Contents

List of Figures

Executive Summary

This work aims to employ incremental SfM. Incremental SfM involves the four main stages namely, (1) Extraction and Matching, (2) Geometric Verification, (3) Reconstruction, and (4) Adjustment. The pipeline for building an SfM application is as follows:

1. **Feature Extraction:** The method should generate a tensor track that defines the matches across various views given a set of images using SIFT and Nearest Neighbours Algorithm.
2. **Estimating the Camera Pose:** Uses an essential matrix, the function computes and finds four possible camera configurations, then chooses the optimal one based on cheirality condition.
3. **Perspective-n-Point algorithm:** The function should register a new picture utilizing 3D-2D correspondences using a linear perspective-n-point approach with RANSAC given a 3D reconstruction of points.
4. **Reconstructing 3D Points:** The function will reconstruct 3D points that have not previously been reconstructed given a newly registered image.
5. **Running Bundle adjustment:** This function will use nonlinear least squares optimization to refine the camera pose and reconstructed points while minimizing re-projection errors.

This method is robust, however there are could be some scalability issues for larger data sets.

1 Abstract

This project aims to develop a pipeline for incremental Structure from Motion. The aim is to generate a 3-Dimensional worldview from a collection of images/video frames. At first, similar set of features between the images are identified. Then, the camera poses along with matched 3D points in the world frame are calculated. Lastly, all the results are merged and outliers are filtered to give an overall 3-Dimensional understanding of the scene.

2 Introduction

Understanding the geometry of the scene is essential for various problems in autonomous mobile robotics. One such challenge for an autonomous mobile robot is to perceive an unstructured environment. The perception and 3D reconstruction of a scene can be achieved through a visual technique called Structure from motion. Structure from motion (SfM) is the process of estimating the 3D structure of a scene from a set of 2D images taken from different viewpoints. SfM is used in many applications, such as 3D scanning, augmented reality (AR), and visual simultaneous localization and mapping (vSLAM). This technique can be combined with data from other peripheral robotic hardware for other various applications.

3 Background & Significance

The groundbreaking work in [2] first introduced the point-based linear correspondences approach to solve SfM, which came to be formally known as the eight point algorithm. In particular, this problem encapsulates two aspects: The first is resolving a 3D structure from a set of overlapping frames that have an offset, and the latter being estimation of the relative camera pose giving the rotation and translation between frames. This method does not require a-prioris and is solved with a highly redundant bundle adjustment method by extracting relevant features from overlapping images [3].

The images are supplied in a certain order, such as the frames of a video or photographs recorded by a robot traversing around a room, in one frequent multi-image situation. If corresponding points between consecutive images can be matched, a simple way to reconstruct a sequence is to use the eight-point algorithm to infer a relation to recover the poses of the first two frames by triangulating their common points, then resection the next frame, triangulate any new points, and repeat: a form of visual ground truthing. This approach can work effectively with properly-calibrated cameras, especially on short image sequences. The method of incrementally growing the reconstruction with each image input is the standard and robust approach [4] which essentially is explored in this work.

4 Methodology

This work aims to employ incremental SfM. The pipeline followed for this work was heavily inspired from Multiview Geometry courses offered by [5], [1], and [6]. Incremental SfM involves the four main stages namely, (1) Extraction and Matching, (2) Geometric Verification, (3) Reconstruction, and (4) Adjustment. The pipeline for building an SfM application is as follows:

Structure from Motion Algorithm

Camera Calibration
Build feature track
Estimate first two camer poses
Initialize pose set P
for i = 2, $\cdots, N-1$**do**
  Estimate new camera pose
  P = P $\bigcup P_i$
  for j $<$ i **do**
    Find new points to reconstruct
    Triangulate point
    Filter out point based on cheirality
    Update 3D point
  end for
  Run bundle adjustment
end for

Although, this method is robust, there are some scalability issues for large data sets.

