CSCI 5980: Assignment #5

Bundle Adjustment

1 Submission

- Assignment due: April 20
- Individual assignment.
- Write-up submission format: a single PDF up to 10 pages (more than 10 page assignment will be automatically returned.).
- MATLAB code submission in conjunction with visualization. The code must be stand-alone. If it does not run my labtop (MATLAB 2016a), no point for programming parts will be given.
- Data submission (input images and result including reconstruction snapshot and reprojection error)
- Submission through Moodle.

CSCI 5980: Assignment #5

Bundle Adjustment

2 Data Capture and Overview

Figure 1: You will capture your cellphone images to reconstruct camera pose and 3D points.

In this assignment, you will use your cellphone images (more than 5) to reconstruct 3D camera poses and points with full bundle adjustment. Make sure you have enough baseline (translation) between images for well conditioned fundamental matrix while retaining enough number of correspondences between image. Avoid a scene dominated by a planar surface, i.e., the images need to contain many 3D objects as shown in Figure 1.

CSCI 5980: Assignment #5

Bundle Adjustment

You will write a full pipeline of the structure from motion algorithm including matching, camera pose estimation using fundamental matrix, PnP, triangulation, and bundle adjustment. A nonlinear optimization is always followed by the initial estimate by linear least squares solution. The pipeline is described in Algorithm 1.

Algorithm 1 Structure from Motion

[Mx, My] = `GetMatches`($\mathcal{I}_1, \cdots, \mathcal{I}_N$)
Normalize coordinate in Mx and My, i.e., $\mathbf{x} = \mathbf{K}^{-1}\mathbf{x}$.
Select two images $\mathcal{I}_{i1}$ and $\mathcal{I}_{i2}$ for the initial pair reconstruction.
[$\mathbf{R}$, $\mathbf{C}$, $\mathbf{X}$] = `CameraPoseEstimation`([Mx(:,$i1$) My(:,$i1$)], [Mx(:,$i2$) My(:,$i2$)])
$\mathcal{P} = \{\mathbf{P}_1, \mathbf{P}_2\}$ where $\mathbf{P}_{i1} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} \end{bmatrix}$, $\mathbf{P}_{i2} = \mathbf{R}\begin{bmatrix} \mathbf{I}_3 & -\mathbf{C} \end{bmatrix}$
$\mathcal{R} = \{i1, i2\}$
while $|\mathcal{R}| < N$ **do**
    i = `GetBestFrame`(Mx, My, $\mathcal{R}$);
    [$\mathbf{R}_i$, $\mathbf{C}_i$] = `PnP_RANSAC`([Mx(:,i) My(:,i)], $\mathbf{X}$)
    [$\mathbf{R}_i$, $\mathbf{C}_i$] = `PnP_Nonlinear`($\mathbf{R}_i$ $\mathbf{C}_i$, [Mx(:,i) My(:,i)], $\mathbf{X}$)
    $\mathbf{P}_i = \mathbf{R}_i \begin{bmatrix} \mathbf{I}_3 & -\mathbf{C}_i \end{bmatrix}$
    for f = 1 : $|\mathcal{R}|$ **do**
        $\mathcal{U}$ = `FindUnreconstructedPoints`($\mathbf{X}$, $\mathcal{R}_f$, i, Mx, My)
        for j = 1 : $|\mathcal{U}|$ **do**
            $\mathbf{u}$ = [Mx($\mathcal{U}_j$, i), My($\mathcal{U}_j$, i)] and $\mathbf{v}$ = [Mx($\mathcal{U}_j$, $\mathcal{R}_f$), My($\mathcal{U}_j$, $\mathcal{R}_f$)]
            $\mathbf{x}$ = `LinearTriangulation`($\mathbf{u}$, $\mathbf{P}_i$, $\mathbf{v}$, $\mathbf{P}_{\mathcal{R}_f}$)
            $\mathbf{x}$ = `NonlinearTriangulation`($\mathbf{X}$, $\mathbf{u}$, $\mathbf{R}_i$, $\mathbf{C}_i$, $\mathbf{v}$, $\mathbf{R}_{\mathcal{R}_f}$, $\mathbf{C}_{\mathcal{R}_f}$)
            $\mathbf{X} = \mathbf{X} \cup \mathbf{x}$
        end for
    end for
    $\mathcal{P} = \mathcal{P} \cup \mathbf{P}_i$ and $\mathcal{R} = \mathcal{R} \cup i$.
    [$\mathcal{P}$, $\mathbf{X}$] = `BundleAdjustment`($\mathcal{P}$, $\mathbf{X}$, $\mathcal{R}$, Mx, My)
end while

