CMSC 733, Computer Processing of Pictorial Information
Project 3: Buildings built in minutes: An SfM Approach
Due on: 11:59:59PM on Sunday, April 02 2017

Prof. Yiannis Aloimonos,
Nitin J. Sanket

March 18, 2017

The aim of this project is to reconstruct a 3D scene and simultaneously obtain the camera poses of a monocular camera with respect to this scene and this procedure is called Structure from Motion (SfM). Your task is to implement the full pipeline of structure from motion including two view reconstruction, triangulation, PnP, and bundle adjustment. For nonlinear optimization parts, you are free to choose an optimizer such as built-in functions in MATLAB or Sparse Bundle Adjustment package found at http://users.ics.forth.gr/~lourakis/sba/. The steps are explained next. **Note that this project has to be done in groups of two and only one submission per group.**

1 Data

You are given 6 images of the building in-front of Levine Hall at University of Pennsylvania captured using a GoPro Hero 3 Black Edition with fisheye lens distortion corrected as shown in Fig. 1. The images can be found in `Data` folder in `.jpg` format. Keypoint matching (SIFT keypoints and descriptors used) data is also provided in the same folder for pairs of images. TThe data folder contains 5 matching files named `matching*.txt` where `*` refers to numbers from 1 to 5. For eg., `matching3.txt` contains the matching between the third image and the fourth, fifth and sixth images, i.e., $\mathcal{I}_3 \leftrightarrow \mathcal{I}_4$, $\mathcal{I}_3 \leftrightarrow \mathcal{I}_5$, and $\mathcal{I}_3 \leftrightarrow \mathcal{I}_6$. Therefore, `matching6.txt` does not exist because it is the matching by itself.

The file format of the matching file is described next. Each matching file is formatted as follows for the i^{th} matching file:

nFeatures: (the number of feature points of the i^{th} image - each following row specifies matches across images given a feature location in the i^{th} image.)
Each row: (the number of matches for the j^{th} feature) (Red Value) (Green Value) (Blue Value) ($\text{u}_{\text{current image}}$) ($\text{v}_{\text{current image}}$) (image id) ($\text{u}_{\text{image id image}}$) ($\text{v}_{\text{image id image}}$) (image id) ($\text{u}_{\text{image id image}}$) ($\text{v}_{\text{image id image}}$) $\cdots$

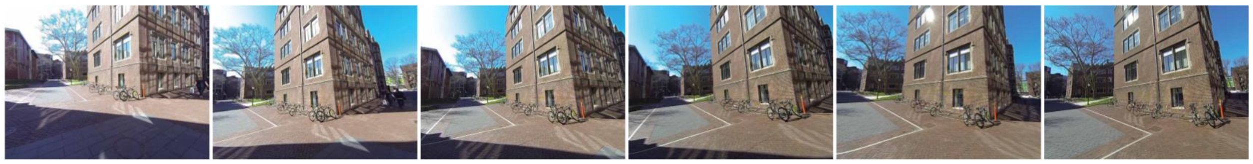

Figure 1: Input Images.

An example of `matching1.txt` is given below:

```
nFeatures: 2002
3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000
2 137 128 105 454.740000 392.370000 4 447.580000 479.360000
```

The images are taken at 1280×960 resolution and the camera intrinsic parameters K are given in `calibration.txt` file. You will program this full pipeline guided by the functions described in following sections. This pseudocode does not include data management, e.g., converting matches obtained from the data files to feature points.

2 Off-the-shelf algorithm: VSfM - 5Pts

Download and install an off-the-shelf structure from motion software VisualSfM (`http://ccwu.me/vsfm/`). Obtain the 3D model with the images given and put the results in your report.

