CMSC733
Project 3: Structure From Motion—Traditional Approach

Nitin Suresh
School of Electrical and
Computer Engineering
University of Maryland - College Park
UID: 113638855

Jo Shoemaker
Computational Linguistics and
Information Processing Lab
University of Maryland - College Park
UID: 115506787

Given images of a scene taken from different positions, it is possible to estimate the real-world, three-dimensional positions of elements in the scene. This has been called the Structure from Motion problem. Our implementation presumes a set of given images I taken by a camera with pre-calibrated intrinsic matrix K, and preliminary corresponding points M_{ij} mapped between all images i and j. Using these values, we are able to estimate camera poses P_i for all images in I and positions in three-dimensional space for a subset of the points in M. We use several techniques to refine these initial estimates.

I. Initial Camera Pose Estimation

We initially use RANSAC to estimate the inlier points between the first two images. These correspondence points are used to estimate the fundamental matrix by stacking and solving $x'Fx = 0$. RANSAC is used along with the fundamental matrix estimation to get the best inlier points. Examples are shown below-

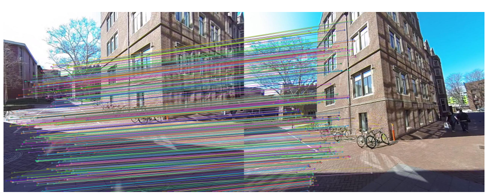

Fig. 1. Correspondence inliers detected through RANSAC between image1 and image2

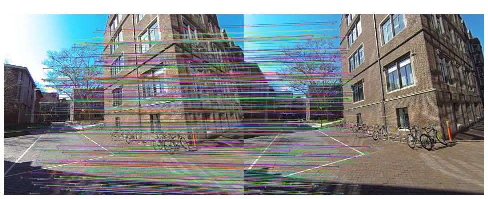

Fig. 2. Correspondence inliers detected through RANSAC between image3 and image4

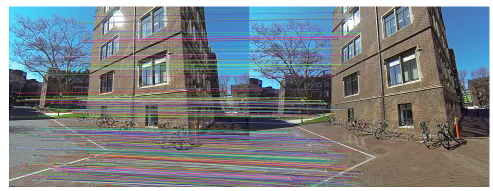

Fig. 3. Correspondence inliers detected through RANSAC between image5 and image6

The equation was solved by taking the SVD and using the last column of V. The essential matrix can be estimated from F since we are given the calibration matrix for the camera ($E = K^T FK$). The relative camera pose P (determined from the rotation matrix R and the camera center translation matrix C) can be determined from the essential matrix, keeping the reference camera at $[I0]$. We get 4 candidate poses, and their corresponding points.

II. Initial 3D Position Estimation

To obtain the correct pose and 3D positions of the points, we perform the cheirality check. For this we check whether $r_3(X - C) > 0$ and additionally whether $[001]^T X$. An interesting point to note here was that if we added an additional constraint here that checked the residuals as well, this improved accuracy. The best pose out of the 4 candidates is the one that satisfies these conditions for the maximum number of points. Nonlinear optimization of the points is carried out by optimizing on the reprojection error. Figure below shows the linearly and non-linearly triangulated points.

III. Additional Camera Pose Estimations

Once we have a fairly reliable set of 3D and image point correspondences to work with, we estimate each additional image's camera poses.

A. *Perspective-N-Point solution and* RANSAC

Once an initial set of 3D points X_i is corresponded with image points x_i for image i, we algebraically solve for P_i in

Step	Mean Reprojection Error			
	Img3	Img4	Img5	Img6
PnPRANSAC	2e37	1e37	3e37	7e37
Nonlinear PnP	303013686	41602560	139605	1312681
Linear Triangulation	70495458	1766038	3937977	1389664
Nonlinear Triangulation	314887	1200472	41541	14596

TABLE I
MEAN REPROJECTION ERROR AFTER EACH REFINEMENT STEP

`LinearPnP.py`. Using six pairs of points S_i and s_i sampled from X_i and x_i, we solve the system

$$S_i^T \left[R_i, t_i\right]^T = (Ks_i)^T$$

for $[R, t]^T$. We negate R if its determinant is -1. Then we find C_i by multiplying t_i by the transpose of R_i. With these components it is trivial to find P_i.

