Final Year Project

Reconstruct3D

Analysis of Structure from Motion Techniques

Group members

Muneeb Aadil (14I-0140)

Adeel Ahmad (14I-0169)

Supervised by

Dr. Sibt ul Hussain

Department of Computer Science

National University of Computer and Emerging Sciences

Islamabad, Pakistan

2018

Authors' Submission

We hereby declare that the work presented in this report is our own work. All the references are properly cited, wherever they are used in the document.

Abstract

To reconstruct a 3D model/mesh, one requires dedicated hardware such as depth-enabled cameras (i.e. RGBD cameras) or stereoscopic cameras. Such hardware is not available within general purpose smartphones and hence this technology is not accessible for the general public. In this project, we aim to reconstruct a 3D model of static scenery from a monocular video sequence.

To start off with the reconstruction process, a user captures a video of the real 3D world. Then, this video is transferred to a remote server for computation of its 3-dimensional model (which is to be done using epipolar geometry techniques). After the server has finished its computation, user can download the reconstructed 3D model of the static environment.

The main steps in this project include feature matching, fundamental matrix estimation, camera pose estimation, triangulation, perspective-n-point, and depth map estimation. Finally, we present a metric for quantifying our results.

Executive Summary

The report focuses on the problem of generating 3D structure from a set of 2D images. This work is intended to remove the literature barrier of introductory knowledge in the domain of 3D vision, by providing concise yet sufficient and complete resources containing the source code, reference results to cross-check other against implementations, and theoretical foundations.

Firstly, we explain the importance of this problem by listing out some of the applications areas where this problem is faced. We also take a look on the challenges commonly faced while designing a robust solution to give us an idea of the difficulty of the problem. Furthermore, we formalize the problem by expressing our objective in mathematical notation.

Having formalized the problem, we then explain the theoretical foundations of 3D vision necessary for understanding standard pipelines of structure from motion. More specifically, we explain (1) a single view geometry: a transformation governing the process of capturing data from 3D world to 2D camera, (2) multi-view geometry: a set of geometric relations that exist between two cameras capturing the same 3D scene from different angles at the same time, (3) process of actually projecting 2D data back to 3D space once we have the necessary entities, and (4) some miscellaneous topics such as matching corresponding points between two images of the same scene, removing outliers for robust computation, etc.

Armed with theoretical foundations, we provide a complete algorithm from theoretical and implementation perspective. We list the major tools and libraries we used in our implementation. We also provide a web interface for users to upload the images and view the results online.

Lastly, we apply our implementation to variety of datasets including standard benchmarks and provide qualitative and quantitative results of each dataset along with the effect on each hyper-parameter on final performance. We also comment on future directions that can be taken to further enhance reproducible research by implementing other pipelines of structure from motion.

Contents

1 Introduction **1**
1.1 Problem Domain 1
1.2 Research Problem Statement 1

2 Literature Review **1**
2.1 Camera Projections 2
2.2 Epipolar Geometry and the Fundamental Matrix 3
2.3 8 Point Algorithm for Fundamental Matrix Estimation 3
2.4 Essential Matrix and Extraction of Camera Matrices 4
2.5 Triangulation 5
2.6 Perspective-n-Point 6
2.7 Random sample consensus 7

3 Proposed Approach **8**
3.1 Feature Matching 8
3.2 Fundamental Matrix Estimation 9
3.3 Pose Estimation 9
3.4 Triangulation 9
3.5 Perspective-n-Point 9

4 Implementation **10**
4.1 Web Interface 10
4.1.1 Back-End Infrastructure 10
4.1.2 Front-End Design 11
4.1.3 Point Cloud Visualization 11

5 Results and Discussion **11**

6 Conclusions **14**

7 References **15**

A Singular Value Decomposition for Solving $Ax = 0$ **17**

1 Introduction

3D reconstruction from video has been an active research area in Computer Vision for the past few years. Some of this research also overlaps with Digital Photogrammetry [1], which involves estimating measurements from photographs. It finds extensive applications in numerous domains, such as medical imaging, robotics, computer graphics, virtual reality etc. This greatly reduces the effort required to assemble a digital map. A user could simply provide an extensive video of an area/scene and obtain the end result through an automated process. Recent applications include virtual tours of cities, tourism, and city planning.

1.1 Problem Domain

In the past, reconstruction was performed using photogrammetric techniques, in which images were manually processed. Due to a large amount of available data, this was a highly expensive and time consuming process. Therefore, this process had to be automated.

The proposed reconstruction process involves several stages, ranging from low-level to high-level processing [2]. A state of the art system would depend on the success of all these stages and in combining the end result. This process is made difficult from the fact that a lot of information is lost as images are projected to 3D space. Apart from this, we have to filter out the useful information and discard all irrelevant details.

