Implementation of Structure from Motion and a Comparison of Eligible Feature Descriptors

Arjun Rajeev Warrier
Northeastern University
warrier.arj@northeastern.edu

Sahil Kiran Bodke
Northeastern University
bodke.s@northeastern.edu

***Abstract*—Structure from Motion (SfM) is a widely-used method in the fields of computer vision and photogrammetry for building a 3D model of a scene from a set of 2D images or video frames. With numerous applications in fields such as robotics, augmented reality, and virtual reality, it is a highly approached application. This project evaluates the performance of feature descriptors - SIFT, BRISK, and ORB for utilisation in keypoint extraction. An incremental SfM pipeline is then implemented using SIFT and its performance is implemented on two different datasets.**

***Index Terms*—3D reconstruction, Structure from Motion, SfM, Feature descriptors, SIFT**

I. Introduction

Image-based 3D reconstruction has been an active research area in computer vision and robotics. One common approach is to use feature-based methods that extract distinctive features from images and match them across multiple views to reconstruct the 3D structure of the scene. Feature descriptors, such as SIFT, BRISK, and ORB, are commonly used to extract these features and match them between images.

In this project, we evaluate the performance of these feature descriptors and their matching capabilities for image-based 3D reconstruction. We also implement an incremental Structure from Motion (SfM) pipeline, as shown in Fig.1, using SIFT and evaluate its performance on different datasets. The pipeline is capable of reconstructing 3D point clouds from a sequence of images captured from different viewpoints, making it useful for applications such as robotics, augmented and virtual reality, and object recognition.

A. Feature Descriptors

For the purpose of this project, we have only considered open-source feature description algorithms like SIFT, ORB and BRISK. SURF, which is another commonly used method has not been used because it is not open-source.

1) Scale-Invariant Feature Transform(SIFT): SIFT is a popular feature detection and description algorithm used in computer vision. It detects keypoints by identifying local extrema in the difference of Gaussian images at multiple scales, refines their locations and scales using Taylor expansion, assigns an orientation to each keypoint based on the gradient directions of nearby pixels, computes a descriptor for each keypoint based on the gradient orientations of nearby pixels, and matches keypoints between images using a nearest-neighbor approach. SIFT is known for its robustness to image transformations, but can be computationally expensive for large images or datasets.

2) Binary Robust Invariant Scalable Keypoints(BRISK): BRISK is a feature detection and description algorithm that detects keypoints using a scale-space pyramid, assigns an orientation to each keypoint, computes a binary descriptor for each keypoint, and matches keypoints using a nearest-neighbor approach. It is faster than SIFT due to its use of binary descriptors and a faster algorithm for keypoint detection, but may not perform as well in scenarios with significant viewpoint changes or image noise.

3) Oriented FAST and Rotated BRIEF(ORB): ORB is a feature detection and description algorithm that uses the FAST algorithm for feature detection, BRIEF algorithm for descriptor computation, and nearest-neighbor approach for keypoint matching. It is known for its speed and robustness to image rotations, making it useful in applications such as robot navigation and image stitching. However, it may not perform as well as other algorithms in scenarios with significant scale variation or image noise.

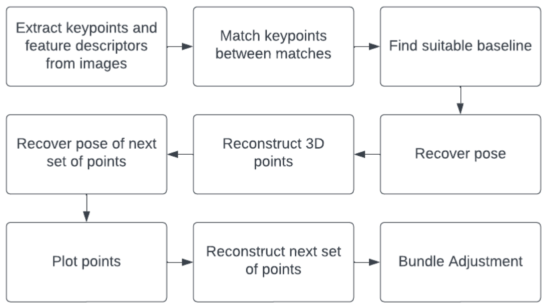
Extract keypoints and feature descriptors from images
Match keypoints between matches
Find suitable baseline
Recover pose
Reconstruct 3D points
Recover pose of next set of points
Plot points
Reconstruct next set of points
Bundle Adjustment

Fig. 1: SFM Pipeline

B. Structure from Motion (SfM) Implementation

The pipeline starts with taking in images/views and finding and describing keypoint features. It then proceeds to match keypoints across different views. A good baseline is chosen by selecting an image pair with the greatest difference in rotation. This baseline is then used to recalculate the camera pose and plot the intital corresponding 3D points by triangulating the 2D keypoints. These 3D points along with the keypoints in the next view is used to estimate the pose of the next view and then the 3D position of the corresponding points. After the initial 3D points are triangulated, the pipeline proceeds to recover the pose of the next view using the Perspective-n-Point (PnP) algorithm, which involves using the 3D points and corresponding 2D keypoints in the next view to estimate the camera pose. The pipeline then triangulates the 2D keypoints from the current view and the baseline view to estimate the 3D position of the corresponding points in the current view.

