ENGN2560 Computer Vision

Lab03: 3D Reconstruction and Bundle Adjustment

The goal of this lab is to:

- Learn how a 3D scene can be reconstructed by various triangulation methods.
- Learn how 3D scene points and camera poses can be optimized through bundle adjustment.

Problem 1. Reconstruction by Triangulation In the lecture, we have learned three different methods of reconstructing points from the estimated camera poses, namely, Symmedian Point Triangulation, Linear Triangulation, and Kanatani's method. This problem continues our previous lab by reconstructing 3D points from inlier point correspondences observed by the images, Figure 1(a). Put your code in the provided `P1_main.m` file. Necessary steps to follow are:

1. **Relative Pose Estimation:** Estimate relative pose $(\mathcal{R}, \mathcal{T})$ of the two images given in Figure 1(a) using essential matrix estimation under a RANSAC scheme. You have done this in Problem 4 of Lab02.
2. **Triangulation:** Use the estimated relative pose and the inlier feature correspondences supporting the pose, implement three triangulation methods. Note that the relative pose $(\mathcal{R}, \mathcal{T})$ estimated under the RANSAC loop is the pose of the second image with respect to the first image, *i.e.* the pose of the first camera $(\mathcal{R}_1, \mathcal{T}_1)$ is $\mathcal{R}_1 = I$, $\mathcal{T}_1 = [0 \quad 0 \quad 0]^T$ while the pose of the second camera is $(\mathcal{R}_2, \mathcal{T}_2)$ is $\mathcal{R}_2 = \mathcal{R}$, $\mathcal{T}_2 = \mathcal{T}$. Thus, the 3D points Γ^w computed by the triangulation methods are located under the first camera coordinate.
   (a) **Symmedian Point Triangulation** minimizes the *geometric* error in 3D. Refer to the Reading Materials posted on the Canvas site for the derivation of the algorithm. (10 points)

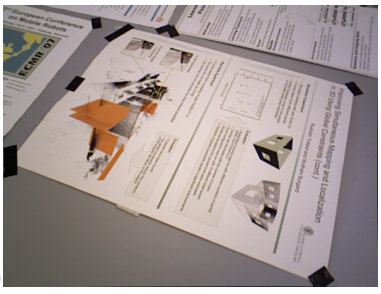
(a)   (b) 

Figure 1: (a) A pair of images used for triangulating 2D feature correspondences to 3D scene points. (b) An additional image is introduced to extend the triangulation task from 2 views to 3 views.

(b) **Linear Triangulation** minimizes the *algorithmic* error. Also refer to the Reading Materials for more information on the algorithm. (10 points)

(c) **Kanatani's Method** works in the *observation space* rather than the *representation space*, and finds an intersected 3D point by minimizing reprojection errors: let γ and $\bar{\gamma}$ be a pair of inlier correspondence we obtained from the two images, and $\hat{\gamma}$ and $\hat{\bar{\gamma}}$ be the ground truth correspondence which are unknown. The goal is that, given the relative pose R and T, find the perturbations of γ and $\bar{\gamma}$ such that the projection rays intersect, while minimizing the extent of the perturbations at the same time, *i.e.*,

$$\min_{\hat{\gamma},\hat{\bar{\gamma}}} \left[|\hat{\gamma} - \gamma|^2 + |\hat{\bar{\gamma}} - \bar{\gamma}|^2 \right], \quad \text{subject to} \quad \hat{\bar{\gamma}}^T E \hat{\gamma} = 0. \tag{1}$$

This is a constrained optimization problem which can be lifted to an non-constrained optimization problem by using a Lagrange multiplier λ:

$$\min_{\hat{\gamma},\hat{\bar{\gamma}},\lambda} \left[|\hat{\gamma} - \gamma|^2 + |\hat{\bar{\gamma}} - \bar{\gamma}|^2 + \lambda \hat{\bar{\gamma}}^T E \hat{\gamma} \right]. \tag{2}$$

