Buildings Built in Minutes

Kirti Kishore	Advait Kinikar	Jason Chen
120148286	120097997	116605403
Group 15	Group 15	Group 15
Section 0201	Section 0201	Section RO01
kiki1@umd.edu	kinikara@umd.edu	jchen777@umd.edu

Abstract

This report will first review the classical structure from motion pipeline. This includes the key concepts such as finding keypoints and descriptors using feature detection algorithm, matching the features and finding correspondences while rejecting the outliers, estimating the essential matrix from the fundamental matrix, using the essential matrix to get the camera pose, triangulation of 3D points, Bundle Adjustment to refine the 3D point, minimizing errors, generating point clouds that bears a resemblance to the actual structure etc. Then, using this solid foundation, we will attempt to to make a hyper-realistic 3D model of an environment in our image dataset.

1. Introduction

1.1. Background

Structure from motion (hereby, SfM) or also known as Tomasi-Kanade Factorization, aims to construct a more useful 3D structure using images taken of common landmarks and environmental features from different camera poses. SfM have have seen significant advancements since its early pioneering in the mid-1990s [6]. This tool is truly powerful because it does not need to rely on more complex hardware such as LIDAR sensors to reconstruct a similarly complex environment; the only restrictions are computing power, time, and the efficiency of the algorithm, something we found out the hard way.

Current capabilities involve compiling any number of unordered images from the internet or another large image database to reconstruct a detailed 3D point cloud map of the environment captured within the confines of the 2D images. A larger sample of images enable better optimization of the reconstruction model. Despite these leaps, the current framework still retain many core components involved in classical 3D reconstruction from 2D images. These core steps will be outlined in this report and broadly include image acquisition, correspondence search, and reconstruction. Finally, these steps will be applied towards the 3D reconstruction of selected images.

1.2. Current Applications

There are a wide variety of uses for SfM algorithms that have spurred innovation in this field. Of course, there are key uses in robotic navigation and especially in the field of SLAM, but this technology's versatility extends to virtual reality, archaeological and architectural modeling, and world mapping (*e.g.* Google Maps).

An example in archaeological modeling is demonstrated in the huge undertaking by researchers from the University of Washington, who aimed to reconstruct the entire city of Rome in less than a day using more than one hundred thousand images found on online image database Flickr [1]. This paper's work pales in comparison, but aims to allow a glance into how the feat was possible.

2. Algorithm Pipeline

Figure 1. This is one of the six representative images that was used to reconstruct the building.

2.1. Image Processing

A representative image can be seen in Fig. 1. Code was borrowed from Project 3 to undistort the images and to ob-

tain the the camera intrinsic matrix, K. Histogram equalization was applied to the gray scale-converted images to normalize the images prior to use.

2.2. SIFT and Brute Force Matching

SIFT stands for scale-invariant feature transform and is a method of reliably isolating the same features between images regardless of scale, rotation, and translation, making it a valuable tool in the first step of the pipeline. Feature points and descriptors are obtained for each image using the SIFT function from OpenCV [3]. We opted for SIFT over ORB for two main reasons: familiarity and accuracy. Because the library of images is limited in size, the performance decrease by ORB does not outweigh the decreased processing power required to use ORB [4]. Furthermore, having covered SIFT in class, there was a better understanding of how it worked behind the scenes.

These feature points are then saved to text files that can then be read from and compare using K-Nearest Neighbor (hereby, KNN) matching. For each pair of images, all the feature points will be compared to each other to find corresponding. According to the OpenCV website, each feature and descriptor will be matched using L2 norm distance calculations and close points are returned as correspondences or pairs of feature points that are candidates for being the same point in global 3D coordinates. KNN was used over FLANN and brute force due to ease of implementation with Lowe's ratio method (0.75), otherwise brute force would have sufficed for such a small image sample size [3]. FLANN is more ideal for larger sample sizes for more complex reconstruction. Fig 2 show the correspondences as a line connecting the pair of candidate feature points.

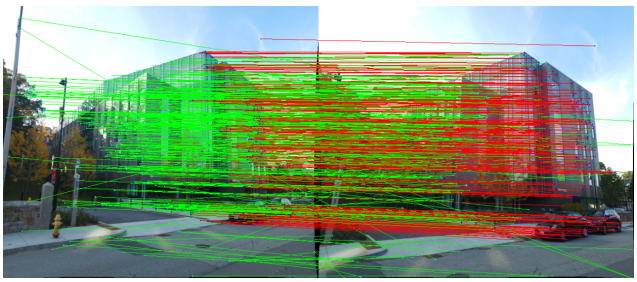

Figure 2. Correspondence points between two images that were accepted and rejected, in green and red, respectively. This image is purely for visualization purposes.

