ENGN2560 Computer Vision
Lab04: Absolute Pose and Visual Odometry

The goal of this lab is to:

- Learn how an absolute pose can be estimated through 3D-2D correspondences.
- Learn how a visual odometry works and explore how the trajectory drift can be reduced by introducing absolute pose estimation and keyframes

Problem 1. Absolute Pose Estimation In the lecture, we learned that an absolute camera pose can be found from at least three 3D-2D correspondences through a Perspective-n-Point (PnP) algorithm. Given three images, the goal of this problem is to estimate the absolute pose of the third image under a RANSAC framework from correspondences between its 2D feature points and 3D points triangulated by the first two images, Figure 1. The camera intrinsic matrix is assumed to be known and is given. Implement an absolute pose estimation in the provided `P1_main.m` by following the steps below. (15 Points)

1. **Triangulate from the First Two Images:** This is the same as you did in the previous lab, namely, *(i)* extract and match SIFT features from the two images `Img1` and `Img2`, *(ii)* estimate the relative pose under a RANSAC loop, *(iii)* triangulate inlier feature correspondences. You can use either one of the triangulation methods from the previous lab. The triangulated 3D points should locate under the first camera coordinate.

2. **Construct 3D-2D Correspondences:** Extract SIFT features from the third image `Img3` and match them with the features on the second image `Img2`. On the second image, isolate the matched features which have corresponding 3D points triangulated from the previous step. These 3D points are thus in correspondence with the 2D feature points on the third image which are matched with the isolated features on the second image. This is the 3D-2D correspondences you will need in the next step.

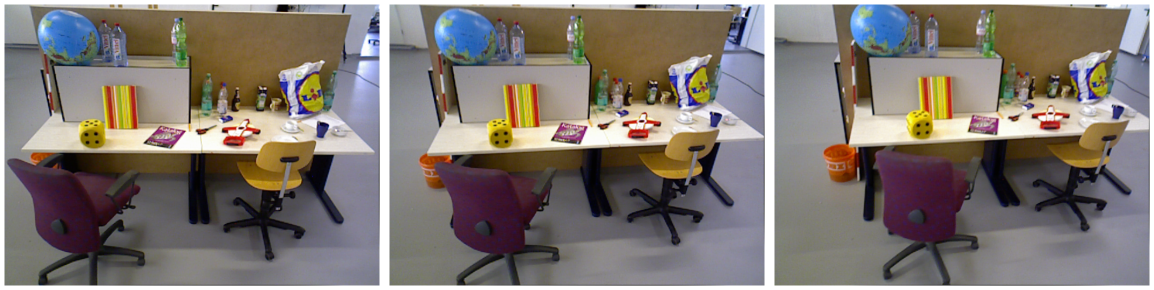

Figure 1: The absolute camera pose of the third image `Img3` (right) can estimated from the correspondences of its 2D observed feature points and the 3D points triangulated by the first two images (`Img1` on the left and `Img2` in the middle). The absolute pose is a pose under the coordinate of the 3D points, namely, the coordinate of the first camera.

3. **Estimate the Absolute Pose Under a RANSAC Scheme:** Create the following function to which the 3D-2D correspondences feed:

```
function [AbsR, AbsT] = Ransac4AbsPose(PARAMS, Points3D, Points2D, ...
K)
```

where PARAMS is a structure of parameters passed to the RANSAC, Points3D is a list of 3D points in correspondence with the 2D points in Points2D, and K is the camera intrinsic matrix. In this function,

   (a) Randomly pick 3 samples from the 3D-2D correspondences.

   (b) Feed the samples to the provided P3P_LambdaTwist.m function, an implementation of a recent P3P solver which returns all (possibly 2 or 4) valid absolute rotation and translation in respective arrays. Follow the instructions in that file to properly structure your inputs and examine how the output is organized.

   (c) For each valid absolute rotation and translation, reproject 3D points from Points3D to the third image. Calculate the reprojection errors with the 2D observations from Points2D. Count the number of inliers for which the reprojection error is less than a threshold defined by PARAMS.INLIER_THRESH.