4.1 Calibration

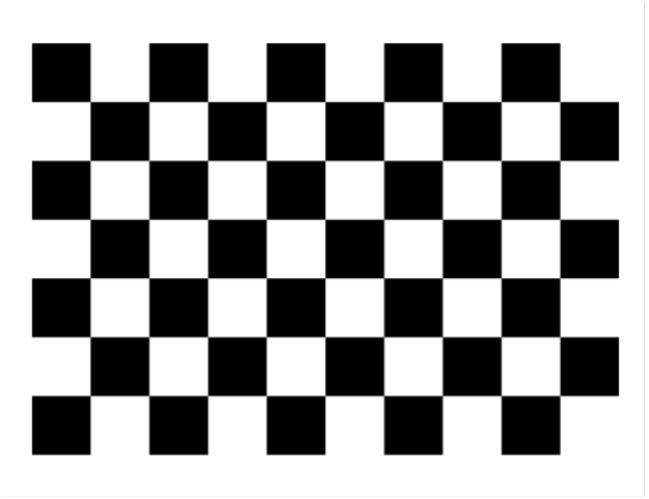
The very first step is camera calibration. It is done to account for the intrinsic parameters of the camera (the visual sensor). The goal is determine a camera intrinsic matrix $K \in \mathbb{R}^3$. This matrix is as follows,

$$\mathsf{K} = \begin{bmatrix} f_x & S & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

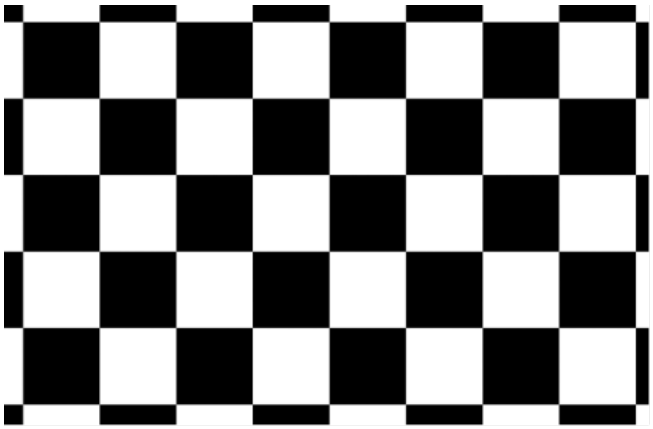
Here f_x and f_y are the x and y focal lengths. C_x and C_y are the x and y coordinates of the optical center in image plane. S is considered to be the Skew between the axes. In most cases skew is considered to be 0 for simplicity in calculations.

$\mathbf{Steps}$: Camera Calibration

1. A given chess board is printed to scale (see Figure 1a).
2. Omit the outer border and use the Inner chessboard pattern (see Figure 1b). It has 9 columns and 6 rows. These values are to be updated in the script.
3. Measure the actual checkerboard size, updating it in the same script.
4. Take in a large set (around 60 images from different angles).



(a) Checkerboard pattern for camera calibration



(b) Inner chessboard pattern

Figure 1: Camera calibration input

4.2 Feature extraction and Matching

The Scale Invariant Feature Transform (SIFT) was opted as a feature detection algorithm to identify the local features i.e. keypoints, in a particular image. These keypoints are scale rotation invariant and that is the reason why SIFT features are preferred over edge features or hog features. Next, feature matching algorithm was performed to create the matches between the keypoints detected in the two images.

All of these matches aren't correct. There exists some outlier matches that should be neglected. One of the reasons for these outlier matches is the noise in the images. Images of a scene, having small distance between their camera centers will have relatively lesser outlier matches unless there is a sharp change in brightness, texture or shading in the images. To remove these outliers, the essential matrix was computed using the 8 point algorithm and then the RANSAC algorithm was applied to obtain a better estimate of the essential matrix (E). This will help have a much better idea of the inlier correspondences.

Essential Matrix is a 3 × 3 matrix which encapsulates the epipolar geometry. It helps to determine the epilpolar line in the second view for a given point in the first view.

$$E * x = l'$$

In other words if a point in 3 dimensional space X is imaged as x in the first view, and x in the second, then the image points satisfy the relation

$$x^T * E * x' = 0$$

This is known as epipolar constraint. An 8-point algorithm was used to estimate the essential matrix. Now in order to get a robust E RANSAC algorithm was utilized. So, out of all possibilities, the E matrix with maximum number of inliers is chosen.