Because our data is noisy, we refine our initial estimate of P_i using the RANSAC algorithm to minimize reprojection error. For each image's P estimate, we run `PnPRANSAC.py` for 2,000 iterations with a tolerable reprojection error of $500,000$. This does not remove outliers from our X_i and x_i point correspondences.

B. Refining Camera Pose with `NonlinearPnP.py`

Using `scipy.optimize.leastsq`, we optimize our P_i to minimize reprojection error for all points in X_i. Table I displays mean reprojection error for each of images 3-6 after optimizing P. These values are extremely high, but they represent a significant improvement over the error values after `PnPRANSAC.py`.

Once this optimization has gotten our P_i as certain as it can be, the X_i values are further refined by linear and nonlinear triangulation as described for the first image pair. Figure 4 shows the camera position and 3D point estimates after these steps.

IV. BUNDLE ADJUSTMENT

We use `python-sba` [1] to perform Sparse Bundle Adjustment to optimize all 3D point estimates X and all camera poses P at once.

REFERENCES

[1] THERIAULT, D., FULLER, N., JACKSON, B., BLUHM, E., EVANGELISTA, D., WU, Z., BETKE, M., AND HEDRICK, T. A protocol and calibration method for accurate multi-camera field videography. *J exp Biol 217* (2014), 1843–1848.

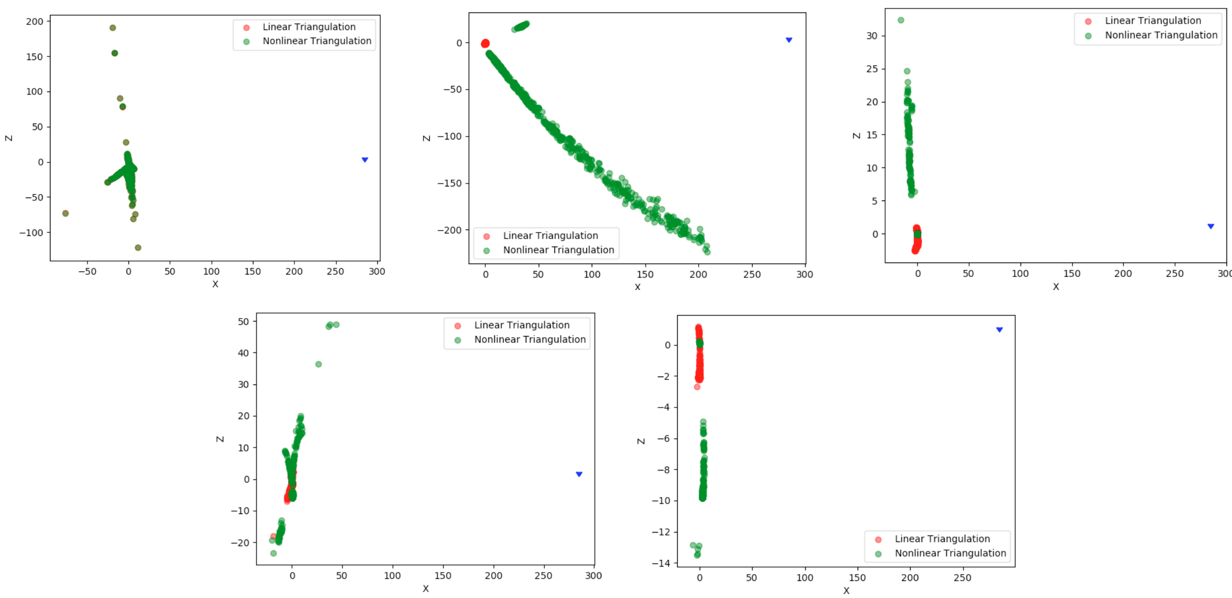

Fig. 4. Linear and Nonlinear Triangulation 3D point and camera pose estimates for Images 1,3,4,5, and 6