3D Reconstruction from Video Frames

Susim Mukul Roy
B20AI043
roy.10@iitj.ac.in

Rohan Singh
B19EE072
singh.77@iitj.ac.in

Abstract

3D reconstruction from images is an open problem that is actively being worked upon. Good solutions can have a massive impact on the CGI and architectural visualization community, as the luxury to simply scan an object and obtain a good digital reconstruction is not available currently at mass-scale. Already existing algorithms require hard to acquire hardware and professional studios which cannot be used out in the field. Most scans must be cleaned up manually by an 3D artist/engineer before use and such techniques are not suitable for end user devices with less computational power. Most of these methods are deep-learning approaches. Therefore, we want to develop a non-deep learning algorithm with better results in terms of time, model complexity and model explainability. We compared our methods with the existing algorithms and had at par performance with them. We compared these methods on parts of the datasets used on those papers as well as tested on our newly collected evaluation set of videos. We observe that our computationally effective model performs similar to existing heavy architectures.

1. Introduction

Recent advances in semiconductor technology have democratized access to cheap, high resolution, high quality cameras. Almost all cell-phones have multiple high resolution cameras embedded within them. This democratization of high quality cameras has led to a huge increase in the volume of photo and video content being generated. However, most of the content being generated today is still 2D, and very little 3D content is being created even today. Structure from Motion (SFM) is a well-studied problem in the field of computer vision and visual perception. SFM algorithms enable us to reconstruct 3D views of an object from multiple 2D views of the object of interest. Over the past few years SFM algorithms have advanced significantly and a number of groups have demonstrated the up-to large city scale 3D reconstruction using SFM algorithms. One such seminal example is 'Reconstructing Rome in a day'[1]. However, despite these technological advances 3D image reconstruction has not been democratized yet, and the amount of 3D content being generated is still rather limited, especially relative to the amount of 2D data that is generated every-day. One potential reason for this is the difficulty in obtaining the large number of views of the object of interest. A naïve way to obtain the different views requires the user to manually click a number of 2D images for the target of interest from different viewpoints. However, this requires care and precision on part of the user. The user has to focus on the required object and has to be careful to minimize the background noise. An alternate approach could be that the user creates a video of the target of interest while ensuring that he has captured multiple views of object and the background noise is minimized. However, even this method requires careful video generation on the user's part. The github link for the problem is :https://github.com/SusimRoy/3D-Image-Reconstruction-from-Video-Frames. The link to the video file demonstrating our work is: Drive

2. Approach

Our approach has been divided into various subsections:- Extracting Keyframes, Projective Scene Reconstruction and Conversion of 3D image from pointcloud data.

2.1. Extracting Keyframes

Given a video, we are extracting only the key frames from that video which are necessary for our task. The number of frames is set by the user and that particular no. of frames will be taken from every video and each frame will be stored in a separate folder.

2.2. Image Rectification

Extraction of Manual Correspondences
Manually extract a minimum of 8 sets of corresponding points between the two stereo images. For this experiment I have used a total of 10 corresponding points between the two images. I have extracted the correspondence points using the SIFT detector and then finding the coordinates of the

matching points.

Estimation of Fundamental Matrix

Once the correspondence points have been manually extracted between the two points, the following are the steps to calculate the fundamental matrix:

1. The correspondence points from each image needs to be normalized using a normalizing matrix T. Given that x' is the corresponding coordinate from the right image and x is the corresponding coordinate from the left image. The normalized points are given by:

$$\hat{\mathbf{x}} = \mathbf{T}\mathbf{x}, \hat{\mathbf{x}'} = \mathbf{T}'\mathbf{x}'$$

Here T and T' are calculated such that all the correspondence points have a mean of 0 and the are at a distance of $\sqrt{2}$ from the center.

