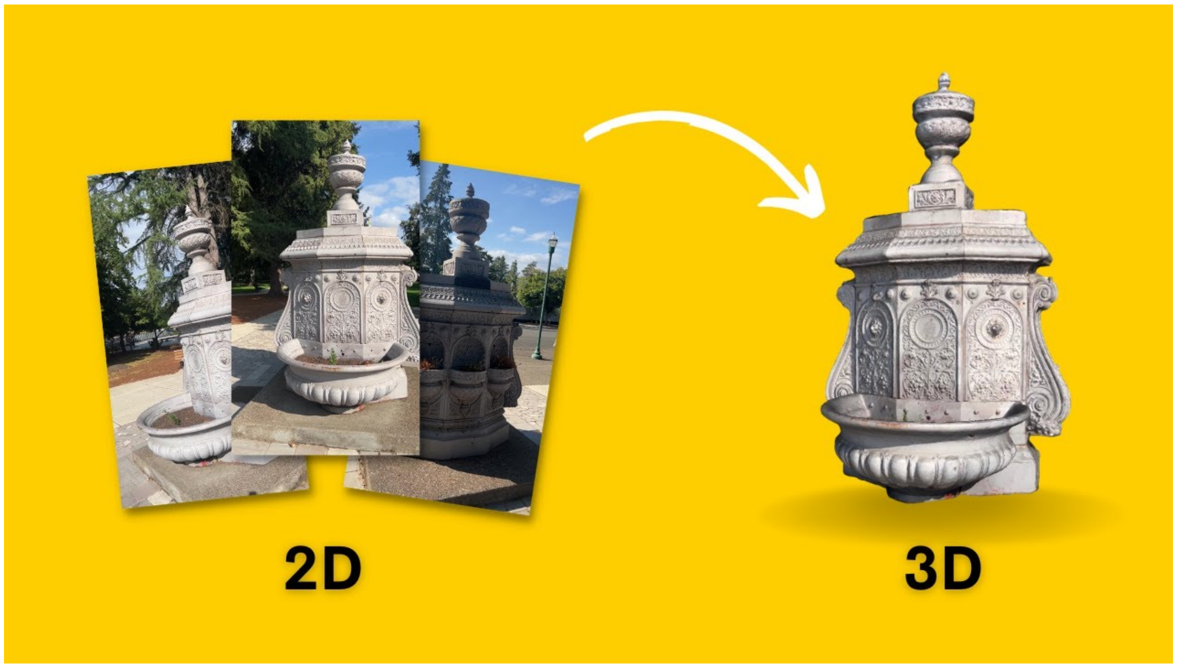
Geometric 3D Reconstruction

Introduction
In an era where digital content is paramount, the transformation of two-dimensional imagery into three-dimensional models represents a significant leap forward in how we interact with and interpret visual data. This report explores the realm of 2D to 3D reconstruction, an innovative process that infuses flat images with depth and vitality, giving rise to tangible, navigable 3D spaces. Through comprehensive analysis and synthesis of current methodologies, this document serves as a detailed guide to the technological advances and the multifaceted applications of this transformative field.

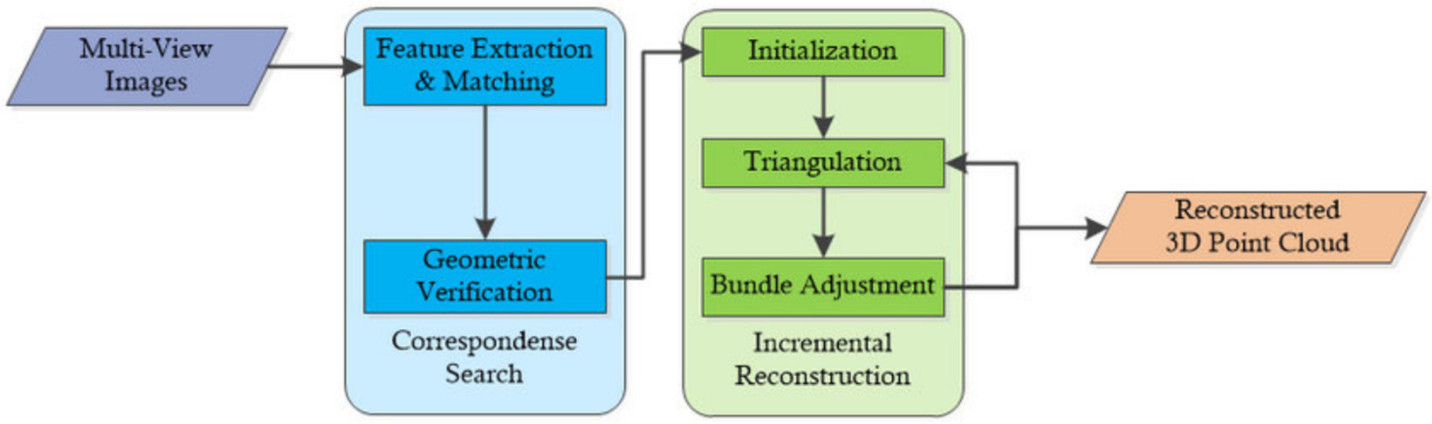
Motivation
The pursuit of 3D reconstruction is driven by the quest to capture more than just the visual cues presented in a 2D frame. There is an inherent desire to recreate the full context of an experience, from the majestic expanse of a landscape to the intricate details of an artifact. This technology empowers us to not only preserve visual memories but also to interact with them, enhancing fields as diverse as virtual reality, urban planning, and historical preservation. It opens a window to revisit the past, design the future, and explore paths untaken, all from a series of 2D snapshots.



