

Virtual Reality Interfaces for 3D model reconstruction and sharing

Student: Andrei-Alexandru Bunea

Supervisor: Prof. Anthony Steed

MSc Computer Graphics, Vision & Imaging

September 2023

This report is submitted as part requirement for the MSc Degree in 'Computer Graphics, Vision & Imaging', at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Acknowledgements

I would like to express my profound gratitude to Professor Anthony Steed, my supervisor, for his invaluable support throughout the whole collaboration. His guidance and mentorship have been instrumental in successfully finishing this project.

I would also like to thank Nels Numan and Ziwen Lu, teaching assistants of the Virtual Environments course, who guided me in the early stages to find and design the idea that was implemented in this final project.

Finally, I would also like to thank my family for their unconditional and unwavering support. They have been the driving force behind every step of my journey, shaping me into the person I am today.

Abstract

In recent years, 3D reconstruction has revolutionised various industries, proving to be a transformative practice in scientific research, e-commerce, and beyond. Studying a 3D model has proven more useful and intuitive than 2D images in analysing objects, helping professionals improve their expertise in medicine, archaeology, art, and many other fields.

This immersive experience not only enhances the assessment of virtual objects but also bridges the gap between the virtual and physical worlds. The integration of this technology has revolutionized the way professionals approach their work, enabling them to gain unparalleled insights into complex structures and artefacts. However, the reconstruction is done mostly by professionals who are passionate about photogrammetry and possess skills and software for 3D reconstruction. They can later choose to publish their work on specialised platforms for 3D portfolios such as “Unity Asset Store” or “Sketchfab”.

This project comes as a solution for users who would like to create their own 3D models without having any prior knowledge of photogrammetry, visualize the results using a virtual reality platform, store their work and easily share it with others.

The project encompasses a web-based user interface acting as a social media platform where users publish 3D models based on uploaded images or videos, a structure-from-motion 3D reconstruction pipeline using “Meshroom”, and a Unity-based application for VR visualization. The VR application allows users to immerse themselves alongside the reconstructed 3D objects, enabling them to inspect the models from various angles and distances.

The project's methodology combines image processing, computer vision, and VR development, resulting in a robust and user-friendly system. Throughout the development process, a thorough review of related literature was conducted to ensure the project's alignment with current advancements in VR applications and photogrammetry.

The evaluation of the system involved user feedback and usability tests, providing valuable insights into the overall effectiveness and user experience. The results demonstrate the system's ability to generate accurate 3D models and deliver an engaging VR experience to users.

Contents

Acknowledgements.....	2
Abstract	3
1. Introduction	6
1.1 Motivation.....	6
1.2 Project structure	8
1.3 Structure of the Report	10
2. Literature Review	11
2.1 Photogrammetry	11
2.2 Structure-from-Motion	14
2.3 Structure-from-Motion Applications.....	18
2.4 Virtual Reality.....	23
3. Implementation	24
3.1.1 Web Application Implementation Details	25
3.1.2 Presentation of the Web Application	26
3.2 The Structure-from-Motion pipeline	41
3.2.1 Camera Initialization	43
3.2.2 Feature Extraction.....	44
3.2.3 Image Matching	45
3.2.4 Feature Matching.....	46
3.2.5 Structure-from-Motion	48
3.2.6 Dense Scene Preparation	49
3.2.7 Depth Map	49
3.2.8 Depth Map Filtering	50
3.2.9 Meshing.....	51
3.2.10 Mesh Filtering	52
3.2.11 Mesh Decimating	53
3.2.12 Mesh Resampling.....	54
3.2.13 Texturing	55
3.3 Unity VR Application	57
3.3.1 VR Application Implementation Details	57
3.3.2. Presentation of the VR Application	58
4. Evaluation.....	66

4.1 Interpretation of the results.....	66
4.2 Methodology.....	67
4.3 The results of the experiment.....	69
4.4 Use cases.....	77
5. Conclusion.....	78
References.....	79

1. Introduction

1.1 Motivation

In today's rapid technological advancement, scientists and researchers have vastly improved their analysis by replacing the ordinary image interpretation with a more comprehensive concept: 3D models.

Using a virtual entity of a physical object and a VR visualisation tool simulates the sensation of looking at real-world objects. Immersing the user in such a way enables in-depth examination and investigation, as it confers a holistic understanding of the object, rather than the analysis of a batch of scans or images taken from different points of view. Moreover, the hands-on experience provides the researchers with a boost of confidence in their expertise, as they are more assured when working with real-world models.

Three-dimensional (3D) modelling has become a transformational force in today's quickly changing technology landscape, having a significant influence on a variety of industries from education to cultural preservation.

The goal of this project is to reduce the gap between the common users who do not possess advanced skills in photogrammetry, and the 3D model creation realm, providing a reliable and easy-to-use platform for 3D reconstruction and model sharing. Moreover, this project is investigating the complex interactions between user-generated 3D models, immersive visualisation and their many applications.

While there are a significant number of tools for either structure from motion reconstruction or VR visualisation of 3D models, there is no such platform that combines both. Moreover, each one of these activities is performed by professionals who own 3D reconstruction software and have advanced expertise in how to perform a reconstruction algorithm or how to create a VR scene where entities can be observed.

In addition, the industry currently lacks a platform for 3D model sharing that is not oriented towards professionals. Not only there are a couple of websites that provide the opportunity for 3D model browsing, but the most frequently used applications such as "Unity Asset Store" [1] or

“Sketchfab” [2] require the designers to perform their own 3D model creation process before uploading the results to their portfolios.

Therefore, the innovation that this project brings to the field is a democratic and accessible process for 3D model production, sharing and exploration by utilising the capabilities of 3D reconstruction complex algorithms, advanced software, and virtual reality. By following this approach, users are relieved from the burden of mastering state-of-the-art algorithms, allowing them to fully enjoy the benefits of creating and exploring virtual representations of real-world models.

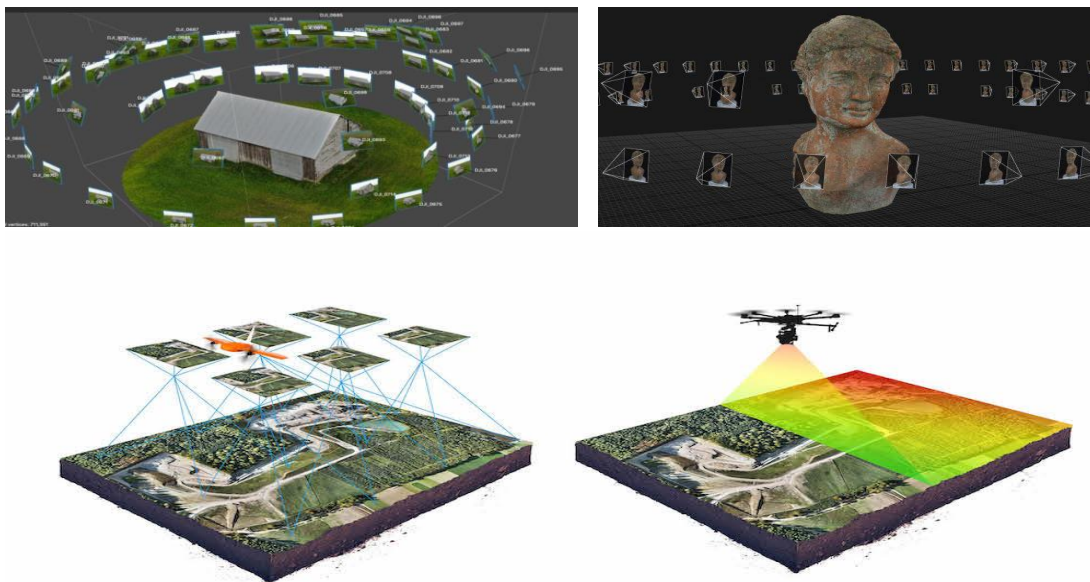


Figure 1 Photogrammetry projects

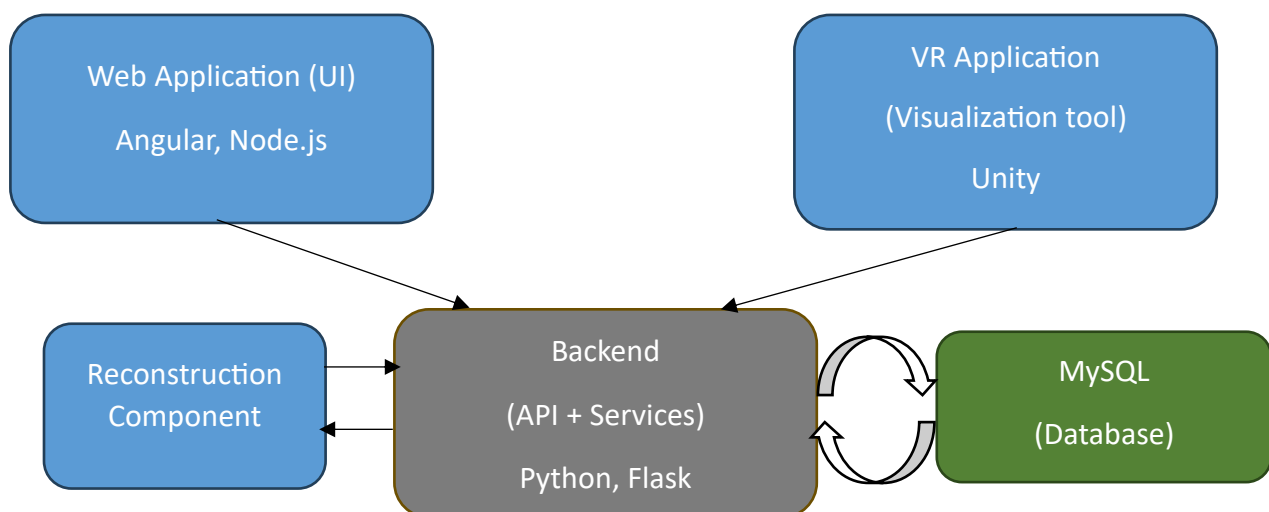
The concept of capturing a 3D scene through visual recording has a rich history, dating back to the World Wars when scientists employed aeroplanes and balloons to survey neighbouring terrains [3]. Since then, the evolution of technology has ushered in a new era in this field, where modern cameras, drones, and laser technology have enabled the recording of physical environments in virtual forms. These technological advancements have propelled photogrammetry into various domains, including medicine, archaeology, geology, forestry, and

numerous others, capturing images from close range to extreme heights using satellites. Since its discovery, photogrammetry has developed into a groundbreaking practice that serves as an invaluable tool for advancing research and exploration.

Vital to this project, Virtual Reality represents a groundbreaking idea that involves crafting an environment where individuals can detach from their actual surroundings and immerse themselves in diverse sensory experiences. This distinctive idea enables users to engage their senses through realistic simulations of real-world experiences, all from the convenience of their home or office, and having access to multiple virtual worlds in a small device such as a pair of VR headset. This project aims to use virtual reality to provide a highly realistic immersive display that stimulates the senses of the users while allowing them to interact with reconstructed 3D models.

1.2 Project structure

The 3D model reconstruction and visualisation project is methodically organised into various components that contribute harmoniously to a coherent and efficient workflow. The process guarantees that 2D photos are successfully transformed into 3D models and transposed in an immersive VR environment. The architecture of the project includes four main components: an Angular web application, a 3D reconstruction component, a Flask backend, and a Unity VR application. All the components work in tandem to offer a smooth and engaging experience.



The interaction with the project begins by accessing the web-based user interface. The frontend component is designed to facilitate friendly designed and intuitive features. To use the web application, the users need to sign up for an account or log in with an existing one. The login credentials are verified by interrogating the project's database. After the successful validation, users are allowed to perform all the activities that the platform offers. First, there is a Dashboard component where the users upload their images or videos in order to create 3D models.

Another feature is the personal portfolio, where users can preview their work, download the models, edit their names, or delete them. In addition, users can search for other users' profiles in the explore section, follow their work, and see their 3D models. Every 3D model can be seen both in VR and in the web application, where a scene is rendered including the targeted reconstructed object.

The second component is responsible for creating the 3D models using a structure from motion automatization on the images or video provided by the user. Using the free, and open-source Meshroom software from AliceVision, the project introduces a script that runs 13 reconstruction stages. At each stage, the pipeline's script configures the parameters that are passed to Meshroom, having multiple scripts, one for each reconstruction step. After feature extraction and matching, camera estimation, dense scene, depth map filtering, mesh decimation and texturing, the 3D models are created and ready to be rendered in the visualisation tools.

The third component is the one that organises all the processes and data flows – the Python, flask backend. Inside the backend, the endpoints for the applications API are defined, providing routes that are used by the web user interface and the VR visualisation tool. Moreover, it processes the upload of images and videos, while preparing the input dataset for the reconstruction pipeline. In order to store details about the users, the application stores their email, name and IDs of the models in a MySQL database, that is queried by the backend when needed.

Last, the fourth component is the Unity VR application which serves as an immersive visualisation tool. The VR application provides support for login, after which the user can preview and spawn their own model and search for the portfolios of the people they follow. In other words, the application has the same features as the web application except for the signup and image upload

for reconstruction. In addition, the VR application provides a unique and immersive tool that facilitates a fast and genuine exploration of 3D models. Inside the virtual scene, the users can interact with the object, rotate, scale, or translate it in order to achieve a better understanding of the model.

1.3 Structure of the Report

This report aims to provide a good understanding of the mechanics behind the implementation and its use cases. In order to achieve this, the report will further describe previous works involving structure from motion, the implementation of this project, its evaluation and the conclusions.

In section 2, we will analyse the literature review and examine the implementation of similar tools, as well as their usage and achievements.

In section 3, the implementation of the project will be discussed, examining step by step the processes behind the four main components: the web user interface, the reconstruction pipeline, the backend, and the Unity VR application.

Section 4 will be designated for the evaluation of the project, studying the performance of the reconstruction process. In addition, we will discuss the feedback from the users who were asked to participate in an experiment where they tested the web user interface and the VR visualisation tool.

Last, section 5 will assess the conclusions regarding this project, its strong features and weaknesses and will consider future work for its improvement.

2. Literature Review

The literature review chapter examines existing research and studies underpinning the field of 3D model creation and visualization. This chapter examines a variety of sources, from basic academic papers to groundbreaking research articles, all of which contribute to the project's foundational knowledge. Through synthesizing these sources, the literature review contextualizes the project within technology and academia and provides guidance for innovative methodologies and strategies.

2.1 Photogrammetry

Photogrammetry is a dynamic and complex process that combines multiple image processing algorithms to create 3D models from sets of overlapping images. Serving as a state-of-the-art process that converts reality into virtual scans, photogrammetry evolved through technological advancements of cameras and computers and is nowadays a novel practice in various fields of research.

In general, a photogrammetric project involves two major steps: acquisition and preparation of the image dataset, and the process of reconstruction. The first step ensures a proper camera setup and scene properties for the desired task while capturing a collection of photos with good overlapping areas, and a full coverage of the area of interest. The preparation of the image dataset may also involve applying special filters or processing techniques to images to potentiate the features of the physical object. Whereas the second step covers the whole process that converts the images into the final 3D model, encapsulating custom algorithms from feature extraction and matching to geometric reconstruction, depth map estimation and texturing. [4]

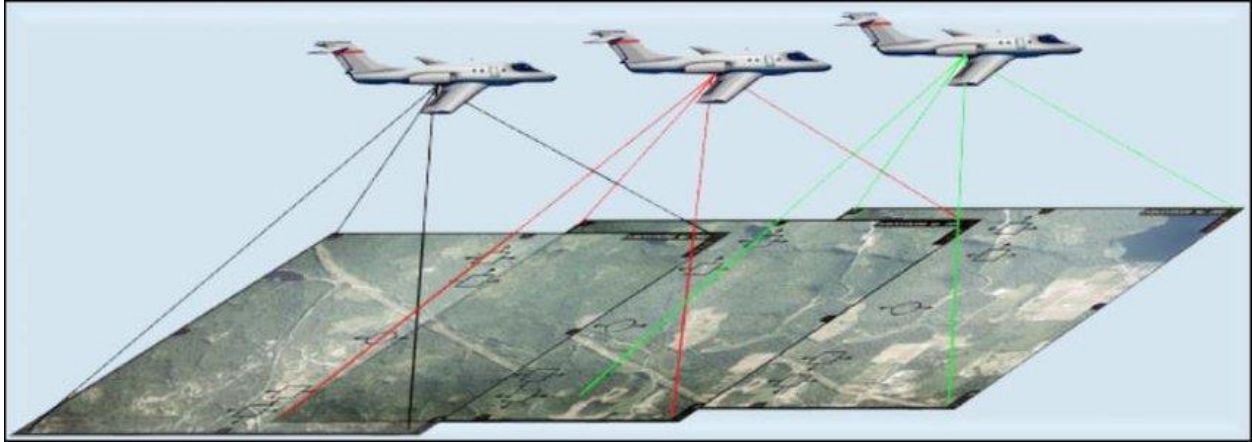


Figure 2 Aerial Photogrammetry illustration

Generally, photogrammetry is divided into aerial and terrestrial photogrammetry. Aerial photogrammetry acquires images taken from lower heights using captures from drones or balloons, medium heights using airplanes or even from astonishing heights using modern satellites. The process of acquiring images for aerial photogrammetry is presented in Figure 2. [5] Terrestrial photogrammetry is performed on images captured from the ground. The images can be acquired from far distances or close range. When photogrammetry is performed on images taken from less than 100 meters, is categorised as close-range photogrammetry. [6]

While aerial photogrammetry is a popular practice in cartography, archaeological exploration and environmental monitoring, close-range photogrammetry has been used in detail-oriented practices such as art restoration, industrial design, and medicine.

The close-range photogrammetry takes a more intimate approach, focusing on capturing fine details of objects from a short distance. This technique usually involves the user of high-resolution cameras to photograph subjects from multiple points of view, creating accurate and detailed 3D models. [7]

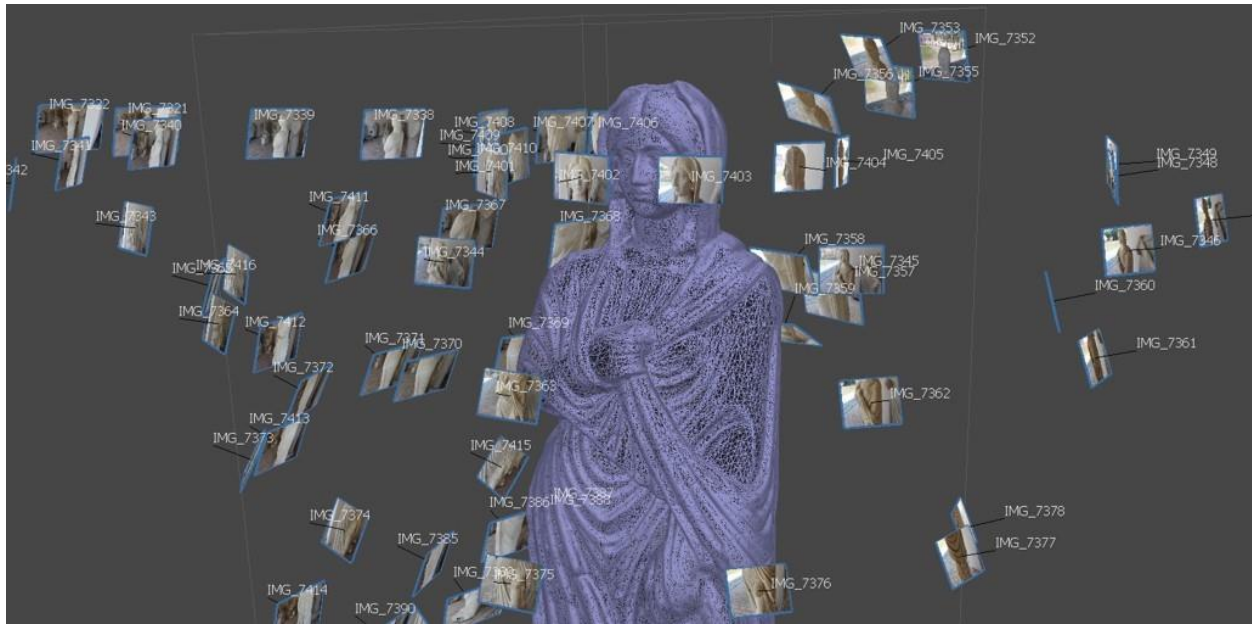


Figure 3 Close-range Photogrammetry application for digital curation

In Figure 3 is presented a close-range photogrammetry application performed by the Institute of America in Transylvania, Romania in 2023 during an archaeological photogrammetry workshop. [8] In the illustration from above we can observe a multitude of high-quality images taken from different perspectives and ranges, photos that come all together to reconstruct a solid virtual representation of the targeted statue. Using photogrammetry to capture every fine detail from images of an object, not only provides a better understanding of the object but also preserves an invaluable and meticulous digital copy of the entity.