1.2 Research Problem Statement

Given a video sequence of static environment, the task is to compute its 3D model/mesh. This problem can be formalized as follows: given m pixels in n images $x_{11}, x_{12} \dots x_{1n}, x_{21} \dots x_{2n}, x_{m1} \dots x_{mn}$, compute the camera projection matrix $P_1, P_2 \dots P_n$ for each image and 3D coordinates of each point $X_1, \dots X_m$ such that each 2D point-correspondences[1] across the images projects to one 3D scene point. This is known as **re-projection error** and its definition is as follows:

$$E = \sum_j^m \sum_i^n \left\| x_j - P_i X_j \right\|^2$$

The goal is to find a set of camera projection matrices $P_1, P_2 \dots P_n$, and 3-dimensional coordinates of each point $X_1, \dots X_m$ that minimize the re-projection error:

$$\min_{P_1, \dots P_n, X_1, \dots X_m} E$$

2 Literature Review

The following sections explain, in detail, the technicalities that deal with reconstruction of 3D environment from 2D images. The algorithms provided in these sections are explained using mathematical equations. Additional reading material is cited for a more in-depth understanding.

[1] 2D Point Correspondences: Set of 2D points, each in different image, which corresponds to the same point in 3D world.

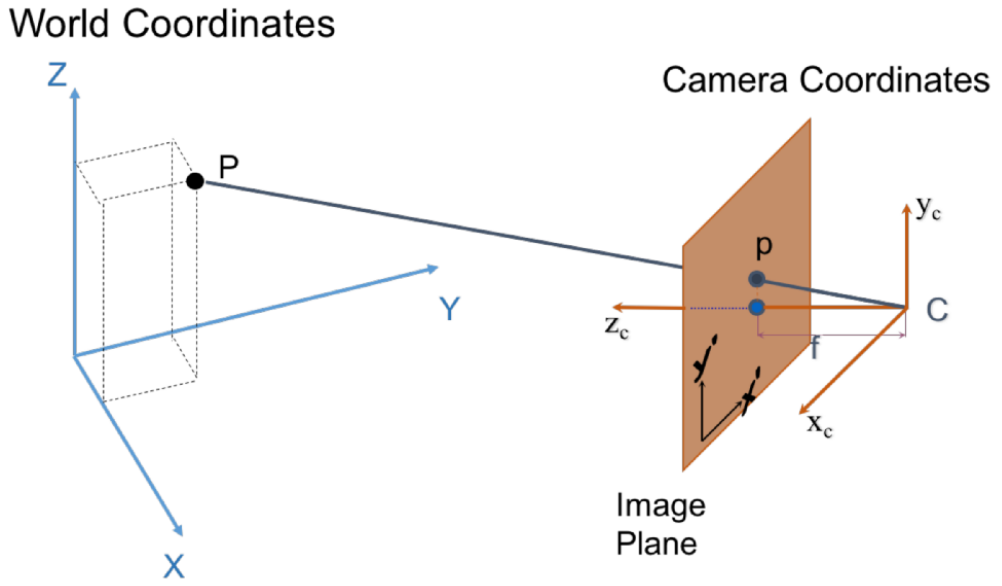


Figure 1: Camera Projections

2.1 Camera Projections

The camera model is responsible for transforming the visual representation of real 3D world into 2D camera pixels [3], [4]. As shown in Figure 1, the camera projection matrix P projects the world coordinate system into image plane i.e. pixel coordinate system which lies at $z = 1$.

Mathematically, the projection matrix P is a 3×4 matrix which can be further decomposed into two matrices as shown below:

$$P = K[R|t]$$

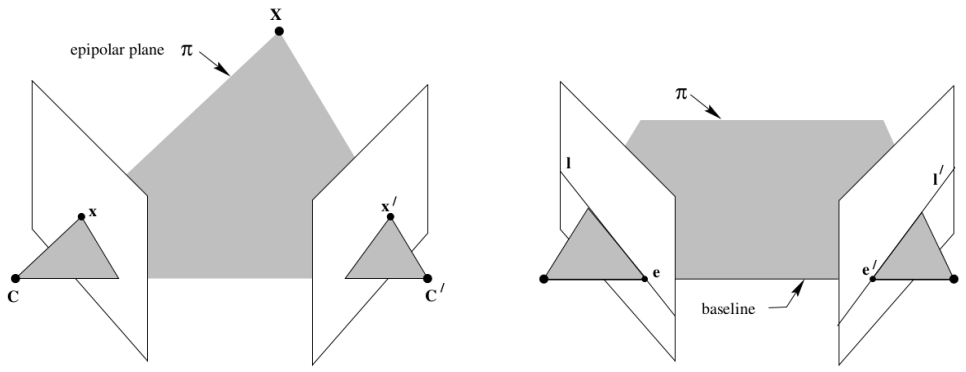
Where $[R|t]$ and K denote the extrinsic and intrinsic camera parameters, respectively. They are briefly explained below:

- **Extrinsic Parameters** are denoted by a 3×4 matrix (3×3 Rotation Matrix R concatenated with 3×1 Translation Vector t). Rotation matrix and translation vector encodes the camera orientation and position (w.r.t world coordinate frame), respectively. Collectively, it transforms 3D world in third person view to first person view. Note that transformation occurs in coordinate space and not the pixel space.