   (d) Repeat steps (a) to (c) over an iterative loop. The number of iterations is defined by PARAMS.RANSAC_ITERATIONS.

   (e) Finally, after the iterative loops ends, pick the absolute rotation and translation supported by the maximal number of inliers as your outputs AbsR and AbsT, respectively.

4. **Evaluate the Estimated Absolute Pose:** Similar to Problem 5 of Lab02, let $\mathcal{R}_g$ and $\mathcal{T}_g$ be the ground truth rotation and translation, while $\mathcal{R}$ and $\mathcal{T}$ be the estimated absolute pose.

   (a) The error of your estimated R can be measured by

$$\Delta\mathcal{R}\left(\mathcal{R}_g, \mathcal{R}\right) = \arccos\left(\frac{trace\left(\mathcal{R}_g^T\mathcal{R}\right) - 1}{2}\right). \tag{1}$$

   (b) The estimated translation $\mathcal{T}$ is up to a scale due to metric ambiguity. Normalize both the ground truth and your estimated $\mathcal{T}$ so that their lengths are 1. Then, the error of your estimated $\mathcal{T}$ is

$$\Delta\mathcal{T}\left(\mathcal{T}_g, \mathcal{T}\right) = \left|\langle\mathcal{T}_g, \mathcal{T}\rangle - 1\right|, \tag{2}$$

   where $\langle\mathcal{T}_g, \mathcal{T}\rangle$ is the dot product of $\mathcal{T}_g$ and $\mathcal{T}$.

Evaluate the estimated absolute pose using the provided ground truth pose. What are the errors of your absolute rotation and translation?

An alternative approach to estimate the pose of image 3 with respect in the coordinate ot image 1 is by *(i)* first estimating relative pose between image 1 and 2, and another relative between image 2 and 3; *(ii)* propogating the two realtive poses to get the relative pose of image 3 with respect to image 1. Evaluate its accuracy. Which method of estimating the camera pose of the third image is more accurate? Why and why not?

Problem 2: Visual Odometry Part I Visual odometry (VO) is a technique which estimates camera poses given a sequence of images, generating a camera trajectory describing how the camera moves when capturing those images. A naive approach is to do relative pose estimation between consecutive frames, and continues to the end of the frame. The goal of this problem is to implement this approach using the provided 180 images from a real dataset, Figure 2, by following the steps below. (25 Points)

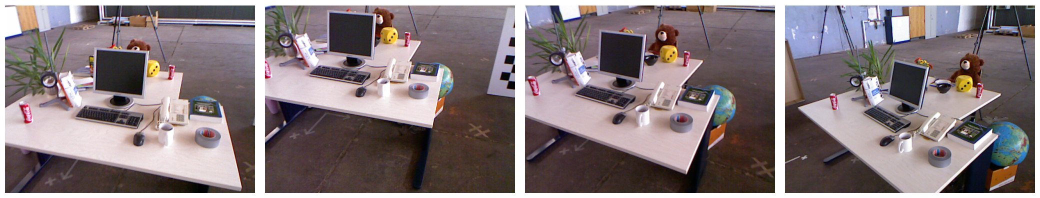

Figure 2: (From left to right) Frames 1, 57, 113, and 176 of the a sequence of 180 images.

1. **Estimate Relative Pose:** Set the first camera pose $(\mathcal{R}_1, \mathcal{T}_1)$ as the origin of world coordinate, *i,e,*, $\mathcal{R}_1 = I$ and $\mathcal{T}_1 = [0\ \ 0\ \ 0]^T$ and loop over the rest of the images. For each image i, $i \geq 2$, estimate the relative pose with respect to the previous image $i-1$ as you had done in previous labs. It is highly suggested to vectorize the code of your RANSAC essential matrix estimation, as it would take a lot of time to estimate relative poses from all consecutive frames of 180 images.

