Landmark 3D Reconstruction and AR Visualization

Muhammad Ahmad Ashraf
26100169 Lahore University of Management Sciences

Sarim Malik
26100129 Lahore University of Management Sciences

ABSTRACT

Reconstructing 3D landscapes and structures from visual data is a cornerstone of modern computer vision, with applications spanning architectural design, simulation environments, urban planning, and virtual reconstructions. This project aims to reconstruct the Brandenburg Gate using the Heritage Recon dataset, leveraging a Structure-from-Motion (SfM) pipeline to generate a detailed 3D model. Additionally, the project explores the integration of augmented reality (AR) for real-world visualization of the model through an Android application, showcasing the potential of combining 3D reconstruction with AR for enhanced spatial interaction. By integrating key computational techniques, the project demonstrates the adaptability of such frameworks for reconstructing complex environments, emphasizing their open-ended potential across diverse domains.

Figure 1: The Brandenburg Gate; target landmark for SfM and AR Visualization

1 INTRODUCTION

Reconstructing 3D landscapes and architectural structures from 2D images is a critical task in computer vision, offering transformative potential for applications like architectural design, urban planning, digital simulations, and immersive virtual environments. This project undertakes the digital reconstruction of the Brandenburg Gate using the Heritage Recon dataset, employing a Structure-from-Motion (SfM) pipeline for precise modeling. The resulting 3D representation is further enhanced by developing an Android application that leverages augmented reality (AR) to visualize the reconstructed model within real-world spaces.

The methodology involves a rigorous Structure-from-Motion (SfM) pipeline that synthesizes key computer vision techniques to produce a detailed 3D model. Tools from *OpenCV* such as ORB and BFMatcher are used for feature detection and matching respectively, while match outlier rejection is implemented via RANSAC to ensure robustness. Subsequent stages include estimating the essential matrix, determining camera poses, validating the chirality condition via triangulation, and performing global pose estimation. Finally, the pipeline generates a textured mesh and visualizes the 3D point cloud using *Open3D*.

The integration of AR through an Android application bridges the gap between digital reconstruction and physical interaction, allowing users to visualize and interact with the model in their environment.

This project highlights the efficacy of SfM pipelines in resource constrained environments and presents a flexible framework applicable to a wide range of domains, from architecture to virtual environment creation. By combining cutting-edge reconstruction techniques with AR technologies, this work offers a flexible framework applicable across various domains.

2 METHODOLOGY

2.1 3D Reconstruction using Structure from Motion Algorithm (SfM)

In this section, we describe the methodology adopted to perform 3D reconstruction of a landmark using the Structure from Motion (SfM) algorithm, applied to multiple images from the Heritage-Recon dataset. The SfM pipeline primarily involves feature detection, matching, camera pose estimation, and ultimately the reconstruction of the 3D structure from the 2D images.

2.1.1 Dataset Preparation and Image Preprocessing. The process begins with the collection of relevant images from the dataset. For this work, images are preprocessed to ensure that they are in a suitable format for feature extraction and matching. First, the images are resized to a standard resolution to improve computational efficiency. Subsequently, each image is converted to grayscale, simplifying the processing by reducing the color information to intensity values. This preprocessing is critical for reducing noise and ensuring consistency across the images.

Additionally, the function *get_top_images_by_translation* is used to select the top N images based on their translation values, which indicate camera movement. The function reads a file containing the translation components (TX, TY, TZ) for each image (provided in the given dataset), calculates the Euclidean norm of each translation vector, and sorts the images by this magnitude. The images with the lowest translation values w.r.t the global origin are selected for their spatial diversity, ensuring a comprehensive set of images for the next steps. In our case, we are using the top 10 images that were retrieved using this function.