Next, the bundle adjustment step is performed, which involves optimizing the camera poses and 3D point positions by minimizing the reprojection error, i.e., the difference between the observed 2D keypoints and the projected 3D points in each view. Bundle adjustment can be performed using various optimization algorithms, such as Levenberg-Marquardt or Gauss-Newton.

Once the bundle adjustment is completed, the 3D points and camera poses are refined, and the final 3D point cloud is generated. The 3D point cloud can then be visualized using a suitable 3D visualization library or software.

Our implementation skips the need to compute and estimate poses for every combination of views and instead proceeds by selecting only image pairs that have good amount of matches between them. This has been seen to reduce the time taken for the pipeline while giving satisfactory 3D point clouds by bundle adjustment.

II. Related Work

For multi-view geometry problems like visual positioning, feature descriptors play a pivotal role. They find applications in a wide range of problems such as image matching, image retrieval, image classification, object classification, target tracking, change detection, etc. Hence, it is important to have a thorough understanding of the various feature descriptors available. Liu, W. et al in their survey [1] provide an in-depth analysis and review of various feature descriptors. They subdivide the traditional descriptors into two categories. One is the image feature descriptors based on local gradients and the second is the image feature descriptors based on image intensity. They start by mentioning the feature extraction as a two-step process - extraction of image feature points and description of image feature points. They talk about various feature detection algorithms such as harris corner detection algorithm, Shi-Tomasi corner detector, Local Binary Pattern (LBP), etc. They then explain about the feature descriptors such as SIFT, SURF, GLOH, etc. and present a comparison between local gradient based descriptors and image intensity based descriptors. They also mention the new avenue of research that has opened up due to the advancement in Machine Learning and Deep Learning algorithms and how traditional feature extraction and description can be done using this Learning based approach.
The feature descriptors are also an integral part of the Structure From Motion (SFM) pipeline which is the problem of reconstructing the 3D model of an object from multiple 2D images. Wu, C. et al worked on improving this problem of SFM. Their work focused on striking the right balance between speed and accuracy. They proposed a method to re-triangulate feature matches that initially failed to triangulate thereby dealing with the accumulated drifts without explicit loop closing. This ensured a higher accuracy. They significantly reduced the number of matched pairs (by up to 95%) but still recovered sufficient number of good matches for reconstruction. They also show through theoretical analysis and experimental validation that using their novel BA and re-triangulation methods, many substeps of incremental SFM, including Bundle Adjustment (BA) and point filtering take O(n) time in practice which is a huge improvement. They reported an average speed of 2 seconds for reconstructing 2 cameras and more than 2000 feature points for a very large photo collection.
Schönberger, Johannes L., et al. proposed a novel technique - COLMAP - to improve upon the existing state of the art SfM algorithms and this pipeline became the gold standard pipeline for evaluating SfM algorithms. The first segment of this pipeline is the 'Correspondence Search' which carries out feature extraction, matching and geometric verification. The next step in the pipeline is the 'Incremental Reconstruction' which consists of initialization, image registration, triangulation, bundle adjustment and outlier filtering. And finally, the 'Reconstruction' step. This technique has been an advancement in regards of advancements, robustness, scalability, and completeness and is still used as a state-of-the-art technique for many SfM algorithms.
Implementations from Harish V. et al [5] and Adil M. et al [7] seem promising as well especially due to the simplicity and convenience that they provide. However, they bring along challenges such as large computation times of about half an hour for a dataset as small as 10 images. This is primarily because they take into consideration all the possible pairs of images for finding matching features. This is a computationally intensive process and hence takes up a large amount of time to execute even for smaller datasets. Moreover, there is no Bundle Adjustment taking place and hence, the initializations do not get refined leading to poor reconstruction.

