CS436 Computer Vision Fundamentals Project Report

Jawad Saeed
Department of Computer Science
LUMS
25100094@lums.edu.pk

Ibrahim Farrukh
Department of Computer Science
LUMS
25100227@lums.edu.pk

1 Introduction

This project involved 3D reconstructions of a famous historical landmark and then deploying the resulting model in an Android App using Augmented Reality. The project was split across 4 different deliverables highlighted as follows:

1. **Task 0:** This task involved the basic setup of the Android application using the provided skeleton code for the **Flutter** application.
2. **Task 1:** This task involved the selection of a landmark to reconstruct from the **Heritage Recon** dataset. Following the selection, the dataset was preprocessed, and a small subset of images was passed through feature matching, camera pose estimation, and, lastly, Linear Triangulation.
3. **Task 2:** Building on the previous task this task involved building a **Structure for Motion** pipeline for 3D reconstruction. Steps involved **Feature Matching and Detection, Camera Pose Estimation, Sparse 3D Reconstruction, and Multi-View Stereo Techniques for Point Cloud Generation**.
4. **Task 3:** The final task involved the deployment of the reconstructed model in the Flutter application that was set up in Task 0.

2 Methodology

2.0.1 Task 0

The first step involved the installation of **Flutter** on the MacOS operating system. This was followed by the installation of **Android Studio Ladybug** for access to the **Android SDK**, which was necessary for the proper functioning of the application. Once the **flutter doctor** command verified installation, the device to be used was connected. For this project, we used a **Samsung Galaxy Note 8 (SM 950F)**. Due to compatibility issues, the app skeleton provided required modifications for version compatibility. This involved using a **Gradle** version that was compatible with **Java 21**.

2.0.2 Task 1

With this task, the first step was to select a model to use for reconstruction. In our case, we selected the **Pantheon** dataset within the Heritage Recon dataset for reconstruction. Once this was selected, the entire dataset was loaded in and preprocessed, which included resizing all the images and converting them to grayscale. Following this, we selected the **Scale Invariant Feature Transform (SIFT)** algorithm for feature detection. Additionally, we implemented a threshold mechanism to filter key

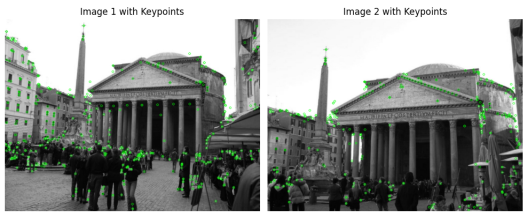

Figure 1: Extracted Features using SIFT algorithm in Task 1

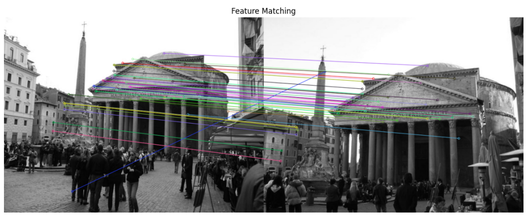

Figure 2: Matched Features using the Brute Force Matcher in Task 1

points based on a certain confidence threshold to ensure good features were extracted. The extracted features were visualized as shown in Figure 1.

Feature matching was performed using the Brute-Force Matcher with L2 norm for descriptor comparison and cross-checking to ensure robust matches. Keypoint descriptors from the two images were matched, and the matches were sorted based on their distances, with lower distances indicating better matches. The top 50 matches were visualized by connecting corresponding key points in the two images, as seen in Figure 2, providing a clear representation of the feature correspondences.

