Reconstruction Project (COL 780)

Ashutosh Agarwal, Shivansh Chandra Tripathi

Abstract

Structure from motion(SfM) is a technique to estimate the 3d structure of objects using their corresponding 2d images taken from multiple views. In the most general case, we've only the images as input, and we are required to predict Camera's intrinsic parameters, and then construct the 3d view. In this project, we'll see a 3d reconstruction model for India Gate. Starting from the base model, we'll see the step by step improvements that we've built on it.

1. Introduction

Structure from motionv (SfM) is the process of reconstructing 3D structure from its projections into a series of images taken from different viewpoints. It is a process of estimating camera pose and retrieving a sparse reconstruction simultaneously. Let us now understand the various steps involved to solve the problem :

1. **Detect Features**: The first stage is correspondence search which finds scene overlap in the input images. It identifies projections of the same points in overlapping images. This step detects features in the images, using any of the feature descriptorsv(HAHOG, SIFT, SURF, ORB, etc.)

2. **Match Features**: This step matches feature points between images. It first determines the list of image pairs to run, and then run the matching process for each pair to find corresponding feature points. Since there are a lot of possible image pairs, the process can be very slow. It can be sped up by restricting the list of pairs to match.

3. **Geometric Verification**: The third stage verifies the potentially overlapping image pairs C. Since matching is based solely on appearance, it is not guaranteed that cor- responding features actually map to the same scene point. Therefore, SfM verifies the matches by trying to estimate a transformation that maps feature points between images using projective geometry. Depending on the spatial configuration of an image pair, different mappings describe their geometric relation. A homography H describes the transformation of a purely rotating or a moving camera capturing a planar scene. If a valid transformation maps a sufficient number of features between the images, they are considered geometrically verified. Since the correspondences from matching are often outlier-contaminated, robust estimation techniques, such as RANSAC, are required. The output of this stage is a set of geometrically verified image pairs C, their associated inlier correspondences M.

4. **Reconstruct**: This step runs the incremental reconstruction process. The goal of the reconstruction process is to find the 3D position of tracks (the structure) together with the position of the cameras (the motion). **Incremental reconstruction algorithm** For this project, we will be using OpenSFM python library to build the reconstruction pipeline. OpenSfM implements an incremental structure from motion algorithm. This is reconstruction algorithm that starts building a reconstruction of a single image pair and then iteratively add the other images to the reconstruction one at a time.The algorithm has three main steps:

   - **Finding good initial pairs**
     To compute the initial reconstruction using two images, there needs to be enough parallax between them. That is, the camera should have been displaced between the two shots, and the displacement needs to be large enough compared to the distance to the scene.
     To compute whether there is enough parallax, we start by trying to fit a rotation only camera model to the two images. We only consider image pairs that have a significant portion of the correspondences that can not be explained by the rotation model. We compute the number of outliers of the model and accept it only if the portion of outliers is larger than 30
     The accepted image pairs are sorted by the number of outliers of the rotation only model.
   - **Bootstrapping the reconstruction**

To bootstrap the reconstruction, we use the first image pair. If initialization fails we try with the next on the list. If the initialization works, we pass it to the next step to grow it with more images.

The reconstruction from two views can be done by two algorithms depending on the geometry of the scene. If the scene is flat, a plane-based initialization is used, if it is not flat, then the five-point algorithm is used. Since we do not know a priori if the scene is flat, both initializations are computed and the one that produces more points is retained.

If the pair gives enough inliers we initialize a reconstruction with the corresponding poses, triangulate the matches and bundle adjust it.

- **Growing the reconstruction**
  Given the initial reconstruction with two images, more images are added one by one starting with the one that sees more of the reconstructed points.

  To add an image it needs first needs to be aligned to the reconstruction. This is done by finding the camera position that makes the reconstructed 3D points project to the corresponding position in the new image. The process is called resectioning.

  If resectioning works, the image is added to the reconstruction. After adding it, all features of the new image that are also seen in other reconstructed images are triangulated. If needed, the reconstruction is then bundle adjusted and eventually all features are re-triangulated.

5. **Undistort**: This step creates undistorted version of the reconstruction, tracks and images. The undistorted version can later be used for computing depth maps.

