CIS581 Final Project

Feature Extraction and Key Point Matching for 3D Reconstruction

Project members:
Yixiao Ling (liing@seas.upenn.edu)
Yufa Zhou (yufazhou@seas.upenn.edu)
Keqi Wu (keqiwu@seas.upenn.edu)

Content

Abstract

In computer vision, graphics, and robotics, reconstructing 3D scenes from 2D photographs has emerged as a prominent study and application area. To create complex 3D representations, the technique combines data from multiple sources, such as pictures, point clouds, or sensor data. Accuracy and efficiency have been significantly increased by recent advances in computer vision, photogrammetry, and machine learning. In our project, we focused on the application of 3D reconstruction in heritage recovery. Also, we performed camera pose estimation, which is vital in autonomous vehicles, helping to understand the vehicle's relative position relative to the road and surrounding objects. We implemented the SFM (Structure from Motion) Pipeline and conducted experiments by selecting various feature extractors and matchers, combining different methods and measuring the performance of the model compared to the SIFT+KNN Baseline. Our intended method: DISK with LoFTR, performed well on the training data as we obtained a large amount of 3D points and low mean reprojection error. This indicates a favorable balance between feature matching quality and quantity.

1. Introduction

Goals and objectives:
In this project, we reconstructed and visualized 3D point clouds and camera poses of 10-20 photos of two objects (Fisher Fine Arts Library and Taj Mahal) taken from different perspectives. We also localized a new query image of Taj Mahal using the reconstructed 3D map. Innovative methods such as SuperPoint, SuperGlue, DISK and LoFTR are used for feature extraction and key point matching. The performance of ensemble methods are listed and compared as well.

Dataset:

Our dataset has 11 images of Fisher Fine Arts Library and 20 images of Taj Mahal photographed at different perspectives. Samples of image set are displayed as follows:

2. Method and Related Work

2.0 Main Technique: Structure from Motion

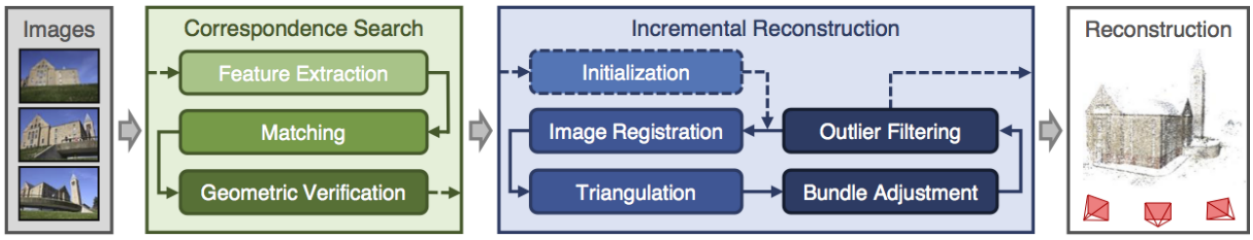

Fig 0. General pipeline of SfM from COLMAP

In the first part, we detected key features within each image that can be accurately tracked across multiple images (shown in Fig1). Then, we established good correspondences between features in different images, filtering out incorrect matches by checking for geometric consistency among the matches.

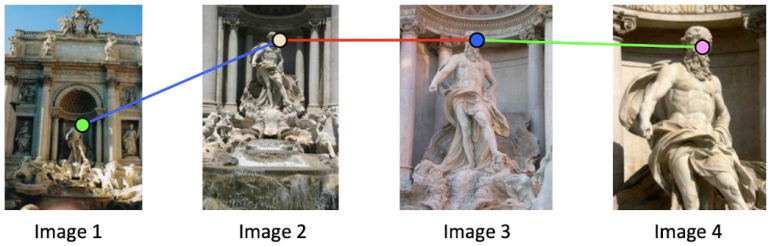

Fig 1. Similar feature across multiple images

In the second part, we firstly chose pairs of images with a high number of reliable feature matches detected. From the matched features, the essential matrix is calculated. The relative pose between the cameras for the two images is then extracted from the essential matrix. Also, when combining camera intrinsics parameters (assumed to be known), we calculated the absolute camera poses. Furthermore, with the above matched feature points, we conducted

triangulation to generate 3D points, which involves projecting rays from the camera centers through the feature points and finding their intersection in 3D space.

We used different combinations of feature extraction & matching methods to find the most effective workflow for 3D reconstruction. We combined traditional methods (SIFT + KNN) with more state-of-the-art methods involving neural networks, such as LoFTR, SuperGlue and LightGlue. The performance between these ensemble methods are listed and compared as well.

2.1 Image retrieval: NetVLAD

In order to handle the computational intensity of reconstructing 3D scenes from 2D images, we selected a subset of relevant and high-quality images using NetVLAD. To extract compact and discriminative representations from images, NetVLAD uses a trainable global image descriptor aggregation technique. NetVLAD is more suited for jobs requiring robustness to changes in viewpoint, lighting conditions, and scale because it aggregates local descriptors into a global representation, in contrast to standard approaches that generate fixed-length descriptors for images.