2.3. RANSAC and the Fundamental and Essential Matrices

```
[[-2.93836855e-07 -1.05473087e-05  2.87067973e-03]
 [ 1.32634480e-05 -1.00302328e-06 -3.44127464e-03]
 [-4.76649107e-03  1.25570784e-03  1.00000000e+00]]
[[-0.01499204 -0.67149611 -0.26166169]
 [ 0.8431948  -0.06922004  0.51307256]
 [ 0.15827746 -0.67332178 -0.15157522]]
```

Figure 3. Fundamental and essential matrices.

RANSAC stands for random sample consensus and is a method of identifying a near-best model for a population with many potential outliers RANSAC is crucial because every step of this pipeline, starting from SIFT feature recognition, accumulates error and can produce outliers [5]. By taking the model with the most sampled inliers, one can then identify those points and go on perform more accurate calculations. In this report, RANSAC will be used to both fit a fundamental matrix F to the population of correspondences, but also fit the camera poses to the population as well to obtain the optimal camera poses.

RANSAC will be applied to sample all of the initial correspondence candidates between each unique pairing of images from the online databases. Then samples of at least eight points are used to solve the equation (1) derived from the epipolar constraint in (2). We solve for F using single value decomposition (hereby, SVD), which also exists as an OpenCV function [3]. Ideally, the result would be the fundamental matrix of rank 2, F.

$$\mathbf{A}\mathbf{x} = \mathbf{0} \tag{1}$$

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = \mathbf{0} \tag{2}$$

$$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

However, F often has a rank of 3 due to error and noise in the feature point correspondences, so it may require enforcement to be rank of 2 by replacing the singular value in the third column with zero. This F matrix can then be used to calculate the essential matrix, E, using the camera intrinsic matrix, K, obtained from image calibration or from manufacturer, shown in (3). Please refer to Fig. 3.

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K} \tag{3}$$

2.4. Triangulation

There are many alignment steps in the classical SfM process, and Triangulation is a crucial one in ensuring the integrity of the final 3D reconstruction [6]. Every skipped step may lead to many feature points being mismatched and potentially removed, resulting in a less robust or a sparser 3D point cloud.

Triangulation is an ancient process in the history of computer vision and involves localizing a point in 3D space using two or more camera views, or in this case, two or more

images taken at different perspectives. This technique provides further robustness in the 3D reconstruction by ensuring at least two different images project their shared correspondence feature point to the same point in the global 3D space. Please refer to Figs. 4 and 5.

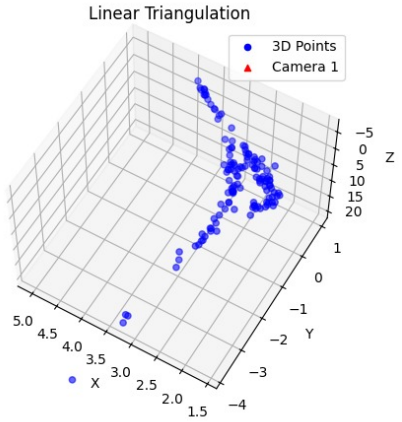

Figure 4. The optimized set of global coordinate points produced by linear triangulation plotted in Matplotlib.

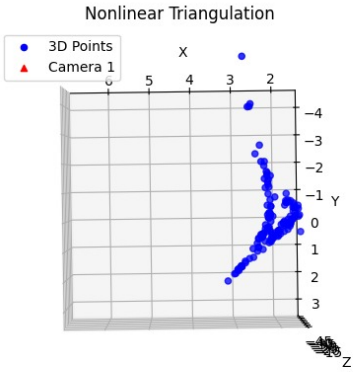

Figure 5. The optimized set of global coordinate points produced by non-linear triangulation plotted in Matplotlib.

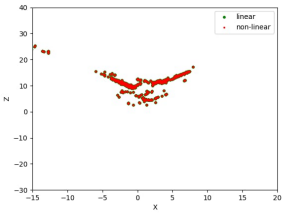

Figure 6. 2D plot of Linear and Non-Linear Triangulation for comparison

The above figure, Figure 6., shows the comparison between linear and non-linear triangulation when plotted in 2D.