6. **Compute Depthmaps**: This step computes a dense point cloud of the scene by computing and merging depthmaps. It requires an undistorted reconstructions.

2. Dataset Collection

The monument we've chosen is India Gate. There should not be too small disparity between different images else the depthmap computation will be less precise. Keeping these points in mind, our initial dataset was comprised of high resolution images collected from google and flickr. Later we made an improved dataset of high resolution images by collecting large number of frames from a drone video of India Gate. We also collected data using screenshots of 360 degree photos on youtube. At last, to get the top views we took screenshots from the satellite view using google earth.

3. Reconstruction Models

3.1. Model 1

Figure 1 shows the result of opensfm pipeline on a set of image dataset of set $\sim$ 100 from google and flickr. As we can clearly see from the figure, only the front facing view for India Gate is being reconstructed from the image.

In these 100 images, we used all 4 view images of the monument, but still we were able to reconstruct only the front face because in most of the images includes the monument details and not its surroundings. Since India gate is symmetrical, we were getting only the front view of the monument because the front face and back face corresponded to the same depth point and our model couldn't distinguish between them.

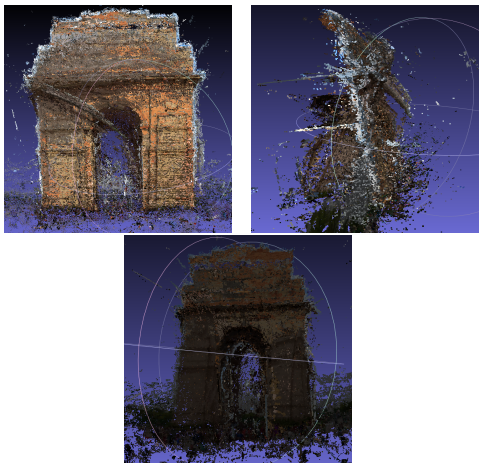

Figure 1. Results of Model 1

3.2. Model 2

Lets discuss solution to the above problem. To break the symmetrical mappings for front-back and both sides, we can include the surrounding details for the image which will break the symmetry between the front and back views per se as we get more context for image points to differentiate between both view points. We increased the data set to $\sim$ 250 to using screenshots of the 360 degree photos on YouTube to include surrounding regions of the monument as well. Figure 2 shows the reconstruction results for this model.

What problems do we see in this model?

1. Illumination variation in the reconstruction

2. Lack of edges on the face of monument

3. Sparse Reconstruction

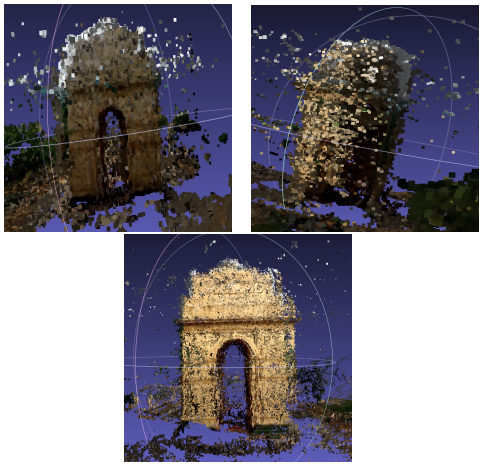

Figure 2. Results of Model 2

3.3. Model 3

The problems in the model above simply indicates problem with our data set. To overcome the illumination, we performed data set cleaning. This is one of the standard practice for any learning task, since a noisy input can affect the entire output. We cleaned those images of the monument, which were taking in the night conditions (since they had lighting in the monument) or had quite different illumination in the day time in comparison to the other images. Next, to resolve the lack of details and sparse reconstruction we increased the data set to $\sim$ 500, by extracting frames from the video and the result using the default parameters is shown in figure 3.

Figure 3. Results of Model 3

4. Scope of Improvements

Let us list the following problems in the above Model 3.

1. Sparse reconstruction of the top of model.
2. The green points along the edges of front of India Gate.
3. Noisy points on the top.
4. Lack of details like written India Gate on the top of monument and the structure on the side view.

The above problems are shown in Figure 4.