Equation 2 is a non-linear function with five unknowns: two from $\hat{\gamma}$, two from $\hat{\bar{\gamma}}$, and one from the Lagrange multiplier λ. The optimal $\hat{\gamma}$ and $\hat{\bar{\gamma}}$ can be solved by numerically minimizing Equation 2. Follow the steps below: (20 points)

i. **Compute an Initial Guess:** For each pair of inlier feature correspondences, do either Symmedian Point Triangulation or Linear Triangulation to get the 3D reconstructed point. Project that 3D point to the two images. The projected points are served as an initial guess for the non-linear optimization. For the Lagrange multiplier, you can pick any arbitrary value as its initial guess, *e.g.* 10.

ii. **Minimize Equation 2:** For *each* inlier correspondence, construct an energy function

```
function Energy = evaluateEnergyFunction(x, gamma, gamma_bar, ...
E)
```

which takes inputs `x` as an array containing five unknowns, or variables ($\hat{\gamma}$, $\hat{\bar{\gamma}}$, and λ), a pair of observed correspondence (γ and $\bar{\gamma}$), and an essential matrix E, returns the energy value, *i.e.*, the evaluation of Equation 2. Then, do non-linear optimization using Levenberg-Marquardt algorithm. You are allowed to use MATLAB's built-in function. A sample piece of code is given to you in order to use the MATLAB's non-linear optimizer:

```
Energy = @(x)(evaluateEnergyFunction(x, gamma, gamma_bar, E));
options = optimoptions('lsqnonlin', 'Display', 'iter');
options.Algorithm = 'levenberg-marquardt';
optimal_points = lsqnonlin(Energy, x0, [], [], options);
```

where `x0` is the initial guess which is also an array containing five variables.

`lsqnonlin` is a matlab built-in least square non-linear optimizer which gives you the optimized points.

iii. **Triangulate Optimal 2D Points:** The optimized point correspondences can then be used to triangulate to find optimal reconstructed 3D points via either Symmeidan Point Triangulation or Linear Triangulation. The optimal 3D points are in the end very close to the intersection of the projection rays.

3. **Evaluation and Comparisons:** All the 3D points reconstructed by the features on the image pair in Figure 1(a) should lie on a plane under the first camera coordinate. The ground truth plane is described by the following plane equation.

$$-0.0003x + 0.5792y + 0.8152z - 1.0545 = 0, \tag{3}$$

where (-0.0003, 0.5792, 0.8152) is the normal vector of the plane, and -1.0545 is the displacement from the origin. They are respectively defined as two members of the parameter structure `PARAMS.GT_PLANE_NORMAL_VECTOR` and `PARAMS.GT_PLANE_DISPLACEMENT`. Using Equation 3, we can evaluate and compare the performances of the above three triangulation methods:

   (a) Scale down your reconstructed points by a factor of 0.29. This is given as a member of the parameter structure `PARAMS.SCALE` in your code.

   (b) Calculate the distances of reconstructed points to the plane. Show the distributions of the distances for three methods. (10 points)

   (c) Try to decrease the RANSAC iterations so that the pose is not very accurate, and triangulate inlier features again. Which triangulation method is more robust to noisy poses? (5 points)

4. **Extending Two Views to Three Views:** Typically, more views produce more robust feature correspondences and reconstruction results. Add a third image, Figure 1(b), and redo steps 1-3. Estimate the relative pose under a RANSAC scheme for image pair of 1 and 2 as well as 1 and 3, so that the relative poses are all with respect to the first camera. Triangulate only the feature correspondences that survive across three views, *i.e.*, the 3D point must be covisible by three views. To evaluate the 3-view triangulation, the scale of the reconstructed 3D points is different compared to the 2-view triangulation:

   (a) Symmedian Point Triangulation: Scale down your reconstructed points by a factor of 0.625.

   (b) Linear Triangulation: Scale down your reconstructed points by a factor of 0.475.

   (c) Kanatani's method: Depend on which method you pick to generate the initial guess, *i.e.*, symmedian point triangulation or linear triangulation, scale down the reconstructed points by a factor of 0.625 or 0.475, respectively.

   Calculate the mean and standard deviation of the 3D point distances to the ground truth plane. How much improvement does 3-view triangulation gain compared to 2-view triangulation? (15 points)

Problem 2. Bundle Adjustment In the lecture, we learned that bundle adjustment (BA) is used to refine both the estimated pose and the 3D reconstructed points together by minimizing the reprojection errors. The variables we are perturbing in BA include 3D points and the camera poses. Suppose there are N cameras and M 3D points, then the total number of variables is $3M + 6N$: each 3D point location has 3 variables and each camera has 3 Euler angles describing the rotation and another 3 for translation. In this problem, we aim to implement a bundle adjustment on three camera poses and many 3D points acquired by three images, Figure 2.