- **Intrinsic Parameters** are denoted with a 3×3 matrix K which encodes camera's internal properties, such as, lens' focal length, principal point, pixel scaling factor etc. It transforms the 3D world in first person view (provided by extrinsic transformation) to 2D pixel in first person view.

Now, given a projection matrix P and point X in 3D world coordinate system, it is mapped to a 2D pixel coordinate system x using the following relation:

$$x = \overbrace{K[R|t]}^{P} X$$



(a) The camera centres, 3-space point X, and its images x and x' lie in a common plane

(b) The camera baseline intersects each image plane at the epipoles e and e'

Figure 2: Epipolar Geometry

2.2 Epipolar Geometry and the Fundamental Matrix

Given two views/images, epipolar geometry is essentially the geometry of intersection between them [6]. For example, when a 3D point X is imaged in two views x and x', the camera centers C and C', image points x and x', and 3D point X lie in the same plane, as shown in Figure 2(a). These geometry principles aid in constraining our search space for finding point correspondences.

A few fundamental terminologies of epipolar geometry are defined as following:

- **Epipole** e is the point of intersection of the line joining the camera centres with the image plane (see Figure 2(b)).
- **Epipolar Plane** π is a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes.
- **Epipolar Line** l is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole (see Figure 2(b)).

The fundamental matrix F is a 2-rank 3×3 matrix which represents epipolar geometry. It describes the geometric relations that exist between two images of the same scene, and is estimated by the number of correspondences between them. A correspondence is a pair of points on the two images that are believed to be projections of the same 3D point. Fundamental matrix F satisfies what's called a Correspondence Condition [8]. It is stated as follows:
The fundamental matrix satisfies the condition that for any pair of corresponding points $x \leftrightarrow x'$ in the two images

$$xFx' = 0$$

2.3 8 Point Algorithm for Fundamental Matrix Estimation

As explained above and in [12], for all point correspondences x and x', It satisfies the following equation:

$$x'^T Fx = 0$$

Expanding and rearranging this equation yields the following formulation:

$$Ax = 0 \tag{1}$$

where

$$A = \begin{bmatrix} u_1^{(1)}u_1^{(2)} & u_1^{(1)}u_1^{(2)} & u_1^{(1)} & v_1^{(1)}u_1^{(2)} & v_1^{(1)}v_1^{(2)} & v_1^{(1)} & u_1^{(2)} & v_1^{(2)} & 1 \\ -- & -- & -- & -- & -- & -- & -- & & \\ u_8^{(1)}u_8^{(2)} & u_8^{(1)}u_8^{(2)} & u_8^{(1)} & v_8^{(1)}u_8^{(2)} & v_8^{(1)}v_8^{(2)} & v_8^{(1)} & u_8^{(2)} & v_8^{(2)} & 1 \end{bmatrix}$$

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

u_i^k = x-coordinate of ith point in kth image
v_i^k = y-coordinate of ith point in kth image
f_i^j = ith row, jth column entry of matrix F

The algorithm consists of the following steps:

8 Corresponding Points First, a total of eight 2D point correspondences between images are determined.

SVD (Singular Value Decomposition) To solve equation (1), we compute SVD of matrix $A = UDV^T$ and reshape the last column of V into a 3×3 matrix which represents the fundamental matrix F'. (See Appendix A for more details.)

Applying Rank Constraint The fundamental Matrix F' computed in the last step will not necessarily have rank 2 (which is one of the requirements for a valid fundamental matrix) due to noise present in the data. Thus, we now compute SVD of $F' = UDV^T$ and explicitly reduce the rank to 2 by setting the last diagonal element of D to zero to get D'. Finally, we get a valid fundamental matrix $F = UD'V^T$.

2.4 Essential Matrix and Extraction of Camera Matrices

The essential matrix is the specialization of the fundamental matrix to the case of normalized image coordinates. In fundamental matrix the assumption of calibrated cameras is removed, so it can be thought of as a generalization of the essential matrix. The essential matrix can be directly computed from the fundamental matrix using equation (2) from which the camera projection matrix is further computed [7], [5].

$$E = K'^T F K \tag{2}$$

For a computed essential matrix $E = Udiag(1, 1, 0)V^T$, and first camera matrix $P = [I|0]$, there are four possible choices for the second camera matrix P'