Moving on, the next step involved extracting the camera and image parameters from the provided files in the dataset. Since these files were in the **bin** format, they were first converted into **txt** files, and then custom parsing functions were used to load the data. The fundamental matrix was computed from the matched keypoints using the **cv2.findFundamentalMat** function with the `RANSAC` method to filter outliers. Subsequently, the essential matrix was derived, enabling the recovery of the relative rotation and translation between the two images using the **cv2.recoverPose** function. The keypoints from the matched features were then triangulated into 3D points in a global coordinate system. This

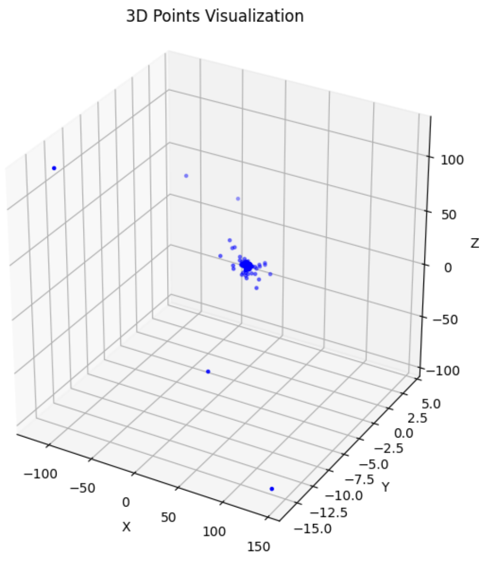

Figure 3: 3D Point Cloud for Task 1

process utilized the projection matrices computed from the camera intrinsics and the recovered pose. These 3D points were visualized as shown in Figure 3.

2.0.3 Task 2

The implementation was designed for the reconstruction of 3D models using datasets for both the **Pantheon** and **Brandenburg Gate**. The preprocessing of these datasets involved manual cleaning to improve the quality of the 3D reconstruction. Cleaning steps included the removal of irrelevant images, such as selfies, images with a high density of humans, or images not focusing on the primary structures (Pantheon or Brandenburg Gate). Additionally, the dataset was adjusted to ensure continuity between consecutive images, minimizing large differences in perspective or coverage, which enhanced the feature matching and reconstruction process.

Initially, images were loaded from the specified dataset directories and resized to a uniform resolution of 800×600 pixels. The images were then converted to grayscale to reduce computational overhead while retaining essential features. Feature extraction was performed using the **SIFT** algorithm, configured to detect up to 5000 features per image. Consecutive image pairs were processed to extract keypoints and descriptors. The **BFMatcher** algorithm, with Lowe's ratio test (set to a relaxed threshold of 0.7), was employed to identify high-quality feature matches. To further improve accuracy, only the top 100 matches for each pair were retained.

Camera parameters and poses were extracted from precomputed metadata files (`cameras.txt` and `images.txt`). The camera intrinsic matrix and the global pose (rotation and translation) for each image were reconstructed. The origin was initialized using the first image in the dataset, and subsequent images' poses were calculated relative to the origin by leveraging the relative rotation and translation matrices derived from the essential matrix. Outlier matches and insufficient feature points were filtered during this process to ensure robustness.

For 3D triangulation, matched keypoints from each image pair were used to generate 3D points in a global coordinate system. Projection matrices were computed for each image pair using their intrinsic and global pose parameters. The **cv2.triangulatePoints** function was used to compute 3D points from the matched keypoints, and the results were aggregated to form a complete point cloud of the structure.

To generate a final 3D mesh, the aggregated point cloud was processed using the **Open3D** library. Normals were estimated for the point cloud, and the Poisson reconstruction algorithm was applied to create a dense mesh. Procedural coloring was then applied to the mesh, where vertex colors were determined based on the normalized height (*z-coordinate*) of the vertices, producing a gradient effect. The final point cloud and 3D mesh were saved as `.ply` files for visualization and further use.

This pipeline was iteratively refined, and its performance was tested on both datasets. Adjustments to the dataset and preprocessing methods significantly enhanced the quality of feature matching and 3D reconstruction, demonstrating the importance of careful dataset preparation and methodical feature extraction.

2.0.4 Task 3

With the model reconstructed, the next step was to deploy it. Since the main goal was deploying the model on the application, we faced difficulties in this part in the sense that the libraries required the **glb** file format. Conversion of the ply files into the **glb** files resulted in errors due to lossy conversion. With this in mind, we decided to use an open-source reconstruction of the **Pantheon** from **Fab** alongside the provided Pantheon ply file in the dataset folder.

