EECS 442: Final Project

Structure From Motion

Kevin Choi
Robotics

Ismail El Houcheimi
Robotics

Yih-Jye Jeffrey Hsu
Robotics

***Abstract*—In this paper, we summarize the method, and results of our projective model structure from motion implementation. Our implementation produces 3D point clouds from a structured sequence of images, such as a video, that look similar to the real world environment. We detail the process used to estimate camera poses and triangulate 3D points from 2D pixels. Our results show that small errors in camera pose estimation accumulates as the sequence of scenes grows which results in large errors in triangulation when using more than three camera frames. Bundle adjustment is suggested to correct for camera pose and triangulation errors.**

I. Introduction

Recovering a 3D model from camera images is an important problem to solve since it simplifies the creation of a virtual environment and localizes the camera relative to the environment. Structure from motion can be used to localize robots and map environments for autonomous navigation, aid in building CAD models, or model archaeological sites for research purposes.

Using digital cameras to reconstruct a 3D model is an attractive solution since they are widely available and low cost. An attempt will be made to replicate Pollefeys paper Visual Modeling with a Hand-Held Camera [2], where a sequence of uncalibrated camera image sequences is used to recreate a 3D model.

The method is based on tracking or matching features between different frames and using these features to find geometric relations between different views. The motion of the camera and the structure of the scene can thus be found. However, an ambiguity on the reconstruction is still present. An initial scale was assumed in the initial frames and then the other reconstructed points were scaled to match that scene.

II. Related Work

Structure from motion has been a popular topic in computer vision for almost two decades. Several different works using either an affine camera model [1] or a projective camera model [2] have been around for years. While many new improvements in both the concept and numerical processing have been made through the years, the basic concept remain more or less the same. Therefore, we chose to reimplementat Pollefeys paper [2] because its approach with projective camera model allows it to work without specific geometry constraint.

III. Methods

A. Preprocessing

Videos were shot at 15 frames per second and then every fifth frame was extracted to be used. This allows for enough change to be perceived between the frames. All images are converted to grayscale before any further analysis.

B. Feature Detection

In order to reconstruct a 3D model of the scene, image features must first be extracted from the different frames. Interesting points were found in each image using the Harris Corner Detector and Speeded Up Robost Feature (SURF) algorithms, which are both built into MATLAB's computer vision toolbox. These functions return the pixel location of interesting points along with their descriptors. Finally we match points from different images using their descriptors and MATLAB's ExtractFeatures() and MatchFeatures() functions.

C. Feature Matching

Our first approach to feature matching was to match features between pairs of consecutive frames from the video. These matched points are then used to find the fundamental matrix and are later used for reprojection and construction of the 3D model.

In order to improve the obtained results, we tracked features across different frames and used all these frames to reproject the points into the 3D scene.

In what follows, "scene" is a set of two consecutive frames in which we match features. A scene consists of two pictures taken at two different camera poses.

D. Fundamental Matrix

A fundamental matrix relates the corresponding pixels between two images and captures geometric information about the camera poses and parameters [3]. To gain a more intuitive understanding of the relationship between two corresponding pixels we first look at the representation of x on the second image plane.

x and x' are the projections of 3D point X on to the first and second image plane respectively($x = PX,\ x' = P'X$). The reprojection of x into 3D space is a line that passes through camera center O_1 and X ($\overline{O_1X}$). From Fig. 1, we see that the projection of camera center O_1 on image plane two is the epipolar point e_2. The projection of $\overline{O_1X}$ on to the second image plane is a line that passes through e_2 and x' and is known as the epipolar line of x. The Fundamental (F) Matrix

