampledir/exampleobject.txt*.
Initialize network and other resources.
$credentialsProvider \longleftarrow$ access credentials from environment variables
Visit OssClient using Endpoint and credentialsProvider
$content \longleftarrow$ the file content with the complete local file path
$request \longleftarrow$ the request to get the object with BucketName, ObjectName, and content
$outcome \longleftarrow$ the result with request for uploading
if *not success* **then**
    output "PutObject fail, code:" + outcome.error().Code() + ", message:" + outcome.error().Message() + ", requestId:" + outcome.error().RequestId()
    return -1
end
Release network and other resources
return 0

- Upload OBJ files to node.js server with Multer storage engine and Alibaba Cloud ali-oss SDK.

Algorithm 4 Upload OBJ files to node.js server with Multer storage engine and Alibaba Cloud ali-oss SDK

Input: req, file, cb
Output: Uploaded file information or error
Function `_handleFile(req, file, cb)`**:**
  Declare `content`
  if *file.buffer exists* **then**
    `content` ← file.buffer
  else
    if *file.stream exists* **then**
      Initialize `buffers` as an empty array `file.stream.on('data', (chunk) => buffers.push(chunk)) file.stream.on('end', () => { content ← Buffer.concat(buffers) _uploadToOSS(file, content, cb) }) file.stream.on('error', (err) => cb(err))`
    else
      `cb(new Error('No file content provided.'))`
    end
  end
Function `_uploadToOSS (file, content, cb)`**:**
  `options ← { progress: (p) => console.log('Progress: p * 100%') }`
  `filename ← guid() file.originalname ← filename + '.obj'`
  `client.put(file.originalname, content, options).then(async (result) => { file.ossUrl ← result.url const res ← await client.putACL(filename + '.obj', 'public-read') cb(null, file) }).catch((err) => cb(err))`
Function *destination(req, file, cb)***:**
  `cb(null, '') // OSS does not need a directory structure`
Function *filename(req, file, cb)***:**
  `cb(null, file.originalname) // Keep original filename`

- Parse OBJ files.

Algorithm 5 Parsing OBJ file

Input: OBJ file text
Output: Material libraries and geometries
Initialize empty arrays for vertex positions, texture coordinates, normals, and colors Initialize empty arrays for WebGL vertex data and geometries Initialize arrays for material libraries and geometries Initialize default material, object, and groups
foreach *line in OBJ file* **do**
    Parse the line to extract keyword and arguments
    if *keyword is a known OBJ keyword* **then**
        Call corresponding handler function with arguments
    else
        Continue to the next line
    end
end
Remove empty arrays from geometries
return *Material libraries and geometries*

- Render a 3D model.

Algorithm 6 Main function for rendering a 3D model

Input: Canvas element
Output: Rendered 3D model on the canvas
Obtain the Canvas container and set up the WebGL context If WebGL context is not available, exit the function Declare vertex and fragment shaders Compile and link the shaders, and find attribute and uniform locations Get the URL containing OBJ file information Fetch the OBJ file content via the URL Parse the OBJ file content and extract geometry information Process each geometry and create buffers Calculate the position and size of objects in the scene to fit the camera Initialize camera position and orientation Set camera view parameters Define user interaction parameters Enter the rendering loop **while** *rendering* **do**
    Update user interaction parameters Adjust Canvas size and viewport Enable depth testing and face culling Compute projection and view matrices Iterate over geometries, set uniform variables Draw geometries Request next frame rendering
end

