lab06-sfm-ransac

Gnkgo, Computer Science, Bsc 7. Semester

Abstract

Here is the report for the lab04-segnet-meanshift.

For this exercise, Copilot was used to help with the syntax. The written report was checked with Grammarly.

1 SFM

In my opinion, this was a challenging task as the code did not compile for a long time and the result is not very clear to me. Nevertheless, the code follows the logic of the well-known SFM algorithm.

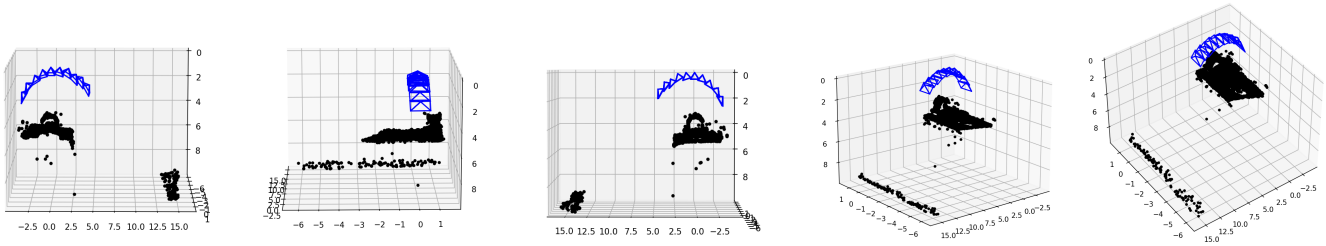

Figure 1: Different perspective of SFM.py algorithm

1.1 EstimateEssentialMatrix

Estimate the essential matrix E encapsulating the epipolar geometry between two calibrated cameras.

- Normalise keypoints using camera matrix K
- Construct constraint matrix
- Solve for E using SVD
- Enforce the rank 2 constraint
- Validate E using epipolar constraint

1.2 TriangulatePoints

Estimate 3D points from 2D correspondences in two views

- Use camera poses (R, t) to build projection matrices P_1, P_2.
- Solve for 3D point positions using SVD of the linear triangulation equation
- Filter points behind the cameras by checking the Z coordinates in each camera space.

1.3 EstimateImagePose

Estimate the pose (R, t) of a camera given 2D-3D correspondences.

- Normalise 2D points with K.
- Construct the projection constraint matrix
- Solve the projection matrix P using SVD
- Extract R and t from P, ensuring that R is a proper rotation matrix

1.4 Triangulate Image

Triangulates new 3D points between a new image and images already registered.

- Loop over all registered images
- Perform triangulation for matches between new image and registered images
- Aggregate all new 3D points and matches

2 Fitting a Line

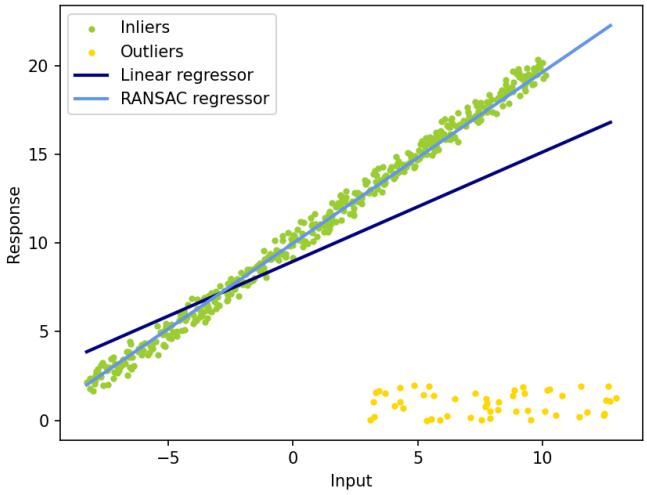

Figure 2: Ransac

This code implements RANSAC to robustly fit a line to data points that contain both inliers and outliers. The parameters given in already given in the exercise were used:

```
iter = 300
thres_dist = 1
n_samples = 500
n_outliers = 50
k_gt = 1
b_gt = 10
num_subset = 5
```

The results after running the code is:

```
Estimated coefficients (GT k, GT b, linear reg. k, linear reg. b, RANSAC k, RANSAC b):
1, 10, 0.6159656578755459 8.96172714144364 0.9643894946568986 9.98292129496683
```

2.1 Least Square

Computes the least square solution for fitting a line $y = ax+b$ to the data points (x, y) with `np.linalg.lstsq`.

2.2 Num Inliers

Counts the number of inliers for a given line $y = ax + b$ and creates a mask to identify those inliers.

- Calculates the perpendicular distance to the line for each point (x_i, y_i):

$$distance = \frac{|y_i = (ax_i + b)|}{\sqrt{k^2 + 1}}$$

- Points with a distance less than the threshold are considered as inliers.

2.3 Ransac

Implements the RANSAC algorithm to fit a line to noisy data with outliers.

- Randomly selects a num_subset of data points to compute a candidate line
- uses least_squares to fit the line
- Uses num_inlier to compute the number of inliers for the line.
- If more inliers than previous, update best parameters