2.1.2 Feature Detection/Matching and Camera Pose Estimation. Feature detection is a fundamental step in the pipeline, where we identify distinctive points or features in each image that can be matched across multiple views. In this approach, we utilize the ORB

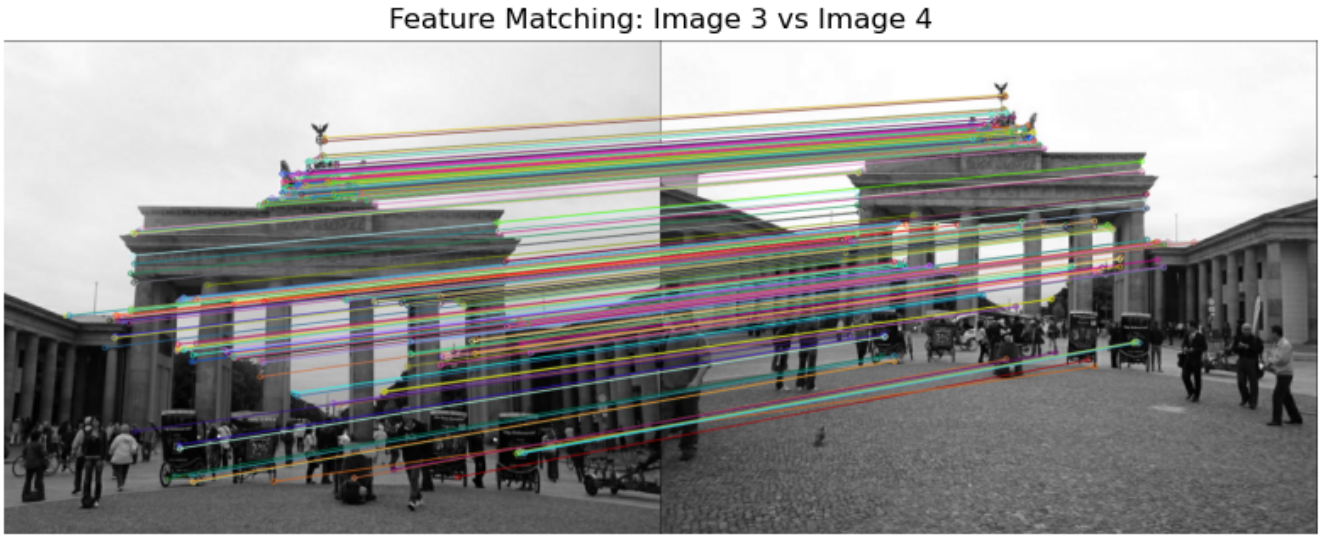

Figure 2: Feature Matching using BF Matcher between image pairs

(Oriented FAST and Rotated BRIEF) feature detector, which is efficient and robust for the task at hand. ORB is applied to each image in the dataset to extract keypoints and descriptors. The descriptors are then matched using the Brute Force Matcher (BFMatcher), which performs a K-NN (K-nearest neighbors) search to identify corresponding features between pairs of images.

To improve the quality of the matches and ensure the reliability of the reconstruction, only the best matches are selected using Lowe's ratio test. This filtering step helps eliminate false matches and minimizes the risk of introducing errors in the 3D reconstruction process.

Once the feature points have been matched between consecutive image pairs, the next step is to estimate the relative poses of the cameras, i.e., their positions and orientations with respect to each other. This is done using the fundamental matrix, which is computed from the corresponding points using the RANSAC (Random Sample Consensus) algorithm to ensure robustness against outliers. The camera intrinsic parameters, such as focal length and principal point, are derived from a separate calibration dataset, allowing us to obtain the essential matrix. The relative pose between two cameras is then recovered by decomposing the essential matrix into rotation and translation components.

The poses for each image pair are computed sequentially, with each new pose being derived relative to the previously computed poses. This is achieved through the iterative process of recovering poses and applying them to the scene geometry.

After computing the relative camera poses, the next step is to integrate these poses into a global coordinate system. This process involves transforming the relative poses into a common reference frame, starting from an arbitrary reference pose (typically the first image) and sequentially applying the relative transformations. This step is crucial for aligning the entire scene and ensuring that the 3D reconstruction is coherent and correctly aligned.

2.1.3 3D Reconstruction and Visualization. With the camera poses computed, the next step in the pipeline is the reconstruction of the 3D structure of the scene. This is achieved by triangulating the matched feature points in 3D space using the camera projections and the estimated camera poses. The triangulation process involves finding the intersection of the rays projected from the cameras through the matched points. However, before proceeding with the triangulation, cheirality checks are performed to ensure that the reconstructed points lie in front of our cameras. The result is a sparse 3D point cloud representing the structure of the scene, with valid and geometrically consistent points.