From Snapshots to Sculpture: Unveiling the magic of 2D to 3D reconstruction, where multiple angles come together to recreate the complete story of an object in space.

Literature Review:
- **Overview:** The process of 3D dense reconstruction typically encompasses several key steps. Image acquisition involves capturing multiple images or video frames of the scene from varied angles, with the quality and resolution of these input images profoundly influencing the accuracy of the final 3D model. Feature detection and matching follow, where distinctive features or keypoints are identified in the images, and correspondences between these features are established. Common algorithms for feature detection include SIFT, SURF, and ORB. Subsequent steps involve camera pose estimation, depth estimation, surface reconstruction, and texturing, ultimately resulting in a photorealistic 3D model. Despite the availability of various algorithms and methods, achieving perfection for any specific task remains elusive.

- **Structure from Motion:** Structure from Motion (SfM) is a computer vision technique designed to estimate the 3D structure of a scene or object from a sequence of 2D images or video frames captured from different viewpoints. This involves analyzing the apparent motion of object points in the image sequence, with key steps mirroring the general 3D reconstruction pipeline. SfM proves valuable in applications such as robotics, aerial mapping, and 3D reconstruction.
- **Shape from Shade:** Shape from Shading (SfS) aims to recover the 3D shape or geometry of an object or scene from a single 2D image by analyzing shading or intensity variations. This technique operates on the assumptions of a Lambertian surface, known lighting conditions, smooth surface, and a single image. Despite its limitations, such as sensitivity to noise and reliance on assumptions, SfS remains crucial for 3D reconstruction, particularly in scenarios with only a single available image.
- **SLAM:** Simultaneous Localization and Mapping (SLAM) is a technique used in robotics and computer vision to estimate the camera's pose and create a 3D map of the environment simultaneously. SLAM-based 3D dense reconstruction, an extension of SfM, aims to generate detailed and accurate three-dimensional representations in real-time. Various methods, including KinectFusion, ElasticFusion, LSD-SLAM, ORB-SLAM, and Dense Tracking and Mapping (DTAM), offer real-time performance and find applications in robotics, autonomous navigation, augmented reality, and virtual reality, excelling in dynamic and real-world scenarios.
- **Summary:** In the realm of image analysis, numerous algorithms and methods have been devised for the identification of distinctive invariant features. Among these, SIFT, pioneered by David Lowe, held a prominent position until the advent of SURF. In response to the intricacies associated with SIFT and SURF, novel binary descriptors like BRISK and FREAK emerged, demonstrating promise in specific cases. While these algorithms serve as viable options for detecting distinctive invariant features across various applications, comprehensive comparisons of their performance and computational costs reveal that none is universally perfect for any given task. In the domain of 3D reconstruction, the foundation lies in establishing matches between multiple images of a common scene. The groundbreaking "Photo Tourism" project, a collaboration between the University of Washington-Seattle and Microsoft, marked a significant stride in this field. Leveraging a multitude of photos from diverse sources as input, the project utilized an incremental bundle adjustment approach to compute each photo's viewpoint and a sparse 3D model. Subsequent developments, namely CMVS and PMVS2, further refined the process to achieve 3D dense reconstruction.



The Process of 2D to 3D Reconstruction

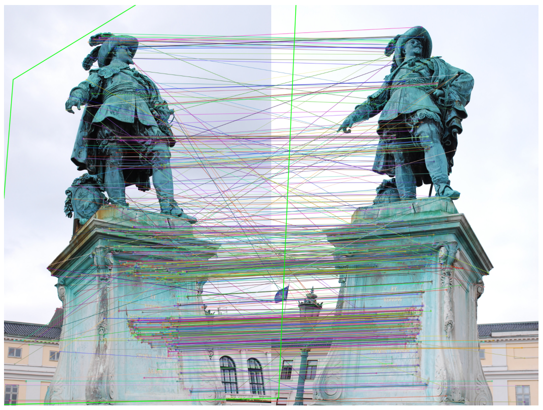
Algorithm

1. Initialize camera parameter for image 1 and image 2
2. Load camera 1 intrinsic parameters with the assumption that the camera 1 is at the origin
3. Feature detection and matching between image 1 and image 2
4. Using essential matrix method to calculate the relative pose of camera 2
5. With both camera parameters using triangulation to get 3D points
6. For img 3 to img N:
    a. Feature detection and matching between image 2 and image 3
    b. Find common points between these features and features in image 1
    c. Use the 3D points, common features to get camera parameters for image 3 using PnP
    d. Use triangulation to get 3D points for common features between image 2 and image 3
    e. Calculate Reprojection Error
    f. Bundle Adjustment

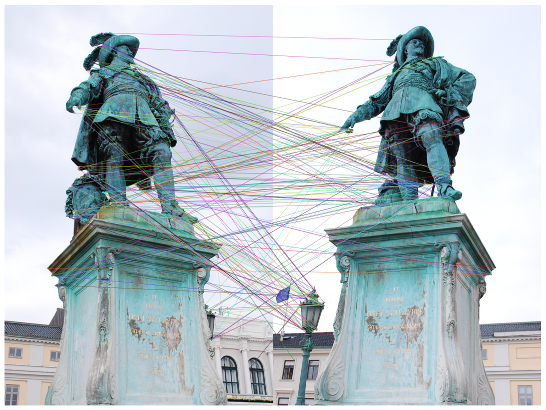
We are given the intrinsic parameter of the camera and we have assumed that this camera has taken all the photos associated with a particular object. We solve this problem for 2 images and then extend it to n images. We assume that the first camera is at the origin, so knowing this we know both the intrinsic and extrinsic parameters of the first camera. For the second camera, we know only the intrinsic parameters and will have to solve for the extrinsic parameters.