2.4.1 Chirality Condition in Triangulation

Following the triangulation process, it is imperative to verify the chirality condition to ensure that each triangulated 3D point lies in front of each camera that captured it. This step is critical as it confirms the physical feasibility of the reconstructed points. The chirality condition is defined mathematically as:

$$\mathbf{R}_i(\mathbf{X} - \mathbf{t}_i)_Z > 0$$

where $\mathbf{R}_i$ is the rotation matrix, $\mathbf{t}_i$ is the translation vector of camera i, and $\mathbf{X}$ is the 3D point in question. This condition checks if the Z-component of the transformed point $\mathbf{X}$ in the camera coordinate system is positive, indicating that the point is in front of the camera.

Implementing and verifying the chirality condition is crucial for maintaining the integrity of the 3D model by ensuring all points are correctly positioned relative to each camera's perspective, thus preventing any erroneous reconstructions or mismatches in the structure. Failure to meet the chirality condition often leads to discarding or re-estimating the 3D points.

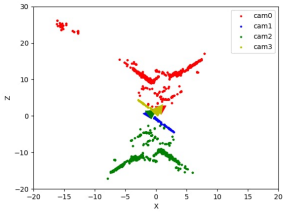

Figure 7. Output after checking chirality condition

2.5. Camera Direction

The process of estimating camera direction involves computing the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between two camera positions. This is typically achieved by performing Singular Value Decomposition (SVD) on the Essential matrix $\mathbf{E}$. The SVD of $\mathbf{E}$ results in two orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$, and these matrices are utilized along with a predefined correction matrix $\mathbf{W}$ to determine possible camera rotations and translations. The matrix $\mathbf{W}$ is defined as:

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$, we derive four possible configurations for the camera's rotation and translation as follows:

$$
\begin{aligned}
C_1 &= \mathbf{U}(:,3), & R_1 &= \mathbf{U}\mathbf{W}\mathbf{V}^T \\
C_2 &= -\mathbf{U}(:,3), & R_2 &= \mathbf{U}\mathbf{W}\mathbf{V}^T \\
C_3 &= \mathbf{U}(:,3), & R_3 &= \mathbf{U}\mathbf{W}^T\mathbf{V}^T \\
C_4 &= -\mathbf{U}(:,3), & R_4 &= \mathbf{U}\mathbf{W}^T
\end{aligned} \tag{4}
$$

Each pair (R_i, C_i) represents a potential solution for the camera orientation and position, reflecting different assumptions about the direction of the translation vector and the rotation of the camera between the two views.

2.6. Perspective-n-Points (PnP)

Once we have estimates for the camera orientations and positions, the next step in the 3D reconstruction process is to map how accurately these cameras can project 3D points in the world onto their 2D image planes. This problem is known as the Perspective-n-Points (PnP) problem, which can be approached using the projection equation:

$$
\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{5}
$$

Here, $\mathbf{K}$ is the intrinsic matrix of the camera, $\mathbf{R}$ is the rotation matrix, and $\mathbf{t}$ is the translation vector, all combining to map points (X_i, Y_i, Z_i) from the world coordinates to (x_i, y_i) on the image plane.

The Direct Linear Transform (DLT) method is often the first approach used to solve the PnP problem due to its straightforward implementation. It requires at least six point correspondences between the 3D world coordinates and their 2D image projections. Although simple, DLT can be very sensitive to outlier data points.

To improve the robustness against outliers, the RANSAC algorithm is frequently employed. RANSAC repeatedly selects random subsets of the point correspondences to estimate the camera pose and evaluates how well these subsets fit the model. The best fit, which maximizes the number of inliers (data points well explained by the model), is then chosen as the solution.

2.7. Bundle Adjustment

Bundle adjustment, (hereby, BA) is the simultaneous adjustment of both the camera pose and their associated global coordinates to minimize the total error and find the best overall poses and alignments [2]. This helps eliminate any accumulated error from triangulation and PnP. Please refer to Fig. 8 BA continues to be the bottleneck of the structure from motion pipeline and is a key step that should not be skipped.