For deployment in an Augmented Reality scenario, we made use of the **ar_flutter_plugin**. Finding the appropriate library was a challenge as our version of the app went through two different iterations using different libraries, such as the **arcore_flutter_plugin** and **sceneviewer**, which did not work. Another challenge was getting the app to work with the new libraries since compatibility and namespace issues arose. This was due to these libraries not being maintained and outdated according to the **Gradle** standards, leading to namespace and **Kotlin** errors. This was resolved by modifying the library code itself in the cache to comply with the Gradle requirements.

With all these issues resolved, the coding part was simple enough, using example code from the creators of the repository itself. This code required modifications in the sense that our model was being loaded locally and not from the web. This required using the **fileSystemAppFolderGLB** node type for the **ARNode** class in the library. The pressing issue in this part came from issues of storing the model and configuring its path. Due to library configurations, the model required to be loaded into the app's storage itself, which is different from the local storage in our project directory. To solve this issue, the file was directly placed in the **com.example.app** folder on the device.

This allowed us to visualize the model in the camera. Here, we faced hardware issues due to the phone's limitations. The provided ply file in the dataset wasn't being visualized since the phone did not have enough computing power to render the model. This was the reason why the downloaded **agrippa_pantheon.glb** worked. The recording attached shows this model visualized.

3 Results and Findings

3.1 Task 0

The flutter application was successfully installed on the phone using the provided skeleton. Figure 4 shows the initial app loading page after installation.

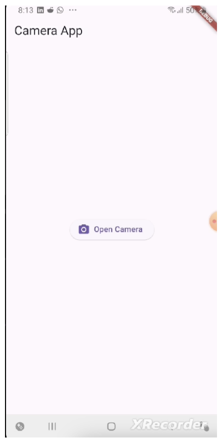

Figure 4: Application after successful installation on phone

3.2 Task 1

The 3D point cloud visualization, shown in Figure 3, highlights a concentration of features in a compact cluster. The results demonstrated that while the pipeline could identify overlapping features between the two images, the 3D reconstruction lacked structural depth and global consistency. The feature points were concentrated around regions of overlap in the two images, leading to a compact cluster without meaningful spatial distribution representative of the full scene. This was expected, as the relative pose estimation and triangulation process rely on a broader range of views to capture the depth and structure of a scene effectively. Furthermore, the limited number of images constrained the ability to recover intricate architectural details or establish accurate global poses. The orientation and scale of the reconstructed 3D points could not be meaningfully inferred, further emphasizing the limitations of this approach. Overall, the task provided valuable insights into the strengths and limitations of the 3D reconstruction pipeline when working with minimal visual data. It demonstrated that the feature detection and matching algorithm performed as intended but highlighted the necessity of using multiple overlapping images, as implemented in subsequent parts, to achieve more comprehensive and accurate reconstructions.

3.3 Task 2

Brandenburg Gate Reconstruction

The reconstruction of the **Brandenburg Gate** dataset, as visualized in the generated `.ply` file (see Figure 5), demonstrates a partially successful 3D model of the iconic landmark. While the overall geometry of the gate, including its general shape, was captured, the reconstruction exhibits certain inaccuracies. Notably, the two displaced pillars on the right side of the visualization highlight potential errors arising from incorrect feature matches. These mismatches are likely caused by interference from people that are still present in the images or by inconsistent image angles, leading to challenges in triangulation and pose estimation.

Furthermore, intricate architectural details, such as the horses atop the gate, were not reconstructed accurately due to insufficient feature matches in those regions. Although the texture quality of the mesh is satisfactory from certain perspectives, it is not perfect. Rotating the model reveals noticeable deformations, particularly on the backside of the gate, which further emphasize the limitations of the dataset and the reconstruction pipeline.

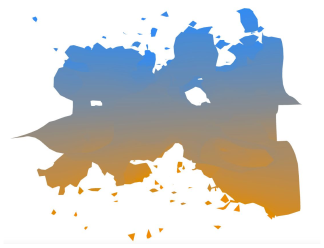

Figure 5: 3D reconstruction of the Brandenburg Gate showing partial reconstruction and displaced pillars.