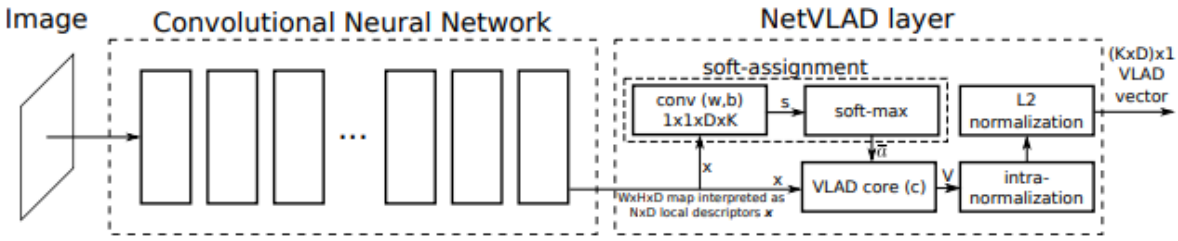

Fig 2. NetVLAD architecture

2.2 Screen some Image Pairs

Initially, we used all possible unique image pairs generated from the scene set. Then, we set the threshold to be 100, which is the minimum number of matches that we expect each image pair to have. If it is less, we discard that pair.

2.3 Image splitting

Each image was divided into four sections, each generating its own set of keypoints, followed by the execution of matchings across all pair combinations (4x4 = 16 pairs) with a batched process.

2.4 Feature Detection

2.4.1 SIFT

We used SIFT as the traditional method to detect and describe local features in images. In this method, firstly, regions that have different intensities at different scales are highlighted and potential key points are extracted. Then, descriptors for each key point are created from the gradients of image intensities around the key point. This traditional approach is invariant to scale and rotation but may struggle with large viewpoint changes and repetitive textures. This can be solved by DISK. It uses neural networks to dense and pixel-wise correspondences feature matching.

2.4.2 SuperPoint

The two primary phases of this deep learning architecture's operation are keypoint detection and description. SuperPoint locates interest points or keypoints in an image during the keypoint detection stage by forecasting the locations of distinguishing image elements like corners, edges, or blobs. These image positions, known as keypoints, remain constant when the image is rotated, scaled, or its illumination is altered. At the description stage, SuperPoint then creates descriptors for every keypoint that is found. Strong keypoint matching between various images is made possible by these descriptors, which encode details about the local image patch surrounding the keypoint. SuperPoint uses a convolutional neural network architecture to effectively complete the tasks of keypoint description and detection. Annotated data is used to train the network to identify trustworthy keypoints and produce discriminative descriptors.

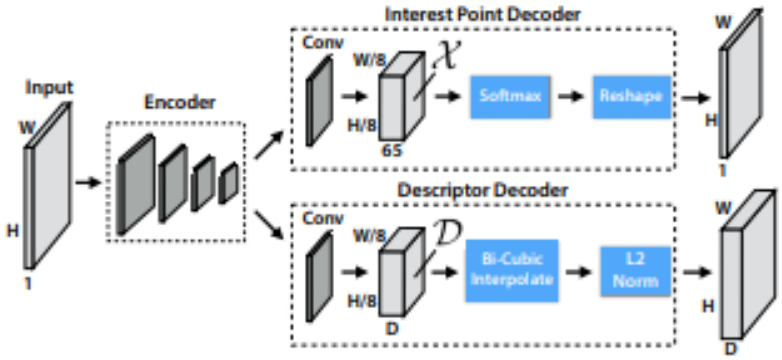

Fig 3. SuperPoint Descriptor

2.4.3 R2D2

Fundamentally, R2D2 consists of two primary parts: a descriptor network and a detector. Finding distinct and key points within an image is the detector's job. These salient features function as pivot points for further examination. Concurrently, the descriptor network produces comprehensive feature descriptors linked to these focal points, encompassing crucial details about the surrounding visual context. R2D2's main objective is to generate local features with outstanding robustness and repeatability on a variety of image datasets. This robustness makes sure that even in the face of changes in lighting, viewpoint, or other image transformations, the identified keypoints and related descriptors stay consistent and dependable.

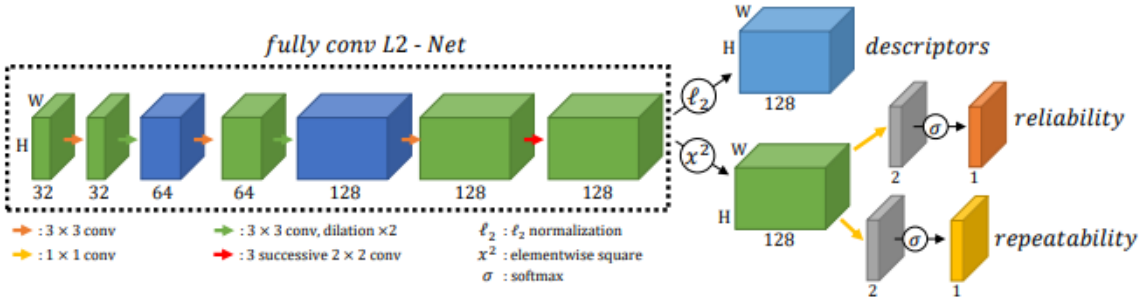

Fig 4. R2D2 Architecture

2.4.4 DISK

DISK presents a novel approach to maximize these local feature descriptors' performance using policy gradient methods. DISK uses a policy gradient-based optimization strategy, which has advantages in learning representations that are more discriminative and efficient for vision tasks than traditional methods that rely on supervised learning or handcrafted features. The incorporation of a differentiable sampling mechanism by DISK is essential to its success. During the training phase, this mechanism makes it easier to choose informative positive and negative samples. Using policy gradient updates, DISK iteratively modifies the descriptor network's parameters in an effort to improve the learned local features' discriminative ability.