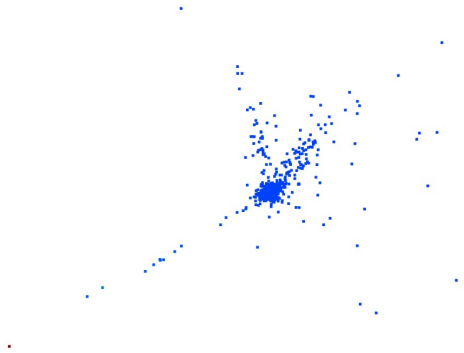

Figure 8. Our final output result after bundle adjustment, the last step of the pipeline. This was produced using Open3D, which we realized too late that it proved better for visualizing 3D point clouds. The above image can be inferred as the top-view of the buildings corner. If you zoom in the purple dot is our camera.

3. Discussion

OpenCV SIFT function was extremely effective in picking out feature points, but the KNN function from penCV combined with Lowe's ratio was very aggressive in the sense a lot of points were removed post-matching. It would have been fine to use BFmatcher, but due to the necessity of Lowe's ratio test, KNN was the best method given those conditions. Moving on to triangulation, PnP, and BA with less than a few hundred key points often caused out code to throw errors as they are reduced to only a few dozen at most towards the end of the pipeline. There appears to be inefficiencies in the algorithm combined with sub-optimal image selection.

Please refer to Figs. 4 and 5. Though it is not the clearest, there is a distinct corner pattern that is indicative of the image that was used for 3D reconstruction. triangulation helps localize the points in space using the different image views, but the sparsity of points made it difficult to troubleshoot potential issues. It is a huge regret is that our algorithm is not refined enough to offer not 3D details that would be more indicative of the structure being imaged.

Due to poor visualization by our algorithm, it was difficult to confirm the camera orientations beyond mathematics and thus may have contributed to some ambiguity in the results obtained later on.

Please refer to Fig. 8. This was a key step applied to to the feature points that made it the most distinguishable representation of the 2D images that we used for the reconstruction pipeline. Though it is far from optimal, it once again has the distinct geometric pattern that bears some, albeit somewhat minor, resemblance to the 2D images we used. Open3D proved to be a step up in being able to visualize and observe the data from different perspective more easily compared to Matplotlib. Matplotlib had a real issue

with zooming away from the data points of interest very easily such that it became hard to visualize.

Our team tried other online images as well, such as a bird model and a cathedral. The results were poor, likely due to those online images having too much noise and differences in lighting, contrast, camera intrinsic matrices, etc, that could not be made up for in the calibration and normalization processes. In general, our results may benefit greatly from better pre-processing prior to input into pipeline.

3.1. Challenges

The biggest challenges during the reconstruction was usually not any particular step as OpenCV has an extensive library of functions that empowers even the most feeble-minded people like us to convert 2D image feature key points to a 3D point cloud. However, what cannot be simplified is array indexing and data management, especially with applying the Lowe's ratio and ensuring the same keypoints were not counted more than once.

There were also issues with a large number keypoint drop offs between steps such as triangulation and and PnP that made the end result have far fewer images. At some points there the remaining keypoints were so sparse that certain functions could not be performed due to lack of points to constrain the systems of equations properly.

3.2. Next Steps

The ideal next step would be to optimize the pipeline algorithm so it can run a larger set of sample images to construct a clearer 3D point cloud more akin to the work by the team behind building Rome in one day. At the very least, there will be more data points for us to distinguish the effectiveness of our pipeline. Though it was a good decision to break up the feature point sampling and matching so the core data points are maintained constant when the remaining code can be worked on and improved.

4. Conclusion

Given the wide adoption of the internet, a trove of images exists online that enables the reconstruction, discovery, and manipulation of a wide variety of natural and man-made structures. This report tries to give teeny tiny glance into the wide range of possibilities enabled by structure from motion even though there were great limitations with our algorithm and hardware in being able to handle complex matrices and large datasets. There is no doubt a lot of room be improvement.

As advancements in silicon and computer vision research is made, 3D reconstruction will become faster and more robust, enabling better integration with not only robotic applications like SLAM, but other components of our lives. This is truly exciting as we are in an age were artificial intelligence and VR/AR are both poised to change how one interacts with the environment. This is but a tiny step in blurring the lines between reality and the 2-dimensional world.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, oct 2011.

[2] Y. Chen, Y. Chen, and G. Wang. Bundle adjustment revisited. *CoRR*, abs/1912.03858, 2019.

[3] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[4] E. Karami, S. Prasad, and M. S. Shehata. Image matching using sift, surf, BRIEF and ORB: performance comparison for distorted images. *CoRR*, abs/1710.02726, 2017.

[5] A. Z. Richard Hartley. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd ed edition, 2003.

[6] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.