2.2 Structure-from-Motion

Structure from Motion (SfM) is a process derived from photogrammetry and encapsulates all its goals. Both SfM and photogrammetry target the reconstruction of 3D scenes or objects from a collection of 2D images. However, SfM incorporates other additional tasks such as estimating the camera position and parameters, being more concerned with capturing relative positions and orientations of images.

In photogrammetry, camera calibration and image orientation are crucial for accurate measurements and are established from the start of the experiment. Ground control points may be used for georeferencing. In contrast, structure from motion is more flexible and automated, as it does not rely on camera calibration and geolocation. Instead, it uses feature tracking across the images to determine the camera parameters. [9]

The key stages of the structure-from-motion pipeline include:

1. Feature Detection

This first step covers the detection of distinctive features in each image such as corners, edges, or other key points. Classic algorithms for feature detection are Harris Corner Detector and ORB.

2. Feature Matching

The second step is responsible for establishing correlations between features extracted in different images. Classic algorithms for this step are RANSAC and SIFT.

3. Camera Pose Estimation

Using the feature correlation between images, the fundamental matrix for camera calibration is estimated. From this matrix, the extrinsic and intrinsic parameters are extracted, measuring the camera position and orientation.

4. Sparse Point Cloud Reconstruction

Based on the information provided by the essential matrices estimated in the previous step, information about the local 3D coordinates of the mesh vertices is extracted, thus constructing a sparse point cloud reconstruction of the model.

5. Bundle Adjustment

This fifth step is an optimization process that refines the estimated camera poses and vertex positions, minimizing the reprojection error. This step ensures that the reconstructed 3D points align as closely as possible with the characteristics of the physical object.

6. Mesh Generation

Additional 3D points are reconstructed by triangulating pixels from all images, completing the point cloud of the scene or model's surface.

7. Texture Mapping

This final step involves mapping the texture to every face of the mesh, using the original images as a reference to obtain similar-looking textures.

The SfM pipeline is a reliable and flexible approach to creating 3D models, recommended for projects that do not have the facilities of setting up the camera parameters. This technique has the same applications as close-range photogrammetry. Advances in computer vision enrich the technical capabilities of the SfM pipeline to produce more accurate structures, render corresponding textures and achieve a faster rendering with a smaller running time.

Thanks to technologies such as Structure from Motion (SfM), the field of 3D reconstruction has made significant progress in recent years. SfM techniques allow for the creation of precise and detailed 3D models from multiple 2D images, making it an essential tool in numerous fields.

The paper "Structure from motion photogrammetric technique" [10] presents the incredible potential of this technique, highlighting its diverse uses in the realms of geoscience, forestry, and cultural heritage conservation. One of the most significant cornerstones in the applicability of SfM is the introduction of advanced software tools that offer an established pipeline. This pipeline comes to help the users, as they can now focus more on the algorithms and parameters chosen, rather than the development of the whole methodology.

In searching for the perfect reconstruction tool that can help the purpose of the project, multiple critical factors were considered. As the software needs to be integrated into the architecture of the system without, being part of an automatised process, it was mandatory that the

reconstruction tool needs to be utilised using libraries or command line interface scripts instead of GUI applications. This single aspect alone significantly reduces the options available, as most of the structure from motion programs are designed to be used in graphic interfaces, as the professionals would check the results of the process at every step of the pipeline.

Another important characteristic of the chosen software was the price and accessibility. The collection of structure from motion programs is divided into two groups: free and open-source tools, and commercial tools. Experimenting with the performances of the project in reconstructing various scenes in different conditions of light and quality, the use of a commercial tool would turn out to be too costly for the purpose of the project. Tools such as 3DF Zephyr, Recap Photo, Agisoft Metashape and Reality Capture are examples of commercial applications that produce excellent results and achieve faster processing times than the open-source platforms. [11]

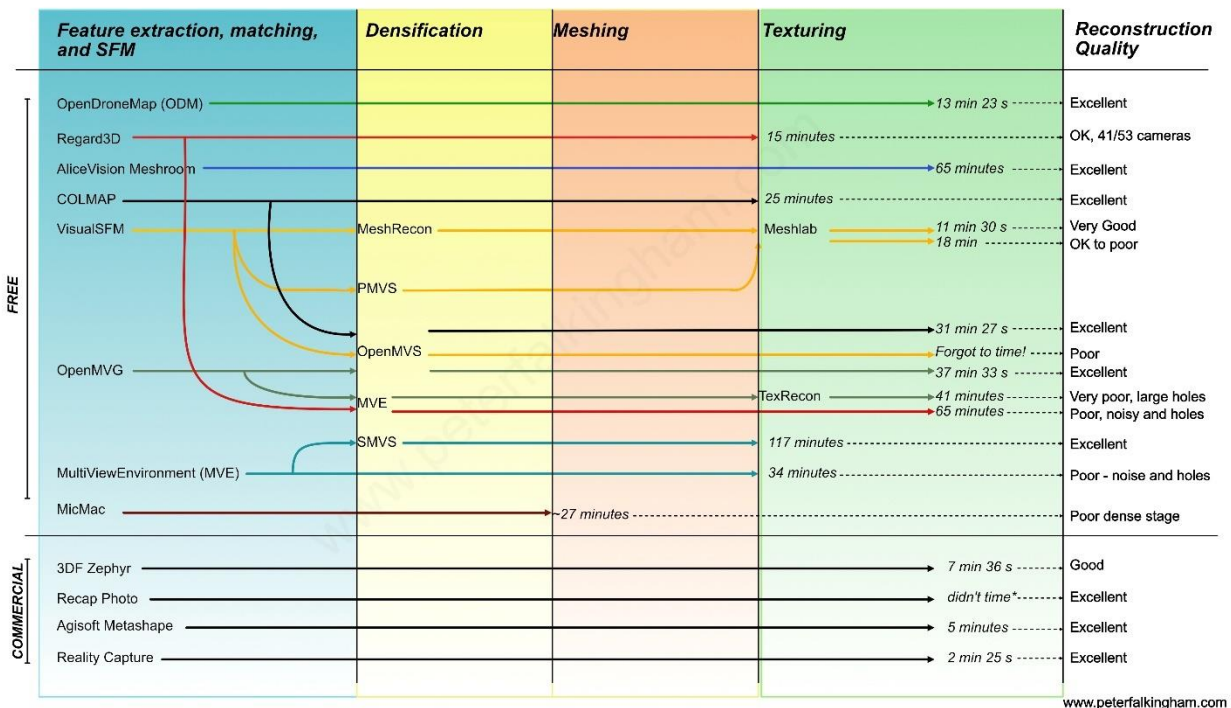


Figure 4 Comparison of Photogrammetry tools

Figure 4 illustrates a comparison done by Prof. Peter L. Falkingham in an article from his personal website [12]. Observing the table, it can be observed that only one reconstruction software for

close-range photogrammetry encapsulates all the stages of the reconstruction from feature extraction to texturing. This software is Meshroom [13]. Moreover, Meshroom is one of the very few tools that can be managed using command line commands, and it is also categorised as producing excellent results.

The paper "AliceVision Meshroom: An open-source 3D reconstruction pipeline" [14] describes Meshroom as software that enables researchers and practitioners to generate high-quality 3D models from photos. The paper presents the usage of the software at each step of the SfM pipeline, introducing the parameters and techniques that are available to the user. The researchers present their results by running the algorithm on multiple image datasets: a fountain dataset, a Buddha statue dataset, and a city wall dataset. For this experiment, the 2021.1.0 release of Meshroom was used.

The results show that the performance of the model is directly proportional to the performance of the hardware that runs the process, as better CPUs and GPUs produce faster results than slower processing units. Moreover, the complexity of the model dictates the estimated time for completion. As shown, the fountain model contained 484383 faces and was reconstructed in 2m and 19s, the Buddha statue contained 3109710 faces and was reconstructed in 28m and 25s, and the city wall contained 3744703 and was reconstructed in 1h and 50m, while all of them were run on the same hardware. [14]

Therefore, the key to mastering tools such as Meshroom to create accurate 3D reconstructions is to research the combination of parameters that are used by the pipeline, as explained in the "Analyzing parameters in 3D reconstruction photogrammetry in Meshroom, a case study" paper [15]. The study reveals several significant insights such as that adjusting the number of input images and their resolution directly influences the accuracy and level of detail in the reconstructed 3D models. Additionally, the quality of feature extraction plays a crucial role in the success of the reconstruction process, as it impacts the identification and matching of key features across images. Moreover, the scientists underline the trade-off between computational time and model fidelity, as more processing time is required for higher-quality feature extraction, and thus a more accurate reconstruction of the 3D model.

The study titled "Evaluation of the 3D Reconstruction Performance of Objects in Meshroom: A Case Study" is a valuable resource for understanding the capabilities and limitations of the Meshroom 3D reconstruction pipeline. This study aligns perfectly with our project goal of optimizing the 3D reconstruction process to create accurate and detailed models. The authors conducted a thorough analysis using different objects with varying complexities and characteristics, which reflects the real-world scenarios that our project aims to address.

By assessing Meshroom's various parameters, including camera settings and feature detection algorithms, the study evaluated the accuracy and quality of the reconstructed models. Regarding the camera settings, the study finds that the most impactful parameters are the focal length, aperture, and image resolution, as they directly impact how the input data is being processed. However, some feature-detecting algorithms may have better performance on certain types of objects. While some of the algorithms perform well on most of the input types, some algorithms may produce even better results for a specific type of observed object. Therefore, experimentation is recommended for more accurate results.

Another interesting finding is that Meshroom facilitates robust reconstruction performance for objects with distinct features and clear textures. Objects like architectural elements and sculptures were reconstructed with remarkable fidelity, capturing fine details and maintaining overall accuracy. However, the study also revealed that Meshroom faced challenges when dealing with texture-less or reflective surfaces. In such cases, the reconstruction quality notably decreased, indicating potential areas for improvement.

2.3 Structure-from-Motion Applications

Westoby et al.'s publication "Structure-from-Motion Photogrammetry: A Low-Cost, Effective Tool for Geoscience Applications" [9] is an important cornerstone in the field of photogrammetry. The research published in the Journal of Applied Geology, investigates the utility and applicability of the Structure-from-Motion reconstruction approach in geoscience, as SfM provides a low-cost and powerful method for creating detailed 3D models from sets of images.

In-depth analyses by Westoby and his colleagues of SfM's inner workings show how well it can extract important elements from a variety of photos to create accurate 3D representations of natural stones. SfM demonstrates how it can create precise reconstructions from a variety of uncalibrated picture sets by iteratively improving feature matching and camera placements. The paper also underlines the SfM's approachability and affordability, shaping a tool that fills knowledge gaps and promotes wider community involvement in geoscientific research.

This study attests to the revolutionary potential of SfM within the geoscience field. The practice of using an intuitive and fast mechanism of producing 3D models out of digital images not only is accessible to every specialist in geology, but it also creates a new dimension in sharing new findings, in a more realistic representation of the uncovered models.

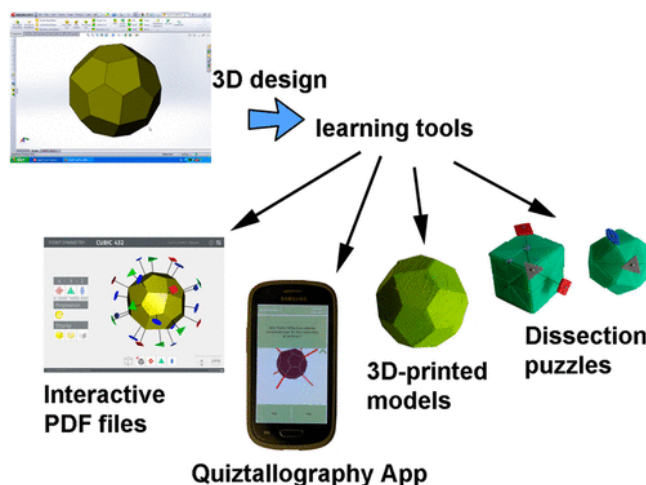


Figure 5 Geology application introduced by Lluís Casa and Eugenia Estop

Geology is only one of the many fields in which the 3D visualisation of virtual models is a plus. Not only that 3D reconstruction is a powerful tool for creating an inventory of findings, but it also is an innovative teaching instrument. Authors Lluís Casas and Eugenia Estop conducted a research project [16] in which students were introduced to virtual 3D models using a mobile application and 3D printed models. Through this approach, students gain a deeper understanding of crystal symmetry and point groups, bridging the gap between theoretical knowledge and practical application, as they were able to study the model as a whole. Users were able to rotate, scale and even section the crystals. Figure 5 illustrates the 3D visualisation feature from the article.

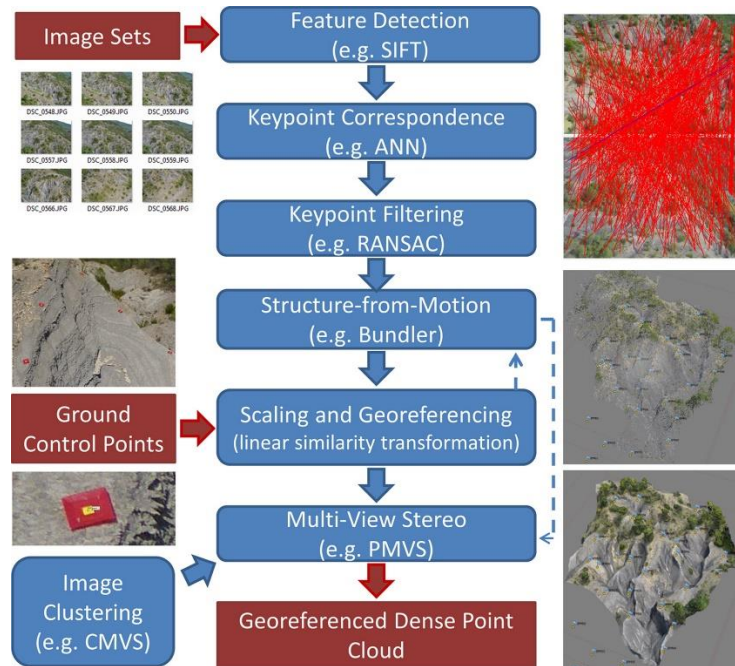


Figure 6 The SfM pipeline used in "Structure from Motion photogrammetry in physical geography"

In the journal article "Structure from Motion photogrammetry in physical geography" [17] the researchers thoroughly explored the practical application of Structure from Motion techniques in physical geography. Utilising a collection of overlapping aerial photographs, they used the photogrammetry technique mentioned before to generate detailed three-dimensional models of geographical features. The results showcased the accuracy and efficiency of SfM in producing complex textured digital representations of landforms, aiding in the analysis of topographic changes and erosion patterns. By demonstrating its capability for capturing accurate 3D models for complex physical landforms, the study highlights SfM as a valuable resource for physical geography researchers. Figure 6 illustrates the pipeline used by the researchers to implement 3D reconstruction, also providing examples along the scheme.

Similar to the previous work is the "Structure from Motion Photogrammetry in Forestry: a Review" journal article [18], which studies the particularities of forests using the SfM technique. The methodology uses overlapping images of the forest environment to create 3D models that are instrumental in tasks such as estimating tree heights and assessing the canopy coverage and vegetation health.

Preserving and indexing the heritage is a topic that fascinates scientists worldwide, as it is a key practice not only to showcase historical and natural artefacts but also to enhance a better understanding of our history and environment. The utilization of photogrammetry and 3D scanning has proven transformative in heritage recording, as highlighted in the research paper "Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning" [19]. This study explores advanced techniques for preserving cultural heritage using high-resolution imagery and precision scanning to generate detailed 3D models of sites and artefacts.



Figure 7 The “Relief” tomb in Cerveteri, Italy



Figure 8 3D Model of the forum in Pompeii

The paper presents various case studies where this approach was employed such as creating virtual models of The “Hanting and Fish” tomb in Tarquinia, Italy, the “Relief” tomb in Cerveteri (Figure 7), Italy, or even an aerial 3D scanning of the forum in Pompeii (Figure 8). Their results showcase efficiency in creating digital representations that not only aid preservation but also contribute to historical research, educational initiatives, and virtual experiences. As we seek to

explore the applications of 3D modelling and photogrammetry in diverse contexts, this study offers insights into the importance of photogrammetry in providing the opportunity to remotely study fascinating sites from all over the world.



Figure 9 3D models of acquired archaeological artefacts

Similar to the previous use case, the article "Digital Replicas of British Museum Artifacts" [20] published in 2023 demonstrates the power of photogrammetry and virtual environments in digitising and showcasing museum artifacts. In this paper, the authors engaged in creating accurate digital replicas of British Museum artefacts using advanced photogrammetry techniques. The process prioritised the capturing of fine details and textures in order to achieve genuine digital replicas of complex artefacts of different sizes, as presented in Figure 9.

By preserving artefacts as 3D models in a virtual environment, the project contributes to broader accessibility and knowledge dissemination, as the results are accessible to the wide public. The authors discuss how these digital replicas can be utilized for educational purposes, allowing users to interact with historical objects in ways previously unimaginable. This research project perfectly aligns with the project of this report and acts as an inspiration and motivation on how 3D model production and sharing could impact the world.

2.4 Virtual Reality

Immersion is the main goal of Virtual Reality. To achieve total immersion, users need to experience a complete detachment from reality, engaging all their senses solely within the virtual environment. To elicit immersion, a virtual environment should include four main components: sensors, simulators, rendering and display. Examples of sensors are trackers, microphones or bio-signal devices that turn physical effects into signals [21]. VR headsets such as the Oculus Quest used in the developing process of this project rely on the controller's sensors and the headset's cameras to provide critical information about the user's position and gestures. The output of the sensors is passed to the simulators that simulate the virtual environment and its interaction with the user based on the data provided. Further, the renderer creates the Virtual Environment visually and produces signals, which are then transformed into physical images by the display.

Unity is one of the most popular tools for creating immersive Virtual Reality applications. It is an optimal solution for developers as its flexibility, feature-rich environment, and wide range of compatibility foster the creation of interactive and captivating VR experiences. Even though Unity was conceived initially for game development, its applications cover a larger usability spectrum, making it a cornerstone development tool for VR applications.

One of Unity's strongest features is its integration with VR hardware, like the Oculus Quest headset used in the project. It provides special support for specialised VR technology such as the XR controllers' tracking mechanism and actionable buttons. It also offers a specialised collection of 3D models, enhancing the performance and creativity of the projects. [22]

Furthermore, the implementation of realistic interactions within the virtual worlds is facilitated by Unity's physics engine which enables users to authentically interact with objects from the virtual setup. Assessing this project's needs, Unity was chosen as the optimal platform for the development of the VR visualisation application. It provided fast protocols for the data acquired from the server at runtime, a detail-oriented interaction with the environment and an aesthetic touch on the design.

3. Implementation

The implementation of the project encapsulates the web and VR applications, the 3D reconstruction component, the MySQL database, and a Flask backend that represents the core of the application. All these components are hosted on a local workstation having an AMD Ryzen 7 5800H CPU with 16GB RAM and an NVIDIA GeForce RTX 3050 GPU. To successfully run the project, all the components need to run simultaneously. The repository of the project is public, and it can be found on the GitHub profile of the author. [23]

First, the MySQL database needs to be up and running. The Flask backend is running using an Anaconda environment that has all the required libraries installed. The web application is hosted using the start command of the npm package manager. And last, the reconstruction component is called by the backend using a bash script and requires the Meshroom packages to be installed.

To use the web application, the only requirement is to use a device that can access the website and upload images or videos from its local memory. The application is built to be accessible from any browser, and it is responsive to any screen dimensions. On the other hand, the VR application is specifically designed for VR headsets, Oculus Quest in particular. Even though the application is running using an APK file that can be opened on various devices, the implementation of the VR application relies on the XR commands and interface of the Oculus Quest, and therefore devices from this collection are recommended for a proper experience.