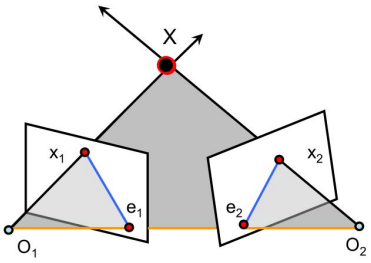


Fig. 1

transforms the homogeneous point x from the first image plane to the homogeneous line $\overline{e_2x'}$ on the second image plane; this line intersects x' so $x' \cdot \overline{e_2x'} = 0$. For pixel corresponding pixels x and x' from the first and second images respectively, the Fundamental Matrix satisfies the the following:

$$x'Fx = 0 \tag{1}$$

n matching pixel points, (u_i, v_i) (first image) and (u_i', v_i') (second image), are used to find the Fundamental Matrix between the images using Eq. 2.

$$\begin{bmatrix} u_1'u_1 & u_1'v_1 & u_1' & v_1'u_1 & v_1'v_1 & v_1' & x_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n'u_n & u_n'v_n & u_n' & v_n'u_n & v_n'v_n & v_n' & x_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ \vdots \\ f_{21} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{2}$$

Using our matching points in Eq. 2 without normalization or filtering incorrect matches yields epipolar lines that have L2 norm errors (distance between epipolar line and matching point) greater than 10% the length of the image. We should reduce this error since the rotation and translation of each pose is derived from the Fundamental matrices. First, we normalize the matching pixels such that the centroid of the points are located at the center of the image and the mean squared distance to the centroid is two pixels. Next, we use the Random Sampling Consensus (RANSAC) iterative method to find the best "matching" Fundamental Matrix for each scene. Eight random points are used to compute the fundamental matrix; epipolar lines l_n that are within five pixels of their respective pixel (u_n', v_n') are given a vote. We repeat this process for 1000 iterations and the fundamental matrix with the most votes is selected as the optimal Fundamental Matrix for the scene.

E. Camera Matrix

The camera matrix P projects the 3D homogeneous point X_n onto the image plane to form the 2D homogeneous point, and is composed of the camera intrinsic and extrinsic parameters. The intrinsic matrix K describes the geometric properties of the camera while the extrinsic matrix M describes the pose of the camera relative to the world frame. We took images of a checkerboard with known dimensions in different poses and used the Camera Calibrator Tool in Matlab to find the intrinsic camera matrix. The intrinsic camera and Fundamental matrices are used to calculate the Essential Matrix from which two rotation and translation matrices can be derived [3]. In practice, the Essential Matrix is not rank 2 due to noise in the image so we enforce this characteristic using Eq. 6. This process is repeated for each scene so that we have four combinations of rotations and translations for every pair of consecutive images.

$$P = KM = K[R|t] \tag{3}$$

$$E = K'FK \tag{4}$$

$$E = USV^T \tag{5}$$

$$S \to S' = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad E' := US'V^T \tag{6}$$

Of the four candidates of R and t (Eq. 8 and Eq. 9) there is only one combination where the triangulated 3D points lie in front of both cameras. In practice, none of the combinations produce triangulated points that only lie in front of both cameras because the Fundamental matrix found from RANSAC does not satisfy Eq. 1 for all points. We pick the pose that generates the most number of points that lie in front of both cameras.

$$E' =: U'S''V'^T \tag{7}$$

$$R_1 = UWV^T \quad R_2 = UW^TV^T \tag{8}$$

$$t_1 = U(:,3) \quad t_2 = -U(:,3) \tag{9}$$

F. Triangulation and Scaling

1) Linear Triangulation Using Two Frames: As shown in Fig. 1, triangulation is the reverse process that reprojects 3D point X from x and x' in the first and second image plane. Ideally, X should be at the intersection of $\overline{O_1x_1}$ and $\overline{O_2x_2}$. However, in practice this is usually not true. One can see in Fig. 2,with noise in images and matching position, $\overline{O_1y_1'}$ and $\overline{O_2y_2'}$ do not intersect each other.