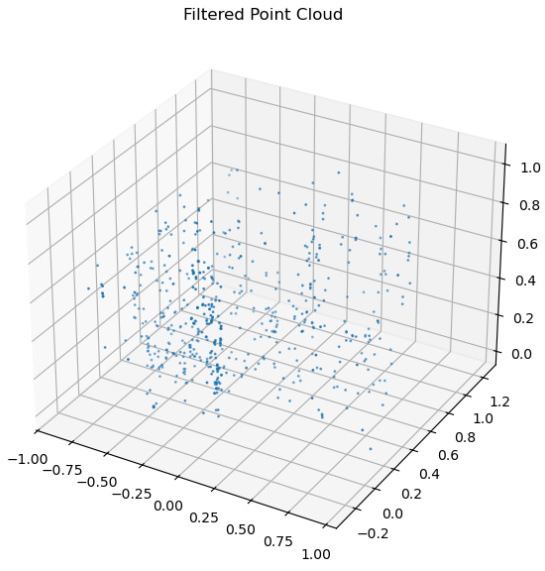

Figure 3: 3-D point cloud of the reconstructed points

2.2 Augmented Reality (AR) Visualization

2.2.1 Application Setup. To facilitate the development of an Android application for augmented reality visualization, we utilized the Flutter framework in conjunction with the *arcore_flutter_plugin (version 0.1.0)*. ARCore, Google's augmented reality SDK, provides cross-platform APIs designed for creating immersive AR experiences on Android devices. The primary goal of the application was to detect planes in the camera feed and use these detected planes as surfaces to visualize 3D models.

The setup involved configuring camera permissions and importing required 3D models as local Flutter assets. Specifically, we used the *brandenburg_simple.glb* model, a minimal open-source 3D model chosen for its low resource demands and compatibility with mobile environments.

During development, we encountered limitations with the *arcore_flutter_plugin*, which had not been updated in over 22 months. These issues included incompatibility with the latest Kotlin versions and mandatory Google Play Services installation requests, hindering its functionality. To address these challenges, we manually modified the plugin's library files to resolve compatibility issues and ensure seamless integration within the application, which may hinder the applications ability to build on other developmental environments.

After development, the application can be exported to a device (Google Pixel 4A in our case) using Flutter to be used for visualizing 3D models.

2.2.2 Application Workflow. The application operates by initializing its AR environment and adjusting to the surrounding environment for a brief period. During this initialization phase, ARCore leverages built-in sensors such as an RGB camera and an Inertial Measurement Unit (IMU) — a combination of an accelerometer, magnetometer, and gyroscope. These sensors are used for calibration and spatial awareness to detect flat surfaces or planes within the environment presented to the camera.

Once a plane is detected, the user can interact with it by tapping on the screen. This tap triggers the application to calculate the precise translation of the 3D object relative to the world coordinate frame. The object is then anchored at the specified location on the detected plane. To manage outdated functionality in the *arcore_flutter_plugin* library, the application manually loads the binary data of the 3D model file and writes it to a temporary directory. This allows the application to treat the model as a local file, enabling seamless integration and visualization.

Users can create multiple 3D visualizations on various detected planes, limited only by the device's capability to identify new planes. Once an object is visualized, users can navigate around the environment to view the object from different perspectives, providing a complete 360-degree visualization experience. The scale of the 3D model is deliberately adjusted to fit within the environmental bounds, ensuring resource efficiency and compatibility with mobile devices.

The summary of the workflow can be described in the pseudocode as follows:

Algorithm 1 Application Workflow

1: **Initialization:** Start AR session and calibrate using IMU sensors.
2: **Plane Detection:** Search for planes in the camera feed and render them.
3: **User Interaction:** Wait for tap on detected plane, capture tap coordinates.
4: **3D Model Loading:** Load and save 3D model from local assets to temporary directory.
5: **3D Model Placement:** Place and scale the 3D model at tapped location.
6: **Visualization:** Allow 360-degree view of the object.
7: **Repeat:** Repeat for any additional detected planes.