Shaping an architecture that encapsulates the components mentioned before, the project aims to provide an authentic, modern, engaging, and reliable experience. The design of the applications is intended to reach the highest standards, and the user interfaces should be user-friendly and provide a quick reaction time from the server. The reconstruction component is responsible for the construction of complete 3D models, focusing on fine details to render the optimal shape and textures, while also striving to achieve small processing times. On average, a waiting time under 10 minutes is targeted, as the user experience shouldn't be neglected, and the users usually expect rapid processes. And last, the Flask backend is expected to instrument a reliable communication with the other components, achieving under 0.1 second for data transfer

to the web interface, and under 1 second for the VR application, as a standard practice to keep the attention of the user. [24]

3.1.1 Web Application Implementation Details

The implementation process focused on the user experience. The main idea of the project was to create an accessible and easy-to-use tool, and naturally, the idea of a web application came as a solution as it can be used by both PC users and mobile users. Moreover, a web application is versatile and can be operated on any device that has a browser, even if it is a laptop, an Android smartphone, an iPhone, or a gaming console.

Based on previous experience, the choice for the implementation of such a user interface was the Angular framework. Angular is a JavaScript-based framework that offers rapid development of single-page applications. The application is modular, as the web application encapsulates multiple individual components or modules that can exchange information and routes in order to achieve the desired flow inside the application.

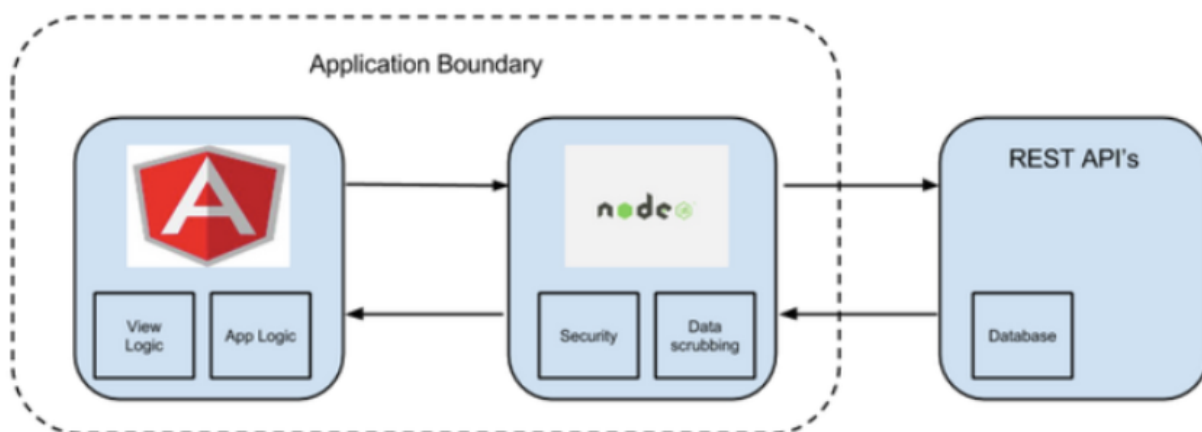
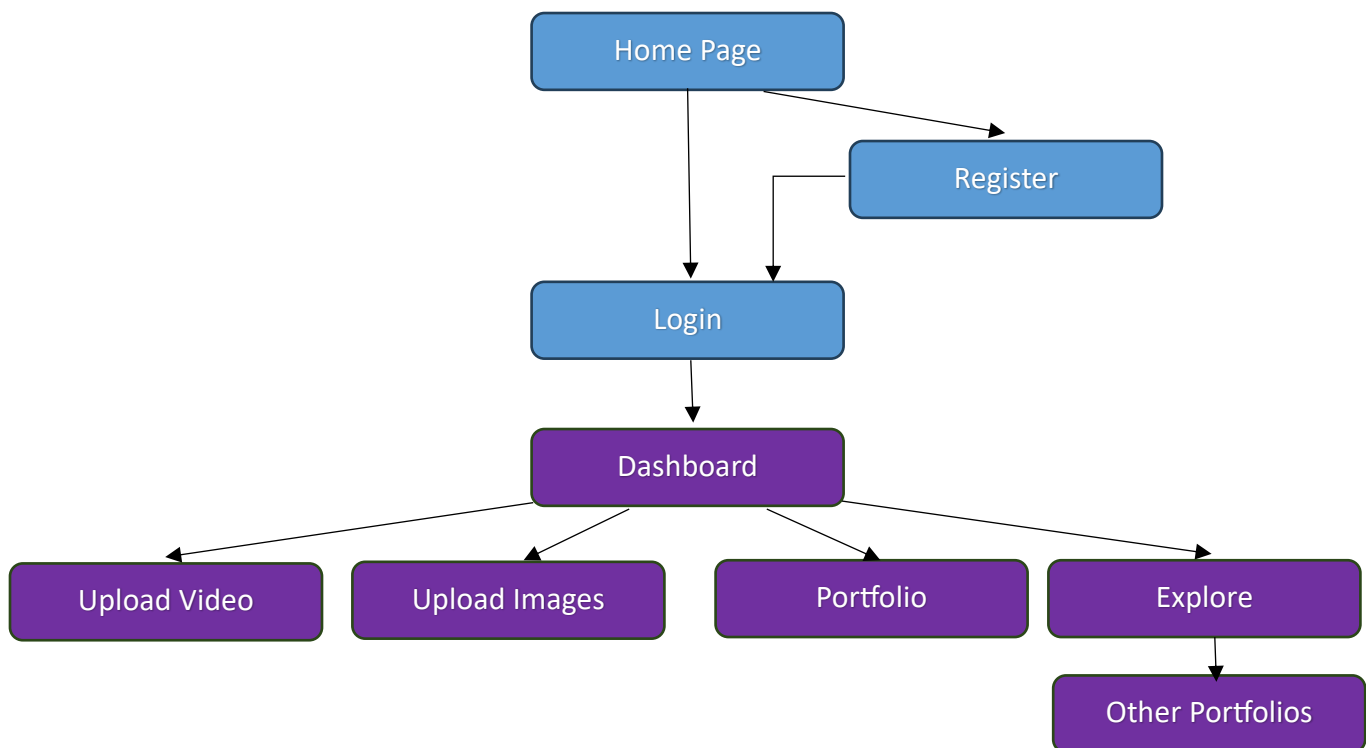


Figure 10 AngularJS Application with NodeJS Gateway

To code such components, the programmer needs to write an HTML file that defines the blueprint of the web page, an SCSS file that is responsible for the design and layout and a TypeScript file that incorporates the logic behind the component. TypeScript is a modern class-oriented programming language that is compiled with JavaScript. The compilation of the code is run on a

Node.js version 18.16.0 server environment and manages the access to the application routes. Moreover, the application's packages are being managed, installed, and incorporated using NPM, version 9.5.1. An overview of the architecture is presented in Figure 10.

3.1.2 Presentation of the Web Application



The hierarchical structure from above describes the user flow inside the application. In the next paragraphs, every component of the user interface will be exemplified and analysed, attaching screenshots from two different types of devices: a Windows laptop using a Chrome-based browser, and an iPhone using a Safari browser.



Figure 11 Laptop (left) and mobile (right) screenshots of the Home Page

First, when opening the link of the web application, the first thing the user will see is the home page. The home page is presented in Figure 11 using screenshots of the two versions of the web page. As presented in the schema from above, inside the home page there are buttons that open the Login and Register routes. When one of the buttons is clicked, the browser redirects the user to the web component that is responsible for the selected task.

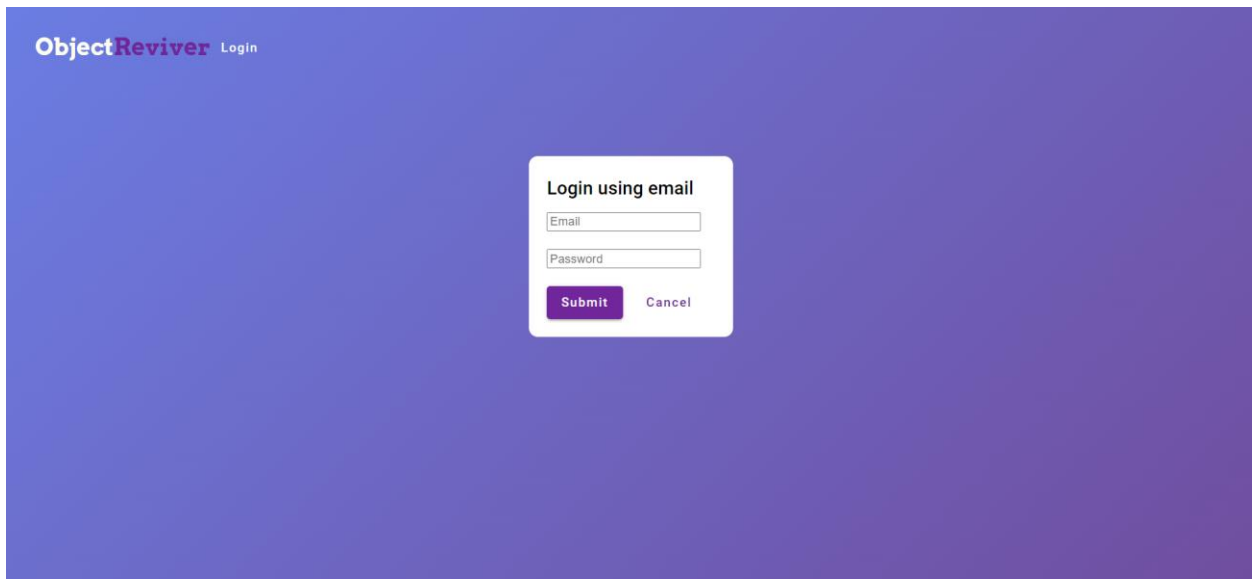
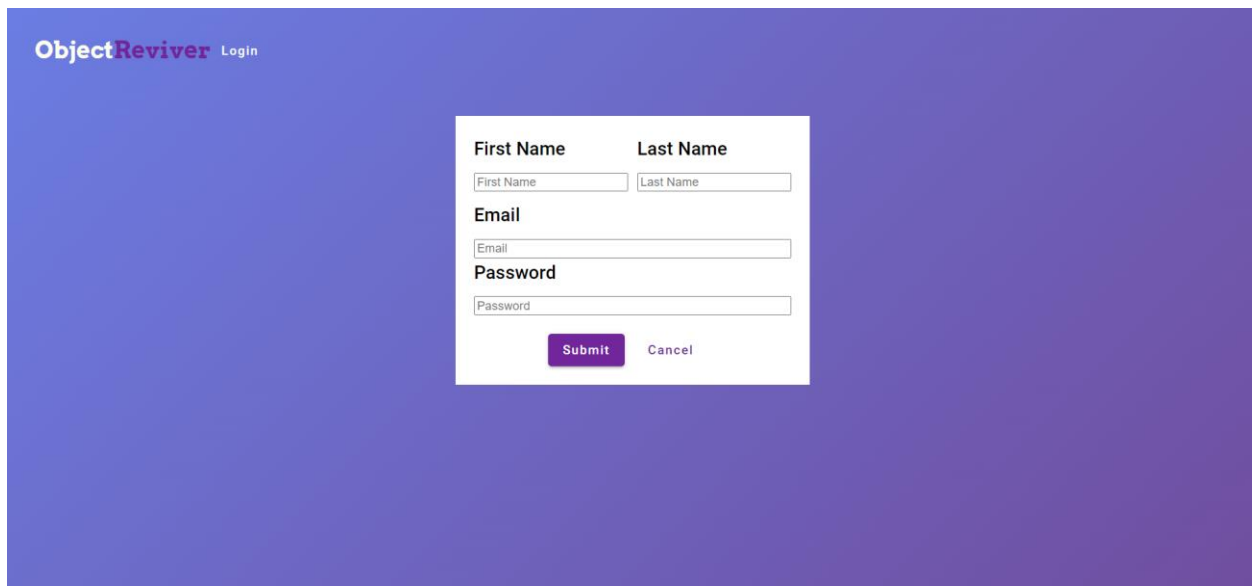


Figure 12 Laptop screenshot of the Login page



ObjectReviver Login

First Name Last Name

First Name Last Name

Email

Email

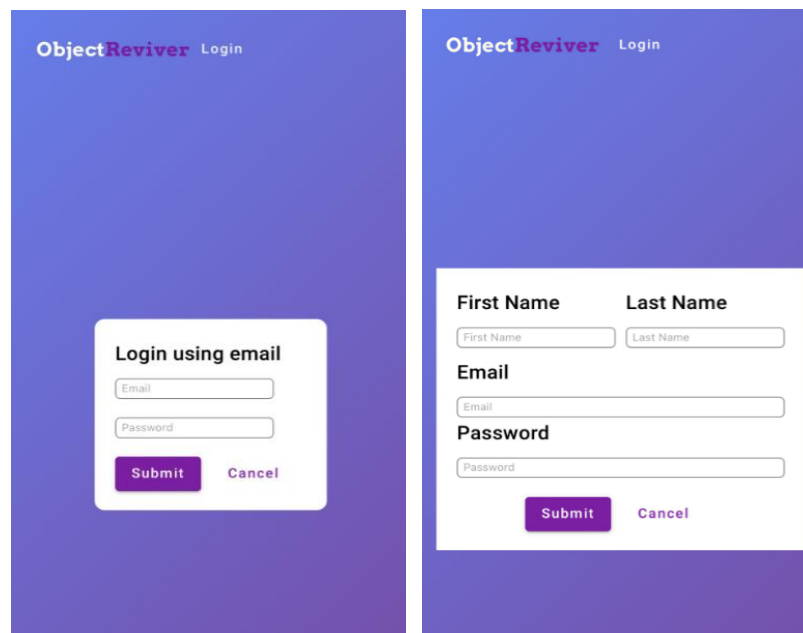
Password

Password

Submit Cancel

Figure 13 Laptop screenshot of the Register page

The Login and Register components have similar structures and designs, as both encapsulate forms where the users type their credentials. The difference between the two pages is that the registration form contains two more input fields for the first and last names of the user, names that will be displayed while exploring the application and will be public to the other users.



ObjectReviver Login

ObjectReviver Login

Login using email

Email

Password

Submit Cancel

First Name Last Name

First Name Last Name

Email

Email

Password

Password

Submit Cancel

Figure 14 iPhone screenshots of the Login page (left) and the Register page (right)

Figure 12 illustrates the laptop version of the Login page, while Figure 13 presents the laptop version of the Register page. The mobile version is also presented, with screenshots of both pages, in Figure 14.

The login process involves the completion of the input fields for email and password. When finished, the submit button is clicked to complete the authentication. When the button is clicked, the TypeScript unit from the component gets the information from the field and passes them to the backend, using the “/login” route of the API.

In the running process of the Flask backend, the function corresponding to the “GET” HTTP method of the “/login” route is triggered. This function decrypts the credentials passed from the user and sends an enquiry to the database using a MySQL connector. The query checks if the table responsible for the indexation of users contains a pair of user ID – password. The result of the search is returned to the backend.

If the search has a result, then the backend returns a favourable message to the frontend. If not, a corresponding message is returned. When the TypeScript unit of the component receives the message, redirects the user to the Dashboard component if the authentication is successful, or remains on the same page if the credentials did not match with the ones from the database.

On the other hand, the Registration does not involve querying the database for a possible match in credentials but creating another entry in the corresponding table. Therefore, after the completion of the input fields of the form and the clicking of the submit button, the TypeScript unit of the registration component sends the information from the form to the “/register” route of the API.

The information is gathered in JSON format by the function responsible for the “POST” HTTP method of the “/register” route. This function creates a MySQL query that inserts the information received in the table that indexes the users with the corresponding values of the columns. After that, the backend sends a message to the frontend according to the successful completion of the task. The registration TypeScript unit will then receive the message, and if the operation is successful then it will redirect the user to the login component.

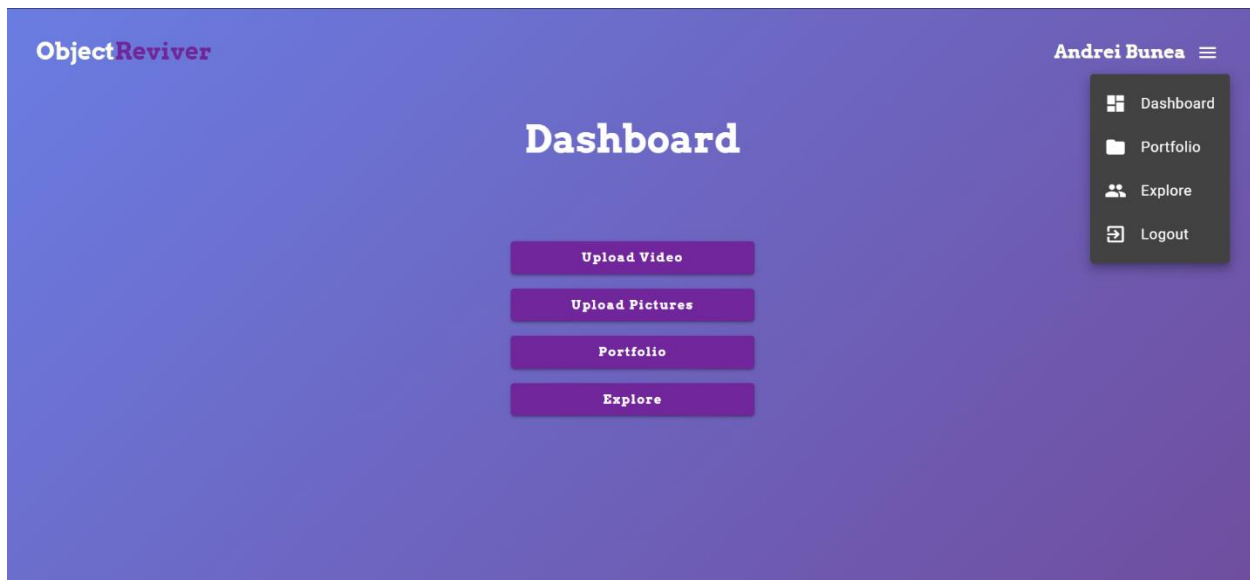


Figure 15 Laptop screenshot of the Dashboard page

The Dashboard is the component that organises every other main component of the application. It presents the principal functionalities inside the application and provides buttons with routes to corresponding web pages. The desktop version of the Dashboard page is illustrated in Figure 15, while the mobile version is presented in Figure 16.

Every web page, starting from the Dashboard, has a common component: the navbar. This component is situated on the very top of the page and contains the logo of the application, the name of the user who is logged in and a drop-down menu. When the three, white parallel lines icon is clicked, the visibility of the menu is triggered.