Next step is to build a feature track that lists the matches across the different views. The track is of size R^{N*F*2}, where N is the total images and F is the number of valid features. Initially F would be the sum of all features found. The track is filled as follows:

$$track[j, feature, :] = x_j \quad if\ x_i == x_j$$
$$track[j, feature, :] = -1 \quad if\ x_i! = x_j$$

The above equation describes that if a feature in i^{th} image is matched to the j^{th} image then it is stored in the $track[j, feature, :]$. The track is build in each iteration and then add them to the full track. The columns of unmatched features are removed using a mask.

Based on the above process we can follow the below pseudo code:

Build Feature Track

---

for i = 0, $\cdots, N-1$**do**
  Extracting SIFT Descriptor of the i^{th} image
end for
for i = 0, $\cdots, N-1$**do**
  Initialize $track_i = -1^{NxFx2}$
  for j = i+1, $\cdots, N-1$**do**
    Matching features between 2 images and normalize it by multiplying the inverse of intrinsic
    Normalize it by multiplying the inverse of intrinsic
    Now estimate the inlier matches using essential matrix
    Update the track i utilizing the inlier matches
  end for
  Remove unmatched features
  Update the main track by: $track_full = track_fullUtrack_i$
end for

4.3 Relative Camera Pose Estimation

The next step is to extract the camera pose. At first the Essential matrix is used to estimate the camera pose. However there exists a projective ambiguity in retrieving the camera pose from the Essential Matrix. This is because there are four possible solutions for a camera pose matrix , $P \in 3 \times 4$.These four possible configurations are for 2 sets of Rotation and 2 sets of Position Matrices. All four possible configurations are shown below.

$$P := R = UWV^T, C = U(:, 3)$$
$$P := R = UWV^T, C = -U(:, 3)$$
$$P := R = UW^TV^T, C = U(:, 3)$$
$$P := R = UW^TV^T, C = -U(:, 3)$$

$$\text{where },W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

It is to be noted that the $det(R) > 0$, if for any case $det(R) < 0$ the pose of the camera must be inverted, i.e. $R = -R$ and $C = -C$.

In order to find the correct camera, pose from the four possible configurations (see **??**) the cheirality condition should be satisfied. This means that the reconstructed points must lie in front of the camera. However, to reconstruct points in 3D space, they need to be triangulated at first.

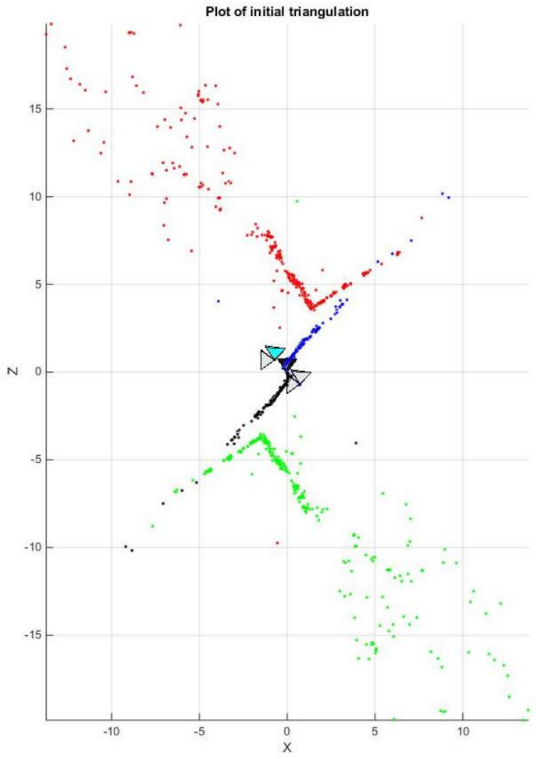


Figure 2: The four possible poses for the camera [1]