CSCI 5980: Assignment #5

Bundle Adjustment

3 Matching

Given a set of images, $\mathcal{I}_1, \cdots, \mathcal{I}_N$, you will find matches across all images where N is the number of images similar to HW #4. Pick a reference image, $\mathcal{I}_{ref}$, and match with other images using SIFT features from VLFeat, i.e., $\mathcal{I}_{ref} \leftrightarrow \mathcal{I}_1, \cdots, \mathcal{I}_{ref} \leftrightarrow \mathcal{I}_N$ (no need to match $\mathcal{I}_{ref} \leftrightarrow \mathcal{I}_{ref}$).

Your matches are outlier free, i.e., bidirectional knn match → ratio test → inliers from the fundamental matrix based RANSAC. Based on the matches, you will build a measurement matrix, Mx and My:

`[Mx, My] = GetMatches(`$\mathcal{I}_1$`, ⋯, `$\mathcal{I}_N$`)`
Mx: F×N matrix storing x coordinate of correspondences
My: F×N matrix storing y coordinate of correspondences

The f^{th} feature point in image $\mathcal{I}_i$ corresponds to a point in image $\mathcal{I}_j$. The x and y coordinates of the correspondence is stored at (f, i) and (f, j) elements in Mx and My, respectively. If (f, i) does not correspond to any point in image $\mathcal{I}_k$, you set -1 to indicate no match as shown in Figure 2.

Important: For better numerical stability, you can transform the measurements to the normalized coordinate by multiplying $\mathbf{K}^{-1}$, i.e., $\mathbf{x} = \mathbf{K}^{-1}\mathbf{x}$ where $\mathbf{x}$ is 2D measured points in homogeneous coordinate. You can run structure from motion in the normalized coordinate by factoring out $\mathbf{K}$. When visualizing projection in the image, the coordinate needs to be transformed back to original coordinate by multiplying $\mathbf{K}$.

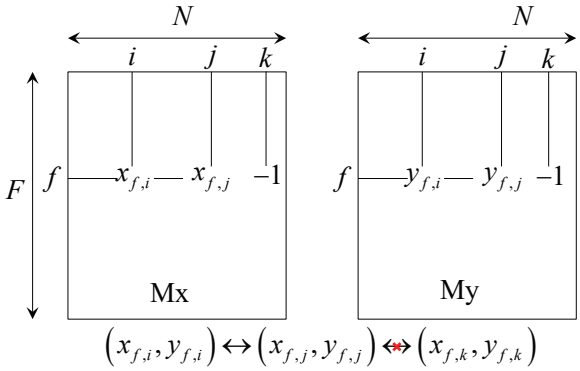


Figure 2: The f^{th} feature point in image $\mathcal{I}_i$ corresponds to a point in image $\mathcal{I}_j$. The x and y coordinates of the correspondence is stored at (f, i) and (f, j) elements in Mx and My, respectively. If (f, i) does not correspond to any point in image $\mathcal{I}_k$, you set -1 to indicate no match.

CSCI 5980: Assignment #5

Bundle Adjustment

4 Camera Pose Estimation

You will write a camera pose estimation code that takes correspondences between two images, $\mathcal{I}_{i1}$ and $\mathcal{I}_{i2}$ where $i1$ and $i2$ are the indices of the initial images to reconstruct selected manually.

