3D Object Reconstruction From Multi-View Images/Videos

Ojas Mandlecha, Dheeraj Bhurewar, Rajnish Gupta, Dhruv Gupta

November 23, 2022

1. Project Summary

The project would aim to obtain a 3D model render from a video or set of images, stitched together and projected on a 3D meshgrid to get a scaled version of the real objects. We would use various functions from OpenCV, to do various matrix calculations, and find various It would provide sparse points of the object using photogrammetry. In the final result we would have a 3D sparse point cloud render of the objects in the given images. We obtained data from various online dataset of images with a given calibration matrix, and we also used our mobile camera to take a few images and do a reconstruction on that. This method is known as Photogrammetry, and it incorporates various mathematical techniques to obtain the relations between the given images.

2. Goals and Objectives

With this project, we aim to find common features between two images, and then find those points on a 3-D mesh. Before we set up to move and take images, we need to know about the camera, and how it relates to objects in 3-D space to that in 2-D. While obtaining an image, the perspective only gives certain information, and our aim is to match that information from two different perspectives, to retrace the origin of that point.

- Given a set of images (obtained either manually or through a video) and obtaining a calibration matrix of the camera through which the images or video are taken.
- Performing SIFT feature matching to get the common points between the two given images.
- Use the features to find the fundamental matrix, and compute Essential matrix form the fundamental matrix
- Using the above parameters, we would need to find the camera poses, in both translational space and rotational space, and then we multiply that with the calibration matrix to obtain the Camera matrix.
- Using the parameters above with the help of triangulation, to get the 3D location of the matched features (points) between pairs of images.
- Once we have the 3-D location of those points/pixels, we would callback the pixels of the original image, and obtain the RGB values of those pixels and store that next to the point we obtained.
- Once we have information of the RGB value, and the position of that pixel in 3-D space, we would do a sparse reconstruction of multiple points.
- We would repeat this over multiple images, and assign all the points the same scaling, and present them in a 3-D mesh.

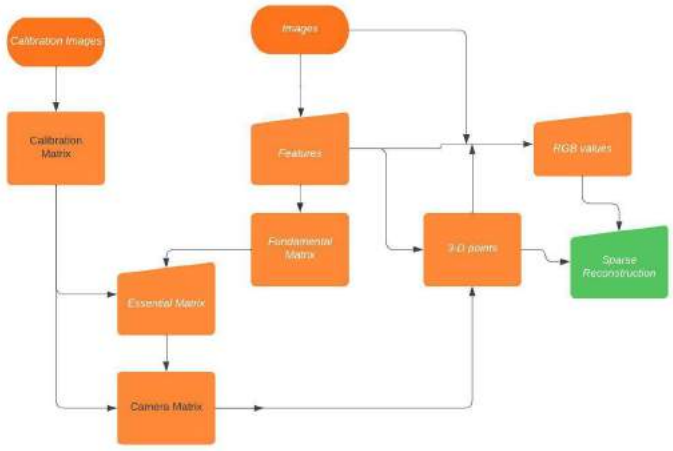


Figure: Procedural Flow of the workings

3. Related Work

Many studies have been conducted on 3D reconstruction of an object from a given set of images using the SFM concept. [1] gives a very detailed description on construction of a 3D structure with various possible methods including but not limited to homography, optical flow, etc while giving a brief explanation on the concepts used. [2] also gives a very brief overview on how to construct a 3D structure by taking images from various views or angles. [3] provides python tools to do photogrammetry which proved to be useful in the project.

4. Methodology:

The basic idea in 3D image reconstruction is that given a set of images {I1, ..., IN} where each image is taken from a different viewpoint, our goal is to use these images to reconstruct a three dimensional representation of the object. More specifically, we will find the motions of the cameras with respect to a world coordinate frame FW. This motion of cameras is also known as camera projection matrices {P1, ... PN}. Using this set of camera projections we will then use different algorithms to recover the 3D structures of the scene.