Triangulation is a process of determining a point in space given it projections on two separate image planes. If $\mathbf{X}$ is the same point being projected in *Image Plane 1* and *Image Plane 2* then it should satisfy the relation given below, where, x_1, x_2 are the image points and $P1$, $P2$ are the respective camera pose matrix

$$x = PX \quad \text{and} \quad x' = P'X$$

However, in a physical scenario such relation is not possible due to uncertain factors like lens distortion, noise, camera correction etc. It is entirely possible that the measured points in the image may never intersect for the same 3D point. This problem can be solved by defining the best estimate for "X" given the two image points and camera poses. This point is usually found by minimizing an error function dependent on the estimated 3D point. This project uses a linear Triangulation solution. This method works for any number of corresponding images but is not projectively invariant. The derivation is as shown below,

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} p1 & p2 & p3 & p4 \\ p5 & p6 & p7 & p8 \\ p9 & p10 & p11 & p12 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} & p_1^T & \\ & p_2^T & \\ & p_3^T & \end{bmatrix} \begin{bmatrix} | \\ \mathsf{X} \\ | \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix}$$

Since both the vectors are in same direction their cross-product would be 0

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \times \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \times \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} = \begin{bmatrix} y_1 p_3^T X - p_2^T X \\ p_1^T X - x_1 p_3^T X \\ x_1 p_2^T X - y_1 p_1^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Similarly for $x' = P'X$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \times \begin{bmatrix} p_1'^T X \\ p_2'^T X \\ p_3'^T X \end{bmatrix} = \begin{bmatrix} y_2 p_3'^T X - p_2'^T X \\ p_1'^T X - x_2 p_3'^T X \\ x_2 p_2'^T X - y_2 p_1'^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Stacking them vertically gives us,

$$\big[AX\big] = \begin{bmatrix} y_1 p_3^T X - p_2^T X \\ p_1^T X - x_1 p_3^T X \\ x_1 p_2^T X - y_1 p_1^T X \\ y_2 p_3'^T X - p_2'^T X \\ p_1'^T X - x_2 p_3'^T X \\ x_2 p_2'^T X - y_2 p_1'^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\big[AX\big] = \begin{bmatrix} y_1 p_3^T - p_2^T \\ p_1^T - x_1 p_3^T \\ x_1 p_2^T - y_1 p_1^T \\ y_2 p_3'^T - p_2'^T \\ p_1'^T - x_2 p_3'^T \\ x_2 p_2'^T - y_2 p_1'^T \end{bmatrix} \big[X\big] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This is of the form $AX = 0$. The values of X can be found using Singular Value Decomposition of A.

$$[USV] = SVD(A)$$

$$X = V(:, end)$$

Now that the position of 3D points has been determined we need to check the cheirality condition. It is checked by the sign of the Z element (in camera co-ordinate systems) with respect to the camera. If a 3D point in front of the camera satisfies the condition for a given camera pose P with rotation matrix R and camera pose C,

$$r_3(X - C) > 0, \text{ where } r_3 = R(3, :)$$

The above condition is then applied for the world coordinate from two different points camera poses. The sufficient AND condition i.e. the condition in which the world coordinate lies in front of both the camera poses is applied. The poses that satisfy this condition are then the true camera poses.

4.4 Perspective-n-Points Algorithm

After generating 3D coordinates in world frame for the features, their respective correspondences are associated with the 2D projections on the image using the linear perspective-n-point (PnP) algorithm. Since the camera poses and world coordinates of points have been calculated, the projection these points into the camera image is used to find the 2D pixel points. However, PnP is prone to outliers in the feature matches, thus RANSAC is applied to minimize the reprojection error to filter them out:

$$minimize_p \sum_{i}^{n} ||f(p, \mathbf{X}_i) - b_i||^2$$

p: vectorized camera pose constituting of camera center and rotation matrix converted to quaternion representation

$$p = \begin{bmatrix} C^T & q^T \end{bmatrix}$$

$f(p, \mathbf{X}_i)$: projection error of i^{th} 3D point $\mathbf{X}_i$ onto camera p,

$$f(p, \mathbf{X}_i) = \begin{bmatrix} u_i/w_i \\ v_i/w_i \end{bmatrix}, \text{ where } \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = R \begin{bmatrix} I_3 - C \end{bmatrix} \begin{bmatrix} \mathbf{X}_i \\ 1 \end{bmatrix}$$