2. Let x' be the (not normalized) correspondence points from the right image while x (not normalized) are the ones from the left image. The fundamental matrix F is given by the following formula:

$$\mathbf{x}_\mathbf{i}'\mathbf{F}\mathbf{x}_\mathbf{i} = \mathbf{0}$$

where,$\mathbf{x'} = [\mathbf{x'}, \mathbf{y'}, 1]^\top$ and $\mathbf{x} = [\mathbf{x}, \mathbf{y}, 1]^\top$

Thus, from N correspondence points we get:

$$A.f = \begin{bmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & y_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N'x_N & x_N'y_N & x_N' & y_N'x_N & y_N'y_N & y_N' & x_N & y_N & 1 \end{bmatrix} f = 0$$

$$f = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Thus, f can be solved by conducting the Single value decomposition of A and found using SVD as the eigen vector corresponding to the smallest eigen value.

3. F needs to rank deficient and needs to have a rank of 2. The rank of F can be conditioned to 2 by using SVD of F. $F = UDV^T$,to set the rank of F to 2, the smallest singular value in D is set to 0. Then $F_{cond} = UD_{cond}V^T$,D_{cond}has the smallest singular value is set to 0.

4. Since F was calculate using normalized points, we need to denormalize the matrix. Denormalizing F can be doing using the following formula:

$$F = T'^\top F_{cond} T$$

Estimation of projection matrices in the canonical form from Fundamental matrix

Once the fundamental matrix F is calculated the projection matrices can be calculated using the following formulas:

$$F.e = 0, e'^\top.F = 0$$

$$P_1 = [I|0], P_2 = [[e']_x F|e']$$

Refining the fundamental matrix using the Levenberg Marquardt algorithm

Once the fundamental matrix and the projection matrices in the canonical form are calculated they can be refined using the LM algorithm. In the following experiment I have done that by refining the matrix P_2.

The LM algorithm combines both the gradient descent and gauss newton methods. In the LM algorithm the step size in each iteration is given by the formula:

$$\overrightarrow{\delta_p} = (J_{\overrightarrow{f}}^\top J_{\overrightarrow{f}} + \mu I)^{-1} J_{\overrightarrow{f}}^\top \overrightarrow{\epsilon}(\overrightarrow{p_k})$$

, $\overrightarrow{\epsilon}(\overrightarrow{p_k} = ||X'_{actual} - F(P_k)||$ is the error function, where $F(P_k)$ is $PX_{worldcoordinate}$. The 3D world coordinates are calculated by triangulating the 2D coordinates x and x'. For each x and x' correspondence pair the following matrix is created:

$$A = \begin{bmatrix} xP_3^\top - P_1^\top \\ yP_3^\top - P_2^\top \\ x'P_3'^\top - P_1'^\top \\ y'P_3'^\top - P_2'^\top \end{bmatrix}$$

3D world coordinate is given by the null vector of A. The null vector can be calculated conducting the SVD of A and taken the eigenvector corresponding to the smallest eigen value.

The cost function for the LM algorithm is given by, $D_2 = \sum ||x - PX_{worldcoordinate}||^2 + ||x' - PX'_{worldcoordinate}||^2$

Once the refined P_2 matrix is calculated, the refined Fundamental Matrix F can be calculated using as $[]_x$ times the first three columns of P_2.

In the results section the LM algorithm is seen to optimize the left and right rectified image.

In the following experiment I have implemented the LM algorithm to optimize the projection matrix using the scipy.optimize.root library. In order to implement the library, I had to create a function that would calculate the error as well as the Jacobian matrix which was then used as an input to the library.

Estimation of Homography matrices to generate the rectified image

Next the homography matrices H and H' need to be calculated to send e and e' to infinity.

1. The following is the procedure to calculate the homography matrix for the right image

i. First, the angle to the epipole with respect to the positive x axis is calculated using the following formula:

$$\theta = tan^{-1}(-1\frac{(\frac{height}{2}) - e'(y)}{(\frac{width}{2}) - e'(x)})$$

ii. Next the matrix G is calculated as:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{bmatrix}$$

where $f = \cos\theta((\frac{width}{2}) - e'(x) - \sin\theta((\frac{heigth}{2}) - e'(y)$.
iii. The rotation matrix R is calculated:

$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

iv. The translation matrix T is calculated:

$$T = \begin{bmatrix} 0 & 0 & -\frac{width}{2} \\ 0 & 1 & -\frac{height}{2} \\ 1 & 0 & 1 \end{bmatrix}$$

v. Next the image center is translated with respect to H2 and next translation matrix T2 is calculated. The final H' matrix is calculated using the following formula

$$H' = T_2 G R T$$

The right epipole is given by $e' = \begin{bmatrix} f & 0 & 1 \end{bmatrix}^\top$
vi. The homography H' can be applied to the right image to get the right rectified image
2. The following is the procedure to calculate the homography matrix for the left image i. First calculate the matrix M, given by $P'P^+$ ii. Next calculate H_0, given by $H_2 M$ iii. Next, the matrix HA by minimizing the following sum of square:

$$\sum (ax_i + by_i + c - x'i)^2$$

$$H_A = \begin{bmatrix} a & b & c \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

iv. The final H matrix is calculated using the following formula:

$$H = H_A H_0$$

v. The homography H can be applied to the left image to get the left rectified image

2.3. Interest Point Detection

SIFT feature detector

Once the rectified images have been obtained then the SIFT is used to extract the features and the corresponding descriptors from the rectified images. In this task I have used the OpenCV inbuilt SIFT function to calculate the key points and their respective descriptor values. The correspondences between the two key points are then established by calculating the minimum sum of squared differences

SSD : Sum of Squared Differences

SSD is one method for establishing correspondence between the interest points detected in the left and right rectified image. In this method we create a (M+1) x (M+1) matrix around the selected corner points. Once the matrix is created SSD is calculate as,

Figure 1. SIFT Correspondence

$$SSD = \sum_i \sum_j |f_1(i,j) - f_2(i,j)|^2$$

, where f_1 is the window around the point in the 1st image and f_2 is the window around the point in the 2^{nd} image. For each point in the first image find the corresponding minimum SSD value in the second image. Once the minimum SSD values have been calculated for each of the key points the we then sort the correspondences based on the ascending order of their SSD values. I have chosen the 40 best interest points between the two rectified images to calculate their corresponding world coordinate values.

Selection on interest points on the object of interest

The following is the algorithm to select only those interest points that lie on the object of the interest:
1. Manually detect the center of the object of interest and the corners of the object of interest using the harris-corner detection algorithm.
2. Calculate the maximum distance between the center and the corners of the object of interest.
3. Once the maximum distance is calculated, only those correspondence that lie within the maximum distance from the center of the object of interest are selected.

2.4. Projective Reconstruction

The following is the procedure for 3D projective reconstruction of the 2D (x, x') correspondence found between the left and right image:
1. Using the correspondence detected using the SIFT feature detectors, estimate the fundamental matrix and canonical forms of the projection matrices.
2. Refine the P_2 matrix using the LM algorithm refined above and calculate the refined fundamental matrix and canonical forms of the refined projection matrices.
3. The 3D world coordinates are calculated by triangulating the 2D coordinates x and x'. For each x and x' correspon-

Figure 2. Combined Pointcloud

dence pair the following matrix is created:

$$A = \begin{bmatrix} xP_3^\top - P_1^\top \\ yP_3^\top - P_2^\top \\ x'P_3'^\top - P_1'^\top \\ y'P_3'^\top - P_2'^\top \end{bmatrix}$$

3D world coordinate is given by the null vector of A. The null vector can be calculated conducting the SVD of A and taken the eigenvector corresponding to the smallest eigen value.

2.5. 3D Visual Inspection

Once the world coordinates of the interest point between the images is calculated, the world coordinates of the corners points of the object of interest are calculated using the procedure above. These points were then plotted using scatter plot feature in matplotlib in python. The interest points were plotted in blue and the corner points are plotted in green. The corners were connected to create the outline of the 3D reconstruction using the plot feature of matplotlib in python

2.6. PointCloud Conversion

On obtaining the pointcloud data, we combine these data from consecutive pairs of frames and recreate the 3D object. Fig2. is an example pointcloud of randomly generated data to show the proof of concept.