2.5 Feature Matching

2.5.1 Rotation Detection

We noticed some scenes in the datasets contain rotated images. Since many popular learning-based matching methods can not handle this rotational case effectively, we rotated one of the query images several times[0, π/2, π, 3π/2] as shown in fig 5. Then we used the sparse method (SIFT and KNN matching) to find matches between the rotated query image and the target image, respectively. This helps to mitigate the drastic reduction in the number of matching points caused by image rotations. The matches for each rotation are stored, the rotation with the largest number of matches is chosen as the best rotation and is used for 3D reconstruction(shown in fig 6). In this particular example, we chose a rotation of 180 degrees and the corresponding filtered good matches are shown in fig 7. However, after the rotation_image function is applied, there are color adjustments occurring although we tried nearest-neighbor interpolation instead of bilinear interpolation.

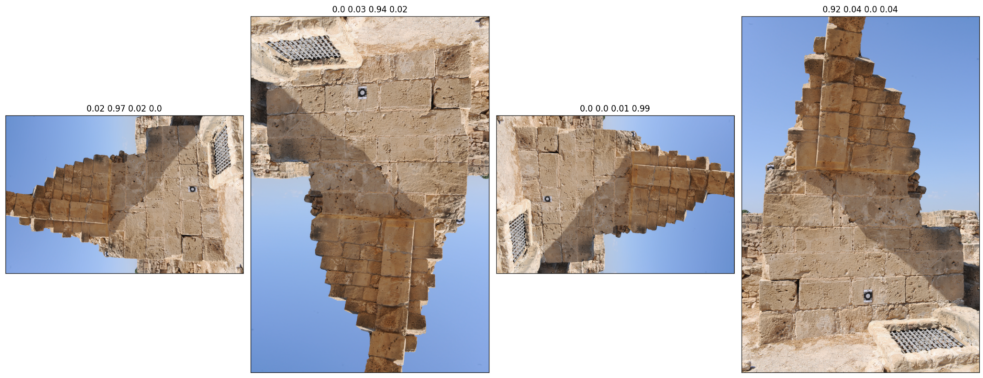

Fig 5. Rotation Sample Image

```
Rotation: 0 degrees, Matches: 1555
Rotation: 90 degrees, Matches: 1111
Rotation: 180 degrees, Matches: 1561
Rotation: 270 degrees, Matches: 1062
Best rotation is 180 degrees with 1561 matches.
```

Fig 6. Rotation Matches

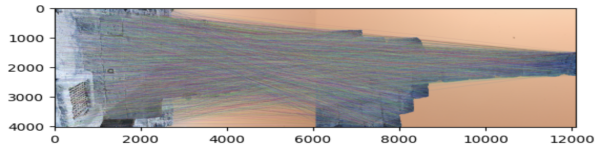

Fig 7. Rotation Match Visualization

2.5.2 Overlap Detection

We used the first round matching results to obtain the overlap region and then performed the second round of matching within them. One thing to notice is that before performing the second round matching, we resized the smaller region in one image and aligned it with the larger region according to the area ratio. We find a sparse matcher is capable of balancing efficiency and effectiveness.

2.5.3 Ensemble of individual methods: KNN match, LoFTR, SuperGlue and LightGlue

We explored the combination of sparse methods (KNN match, SuperGlue, LightGlue) and one dense method (LoFTR).

2.5.3 (1) KNN match

With the descriptor found via SIFT, the k closest matches for each descriptor can be found via k-Nearest Neighbors (kNN) Matching. Also, to improve the quality of feature matches, ratio test with threshold of 0.8 is performed to discard inappropriate matches. Finally, with the good matches found, geometric transformations (like scaling, rotation, and translation) between the matched points in different images are computed.

2.5.3 (2) SuperGlue

There are two main parts to this algorithm: a learned matching module and a descriptor. The descriptor is used to extract local features from images, usually keypoints and the descriptors that go with them. In the meantime, the learned matching module estimates the similarity between pairs of descriptors and determines whether they correspond to the same real-world point by using deep learning techniques to predict the likelihood of correspondences. SuperGlue

stands out for its capacity to manage confusing or difficult matching situations by taking into account not just the individual descriptors but also their connections and the environment in which they occur. Contextual knowledge improves matching accuracy and strengthens its resistance to different viewpoint shifts, various transformations, and difficult real-world scenarios.