Here, $\mathbf{X}_i \in \mathbb{R}^3$ which is the i^{th} 3D point and b_i is the 2D point $\begin{bmatrix} x_i y_i \end{bmatrix}^T$
The Levenberg-Marquardt used to minimizing the reprojection error is given as,

$$p = p + \Delta p,$$
$$\text{where } \Delta p = \left(\sum_{i}^{n} \frac{\partial f_i^T}{\partial p} \frac{\partial f_i}{\partial p} + \lambda I \right)^{-1} \sum_{i}^{n} \frac{\partial f_i^T}{\partial p} (b_i - f_i)$$

Here, the pose Jacobain for the i^{th} point is $\frac{\partial f_i}{\partial p} \in \mathbb{R}^{2\times 7}$. The damping factor λ can generally be chosen between 0 – 10, was selected as 1.

Post this process, reconstruction is performed with the 3D points that hadn't been considered. With triangulation, the newly added set of points are reconstructed as well. The reprojection error is minimized as

$$minimize_{\{\mathbf{X}_j\}} \sum_{k=1}^{2} ||f(p_k, \mathbf{X}_m) - b_{k,j}||^2$$

Here, $\mathbf{X}_j$ represents the j^{th} point in 3D world frame, the k^{th} camera pose is given by p_k, the j^{th} 2D point in the k^{th} image is $b_{k,j}$.

This minimization can again be performed with the Levenberg–Marquardt method by updating the camera pose $\mathbf{X}$

$$\mathbf{X}_j = \mathbf{X}_j + \Delta X_j$$
$$\text{where } \Delta \mathbf{X}_j = \sum_{k=1}^{1} \left(\frac{\partial f_{k,j}^T}{\partial \mathbf{X}} \frac{\partial f_{k,j}}{\partial \mathbf{X}} + \lambda T \right)^{-1} \frac{\partial f_{k,j}^T}{\partial \mathbf{X}} (b_{k,j} - f_{k,j})$$

4.5 Bundle Adjustment

This step of the algorithm is a state estimation technique which is employed after all camera poses and 3D points have been estimated. Using nonlinear least squares optimization, the previously initialized reconstruction is refined by further minimizing reprojection error.

$$minimize_{\{p_k\}\{\mathbf{X}_j\}} s_{k,j} ||f(p_k, \mathbf{X}_j) - b_{k,j}||^2$$

Here, K is the number of images, while J is the number of 3D points.The visibility indicator $s_{k,j} = 0, 1$, is 1 if the j^{th} point is visible in the k^{th} image and 0 otherwise.

Bundle adjustment is the computationally heavy final step of the process that produces the results for reconstruciton. It gives a statistically optimal solution making assumptions of gaussian noise, data associations, etc. The structure of the image is exploited in this process. The sparse nature of the solutions helps solving the large number of linear equations.

5 Results

Camera Calibration The calibration was performed on an Android camera with a set of 50-60 images of the checkerboard. The corner points detected in the checkerboard pattern can be seen in Figure 3

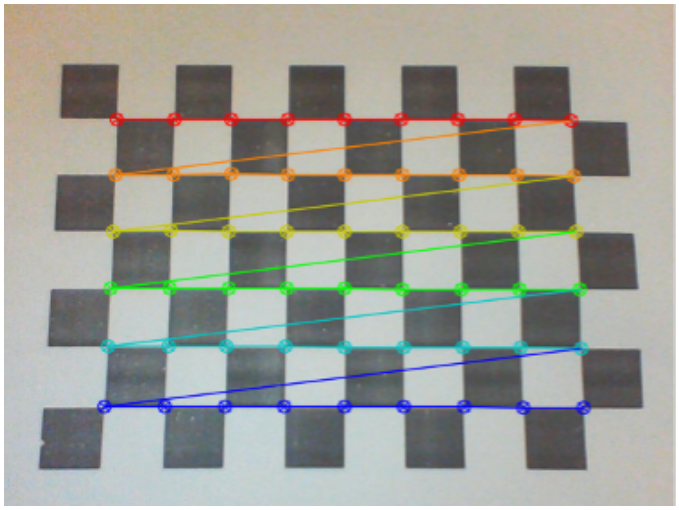


Figure 3: Keypoints detected in camera calibration

The intrinsic parameter obtained from this method is as follows:

$$K = \begin{bmatrix} 3228 & 0 & 600 \\ 0 & 3238 & 450 \\ 0 & 0 & 1 \end{bmatrix}$$

Building Feature Track The feature track are based on the set of images presented in Figure 4.