The menu is implemented as a list with the 3 main pages of the application (Dashboard, Portfolio and Explore), and also the Logout

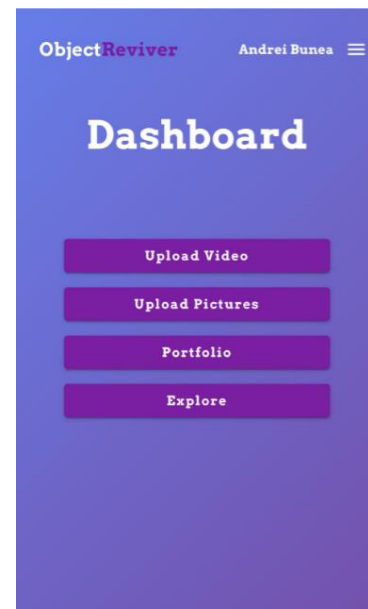
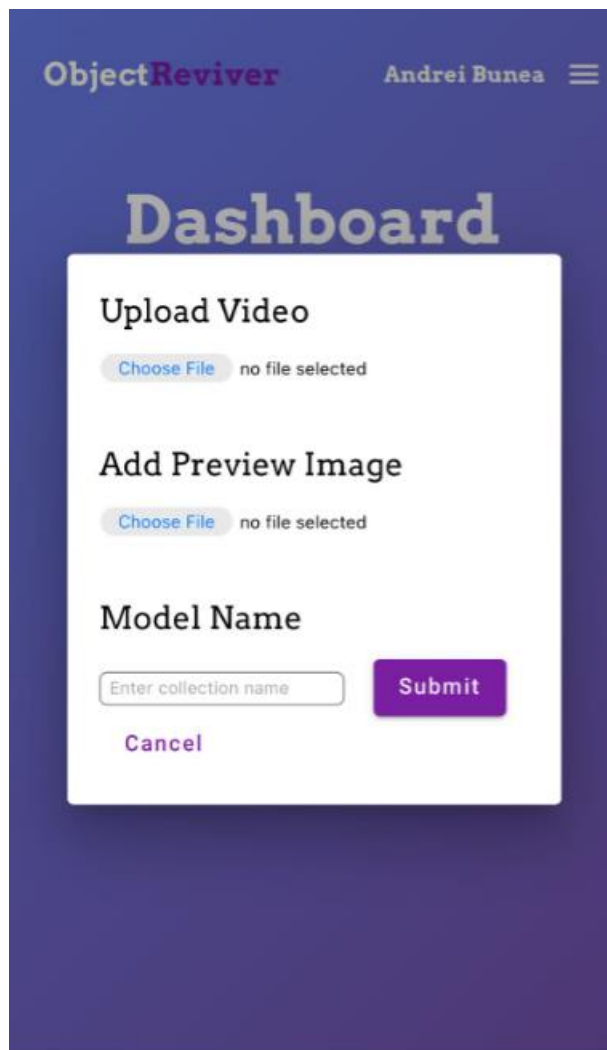


Figure 16 Mobile screenshot of the Dashboard Page

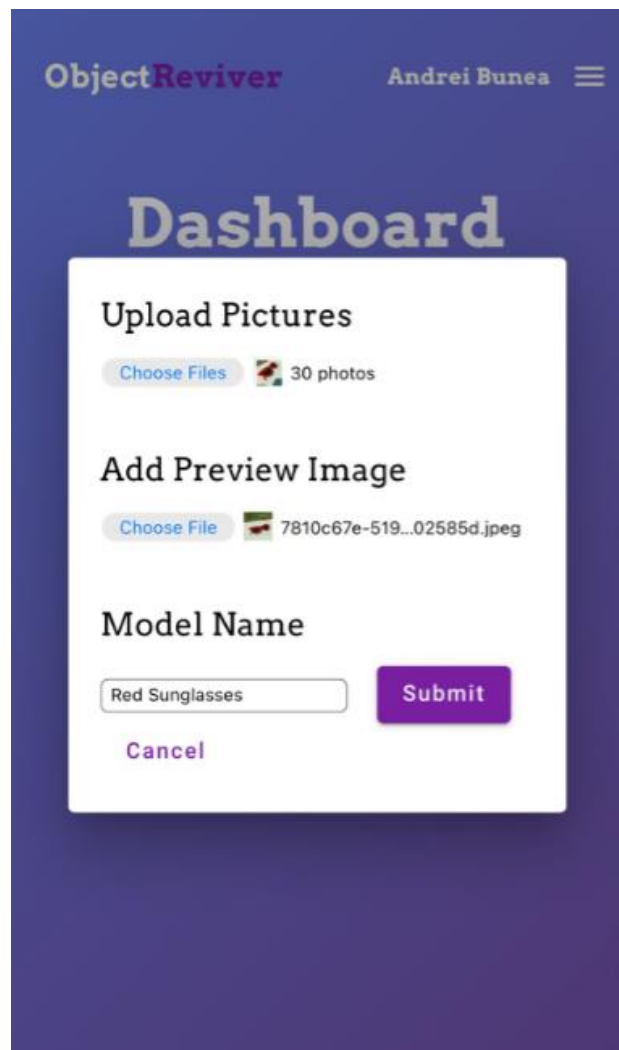
functionality that ends the session for the current user and redirects to the home screen. To exemplify the actionable menu, the icon has been clicked in Figure 15.

To explore the functionalities of this web page, the report will analyse each one of the four functionalities offered by the Dashboard page. The first two buttons are intended for initialising the process of 3D model reconstruction. To start the process, the buttons open a modal containing a form where the user needs to upload a video or a set of images, depending on which one of the two buttons he clicks.



The screenshot shows the 'ObjectReviver' dashboard with the user 'Andrei Bunea'. A modal titled 'Dashboard' is open, specifically the 'Upload Video' section. It contains a 'Choose File' button with the text 'no file selected' next to it. Below this is the 'Add Preview Image' section, also with a 'Choose File' button and 'no file selected' text. At the bottom, there is a 'Model Name' section with a text input field containing 'Enter collection name', a purple 'Submit' button, and a purple 'Cancel' button.

Figure 17 Upload Video modal



The screenshot shows the 'ObjectReviver' dashboard with the user 'Andrei Bunea'. A modal titled 'Dashboard' is open, specifically the 'Upload Pictures' section. It contains a 'Choose Files' button with a photo icon and the text '30 photos' next to it. Below this is the 'Add Preview Image' section, with a 'Choose File' button and a file name '7810c67e-519...02585d.jpeg' next to it. At the bottom, there is a 'Model Name' section with a text input field containing 'Red Sunglasses', a purple 'Submit' button, and a purple 'Cancel' button.

Figure 18 Upload Pictures modal

Figure 17 and Figure 18 illustrate two modals, each one corresponding to one of the two buttons. The first of the figures shows the uncompleted form for the submission of a video as input data

for the reconstruction process. The second figure depicts the process of uploading a set of pictures and in addition to the first image, the form is completed with an example of inputs.

As observed, the “Upload Video” button opens a modal where the first item to be uploaded is a video file, while the “Upload Pictures” button opens a modal where the user must upload a set of images from the device in use. This is the only difference in the structure of the form. All the other items are identical between the two modals. Both require the user to upload a preview image. This image is used to present the 3D model to the other users without clicking to interact with the object. It is also used to highlight that the model reconstructed from the uploaded data is still in the process of reconstruction, as the construction of the 3D model can take some more time in some cases. Thus, when the model is still in the process of being accomplished, the preview image will signal the future presence of an item in the user’s portfolio.

Another item that is required by the modal is the model’s name, in order to facilitate a description of the model for the other users to see. Alongside the preview image, these two inputs are the two key factors that describe a 3D model and highlight the difference between an object and the others. The preview image, the name of the model and the path to the model itself are stored in the database, having a unique ID that is paired with the ID of the user.

After the completion of the form, the “Submit” button triggers the reconstruction process as the information from the form is sent to the API by calling the “/upload/video” or “/upload/images” endpoints.

When the “/upload/video” endpoint of the API is triggered, the corresponding function is responsible for creating an image dataset from the video and sending it as input to the Structure-from-Motion script and storing the result alongside the preview and model name. To extract the image dataset from the video, the function extracts frames from the video at an interval of 1 second. The extracted frames are saved in the image input folder of the reconstruction script.

The function corresponding to the “/upload/images” HTTP request saves the images directly without running any algorithms on the received image dataset. The images are saved in the same folder as the frames extracted from the video. A very important observation is that every time

the reconstruction mechanism is triggered by uploading videos or images, all the directories that contain data from previous inputs are cleared.

After the successful saving of the images in the input directory of the reconstruction script, the backend starts the running process of the script that handles and manages the execution of Meshroom using only the command line interface. More about this script will be covered in a section dedicated just to this process.

After the completion of the 3D model, the result is saved in the local memory of the server running the backend, in a directory created just for the user that owns the model. The result of the reconstruction process is an .obj file that contains the scene with the 3D object, an .mtl file that describes the association between texture and vertices of the mesh, and multiple texture files saved in a .png format.

The files describing the mesh are stored in the local memory of the server because storing them in the database could alter the information from the files. To store such files in a table requires an encoding of the data, and images and .obj files are sensible to this encoding, as the results of the decoding process can lead to data alteration. However, the directory system of the server is well structured, and the application assures good storage of the data, preventing the loss of the information and fault access from other users who do not own the directory.

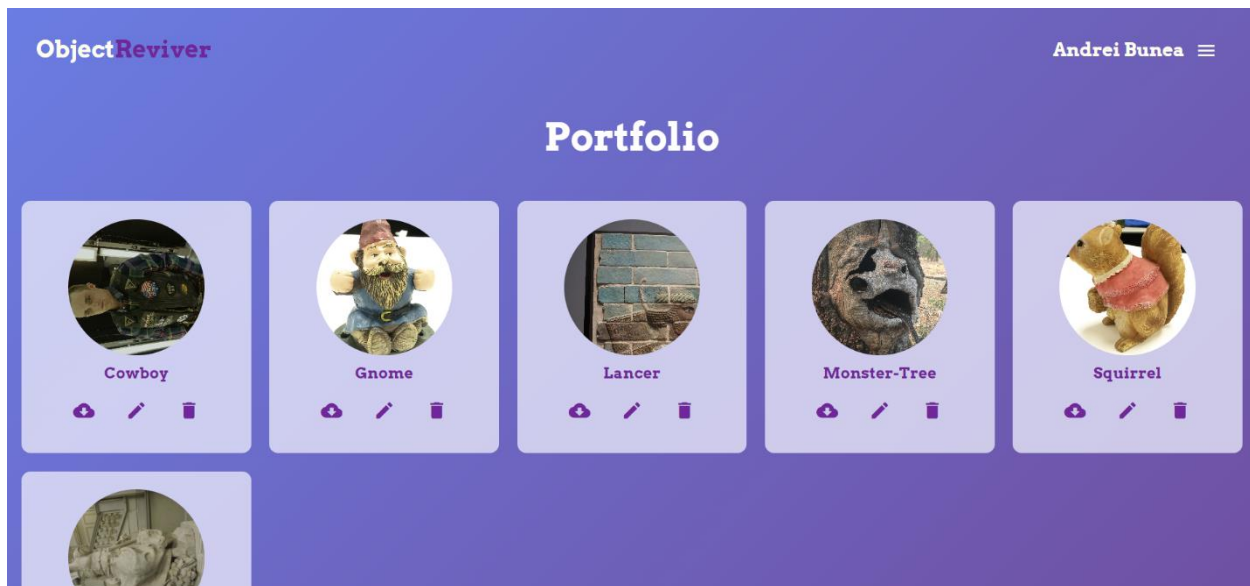


Figure 19 Laptop screenshot of the Portfolio page

Next in line to be discussed is the Portfolio page. This component is designed to present the 3D models created by the user. Every object is displayed in a dynamic grid and is represented by its preview image and model name. For each model, under their name, there are 3 icons that represent actions available to the user. The three actions are: downloading the model as a zip archive, editing the model by modifying the preview image or its name, or deleting the model from the portfolio. The desktop version of the website is illustrated in Figure 19.

Looking at both images, it can be observed that the grid used for listing the models is dynamic, adapting the number of items on each row depending on the width of the screen.

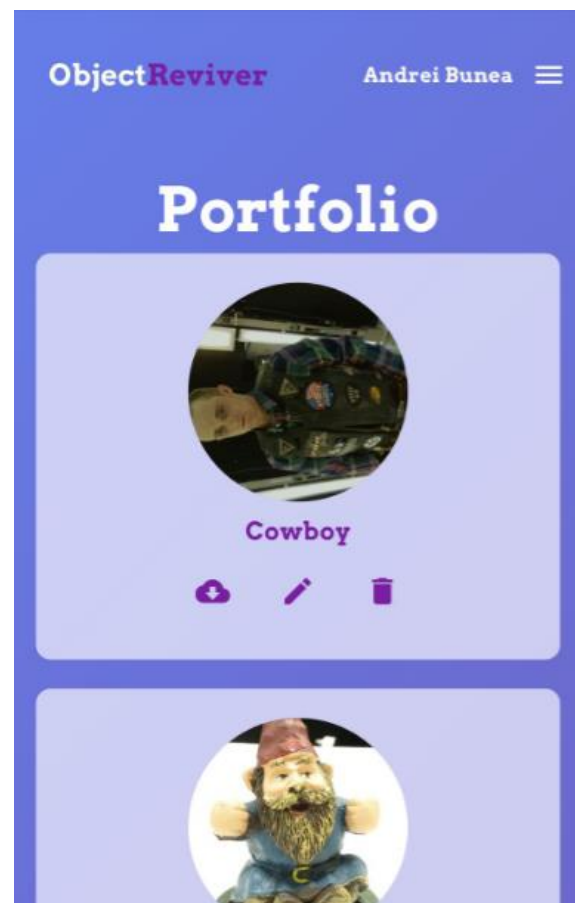


Figure 20 Mobile screenshot of the Portfolio page

Moreover, the Portfolio page offers the opportunity to view the 3D models and interact with them. This action is launched by clicking on one of the objects from the list.



Figure 21 Three.js scene of the Monster-Tree model

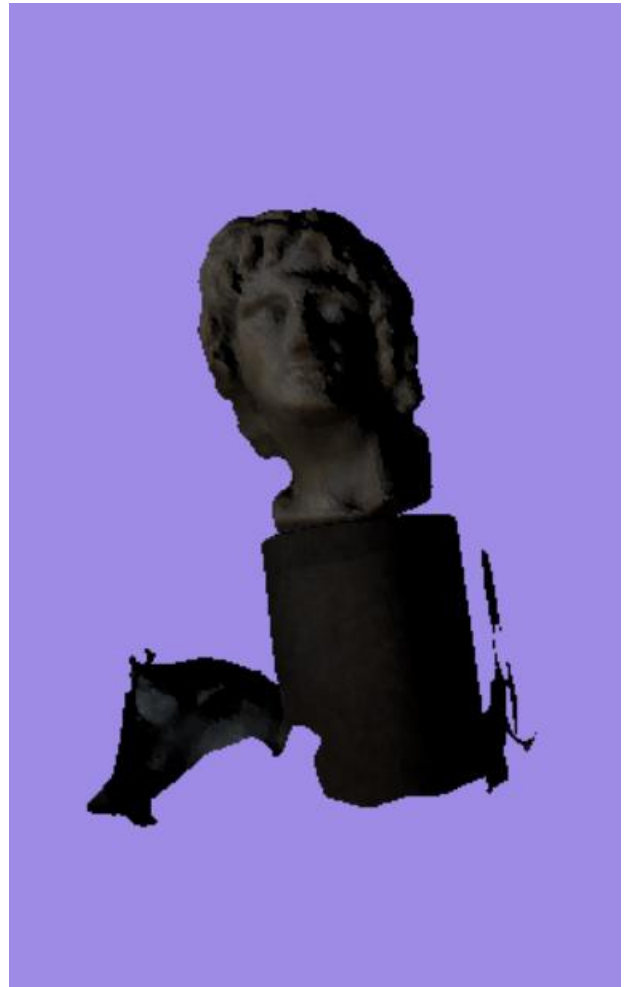


Figure 22 Three.js scene of the Alexander the Great model

In Figure 21 and Figure 22, some examples of this functionality are presented. In these images, two models were clicked for 3D analysis in the virtual scene of the application: a “Monster-Tree” model and a “Alexander the Great” model. Both models were created using the application and its Structure-from-Motion pipeline.

Moreover, the virtual scene rendered inside the application is responsive to the user's actions, and the user can interact with the model by rotating or scaling. These actions are performed by moving the cursor in a direction while holding the click, or the analogue interaction in the case of the mobile version.

The scene is rendered using Three.js inside the Angular application. Three.js is a versatile JavaScript library that empowers the creation of 3D scenes inside applications, providing a simpler version of WebGL tools and functionalities. To achieve this setup for rendering, the TypeScript unit manages the functionalities of the Three.js library.

First, the scene is instantiated using the built-in function 'THREE.Scene()', and the colour of the background is established. Now that the scene is defined, a camera needs to be instantiated and positioned. The model preview component uses the 'THREE.PerspectiveCamera' with the camera frustum vertical field of view set at 75, camera frustum near plane at 0.1 and camera frustum far plane at 1000. The renderer, a 'THREE.WebGLRenderer' instance, is initialized and associated with the HTML container. This renderer is responsible for converting the 3D scene into visible content on the user's screen. The camera's position is set to provide a suitable view of the scene, and a window resize event listener is added to maintain responsiveness. In addition, a directional light is inserted in the scene to lighten the 3D models.

Next, the model needs to be loaded using its data files and rendered inside the scene. The THREE.js loaders 'OBJLoader', 'MTLLoader', and 'TextureLoader' retrieve data from the .obj, .mtl, and texture files, respectively. After the mesh is correctly constructed and the textures are mapped accordingly, the 3D object is added to the scene.



Figure 23 Rotated model of a statue using Orbit Controls

The interaction is provided by adding Orbit Controls from the THREE.js library, allowing users to explore the 3D scene. These controls come with various parameters for damping, rotation speed, and zooming. Their event listeners ensure that user actions trigger the correct modifications to the rendered scene. An example where the object has been rotated is illustrated in Figure 23.

In addition, an animation loop is implemented, refreshing at every frame. Within this loop, the 'controls.update()' method keeps the controls responsive, and the render function is invoked to refresh the scene rendering. This dynamic combination brings life to the 3D model, making it interactive and engaging.

This model preview component not only allows the user to interact with the scanned item but also provides the opportunity to quickly observe the correctness of the model. Thinking of a use case of the application where the user uploads a dataset of images for reconstruction, accessing this feature at the end of the SfM process indicates if the results are satisfying or if the user should try again, using another set of images. By using this feature, the user can learn which parts of objects to focus on when taking pictures, paying attention to specific details.

The last button in the Dashboard menu is the Explore one. When clicked, the button redirects the user to the explore page where the user can search for others and see the pages that he is following. To avoid losing track of the creators that you like, the application introduces a mechanism for following other pages.

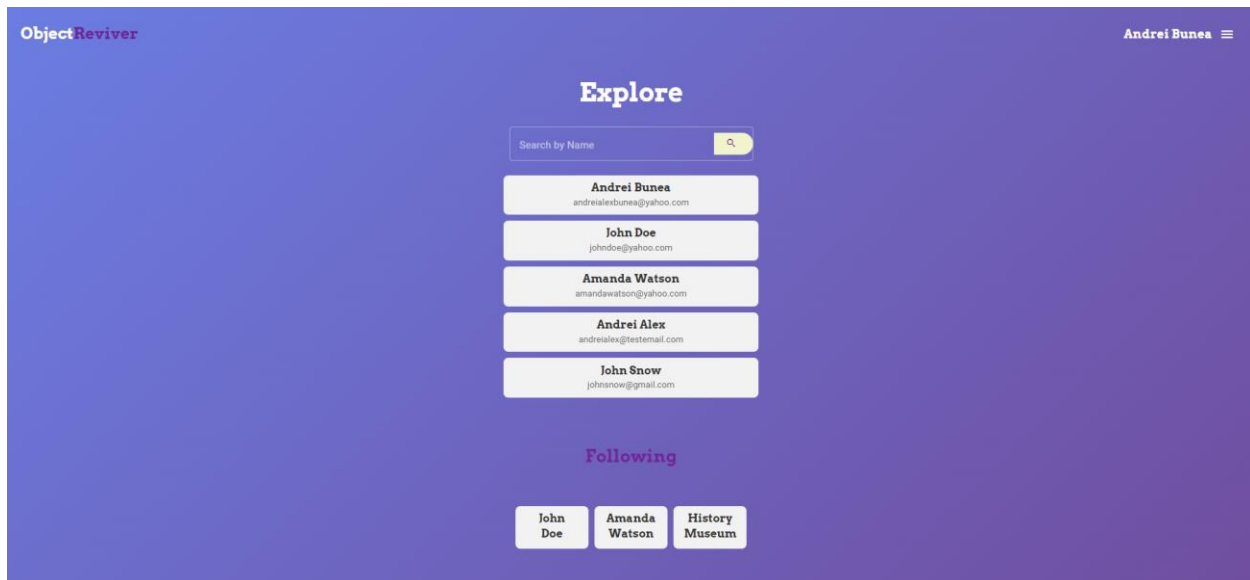


Figure 24 Screenshot of the Explore page

In Figure 24 the Explore page is presented, displaying the section for user search and the section for checking on the followed users. The user search is done by extracting the words from the text input and searching in the database users that have either the first or the last name like any of the words.

An example of a search is presented in Figure 25, where the searched user was “Andrei”, and the application returned the only two users that have this word in their name. The first user is highlighted with red because it is hovered by the mouse, and when hovered buttons have this effect applied on.