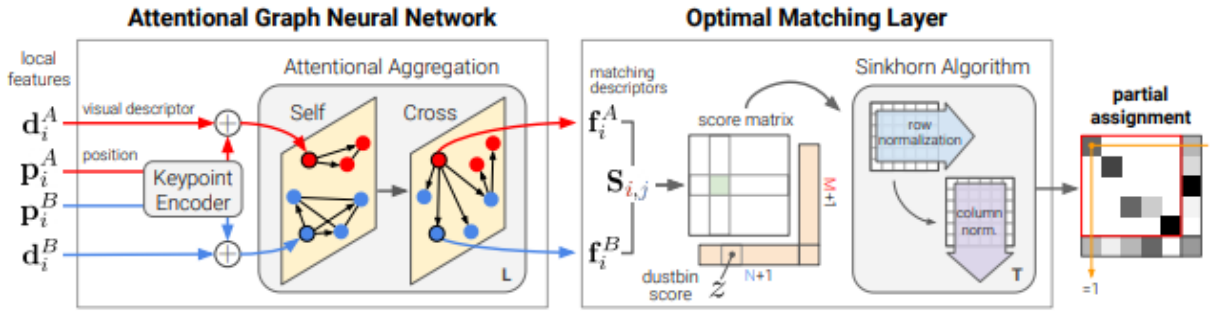

Fig 8. SuperGlue Architecture

2.5.3 (3) LightGlue

Based on factors like visual overlap, appearance changes, or discriminative information, LightGlue adapts to the level of difficulty associated with each image pair. For pairs that are intuitively easy to match, the inference proceeds much more quickly than it does for pairs that are difficult to match; this behavior is reminiscent of how humans interpret visual information. The model can then examine these correspondences and determine whether more computation is needed by predicting a set of correspondences following each computational block. In order to focus on the covisible area, LightGlue also eliminates points that are not matchable early on.

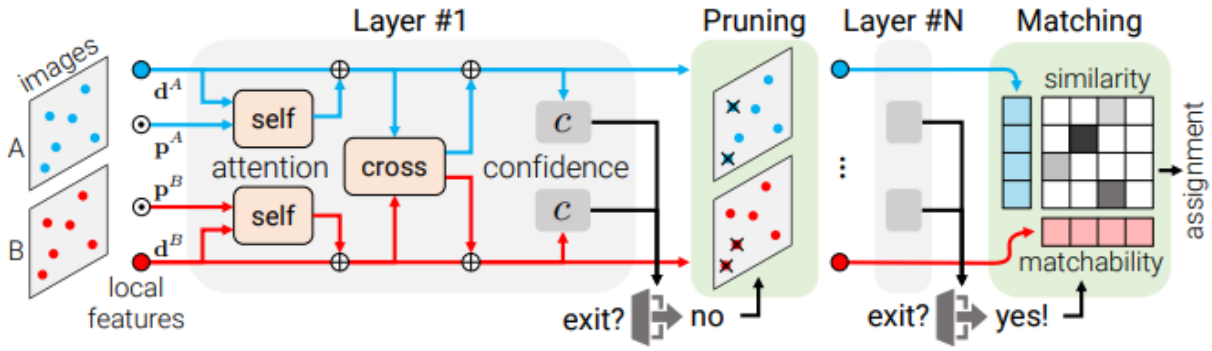

Fig 9. LightGlue Architecture

2.5.3 (4) LoFTR

LoFTR passes the extracted features through the transformer, where the self-attention mechanism is used to allow the model to weigh and integrate information from different parts of the image, providing a global view of the image and structure of scene. Therefore, LoFTR can extract high-quality semi-dense matches even in indistinctive regions with low-textures, motion blur, or repetitive patterns.

Robustness: LoFTR is designed to be robust against large changes in scale, rotation, and viewpoint. This makes it particularly useful for matching features in images taken from different angles or distances.

Deep Learning-Based: As a deep learning-based method, LoFTR can potentially provide more accurate and meaningful matches by learning feature representations that are more invariant to image transformations.

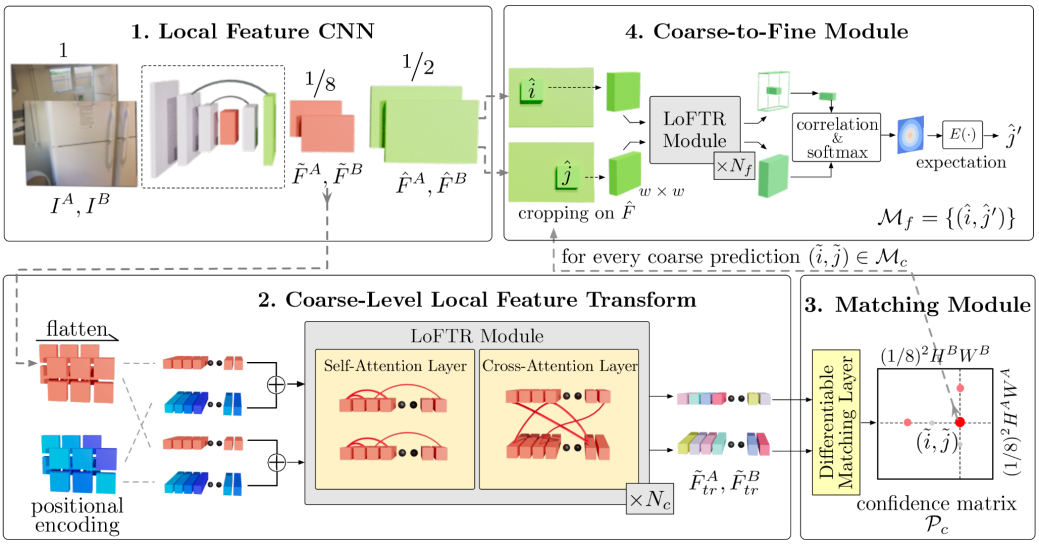

Fig 10. Architecture for LoFTR

2.5.4 Camera Calibration and Pose Estimation: essential matrix

We used the essential matrix to estimate the relative pose (rotational and translational matrix) between two camera views. Since this project includes translational camera movement instead of purely rotational movement, essential matrix is more appropriate than homography matrix.
N.T. Intrinsic camera calibration information is assumed to be known. We set focal length as (1000,1000), coordinates of the principal point as (960,548).