Appendix C Arduino code

```ino
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"  // Include the Wire library for I2C communication
#endif
#include <Servo.h>  // Include the Servo library to control servo motors

// MPU6050 sensor setup
MPU6050 mpu;
Servo servoZ;  // Servo for controlling the Z-axis
Servo servoY;  // Servo for controlling the Y-axis
Servo servoX;  // Servo for controlling the X-axis
float OffsetOnZ;  // Offset to calibrate the Z-axis
int calibrationCount = 0;  // Counter to track number of calibrations

#define OUTPUT_READABLE_ZYX  // Define to enable Z, Y, and X output

#define INTERRUPT_PIN 2  // Pin number for the MPU6050 interrupt

bool dmpReady = false;  // True if DMP initialization was successful
uint8_t mpuIntStatus;   // Stores the latest interrupt status
uint8_t devStatus;      // Stores the return status from device operations
uint16_t packetSize;    // Stores the expected DMP packet size
uint16_t fifoCount;     // Count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer for sensor data

// Orientation and motion variables
Quaternion q;                           // Quaternion for calculating orientations
VectorInt16 rawAccel;                   // Raw accelerometer data
VectorInt16 accelWorldFrame;            // Accelerometer data in world frame
VectorInt16 accelGravityFrame;          // Accelerometer data in gravity frame
VectorFloat gravityVec;                 // Gravity vector
float ZYX[3];                           // Z, Y, X in degrees

// Function called when DMP data is ready
volatile bool mpuInterrupt = false;
void dmpDataReady() {
  mpuInterrupt = true;
}

void setup() {
  initializeCommunication();
  initializeMPU();
  initializeServos();
}

void initializeCommunication() {
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  Wire.setClock(400000); // Set I2C clock speed to 400kHz
#endif
}

void initializeMPU() {
  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT);
  devStatus = mpu.dmpInitialize();
  setSensorOffsets();

  if (devStatus == 0) {
    mpu.setDMPEnabled(true);
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();
    packetSize = mpu.dmpGetFIFOPacketSize();
    dmpReady = true;
  }
}

void setSensorOffsets() {
  mpu.setXGyroOffset(17);
  mpu.setYGyroOffset(-69);
  mpu.setZGyroOffset(27);
  mpu.setZAccelOffset(1551);
}

void initializeServos() {
  servoZ.attach(10);
  servoY.attach(9);
  servoX.attach(8);
}

void loop() {
  if (!dmpReady) return;

  handleDMPData();
}

void handleDMPData() {
  if (waitForDMPData()) {
    readDMPData();
    if (calibrationCount <= 300) {
      performCalibration();
    } else {
      adjustAndOutput();
    }
  }
}

bool waitForDMPData() {
  while (!mpuInterrupt && fifoCount < packetSize) {
    fifoCount = mpu.getFIFOCount();
  }
  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();

  fifoCount = mpu.getFIFOCount();
  return (mpuIntStatus & _BV(MPU6050_INTERRUPT_DMP_INT_BIT)) && fifoCount >= packetSize;
}

void readDMPData() {
  if (fifoCount >= 1024 || (mpuIntStatus & _BV(MPU6050_INTERRUPT_FIFO_OFLOW_BIT))) {
    mpu.resetFIFO();
    fifoCount = 0;
    return;
  }

  mpu.getFIFOBytes(fifoBuffer, packetSize);
  fifoCount -= packetSize;
  interpretMotionData();
}

void performCalibration() {
  OffsetOnZ = ZYX[0];
  calibrationCount++;
}

void adjustAndOutput() {
  ZYX[0] -= OffsetOnZ;  // Adjust Z based on the calibrated offset
  int zVal = map(ZYX[0], -90, 90, 0, 180);
  int yVal = map(ZYX[1], -90, 90, 0, 180);
  int xVal = map(ZYX[2], -90, 90, 180, 0);

  servoZ.write(zVal);
  servoY.write(yVal);
  servoX.write(xVal);
}

void interpretMotionData() {
#ifdef OUTPUT_READABLE_ZYX
  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravityVec, &q);
  mpu.dmpGetYawPitchRoll(ZYX, &q, &gravityVec); // Correct function call

  // Convert Z, Y, and X from radians to degrees
  ZYX[0] = ZYX[0] * 180 / M_PI;
  ZYX[1] = ZYX[1] * 180 / M_PI;
  ZYX[2] = ZYX[2] * 180 / M_PI;
#endif
}
```

Figure 27: Arduino code for Assistive Scanning Subsystem