We then find the matching keypoints between image 1 and image 2 using SIFT-RANSAC. We also compare the keypoints we obtain after using ORB.

SIFT-RANSAC



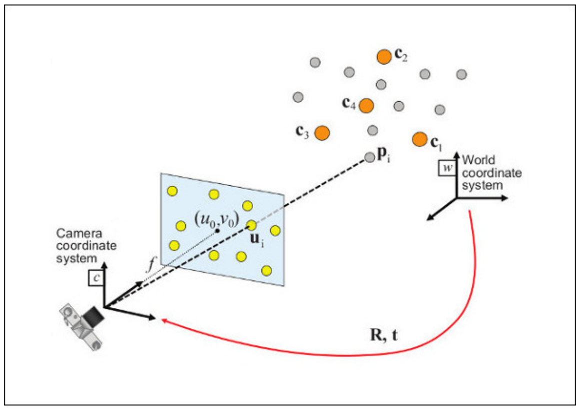
ORB



We can see that learn a denser representation from SIFT and hence we go ahead with SIFT as we want to get a denser 3D reconstruction.
Using these features, camera matrix of camera 1 we are able to calculate the extrinsic parameters for camera 2 using essential matrix and recover pose functions. Using these two camera parameters and the common features we use triangulation to obtain the 3D point locations of these matching features.
Now we will extend this logic for images 3 to images N. Our goal is to find the camera parameters of each of the cameras and then we will use triangulation to get the 3D point locations. We explain our procedure for image 3 and then it can be extended to the rest of the images as well. We find the matching features between image 3 and image 2. We then see if there are any common points between these features and the features in image 1. The reason we do this is because if there any common points then we already have their 3D point clouds calculated. This perfectly falls into our perspective n point criteria. We can calculate the camera parameters of camera 3 with matching key points, common 3D points and camera parameters of camera 2. With these calculated camera 3 points we are able

to get the 3D position of the remaining matching features using triangulation. This can be extended for all the N images and we get our final 3D point cloud.

At each step our goal is to minimize the reprojection error of the 3D point cloud onto the image plane. To minimize it we use bundle adjustment with the help of the least squares optimizer in scipy. We refine our estimated camera parameters and the position of the 3D points. This is a very slow process which can take hours to run, hence we keep it as an additional feature on top of our system.

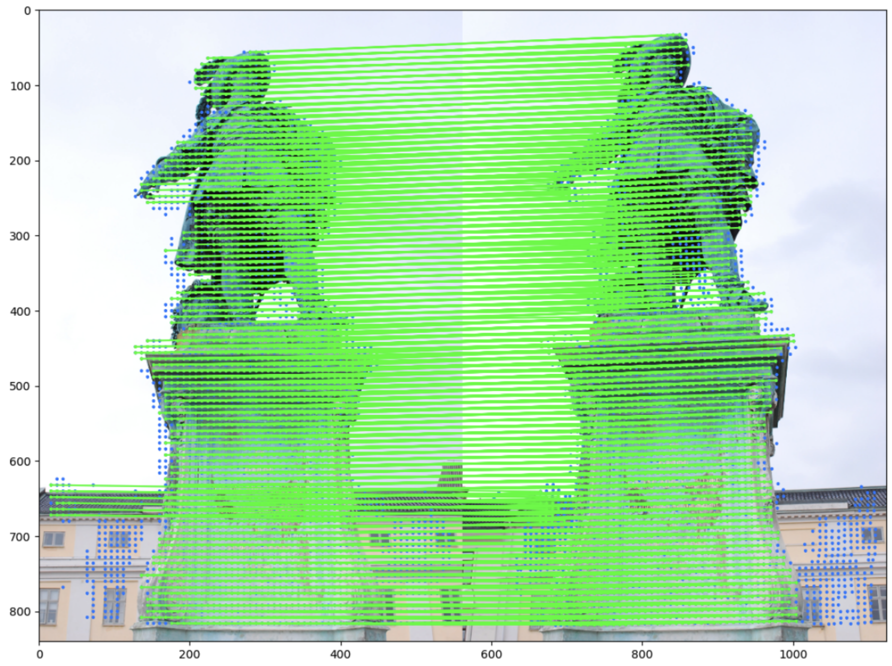


<u>Enhancements:</u>

The methodology that we used until now included using SIFT features detected in the earlier steps and to project them onto the 3D space. Since we were only using the SIFT feature points to be projected onto the 3D space, we were getting sparse representations for the 3D object reconstructed. Our main goal now became to get denser representations for the object we were reconstructing. To do this, we employed two main techniques, one was to incorporate additional features (extracted through a Deep Learning algorithm) and other was to add Dense Matching. The below sections explain the two techniques in detail.