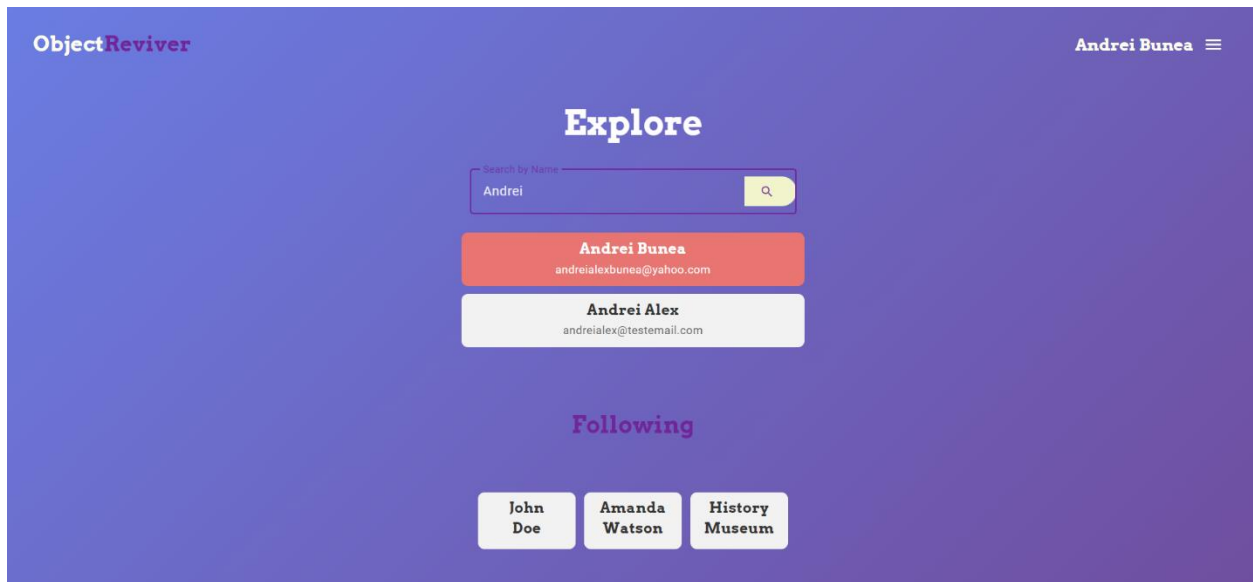


Figure 25 Searching for a user

When one of the users is clicked, their profile is opened for the current user to explore. On top of the page, it displays information about the user such as their name and email, and indicates if the current user follows the page or not. Below these, a grid with the owned models is displayed, similar to the one from the portfolio page with models owned by the current user. The difference between this grid and the previous one is the lack of accessibility in terms of downloading, editing, and deleting the model. Naturally, a different user from the owner of the model should not have access to such actions upon a foreign item.

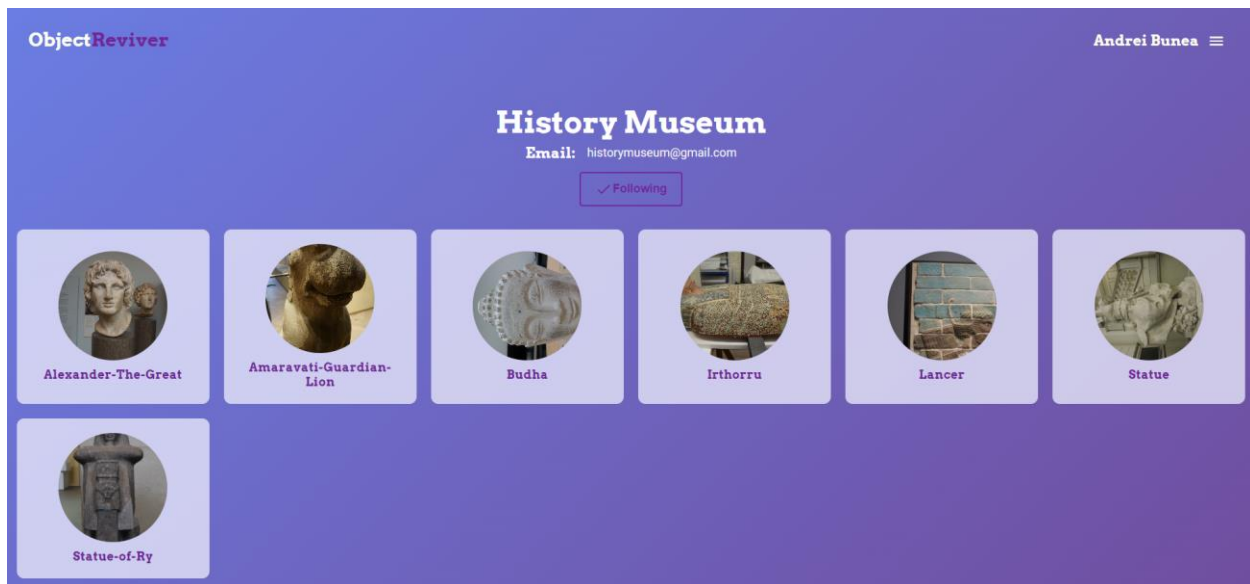


Figure 26 Profile of another user

Figure 26 illustrates how the profile of the “History Museum” user is presented to the current user. Even though the models are not editable by the user, and they cannot be downloaded or deleted, the user can click on any item to open the model preview component, which uses THREE.js to render a scene encapsulating the selected item.

In addition, the user can follow or unfollow any user at any time. The user relationships are stored in the database, with their IDs mapped to each other.

In the end, the user can click the Logout button from the dropdown menu to leave the current session.

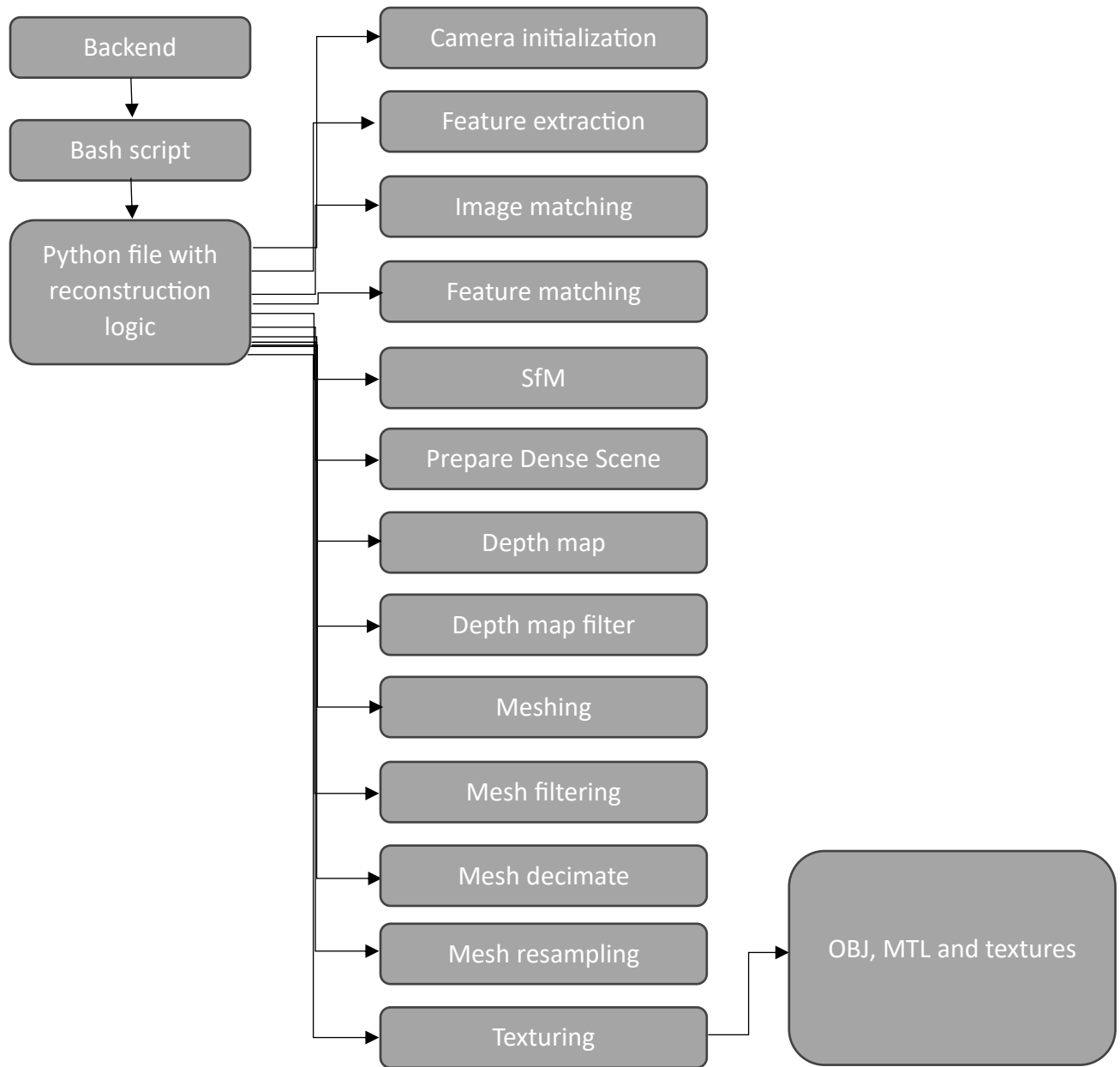
3.2 The Structure-from-Motion pipeline

The reconstruction of the 3D model relies on the input dataset (images or video) uploaded by the user and is facilitated by the Meshroom software. However, Meshroom is designed as a GUI application, and most of the users explore the software by navigating through the desktop application. To use this program as part of the project, the main challenge was to automatise the reconstruction process, without any human input.

The automatization has been realised by manipulating the reconstruction scripts of the open-source software, using a script. Every processing step of Meshroom has its own Python executable, and thus instead of using the GUI of the application, the creation of 3D models can be done by running one by one, in chronological order, the scripts that are responsible for the Structure-from-Motion pipeline.

After creating a Python file to implement the reconstruction logic and defining the parameters and algorithms used in the Meshroom scripts, a bash script is created to execute the process. The bash script is designed to run on Windows and is responsible for running the logic of the custom reconstruction Python file. In addition, the script passes to the Python executable the location of the input dataset, the location of the output directory that stores the result, and the location of the Meshroom installation directory.

To run this script, the Flask backend simply starts a new system process that runs the executable bash file.



The previous figure describes the whole process, underlining the steps of the reconstruction and the relation between the components involved. Further on, the report is going to focus on each step of the reconstruction pipeline, highlighting the choice of parameters, algorithms and other processing techniques used.

3.2.1 Camera Initialization

The first step of the 3D reconstruction workflow is the camera initialization. This step is performed to estimate the initial camera positions and orientations in the virtual scene. This step is crucial, as it represents the starting point of the future processes. The camera initialization process is driven by the images themselves and relies on metadata from the images to estimate the camera parameters. The process proceeds as follows:

- Establishing the input images. Each one of the images from the dataset captures different views of the scene, at different angles.
- Using Meshroom's sensor database to approximate camera parameters. The sensor database stores camera sensor and calibration data for various camera models, aiding in the precise estimation of camera parameters based on sensor size, lens distortion, and other characteristics. It enables software to understand how light passes through the lens and is recorded by the sensor.
- Estimation. Meshroom uses "Bundle Adjustment" during camera initialization to refine estimated camera poses by iteratively minimizing differences between projected 2D image points and actual image features.
- Initial Approximation. Meshroom creates an initial approximation of camera poses before feature extraction. This approximation assumes a rough camera alignment, such as all cameras looking at the centre of the scene, to set a baseline for subsequent adjustments.
- Refinement. As the process continues further, the software considers the features extracted in the next steps and their 3D positions are estimated using the determined camera geometry. Iteratively, the Bundle Adjustment mechanism refines the camera poses to achieve the best possible alignment between the projected features and their true positions extracted from different images.

The parameters of this first step that have been defined by the Python script are:

- Image Folder: Path to the folder containing input images.
- Sensor Database: Path to the Meshroom sensor database.
- Output: Path to the directory that stores the camera initialization results.
- Default Field of View: A default field of view of 45 degrees is chosen, as it provides a balanced view without excessive distortion or narrow focus. It is suitable for a wide range of scenes.
- Allow Single View: This parameter is set to 1 to allow the reconstruction to proceed even if only a single view is available. This can be useful for situations where some images fail to produce features or matches.

The result is a 'cameralnit.sfm' file containing the estimated camera poses and orientations for each image.

3.2.2 Feature Extraction

In Step 2 of the Meshroom photogrammetry workflow, feature extraction is performed to identify and extract distinctive features from the input images. The features are then stored in files that describe their characteristics. These features play a crucial role in subsequent steps like feature matching and 3D reconstruction.

The features extracted are represented as key point descriptors. These descriptors are numerical representations that encode the information about the key point's location, orientation, and appearance. In this project, the SIFT descriptor has been used for feature extraction and matching. The SIFT descriptor encodes key point information compactly and distinctively, allowing for efficient matching across images.

The output files of the extracted features include information about feature location, orientation, descriptor, scale, and other metadata. The extracted feature files are saved in a designated directory, and each image usually has its corresponding feature file. These files are created during the feature extraction process and serve as the input for subsequent steps like feature matching.

The parameters defined for the Feature Extraction step are:

- Input: Path to the camera initialization file (cameraInit.sfm) generated in the previous step. This file contains the estimated camera poses and orientations for each image.
- Output: Path to the directory where the feature extraction results will be stored.
- Force CPU Extraction: This parameter is set to 1, indicating that the feature extraction process should be performed using the CPU rather than any GPU acceleration. This can be useful when GPU resources are limited or unavailable.
- Range Start and Range Size: These parameters are used to specify a subset of images for processing in case the total number of images is large. Images are processed in batches to manage memory and computation resources. In the project's script, the threshold for starting to process the images in batches is 40 images.

3.2.3 Image Matching

In the third step of the workflow, the goal is to establish correspondences between distinctive features extracted from different images in the dataset. These correspondences are necessary for determining how different images overlap and contribute to the 3D reconstruction. To achieve this, the process of image matching involves finding pairs of images that share common features and matching those features to establish reliable connections between images.

The image matching process proceeds as follows:

- Input Data: The input for this step includes the camera initialization results obtained from the first step and the extracted feature files generated in the second step.
- Features Matching: Meshroom utilizes the information from the feature files to find matching features between pairs of images. By comparing the descriptors of key points, the software identifies features that likely correspond between images. This process helps create a set of feature matches that can be used to estimate the relative poses of the cameras.

- Geometric Verification: To ensure the accuracy of the matches, geometric verification techniques are applied. These techniques help filter out incorrect or noisy matches. One common approach is using the fundamental matrix, which relates the coordinates of matched points between two images.
- Output: The result of the image matching step is a file named "imageMatches.txt." This file contains information about the matched features and their correspondences between pairs of images. It serves as input for the subsequent feature matching step.

The parameters defined for the Image Matching step are:

- Input: Path to the camera initialization file ('cameraInit.sfm').
- Features Folders: Path to the directory where the feature extraction results are stored.
- Output: Path to the directory where the matched features and their correspondences will be saved.
- Tree: Path to the tree structure used for efficient feature matching. In this case, it is pointing to a pre-defined tree structure.

3.2.4 Feature Matching

In the Feature Matching phase, the process focuses on establishing correspondence between key points extracted from candidate image pairs. This intricate step contributes significantly to the creation of a coherent 3D reconstruction. Using the files of the extracted features, Meshroom locates matching templates across pairs of images by evaluating key point descriptors and assembling likely corresponding pairs of features.

From pairs of matched images from the previous step, photometric matches between the sets of descriptors from the two images are analysed. For every feature from the first image, a list of features from the second image that match the first feature is extracted. As multiple features could match with the first one, thresholding based on neighbouring descriptors from the two images is implemented, as described by David Lowe in "Distinctive image features from scale-

invariant keypoints” [25]. As this process is computationally intensive, optimization algorithms such as Approximate Nearest Neighbour and Cascading Hashing [26] are applied.

After that, geometric filtering is applied to the features, using the outlier removal algorithm RANSAC from epipolar geometry. The essential matrix is determined using a small set of correspondences. Afterwards, the essential matrix is used to find the features that validate the model after the RANSAC iteration.

The outcome of this step is a compilation of feature matches, essentially creating a map of relationships between key points across the various images. This map is instrumental in estimating the relative poses of the cameras capturing the images, a crucial aspect for the subsequent phases of the workflow.

The parameters that govern the "Feature Matching" step include:

- Input: Path to the camera initialization file.
- Output: Path to the directory where the feature matching results will be stored.
- Descriptor Types: Specifies the type of feature descriptor used for matching. In this project, the SIFT descriptor is used.
- Photometric Matching Method: Determines the photometric matching algorithm used to establish feature correspondences. In this project, the value "ANN_L2" is set, which stands for Approximate Nearest Neighbour with L2 distance. This approach uses fast approximate search methods to find matches, resulting in both speed and accuracy.
- Geometric Estimator: Defines the geometric estimation method employed to refine feature matches. In this project, "acransac" (A-Contrario RANSAC) is used.
- Distance Ratio: A threshold parameter used to filter and retain the most reliable matches. The value 0.8 has been set for the distance ratio between the first- and second-best matches considered. This helps to filter out ambiguous matches and retain more confident correspondences.
- Max Iteration, Geometric Error, Max Matches, and other parameters: Control various aspects of the matching process, such as iteration count and error thresholds.

3.2.5 Structure-from-Motion

The fifth step of the pipeline aims to determine a rigid structure of the 3D scene and internal calibration of all the cameras by deducing the geometric relationship between all the features in the input images. This process is iterative, starting with a two-view reconstruction and adding neighbouring views until the scene is complete.

Tracks representing points visible from multiple cameras are created by fusing all feature matches between image pairs. After defining the tracks, the algorithm must choose the best initial image pair as a cornerstone event that dictates the final quality of the reconstruction.

The image pair should have robust matches and reliable geometric information, maximizing the number of matches and corresponding features. In addition, a large enough value for the angle between the two cameras is important as it should provide reliable geometric information.

Next, the fundamental matrix between the two images is determined, considering the first image as the origin of the coordinate system. Calculating the fundamental matrix, the model now can determine how the 2D points are transposed in the 3 dimensions, determining the 3D coordinates of the features from the images.

After calculating the fundamental matrix, the algorithm selects the images that have strong associations with the already reconstructed 3D points. This step is done using the next best views selection algorithm. Based on these associated images, the algorithm performs the resectioning of the cameras of the corresponding images. This refinement is conducted using a Perspective-n-Point algorithm with RANSAC, and the goal is to find the parameters of the camera that validate most of the associations between features.

After this step, the algorithm performs Bundle Adjustment to refine the intrinsic and extrinsic parameters of all the cameras involved as well as the positions of the 3D points from the scene. As more points are added to the scene, more images would match the features already present in the reconstruction. Iterating the addition of new views from images, creating more and more 3D points in the mesh, the algorithm continues until no other views can be added to the scene.

The parameters used for this step are:

- Input: Path to the camera initialization file.
- Output: Specifies the directory where Structure from Motion results will be saved, including reconstructed cameras, poses, and scene structure.
- Output Views and Poses: Specifies the directory to save camera views and poses from Structure from Motion.
- Extra Info Folder: Specifies the directory where additional information related to the reconstruction process will be saved.
- Features Folders: Specifies the directory containing the feature information extracted in the Feature Extraction step. This data is used for matching and reconstruction.
- Matches Folders: Points to the directory holding the feature matching results generated in the Feature Matching step. These matches are used for estimating camera poses and scene structure.

3.2.6 Dense Scene Preparation

This step prepares the data for dense reconstruction. It uses the 'sfm.abc' file generated from the previous step. The result is a set of depth maps corresponding to the images. During this step, the accurate camera poses, orientations, and scene structure obtained from the structure from motion phase are used to create a strong foundation for generating detailed depth maps. This preparation ensures that subsequent stages operate on a well-organized and coherent scene, leading to more accurate depth map generation.

3.2.7 Depth Map

The goal of the Depth Map step of the pipeline is to retrieve the depth value of each pixel for all the cameras that have been calibrated in the Structure from Motion step. To achieve this, Meshroom uses an implementation of the Semi-Global Matching algorithm as presented by Heiko Hirschmuller in "Accurate and efficient stereo processing by semi-global matching and mutual information" [27].

A set of N cameras closest to each image is selected to create fronto-parallel planes. These planes intersect with the optical axis of neighbouring cameras and produce volumetric volumes with multiple depth candidates per pixel. The Zero Mean Normalized Cross-Correlation algorithm is then used to estimate the similarity of each candidate point, creating a volume of similarities. The volume is filtered on both the X and Y axes to identify the local minima, and the depth value is stored in a depth map. A refinement step is applied to achieve sub-pixel accuracy.

The parameters used at this step are:

- Input: Path to the 'sfm.abc' file obtained from the Structure from Motion step.
- Output: Path to the directory where the depth map results will be stored.
- Images Folder: Path to the folder containing images in .exr format resulted after the preparation of the dense scene.
- Downscale: A factor by which the input images are downsampled before depth map estimation. The value is set to 2, which means the images are halved in resolution. This downscaling helps in speeding up the computation and reducing memory usage, while still providing a reasonable level of detail.