To perform this, we will construct a pipeline that will consist of two main parts, data association, structure-from-motion (SfM). Data association is used to check whether a pair of images are similar to each other. Two images can be checked for similarity by using image correspondences and robust two-view geometry. Structure-from-motion is responsible for the initial reconstruction using pose estimation and triangulation techniques and refining this using the bundle adjustment algorithm.
The main pipeline of the 3D reconstruction is as follows:

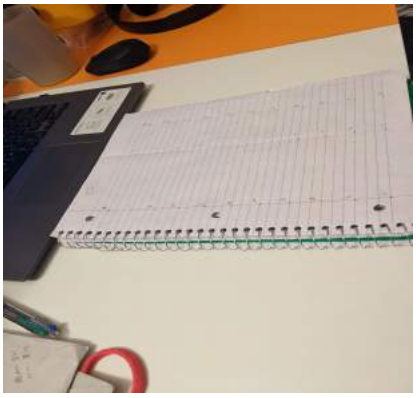
A. Camera Calibration.
B. Image Capture.
C. Structure From Motion (SFM).
D. Perspective and Point Calculation (PnP)
E. 3D Model Construction
F. Reprojection Error
G. Bundle Adjustment

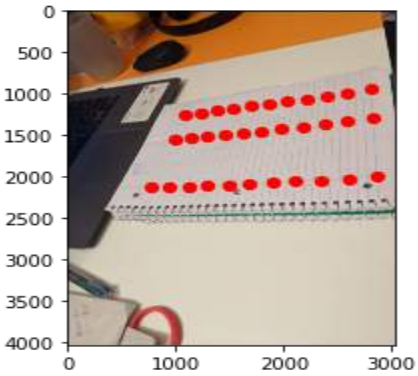
A. Camera Calibration:

Checkerboard Calibration uses calibration plates to compute intrinsic parameters such as lens distortion and perspective distortion and generate images that are distortion free. After

checkerboard calibration, a transform between Raw2D and Plate2D will be established. Checkerboard calibration targets are also used in other calibrations such as hybrid hand-eye calibration, hybrid cross calibration or hybrid manual calibration. In all these calibrations, the checkerboard is used to compute intrinsic parameters.

Similarly we do the camera calibration by plotting points on a sheet of paper and then tracking it. The Figure below illustrates the tracked points.





This gives us the calibration matrix K.

$$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

B. Image Capture:

Our Dataset consists of objects having different shapes, sizes, coloration and texture contour. Some of the objects we used are coins, a screwdriver, a detergent bottle and a fountain. We experimented with all of these objects to see which object when through the pipeline leads to the best 3D reconstruction.

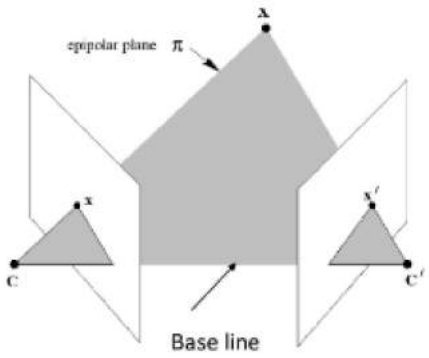
We began by capturing the video of the above mentioned objects from all around them so that we can capture frames from all angles. We then extracted the frames after an interval of 10 frames because we were able to get the best results at this interval (This number was found by experimentation).

C. Structure From Motion (SFM):

We used SIFT to get the common features between pairs of images. Using the Fundamental matrix (F) and obtained Calibration matrix (K) the Essential matrix (E) is calculated using the below mentioned formula

$$E = K.T \,@\, F \,@\, K$$

The fundamental matrix, denoted by FF, is a 3×3×3 (rank 2) matrix that relates the corresponding set of points in two images from different views (or stereo images). In order to understand what a fundamental matrix actually is, we need to understand what epipolar geometry is.



The *epipolar geometry* is the intrinsic projective geometry between two views. It only depends on the cameras' internal parameters (KK matrix) and the relative pose i.e. it is independent of the scene structure.

The Fundamental matrix is only an algebraic representation of epipolar geometry. Unlike homography, in Fundamental matrix estimation, each point only contributes one constraint as the epipolar constraint is a scalar equation. Thus, we require at least 8 points to solve the above homogenous system. That is why it is known as the Eight-point algorithm.