$[\mathbf{R}, \mathbf{C}, \mathbf{X}] = \texttt{CameraPoseEstimation}(\mathbf{u}_1, \mathbf{u}_2)$

$\mathbf{R}$ and $\mathbf{C}$: the relative transformation of the $i2$ image
$\mathbf{u}_1$ and $\mathbf{u}_2$: 2D-2D correspondences

As studied in HW #4, you will compute:

1. Fundamental matrix via RANSAC on correspondences, Mx(:,$i1$), My(:,$i2$)
2. Essential matrix from the fundamental matrix
3. Four configurations of camera poses given the essential matrix
4. Disambiguation via chierality (using 3D point linear triangulation):
   $\mathbf{X} = \texttt{LinearTriangulation}(\mathbf{u}, \mathbf{P}_i, \mathbf{v}, \mathbf{P}_j)$

Write-up:

(a) Inlier matches (b) 3D camera pose

Figure 3: Camera pose estimation.

(1) Visualize inlier matches as shown in Figure 3(a).

(2) Visualize camera pose and 3D reconstructed points in 3D as shown in Figure 3(b).

CSCI 5980: Assignment #5

Bundle Adjustment

5 Nonlinear 3D Point Refinement

You will write a nonlinear triangulation code. Given the linear estimate for the point triangulation, $\mathbf{X}$, you will refine the 3D point $\mathbf{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^\mathsf{T}$ to minimize geometric error (reprojection error) via iterative nonlinear least squares estimation,

$$\Delta\mathbf{X} = \left(\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}^\mathsf{T} \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} \right)^{-1} \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}^\mathsf{T} \left(\mathbf{b} - f(\mathbf{X}) \right). \tag{1}$$

Write-up:

(1) Derive the point Jacobian, i.e., $\frac{\partial f(\mathbf{X})_j}{\partial \mathbf{X}}$ and write the following code.
`df_dX = JacobianX(K, R, C, X)`

(2) Write a code to refine the 3D point by minimizing the reprojection error and visualize reprojection error reduction similar to Figure 5.
$\mathbf{X}$ = `NonlinearTriangulation`($\mathbf{X}$, $\mathbf{u}_1$, $\mathbf{R}_1$, $\mathbf{C}_1$, $\mathbf{u}_2$, $\mathbf{R}_2$, $\mathbf{C}_2$)

Algorithm 2 Nonlinear Point Refinement

$\mathbf{b} = \begin{bmatrix} \mathbf{u}_1^\mathsf{T} & \mathbf{u}_2^\mathsf{T} \end{bmatrix}^\mathsf{T}$
for j = 1 : nIters **do**
    Build point Jacobian, $\frac{\partial f(\mathbf{X})_j}{\partial \mathbf{X}}$.
    Compute $f(\mathbf{X})$.
    $\Delta\mathbf{X} = \left(\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}^\mathsf{T} \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} + \lambda \mathbf{I} \right)^{-1} \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}^\mathsf{T} \left(\mathbf{b} - f(\mathbf{X}) \right)$
    $\mathbf{X} = \mathbf{X} + \Delta\mathbf{X}$
end for

CSCI 5980: Assignment #5

Bundle Adjustment

6 Camera Registration

You will register an additional image, $\mathcal{I}_j$ using 2D-3D correspondences.

Write-up:

(1) (3D-2D correspondences) Given 3D triangulated points, find 2D-3D matches, $\mathbf{X} \leftrightarrow \mathbf{u}$.

(2) (Perspective-n-Point algorithm) Write a code that computes 3D camera pose from 3D-2D correspondences:

`[R, C] = LinearPnP(u, X)`

$\mathbf{X}$: $n \times 3$ matrix containing n 3D reconstructed points
$\mathbf{u}$: $n \times 2$ matrix containing n 2D points in the additional image $\mathcal{I}_3$
$\mathbf{R}$ and $\mathbf{C}$: rotation and translation for the additional image.
Hint: After the linear solve, rectify the rotation matrix such that $\det(\mathbf{R}) = 1$ and scale $\mathbf{C}$ according to the rectification.