2.6 Matched feature points triangulation

3D Geometry Reconstruction: COLMAP

With the extracted and matched features, COLMAP estimates the 3D positions of these points and the camera parameters (position, orientation) for each image, contributing to a sparse 3D model of the scene. With this sparse model, COLMAP performs a dense reconstruction using MVS algorithms. In this process, more details could be filled in and thus a denser and more complete 3D model can be created. Furthermore, COLMAP refines the model through bundle adjustment, minimizing errors and improving the accuracy of the reconstruction.

N.T. In this project, we used shared camera parameters since we noticed that most scenes have been taken with the same camera and therefore decided to force COLMAP to use the same camera for all images in a scene if all images have the same shape.

3. Experiment, Result and Benchmarking Analysis

3.1 Main Experiments and Results

3.1.1 3D reconstruction of Fisher Fine Arts Library.

We first tried 11 images of Fisher Fine Arts Library photographed at different perspectives.

We visualized which key points were triangulated into the 3D model in two images as example.

The reconstructed 3D model is displayed as follows:

Fig 11. Sample Images of Fisher Library, Key Points and 3D Visualization

3.1.2 More accurate 3D reconstruction of Taj Mahal with ensemble methods and query image localization

Then, we input more images — 20 images of Taj Mahal and the subset of image set is as follows:

Fig 12. Sample Images of Taj Mahal

In this section, we tried various combinations of feature extraction & matching methods. The ensemble methods can be divided into two types: sparse and dense.

Figure 13 demonstrates the result of using sparse feature extraction & matching technique and those key points being triangulated are displayed correspondingly.

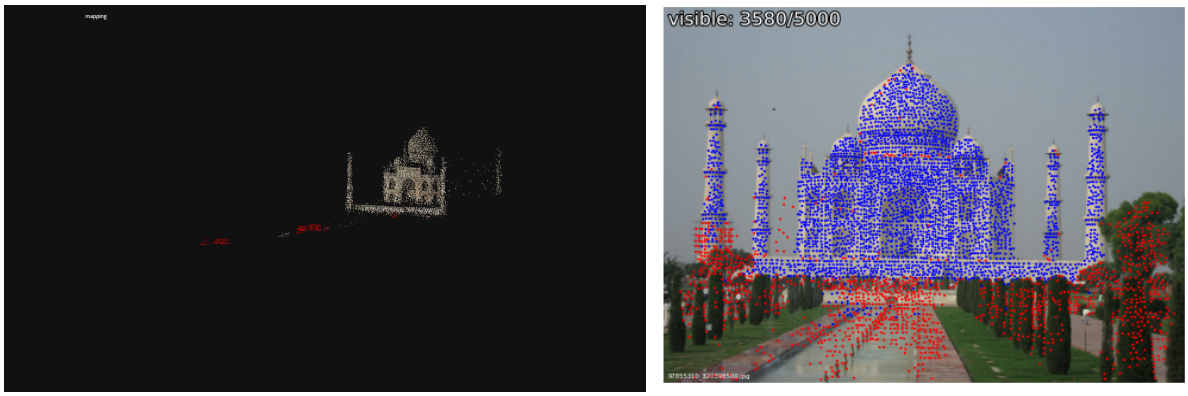

Fig 13. Sparse Construction

Figure 14 demonstrates the result of using dense feature extraction & matching technique and those key points being triangulated are displayed correspondingly.

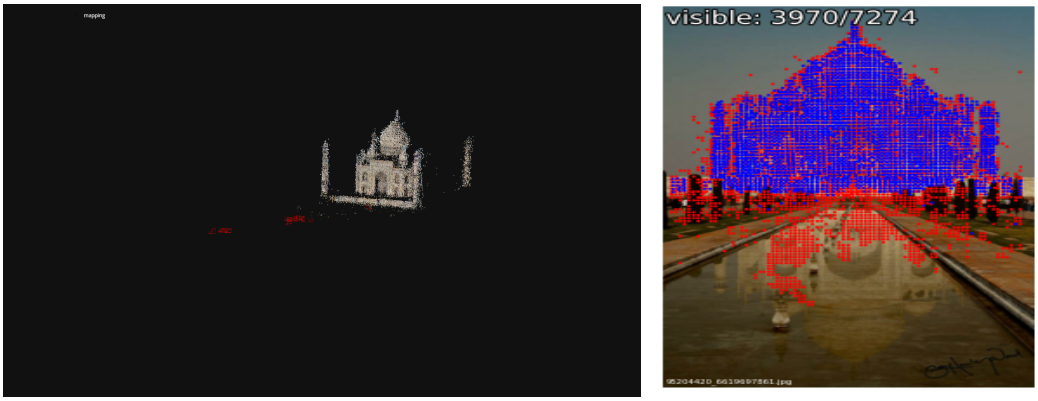

Fig 14. Dense Construction

With the above reconstructed 3D map, we localized a new query image of Taj Mahal. We repeated the procedure again, extracting features for the query and matching them exhaustively with all mapping images that were successfully reconstructed. We visualized the correspondences between the query images with a mapping image as an example shown in figure 15.