3.2.8 Depth Map Filtering

The Depth Map Filtering step aims to resolve the consistency of the original depth maps. Certain depth maps will claim to see areas that are occluded by other depth maps. The filtering step isolates these areas and forces depth consistency.

The process involves applying filtering techniques to smooth noisy depth values, ensuring that neighbouring pixels have consistent depth values. Due to occlusions, reflections or other artifacts, outliers can be produced leading to inaccurate reconstructions. These outliers are eliminated at this step, using outlier removal techniques.

The parameters passed to this process are the location of the 'sfm.abc' file resulted from the structure from motion step, the location of the output directory where the results should be stored and the path to the directory of the depth map results.

3.2.9 Meshing

At this step, information from structure from motion point cloud and depth map is used to generate an initial mesh of the scene.

First, data from all the depth map files is fused into an octree. The cells of the octree store compatible depth values. Next, the algorithm performs a 3D Delaunay tetrahedralization, and the weights of the cells and the facets that connect them are determined based on a voting procedure, as explained by Jaconsek and Pajdla in “Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces” [28]. This procedure enhances the accuracy of the mesh by considering the reliability of the depth data.

To optimally cut the volumes, a Graph Cut Maxx-Flow algorithm is applied [29]. This algorithm aids in carving out the mesh surface from the volume, ensuring an accurate representation of the scene's geometry.

After extracting the initial mesh, a refinement step is carried out to address local imperfections. Laplacian filtering is applied to the surface of the mesh, helping to smooth and enhance the overall quality of the mesh. This filtering process aids in removing small-scale irregularities and further improving the visual fidelity of the reconstructed scene.

In essence, the meshing step transforms the combined depth information and point cloud data into a detailed and coherent mesh representation of the scene. This foundational mesh serves as the basis for subsequent stages of the 3D reconstruction process.

The parameters used for meshing are:

- Input: The path to the SfM results file.
- Output: The path to the directory where the meshing results are stored.
- Depth Maps Folder: The directory containing depth map results from previous steps.
- Max Input Points: This parameter determines the maximum number of input points considered during the meshing process. It allows for controlling the density of the point cloud, which in turn affects the level of detail in the resulting mesh. For this implementation, the value is set to 50,000,000. This choice appears well-considered

as it allows for a substantial number of input points to be utilized in the meshing process. By accommodating a large number of points, the resulting mesh can capture finer details and intricacies of the scene, leading to a more visually accurate representation.

- **Max Points:** The Max Points parameter sets a limit on the maximum number of points that the final mesh should contain. It assists in managing the complexity and size of the generated mesh. In this project, the value is set to 1,000,000. This choice is a practical decision to manage the complexity of the final mesh. By limiting the total number of points in the output mesh, the computational resources required for subsequent visualization and processing are reduced. It strikes a balance between detail and resource efficiency.

3.2.10 Mesh Filtering

In the mesh filtering step, the goal is to refine and enhance the quality of the 3D mesh generated in the previous steps. This step employs various techniques to improve the visual appearance and structural integrity of the mesh. It eliminates noise, fills small holes, and ensures that the mesh is suitable for subsequent applications such as visualization and analysis.

The script uses the following parameters for the mesh filtering step:

- **Input Mesh:** Path to the 3D mesh obtained from the meshing step.
- **Output Mesh:** Path where the filtered mesh will be saved.
- **Keep Largest Mesh Only:** A Boolean parameter that determines whether to keep only the largest connected component of the mesh. It is set to "True" which means that only the largest mesh component will be retained, and any smaller components will be discarded.

3.2.11 Mesh Decimating

Mesh decimation is the process of reducing the number of vertices, edges, and faces in a 3D mesh while preserving its shape and visual quality. The decimation step is reducing the density of the mesh by eliminating certain vertices from the structure. The vertices of the resulting mesh will have the exact same position and characteristics as before this step.

The algorithm involves identifying pairs of vertices that can be collapsed into a single vertex. This selection is based on the calculated error metrics. Vertices with lower error metrics are preferred for collapsing, as they have a smaller impact on the overall mesh quality. For each selected vertex pair, collapse them into a single vertex. This process involves updating neighbouring vertices, triangles, and their connectivity to maintain the mesh's integrity.

After a vertex is collapsed, the neighbouring triangles should be updated to adjust to the new position of the vertex. This ensures that the mesh is free of gaps and overlaps. This is useful for complex and detailed meshes with a large number of polygons, as it improves rendering performance and reduces memory usage.

The script uses the following parameters for the mesh decimation step:

- Input: Path to the 3D mesh obtained from the mesh filtering step.
- Output: Path where the decimated mesh will be saved.
- Simplification Factor: A value between 0 and 1 that determines the degree of simplification. A smaller value preserves more details, while a larger value results in greater simplification. It is set to 0.8, indicating a moderate reduction in complexity.
- Max. Vertices: Maximum number of vertices allowed in the decimated mesh. The script sets the parameter 15000 ensuring that oversimplification of the mesh is avoided.

3.2.12 Mesh Resampling

Mesh resampling is a crucial step in the 3D mesh processing pipeline that aims to further optimize and refine the quality of a simplified mesh obtained through filtering and decimation. This step involves adjusting the vertex density and triangle distribution while maintaining a similar shape to achieve a balance between computational efficiency and visual fidelity.

Mesh resampling often involves subdivision, where new vertices and triangles are introduced to the mesh. This helps improve the representation of complex geometries and maintains important details. During subdivision, vertices are added to the edges and faces of existing triangles. This results in a more uniform distribution of vertices, which can be beneficial for maintaining the shape of the original object.

After introducing new vertices, their positions are adjusted to ensure that the overall shape and curvature of the mesh are preserved. This is often achieved using interpolation techniques that consider nearby vertices' positions and properties. New triangles are formed using the newly introduced vertices along with existing vertices. These triangles help capture finer details and features in the mesh.

During resampling, mesh quality is monitored to prevent issues such as folding, self-intersections, or irregular triangles. Corrective measures are taken if needed to maintain a complete and visually pleasing mesh. The output is a refined and denser mesh that balances visual quality and computational efficiency.

The parameters used in the resampling step are the same as the ones used for decimation. The input parameter points to the mesh resulting from the decimation step, and the output represents the path to the result of the resampling step. The simplification factor is once again 0.8 and the maximum vertices value is 15000.

3.2.13 Texturing

In the texturing step, the script generates textures for the 3D mesh using images from the input dataset. The mesh undergoes a UV unwrapping process, and the textures are mapped according to the UV coordinates. The generated textures are applied to the mesh to provide a visually realistic appearance.

The main task of this step is to associate the textures from images with the geometry of the mesh. This involves the use of the dense point cloud obtained from the meshing step to guide the alignment. Subsequently, textures are generated from the input images and projected onto the mesh's surface based on its UV mapping. The UV unwrapping process determines how textures are laid out in 2D space for proper alignment with the 3D geometry. This process is critical for obtaining optimal results.

The texturing step of the mesh processing pipeline offers three unwrapping methods: Basic, LSCM (Least Squares Conformal Maps), and ABF (As-Rigid-As-Possible Block Optimization). The Basic method is the fastest and simplest method. Its reduced running time was the main reason why this unwrapping technique, along with the fact that it is the only method that can be used for larger models (> 600k faces). LSCM is suitable for models up to 600k faces, and ABF is designed for models up to 300k faces. These two methods focus on space optimization and produce a single texture file, while the basic method can produce multiple files.

All the unwrapping methods mentioned before are based on multi-band blending principles [30], visibility-based weighting, and varying resolution [31].

The accurate UV unwrapping, image-to-mesh alignment, and high-quality textures determine an enhanced sense of realism to the final 3D model, making it suitable for immersive visualizations and various applications.

The script employs the following parameters for the texturing step, each carefully selected to achieve optimal results:

- Input: Path to the dense point cloud obtained from the meshing step. This point cloud serves as a reference for aligning textures to the 3D mesh.

- Input Mesh: Path to the resampled 3D mesh obtained from the mesh resampling step. This mesh acts as the canvas onto which textures will be projected.
- output: Path to the directory where the texturing results will be stored.
- Images Folder: Path to the folder containing the images used for texturing. These images provide the visual information that will be projected onto the 3D mesh.
- Texture Side: The desired size for the generated textures. A value of 4096 pixels is chosen to ensure high-resolution textures, capturing fine details and minimizing pixelation.
- downscale: A factor by which the input images are downscaled before texturing. A downscale value of 4 strikes a balance between maintaining detail and reducing memory and processing demands.
- Unwrap Method: The method used for UV unwrapping, which determines how the 2D textures are mapped onto the 3D mesh. The script opts for the "Basic" method, which provides a straightforward unwrapping process while ensuring consistent and visually appealing texture alignment.

The choice of these parameter values is grounded in practical considerations. The “textureSide” of 4096 pixels ensures sharp and detailed textures, crucial for an enhanced visual appearance. The downscale factor of 4 balances computational efficiency with preserving fine details from the images. The "Basic" unwrap method simplifies the UV mapping process while delivering consistent and realistic texture projection.

The script empirically selects parameter values to ensure the 3D model is visually appealing, accurately aligned, and suitable for various applications such as visualizations and presentations. The result is a .obj file containing the mesh structure, a .mtl file that describes the mapping between texture files and the mesh vertices, and a set of .png images representing the textures.

3.3 Unity VR Application

3.3.1 VR Application Implementation Details

While the web application is designed to focus on uploading and exploring content while interacting with other users' portfolios, the VR application provides an authentic visualisation tool of the 3D models. Facilitating a genuinely immersive experience, the VR application creates the perfect environment where the user can interact with the virtual objects while exploring their fine details.

The VR application has been implemented using the Unity development environment. Integrating the VR controls and packages quickly and accurately, Unity enhances the immersion as it provides critical tools for the simulation of natural behaviour and movement inside the virtual scene. Moreover, the "Ubiq" package implemented by the Immersive Virtual Environments Laboratory of UCL [32] provides built-in methods for the integration of XR controllers and virtual avatar design. The library successfully guarantees the movement inside the scene, by mapping the buttons and triggers from the controllers to movement actions and teleportation. In addition, the project uses the avatar body and hands design implemented in the "Ubiq" library.

The virtual scene of the project is designed using custom primitives or elements from the "Ubiq" package. The logic behind the UI is ensured by C# scripts associated with game objects that control specific actions such as user personal information, retrieval of portfolios and generation of 3D models based on their data files.

All information regarding the user's personal information, portfolio or 3D model files is stored in the database or local server. This information is retrieved from the server at runtime when needed, using the API provided by the Flask backend, while focusing on minimizing the waiting time of the server response. Due to the high memory demand of 3D models as their data files include high-quality textures and complex object structures, effective information transfer strategies from the server are necessary.

The application is developed inside the Unity platform, and the environment's game engine provides the building of the application inside an APK file that is later uploaded to the VR headset.

3.3.2. Presentation of the VR Application

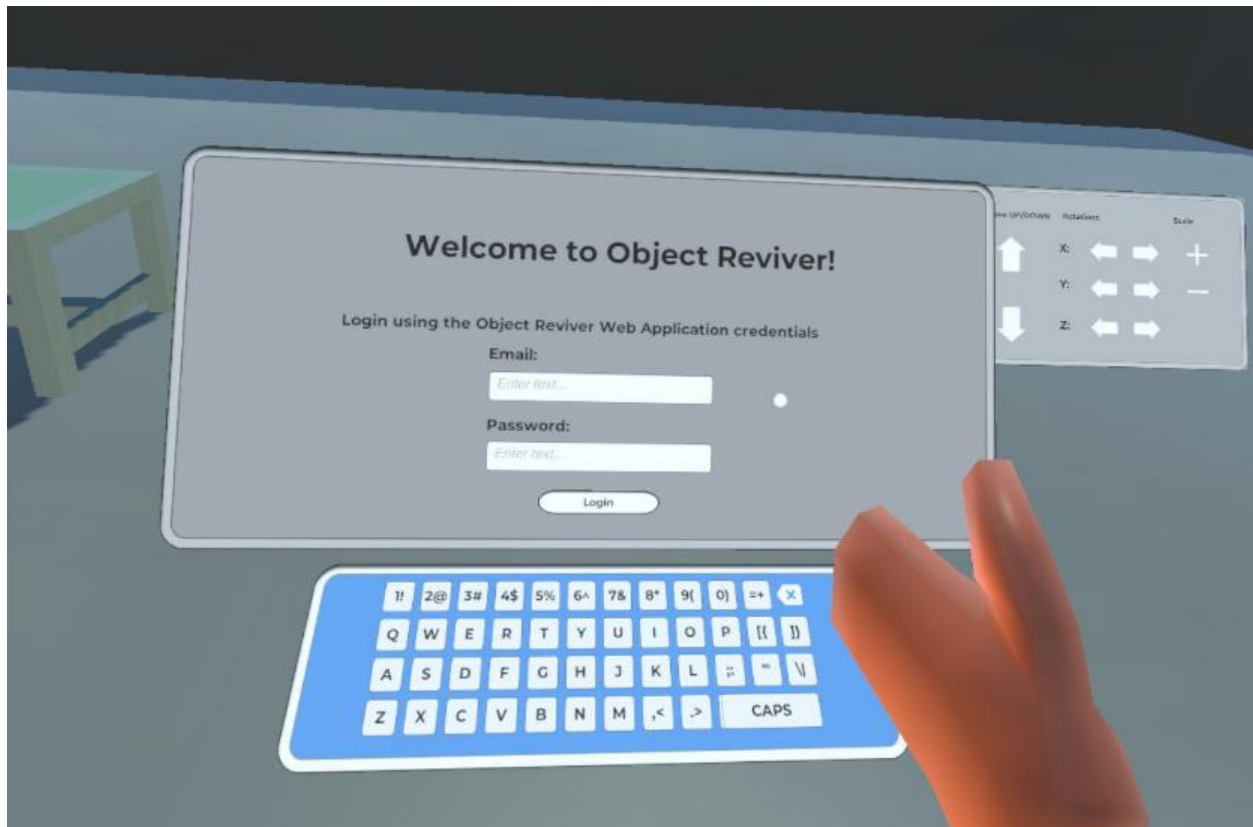


Figure 27 Starting position inside the VR application

When starting the Unity application inside the VR headset, the user is immersed straight inside the virtual scene where all the activities are performed. The scene encapsulates different design items, a keyboard, and panels that control the flow of the application.

The application has two main canvases designed for application control, both of them illustrated in Figure 27. The one that is closest to the user is responsible for all the logic regarding login, portfolio presentation, object spawning and exploring other users. This panel is navigable, and it changes the displayed canvas according to the user input. The second panel is further to the starting point and is designed specifically for the interaction with the object, ensuring the translation, rotation, and scaling of the object.

The keyboard is used for typing using XR controllers inside text inputs. It is implemented using buttons for every key of the keyboard on top of a canvas. All the keys from a physical keyboard are present, as the user can use the CAPS key to type capital letters or symbols above the numeric keys. In addition, it encapsulates a deletion mechanism facilitated by a backspace key.

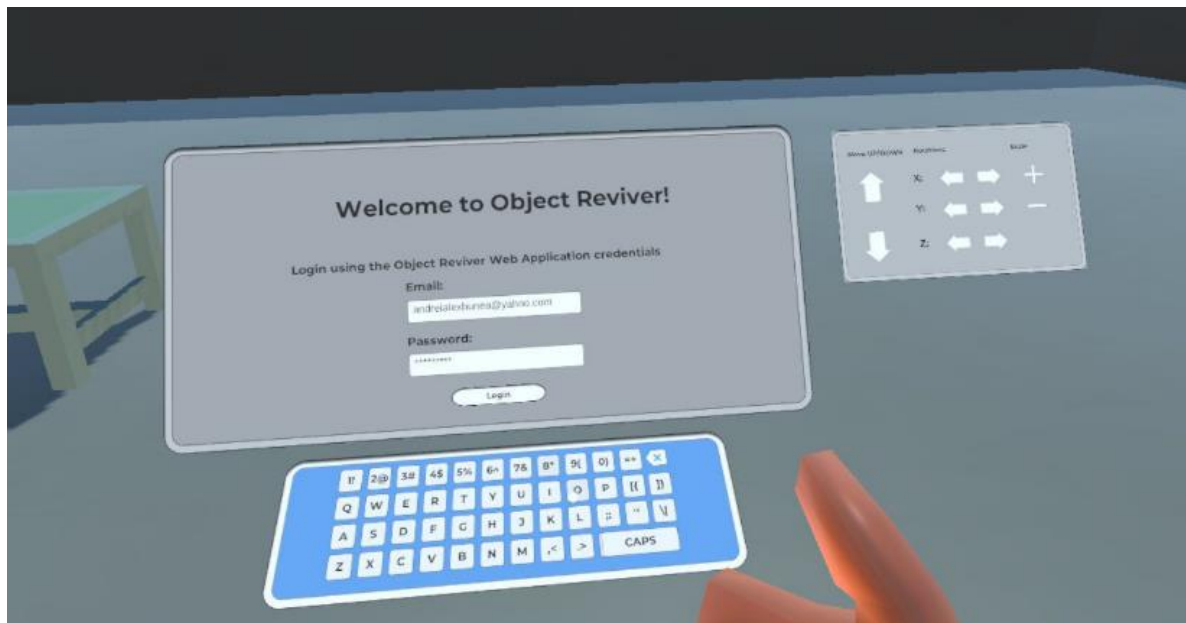


Figure 28 Login form completed using the keyboard

The VR application does not support all the functionalities provided by the web application. The visualisation tool is designed to only be accessible by already signed-up users, and therefore the sign-up function is not present in the scene.

The initial canvas encapsulates a welcoming message and asks the user to complete the login form, by typing inside the email and password inputs. The logic behind the keyboard and the login process is controlled inside a corresponding script that handles the controls of the keyboard, and the submission of the login credentials. When an input is selected, the pressed keys will write text inside the corresponding form input.

When the login button is pressed, the script sends a POST HTTP request to the “/login” endpoint of the server. It is the same endpoint used for the login process inside the web application.

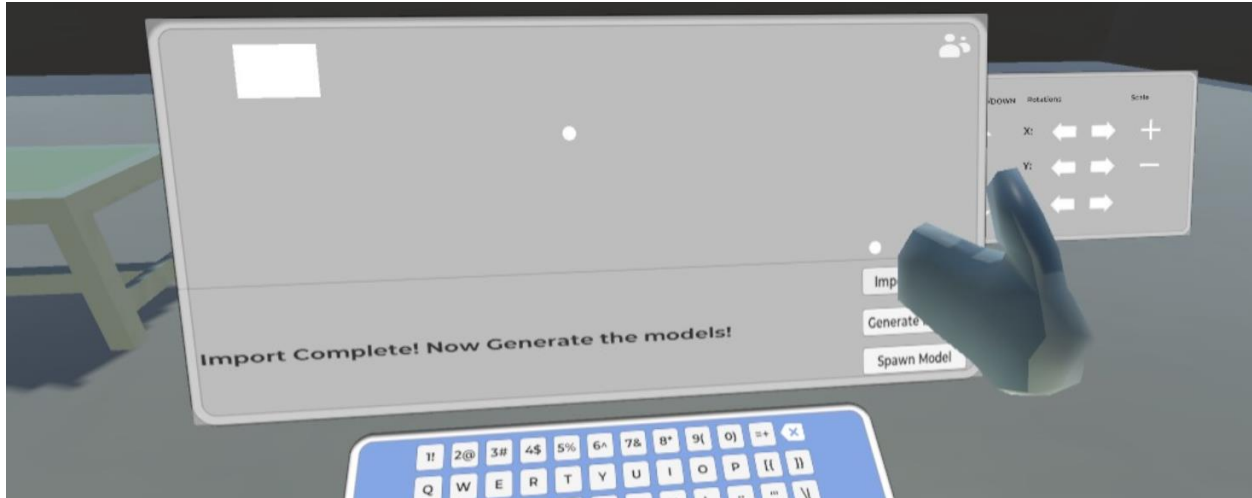


Figure 29 Importing the 3D models

After logging inside the application, the main panel changes the canvas, providing the import and spawn of 3D models from the user's portfolio, and the exploration of the users that are followed by the current account (Figure 29).

To ensure a fast flow of the imported data regarding the 3D models, the process is divided into three main steps. The first step is acquiring the data from the server. At this step, the “Import Models” button needs to be actioned, and a script associated with an empty game component responsible for this action calls the API in order to retrieve the data files of all the models reconstructed by the current user. The files are the preview image, the .obj file, the .mtl file and the texture files. These files are later saved inside the local memory of the device that runs the application. After this process is finished, a message is displayed to inform the user.