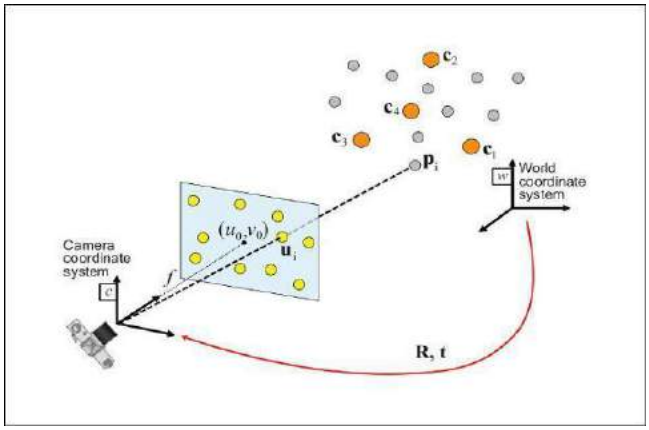
$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_my_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

Since we have computed the FF using epipolar constraint, we can find the relative camera poses between the two images. This can be computed using the Essential Matrix, EE. Essential matrix is another 3×33×3 matrix, but with some additional properties that relate the corresponding points assuming that the camera obeys the pinhole model (unlike FF).

The camera pose consists of 6 degrees-of-freedom (DOF) Rotation (Roll, Pitch, Yaw) and Translation (X, Y, Z) of the camera with respect to the world. Since the EE matrix is identified, (C_1,R_1), (C_1,R_1), (C_1,R_1), (C_1,R_1) where C $\in R^3$ is the camera center and R $\in SO(3)$ is the rotation matrix can be computed.

D. <u>Perspective-n-Points (PnP):</u>

Our pipeline uses incremental SfM which builds upon a baseline reference of 3D points to get 3D reconstruction from many views. Now, since we have a set of (n) 3D points in the world, their 2D projections in the image and the intrinsic parameter; the 6 DOF camera pose can be estimated using linear least squares. This fundamental problem, in general, is known as Perspective-n-Point (PnP). For there to exist a solution, n ≥ 3. PnP is prone to error as there are outliers in the given set of point correspondences.



E. 3D Model Construction:

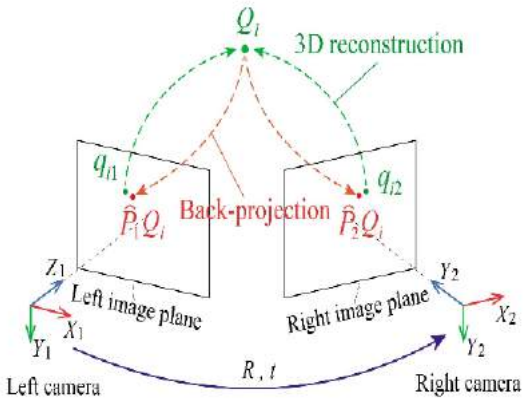
Triangulation is performed to determine a point in 3D space given its projections onto two, or more, images. The above procedure is followed iteratively over the three images in sequence and the 3D points are appended to the list. These matched features are traced back to the image to get the required RGB values. With 3D mounts and their RGB values, we then construct a sparse reconstruction of the object present in the data.

F. Reprojection error:

Once the 3D coordinates of the point are computed, the 3D point is reprojected on all the images that it appears. This error depends on the quality of the camera calibration (position and orientation), as well as on the quality of the marked point on the images (position and zoom level at which the point is marked).

G. Bundle Adjustment:

Finally we do Bundle adjustment (not implemented) to refine the 3D reconstruction. Bundle adjustment boils down to minimizing the reprojection error between the image locations of observed and predicted image points, which is expressed as the sum of squares of a large number of nonlinear, real-valued functions. Thus, the minimization is achieved using nonlinear least-squares algorithms. A popular technique is the Levenberg–Marquardt algorithm



5. Results & Analysis

We tested our code on various datasets which included a few of our image sets as well. Following are the results when we tested our code on the fountain data set. Here are the two images from different perspectives.

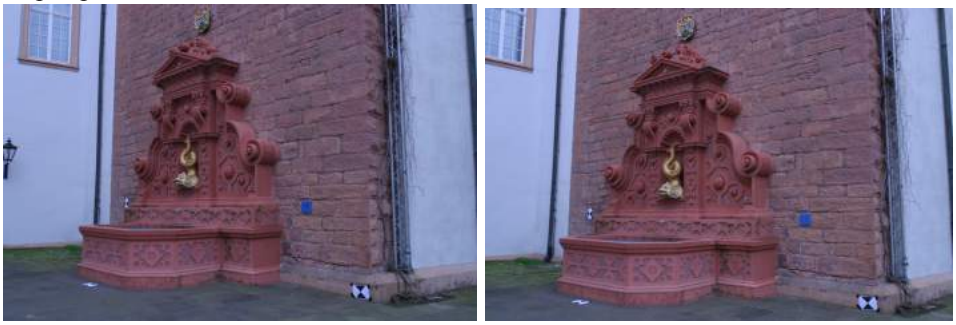


Fig: First set of image pair of fountain dataset

Here is another image with features matched, which are calculated using SIFT Feature matching. We will further use it to calculate the Fundamental matrix, to get the pixel values from the original image, and use the location of these points to find them in scaled 3-D space.