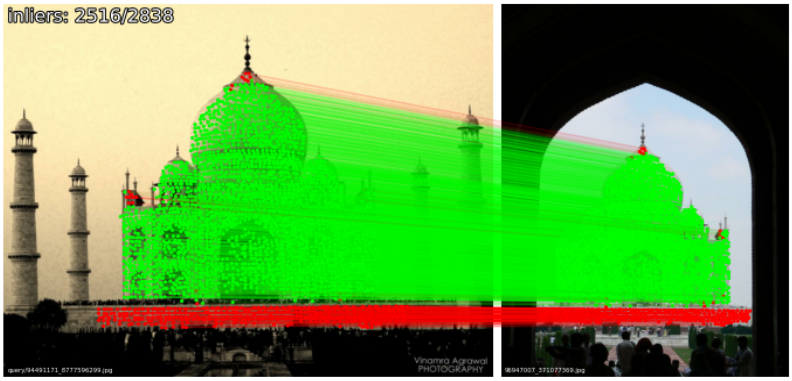

Fig 15. Query Image matching

Last but not least, we visualized the estimated camera pose in the 3D map as well.

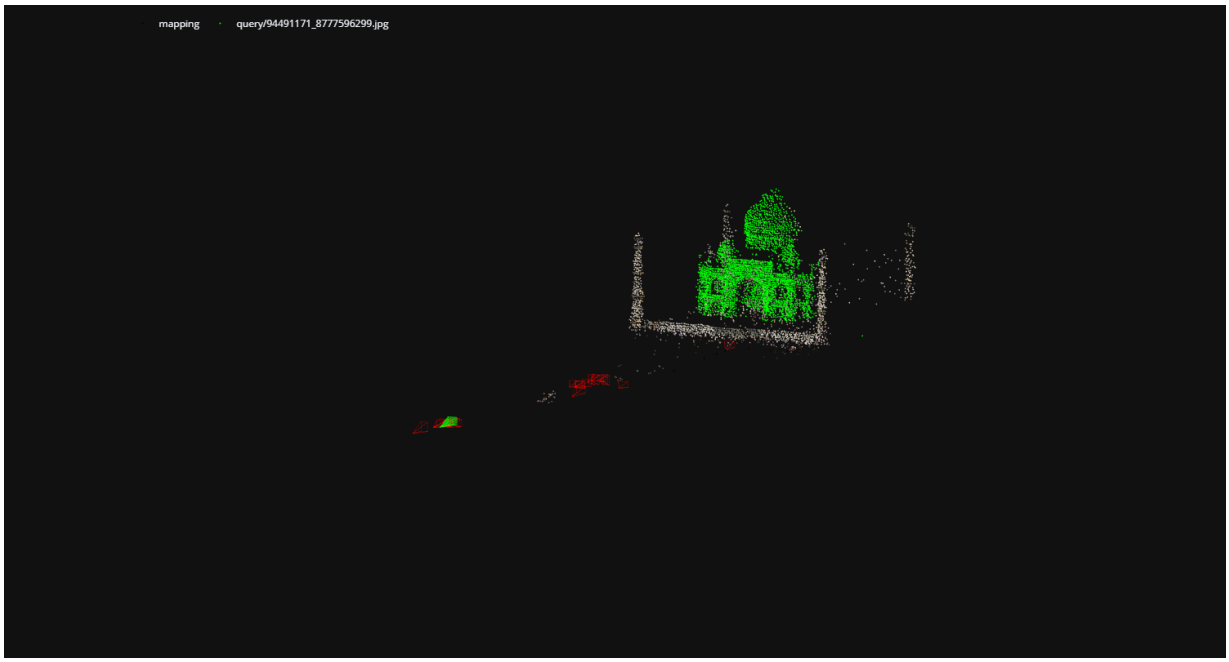

Fig 16. Query Image Camera View Projection

3.1.3 Benchmarking analysis

In our project, we chose following evaluation metrics:

- Number of 3D points

It is the total count of triangulated 3D points that the system has reconstructed from 2D images.

- Number of Observations

It measures the total number of matched keypoints across all image pairs, which are used to triangulate the 3D points.

- Mean Reprojection Error

It is calculated by projecting 3D points back into the images using the estimated camera poses and then measuring the average distance between the projected points p_{ij} and the actual image points $\widehat{p}_{ij}$. The error is averaged over all points in all images.

$$Mean\ Reprojection\ Error\ =\ \frac{1}{N} *\ \Sigma_{ij}||p_{ij} - \widehat{p}_{ij}||$$

- Mean Track Length

It measures the average number of images in which a 3D point is observed.

$$Mean\ Track\ Length\ =\ \frac{Sum\ of\ Lengths\ of\ all\ tracks}{Number\ of\ 3D\ points},$$

where a “tracks” refers to a series of corresponding feature points across multiple images that are the projection of the same 3D points in the scene; The “Length of a Track” is the number of images in which the corresponding feature points appear.