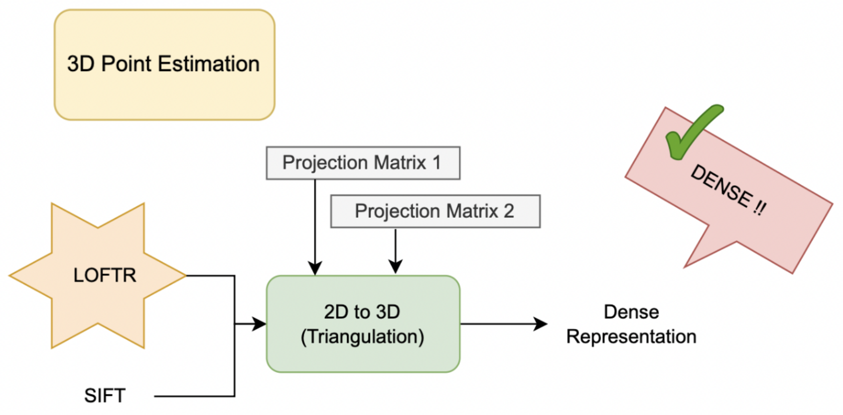
1. Incorporating Deep Learning based features (LOFTR features):

While doing the literature review, we found a Deep Learning based algorithm for finding features between two images. It uses Convolutional Neural Networks to encode input images and then uses self-attention and cross-attention in transformers to generate features common to both images. This way, the algorithm also finds features common to both images. The algorithm also returns the score of each of the features it finds. Thus, this seemed a perfect fit for our use case and hence we decided to incorporate LOFTR features into our model.



LOFTR Feature Detection Algorithm Results on Dataset.

Once we found the SIFT features and Projection Matrices, instead of just projecting the SIFT features onto the 3D space, we, used SIFT features along with the LOFTR features and since we already had the Projection matrices, it became feasible for us to project any point onto the 3D space. This way we incorporated LOFTR features into our pipeline.



Reconstruction Model with SIFT and LOFTR Features

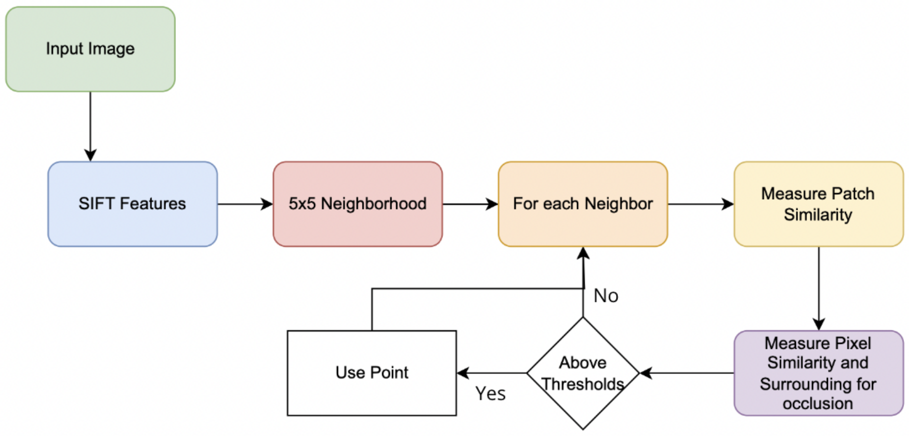
A plus point about using LOFTR is that the number of features found by LOFTR algorithm are 20-40x times the number of features found by the SIFT algorithm. However, through manual inspection, a caveat that we observed, was that the features found by LOFTR are very random and quite a lot of them can point to the background image which we did not want to use. Another issue we found using too many LOFTR features between multiple pairs of images, was that once all the 3D points were projected into the 3D space, it resulted in some noise. Thus, we decided to use the LOFTR features with reservations by applying a very high confidence threshold.

2. Including Dense Matching:

Again, while doing the literature survey, we came across a technique - Dense Matching in the publication "Match Propagation for Image-Based Modeling and Rendering" in the "IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002" [10] journal and it really promised of better reconstruction results. The whole of the algorithm was out of our scope, however, we chose to implement a subpart of the algorithm by tweaking the original one for our use-case. The original algorithm proposed considering a window (5x5) around the 2D points detected by a feature detection algorithm, like SIFT, and then expanding from there. The neighborhood points are matched one by one in the two images and the matches are stored in the Heap data structure where the best match is picked at an iteration and again the neighborhood of that new point is picked up. Thus, this process recursively finds and considers the best matches in the image and then expands from there if the point has not been considered already. We store all these, points (and this way the points detected by the original feature detector are also picked up) and hence we get a set of good points to reconstruct the 3D images.

For our use-case, we did not implement the algorithm recursively and simply used the algorithm on the features detected by the SIFT algorithm. We pick up the SIFT feature points, consider their neighborhood, pick up the best matches and simply use them instead of expanding. This helps us not to deviate too much from the SIFT features (which turn out to be very accurate but super sparse) and tackles the challenge of sparsity by adding the relevant neighborhood points.

Just like in the case of LOFTR features, we again were very conservative in terms of the additional 2D points we add to the 3D reconstruction in addition to the SIFT features points. We were highly restrictive in terms of choosing the thresholds.