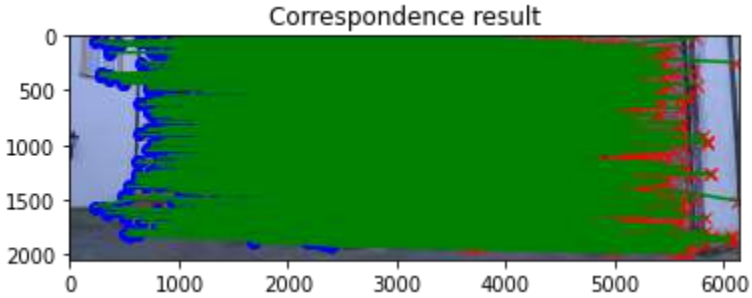


Fig: Feature matching using SIFT

The features are matched well, especially on the object of interest, but we also see a lot of features surrounding it. Approximately 770 points were obtained between the first pair which is a pretty good number. We can increase this number using more images, but we also need to keep in mind that the trend of obtaining feature points is not linear, as a lot of images would have some common points. The following sparse 3D reconstruction is obtained from the above two views.

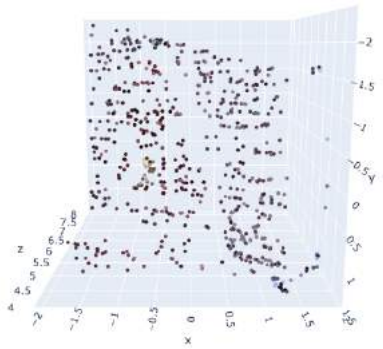


Fig: 3D reconstruction of points between first set of image pair of fountain dataset

Next, we have the reconstruction from multiple images, and as one would expect, we can find more

spaces filled in the points, and they retain the pixel value close to that of the original image. As there can also be small errors and outliers while calculating the fundamental matrix but any two images, those small errors which is defined as reprojection error.