III. Methods

This section describes the end-to-end Structure From Motion pipeline that was implemented for this project.

A. Dataset

The Structure From Motion (SFM) pipeline was run on two datasets. The first was the 'Temple Ring' dataset obtained from Middlebury mview datasets and the second was the 'Viking' dataset which was taken from a custom dataset taken from the TUM dataset. The Middlebury dataset is a widely used benchmark dataset for evaluating computer vision algorithms. Daniel Scharstein and Richard Szeliski first introduced it in 2001, and since then it has developed into a standard dataset for evaluating and comparing various algorithms. It consists of various subsets of datasets for stereo images, optical flow, multi-view geometry, disparity maps, etc. The Temple Ring dataset that has been used by us is taken from Middlebury's 'mview' dataset which is specifically designed for evaluating multi-view stereo geometry algorithms. This dataset is a reproduction of the ”Temple of the Dioskouroi” in Agrigento, Sicily. It consists of 46 image sequences that are 640x480 each. The ’Viking’ dataset, on the other hand, is a sequence of 49 images each of which has a resolution of 504x672. For ensuring good matches and lesser outliers, all the images in the aforementioned datasets have a black or dark background.

B. *Feature Detection and Matching*

Feature detection and matching begins with choosing a detector. We have chosen the SIFT detector for our SFM pipeline as it is robust to changes in illumination, viewpoints, and occlusion which makes it suitable for a wide variety of applications. SIFT detects features and returns feature descriptors for an image where each descriptor is a 128-dimensional vector. The next step is ’Feature Matching’. For this task, the matcher that we used was ’BFMatcher’ which is a Brute-Force Matching algorithm for matching keypoint descriptors between two images. The BFMatcher is a simple and versatile algorithm that is compatible with a variety of feature descriptors such as SIFT, SURF, ORB, etc. It is robust to noisy and ambiguous features and is highly efficient for small datasets, which is exactly the kind of data that we are working with. This makes it faster than other more complex matchers. It uses the K-Nearest Neighbors’ (KNN) algorithm for computing the distances between descriptors. By default, the Opencv kNNMatch() function uses Euclidean distance as a distance metric which is given as follows:

$$distance = \sqrt{\sum_{i=1}^{n}(d_{2,i} - d_{1,i})^2} \tag{1}$$

where n is the number of dimensions in the descriptor, $d_{1,i}$ and $d_{2,i}$ are the i-th elements of the two descriptors being compared.

However, other distance metrics such as Manhattan distance, Hamming distance, etc. can also be used. The top two closest descriptors are then considered for the ’Lowe’s ratio test’. Lowe’s ratio, which is the ratio of the distance of the closest descriptor to that of the second closest descriptor, was considered to be 0.7. Thereafter, images that have less than 20 matches are considered as non-usable and hence rejected. After removing the outliers, the ’Fundamental Matrix’ was computed which finds transformations between two images given points in those two images. The computation of the fundamental matrix is aided by the RANSAC algorithm is used for removing the outlier keypoints. In this manner, we reduce the number of erronous 3D reconstructions and the total computation time of the entire pipeline by ignoring wasteful pose estimations and 3D reconstructions.

C. *Reconstruction*

Reconstruction is the process of generating 3D model of a scene from a set of 2D images. It involves estimating 3D points in a scene as well as the corresponding camera parameters for each image. Here, the reconstruction process

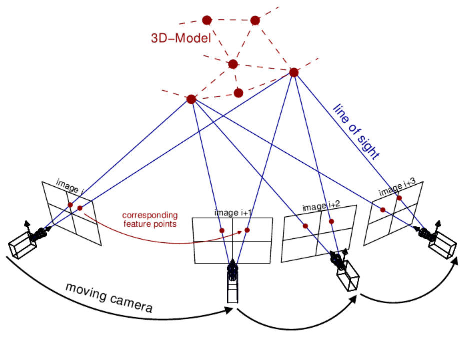

Fig. 2: Triangulation

starts by first computing the camera pose for the initial pair of images or the baseline images. The images that have the greatest difference rotation are chosen as the baseline images as this ensures that the baseline is a stable one. Once the baseline images are determined, the Essential Matrix is computed using the

$$E = K_2^{-T} F K_1 \tag{2}$$

where E is the Essential matrix, F is the Fundamental matrix, and K_1 and K_2 are the intrinsic camera matrices corresponding to the two images.