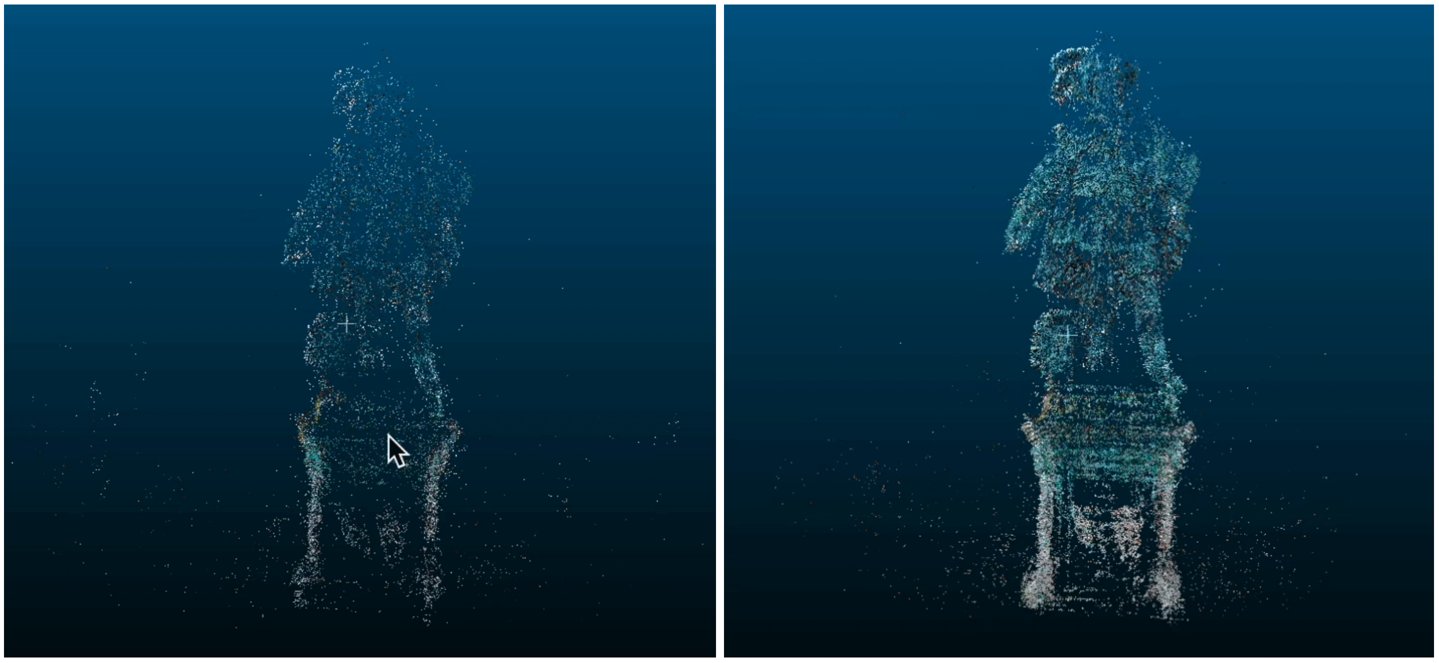


Our modified Dense Matching Algorithm.

However, even with very tight thresholds, we were able to increase the feature points by up to 3-4 times for an image where SIFT features were detected nicely. Thus, the overall impact of the enhancements we added was pretty significant.

Improved Results:

1. Augmenting with LOFTR features



a) 3D Output using SIFT + RANSAC

b) 3D Output using SIFT + RANSAC + LOFTR

We can observe from (a) and (b) that adding optimal LOFTR features results in a denser reconstruction due to the following reasons:

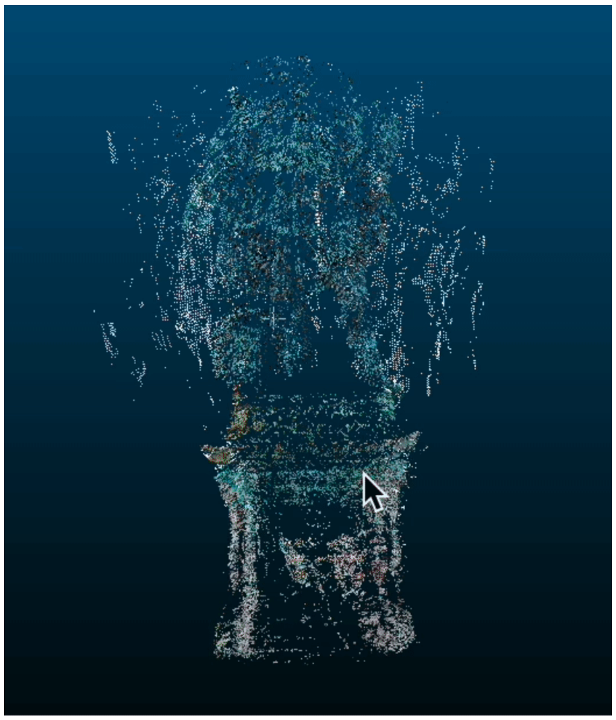
- Robustness to Outliers:

    - RANSAC is effective at filtering out outliers during the matching process, leading to more accurate correspondences between keypoints in different frames.

- Local Tracking and Temporal Consistency with LOFTR:
    - LOFTR contributes to temporal consistency by considering the local tracking aspect. This means that features are not just matched in a single frame but are tracked across frames, providing additional information for denser reconstruction.

- Complementary Nature of Techniques:
    - SIFT provides distinctive features, RANSAC ensures robust estimation of transformations, and LOFTR contributes to temporal tracking. The combination of these techniques addresses different aspects of the reconstruction process, complementing each other for a more comprehensive solution.

One important observation while incorporating dense LOFTR features is that the confidence threshold must be set very high to filter out any unnecessary background information.



