Computer Vision
ETH Zurich, Fall 2023
Hardik Shah

Exercise 7: Structure from Motion and RANSAC

Hardik Shah

1 Structure from Motion

Structure from Motion (SfM) is a computer vision technique that reconstructs the 3D structure of a scene from a sequence of 2D images. The basic idea is to track features across multiple images and triangulate their positions in 3D space. The algorithm estimates camera poses and 3D point positions iteratively, gradually expanding the reconstructed scene.

1.1 Implementation and Algorithm Outline

The Structure from Motion (SfM) algorithm is implemented by two modules:

1. `geometry.py`: handles essential matrix estimation, pose decomposition, point triangulation, and pose estimation.
2. `corrs.py`: manages the correspondence finding and updating of the reconstruction state.

Below is a high level outline of the overall algorithm:

1. **Image Loading and Feature Matching:**
   The algorithm starts by loading a sequence of images from a specified data folder. Feature matches between image pairs are obtained using precomputed feature matches stored in the data folder.
2. **Initialization:**
   Two initial images are selected to bootstrap the reconstruction process. The essential matrix is estimated using the selected image pair.
3. **Relative Pose Estimation:**
   The essential matrix is decomposed to obtain four possible relative poses (rotation and translation). For each pose, the algorithm attempts to triangulate 3D points using the `TriangulatePoints` function.
4. **Selecting the Correct Pose:**
   The correct pose is determined by selecting the solution that produces the most 3D points visible in front of both cameras. The selected pose is applied to the images.
5. **Triangulation and Point Correspondences Update:**
   The initial set of 3D points is triangulated using the selected pose. The newly triangulated points are added to the images as 2D-3D correspondences.

6. **Map Extension:**
   The algorithm iteratively extends the reconstructed map by registering new images and triangulating additional points. Correspondences between 2D keypoints and 3D points are established. The new image pose is estimated, and the image is added to the set of registered images. Triangulation is performed with all previously registered images. The 3D points and image correspondences are updated.

1.2 Results

The implemented pipeline successfully reconstructs the 3D structure of the fountain from a sequence of 2D images. Results are shown in Figure 1 and Figure 2.

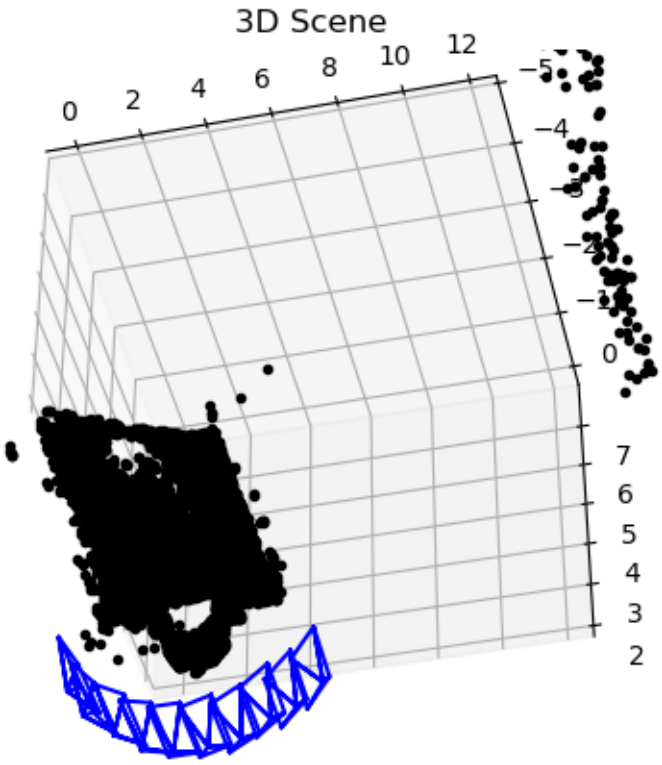

Figure 1: Visualizing the triangulated 3D points along with estimated camera poses

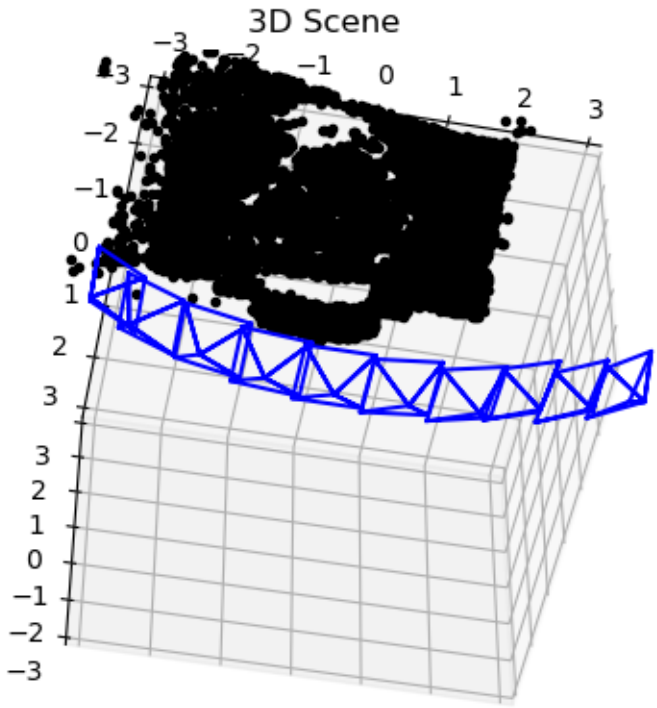

Figure 2: Zooming into the fountain, ignoring points on the other wall.