3 Overview

The full pipeline you are expected to implement is give in Algorithm 1.

```
Data: Image Matches, $K$
Result: X, C, R
for all possible pair of images do
    // Reject outlier correspondences
    [x1, x2] = GetInliersRANSAC(x1, x2);
end
// For first two images
F = EstimateFundamentalMatrix(x1, x2);
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset, Rset] = ExtractCameraPose(E);
// Perform linear triangulation
for $i$ = 1:4 do
    Xseti = LinearTriangulation(K, zeros(3,1), eye(3), Cseti, Rseti, x1, x2);
end
// Check cheirality condition
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);
// Perform Non-linear triangulation
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2, X0));
Cset = C, Rset = R;
// Register camera and add 3D points for the rest of images
for $i$=3:I do
    // Register the $i^{\text{th}}$ image using PnP.
    [Cnew Rnew] = PnPRANSAC(X, x, K);
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
    Cset = Cset∪Cnew, Rset = Rset∪Rnew;
    // Add new 3D points.
    Xnew = LinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);
    Xnew = NonlinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2, X0);
    X = X∪Xnew;
    // Build Visibility Matrix.
    V = BuildVisibilityMatrix(traj);
    // Perform Bundle Adjustment.
    [Cset Rset X] = BundleAdjustment(Cset, Rset, X, K, traj, V);
end
```

Algorithm 1: Structure from Motion Pipeline.

4 Matching - 10Pts

In this section, you will refine matches provided by the matching data files by rejecting outlier matches based on fundamental matrix.

4.1 Fundamental Matrix Estimation

Given $N \geq 8$ correspondences between two images, $x_1 \leftrightarrow x_2$, implement the function `EstimateFundamentalMatrix.m` that linearly estimates a fundamental matrix, F, such that $x_2^T F x_1 = 0$. The fundamental matrix can be estimated by solving linear least squares ($Ax = 0$). Because of noise on correspondences, the estimated fundamental matrix can be rank 3. The last singular value of the estimated fundamental matrix must be set to zero to enforce the rank 2 constraint.

4.2 Match Outlier Rejection via RANSAC

Given $N \geq 8$ correspondences between two images, $x_1 \leftrightarrow x_2$, implement the function `GetInliersRANSAC.m` that estimates inlier correspondences using fundamental matrix based RANSAC. The pseduo-code for RANSAC algorithm is given in Algorithm 2.

```
n=0;
for $i$ = $1{:}M$ do
    // Choose 8 correspondences, $\hat{x}_1$ and $\hat{x}_2$ randomly
    F = EstimateFundmentalMatrix($\hat{x}_1$, $\hat{x}_2$);
    $\mathcal{S} = \emptyset$;
    for $j$ = $1{:}N$ do
        if $| x_{2j}^T F x_{1j} | < \epsilon$ then
            $\mathcal{S} = \mathcal{S} \cup \{j\}$
        end
    end
    if $n < | \mathcal{S} |$ then
        $n = | \mathcal{S} |$;
        $\mathcal{S}_{in} = \mathcal{S}$
    end
end
```

Algorithm 2: GetInliersRANSAC Pipeline.

5 Relative Camera Pose Estimation - 10Pts

In this section, you will initialize relative camera pose between the first and second images using an essential matrix, i.e., $(0, I_{3\times3})$ and (C, R).

5.1 Essential Matrix Estimation

Given F, estimate $E = K^T F K$ by implementing the function `EssentialMatrixFromFundamentalMatrix.m`. An essential matrix can be extracted from a fundamental matrix given camera intrinsic parameter, K. Due to noise in the intrinsic parameters, the singular values of the essential matrix are not necessarily (1,1,0). The essential matrix can be corrected by reconstructing it with (1,1,0) singular values, i.e.,

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

5.2 Camera Pose Extraction

Given E, enumerate four camera pose configurations, (C_1, R_1), (C_2, R_2), (C_3, R_3), and (C_4, R_4) where $C \in \mathbb{R}^3$ is the camera center and $R \in SO(3)$ is the rotation matrix, i.e., $P = KR\left[I_{3\times 3} \quad -C\right]$. Implement the function `ExtractCameraPose.m` following the guidelines given next.
There are four camera pose configurations given an essential matrix. Let $E = UDV^T$ and $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. The four configurations are enumerated below:

1. $C_1 = U(:,3)$ and $R_1 = UWV^T$
2. $C_2 = -U(:,3)$ and $R_2 = UWV^T$
3. $C_3 = U(:,3)$ and $R_3 = UW^TV^T$
4. $C_1 = -U(:,3)$ and $R_4 = UW^TV^T$

Note that the determinant of a rotation matrix is one. If $\det(R) = -1$, the camera pose must be corrected, i.e., $C = -C$ and $R = -R$.