2. **Rescale the Relative Pose:** Each estimated relative pose $(\mathcal{R}_{i,i-1}, \mathcal{T}_{i,i-1})$ is up to a scale due to metric ambiguity, *i.e.*, the magnitude of the relative translation vector is always 1. To compare the estimation accuracy with the ground truths, each relative pose has to be scaled properly using the scale of the ground truth. Let $(\mathcal{R}_{g,i}, \mathcal{T}_{g,i})$ be the ground truth pose i so that the camera center $c_{g,i} = -\mathcal{R}_{g,i}^T \mathcal{T}_{g,i}$. The relative scale s of camera pose i with respect to $i-1$ is
$$s = \left\| c_{g,i} - c_{g,i-1} \right\|, \tag{3}$$
such that the scaled relative pose is $(\mathcal{R}_{i,i-1}, s\mathcal{T}_{i,i-1})$.

3. **Propogate Relative Poses:** To create a camera trajectory, all camera poses have to be in the same coordinate, *e.g.*, the world coordinate. Propogate the scaled estimated relative pose so that all poses are located in the first camera coordinate.

4. **Visualize the Trajectory:** A function `Visualize_Trajectory.m` is provided for you to plot your visual odometry results. Refer to the instructions in that function for more information on the input arguments. An example result is given in Figure 3, showing how the relative pose error is propogated as the camera moves, leading to a motion drift.

5. **Evaluate the Trajectory Accuracy:** One approach to measure the accuracy of your trajectory is by means of the root mean square error (RMSE) for rotations and translations. Let $\Delta\mathcal{R}_i$ and $\Delta\mathcal{T}_i$ be the rotation and translation error of pose i using Equations 1 and 2, respectively. The RMSE for rotation and translation are,
$$RMSE(\mathcal{R}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \Delta\mathcal{R}_i^2}, \quad RMSE(\mathcal{T}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \Delta\mathcal{T}_i^2}. \tag{4}$$

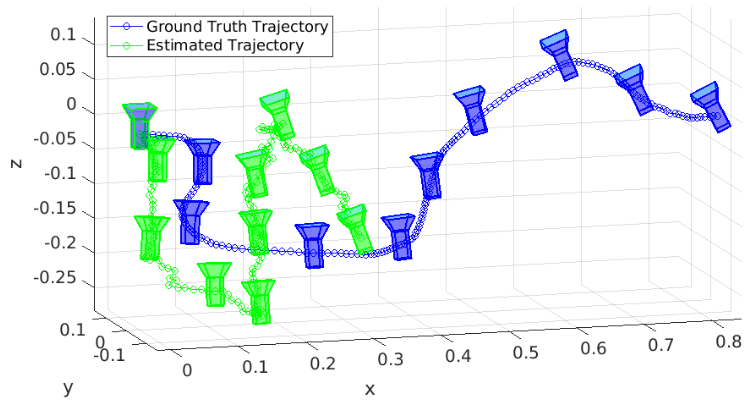

Figure 3: A naive visual odometry relying only on propogated relative poses produces a trajectory (green) with a significant motion drift, compared to the ground truth trajectory (blue). Each circle is a camera pose, while the plotted cameras are selected every 20 frames uniformly from the image sequence.

Put your code in the provided `P2_main.m` file. Show your visualized trajectory and the accuracy in the report.

Problem 3: Visual Odometry Part II In the lecture, we learned multiple ways to reduce the camera motion drift imposed by the accumulated relative pose error we observed in the previous problem. This includes absolute pose estimation and keyframes, both of which are introduced in our visual odometry pipeline as the goal of this problem. We will use the same image sequence in Problem 2. Follow the steps below. (60 Points)

1. **Initialization:** Set the first camera pose $(\mathcal{R}_1, \mathcal{T}_1)$ as the origin of world coordinate, *i,e,*, $\mathcal{R}_1 = I$ and $\mathcal{T}_1 = [0\ \ 0\ \ 0]^T$ and loop over the rest of the images. For the first two frames, estimate their relative pose, and triangulate inlier feature correspondences to create a bunch of 3D points located in the first camera coordinate. The first two images are marked as keyframes.

2. **Estimate Absolute Camera Pose:** For each image i, $i \geq 3$, match 2D features with the features from the last keyframe, and estimate the absolute camera pose i. This is similar to Problem 1.