Figure 4: Set of Images used as input for Structure from Motion

After extracting features through SIFT, following were the results:
found 4487 features in image 1, found 4371 features in image 2,
found 4534 features in image 3, found 4519 features in image 4,
found 4592 features in image 5, found 4685 features in image 6.

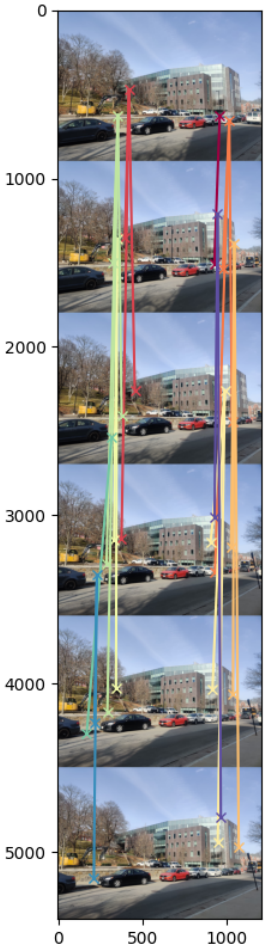
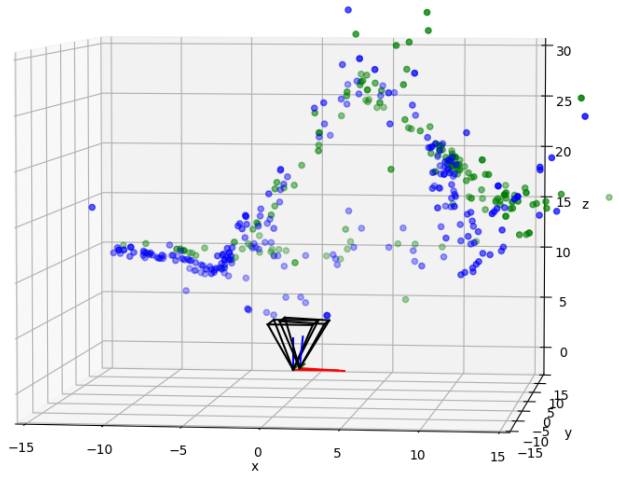
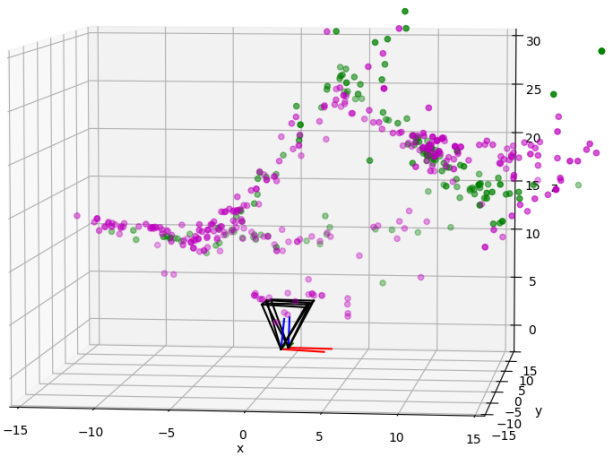


Figure 5: Feature track built after feature matching

Triangulation The 3D points triangulated from an image pair are seen in Figure 6.