3. Experiments and Results

Our main task was to combine the pointcloud data from consecutive frames in order to recreate the 3D object. In this regard, we have attached pictures of our reconstructed 3D pointcloud data. We could not find work done prior to this in this line and hence couldn't show comparisons. Moreover, constructing each such pairs of images and their corresponding point-clouds was highly time consuming and hence it was not possible to do it on the entire dataset given our computational resources. However, we tested on a few images whose results were fairly good.

4. Limitations

The limitations in this approach are:

1. The SIFT detector might not always correctly identify the corners in the object of interest.
2. The Harris corner detector needs to be tuned as per the object of interest in order to properly detect the corners.
3. The algorithm assumes certain properties such as non-singularity of some matrices, existance of non-trivial solution of null-space of the rectified-image matrix etc. which might not be always true as it depends on the dataset images.

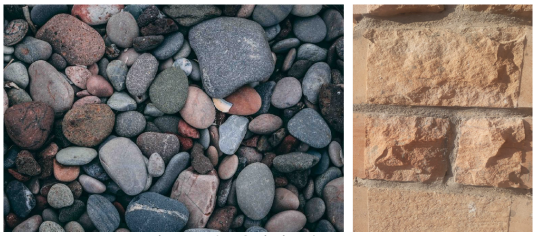

Figure 3. Original Images

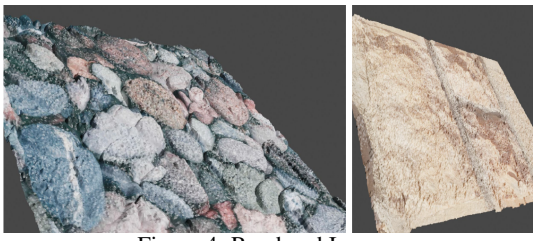

Figure 4. Rendered Images

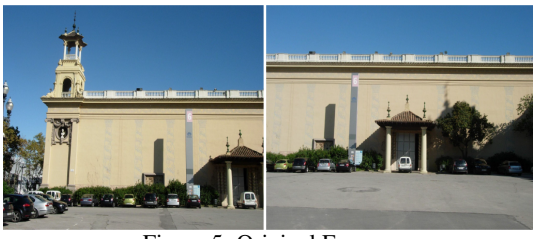

Figure 5. Original Frames

5. Evaluation Set

Our evaluation set includes videos of objects at IITJ campus. We captured videos of those objects and converted them into frames using our code. An example is shown in Fig 3. which are pictures of the hostel walls and random rocks in the campus. Consequently, Fig 4 shows the rendered 3D images of those which is basically the step of pointcloud conversion from 2D image to 3D structure.

6. Datasets used

We used the datasets provided by researchers at ETH Zurich, namely the ETH SfM dataset. The dataset consists of frames from videos made in from 4 different locations of France. Fig 5. is an example from that dataset with 2 images from Barcelona.

7. Ethical issues

One of the major ethical issues is reconstruction of facial structures of people without their consent. It may happen that someone illegaly gets hold of a video of a targeted person and using this techqiue, they create a 3D structure of their face for malicious activities such as creating a doppleganger face etc. Other concerns include estimating the structure of hidden objects whose videos are available. However, recreating the 3D structure might lead to conspicable actions on part of the wrong-doer.

8. Conclusion

Our work shows that less computation techniques exist for recreating 3D structures which are fairly at par with techniques which require high resources and huge datasets. We claim that there exists non-deep learning techniques to achieve the aforementioned tasks which can be extended further and generalized to other environment settings too.

9. References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. 2011. Building Rome in a day. Commun. ACM 54, 10 (October 2011), 105–112. https://doi.org/10.1145/2001269.2001293
[2] https://www.camcalib.io/post/stitching-point-clouds-from-multiple-cameras

https://itecnote.com/tecnote/python-opencv-depth-map-from-uncalibrated-stereo-system/

Bleyer, Michael Gelautz, Margrit. (2008). Simple but Effective Tree Structures for Dynamic Programming-Based Stereo Matching.. Int Conf Comput Vis Theory and Appl. 415-422.

https://icu.ee.ethz.ch/research/datsets.html