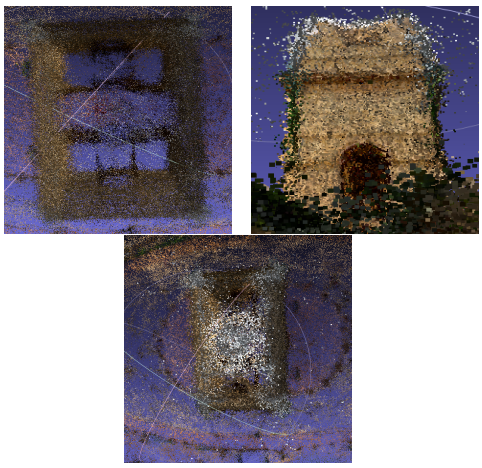

Figure 4. Sparse reconstruction, Green points on edges, Noisy top

4.1. Sparse reconstruction of the top of model

We thought that this problem exists due to lack of images from the top view from all of our sources, videos, internet scraping and 360 view photos. To overcome this caveat, we clicked some screenshots from google Earth for top view. Since, google earth gives satellite views, we were able to obtain $\sim$ 10 images for the top view. We included those images the data set and performed reconstruction pipeline once again. Figure 5 shows slight improvement on the top view, but we were still not able to achieve much dense reconstruction of top view due to lack of images.

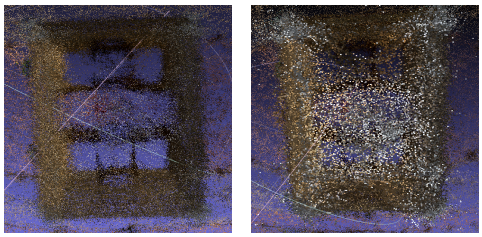

Figure 5. Sparse reconstruction, Dense reconstruction

4.2. Green points along the edges

We reasoned that this may be caused due to problems in feature matching and feature descriptors for images in the opensfm pipeline. So, this section we will experiment with different feature descriptors and matches.

4.2.1 Feature Descriptors

We experiment with 4 descriptors in the openSFM pipeline, SIFT, SURF, ORB and HAHOG.

Figure 6. With ORB as feature detector

It can be from 6 seen that ORB mapped side and front to depths which couldn't preserve the monument geometry, next we use SIFT and the results are shown in figure 7(Note that we didn't mention SURF because it's outputs were similar to SIFT, but we choose SIFT because it performed better in further improvements with its parameter tweaked). On the other hang HAHOG gives a sparse construction.

4.2.2 Parameters for SIFT

Next, we tried to tune the parameters of the SIFT descriptors. We performed 3 experiments here. The SIFT detector is controlled mainly by two parameters: the peak threshold and the (non) edge threshold. The peak threshold filters peaks of the DoG scale space that are too small (in absolute value). Increasing it means more features, so may be better matching. The edge threshold eliminates peaks of the DoG scale space whose curvature is too small (such peaks yield badly localized frames). Decreasing it means more features, so may be better matching.
The default parameters for SIFT in opensfm are:
sift peak threshold: 0.01
sift edge threshold: 10

Peak threshold	edge threshold
0.01	10
0.001	10
0.01	100
0.001	100

It can be seen in Figure 7 that first one is better of the three. So, we fix the parameters of sift as
sift peak threshold: 0.001
sift edge threshold: 10

4.2.3 Feature Matchers

For matching, we used FLANN matcher which is the default parameter in opensfm. We tried using BoW matcher too, but since it only works with HAHOG descriptors and

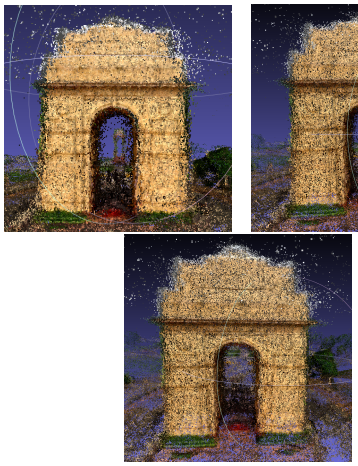

Figure 7. SIFT with (peak,edge) threshold as (0.001,10), (0.01,100) and (0.001,100) respectively.