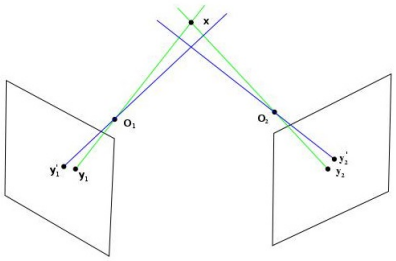


Fig. 2: Error in Triangulation

Triangulation then becomes a process that finds a least square solution for 3D point X that minimizes the $||error||_2$ from X to both $\overline{O_1x_1}$ and $\overline{O_2x_2}$. From [3], we see that the problem can be formed into Eq. 10.

$$u(p^{3T}X) - (p^{1T}X) = 0 \tag{10a}$$
$$v(p^{3T}X) - (p^{2T}X) = 0 \tag{10b}$$
$$u(p^{2T}X) - v(p^{1T}X) = 0 \tag{10c}$$

,where X is a 3D point, and x and y are the corresponding projection location in a image plane with projection matrix P. p^{iT} is i^{th} row of P. One can see Eq. 10 the relation between 3d point and projected point. To solve X, the following equation is formed from Eq. 10.

$$AX = 0 \tag{11}$$

,where

$$A = \begin{bmatrix} up^{3T} - p^{1T} \\ vp^{3T} - p^{2T} \\ u'p'^{3T} - p'^{1T} \\ v'p'^{3T} - p'^{2T} \end{bmatrix}$$

, x, y,and p are image coordinate and projection matrix from image 1 and x', y',and p' are image coordinate and projection matrix from image 2.
Use SVD to solve Eq. 11.The eigen-vector associated with the smallest σ is the solution to X.

2) Linear Triangulation Using Multiple Frames: Since camera pose estimation only produces direction of translation. The translations from different scenes are not in scale with each other. In order to match 3D points from different scene together, translations need to be rescaled. In Fig. 3, one can see that the projects of a 3D point in two scenes do not match up with each other before scaling. Ideally,

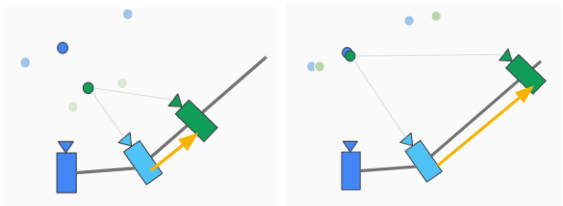
$$X^1 = sX^2 \;\; X^1, X^2 \in R^3 \;\; s \in R^1 \tag{12}$$

However, because of noise in R, there is usually no K that satisfies Eq. 12. Instead, normal equation as Eq. 13 is used to solve k that minimizes error between X^1 and X^2.

$$s = < X^1, X^1 >^{-1} < X^2, X^1 > \tag{13}$$

Since camera matrices from different scenes are based on local coordinate system of first camera in each scene , in order to combine all projections into one, all camera matrices need to be in the same coordinate system. We choose the coordinate system of the first scene as the universal frame system, and use Eq. 14 to form P_i^{world}

$$P_i^{world} = Kt_1R_1t_2R_2 \cdots t_iR_i \tag{14}$$



(a) Before Scaling (b) After Scaling

Fig. 3: Translation Scaling

With P_{world} in hand, we now can perform the triangulation process from Sec. III-F1 again to project every point into same reference system.

G. Batch Triangulation

After acquiring P_{world}, 3D point projections can be optimized by reprojecting from all the images that a 3D point is visible in [3][4]. Before doing batch triangulation, a visibility matrix V needs to be constructed first.

$$V_{ij} = \begin{cases} 0 & \text{if 3D point } i \text{ is not visible in image } j \\ 1 & \text{if 3D point } i \text{ is visible in image } j \end{cases}$$

Then using Eq. 15 and the triangulation result from Sec. III-F2 as initial guess for non-linear solver to find a best estimation that minimize the error in all the scenes.

$$\underset{\{C_i,q_i\}_{i=1}^{I},\{X\}_{j=1}^{J}}{minimize} \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij} \left(\left(u_{ij} - \frac{P_{i1}^{\top}\tilde{X}_j}{P_{i3}^{\top}\tilde{X}_j} \right)^2 + \left(v_{ij} - \frac{P_{i2}^{\top}\tilde{X}_j}{P_{i3}^{\top}\tilde{X}_j} \right)^2 \right) \tag{15}$$

$P_i$is the camera matrix for camera i
$q_{ij} = (u_{ij}, v_{ij})$ is the image coordinate of 3D point j (if exists)
X_j is the 3D coordinate ofX_j

H. Pixel Color

The reconstructed 3D points were colored by extracting the pixel color from the matched points and assigning that color to the point.