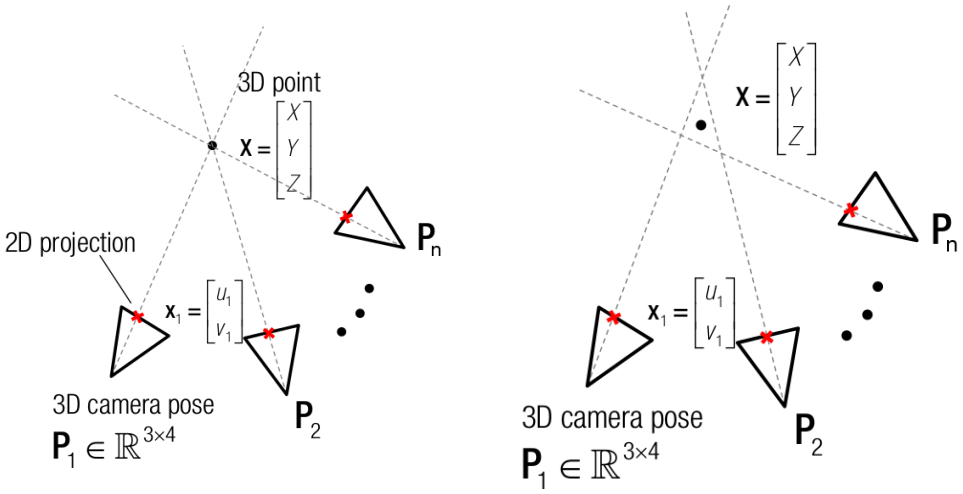
$$P' = [UWV^T| + u_3]$$
$$P' = [UWV^T| - u_3]$$
$$P' = [UW^TV^T| + u_3]$$
$$P' = [UW^TV^T| - u_3]$$

where

$$u_3 = \text{Third column of matrix } U$$
$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.5 Triangulation

Triangulation is a problem of estimating a 3D point X, given its point correspondences $x_1, x_2, \ldots, x_n$ and camera matrices $P_1, P_2, \ldots, P_n$ computed for each image $I_1, I_2, \ldots I_n$ [9]. There are generally two approaches followed to solve this problem: Linear and Nonlinear; both of which are explained below:



(a) Ideal Case (b) Pratical Case

Figure 3: Triangulation Setups

Linear Triangulation: As the name suggests, linear triangulation tackles this problem assuming a linear relationship between the variables since, as shown in Figure 3(a), all 3D rays intersect at the same point. Ultimately, solving this problem translates to solving the following equation.

$$\begin{bmatrix} \begin{bmatrix} x_1 \\ 1 \end{bmatrix}_{\mathbf{x}} P_1 \\ \begin{bmatrix} x_2 \\ 1 \end{bmatrix}_{\mathbf{x}} P_2 \\ \vdots \\ \begin{bmatrix} x_n \\ 1 \end{bmatrix}_{\mathbf{x}} P_n \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = 0 \tag{3}$$

Nonlinear Triangulation: As shown in Figure 3(b), in the presence of noise these rays can not be guaranteed to cross, so this becomes a difficult problem. Thus, a different formalization is used to solve the problem i.e mini-

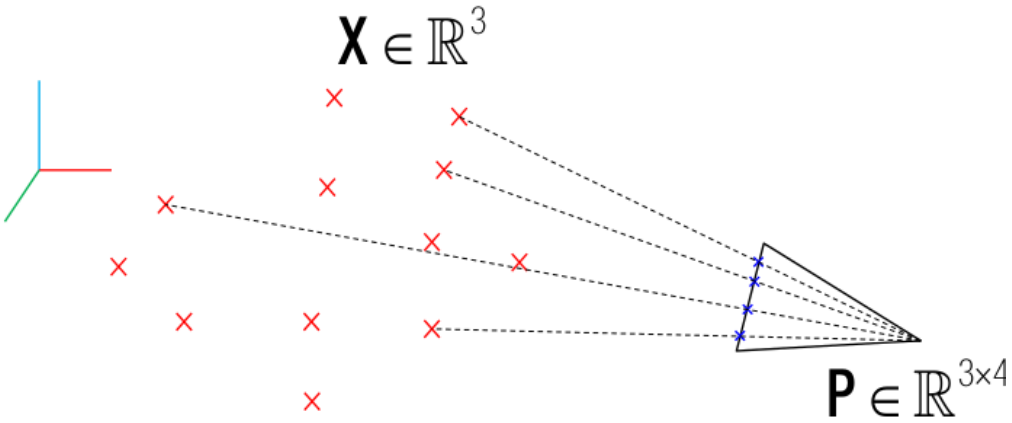


Figure 4: Perspective-n-Point Setup

mize the re-projection error.

$$\min_{X} \sum_{i}^{n} [(x_i - u_i/w_i)^2 - (y_i - v_i/w_i)^2] \tag{4}$$

Where

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = KR_i(X - C_i)$$

C_i = ith Camera Optical Center
R_i = ith Camera Orientation

Variables are updated iteratively as following:

$$X_{new} = X_{old} + \delta X \tag{5}$$

Where

$$\delta X = (J^T J)^{-1} J^T (b - f(X)) \tag{6}$$

$$b = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & .. & x_n & y_n \end{bmatrix}^T \tag{7}$$

$$f(X) = \begin{bmatrix} u_1/w_1 & v_1/w_1 & u_2/w_2 & v_2/w_2 & .. & x_n/w_n & v_n/w_n \end{bmatrix}^T \tag{8}$$

$$J = \begin{bmatrix} (\partial f_1/\partial X)^T & (\partial f_2/\partial X)^T & .. & (\partial f_n/\partial X)^T \end{bmatrix} \tag{9}$$

2.6 Perspective-n-Point

Perspective-n-Point is a problem of estimating camera projection matrix P, given a set of $X \leftrightarrow x$ correspondences and camera intrinsic parameters K [10], as shown in Figure 4.