HAHOG decriptors gave bad results, we used matcher as FLANN. FLANN stands for Fast Library for Approximate Nearest Neighbors. It contains a collection of algorithms optimized for fast nearest neighbor search in large data sets and for high dimensional features. It works faster than BF-Matcher for large data sets. We will see the second example with FLANN based matcher. Also, the second reason for using this the time constraints, since we found out FLANN based matching was faster than BoW and of course brute force, we decided to go ahead with it.
Where we able to solve the green point on the edge problem? No, but experimenting with this, we we able to get a denser reconstruction.

4.3. Noisy points on the top

The reason for noisy top will be the projection of noisy points in the 3d reconstruction. So, to overcome this we performed an experiment to change the bundle adjustment loss. A LossFunction is a scalar valued function that is used to reduce the influence of outliers on the solution of non-linear least squares problems. In addition to SoftOne loss we used to more losses to see if we can get rid of noisy points at the top of the model namely:

Huber Loss Huber loss is a loss function used in robust regression, that is less sensitive to outliers in data than the squared error loss.

$$\rho(s) = \begin{cases} s & s \leq 1 \\ 2\sqrt{s} - 1 & s > 1 \end{cases}$$

Cauchy Loss

$$\rho(s) = \log(1 + s)$$

Figure 8 shows that Huber loss performs better in comparison to the Cauchy loss and Softone loss that is the default parameter. Cauchy loss performs similar to the Softone loss. Even though the density reduced but the noise is

reduced. As, we can see those dense points were actually the noisy points.

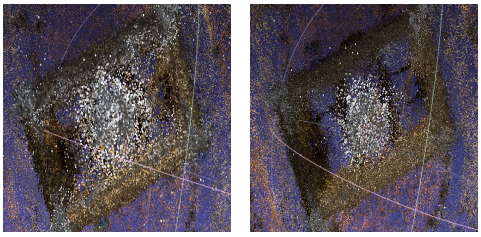

Figure 8. Before, After

4.4. Lack of details

The reason for noisy top will be the projection of noisy points in the 3d reconstruction. So, to overcome this we performed two experiments:

1. **Changing the triangulation threshold** We thought that lack of the details was due to sparsity of the construction. So, to overcome, we increased the triangulation threshold, which indicated hon much error can a point have for it to be present in the construction. Unfortunately, it backfired which is clearly visible in Figure 9. The solid on one of the pillars is protruded.

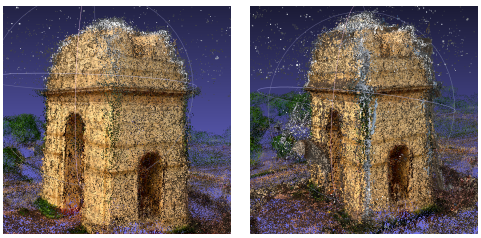

Figure 9. Before, After

2. **Adding more images corresponding to the detailed feature** To do this, we took the highest quality India Gate images, and obtained more images that corresponds to features like India Gate written on top, wheel and the side structure. These are shown in Figure 10. After running a reconstruction, we saw small refinements in the small objects. This is shown in Figure 11.

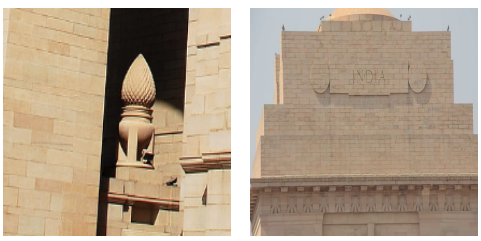

Figure 10. Detailed Featured of Monuments

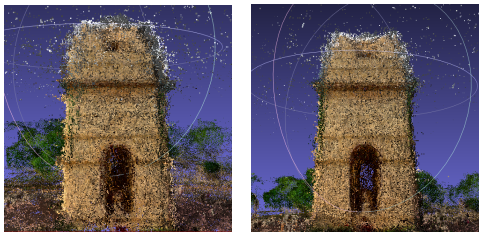

Figure 11. Before, After

References

[1] https://www.opensfm.org

[2] https://medium.com/@kazuyukimiyazawa/revisit-structure-from-motion-revisited-part-1-b1edde150a06

[3] http://www.ceres-solver.org

[4] https://blog.mapillary.com/update/2016/10/31/denser-3d-point-clouds-in-opensfm.html

[5] https://github.com/rajkataria/ReliableResectioning

[6] https://www.google.com

[7] https://www.flickr.com

[8] https://www.youtube.com