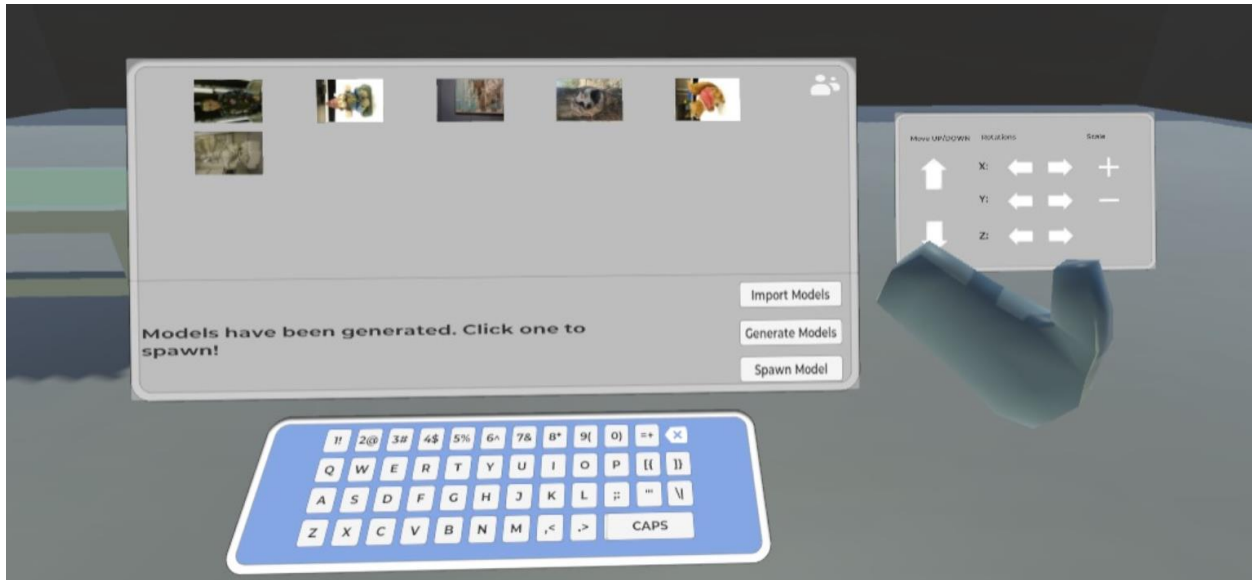


Figure 30 Portfolio canvas

After the importing process is finished, the user can trigger the “Generate Models” button to generate the portfolio. The current canvas changes, enlisting the 3D models accessible by the user, by displaying their preview image (Figure 30).

Now, the user can see all the available 3D models and can select one to be displayed. To do so, the user clicks on the preview image of the object he wants to spawn. After the selection of the model, the user needs to press the “Spawn Model” button to generate the object inside the scene.

The clicking of this button triggers a spawning function inside a script assigned to an empty game object responsible for this action. Having the data files inside the local memory, the script uses the “Runtime OBJ Importer” library [33] to generate the object. The script handles the spawning of the object, using the importer to load the object using the .obj and .mtl files, setting the initial position and scale and adding a mesh renderer component. In addition, the script ensures that the object is placed on the floor, casting rays to determine the object's bounds and the position of the floor of the scene.



Figure 31 Result of the spawning process

Figure 31 illustrates the result of the spawning process. The object is spawned behind the two panels, in the back of the scene. This object is reconstructed using images of an Egyptian sarcophagus. Reconstructed using high-quality images, the object strikes with fine details, perfectly reproduced by the structure from motion algorithm (Figure 32).



Figure 32 Details of the 3D model

The second panel, as mentioned before, facilitates the translation, rotation, and scale of the 3D object. Figure 33 presents an example where the panel buttons have been used to translate, rotate, and scale the reconstructed model of a statue.

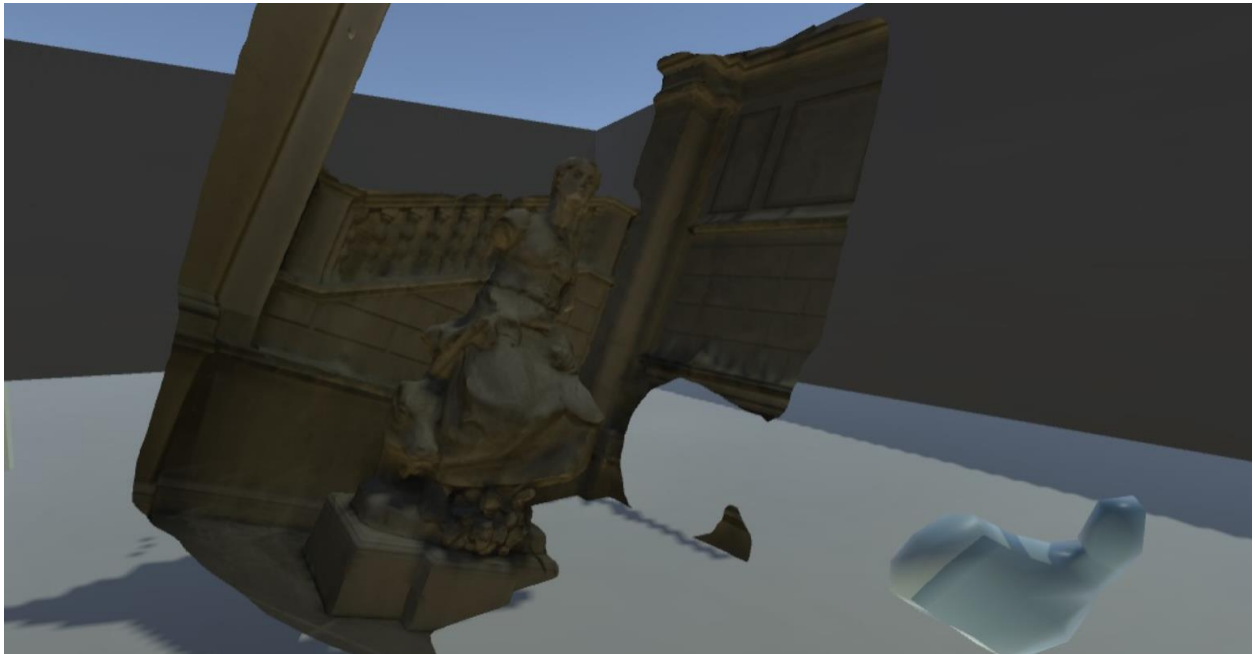


Figure 33 Rotated object using the second panel

Another functionality provided by the main panel is browsing for other portfolios. This process targets only the users that the current account follows, using the following mechanism provided by the web application. This canvas enlisting the following users is displayed when clicking the icon on the top-right corner of the portfolio panel. This icon can be observed in Figure 30.

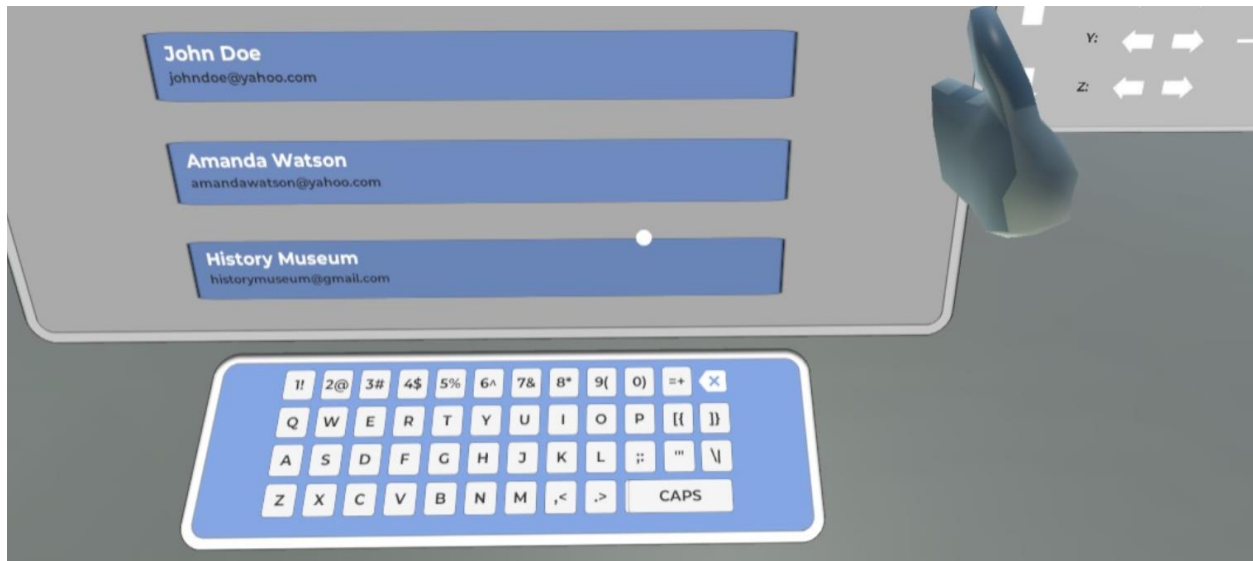


Figure 34 Following canvas

An example of the canvas mentioned before is presented in Figure 34. The followed users are enlisted dynamically inside the canvas. For each one of the followed users, their full name and email are displayed inside actionable buttons. When one of these buttons is clicked, the canvas panel redirects to the portfolio of the corresponding user. The current user can explore the portfolio of the followed user and spawn the objects from their portfolio.

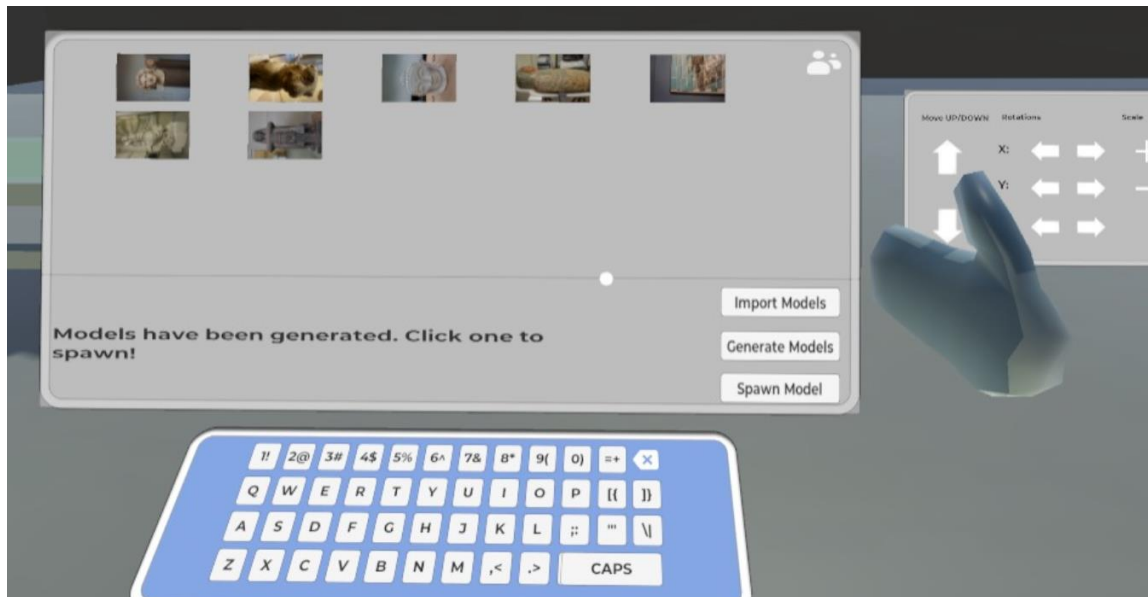


Figure 35 History Museum portfolio

In Figure 35, the logged-in user selected the “History Museum” profile from the followed users list, and the corresponding portfolio was displayed.

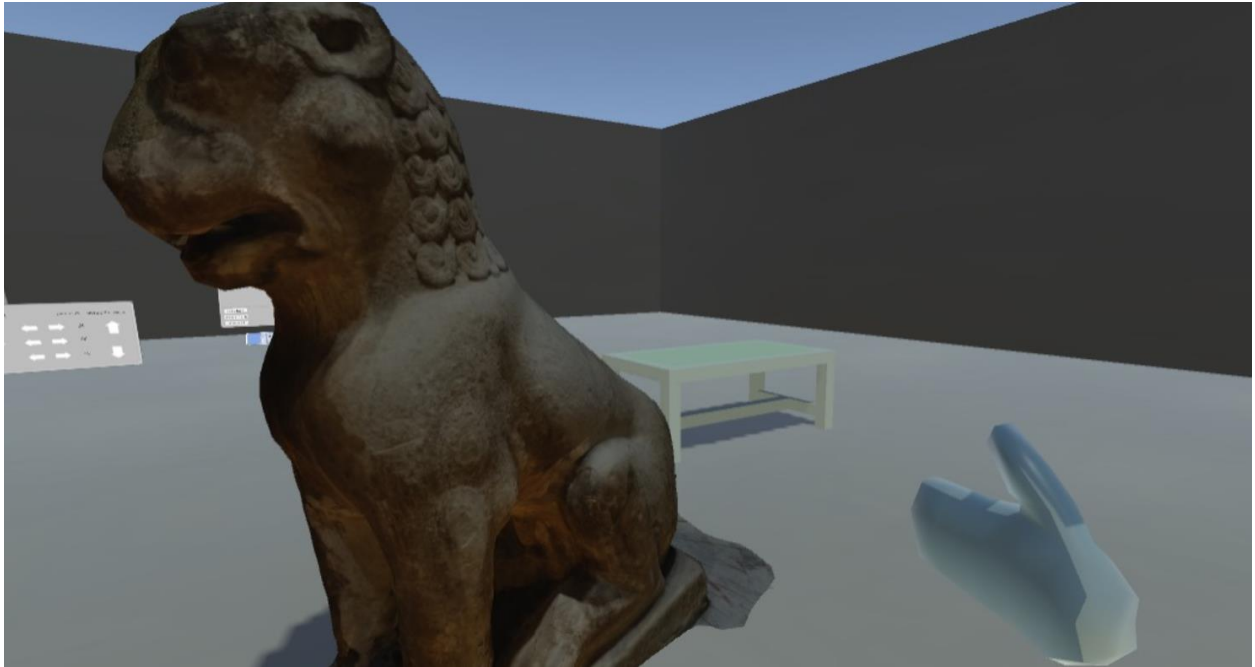


Figure 36 Amaravati Guardian Lion

The user explored the “History Museum” portfolio and spawned the second item from the listed objects. The spawned object is a reconstructed digital replica of the Amaravati Guardian Lion, and the result is presented in Figure 36.

4. Evaluation

4.1 Interpretation of the results

As mentioned before, many parameters of the 3D reconstruction pipeline guided by the Meshroom software influence the quality of the result. The choice of values for those critical parameters balances the quality of the mesh with the reconstruction time. As almost every step of the reconstruction process is processed in parallel using a multi-core and multi-threads approach, the available hardware resources dictate the performance of the application.

Examples of parameters that influence the response time are the unwrapping method, the simplification factor and the number of vertices used for the meshing operations. In addition, the downscale factor used in depth map estimation and the choice of feature extraction parameters can impact the speed of earlier steps without drastically affecting quality.

As the project is implemented on fixed parameters, a more important discussion is the results produced by the established structure. Even though the script is designed to run with the same arguments every time, the context of the taken images or video directly influences the quality of the reconstructed mesh.

Empirically, the 3D models resulting from this reconstruction process indicate some of the key factors that determine a good result. First, the light must be present in the captured scene, and all the key features of the object need to be illuminated. The reconstruction process has good results in identifying features if they are lightened, rather than shadowed. In addition, natural light is more effective than artificial light, identifying more feature points that determine the point-cloud of the mesh. When favourable light conditions as presented before were used, the resulting models were complete, and their texture presented details. However, when the image dataset depicting the scene featured shadowed parts, the 3D reconstructed model contained holes in its mesh, and the details of the object were too darkened to be recognised. This conclusion was drawn by capturing the same object outdoors and indoors, with different levels of luminosity, camera quality and shadowed areas. Multiple objects were used for this experiment, objects that present complex shapes and fine details that challenge the reconstruction process.

Other than light, the quality of the images has a great impact on the reconstruction process. To capture the fine details that induce the feeling of looking at a real object when looking at its virtual scan, a high-quality camera is needed. The quality of the camera is directly proportional to the number of features found by the reconstruction algorithm, as it facilitates a better resolution. An object with simple characteristics in terms of shape and textures can be successfully captured using a lower-quality camera, as it does not require the identification of fine details. However, if a complex object needs to be reconstructed, then its details need to be perfectly visible in clear, focused images. To capture a detail of an object, the respective feature needs to present a distinctive contour in the taken images, as the feature detection algorithms rely especially on gradients. In summary, if an object's features are blurred or obscured due to insufficient lighting, they cannot be accurately reconstructed.

To evaluate the performance of the project, multiple aspects need to be assessed. Both web and VR applications need to facilitate a user-friendly interface, with accessible functionalities and rapid response from the server to the user input. The experience should be engaging, and the VR visualisation tool needs to maintain the immersive sensation. In addition, the 3D reconstruction should create complete models that recreate the shape and textures of the captured scenes. A successful reconstruction of a scene should induce the feeling that the scanned item presented in the virtual environment is real. All these aspects need to be experimentally tested, and the feedback of the participants must be recorded.

4.2 Methodology

As the project is centred around user experience, the most effective way to evaluate its implementation is by conducting an experiment where participants interact with the web and VR applications. Feedback forms will be used to collect their thoughts on the project. This type of evaluation is crucial because certain metrics cannot be used to assess the 3D reconstruction process. The success of a reconstruction and its details can only be optimally evaluated by the human eye.

To evaluate the performances of the project, an experiment was conducted on 15 participants. The participants were selected based on the personal connections of the author. Every participant was informed about the experiment beforehand and was asked to prepare a set of images or a video of a scene that they wanted to reconstruct virtually. All the participants signed a consent form and accepted to participate without getting any financial compensation in return.

The project aims to offer an authentic experience to all users, irrespective of their level of expertise. Therefore, participants with varying levels of experience were selected, ranging from those who had never used a VR headset before to professionals with experience in 3D reconstruction. The participants were asked to fill out a Google form before the experience, in order to keep track of their level of expertise in using 3D reconstruction software and VR applications.

After the submission of the form, each user was asked to take a video or gather a set of images of a scene that they wanted to reconstruct. They were previously informed that the best results are obtained using highly illuminated and high-quality images.

Participants were asked to connect to the web application using the local IP address of the computer where the application was hosted and then access the video or images for reconstruction stored on their mobile phone or laptop. They were asked to log into the application and upload their data to create a virtual representation of the scene. The waiting time until the finalisation of the reconstruction process was recorded, and while waiting for the result, participants were encouraged to interact with all the features of the application (personal portfolio, user search, explore other profiles, preview models in a virtual scene using the THREE.js module).

When the reconstruction was finished, participants would investigate the results, interacting with the 3D model inside the model preview component using the orbit controls.

After testing all the features of the web application spending an average of 10 minutes (without counting the time spent waiting for the reconstruction), they were asked to put on the VR headset. The headset was provided by the author of the experiment, and the VR application of

the project was already running. The controls and mechanics of the avatar movement and item selection were explained to the participants during the process.

The participants were asked to move around the scene, evaluate the design of the application and the immersive feeling. They were then asked to use the virtual keyboard to log into the application and generate their portfolio. After the import of the models, participants would spawn the scene that they reconstructed, and explore the result. They were instructed to interact with a panel responsible for translating, rotating, and scaling the object. After exploring the results of their reconstructed object, the participants explored the works of other users, spawning any 3D model that they wanted.

At the end of the experience, every participant was asked to complete a post-experience Google form, offering feedback regarding the performances of the application, the immersive sensation that they felt and the correctness of the reconstruction process.

4.3 The results of the experiment

The first results that need to be assessed are the responses to the pre-experience questionnaire. As mentioned before, this questionnaire was filled out by the participants before the start of the experiment. It was used to extract conclusions on how previous experience can influence the use of the VR application and the 3D reconstruction module.