Figure 2: Bundle adjustment optimizes the initial estimated camera poses of the three images and their corresponding many triangulated 3D points together in one shot.

The data used to do BA is already processed and provided in this lab. Specifically,

1. **Three camera poses** are provided individually. Rotation and translation of the first camera are given by `Rotation_View1.mat` and `Translation_View1.mat`, respectively. For another two camera poses, the file names follow the same pattern. These camera poses are under the world coordinate, obtained by matching SIFT features and estimating relative poses under a RANSAC framework, just like you did before.

2. **Intrinsic Matrix** is provided by `Intrinsic_Matrix.mat`. The focal length of the y-direction is negative because of the irregular coordinate the dataset used when generating the images. Just go straight to use it without hesitation.

3. **3D Triangulated Points** are provided by `Points3D.mat`. When you load the data, it is an array named `Gamma_w`: a set of 3D points under the world coordinate. There are 249 3D points visible by either only the first and second views, only the first and the third views, or all the three views.

4. **BA Data** which describes the connection between the camera, feature locations, 3D points, *etc*, given by `BA_Data.mat`. The array has four columns, and for each row, the data is

   `<Camera ID> <3D Point ID> <SIFT Location x> <SIFT Location y>`

   where `<Camera ID>` is either 1, 2, or 3; `<3D Point ID>` is the index of the 3D point visible by that camera, pointing to the row index of the array `Gamma_w`;
   `<SIFT location x>` and `<SIFT location y>` are the coordinate of the SIFT feature location to which the 3D point indexed by `<3D Point ID>` should project. A

screenshot of a subset of the `BA_Data` is shown in Figure 3: the first row says that 3D point 1 is visible by camera 1 which has the corresponding SIFT location (134.9131, 91.9117); the second rows says that 3D point 1 is also visible by camera 2 which has the corresponding SIFT location (8.3182, 50.1115).

	1	2	3	4
1	1	1	134.9131	91.9117
2	2	1	8.3182	50.1115
3	1	2	154.5204	77.4262

Figure 3: A screenshot of a subset of `BA_Data` shows how the cameras, 3D points, and observed SIFT feature locations are structured.

The total number of variables is thus $249\times3 + 6\times3 = 765$. Using the provided data, follow the steps below to complete your BA. Put your code in the provided `P2_main.m` file. (25 points)

1. **Organize the variables:** create a 1×765 array called `x0` containing all variables. Insert the camera poses and 3D points loaded by the provided data to that array. They are the initial guess for the BA optimization. For the camera poses, use MATLAB's built-in function `rotm2eul` to convert a rotation matrix to 3 Euler angles. In the array `x0`, you have the freedom to decide the order of the variables, *e.g.* the first 18 variables are 3 camera poses, and the last 747 variables are the 3D points.

2. **Create an Energy Function:** Similar to the implementation of the Kanatani's triangulation, create an energy function:

```
function Error = getReprojectionError(BA_Data, K, x)
```

   which accepts the `BA_Data` array, the intrinsic matrix K, and the variable array `x`, and return the sum of all squared reprojection error. In your energy function, you can convert the Euler angles to a rotation matrix by `eul2rotm` function.

3. **Minimize the Energy Function:** Use MATLAB's `lsqnonlin` solver, minimize the energy function you defined in the previous step via Levenberg-Marquardt algorithm. A sample piece of code is given below.

```
ff = @(x)(getReprojectionError(BA_Data, K, x));
options = optimoptions('lsqnonlin', 'Display', 'iter');
options.Algorithm = 'levenberg-marquardt';
BA_Output = lsqnonlin(ff, x0, [], [], options);
```

   Since the number of variables is comparatively large in the BA optimization, it would take about a minute to complete using around 60-70 iterations. You shall observe the drop of the residuals during the optimization.

Once your BA completes, the output of `lsqnonlin` is the optimized variables. Compute the mean and standard deviations of the reprojection error before and after BA. How much improvement does your BA give? Also show the distributions of the reprojection errors before and after BA. (5 points)