2 RANSAC

Random Sample Consensus (RANSAC) is an iterative algorithm widely used for robust model fitting in the presence of outliers. The implemented RANSAC algorithm is designed to fit a linear model to a set of 2D data points corrupted by noise and outliers. The algorithm iteratively selects random subsets of data, fits a model to each subset, and evaluates the goodness of fit by counting inliers. The final model is selected based on the subset that maximizes the number of inliers.

2.1 Implementation and Algorithm Outline

A high level outline of the algorithm is given below:

1. **Least Squares Solution:**
   The `least_square` function computes the least-squares solution for a given set of input points (x, y). It utilizes `np.linalg.lstsq` to find the coefficients of a linear model (k, b), where $y = kx + b$.

2. **Number of Inliers:**
   The `num_inlier` function calculates the number of inliers for a given model (k, b) and a threshold distance (`thres_dist`). It returns the count of inliers and a boolean mask indicating the indices of inliers.

3. **RANSAC Algorithm:**
   The `ransac` function is the core of the RANSAC algorithm. It iteratively performs the following steps:

   (a) Randomly selects a subset of `num_subset` samples from the data.

   (b) Computes the least-square solution for the selected subset.

   (c) Calculates the number of inliers and inlier mask based on the entire dataset and the computed model.

   (d) Updates the best solution if the current model has more inliers than the previous best.

2.2 Results

The results demonstrate the effectiveness of the RANSAC algorithm in handling datasets with outliers. In Figure 3, the scatter plot showcases the inliers and outliers, and the fitted linear models are overlaid for both the RANSAC and least-squares approaches.

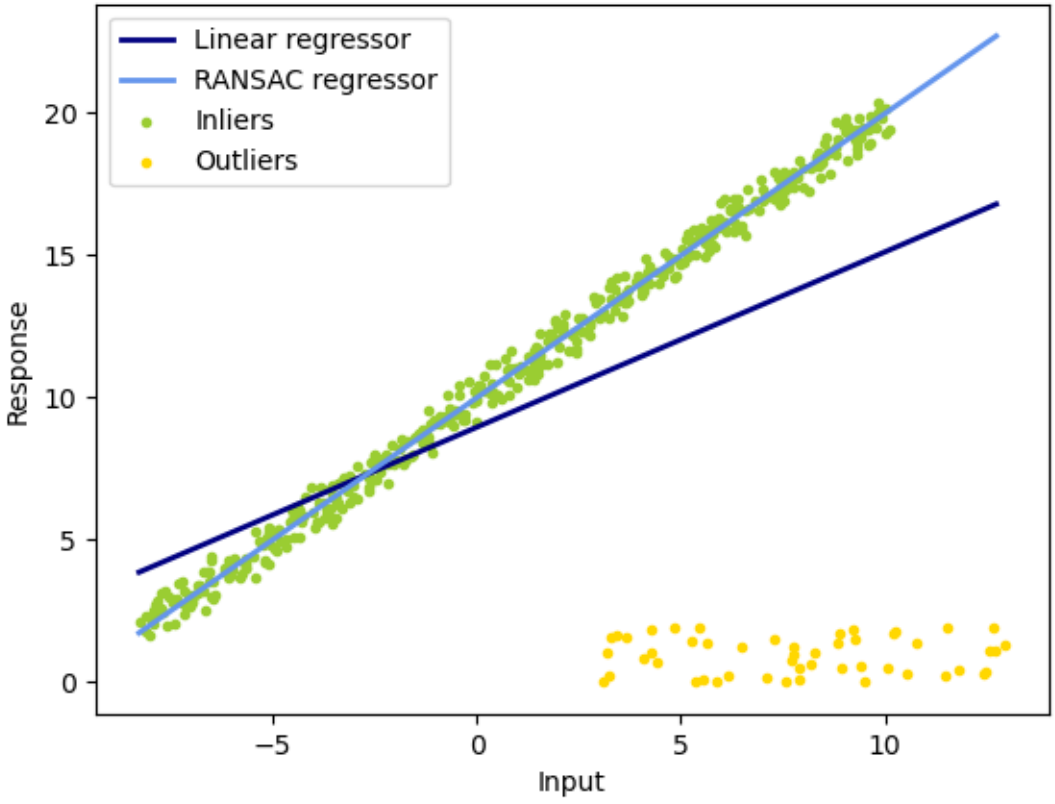

Figure 3: Comparison of RANSAC estimate and Least Square estimate.

The RANSAC model fitting is visibly more robust, accurately capturing the underlying trend in the presence of outliers. In contrast, the least-squares fit is more influenced by the outliers, resulting in a model that deviates significantly from the ground truth.

This stark difference highlights the capability of RANSAC to identify and discard outliers during the model estimation process.
Furthermore, the estimated coefficients (k, b) are printed for both methods in Figure 4, emphasizing the superior performance of RANSAC in the presence of noise and outliers. The RANSAC algorithm's ability to iteratively select robust subsets and disregard outliers contributes to its reliability in providing accurate model estimates.

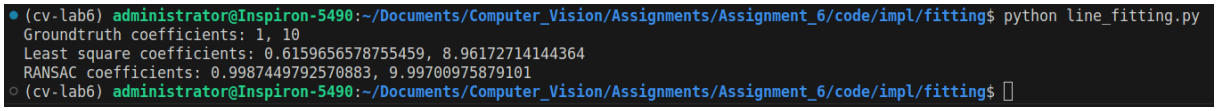

Figure 4: Estimated coefficients (k, b) are printed for both methods.

In practical applications where datasets may contain noise or outliers, RANSAC emerges as a valuable tool for robust model fitting, offering improved resilience against the adverse effects of erroneous data points.