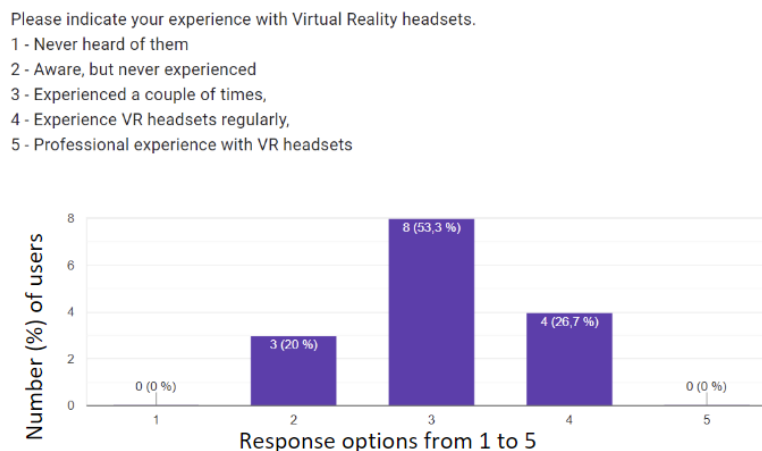


Figure 37 Answers to the question regarding previous experience with 3D reconstruction

After analysing the pre-experience questionnaire responses (Figure 37), it was observed that 7 out of 15 participants were aware of 3D reconstruction but had never experienced the process. However, when asked about their experience with VR systems, 8 participants responded that they had previously used VR headsets a couple of times, while 3 were aware of the technology but never experienced and the other 4 had experienced VR headsets regularly.

Aiming to gather detailed feedback from the participants, the post-experience form encapsulates 25 questions that target the interaction with the web application, reconstruction process and VR application.

The first two questions of the form ask about how easy was for the participants to interact with the web and VR applications, grading the experience from 1 to 5, where 1 represents an extremely difficult navigation and 5 represents an extremely easy navigation. All the participants scored the easiness of navigating the web application with 5, while for the VR application, 14 out of 15 participants answered the question with 5, while the other user answered with 4.

The participants were asked through the third question whether the data used for reconstruction was in the form of images or a video. Thirteen out of fifteen participants chose images, while the remaining two selected the video option. In addition, the fourth question requires to categorize the device used to take the images/video. Most of the users, 7 out of 15, used a professional camera, while 4 other users used a mobile phone with LiDAR technology and the other 4 used a modern mobile phone without the LiDAR technology. In addition, as the next two questions record, a third of the participants captured the scene for reconstruction indoors, while the other 10 users captured the objects outdoors.

How complex do you appreciate your scanned 3D models to be?

1 - Extremely Simple: The scanned 3D models are extremely basic and lack complexity, consisting of only a few simple shapes or features.

2 - Simple: The scanned 3D models are relatively straightforward and have minimal complexity, with uncomplicated shapes and limited detail.

3 - Moderate: The scanned 3D models have a moderate level of complexity, featuring a mix of simple and more intricate elements, providing some level of detail.

4 - Complex: The scanned 3D models are fairly complex, containing intricate details and a high level of complexity in their structure and design.

5 - Extremely Complex: The scanned 3D models are exceptionally complex, featuring a high level of intricate details, intricate shapes, and a considerable level of complexity in their overall composition.

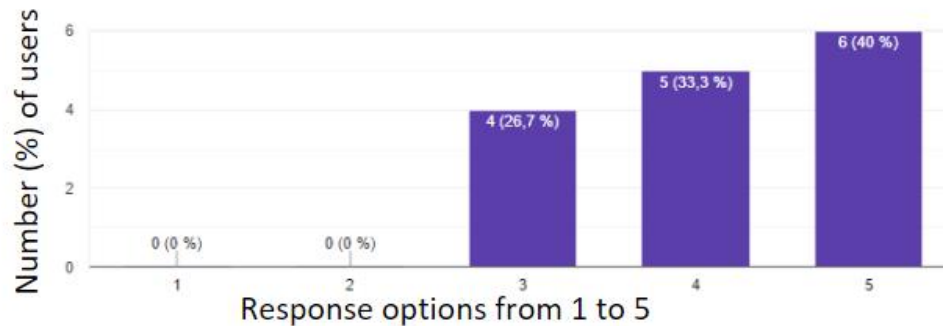


Figure 38 Responses to the complexity of the object question

Asked about the objects that they wanted to reconstruct using the uploaded data, 40% of the users assessed their object as extremely complex, having a high level of fine details and intricate shapes, while 33.3% checked the “Complex” option and the other 26.7% checked the “Moderate” box (Figure 38). Examples of objects assessed as extremely complex by the participants are the façade of a building, a mural, and a panoramic scan of a square from a village. On the other end, the “Moderate” objects are a Rubik’s cube, a cupboard, a coconut and a post box.

Please check the boxes corresponding to the textures of the objects that you scanned.

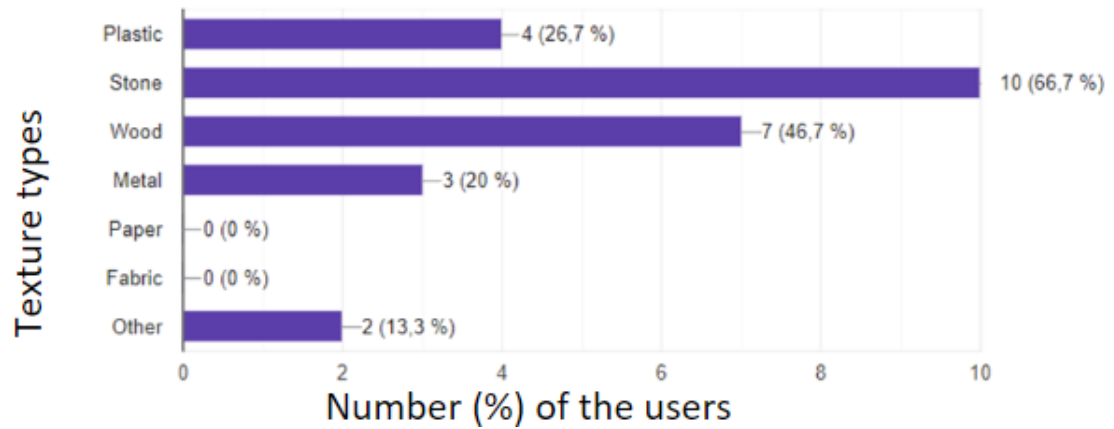


Figure 39 Materials of the scanned objects

Figure 39 illustrates the responses to the question that classifies the materials of the scanned objects, to detect any material that produces bad results.

Next, participants were asked to rate the smoothness of the web and VR applications on the devices used during the experiment. The feedback was favourable, with 100% of the users marking the web application as excellent and 93.3% of them marking the VR application as excellent as well.

How effective was the application in building the 3D model based on your input?

1 - Extremely Ineffective: The application failed to generate a usable 3D model from the input, and the results were essentially unusable or incorrect.

2 - Ineffective: The application struggled to create a satisfactory 3D model, producing inaccurate or incomplete results that required substantial manual correction.

3 - Average: The application was moderately effective in building a 3D model but required significant user intervention and post-processing to achieve an acceptable result.

4 - Effective: The application performed well in generating a 3D model, with relatively minor manual adjustments needed to refine the output.

5 - Highly Effective: The application excelled in building a 3D model based on your input, delivering accurate and high-quality results with minimal user intervention.

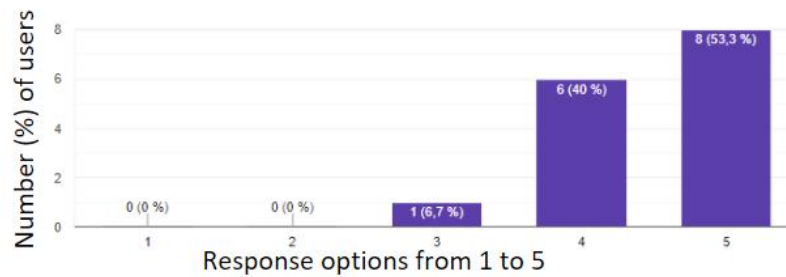


Figure 40 Feedback on the reconstruction result

Figure 40 presents the feedback on the reconstruction results, analysing its performance in realising accurate 3D models. The results were marked as effective or above by 14 out of 15 participants. While 40% of the participants considered the mesh as a high-quality result but needing intervention in separating the main object from other parts of the scene (Figure 41), most of the users (53.3%) considered their results as perfect (Figure 42).



Figure 41 Objects rated with 4/5 due to untrimmed sections



Figure 42 3D models rated with 5/5

As observed from the comments of the participants, most of the users that met the perfect conditions for photogrammetry, meaning high illumination and high-quality cameras, managed to produce perfect reconstructions. In addition, the context of the captured image is very important. In Figure 41, two 3D models are illustrated that produce high-quality results but do not manage to successfully separate the targeted object from the rest of the scene, due to the context of the object in the captured environment. As the main object is highly linked in the physical world with other parts of the scene that are not part of the mesh, the task of identifying the bounds of the targeted model is extremely complex.



Figure 43 Image from the uploaded dataset

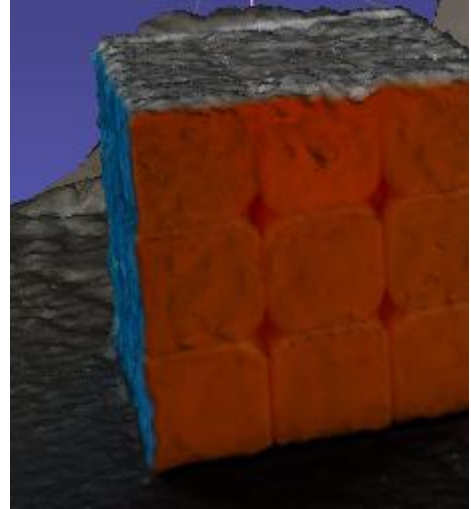


Figure 44 Reconstruction result

Last, the report needs to analyse the only reconstruction marked as average. This model is the Rubik's cube, and the device used to capture the images is a modern mobile phone. The dataset used to create the 3D model is a collection of images. The uploaded images present characteristics that need to be avoided in the process of reconstruction, such as blurred sections and bad illumination. Moreover, the cube is made of plastic which is very glossy, and the reflected light is visible on the surface of the cube. An example of an image from the uploaded dataset is presented in Figure 43, where the glossiness of the material is observed. Moreover, the direction of the casted light is clear, as the orange surface is shadowed. In conclusion, the lack of quality and focus of the mobile phone's camera, the lack of illumination and the texture are critical factors that lead to unappealing results.

While some of the least complex objects required between 1 and 3 minutes from the moment of uploading the images until the completion of the reconstruction process, some of the objects required longer times.

How long did you have to wait for your 3D model to be created from your input?

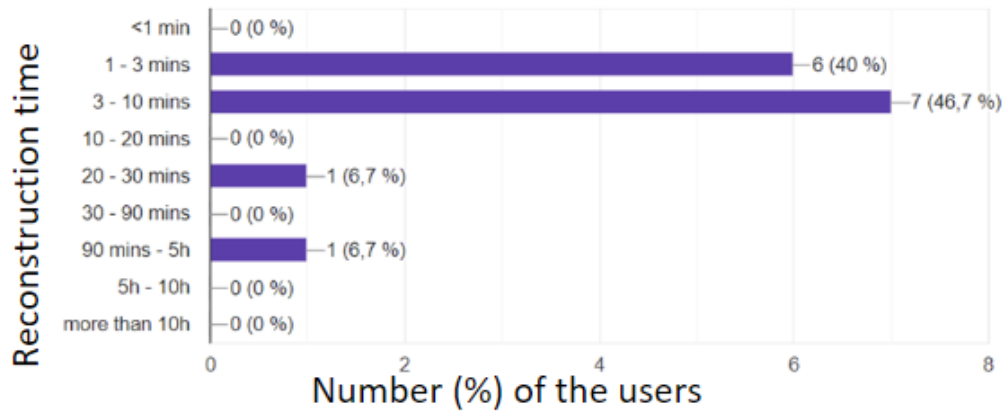


Figure 45 Waiting times for the reconstruction

Most of the complex models required 3 to 10 minutes for the reconstruction. However, the most complex two models required more time. The reconstruction of the panorama view of a village's square was realised in 22 minutes, and the reconstruction of the complex Rosetta Stone lasted 4 hours. These times are achieved by hosting the application on a personal computer. The performance of the reconstruction process is proportional to the hardware specifications of the server, as better processing units produce faster times.

In the end, the last batch of questions evaluated how engaging the whole experiment was, and how effective was the VR application in inducing an immersive sensation. The participants marked the web and VR applications as "Highly Appealing", and "Highly Engaging", and reported that the VR application provided a fully immersive virtual experience. They also noted that the 3D models felt realistic due to the fine details of the reconstruction, and they could explore the object in its natural size right in front of their eyes.

4.4 Use cases

As mentioned in the introduction chapter, the 3D reconstruction pipeline can bring significant contributions in a vast range of applications and research areas. Virtual representations of real objects can enhance both research and teaching.

One use case that is already implemented inside the project, is using the application as a tool to present virtual museum artifacts. The scenario is represented by the “History Museum” user, which has in their portfolio 7 artifacts captured from different sources. Administrating such an account, interesting objects from all over the world can be registered in a virtual form, capturing all their fine details that fascinate the public. This collection of virtual artifacts can be an important tool for archaeologists or historians who need to study historic art, in an immersive environment, without needing to travel across the world.

Moreover, this profile can be used as an educational tool. When teaching subjects such as history in class, teachers often use images to portray historic objects related to their course. Presenting images has been proven to successfully captivate the student’s attention, as visualising the information is more attractive than reading it in most cases.

Images are already useful in aiding teachers to present their lessons effectively. However, if students could observe a 3D replica of the scene depicted in the images while immersing themselves in virtual reality, it would be an even more effective tool for capturing their attention and stimulating their curiosity. Not only the students would be more fascinated by the presented objects, but exploring the 3D model will enhance their understanding of the given scene, as they would be encouraged to explore all the fine details.

Another use case that was recorded using the scanned models of the participants involved in the experiment, was capturing architectural elements. Some of the participants realised interesting reconstructions of statues, facades or towers using images or videos. Following the results, they were impressed with the opportunity of recording a 3D model of every interesting architectural element they find in their travels and were keen on using the virtual reconstructions to share their experience of observing the physical scene with other people.

5. Conclusion

The responses to the post-experience questionnaire project the satisfaction of the users regarding the design and functionality of the project. The feedback received underlines an engaging experience that manages to fulfil the creative needs of the participants.

Future work may include an upgrade in terms of performance to the reconstruction process. Balancing powerful hardware resources with the number of user requests, a more intensive unwrapping technique can be introduced, and more vertices can be considered for meshing. For the VR application, future work may involve the integration of the “Ubiq” networking scheme that allows multiple users in the same scene, while synchronising all the rendered elements.

The web application provides a straight-forward interaction with the reconstruction process, offering an effective tool for previewing the results of the structure from motion pipeline and exploring the work of other users. Accomplishing the initial goals, the web application makes 3D reconstruction accessible to both experienced and inexperienced users and serves as a platform that facilitates quick object sharing and exploration.

The reconstruction process is able to produce high-quality results for a vast range of items of different sizes and textures. Balancing the performance of the reconstruction with the processing time, the component responsible for 3D reconstruction creates qualitative models that satisfy the user’s needs.

The VR application provides a virtual visualisation tool that successfully induces an immersive sensation. Facilitating an authentic interaction with virtual representations of physical objects, this application has proved to be a powerful tool for multiple purposes, such as research or teaching. Following the experiment mentioned in the last chapter, the VR application was highly preached, as it offered the participants a unique experience that left them impressed with the ability of virtual environments and photogrammetry to replicate reality.

In conclusion, combining state-of-the-art techniques of implementing a 3D reconstruction process, a web application and a VR application, the project successfully introduces a truly authentic and engaging experience.

References

- [1] "Unity Asset Store," [Online]. Available: <https://assetstore.unity.com/>.
- [2] "Sketchfab," [Online]. Available: <https://sketchfab.com>.
- [3] S. Gosh, "History of photogrammetry," 1981.
- [4] T. Schenk, "Introduction to Photogrammetry," *The Ohio State University, Columbus*, 2005.
- [5] A. & A. F. M. & M. M. Jebur, "3D City Modelling by Photogrammetric Techniques," 2017.
- [6] D. V. J. K. R. W. Ruinian Jiang, "Close-range photogrammetry applications in bridge measurement: Literature review," *Measurement*, pp. 823-834, 2008.
- [7] E. M. B. J. S. & M. J. C. Mikhail, *Introduction to modern photogrammetry*, John Wiley & Sons, 2001.
- [8] "Archeological Institute of America," [Online]. Available: <https://www.archaeological.org/fieldwork/archaeological-photogrammetry/>.
- [9] J. B. N. G. M. H. J. R. M.J. Westoby, "'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications," *Geomorphology*, vol. 179, no. 2012, pp. 300-314.
- [10] A. a. S. G. Eltner, "Structure from motion photogrammetric technique," *Developments in Earth surface processes*, 2020.
- [11] A. A. W. H. S. M. A. S. S. A. K. A. S. a. B. A. Qureshi, "Evaluation of Photogrammetry Tools following Progress Detection of Rebar towards Sustainable Construction Processes," *Sustainability*.
- [12] P. P. L. Falkingham, "Free and Commercial Photogrammetry software review: 2020 [with minor updates in 2021]," [Online]. Available: <https://peterfalkingham.com/2020/07/10/free-and-commercial-photogrammetry-software-review-2020/>.
- [13] "Meshroom," AliceVision, [Online]. Available: <https://github.com/alicevision/Meshroom>.
- [14] C. G. S. C. L. G. P. C. F. M. B. D. L. G. a. L. Y. Griwodz, "AliceVision Meshroom: An open-source 3D reconstruction pipeline," in *Proceedings of the 12th ACM Multimedia Systems Conference*.
- [15] I. E. a. A. Kuqi'', "Analyzing parameters in 3D reconstruction photogrammetry in Meshroom, a case study," in *11th Mediterranean Conference on Embedded Computing (MECO)*, 2022.
- [16] L. a. E. E. Casas, "Virtual and Printed 3D Models for Teaching Crystal Symmetry and Point Groups," *Journal of Chemical Education*, 2015.

- [17] M. W. C. J. L. & Q. D. J. Smith, "Structure from motion photogrammetry in physical geography," *Progress in Physical Geography: Earth and Environment*.
- [18] J. C. C. P. S. P. L. O. J. a. R. J. Iglhaut, "Structure from Motion Photogrammetry in Forestry: a Review," *Current Forestry Reports*, 2019.
- [19] F. Remondino, "Heritage recording and 3D modeling with photogrammetry and 3D scanning," *Remote Sensing*, 2011.
- [20] G. B. P. G. T. F. M. J. a. S. A. Patrucco, "DIGITAL REPLICAS OF BRITISH MUSEUM ARTEFACTS," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2023.
- [21] R. P. D. a. W. G. Pausch, "Quantifying immersion in virtual reality," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*.
- [22] J. G. P. W. D. H. A. C. A. a. K. S. Jerald, "Developing virtual reality applications with Unity," *IEEE Virtual Reality*.
- [23] A. Bunea, "MSc Final Project Repository," [Online]. Available: <https://github.com/andreibunea99/MSc-Final-Project>.
- [24] J. Nielsen, "Response times: the three important limits," *Usability Engineering*.
- [25] D. Lowe, "Distinctive image features from scale-invariant keypoints," *nternational journal of computer vision*, 2004.
- [26] I. S. J. a. P.-P. Torres-Xirau, "Fast Approximate Nearest-Neighbor Field by Cascaded Spherical Hashing," in *Asian Conference on Computer Vision*.
- [27] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.
- [28] M. a. P. T. Jancosek, "Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces," in *International scholarly research notices*, 2014.
- [29] Y. a. K. V. Boykov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Energy Minimization Methods in Computer Vision and Pattern Recognition: Third International Workshop*.
- [30] P. a. A. E. Burt, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics (TOG)*, 1983.
- [31] C. P. J. a. K. R. Allene, "Seamless image-based texture atlases using multi-band blending," in *19th international conference on pattern recognition*, 2008.

- [32] I. V. E. L. -. UCL, "Ubiq," [Online]. Available: <https://vr.cs.ucl.ac.uk/ubiq/>.
- [33] Dummiesman, "Unity Asset Store - Runtime OBJ Importer," [Online]. Available: <https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547>.
- [34] Meta, [Online]. Available: <https://about.fb.com/news/2020/09/introducing-oculus-quest-2-the-next-generation-of-all-in-one-vr/>.
- [35] I. a. K. A. Enesi, "Evaluation of the 3D Reconstruction Performance of Objects in Meshroom: A Case Study".