Type	Pairing	Extractor	Matcher	3D Points	Number of Observations	Mean Reprojection Error	mean_track_length
Sparse	NetVLAD	SuperPoint	KNN	1035	6551	0.910912	6.32947
		SuperPoint	SuperGlue	1934	14749	1.20304	7.62616
		SuperPoint	LightGlue	1983	14825	1.20297	7.47605
		DISK	LightGlue	5141	65123	1.00131	12.6674
Dense		SuperPoint	LOFTR	20121	105956	0.531624	5.26594
		SIFT	LOFTR	20147	105779	0.532833	5.25036
		R2D2	LOFTR	20234	105904	0.528369	5.23396
		DISK	LOFTR	20135	105742	0.53345	5.25165
BaseLine		SIFT	KNN	697	3411	0.787575	4.89383

From the above result table, we can observe that:

1. In SfM, there is often a trade-off between the density of the reconstruction, the accuracy of 3D points and the computation resources required.
2. Sparse methods use a smaller number of feature points and are generally faster but less accurate than dense methods.
3. Using DISK as feature extractor and LoFTR as feature matcher has low mean reprojection error and a high number of 3D points. This demonstrates a favorable balance between feature matching quantity and quality.
4. When comparing methods performance of sparse type, we found DISK provides much more 3D points compared with other sparse methods.
5. When comparing methods performance of dense type, we found the choice of feature matcher matters, in designing an effective SfM workflow. LOFTR demonstrates strong performance regardless of extractor used, though it takes much more time to run.
6. When it comes to application in real world scenarios, for applications where accuracy is critical, such as precise architectural reconstructions, the methods with lowest mean reprojection errors (R2D2 with LoFTR, SIFT with LoFTR and DISK with LoFTR) might be preferable despite producing fewer 3D points.
7. If the goal is to capture a large number of features for broader reconstruction, such as in large-scale environment mapping, methods that produce more 3D points might be more appropriate, like NetVLAD with SuperPoint and LoFTR.

3.2 Other Results

3.2.1 Feature Matching with image splitting

In this subsection, we performed feature matching using SIFT & BFMatcher on images *with and without splitting*. We found splitting each image into sections and then performing feature matching often results in finding more good matches compared to doing without splitting. The first obvious reason is that by splitting, the feature detection algorithm can focus on local details that might be overlooked in a full-image analysis. The second reason can be that smaller sections can reduce ambiguity since in a full image, certain features might appear similar to multiple features in another image, leading to incorrect matches. One thing to notice is that splitting images allows for parallel processing of each section in a batched process, thus improving calculation efficiency.

3.2.1.1 With image splitting

We divided each image into four sections. Four sets of keypoints are generated respectively and matched across all combinations (4x4 = 16 pairs). In this section, we found that the total number of good matches across all sections is 300.

Fig 17. Image Split Match

3.2.1.2 Without image splitting

If we applied SIFT and BFMatcher directly without image splitting, we found 130 good matches with a threshold of 0.75.

Fig 18. Match without Split

3.2.2 Feature Matching with Overlap Detection — using LoFTR and SIFT in sequence

In this subsection, we used the first round matching results to obtain the overlap region and then performed the second round of matching within them.

LoFRT is used first to match features between the two full images. The matched keypoints obtained from LoFTR are then used to identify overlapping regions in the two images. The overlapping regions are resized then. After that, SIFT is used to perform feature extraction and matching within these specific regions. The Brute-Force Matcher is then used to match the SIFT descriptors between the two overlapping regions.

This approach leverages the strengths of both methods: LoFTR's effectiveness in handling large-scale and perspective changes, and SIFT's robustness in detailed feature matching within a constrained area.

Overlap detection helps in identifying the common regions between images where feature matching is likely to be more successful. By focusing on overlapping regions, the algorithm can concentrate computational resources where they are most needed. Also, By identifying overlapping regions first, we can reduce the likelihood of false matches (matches with features that are not actually corresponding points in the scene), particularly in images with repetitive textures or patterns.

Note that the LoFTR model is expecting single-channel (grayscale) images as input instead of a three-channel RGB images.

