Project 2: Buildings Built in minutes: SfM and NeRF

Krunal M. Bhatt
Masters of Science in Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: kmbhatt@wpi.edu

USING 1 LATE DAY

Jesulona Akinyele
Masters of Science in Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: jfakinyele@wpi.edu

USING 1 LATE DAY

***Abstract*—The report presents our understanding and experimentation with reconstructing a 3D scene and simultaneously obtaining camera poses of a monocular camera w.r.t given scene. We reconstruct the scene from a set of given images with different POV's (which is similar to a camera in motion). Phase 1 shows the basic traditional approach to solving the problem. We use the images provided to us for the project.**

I. Phase 1: Traditional Approach

Phase 1 shows a traditional approach as to how a rigid scene is recreated in 3D from a set of images having different points of view. We also obtain the camera pose simultaneously while recreating the rigid scene in 3D. This approach is called Structure from Motion(SfM). The traditional approach has the following outline: Feature Matching and Outlier Rejection using RANSAC, Estimation of the Fundamental Matrix, Estimation of Essential Matrix from the Fundamental matrix, Estimating Camera Pose from the Essential Matrix, Checking for Cheirality Condition using Triangulation, Perspective-n-point, Bundle Adjustment. We explain our approach to solving them in the following subsections. The subsections contain the output for the steps discussed above.

The dataset has been provided to us for the same. Images are in *.png* format. They are taken from a Samsung S22 Ultra's primary camera at f/1.8 aperture, ISO 50, and 1/500 sec shutter speed. The camera is calibrated after resizing using a Radial-Tangential model with 2 radial parameters and 1 tangential parameter using MATLAB R2022a's Camera Calibrator Application. Images provided are distortion-free and resized to 800 x 600px. Additionally, as SfM relies heavily on good features and their matching, we have been given the keypoint matching data in the $matching^*.txt$ file. A separate relative keypoint matching file is there for every image in the dataset. Fig. 1 shows the dataset images used for the programming exercise.

Fig. 1. Dataset for the project

A. Feature matching, Fundamental Matrix, and RANSAC:

Keypoint matching using Scale-Invariant Feature Transform or SIFT key points is a fundamental task. SIFT detects key points that are invariant to scale, rotation, and illumination, after which, it computes descriptors which are neighborhoods around the key points. Descriptors are matched and corresponding points are found in the second image and the closest match is found based on a distance metric. Now, before these matches are rejected, let's look at what the Fundamental Matrix is.

1) Fundamental Matrix : Denoted by **F**, the fundamental matrix is a 3x3 (rank 2) matrix that relates the set of points in two images from different views. The fundamental matrix depends on the epipolar geometry between two views. Geometry is an intrinsic projective geometry between the two views. This geometry depends on the camera's internal parameters (K matrix) and the relative pose i.e.: it is not dependent on the structure of the scene. We will see more of epipolar geometry in the next sub-section.

The **F** matrix is a geometric and arithmetic representation of the epipolar geometry. We use an epipolar constraint or correspondence condition which is

$$\mathbf{x}_i' \mathbf{F} \mathbf{x}_i = 0$$

Since F is a 3x3 matrix we can write a homogenous linear system with 9 unknowns as follows:

$$\mathbf{x}_i' = \begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}, \quad \text{and} \quad \mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}.$$

Expanding, we get:

$$x_i' f_{11} x_i + x_i' y_i f_{21} + x_i f_{31} + y_i x_i f_{12} + y_i y_i f_{22} + y_i f_{32} + x_i f_{13} + y_i f_{23} + f_{33} = 0.$$

Simplifying for m correspondences:

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

In a similar setting while performing homography between two images, was calculated. 4 points were used in that. Unlike homography, while estimating the F matrix, each point only contributes to one constraint the epipolar constraint. Therefore, we use 8 point algorithm to solve the homogeneous system above. The Fig. 2 shows the estimated F matrix.

$$F = \begin{pmatrix} 3.00189977 \times 10^{-6} & -2.45583926 \times 10^{-5} & 6.88351304 \times 10^{-3} \\ 1.30755935 \times 10^{-5} & 4.35858746 \times 10^{-6} & -9.62965121 \times 10^{-3} \\ -6.76368496 \times 10^{-3} & 1.45549178 \times 10^{-2} & -9.99801127 \times 10^{-1} \end{pmatrix}$$

Fig. 2. The Fundamental Matrix F

2) Epipolar Geometery: For instance, we observe **X** in the 3D-space, as captured in x in the first image and x' in the second, we form an epipolar plane π that has the points **x, x',** and **X** as coplanar points. The Fig. 3 shows the epipolar geometry visually.