CSCI 5980: Assignment #5

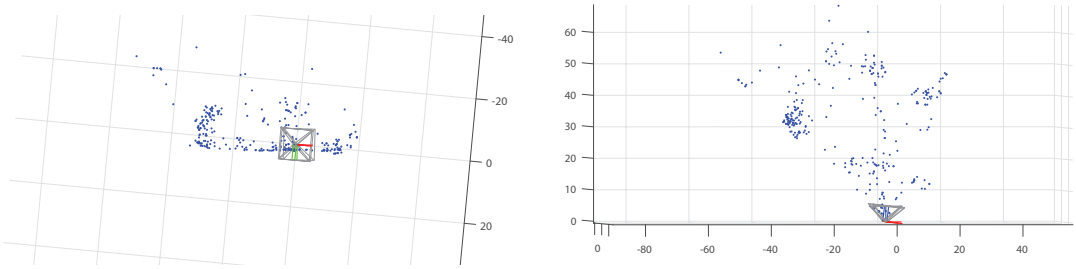
Bundle Adjustment

(3) (RANSAC PnP) Write a RANSAC algorithm for the camera pose registration (PnP) given n matches using the following pseudo code:

Algorithm 3 PnP RANSAC

$nInliers \leftarrow 0$
for $i = 1 : M$ **do**
    Choose 6 correspondences, $\mathbf{X}_r$ and $\mathbf{u}_r$, randomly from $\mathbf{X}$ and $\mathbf{u}$.
    `[`$\mathbf{R}_r$`, `$\mathbf{t}_r$`] = LinearPnP(`$\mathbf{u}_r$`, `$\mathbf{X}_r$`)`
    Compute the number of inliers, n_r, with respect to $\mathbf{R}_r$, $\mathbf{t}_r$.
    if $n_r > nInliers$ **then**
        $nInliers \leftarrow n_r$
        $\mathbf{R} = \mathbf{R}_r$ and $\mathbf{t} = \mathbf{t}_r$
    end if
end for

Visualize 3D registered pose as shown in Figure 4.



(a) Front view (b) Top view

Figure 4: Additional image registration.

(4) (Reprojection) Visualize measurement and reprojection to verify the solution.

CSCI 5980: Assignment #5

Bundle Adjustment

7 Nonlinear Camera Refinement

Given the initial estimate $\mathbf{R}_i$ and $\mathbf{t}_i$, you will refine the camera pose to minimize geometric error (reprojection error) via iterative nonlinear least squares estimation,

$$\Delta\mathbf{p} = \left(\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}^{\mathsf{T}} \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}\right)^{-1} \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}^{\mathsf{T}} \left(\mathbf{b} - f(\mathbf{p})\right),$$

$$f(\mathbf{p}) = \begin{bmatrix} u_1/w_1 \\ v_1/w_1 \\ \vdots \\ u_n/w_n \\ v_n/w_n \end{bmatrix}, \qquad \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{R}_i \begin{bmatrix} \mathbf{I}_3 & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \tag{2}$$

where $\mathbf{p} = \begin{bmatrix} \mathbf{C}^{\mathsf{T}} & \mathbf{q}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$. $\mathbf{C} \in \mathbb{R}^3$ is the camera optical center and $\mathbf{q} \in \mathbb{S}^3$ is the quaternion representation of the camera rotation.

It is possible to minimize the overshooting by adding damping, λ as follows:

$$\Delta\mathbf{p} = \left(\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}^{\mathsf{T}} \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}} + \lambda\mathbf{I}\right)^{-1} \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}^{\mathsf{T}} \left(\mathbf{b} - f(\mathbf{p})\right), \tag{3}$$

where λ is the damping parameter. You can try $\lambda \in [0, 10]$.

Note that the conversion between quaternion and rotation matrix is given as follows:

$$\mathbf{R} = \begin{bmatrix} 1 - 2q_z^2 - 2q_y^2 & -2q_z q_w + 2q_y q_x & 2q_y q_w + 2q_z q_x \\ 2q_x q_y + 2q_w q_z & 1 - 2q_z^2 - 2q_x^2 & 2q_z q_y - 2q_x q_w \\ 2q_x q_z - 2q_w q_y & 2q_y q_z + 2q_w q_x & 1 - 2q_y^2 - 2q_x^2 \end{bmatrix},$$

$$\mathbf{q} = \begin{bmatrix} q_w \\ (\mathbf{R}_{32} - \mathbf{R}_{23})/(4q_w) \\ (\mathbf{R}_{13} - \mathbf{R}_{31})/(4q_w) \\ (\mathbf{R}_{21} - \mathbf{R}_{12})/(4q_w) \end{bmatrix}, \quad \text{where} \quad q_w = \frac{\sqrt{1 + \mathbf{R}_{11} + \mathbf{R}_{22} + \mathbf{R}_{33}}}{2}, \qquad \|\mathbf{q}\| = 1 \tag{4}$$