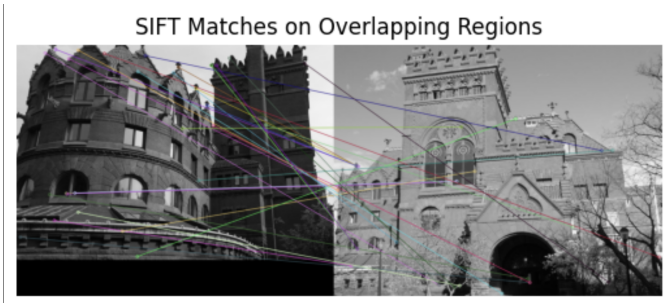

Fig 19. Overlap Match

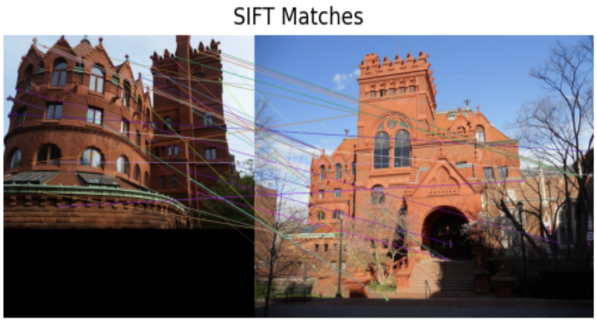

Fig 20. Non overlap - SIFT

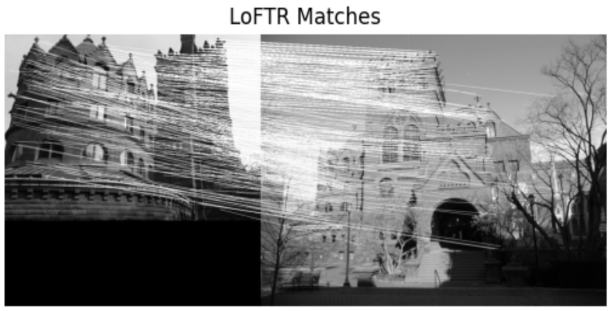

Fig 21. Non overlap - LoFTR

Then, we also implement feature matching without overlap detection but with a method ensemble of LoFTR and SIFT. That means, use LoFTR to initially identify broad matches across the images and then apply SIFT for detailed matching within the regions identified by LoFTR.

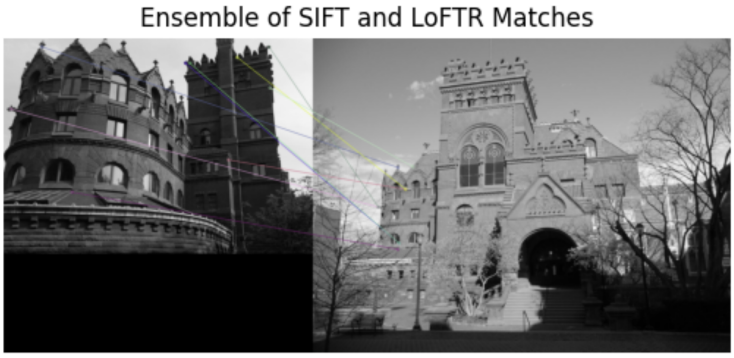

Fig 22. SIFT with LoFTR

3.2.3 Benchmarking Analysis

This is the performance table with regard to the above four different methods,

Method	Time efficiency (s)	Num of good matches
Overlap detection (LoFTR—>SIFT)	16.876	32
SIFT	0.176	41
LoFTR	16.950	382
LoFTR + SIFT	30.153	9

From the above result, we observed that

1. Adding overlap detection to the feature matching process can ensure the relevance of the matches but does not necessarily translate to good matches. It also adds to the computational time.
2. Combining methods does not necessarily lead to better performance. In this case, using LoFTR and SIFT together without overlap detection significantly reduces the number of good matches and increases the time required.

4. Qualitative analysis

4.1 Ideal Dataset and challenges

The ideal dataset for this project would be high-resolution images with a wide baseline and sufficient overlap, taken under consistent lighting conditions, and with rich textures to facilitate feature detection and matching. Addressing images of varying quality, lighting conditions, occlusions are challenges we faced in the feature extraction & matching process.

4.2 Potential Use-Cases and Benefactors

This project has wide use-cases and many benefactors. Firstly, it helps Heritage Conservation. 3D models of archaeological sites can be created for preservation, study, and restoration. This is particularly valuable for sites that are inaccessible or at risk of damage from environmental factors or human activities. Additionally, it benefits Robotics and Autonomous Systems. By transforming 2D sensory data—such as images from cameras—into 3D and estimating poses, these systems are able to better understand their surroundings and decide how to move and interact with it. Furthermore, it benefits Urban Planning and Development. Urban planners can use 3D models to visualize new projects within the existing city context, aiding in design and impact analysis.

4.3 Social Aspects

In terms of social impacts, this project should take cultural sensitivity into consideration. That includes respecting the narratives and histories associated with the sites, ensuring that the reconstructions do not misrepresent or appropriate cultural heritage.

As for any potential privacy issues, images may inadvertently capture individuals, private homes, or sensitive locations. The privacy concern arises when these images are used without the consent of the individuals or property owners depicted. To mitigate privacy issues, the project must ensure that any identifiable information is either anonymized or blurred out, or that explicit consent is obtained from the individuals or property owners whose images or properties are captured.

5. Conclusion of Learning Experience

The main challenge we faced was to familiarize ourselves with SfM technique. We studied following materials:
https://image-matching-workshop.github.io/slides/slides-eric.pdf
https://ducha-aiki.github.io/wide-baseline-stereo-blog/2023/07/05/IMC2023-Recap.html
https://www.youtube.com/watch?v=9JpGjpITiDM

Reference

Check Orientation: https://github.com/ternaus/check_orientation

COLMAP: https://colmap.github.io/

Dataset: https://www.kaggle.com/competitions/image-matching-challenge-2023/data

DISK: https://arxiv.org/abs/2006.13566

Hierarchical Localization: https://github.com/cvg/Hierarchical-Localization

LightGlue: https://github.com/cvg/LightGlue

LoFTR: https://zju3dv.github.io/loftr/

NetVLAD: https://arxiv.org/abs/1511.07247

R2D2: https://arxiv.org/abs/1906.06195

SIFT algorithm: https://www.cs.ubc.ca/~lowe/keypoints/

Sparse Modeling: https://demuc.de/tutorials/cvpr2017/sparse-modeling.pdf

Structure from Motion: https://inst.eecs.berkeley.edu/~cs180/fa23/Lectures/sfmmvs.pdf

SuperGlue: https://arxiv.org/abs/1911.11763

SuperPoint: https://arxiv.org/abs/1712.07629