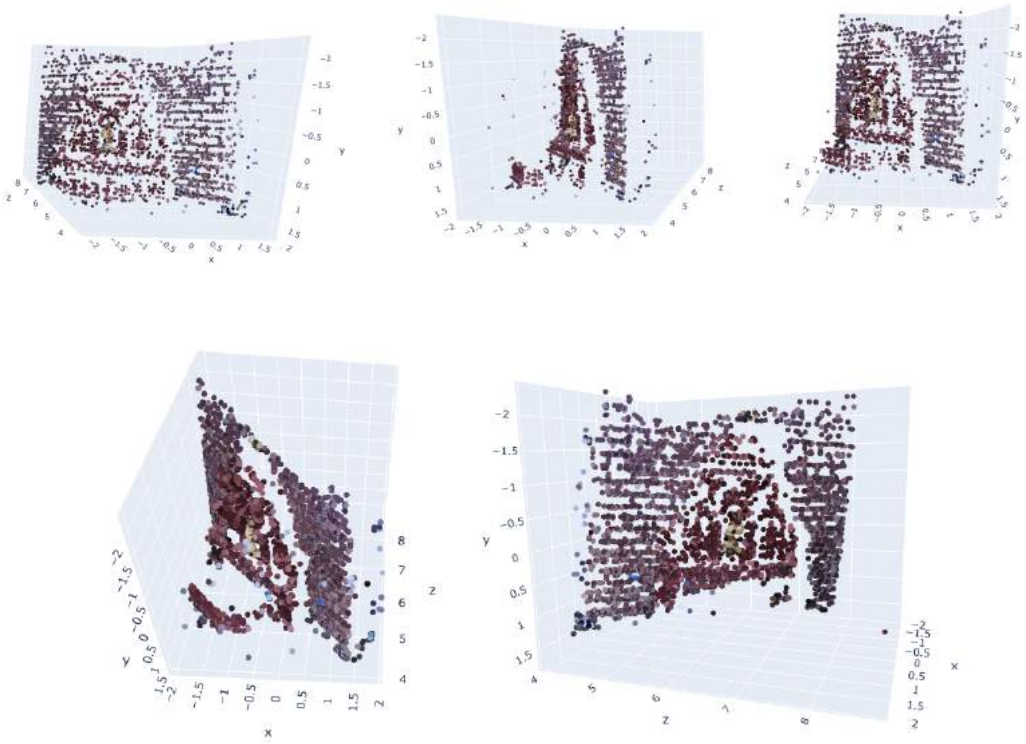


Fig: Multiple view of sparse 3D reconstruction

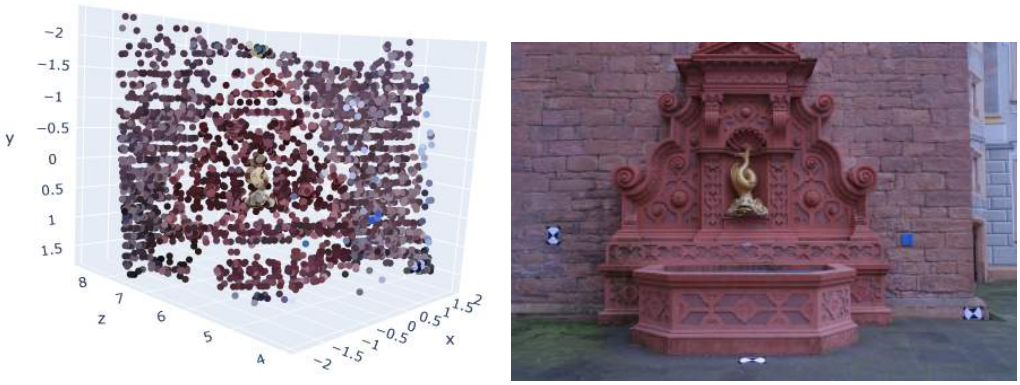


Fig: The sparse reconstruction with its image (Front view)

Due to reprojection error and not using bundle adjustment, there were few outliers which are removed by applying limits to the coordinates. Here we have various views of our final projection. with a total of 6000 points, which we obtained from multiple images.We can still see the images on this page to have a skewed view from a certain perspective, which can be made better with bundle adjustment, and removing those small errors. As compared to the view from only two images, we can certainly notice more colors. The color of the bird, which is golden in the center, does look pretty interesting. You can move around an interactive view in the code, and visualize it in a much better way.

6. Challenges

a. **Obtaining more features:** One of the challenges we initially had on a different set of images was to find the matching features. To have information of more matching pixels, we tried to have transformation from one image to other through homography, and finding more matching pixels. While it allowed us to obtain the corresponding points. When we plotted that in 3-d mesh, the had very less depth, and were mainly plotted as 2-D pixels. Possible solutions to this include using Optical flow (was difficult to implement), and a form of suppression to get more features on the object of interest rather than the background.

Fig: Dense reconstruction after using points from homographic transformation.

b. **Scaling points between multiple images:** While scaling images, a lot of the information was lost, which raised concerns about our feature matching. A possible solution for this could be to use more computationally efficient feature matchers.

c. **PnP Matching:** While trying to obtain the camera pose for subsequent images, it was difficult to find a common track between a subset of images. A workaround was used to match points within a threshold (1 pixel of each other), but this is also what was responsible for the skewing of the final result, most likely.

d. **Sensitivity:** A lot of the calculations involved are very sensitive to minor shifts in coordinates or parameters. Most importantly, the triangulation is not exact, and the results obtained are approximations which need to be corrected using bundle adjustment.

7. References

1. David Millán Escrivá and Roy Shilkrot, 'Mastering OpenCV 4: A Comprehensive Guide to Building Computer Vision and Image Processing Applications with C++', 3rd Edition.
2. Zhang H, Wei Q, Jiang Z. 3D Reconstruction of Space Objects from Multi-Views by a Visible Sensor. Sensors (Basel). 2017 Jul 22;17(7):1689. doi: 10.3390/s17071689. PMID: 28737675; PMCID: PMC5539474.
3. Moulon, Pierre & Bezzi, Alessandro. (2011). Python Photogrammetry Toolbox: A free solution for Three-Dimensional Documentation.
4. https://github.com/FlagArihant2000/sfm-mvs
5. https://cmsc426.github.io/sfm/
6. Frederick M. Weinhaus and Venkat Devarajan. 1997. Texture mapping 3D models of real-world scenes. ACM Comput. Surv. 29, 4 (Dec. 1997), 325–365. https://doi.org/10.1145/267580.267583