Using this matrix, the camera pose for the baseline images is computed using triangulation. The method used for this is the Direct Linear Transform (DLT). Once this is done, the reconstruction is augmented and refined further for all the other images.

The reconstruction uses PnP algorithm for computing the relative camera poses. During this entire process, the number of inliers returned may be less or the triangulation error may be very high. To handle this, Bundle Adjustment (BA) is used. Bundle Adjustment is the process of refining 3D coordinates of the scene and the camera parameters and is therefore used as the standard algorithm for minimizing the reprojection error. A high reprojection may be traced back to errors during the initialization step which occur due to measurement errors, image noise and occlusions. Bundle Adjustment is useful for refining these estimates and hence reducing the reprojection error. In the bundle adjustment step, a cost function is computed using camera parameters, number of resected cameras, number of triangulated points, indices of cameras viewing each 2D observation, 3D - 2D point correspondences and the camera intrinsic matrix. This function is typically expressed as a least squares problem and Bundle Adjustment tries to minimize this function in order to refine the camera pose and estiimated 3D point positions.

D. *Creating and Visualizing Point Cloud*

Once all the 3D point positions have been computed, a point cloud is created using all the 3D points. The ’open3d’ library provides functions for creating a point cloud from 3D points and then saving it.

To visualize the generated point cloud, we used a third-party visualization software - MeshLab - which is a 3D mesh processing software system that is suited for processing large meshes and provides a host of tools for editing, rendering, converting, and texturing meshes.

IV. Experiments and Results

A. *Comparison of Performance of Feature Descriptors and Matching*

To achieve the first objective of our project, we tested the performance of three commonly used feature descriptors - SIFT, BRISK, and ORB - and their matching capabilities on 46 images from the TempleRing set (Middlebury dataset). The comparison metrics used in our experiments were computation time, number of keypoints detected, and the number of matches possible.

The experiment was conducted on a single image to test the keypoint detection capabilities of the different methods. Results were recorded in Table.I on the image shown in Fig.**??**.

TABLE I: Output comparison of feature descriptors on one image

Descriptor	Computation Time (ms)	Number of Keypoints
SIFT	44.246	787
BRISK	9.019	636
ORB	3.943	1000

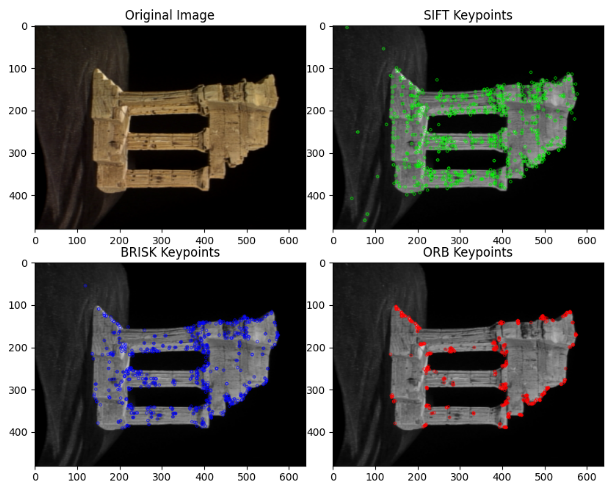

Fig. 3: Keypoints from SIFT(2), BRISK(3) and ORB(4)

Brute force matching was then done on a pair of images to test matchability / utilizability of acquired keypoints. The same experiment was then run on all the 46 images from the TempleRing set, to find overall computation time. The results of the matching have been summarized in Table.II with the matched images shown in Fig.4,5,6.

Based on the results provided in the two tables, we can observe that ORB performed the best in terms of both computation time and number of keypoints detected. SIFT performed moderately well in terms of computation time and number of keypoints detected, while BRISK was the fastest but generated the least number of keypoints.