Although, there are multiple approaches to solving this problem, only the standard algorithm is briefly explained here for brevity.

Linear Standard Method: As suggested by its name, this method assumes linear relationship among the variables and devises a solution based on this assumption. According to this method, solving PnP translates to finding solution to the following relation:

$$\begin{bmatrix} 0_{1\times4} & -X^T & vX^T \\ X^T & 0_{1\times4} & uX^T \\ -vX^T & uX^T & 0_{1\times4} \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0 \tag{10}$$

Where $x = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$, and P_i is the ith row of camera projection matrix P. Now, given that P is estimated, R and t is further computed as follows:

$$R' = K^{-1}P_{1:3} \tag{11}$$

$$R = UV^T \tag{12}$$

$$t = K^{-1}P_4/\sigma_1 \tag{13}$$

where $R' = UDV^T$ and $D = diag(\sigma_1, \sigma_2, \sigma_3)$. Note that equation (12) is used to enforce orthogonality constraint of rotation matrix R.

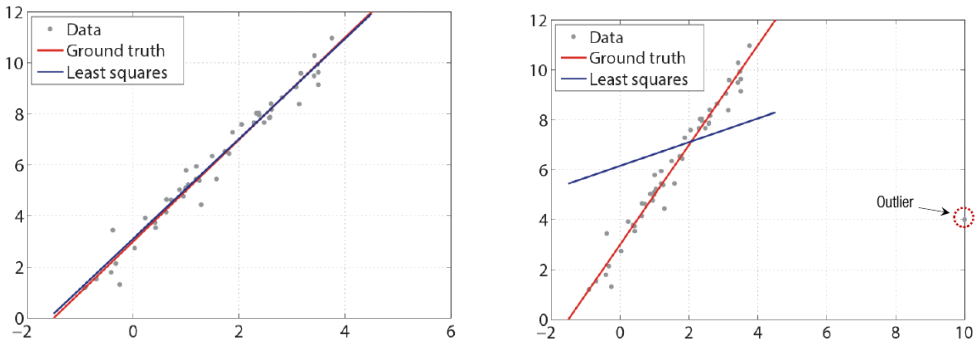
2.7 Random sample consensus

Random sample consensus (RANSAC) is an outlier detection method used to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates [13]. We use this method for fitting a line such that it passes through a maximum number of points.

There are several other methods apart from RANSAC, such as the least squares method. For line fitting, we are given with a set of x and y points that lie close to the line. Our goal is to fit a line represented by a three parameter family of homogeneous points on the line, e, f, and g. This can be written as a linear combination where each row contains the homogeneous coordinates of the point, which is repeated N times, i.e., number of points, as shown in equation (14). This can be solved using SVD where we decompose the vector A into a three column vector $\begin{bmatrix} V_1 & V_2 & V_3 \end{bmatrix}$. The last column of this vector gives us the solution.

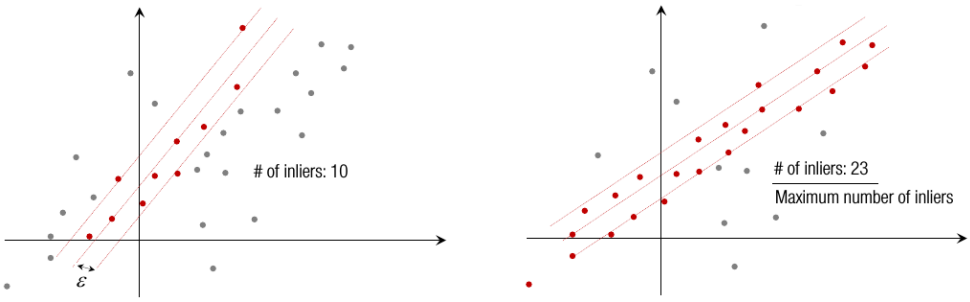
However, this method is extremely sensitive to outlier points and can thus fit a line that is off course. This is because the cost function we are minimizing is the distance between the point and the line. This is shown in Figure 5.

$$E = \left\| \begin{bmatrix} x_1 & y_1 & 1 \\ . & . & . \\ . & . & . \\ . & . & . \\ x_N & y_N & 1 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \end{bmatrix} \right\|^2 = \|Ax\|^2 \tag{14}$$



(a) Outlier absent (b) Outlier present

Figure 5: Least squares method



(a) (b)

Figure 6: RANSAC method

RANSAC solves this problem by removing outlier points if its error is greater than a certain threshold ε. The strategy here is to find a model that accords with the maximum number of samples. It operates by randomly sampling points that belong to a group. If p is the probability of each point belonging to a group, then p^k is the probability of k times while remaining in the same group. As we want this probability to be high, we sample only two points. Next, we find the number of inlier points. This procedure is repeated until we find a set of points that contain the maximum number of inliers. The probability of choosing an inlier is given by:

$$w = \frac{\#\ of\ inliers}{\#\ of\ samples} \tag{15}$$

This refined approach gives us much better immunity against outliers, as shown in Figure 6.