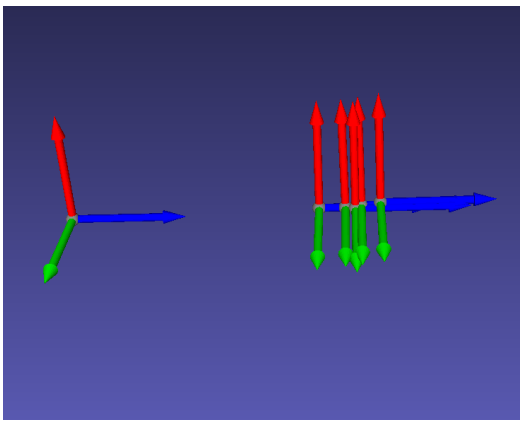
(a) Triangulation of points from image 1 to 3



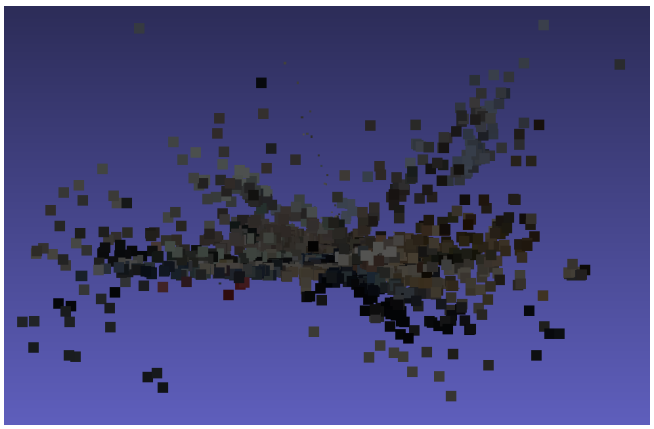
(b) Triangulation of points from image 2 to 3

Figure 6: 3D Point Triangulation

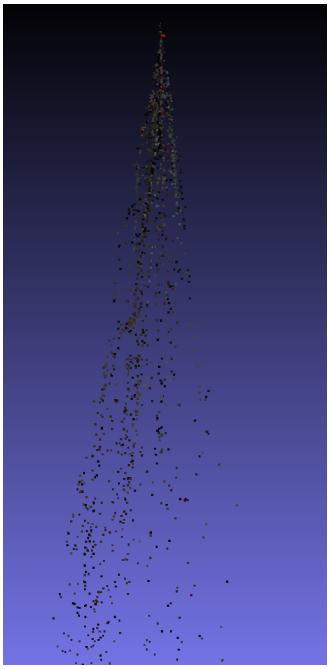
Reconstruction A sparse cloud reconstruction was obtained after performing bundle adjustment through the least squares optimizer method. The results of the reconstruction are presented in Figure 7.



(a) Camera pose estimates



(b) Front view of sparse point cloud



(c) Top view of sparse point cloud

Figure 7: Sparse 3D reconstruciton results

6 Discussion & Conclusion

Overall, the goal of this project was successfully achieved. A sparse point cloud was generated using incremental SFM. Important concepts such as feature matching, camera pose estimation, perspective point algorithm and bundle adjustment were understood and successfully implemented to achieve the output.

However, this work could be improved with further optimised triangulation methods. It would also require more efficient filtering techniques (perhaps by involve learning methods) both at detection as well as estimation stage. Future scope of this project involves generating a dense

point cloud reconstruction. Another possible direction to explore would be Global SfM, instead of incremental approach. Instead of incrementally increasing the number of images, Global SfM takes into account the entire view graph. This approach produces comparable or better results and is considered to be faster than the traditional approach.

References

[1] Yiannis Aloimonos. University course materials - citing and referencing - subject guides at monash university. https://guides.lib.monash.edu/citing-referencing/apa-university-course-materials. (Accessed on 12/15/2021).

[2] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[3] Keith N Snavely. *Scene reconstruction and visualization from internet photo collections*. University of Washington, 2008.

[4] Changchang Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013.

[5] Hyun Soo Park. https://www-users.cse.umn.edu/~hspark/csci5563_s2021/csci5563.html. https://www-users.cse.umn.edu/~hspark/csci5563_S2021/csci5563.html. (Accessed on 12/15/2021).

[6] Silvio Savarese. Cs231a: Computer vision, from 3d reconstruction to recognition. https://web.stanford.edu/class/cs231a/. (Accessed on 12/15/2021).

Appendix

All source code is in the following GitHub repository: https://github.com/saammmy/CV_SFM

For any issues in accessing to the repository, please contact Aditya Mehrotra or Samarth Shah via email.