CSCI 5980: Assignment #5

Bundle Adjustment

Write-up:

(1) Derive the quaternion Jacobian to rotation using Equation (4), i.e., $\frac{\partial \mathbf{R}}{\partial \mathbf{q}}$ and write the following code. Note: ignore the normalization $\|\mathbf{q}\| = 1$.
`dR_dq = JacobianQ(q)`

(2) Derive the rotation Jacobian to projection using Equation (2), i.e., $\frac{\partial f(\mathbf{p})_j}{\partial \mathbf{R}}$ where $f(\mathbf{p})_j = \begin{bmatrix} u_j/w_j & v_j/w_j \end{bmatrix}^\mathsf{T}$ and write the following code. Note: use vectorized form of the rotation matrix.
`df_dR = JacobianR(R, C, X)`

(3) Derive the expression of $\frac{\partial f(\mathbf{p})_j}{\partial \mathbf{q}}$ using the chain rule.

(4) Derive the camera center Jacobian to projection using Equation (2), i.e., $\frac{\partial f(\mathbf{p})_j}{\partial \mathbf{C}}$ and write the following code.
`df_dC = JacobianC(R, C, X)`

(5) Write a code to refine the camera pose by minimizing the reprojection error and visualize reprojection error reduction as shown in Figure 5:

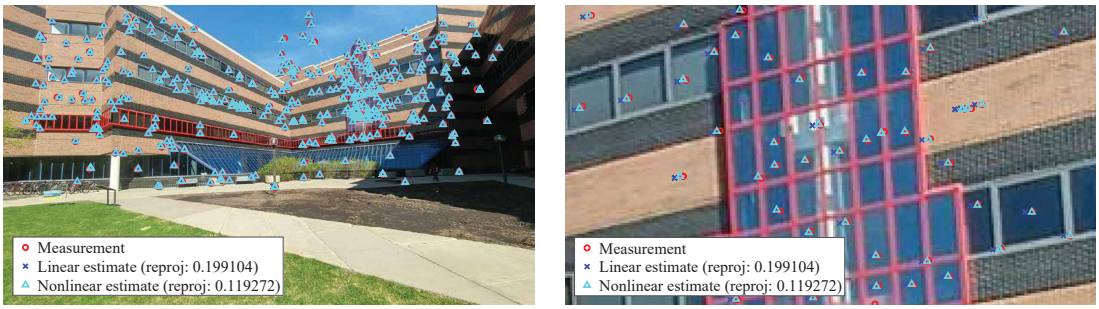
`[R, C] = PnP_Nonlinear(R C, u, X)`

Algorithm 4 Nonlinear Camera Pose Refinement

$\mathbf{p} = \begin{bmatrix} \mathbf{C}^\mathsf{T} & \mathbf{q}^\mathsf{T} \end{bmatrix}^\mathsf{T}$
for j = 1 : nIters **do**
  $\mathbf{C} = \mathbf{p}_{1:3}$, $\mathbf{R}$=`Quaternion2Rotation`$(\mathbf{q})$, $\mathbf{q} = \mathbf{p}_{4:7}$
  Build camera pose Jacobian for all points, $\frac{\partial f(\mathbf{p})_j}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial f(\mathbf{p})_j}{\partial \mathbf{C}} & \frac{\partial f(\mathbf{p})_j}{\partial \mathbf{q}} \end{bmatrix}$.
  Compute $f(\mathbf{p})$.
  $\Delta\mathbf{p} = \left(\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}^\mathsf{T} \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}} + \lambda \mathbf{I}\right)^{-1} \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}^\mathsf{T} (\mathbf{b} - f(\mathbf{p}))$ using Equation (3).
  $\mathbf{p} = \mathbf{p} + \Delta\mathbf{p}$
  Normalize the quaternion scale, $\mathbf{p}_{4:7} = \mathbf{p}_{4:7}/\|\mathbf{p}_{4:7}\|$.
end for

CSCI 5980: Assignment #5

Bundle Adjustment



(a) Reprojection error (b) Close up

Figure 5: Nonlinear refinement reduces the reprojection error (0.19→0.11).