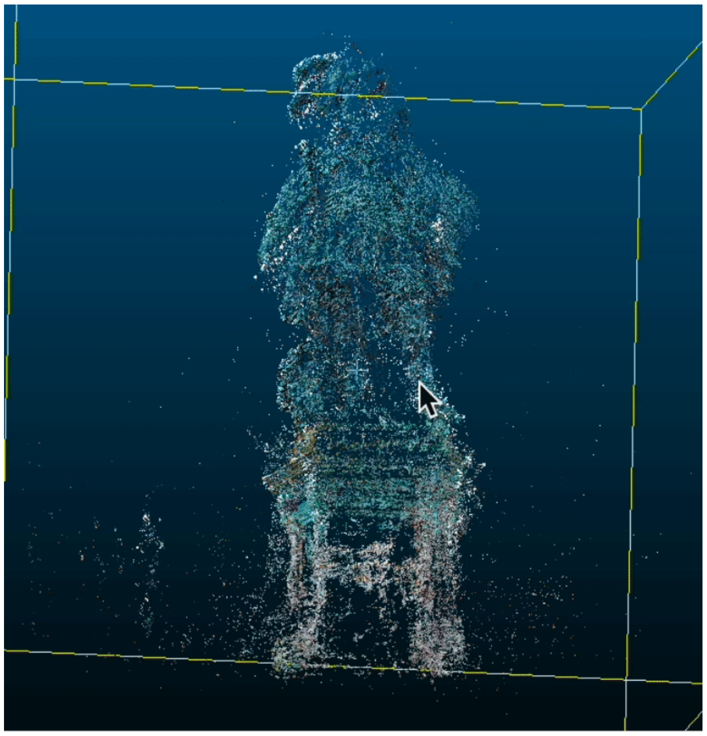
We can see that the output deteriorates if the threshold is not optimized due to the following reasons:

- Inclusion of Ambiguous Keypoints:
    - Lowering the confidence threshold can result in the inclusion of keypoints that are more ambiguous or less distinctive.
    - Ambiguous keypoints might correspond to background elements or other non-discriminative features, contributing to the capture of unnecessary background information in the tracking.

- Increased Sensitivity to Noise:
    - Lowering the confidence threshold makes LOFTR more sensitive to noisy or less reliable matches.
    - Noisy matches, especially those associated with background elements, can be falsely included in the tracking process.

- Reduced Filtering of Outliers:
  - LOFTR employs confidence thresholds to filter out unreliable matches and outliers.
  - Lowering the threshold reduces the stringency of this filtering process, allowing more matches, including those that may correspond to background regions, to be considered during tracking.

2. Incorporating Dense Matching



Adding dense matching to the combination of SIFT + RANSAC + LOTR leads to a denser output due to the following reasons:

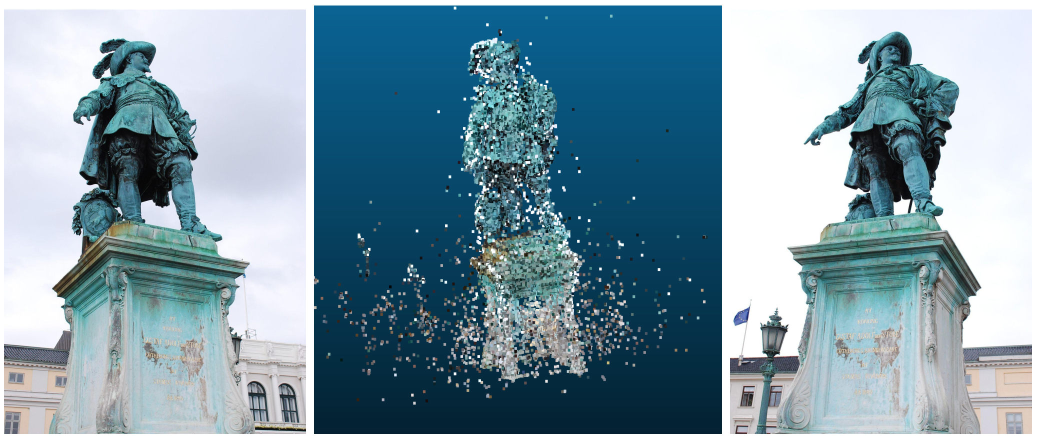
- Complementing Sparse Matching:
  - SIFT, RANSAC, and LOFTR typically operate on sparse feature correspondences. These methods identify and match distinctive keypoints or features in the images.
  - Dense matching, on the other hand, involves matching information across a dense grid of pixels or patches in the images, providing additional correspondences beyond the sparse set of keypoints.
  - This increased number of correspondences contributes to a denser set of data for subsequent stages of the pipeline, such as the application of geometric transformations and filtering with RANSAC.
  - The combination of sparse and dense matching allows for a more comprehensive and detailed understanding of the scene.

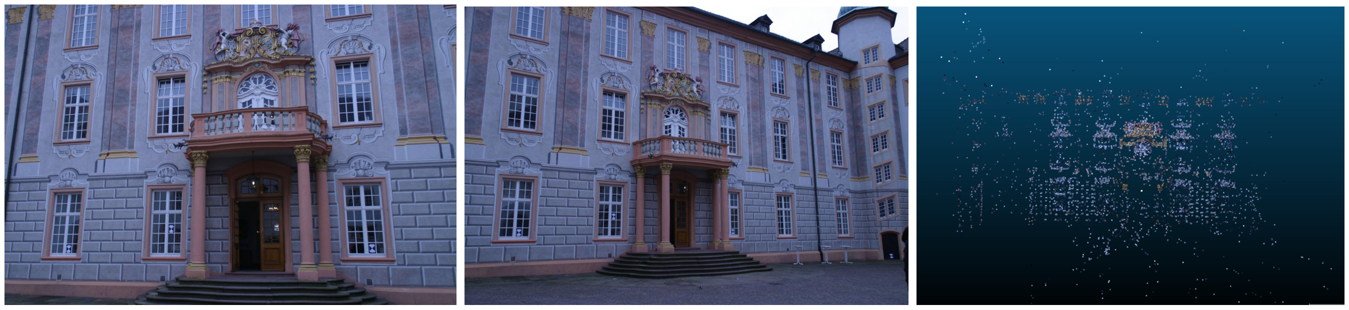
- Improved Coverage:
  - Dense matching helps cover areas that may not have distinct keypoints. Sparse methods like SIFT might struggle in regions where keypoints are not well-defined or in ambiguous or repetitive scenes where distinguishing keypoints becomes difficult.
  - By performing dense matching, more information is gathered from these less distinctive regions.