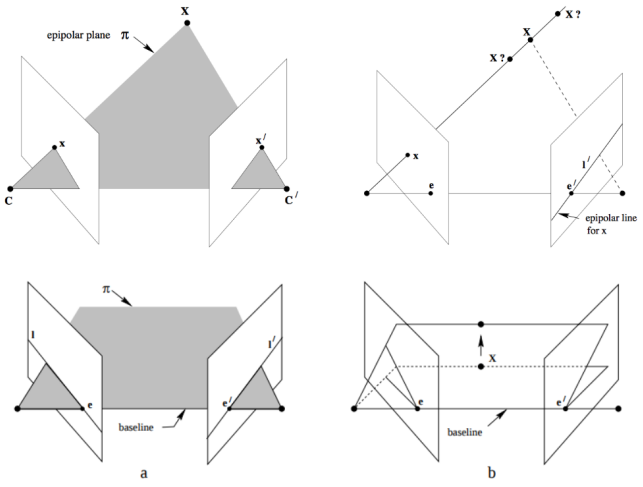

Fig. 3. Epipolar geometry

We have a **Epipole** as the point of intersection of the line joining the camera centers with the image plane (**e** and **e'**). The **Epipolar Plane** π is the plane containing the baseline. The **Epipolar line** is the intersection of the epipolar plane with the image plane. All the epipolar lines intersect at the epipole.

3) Match Outlier Rejection via RANSAC: In this step in the process, we remove the outliers that are generated after SIFT. The data is bound to be noisy after we perform SIFT. F matrix with maximum numbers of inliers is chosen. Fig. 4 shows the same.

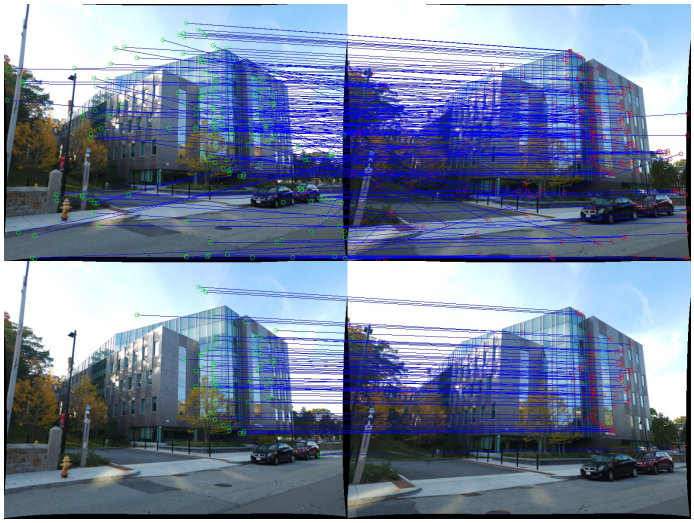

Fig. 4. RANSAC Output

B. Estimate Essential Matrix from Fundamental Matrix:

Now that we have **F** from the epipolar constraints, we can find the relative camera poses between the two images. This can be computed using Essential Matrix **E**. Essential Matrix is another 3x3 matrix but has additional properties that relate the points assuming that the cameras follow the pinhole model. We can represent this as

$$\mathbf{E} = \mathbf{K}^T\mathbf{F}\mathbf{K}$$

where K is the camera calibration matrix or camera intrinsic matrix. It is evident from the above equation that we can extract the E matrix with the help of the above equation. E can be described as:

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

Here, F is defined in the original image space (i.e. pixel coordinates) whereas E is in the image coordinates. The Fig. 5 shows the essential matrix which is calculated.

$$E = \begin{pmatrix} 0.24902383 & -0.43240625 & -0.00646718 \\ -0.43245373 & 0.75092275 & 0.00842021 \\ 0.00403665 & -0.00981988 & 0.99994265 \end{pmatrix}$$

Fig. 5. The Essential Matrix E

C. Estimating Camera Pose from E matrix:

The camera pose has six degrees of freedom w.r.t to world. Since the E matrix is identified, the four camera poses are: $(C1, R_1), (C_2, R_2), (C_3, R_3), and (C_4, R_4)$ where $\mathbf{C} \in R^3$ is

the camera center and R $\in$ SO(3) is the rotation matrix, can be computed. Thus, the camera pose can be written as

$$P = KR[I_{3x3} - C]$$

Let,

$$E = UDV^T and\ W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The four configurations can be written as:

$$C_1 = U(:,3)\ and\ R_1 = UWV^T$$
$$C_2 = -U(:,3)\ and\ R_2 = UWV^T$$
$$C_3 = U(:,3)\ and\ R_3 = UW^TV^T$$
$$C_4 = -U(:,3)\ and\ R_4 = UW^TV^T$$

