Programming Assignment - 4
Structure from Motion and Multi-view Stereo
EE6132

TA: Salman Siddique Khan

November 23, 2020

Notes:

1. Please use moodle dicussion threads for posting your doubts.
2. Before posting any question, check if the same question has been asked earlier.
3. Submit a single zip file in the moodle named as PA4_Rollno.zip containing report and folders containing corresponding codes.
4. Read the problem fully to understand the whole procedure.
5. Comment your code generously.
6. Put titles to all of the figures shown.
7. You are supposed to use either MATLAB or Python for this assignment.

Data can be found at:
`https://tinyurl.com/mcv2020`

1 Problem Statement

In this assignment, you are required to implement a pipeline for dense depth reconstruction from a sequence of images captured using a calibrated smartphone camera.

2 Tasks

1. **Structure from Motion**. In the shared drive folder, there is a .mat file named 'matchedPoints.mat'. The file contains the matched keypoints (in image coordinates) across the sequence of 25 frames. Use these points and the camera intrinsic matrix (also provided in the shared drive), to perform bundle adjustment and obtain the 3D points and the camera poses for the different views. To make the optimization process simpler, use the following two assumption:

   - You can make small angle approximation i.e. $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. As a result of this assumption, your 3D rotation matrix for each view can be written as follows:

$$R_i \approx \begin{bmatrix} 1 & -\theta_i^z & \theta_i^y \\ \theta_i^z & 1 & -\theta_i^x \\ -\theta_i^y & \theta_i^x & 1 \end{bmatrix} \tag{1}$$

Here $\Theta_i = [\theta_i^x, \theta_i^y, \theta_i^z]$, is the angular displacement of the camera in the i^{th} view.

- Instead of solving for all the coordinates (X,Y and Z) of the 3-D points, you can parametrize the points by their inverse depth reducing the number of unknowns and making the optimization easier. Specifically, if (x_j, y_j) is the projection of the j^{th} 3-D point in the reference frame[1], then the same j^{th} 3-D point can be represented as $P_j = \frac{1}{w_j}[x_j, y_j, 1]^T$. Here, $w_j = \frac{1}{z_j}$ is the inverse-depth of the j^{th} 3-D point.

The projection of 3-D point P_j in the i^{th} image is $p_{ij} = [p_{ij}^x, p_{ij}^y]^T$. $\pi : \mathcal{R}^3 \rightarrow \mathcal{R}^2$ is the projection function i.e. $\pi([x, y, z]^T) = [x/z, y/z]^T$. Correspondingly the cost function for bundle adjustment becomes,

$$F = \sum_{i=1}^{N_c} \sum_{j=1}^{N_p} ||p_{ij} - \pi(R_i P_j + T_i)||^2 \tag{2}$$

Here N_p is the number of 3-D points to reconstruct and N_c is the number of views (or frames). You can use your favorite optimization routine (like lsqnonlin() in MATLAB or scipy.optimize.least_squares() in python) to minimize F in 2. F in Equation 2 is minimized with respect to the pose for each view and the inverse depth w_j for each point. The steps involved in SfM are as follows:

(a) The matched key-points provided in the .mat file are in image coordinates. Write a function that converts the matched points given in image coordinates to the normalized camera coordinates compatible with equation 2. To do this, subtract the principal point from the matched points provided to you and divide the difference by the focal length. For example, if one of the matched points is $m_{ij} = [m_{ij}^x, m_{ij}^y]$ in image coordinates, then to obtain $p_{ij} = [p_{ij}^x, p_{ij}^y]$ in normalized camera coordinates, perform the following operation $p_{ij}^x = (m_{ij}^x - c_x)/f_x$ and $p_{ij}^y = (m_{ij}^y - c_y)/f_y$. Here $[c_x, c_y]$ is the principal point and f_x and f_y are the focal lengths. These can be obtained from the intrinsic matrix provided to you.

(b) Write a projection function that takes a 3D point in world coordinate (P_j), rotation angles (Θ_i), and translation vector (T_i) and projects it to the normalized camera coordinate i.e. performs $\pi(R_i P_j + T_i)$.

(c) Using the projection function just defined, write a function that calculates the reprojection error given the rotation angles, translation vector of all the views, and the inverse depth of all the 3D points.

(d) Using an optimization routine (e.g. lsqnonlin in MATLAB or scipy.optimize.least_squares in python), minimize the reprojection error function defined above with respect to the rotation angles (Θ_i) and translation (T_i) for each view and the inverse depth (w_j) of each 3D point. You can initialize the Θ_i and T_i as zero vectors and w_j as vector of ones. You can choose the first frame as the reference frame.

Once you obtain the 3-D points, plot them as a 3-D point cloud. Perform this experiment for 5, 15 and 25 frames and report the point cloud obtained in each case.

2. **Plane Sweep.** Minimizing F in Equation 2 provided us with depths of a sparse set of 3-D points with respect the reference frame. To obtain dense depth reconstruction, use the camera rotation matrices and translation vectors obtained from the bundle adjustment problem of Equation 2 along with the sequence provided in the shared folder in a plane-sweeping framework. For plane-sweeping algorithm, one needs to find the plane-induced homography. For a plane at depth d and with normal vector $\boldsymbol{n}$, the plane-induced homography mapping the reference frame to the i^{th} frame is given as

$$H_{d,i} = K[R_i - (\boldsymbol{n}T_i^T)/d]K^{-1} \tag{3}$$

As we are only concerned with the planes parallel to the reference frame, $\boldsymbol{n} = [0, 0, -1]^T$. K is the intrinsic matrix for the camera. You can use a set of 10 candidate depth planes within the minimum

[1] (x_j, y_j) is in normalized camera coordinates i.e. after subtraction of principle coordinate from the image coordinates and division of the difference by the focal length in each direction.

and the maximum depth obtained from the above bundle adjustment problem. The steps involved in plane sweeping are as follows:

(a) Map the i^{th} frame to the reference frame through the plane induced homography $H_{d,i}^{-1}$. Perform this warping for each frame and each candidate depth plane. Stack the warped frames into a tensor. This tensor will be of size $H \times W \times D \times N_c$ where H and W are the height and width of the frames, D is the number of candidate depth plane and N_c is the number of views or frames.

(b) Find the variance across all the warped frames for each candidate depth. This will reduce the tensor to a cost volume of dimension $H \times W \times D$. Find the depth for each pixel as the one that gives the minimum variance.

Plot the obtained depth map alongside the reference frame. Perform this experiment for 5, 15 and 25 frames and report the depth map obtained in each case. How does increasing the number of frames change the quality of the depth map reconstruction?