8 Bundle Adjustment

You will write a nonlinear refinement code that simultaneously optimizes camera poses and 3D points using the sparse nature of the Jacobian matrix.
`[`$\mathcal{P}$`, X] = BundleAdjustient(`$\mathcal{P}$`, X, `$\mathcal{R}$`, Mx, My)`

For example, consider 3 camera poses and 2 points. The Jacobian matrix can be written as follows:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f(\mathbf{p}_1,\mathbf{X}_1)}{\partial \mathbf{p}_1} & \mathbf{0}_{2\times 7} & \mathbf{0}_{2\times 7} & \frac{\partial f(\mathbf{p}_1,\mathbf{X}_1)}{\partial \mathbf{X}_1} & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 7} & \frac{\partial f(\mathbf{p}_2,\mathbf{X}_1)}{\partial \mathbf{p}_2} & \mathbf{0}_{2\times 7} & \frac{\partial f(\mathbf{p}_2,\mathbf{X}_1)}{\partial \mathbf{X}_1} & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 7} & \mathbf{0}_{2\times 7} & \frac{\partial f(\mathbf{p}_3,\mathbf{X}_1)}{\partial \mathbf{p}_3} & \frac{\partial f(\mathbf{p}_3,\mathbf{X}_1)}{\partial \mathbf{X}_1} & \mathbf{0}_{2\times 3} \\ \frac{\partial f(\mathbf{p}_1,\mathbf{X}_2)}{\partial \mathbf{p}_1} & \mathbf{0}_{2\times 7} & \mathbf{0}_{2\times 7} & \mathbf{0}_{2\times 3} & \frac{\partial f(\mathbf{p}_1,\mathbf{X}_2)}{\partial \mathbf{X}_2} \\ \mathbf{0}_{2\times 7} & \frac{\partial f(\mathbf{p}_2,\mathbf{X}_2)}{\partial \mathbf{p}_2} & \mathbf{0}_{2\times 7} & \mathbf{0}_{2\times 3} & \frac{\partial f(\mathbf{p}_2,\mathbf{X}_2)}{\partial \mathbf{X}_2} \\ \mathbf{0}_{2\times 7} & \mathbf{0}_{2\times 7} & \frac{\partial f(\mathbf{p}_3,\mathbf{X}_2)}{\partial \mathbf{p}_3} & \mathbf{0}_{2\times 3} & \frac{\partial f(\mathbf{p}_3,\mathbf{X}_2)}{\partial \mathbf{X}_2} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_\mathbf{p} & \mathbf{J}_\mathbf{X} \end{bmatrix} \quad (5)$$

where $\mathbf{J}_\mathbf{p}$ and $\mathbf{J}_\mathbf{X}$ are the Jacobian for camera and point, respectively, and $\lambda \in [0, 10]$.

The normal equation, $\mathbf{J}^\mathsf{T}\mathbf{J}\Delta\mathbf{x} = \mathbf{J}^\mathsf{T}(\mathbf{b} - f(\mathbf{x}))$ can be decomposed into:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\mathsf{T} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta\widehat{\mathbf{p}} \\ \Delta\widehat{\mathbf{X}} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_\mathbf{p} \\ \mathbf{e}_\mathbf{X} \end{bmatrix}, \quad (6)$$

where

$$\mathbf{A} = \mathbf{J}_\mathbf{p}^\mathsf{T}\mathbf{J}_\mathbf{p} + \lambda\mathbf{I}, \qquad \mathbf{B} = \mathbf{J}_\mathbf{p}^\mathsf{T}\mathbf{J}_\mathbf{X}, \qquad \mathbf{D} = \mathbf{J}_\mathbf{X}^\mathsf{T}\mathbf{J}_\mathbf{X} + \lambda\mathbf{I}$$
$$\mathbf{e}_\mathbf{p} = \mathbf{J}_\mathbf{p}^\mathsf{T}(\mathbf{b} - f(\mathbf{x})), \qquad \mathbf{e}_\mathbf{X} = \mathbf{J}_\mathbf{X}^\mathsf{T}(\mathbf{b} - f(\mathbf{x}))$$

where $\widehat{\mathbf{p}} = \begin{bmatrix} \mathbf{p}_1^\mathsf{T} & \cdots & \mathbf{p}_I^\mathsf{T} \end{bmatrix}$ and $\widehat{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_1^\mathsf{T} & \cdots & \mathbf{X}_M^\mathsf{T} \end{bmatrix}$ where I and M are the number of images and points, respectively.