IV. Results and Discussion

A. Equipment

- Motorola Nexus 6
- Sony α77 with Sony Carl Zeiss Vario-Sonna T* DT 16-80 mm F3.5-4.5 ZA with focal point locked at 16mm and focus locked at ∞

Originally, Nexus cell phone was used to take our sample pictures. However, due to the constraints on the camera, the images are dim and not as sharp as we would like them to be. Moreover, we can't disable auto-focus to lock the focal point of the camera. We switched to a Sony DSLR for taking image samples and it provided less distortion and better images overall.

B. Feature Detection and Matching

The SURF algorithm detects more features than the Harris Corner algorithm, as seen in Fig. 4 and 5, which is useful for visualizing our triangulated points as it is difficult to discern objects in a sparse point cloud. In our initial testing, the pictures we used were blurry and low resolution which resulted in very few detected features when using Harris Corner Detector; in some cases it was less than eight points, which is the minimum to find a Fundamental Matrix. With the image test set presented in this paper, using the SURF feature detector returned an average of 800 matched points between pairs of images. Some matches were erroneous and lead to a Fundamental matrix that did not match the actual points. We got rid of these matches in our RANSAC algorithm.

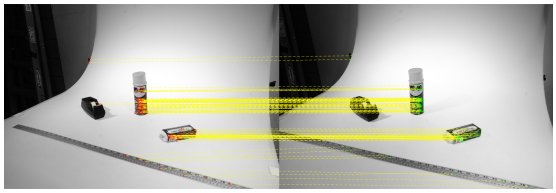


Fig. 4: Matching features found with Harris Corner Detector

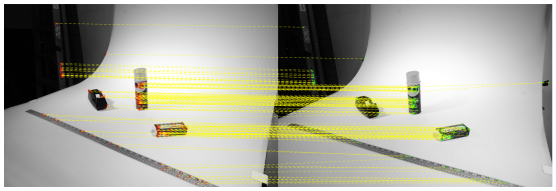
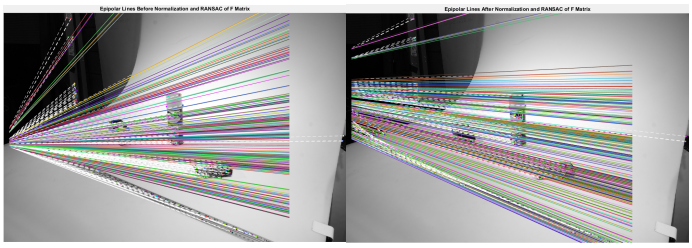


Fig. 5: Matching features found with SURF Detector

C. Fundamental Matrix

Initially we used the Fundamental Matrix calculated from the over-determined version of Eq. 2 but realized that a small number mismatched corresponding points caused large L2 norm errors between epipolar lines and their corresponding points. To eliminate the affect of outliers in the calculation of the Fundamental Matrix for each scene, we used RANSAC. As seen in Fig. 4, the epipolar lines yielded from the RANSAC method pass much closer to their respective points. We also optimized the number of RANSAC iterations we ran to reduce computation time for the Fundamental Matrix. The rate of inliers detected logarithmically decreases as the number of iterations increase while the time to perform the calculations linearly increases. From Fig. 7 we see that iterations beyond 500 gain less than 1% more inliers for every 100 additional iterations. We run our RANSAC using 200 iterations to find the Fundamental Matrix for each scene, which provides about 78% inliers and takes eight seconds for each scene (8 CPUS @ 3.5 GHz).