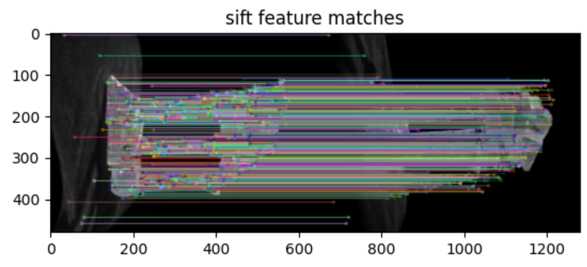

Fig. 4: Matches using SIFT keypoints.

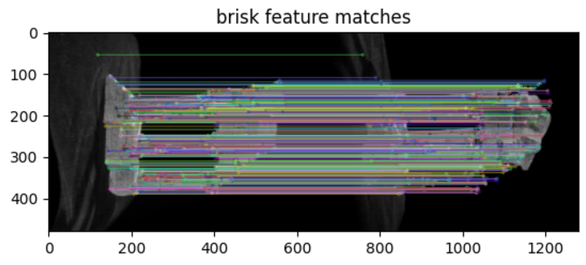

Fig. 5: Matches using BRISK keypoints.

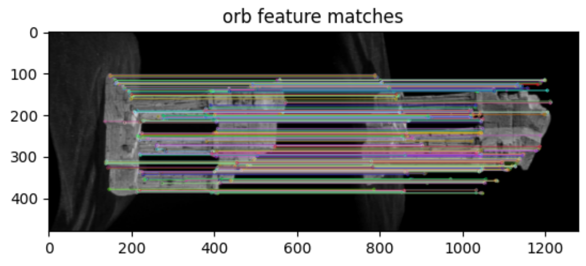

Fig. 6: Matches using ORB keypoints.

Regarding the matching performance, we can see that ORB again outperformed the other two descriptors with the highest number of matches and the shortest computation time. SIFT was the second fastest and generated a moderate but more accurate number of matches, while BRISK was the slowest and generated the least number of matches among the three feature descriptors.

TABLE II: Output Comparison of matching for a pair of images.

Descriptor	Computation Time (ms)	Number of Matches
SIFT	2.58	776
BRISK	4.35	695
ORB	2.97	1000

B. *Objective 2: Implementation of an Incremental SfM Pipeline*

For the second objective of our project, we implemented an incremental SfM pipeline using SIFT as the feature descriptor. We ran the pipeline on two different datasets, each with

TABLE III: Performance of incremental SfM pipeline on different datasets

Dataset	Computation Time (s)	Number of Images	Image Size	Average Number of Keypoints
templeRing	256.322	46	640x480	833.0652
Viking	379.209	49	504x672	1215.9796

varying numbers and sizes of images. The metrics used to evaluate the performance of our pipeline were computation time, number of images, size of images, and visual sparseness of the resulting 3D point cloud.

Before the implementation of this pipeline however, we were able to test the performance of some other SfM implementations available online that used all possible image pairs for reconstruction. These took The following table II summarizes the results of our pipeline.

From the table, we can observe that the computation time of our incremental SfM pipeline increased with the number and size of images. However, the visual sparseness of the resulting 3D point cloud decreased with the number of images, indicating that our pipeline was able to generate denser point clouds as the number of input images increased. Image size also played a factor in increasing the time taken proportionately.

However, where our initial implementations took 30 minutes for a working on a set of 10 images, our final version with the reduced image pairs used and with bundle adjustment is vastly faster and gives better results. The results have been shown in Fig.7 and Fig.8. The point clouds have been visualized using Meshlab for better in inscrutability.

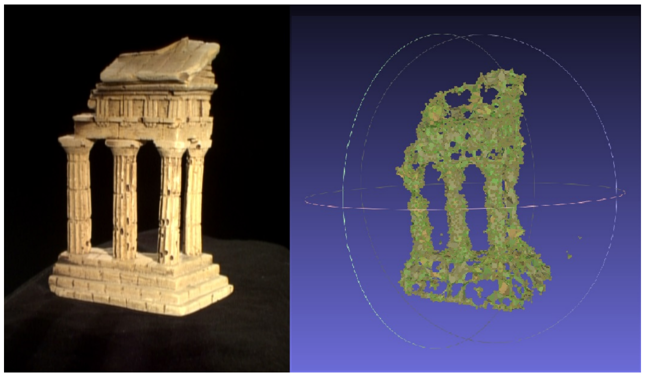

Fig. 7: TempleRing dataset.