3 Proposed Approach

The steps involved in the reconstruction process are as follows.

3.1 Feature Matching

As explained above, a set of 2D point correspondences are required for various steps. First, two video frames are selected and 2D point correspondences are

found using an automated algorithm. Specifically, we used SIFT (Scale Invariant Feature Transform) [11] algorithm for computing features across the two images. These features are matched using nearest neighbor technique (L2 norm is used as a distance metric). Examples are shown in Figure 9(c) and 10(c).

3.2 Fundamental Matrix Estimation

Feature matching algorithms often match wrong correspondences that adds an overhead for removing outliers points i.e false correspondences. Due to this reason, the 8 point algorithm is used in conjunction with random sample consensus (RANSAC). Precisely, the algorithm proceeds as follows:

1. Pick any 8 random sample points and assume them to be correct correspondences.
2. Run 8 point algorithm (detail in section 2.3) using the points picked in previous step.
3. Count the number of inliers.
4. Go to step 1 unless number of pre-specified iterations are completed.

Finally, the fundamental matrix F having the most inliers is assumed to be the correct one.

3.3 Pose Estimation

Once the fundamental matrix F is computed, equation (2) is used to compute essential matrix to ultimately compute four possible camera projection matrices. The ambiguity among the four matrices is resolved through point triangulation i.e checking which P among the four projects *all* 2D image points in-front of them ($z >= 0$).

3.4 Triangulation

Now that we have point correspondences $x_1, x_2, \ldots, x_n$ (corresponding to the same 3D point in real scene) and camera matrices $P_1, P_2, \ldots, P_n$ computed for each image $I_1, I_2, \ldots I_n$, we can use equation (3) to solve for X.

We use singular value decomposition of the left matrix to get UDV^T, where the last column of V is the back-projected 3D point.

3.5 Perspective-n-Point

SIFT features of the next frame are computed again to ultimately match 2D feature coordinates with 3D points already triangulated (in previous step).
Now that we have a set of $X \leftrightarrow x$ correspondences and camera intrinsic parameters K, we can use equation (10) to solve for camera projection matrix P. And we do so using, again, singular value decomposition as explained in section 2.6.

4 Implementation

Our implementation can be broadly divided into the following two modules:

Video Transfer Video stream is captured from an Android smartphone over a localhost connection, using an Android application named `IP Webcam`, which is then transfered to a host computer. To extract individual frames, `FFmpeg` is used.

Structure from Motion Pipeline The source code for structure from motion pipeline is written in `Python`. We primarily used the `OpenCV` library for most of our computational tasks. The plotting is done using `Matplotlib`.

An algorithmic description of our process is given below. All these step are performed using the `OpenCV` library:

1. **Feature Matching** Given two video frames, their 2D point correspondences are found using the SIFT algorithm. The `detector.detectAndCompute` function serves this purpose.
2. **Fundamental Matrix Estimation** The output from our previous step is refined using the 8 point algorithm, which gives us the fundamental matrix with most inliers. This function is named `cv2.findFundamentalMat` in `OpenCV`.
3. **Pose Estimation** To estimated the camera pose, we use the `cv2.recoverPose` function. The ambiguity among the output is resolved using point triangulation.
4. **Triangulation** Using the `cv2.triangulatePoints` function, we apply SVD on our previous output to get UDV^T, where the last column of V is the back-projected 3D point.
5. **Perspective-n-Point** In the last step, we estimate the camera projection matrix P using the `cv2.solvePnPRansac` function.

4.1 Web Interface

The front-end web interface for our project is made using the Django web framework. It allows the user to upload image sequences (JPG, PNG format) which are sent to the server for 3D reconstruction. Once the reconstruction process is finished, the result is visualized using the `Three.js` JavaScript library. We use Google Cloud for hosting the website, which can be accessed through a public IP address.

There are three pages provided by the website, namely, the Home page, How it Works page, and the Team page.

4.1.1 Back-End Infrastructure

To keep track of user images, we store the data in an SQLite database. It consists of the following fields: `session_id`, `file_name`, `file_size`, `file_format`, and `date_created`. The actual image data is stored in a separate directory and `file_name` is be used to query the image. For loading the images relevant to a particular user, the `session_id` value is used, which has a unique value for each user session.

4.1.2 Front-End Design

A screenshot of our project's homepage is provided in Figure 7.