3 RESULTS

3.1 3D Reconstruction using Structure from Motion (SfM)

The Structure-from-Motion (SfM) pipeline developed successfully generated a 3D point cloud representation of the scene. This point cloud captures the geometric structure and spatial distribution of features within the environment, providing a sparse yet informative model of the scene. Each point in the 3D cloud corresponds to a feature detected in the images and triangulated using the camera poses. Notably, the reconstructed model remotely resembles the actual structure of the Gate, with clear representations of the six pillars and the top dome visible in the point cloud's architecture. The model offers valuable insight into the spatial relationships between these elements, laying the groundwork for further analysis and refinement.

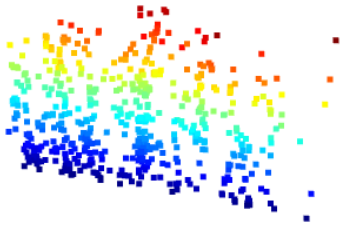

Figure 4: 3D point cloud visualized with *o3d.visualization.draw_geometries*

3.2 AR Visualization

On the other end, the AR Visualization Application successfully detected various planes within a multi-plane environment and enabled the visualization of a simplified Brandenburg Gate (Figure 5). The

application performed well in multiple scenarios, offering a smooth visualization experience. However, due to the older hardware of the Google Pixel 4A, there were instances where the calibration effect would diminish over time. This occasionally caused the 3D models to become misplaced relative to their intended locations.

Once the application successfully detected the environment, users could easily place the 3D model on any flat surface of their choice. They were also able to navigate around the environment to achieve a 360-degree view of the model. The model dynamically cast shadows based on the light sources in the environment, enhancing the immersive experience as if the model were present in real life (Figure 6).

Figure 5: AR visualization of the simplified Brandenburg Gate (Front-view)

Figure 6: AR visualization of the simplified Brandenburg Gate (Side-view)

4 CONCLUSION AND FUTURE CONSIDERATIONS

Despite promising results, this project had some limitations:

- The SfM pipeline utilized a small subset of 10 images, many of which contained noise such as people in the foreground. This impacted feature detection and resulted in a less detailed 3D reconstruction. *Future work should employ a larger, cleaner dataset to enhance feature extraction and model resolution.*
- Noise and limited viewpoints restricted robust feature matching and model coverage, leading to inaccuracies and a front-only perspective of the Gate. *Addressing these issues could involve improved feature matching techniques and capturing images from multiple angles for a more complete 3D reconstruction.*
- One key optimization step that was omitted in this pipeline (due to high computation costs) was bundle adjustment, which helps minimize reprojection errors and refine the overall model. *In future iterations, integrating bundle adjustment could lead to better optimization of the camera parameters and 3D points.*
- The *arcore_flutter_plugin* is outdated and lacks support for precise anchor setup without compromising scalability. Switching to a more up-to-date library in the future could improve performance and enable explicit calibration for more robust results.
- The application struggles to load heavy models, such as detailed versions of the Brandenburg Gate, due to performance constraints. Future iterations could explore optimization algorithms to compress models for smoother and more visually appealing experiences.
- The application faced significant dependency conflicts, resulting in longer development times. A more streamlined approach to managing libraries could reduce this complexity in future development.

In conclusion, the SfM pipeline successfully generated a 3D reconstruction of the scene, demonstrating the potential of this technique to create detailed models from images. Similarly, the AR-based visualization application was able to detect planes in the environment and allow users to interact with a 3D model of the Brandenburg Gate in real time. Despite some challenges, such as the outdated *arcore_flutter_plugin*, which limited the scalability and accuracy of anchor placement, the results provided valuable insights into the scene's structure. The application also faced performance limitations with complex models due to the device's capabilities, but optimization algorithms could improve future visualizations. Furthermore, the Structure-from-Motion results, combined with AR visualization, highlighted the importance of dataset quality and feature matching for more accurate 3D reconstructions.

Looking ahead, the application could evolve into an integrated system, where users can take photos of real-world objects in real time to generate 3D models that could then be visualized using AR. This merger of SfM and AR would offer a powerful tool for interactive, real-time 3D modeling and visualization.