V. Discussion and Summary

Our experiments first compared the performance of SIFT, BRISK and ORB feature descriptors for image-based reconstruction. It then involved the implementations of incremental 'Structure from Motion.' From the results, the following insights were obtained.

1) The choice of feature descriptor depends on the specific requirements of the application. ORB may be best for speed and a large number of matches, while SIFT may be more suitable for accuracy. BRISK may be useful for speed but with less detail.

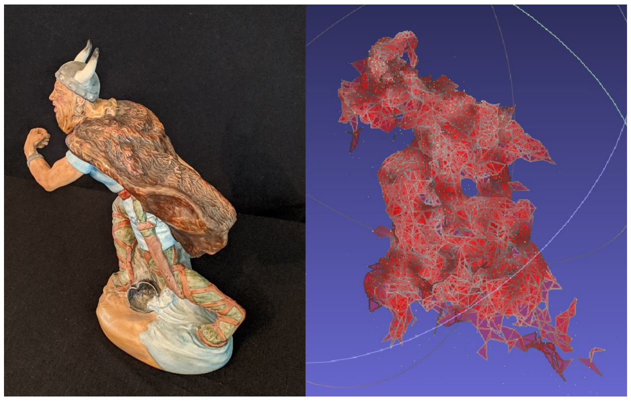

Fig. 8: Viking dataset.

2) Outlier removal with RANSAC and bundle adjustment the obtained 3D point clouds were of better shape and more resembled the actual object. By limiting the combinations of views used, it was possible to speed it up, however, this speed would not reach a LiDAR implementation.
3) The application of deep learning to the SfM problem, titled DeepSfM has been introduced and boasts much faster reconstruction times. This paved the way for real-time environment mapping without the requirement for LiDARs.
4) The formation of 3D objects from point clouds is a possible application of the pipeline implemented here. These objects can then be used in augmented reality applications

For an efficient implementation of SfM, the choice of feature descriptors and matches are not the only governing factors. Outlier detection and removal, pose estimation, reconstruction and bundling are also essential components whose variations that can be explored as possible future work.

VI. Conclusion

Our project demonstrated that the choice of feature descriptor is critical for image-based 3D reconstruction. ORB is best for speed and generating a large number of matches, while SIFT is more suitable for accuracy. BRISK may be useful for speed but with less detail. Our incremental SfM pipeline using SIFT showed promising results for 3D reconstruction of objects from videos, with potential applications in architecture, engineering, and entertainment. The improvement of speed and accuracy are incredibly apparent with the addition of bundle adjustment and removal of outliers. Further work could optimize the pipeline for larger datasets and explore the performance of other feature descriptors.

References

[1] Liu, W., Wang, S., Deng, Z., Fan, T., Jia, M., Chen, H. ”A Review of Image Feature Descriptors in Visual Positioning.” IPIN-WiP (2021).

[2] Schonberger, Johannes L., and Jan-Michael Frahm. ”Structure-from-motion revisited.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[3] Schönberger, Johannes L., et al. ”Pixelwise view selection for unstructured multi-view stereo.” Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14. Springer International Publishing, 2016.

[4] Wu, C., 2013, June. Towards linear-time incremental structure from motion. In 2013 International Conference on 3D Vision-3DV 2013 (pp. 127-134). IEEE.

[5] Harish V. (2021, April 27). Structure-from-Motion. GitHub. https://github.com/harish-vnkt/structure-from-motion

[6] Institute of Communications Technology. (n.d.). Motion Estimation. Leibniz University Hannover. Retrieved April 27, 2021, from http://www.tnt.uni-hannover.de/project/motionestimation/

[7] Adil M., Hussain A. (2021, April 27). How to SFM. GitHub. https://github.com/muneebaadil/how-to-sfm