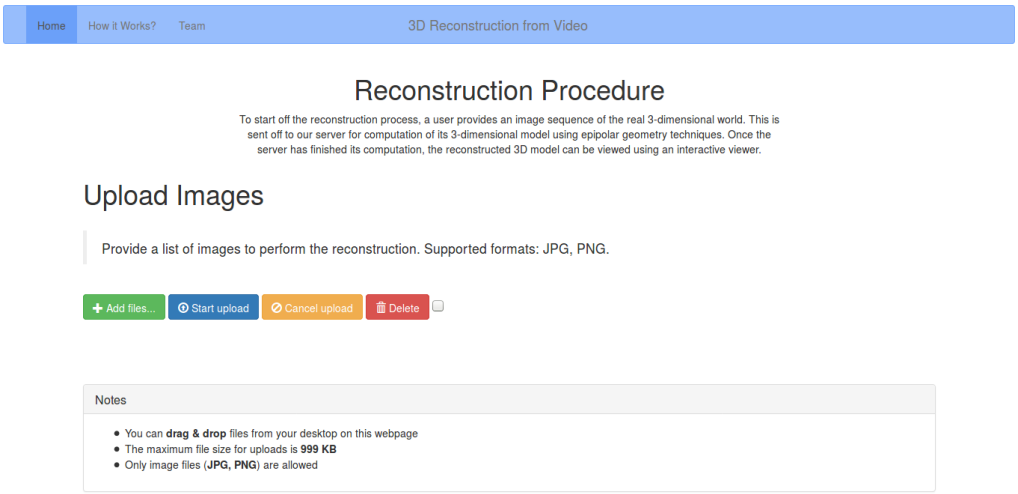


Figure 7: Homepage

4.1.3 Point Cloud Visualization

As mentioned earlier, we make use of the `Three.js` JavaScript library for visualizing the end result, which is in the form of a point cloud. The library supports adding colors to the point cloud, and provides further controls for interacting with the points, such as zooming in and out, rotating and moving the camera. Figure 8 shows this in action.

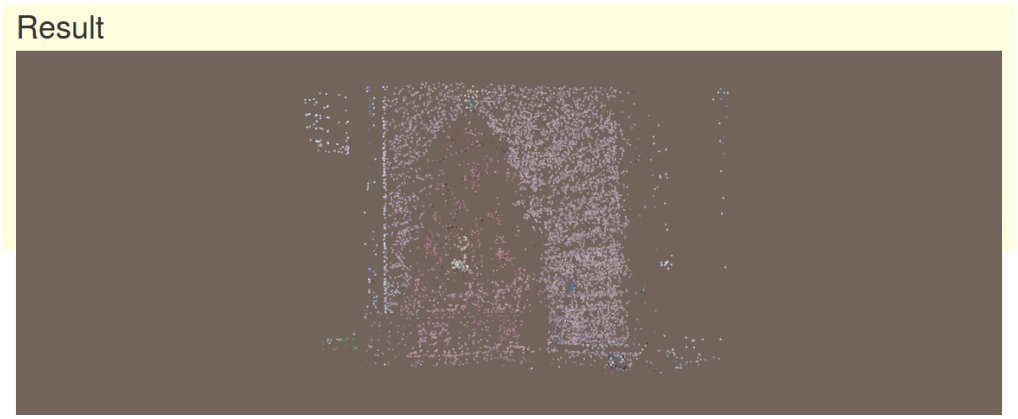


Figure 8: Interactive Point Cloud Viewer

5 Results and Discussion

We applied the above mentioned techniques to a variety of videos (shot in 30 frames per second) in both indoor and outdoor environments. The qualitative results are shown in figure 9 and 10.

For evaluating our system, we calculate the re-projection error and number of SIFT keypoint matches, as shown in Table 1.

	Statistics		
Dataset	SIFT keypoints count	Inliers count	Ratio
Castle (frame 19)	4000	1221	0.30525
Castle (all frames)	4000	1054.5925	0.26364
Car (frame 6)	4000	2271	0.56775
Car (all frames)	3935.9009	1239.9369	0.31503
Building (all frames)	3606.4721	1382.4163	0.38331
Boot (all frames)	744.3125	54.0625	0.07258
Average	3381.1142	1203.8347	0.31792

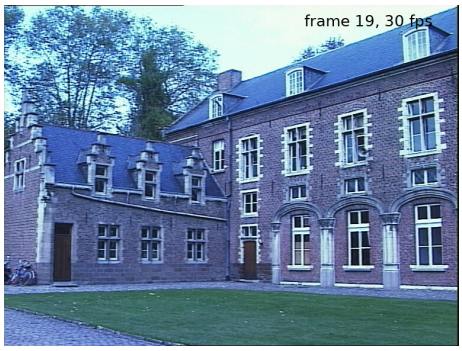
Table 1: Number of matches

The table above lists two important entities; number of SIFT keypoint matches and the inliers count. These values can be used as an evaluation metric for our system, as their ratio should approach 1, which will give us better results.

Another evaluation metric we use is the re-projection error, which is a numeric value corresponding to the image distance between a projected point and a measured one. Minimizing the re-projection error can be used for estimating the error from point correspondences between two images. This is shown below in Table 2.