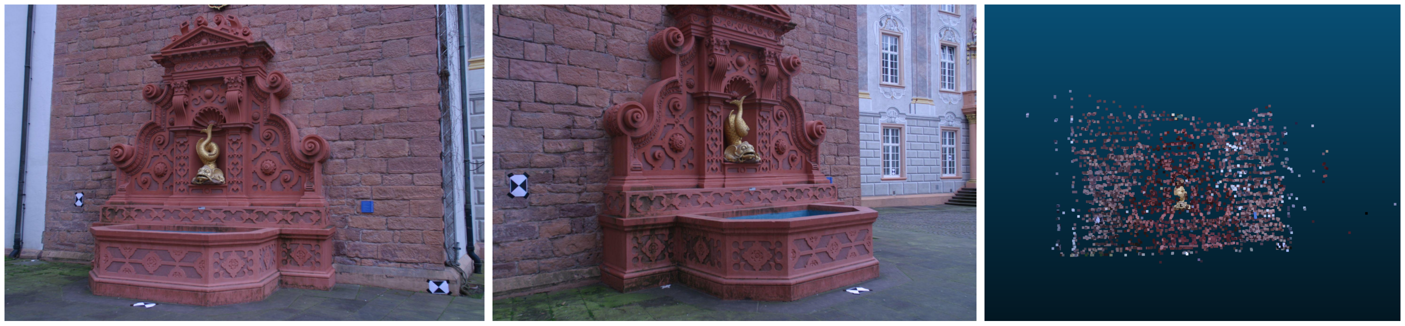
- Increased Robustness to Occlusions:
  - Dense matching is less prone to occlusions since it considers a larger number of points in each image.
  - In scenarios where objects are partially occluded or overlap, dense matching can still provide correspondences in regions not affected by occlusions, allowing for more accurate interpolation and estimation of pixel values.

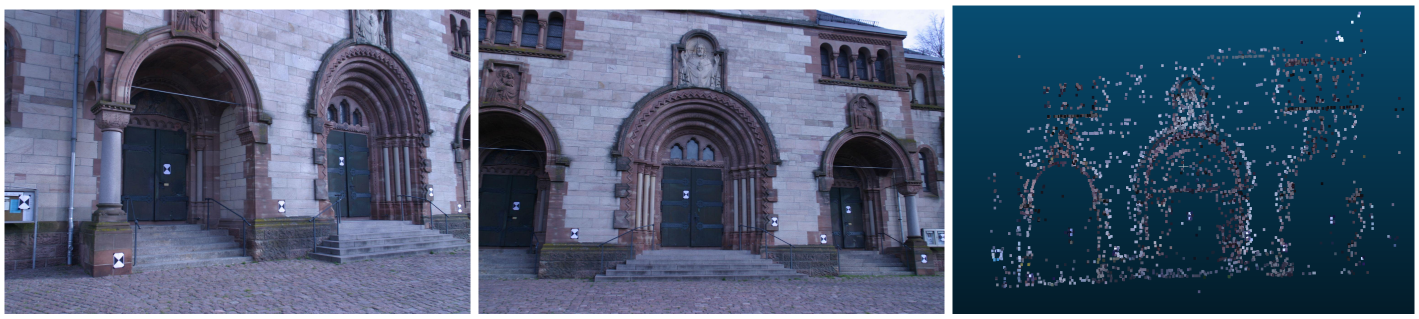
Experiments:

We ran a few ablations to test the algorithm in low resource scenes (with fewer images). The results are shown below:


a) Input Images and 3D Output with **57** images per scene


b) Input Images and 3D Output with **10** images per scene


c) Input Images and 3D Output with **11** images per scene


d) Input Images and 3D Output with **8** images per scene

The quality of the reconstructed output tends to deteriorate when the dataset has fewer images per scene due to several reasons:

- Limited Viewpoints and Sparse Correspondences:
    - With fewer images per scene, there is a limited set of viewpoints available to capture the entire scene comprehensively.
    - Lack of diverse viewpoints can result in incomplete coverage of the scene, leading to missing details and potential gaps in the reconstructed output.
- Less Redundancy:
    - Redundancy in image data is crucial for robust reconstruction. Having more images of the same scene allows the algorithm to handle outliers and errors more effectively.
    - Fewer images mean less redundancy, making the reconstruction more susceptible to noise and outliers, which can negatively impact the quality of the output.
- Limited Baseline for Triangulation:
    - Triangulation relies on having a sufficient baseline between multiple camera views for accurate depth estimation.
    - A limited number of images may result in smaller baselines, making it challenging to accurately triangulate points and leading to less precise reconstructions.
- Increased Sensitivity to Calibration Errors:
    - Camera calibration parameters play a crucial role in 3D reconstruction. With fewer images, there is less opportunity to refine and accurately estimate these parameters.
- Difficulty Handling Occlusions:
    - In scenes with fewer images, occlusions may be more challenging to handle. Occluded areas may lack sufficient information for accurate reconstruction.
    - Limited views make it harder to infer the structure behind occluded regions, leading to inaccuracies and artifacts in the reconstructed output.