We take the $det(R)$ ad if it should be equal to one. If $det(R) = -1$, we correct the camera pose by changing the sign of C and R to Negative. The Fig. 6 shows the camera pose. We

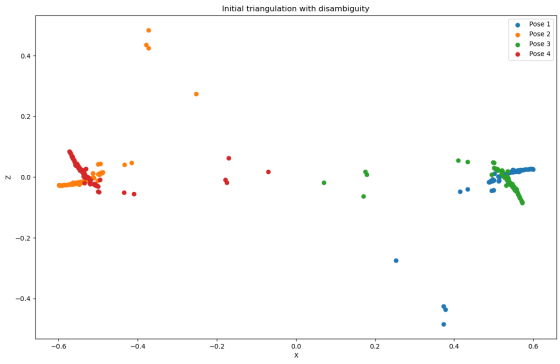

Fig. 6. Pose Estimation

D. Triangulation Check for Cheirality Condition:

Till now, we calculated four poses for different cameras for a pair of images using the essential matrix. Now, we triangulate the 3D points, given two camera poses.

To get the correct camera pose, we need to remove the disambiguity. This can be done by checking the cheirality condition, triangulating the 3D points using linear least squares to check the sign of the depth in Z in the camera coordinate system using linear least squares to check the sign of the depth Z in the camera coordinate system w.r.t to camera center. Fig. 7 shows the triangulation plot with disambiguity removed and non linear triangulation implimented.

E. Perspective-n-Points:

We now have a set of n 3D points in the world, their 2D projections in the image and intrinsic params, the 6DOF camera pose can be estimated using linear least squares. We call this Perspective-n-point(PnP). For a solution to exist in this particular problem, $n \geq 3$. In many methods, it is assumed

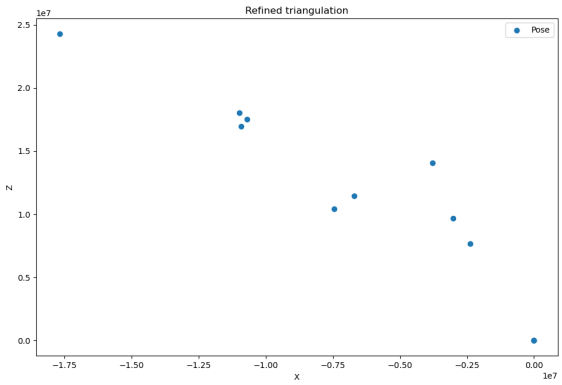

Fig. 7. Triangulation Output

that the camera is calibrated. We learn here a vanilla version of PnP, where we register a new image given image-world correspondences, X to x which is then followed by nonlinear optimization.

1) Linear Camera Pose Estimation : Now that we have $X \leftrightarrow x$ and K, we estimate the camera pose using linearPnP function. We isolate the camera parameters by normalizing 2D points by the intrinsic parameter. Linear least squares system that related the 3D and 2D points can be solved for (C,R) where,

$$t = -R^TC$$

Rotation matrix $R \in SO^3$ would not be correct since linear least squares do not force orthogonality. To resolve this wr correct the matrix $R = UV^T$ where $R = UDV^T$.

In order to estimate the camera's posture accurately, the linear Perspective-n-Point (PnP) technique requires a minimum of 6 correspondences. This is because the algorithm requires constraints. A minimum of six constraints are required to solve for the six unknowns that comprise the position (translation) and orientation (rotation) of the camera. These constraints are contributed by each correspondence between a 3D point and its corresponding 2D projection. These constraints are two for position (x, y) and one for depth (z). This cutoff point guarantees that there is enough data to build a system of linear equations that includes intrinsic (calibration) and extrinsic (rotation and translation) factors, allowing for precise camera pose estimation.

2) PnPRANSAC : PnP RANSAC, or Perspective-n-Point with Random Sample Consensus, is a reliable variation of the Perspective-n-Point algorithm that is frequently used in robotics and computer vision applications to estimate the pose (position and orientation) of a camera in a scene using 3D–2D point correspondences. PnP RANSAC iteratively selects subsets of correspondences, frequently surpassing this minimal threshold, to estimate the camera posture, in contrast to the linear PnP technique which requires at least 6 correspondences. This procedure aids in reducing the impact of anomalies or

inaccurate correlations found in the data. Fig. 8 shows the algorithm implemented in the code.