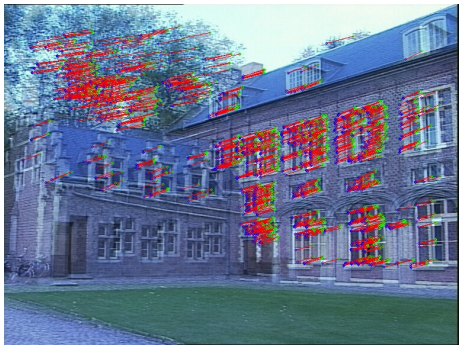
Dataset	Average error
Castle (frame 19)	308394.2187
Castle (all frames)	133857.5355
Car (frame 6)	60877.8515
Car (all frames)	569505.4980
Building (all frames)	3976174.3395
Boot (all frames)	42686.0363

Table 2: Re-projection error

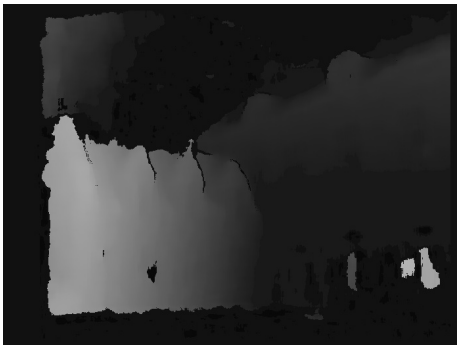


(a) Input image

(b) Rectified



(c) SIFT keypoint matches



(d) Disparity map

Figure 9: Castle

(a) Input image

(b) Rectified

(c) SIFT keypoint matches

(d) Disparity map

Figure 10: Car

6 Conclusions

Using a monocular camera, we presented a system which takes input in the form of images and applies all the necessary steps for reconstructing a 3D model. The approach estimates camera pose, SIFT keypoint matches, and disparity map.

To allow for the correct fusion of resulting images and to better generalize our system, we calculate the re-projection error which allows us to evaluate our system. Our system does not require any initialization of scene structure and works for all input types, i.e., both outdoor and indoor environments.

A web interface was created which allows users to feasibly use our system. The resulting point cloud was visualized on our web page using a JavaScript viewer.

7 References

[1] Nuttens, Timothy, Alain De Wulf, Rudi Goossens, et al., "Digital Photogrammetry Used for the 3D Reconstruction of the Walls Around the Acropolis of Titani (Greece)", 2010.
https://biblio.ugent.be/publication/1044358

[2] Robert Collins, "Stereo Reconstruction: Steps to General Stereo", 2007.
http://www.cse.psu.edu/~rtc12/CSE486/lecture21_6pp.pdf

[3] Augusto Román, "Multiperspective Imaging for Automated Urban Visualization", 2006.
https://graphics.stanford.edu/papers/aroman_thesis/UrbanMPI-thesis.pdf

[4] Hartley, Richard, and Andrew Zisserman, "Multiple View Geometry in Computer Vision", 2nd ed. Cambridge University Press, pp. 153-176, 2004.

[5] Darius Burschka, Elmar Mair, "Direct Pose Estimation with a Monocular Camera", 2017.
https://arxiv.org/pdf/1709.05815.pdf

[6] Hartley, Richard, and Andrew Zisserman, "Multiple View Geometry in Computer Vision", 2nd ed. Cambridge University Press, pp. 239-241, 2004.

[7] Hartley, Richard, and Andrew Zisserman, "Multiple View Geometry in Computer Vision", 2nd ed. Cambridge University Press, pp. 257, 2004.

[8] Hartley, Richard, and Andrew Zisserman, "Multiple View Geometry in Computer Vision", 2nd ed. Cambridge University Press, pp. 247, 2004.

[9] Richard Hartley, Peter Sturm, "Triangulation", Springer-Verlag, 970, pp. 190-197, 1995, Lecture Notes in Computer Science (LNCS).
http://www.springerlink.com/content/7027546138383587

[10] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang and Hang-Fei Cheng, "Complete solution classification for the perspective-three-point problem", in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 8, pp. 930-943, Aug. 2003.
http://ieeexplore.ieee.org/document/1217599

[11] D. G. Lowe, "Object recognition from local scale-invariant features", Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, 1999, pp. 1150-1157 vol.2.
https://ieeexplore.ieee.org/document/790410/

[12] R. I. Hartley, "In defense of the eight-point algorithm", in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 6, pp. 580-593, Jun 1997. In Defence of the 8-point Algorithm, Richard I. Hartley, 1997
https://ieeexplore.ieee.org/iel1/34/13258/00601246.pdf

[13] Yaniv Z. "Random Sample Consensus (RANSAC) Algorithm, A Generic Implementation", Oct. 2010.
http://www.insight-journal.org/browse/publication/769

A Singular Value Decomposition for Solving $Ax = 0$

Often, we need to solve $Ax = 0$, but due to noise in the data, this equation does not exactly hold in practice. Due to which the problem reduces to

$$\min_x \|Ax\| \tag{16}$$

Thus, we try to find, not the exact nullspace of A, but rather a vector x close to nullspace. This vector corresponds to the row-space vector of A which contributes minimally in matrix A construction. Thus

$$x = V(:, end) \tag{17}$$

Where

$$A = UDV^T \tag{18}$$