Future Work:

1. Improve Generalizability:

We plan on combining our current pipeline with a trained deep learning model that can enhance the generalization of the system in varied environments.

Here's how we plan to integrate these components for improved performance:

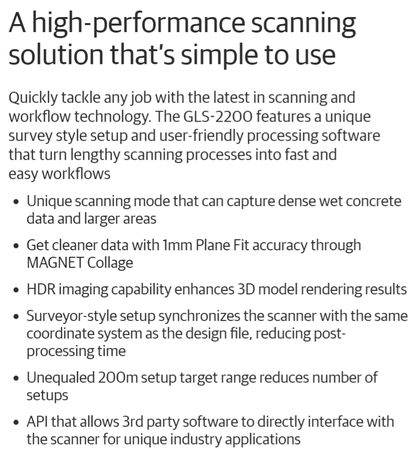
- Hybrid Deep Learning-based Feature Matching:
    - Train a deep learning model to perform feature matching (we were using a pretrained LOFTR model earlier just for testing). This model can learn complex patterns and relationships in the data, allowing for more accurate and robust matching compared to traditional methods.
      OR
    - Utilize transfer learning to fine-tune a model that has been pre-trained on a large dataset to adapt it to the specific requirements of our environment.

- Training the Deep Model for Generalization:
    - Train the model on a diverse dataset that represents the variations expected in different environments. This can include changes in lighting conditions, viewpoints, and scene complexity.
    - Augment the training data to expose the model to a broader range of scenarios, helping it generalize better to unseen environments.

- Adaptive Feature Selection:
    - Implement mechanisms to dynamically adjust the contribution of SIFT and deep learning-based features based on the characteristics of the environment.
    - This adaptability allows the system to prioritize one type of feature over another depending on the scene, leading to improved generalization.

2. Create a Custom Dataset:

- Capture scenes from different viewpoints with sufficient coverage, varied lighting conditions and minimal reflections.
- Calibrate cameras to leverage intrinsic parameters such as focal length and principal point for accurate reconstruction.
- Annotate images with ground truth information including marking keypoints and object boundaries.
- Capture depth information along with RGB images using depth sensors or stereo cameras.



A high-performance scanning solution that's simple to use

Quickly tackle any job with the latest in scanning and workflow technology. The GLS-2200 features a unique survey style setup and user-friendly processing software that turn lengthy scanning processes into fast and easy workflows

- Unique scanning mode that can capture dense wet concrete data and larger areas
- Get cleaner data with 1mm Plane Fit accuracy through MAGNET Collage
- HDR imaging capability enhances 3D model rendering results
- Surveyor-style setup synchronizes the scanner with the same coordinate system as the design file, reducing post-processing time
- Unequaled 200m setup target range reduces number of setups
- API that allows 3rd party software to directly interface with the scanner for unique industry applications

CERLAB's Laser Scanner ArUco can be used to capture ground truth 3D information

References

1. Lowe, D. G. (2004). "Distinctive image features from scale-invariant keypoints." International Journal of Computer Vision, 60(2), 91-110.

2. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). "ORB: An efficient alternative to SIFT or SURF." In 2011 International Conference on Computer Vision (ICCV) (pp. 2564-2571).

3. Hartley, R., & Zisserman, A. (2003). "Multiple View Geometry in Computer Vision" (2nd ed.). Cambridge University Press.

4. Lepetit, V., Moreno-Noguer, F., & Fua, P. (2009). "EPnP: An accurate O(n) solution to the PnP problem." International Journal of Computer Vision, 81(2), 155-166.

5. Hartley, R., & Zisserman, A. (2003). "Multiple View Geometry in Computer Vision" (2nd ed.). Cambridge University Press.

6. Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W. (2000). "Bundle adjustment — a modern synthesis." In Vision algorithms: Theory and practice (pp. 298-372). Springer.

7. Hu, Q., Yang, B., Xue, H., Kong, H., & Wang, Y. (2021). "LOFTR: Descriptor-Based Long-Term Visual Localization for Aerial and Ground Robots." arXiv preprint arXiv:2104.15536.

8. Goesele, M., Snavely, N., Curless, B., Hoppe, H., & Seitz, S. M. (2007). "Multi-view stereo for community photo collections." In 2007 IEEE 11th International Conference on Computer Vision (ICCV) (pp. 1-8).

9. Yan, Z., Liu, Z., Li, P., Zhou, Q., Lei, J., & Liao, R. (2020). "DeepLO: Towards Efficient and Stable

Monocular Egomotion Estimation with Deep Neural Networks." In European Conference on Computer Vision (ECCV) (pp. 593-608).

10. Maxime Lhuillier, Long Quan. Match Propagation for Image-Based Modeling and Rendering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24, p. 1140-1146. ffhal-00118524f

README for the code:

1. main.py: Script for 3D reconstruction
2. feature_matching.py: Comparing SIFT and ORB features
3. utils.py: Helper functions for 3D reconstruction
4. dataset: Consists of input images and camera intrinsic parameters
5. res: Consists of output ply file and camera parameters stored

Steps to run code:

1. Insert images folder in input folder with intrinsic camera parameter matrix named as K.txt
2. Install cv2, numpy, scipy, os
3. Activate your environment and run main.py
4. Output is stored in res folder
5. Visualize output using Cloud Compare