$n = 0$
for i = *1:M* **do**
  // Choose 6 correspondences, $\hat{X}$ and $\hat{x}$, randomly
  [C R] = LinearPnP($\hat{X}$, $\hat{x}$, K);
  $\mathcal{S} = \emptyset$;
  for j = *1:N* **do**
    // Measure Reprojection error
    $e = \left(u - \frac{P_1^T \tilde{X}}{P_3^T \tilde{X}}\right)^2 + \left(v - \frac{P_2^T \tilde{X}}{P_3^T \tilde{X}}\right)^2$;
    if $e < \epsilon_r$ **then**
      $\mathcal{S} = \mathcal{S} \cup \{j\}$
    end
  end
  if $n < |\mathcal{S}|$ **then**
    $n =| \mathcal{S} |$;
    $\mathcal{S}_{in} = \mathcal{S}$
  end
end

Fig. 8. PnP RANSAC

PnP RANSAC offers a more reliable and accurate estimation of the camera pose in the presence of noise or outliers in the correspondence data by iteratively fitting models to random subsets of the data and choosing the best model based on consensus, usually measured by the number of inliers supporting the model.

3) Nonlinear PnP : A variation of the PnP algorithm called nonlinear PnP (Perspective-n-Point) increases accuracy by fine-tuning the initial camera posture estimate derived from techniques such as linear PnP or PnP RANSAC. Nonlinear PnP iteratively refines the camera pose using optimization techniques as Levenberg-Marquardt optimization, in contrast to linear approaches, which solve for the camera pose using linear techniques and may have accuracy constraints. By using the current estimate of the camera position, this optimization procedure minimizes the reprojection error, a measure of the difference between the observed 2D image points and the 3D points projected onto the picture plane.

The geometrically meaningful reprojection error between measurement and projected 3D points is given as follows:

$$\min_{C,R} \sum_{i=1,J} \left(u^j - \frac{p_1^T X_i}{p_3^T X_i}\right)^2 + \left(v^j - \frac{p_2^T X_i}{p_3^T X_i}\right)^2$$

Here X_i, is the homogeneous representation of X. A compact representation of the rotation matrix using quaternion is a better choice to enforce orthogonality of rotation matrix as :

$$\min_{C,q} \sum_{i=1}^{J} \left(w_j - \frac{p_i^T X_j}{p_i^T X_j}\right)^2 + \left(v_j - \frac{p_i^T X_j}{p_i^T X_j}\right)^2$$

F. Bundle Adjustment:

After all the camera poses and 3D points are estimated, we refine 3D points that are initialized by previous reconstructions which were due to minimization of reprojection error.

We build a visibility matrix which finds the relationship between camera and a point. This is a binary matrix V which is IxJ

By minimizing the reprojection error—the discrepancy between the observed 2D image points and the 3D points projected onto the picture plane using the currently estimated camera poses and 3D structure—bundle adjustment optimizes the overall 3D reconstruction. Bundle Adjustment enhances the overall consistency and precision of the reconstruction by optimizing both the 3D structure (the locations of the rebuilt points) and the camera poses (the positions and orientations of the cameras). Non-linear optimization techniques like Gauss-Newton and Levenberg-Marquardt are commonly used to implement it. These algorithms modify the parameters iteratively to minimize the reprojection error. Bundle adjustment is crucial for fine-tuning large-scale reconstructions, fixing camera position drift or inaccuracies, and enhancing the overall quality of the 3D scene reconstruction.

We also have the Table I for reprojection error:

Method	Error
Linear triangulation	25.32
Non-linear triangulation	24.6
Linear PnP	N/A
Non-linear PnP	N/A

TABLE I
REPROJECTION ERRORS

REFERENCES

[1] https://rbe549.github.io/spring2024/proj/p2/
[2] http://cvrs.whu.edu.cn/downloads/ebooks/Multiple%20View %20Geometry%20in%20Computer%20Vision%20(Second%20Edition).pdf
[3] https://cmsc426.github.io/math-tutorial/#svd
[4] https://en.wikipedia.org/wiki/Eight-point_algorithm
[5] https://users.cecs.anu.edu.au/ hongdong/new5pt_cameraREady_ver_1.pdf
[6] https://www.microsoft.com/en-s/research /wp-content/uploads/2016/02/tr98-71.pdf