CSCI 5980: Assignment #5

Bundle Adjustment

The decomposed normal equation in Equation (6) allows us to efficiently compute the inverse of $\mathbf{J}^\mathsf{T}\mathbf{J}$ using Schur complement of $\mathbf{D}$:

$$\Delta\widehat{\mathbf{p}} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^\mathsf{T})^{-1}(\mathbf{e}_\mathbf{p} - \mathbf{B}\mathbf{D}^{-1}\mathbf{e}_\mathbf{X}),$$
$$\Delta\widehat{\mathbf{X}} = \mathbf{D}^{-1}(\mathbf{e}_\mathbf{X} - \mathbf{B}^\mathsf{T}\Delta\widehat{\mathbf{p}})$$

where $\mathbf{D}$ is a block diagonal matrix whose inverse can be efficiently computed by inverting small block matrix:

$$\mathbf{D} = \begin{bmatrix} \mathbf{d}_1 & & \\ & \ddots & \\ & & \mathbf{d}_M \end{bmatrix}, \qquad \mathbf{D}^{-1} = \begin{bmatrix} \mathbf{d}_1^{-1} & & \\ & \ddots & \\ & & \mathbf{d}_M^{-1} \end{bmatrix} \tag{7}$$

The bundle adjustment algorithm is summarized in Algorithm 5. Note that not all points are visible from cameras. You need to reason about the visibility, i.e., if the point is not visible from the camera, the corresponding Jacobian and measurement from $\mathbf{J}$ and $\mathbf{b}$ will be omitted, respectively.