3. **Keyframe Selection:** Decide whether image i is a keyframe based on the following two criterion:

   (a) If the number of images between image i and the last keyframe is greater than a threshold, *e.g.* 20 images, then enforce image i as a keyframe. This threshold is defined by `PARAMS.NUM_OF_FRAMES_FROM_LAST_KF`.

   (b) Let the total number of feature correspondences between image i and the last keyframe be N_F. Among them, a total of N_p features have corresponding 3D points, while N_q features do not, so that $N_F = N_p + N_q$. If the ratio $\frac{N_q}{N_F}$ drops below a threshold, *e.g.* 50%, then mark image i as a keyframe. This threshold is defined by `PARAMS.RATIO_OF_COVISIBLE_POINTS_FROM_LAST_KF`.

   Note that in the lecture, we say the second threshold is 25%; however, as a practice in this lab, 50% is a better choice.

4. **Triangulate Newly Observed Features:** If image i is a keyframe, then do the following steps. Otherwise, return to step 2.

   (a) Using the estimated absolute pose you had in step 2, pick inliers from N_q feature correspondences by the distances from points to epipolar lines. This requires you to compute the relative pose of image i with respect to the last keyframe.

   (b) Triangulate inlier feature correspondences. If your triangulation method uses the relative pose of image i with respect to the last keyframe, the triangulated 3D points are located in the coordinate of the last keyframe. Thus, transform the triangulated 3D points from the coordinate of the last keyframe to the coordinate of the first frame to keep all 3D points in the same coordinate.

   (c) Store *(i)* the newly triangulated points, and *(ii)* the points that were already in the 3D space visible by the image i, *i.e.*, the 3D points you used for estimating the absolute pose of image i in step 2. These points are then used for absolute pose estimation of subsequent images.

(d) Mark image i as the keyframe. Store all indices of the keyframes throughout the entire image sequence.

5. **Scale the Camera Poses:** Although there is metric ambiguity which requires the scale from the ground truth poses to properly scale up/down our estimated poses, we only need one scale in this problem as all poses are estimated with respect to the same coordinate (the first camera coordinate). Thus, compute the scale s from the first two frames using Equation 3, and scale up/down your estimated pose.

6. **Visualize the Trajectory:** Similar to Problem 2, use the provided function to show the trajectory. An example is given in Figure 4. Make sure to plot the cameras associated with keyframes selected from the pipeline.

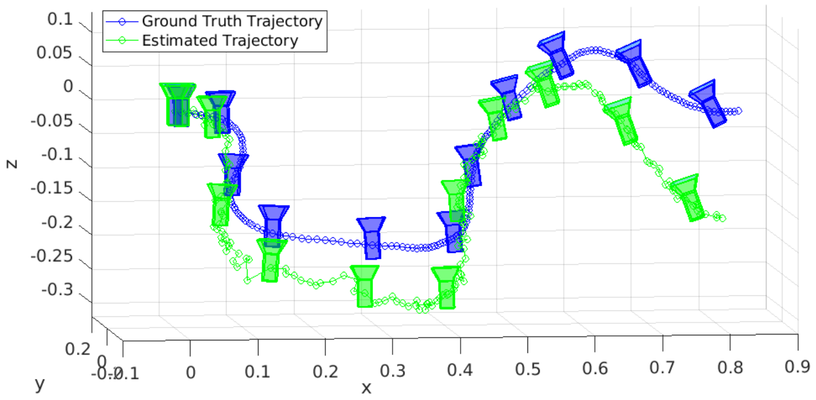

Figure 4: Using absolute pose estimation and the idea of keyframes, the camera motion drift is significantly reduced. The plotted cameras are the keyframes selected by the pipeline.

7. **Evaluate the Trajectory Accuracy:** Similar to Problem 2, use RMSE to measure the accuracy of your estimated camera trajectory.

Put your code in the provided `P3_main.m` file. Show your visualized trajectory and the accuracy in the report. How much improvement does the second visual odometry pipeline (Problem 3) provide over the first pipeline (Problem 2)?