(a) Overdetermined equation (b) Normalization & RANSAC

Fig. 6: Solving the F matrix using a overdetermined equation results in large epipolar line errors. A majority of epipolar lines come within 5 pixels of their matching point when normalization and RANSAC is used to find the F matrix

Lastly we filter our list of matched points by removing the outliers (points more than five pixels from their epipolar line) which is about 25% of each list. We remove more than just mismatched points because points with large errors to their epipolar line will have poorly triangulated 3D points.

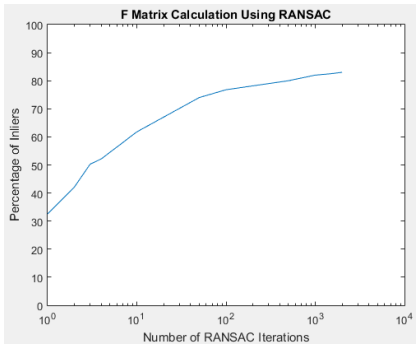


Fig. 7: RANSAC is most effective between 100 to 500 iterations. Beyond this, there is less than 1% improvement per 100 additional iterations

D. Triangulation and Scaling

1) Linear Triangulation Using Two Frames: Fig. 9-a and b are the example images used for our two frame linear triangulation demonstration. Using camera poses from previous section, we perform triangulation using Eq. 11. Nearly all four candidate poses of each scene (two rotations and translations) show points behind the camera so we choose the pose with the most number of points in front of the camera. From Fig. 12, we see that our method chooses the correct pose as all our cameras are pointing towards the real world scene and the cameras are translating in the $-x$ direction. Fig 8 shows the calculated 3D point cloud using two frames.

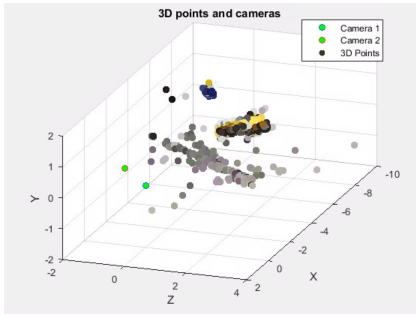


Fig. 8: 3D point cloud generated using two images.

2) Linear Triangulation Using Multiple Frames: Fig. 9-a,b, and c are the example images used for our three frame linear triangulation demonstration. As mentioned in Sec. III-F2, the first step for multi-frame triangulation is to scale each scene's translation vector such that common triangulated points are matched across all scenes. Fig. 10 shows the camera poses before and after scaling.

We then perform triangulation using the P_{world} acquired from the previous section. Fig. 11 shows the resultant point cloud. Note that several points that belong to same object are far apart in our generated point cloud. This is due to image noise

and error in camera pose estimation (rotation and translation for each scene) which causes triangulation error.

(a) Frame 1 (b) Frame 2 (c) Frame 3

Fig. 9: Image Test Set

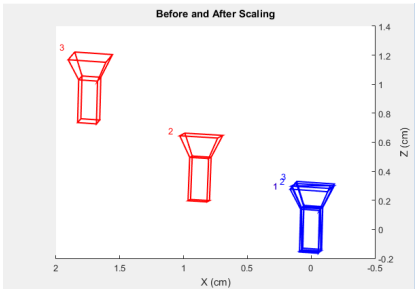


Fig. 10: Camera poses before and after scaling. Blue cameras show the camera poses before scaling. Red cameras show the camera poses after scaling

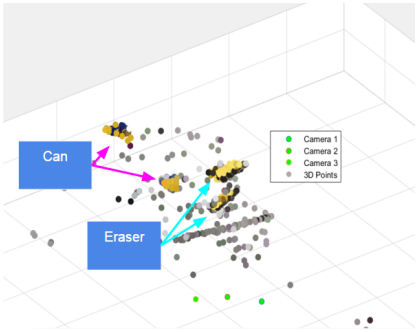


Fig. 11: Three Frame Triangulation Result. Can and eraser each form two separate point cloud.