CSCI 5980: Assignment #5

Bundle Adjustment

Algorithm 5 Bundle Adjustment

$\widehat{p} = \begin{bmatrix} \mathbf{p}_1^\mathsf{T} & \cdots & \mathbf{p}_I^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and $\widehat{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_1^\mathsf{T} & \cdots & \mathbf{X}_M^\mathsf{T} \end{bmatrix}$
for iter $= 1 :$ nIters **do**
  Empty $\mathbf{J_p}$, $\mathbf{J_X}$, $\mathbf{b}$, $\mathbf{f}$, $\mathbf{D}_{\text{inv}}$.
  for $i = 1 : M$ **do**
    $\mathbf{d} = \mathbf{0}_{3\times 3}$
    for $j = 1 : I$ **do**
      if the i^{th} point is visible from the j^{th} image **then**
        $\mathbf{J}_1 = \mathbf{0}_{2\times 7I}$ and $\mathbf{J}_2 = \mathbf{0}_{2\times 3M}$
        $\mathbf{J}_1(:, 7(j-1)+1 : 7j) = \frac{\partial f(\mathbf{p}_j, \mathbf{X}_i)}{\partial \mathbf{p}_j}$
        $\mathbf{J}_2(:, 3(i-1)+1 : 3i) = \frac{\partial f(\mathbf{p}_j, \mathbf{X}_i)}{\partial \mathbf{X}_i}$
        $\mathbf{J_p} = \begin{bmatrix} \mathbf{J}_\mathbf{p}^\mathsf{T} & \mathbf{J}_1^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and $\mathbf{J_X} = \begin{bmatrix} \mathbf{J}_\mathbf{X}^\mathsf{T} & \mathbf{J}_2^\mathsf{T} \end{bmatrix}^\mathsf{T}$
        $\mathbf{d} = \mathbf{d} + \frac{\partial f(\mathbf{p}_j, \mathbf{X}_i)}{\partial \mathbf{X}_i}^\mathsf{T} \frac{\partial f(\mathbf{p}_j, \mathbf{X}_i)}{\partial \mathbf{X}_i}$
        $\mathbf{b} = \begin{bmatrix} \mathbf{b}^\mathsf{T} & \mathbf{u}_{ij}^\mathsf{T} \end{bmatrix}$
        $\mathbf{f} = \begin{bmatrix} \mathbf{f}^\mathsf{T} & \mathbf{x}_{ij}^\mathsf{T} \end{bmatrix}$ where $\mu \begin{bmatrix} \mathbf{x}_{ij} \\ 1 \end{bmatrix} = \mathbf{R}_j \begin{bmatrix} \mathbf{I} & -\mathbf{C}_j \end{bmatrix}$
      end if
    end for
    $\mathbf{d} = \mathbf{d} + \lambda \mathbf{I}$
    $\mathbf{D}_{\text{inv}} = \texttt{blkdiag}(\mathbf{D}_{\text{inv}}, \mathbf{d}^{-1})$
  end for
  $\mathbf{e_p} = \mathbf{J}_\mathbf{p}^\mathsf{T}(\mathbf{b} - \mathbf{f})$
  $\mathbf{e_X} = \mathbf{J}_\mathbf{X}^\mathsf{T}(\mathbf{b} - \mathbf{f})$
  $\mathbf{A} = \mathbf{J}_\mathbf{p}^\mathsf{T}\mathbf{J_p} + \lambda\mathbf{I}$, $\mathbf{B} = \mathbf{J}_\mathbf{p}^\mathsf{T}\mathbf{J_X}$, $\mathbf{D}^{-1} = \mathbf{D}_{\text{inv}}$
  $\Delta\widehat{\mathbf{p}} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^\mathsf{T})^{-1}(\mathbf{e_p} - \mathbf{B}\mathbf{D}^{-1}\mathbf{e_X})$
  Normalize quaternions.
  $\Delta\widehat{\mathbf{X}} = \mathbf{D}^{-1}(\mathbf{e_X} - \mathbf{B}^\mathsf{T}\Delta\widehat{\mathbf{p}})$
end for

Write-up: You will first start with two images and 10 3D points to test your bundle adjustment program.

(1) Derive $\mathbf{J_p}$ and $\mathbf{J_X}$.

(2) Run Algorithm 5 and visualize the reprojection error similar to Figure 5.

CSCI 5980: Assignment #5

Bundle Adjustment

9 Putting All Things Together

Write-up: You will run with all images and 3D points based on Algorithm 1.

(1) Visualize 3D camera pose and points as shown in Figure 6.

(2) Visualize reprojection for all images.

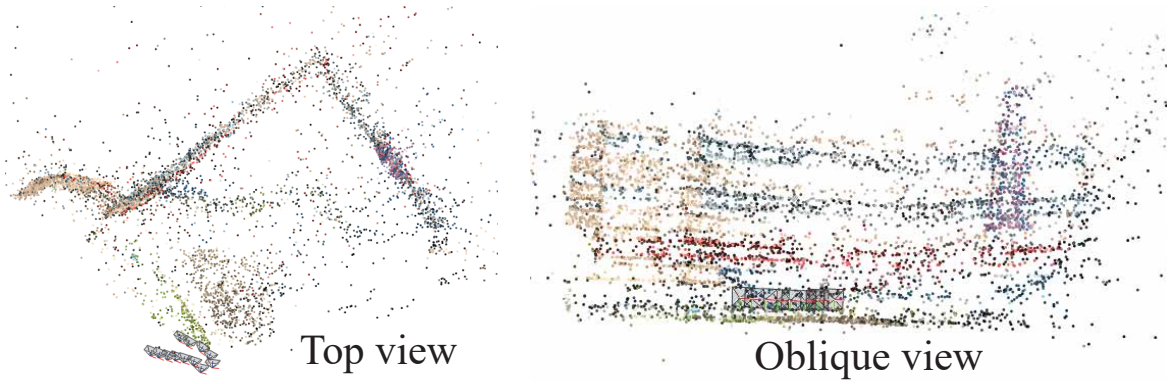


Figure 6: You will reconstruct all images and 3D points using structure from motion.