Pantheon Reconstruction

The reconstruction of the **Pantheon** point cloud captures some architectural elements, such as the triangular pediment and portions of the pillars, especially when viewed from specific angles (see Figure 6). However, the generated model suffers from significant distortions and incompleteness. These issues can primarily be attributed to the lack of continuity and similarity between consecutive images in the dataset. The overall quality of the reconstruction was further impacted by the high number of images containing humans in the foreground of the Pantheon, which distorted the global pose estimation. Consequently, the triangulated 3D points lacked precision, leading to a model that does not accurately represent the architectural details of the Pantheon.

Note on both results

The code has been verified through github repos, articles, and the teaching staff as well. However, the only reason that I can think of for it to not construct a proper 3d reconstruction or even match it is irregularity in the dataset which can be further refined for a proper 3d reconstruction to ensure continuity among sequential images.

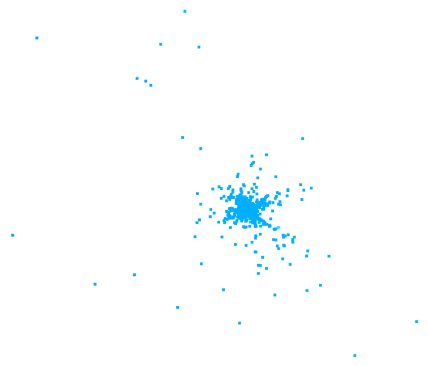

Figure 6: 3D point cloud reconstruction of the Pantheon showing incomplete and distorted results.

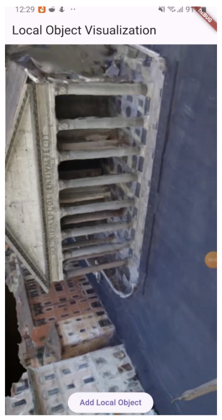

Figure 7: Deployed Pantheon Model

3.4 Task 3

Following the successful model reconstruction and app installation, we proceeded to load the model in an **Augmented Reality** scenario. For this, we made use of the aforementioned **agrippa_pantheon.glb** file downloaded from **Fab**. The code for deploying the model was retrieved from the following **Github** repository.

With the code used the structure of the app changed in the sense that now is loads directly into the phone's camera and a button is pressed to load the local model from the device storage to display. The displayed model is shown in Figure 7.

The video recording for the visualization was choppy due to the hardware limitations of the selected phone, as it could not handle large GLB files for deployment. The deployment of the downloaded GB file wasn't smooth as well, considering that, at times, the device ran out of GPU memory, leading to the app crashing.

4 Conclusion

This project successfully explored the end-to-end process of 3D reconstruction of a historical landmark, followed by its deployment in an Augmented Reality (AR) application. Through a sequence of structured tasks, key technical concepts were applied and integrated to achieve the final goal.

The project began with setting up the development environment using Flutter and ensuring compatibility across various tools and devices. This foundational step was critical in enabling the smooth deployment of subsequent deliverables. The use of SIFT for feature extraction and the Brute-Force Matcher for feature matching allowed robust detection of correspondences across images in the Heritage Recon dataset. Linear triangulation enabled the initial 3D reconstruction, while the subsequent implementation of a Structure from Motion pipeline allowed for a more refined and dense 3D point cloud generation.

Despite challenges such as dataset compatibility issues, library version mismatches, and computational limitations, we tackled them using alternative open-source models and library modifications to overcome significant roadblocks during the AR integration stage. The deployment of the reconstructed model in the Android application marked a successful culmination of the project, with the AR visualization effectively showcasing the Pantheon model.

In conclusion, this project highlighted the importance of combining theoretical knowledge with practical implementation to solve real-world problems. Integrating computer vision techniques for 3D reconstruction and AR deployment underscored the potential of such technologies in applications like cultural preservation and interactive education. While the project achieved its primary objectives, future work could focus on improving the robustness of the reconstruction pipeline, optimizing AR rendering for lower-powered devices, and expanding the application to support other landmarks.