E. Batch Triangulation

From Sec. IV-D2, it is clear that performing triangulation naively across scene only generates disperse 3D clouds. In order to bring 3D projections of same point close together, we perform batch triangulation using Eq. 15 to produce one best projection estimation for a single point across several scenes. Fig. 12 shows the result of batch triangulation using the same example from Sec. IV-D2. Notice that compared to Fig. 11, the point clouds for the Can and Eraser are closer together.

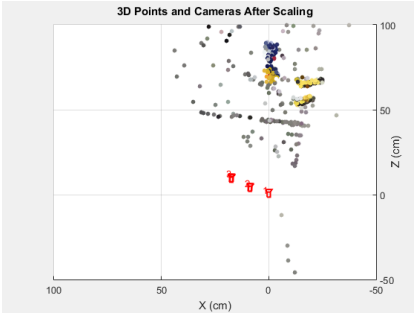


Fig. 12: Batch triangulation result using three cameras from a bird's eye view

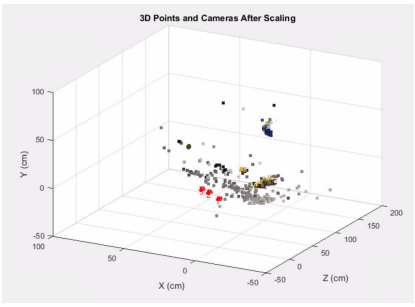


Fig. 13: Batch triangulation result using eight cameras

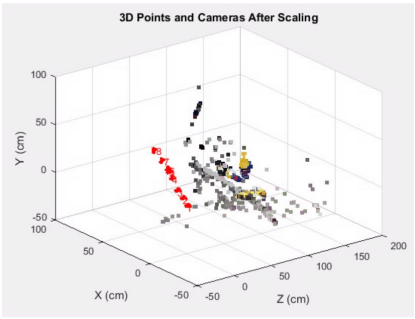


Fig. 14: Batch triangulation result using eight cameras

The L-2 normed error between ground truth and estimated pose is summarized in Table **??**. Fig. 13 shows the final colored 3D model using three frames while Fig.14 shows the result when using eight frames. It is apparent that errors in our camera pose calculations accumulate as the camera sequence get further away from the original camera, which results in point clouds that show several repetitions of the same object in different locations.

	Error
Camera 1	0%
Camera 2	19.85%
Camera 3	16.65%

TABLE I: L2 norm error between camera poses and ground truth

V. Conclusion

The result of our two frame reconstruction using linear triangulation is pretty good when compared to ground truth, while multi-frame reconstruction with linear triangulation produces duplicate point of cloud. Although we are able minimize triangulation errors using batch triangulation, many points corresponding to same real world object appear far apart in the 3D point cloud. This is due to the fact that we are not eliminating error in our camera pose estimations and is apparent in sequences involving more than three cameras. The point cloud shows the same object repeated in several locations. One way to reduce error for camera pose estimation is bundle adjustment. Instead of only calculating a 3D point that minimize the error through every scene, bundle adjustment also estimates new camera poses to better fit projection of 3D points and their corresponding image location. The future work of our project will be to initialize a bundle adjustment toolbox, such as Sparse [5], using our camera pose estimation.

References

[1] Lucas, Bruce D., and Takeo Kanade. "Shape and Motion from Image Streams under Orthography: a Factorization Method" International Journal of Computer Vision, 9:2, 137-154, 1992.

[2] Pollefeys, Marc, et al. "Visual modeling with a hand-held camera." International Journal of Computer Vision 59.3 (2004): 207-232.

[3] Richard Hartley , and Andrew Zisserman. "Multiple View Geometry in Computer Vision Second Edition" Cambridge University Press, March 2004

[4] Jianbo Shi. "Project 2: Structure from Motion", University of Pennsylvania, 2015

[5] Kurt Konolige. "Sparse Sparse Bundle Adjustment" Willow Garage, 2010