6 Triangulation - 15Pts

In this section, you will triangulate 3D points given two camera poses followed by nonlinear optimization. This triangulation also allows you to disambiguate four camera pose configuration obtained from the essential matrix.

6.1 Linear Triangulation

Given two camera poses, (C_1, R_1) and (C_2, R_2), and correspondences $x_1 \leftrightarrow x_2$, triangulate 3D points using linear least squares (implement the function `LinearTriangulation.m`).

6.2 Camera Pose Disambiguation

Given four camera pose configurations and their triangulated points, find the unique camera pose by checking the cheirality condition - the reconstructed points must be in front of the cameras (implement the function `DisambiguateCameraPose.m`). The sign of the Z element in the camera coordinate system indicates the location of the 3D point with respect to the camera, i.e., a 3D point X is in front of a camera if (C, R) if $r_3^T(X - C) > 0$ where r_3 is the third row of R. Not all triangulated points satisfy this condition due to the presence of correspondence noise. The best camera configuration, (C, R, X) is the one that produces the maximum number of points satisfying the cheirality condition.

6.3 Nonlinear Triangulation

Given two camera poses and linearly triangulated points, X, refine the locations of the 3D points that minimizes reprojection error (implement the function `NonlinearTriangulation.m`). The linear triangulation minimizes algebraic error. Reprojection error that is geometrically meaningful error is computed by measuring error between measurement and projected 3D point.

$$\min_{x} \sum_{j=\{1,2\}} \left(u^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}\tilde{X}}\right)^2 + \left(v^j - \frac{P_2^{jT}\tilde{X}}{P_3^{jT}\tilde{X}}\right)^2$$

Here j is the index of each camera, $\tilde{X}$ is the homogeneous representation of X. P_i^T is each row of camera projection matrix, P. This minimization is highly nonlinear because of the divisions. The initial guess of the solution, `X0`, estimated via the linear triangulation is needed to minimize the cost function. This minimization can be solved using a nonlinear optimization toolbox such as `fminunc` or `lsqnonlin` in MATLAB.

7 Perspective-n-Point - 25Pts

In this section, you will register a new image given 3D-2D correspondences, i.e., $X \leftrightarrow x$ followed by nonlinear optimization.

7.1 Linear Camera Pose Estimation

Given 2D-3D correspondences, $X \leftrightarrow x$, and the intrinsic parameter, K, estimate a camera pose using linear least squares (implement the function `LinearPnP.m`). 2D points can be normalized by the intrinsic parameter to isolate camera parameters, (C, R), i.e., $K^{-1}x$. A linear least squares system that relates the 3D and 2D points can be solved for (t, R) where $t = -R^TC$. Since the linear least square solve does not enforce orthogonality of the rotation matrix, $R \in SO(3)$, the rotation matrix must be corrected by $R = UV^T$ where $R = UDV^T$. If the corrected rotation has -1 determinant, $R = -R$. This linear PnP requires at least 6 correspondences.

7.2 PnP RANSAC

Given $N \geq 6$ 3D-2D correspondences, $X \leftrightarrow x$, implement the following function that estimates camera pose (C, R) via RANSAC (implement the function `PnPRANSAC.m`). The algorithm for this part is given in Algorithm 4.

```
$n = 0$
for $i$ = 1:M do
    // Choose 6 correspondences, $\hat{X}$ and $\hat{x}$, randomly
    [C R] = LinearPnP($\hat{X}$, $\hat{x}$, K);
    $\mathcal{S} = \emptyset$;
    for $j$ = 1:N do
        // Measure Reprojection error
        $e = \left(u - \frac{P_1^T \tilde{X}}{P_3^T \tilde{X}}\right)^2 + \left(v - \frac{P_2^T \tilde{X}}{P_3^T \tilde{X}}\right)^2$;
        if $e < \epsilon_r$ then
            $\mathcal{S} = \mathcal{S} \cup \{j\}$
        end
    end
    if $n < |\mathcal{S}|$ then
        $n =| \mathcal{S} |$;
        $\mathcal{S}_{in} = \mathcal{S}$
    end
end
```

Algorithm 4: PnPRANSAC

7.3 Nonlinear PnP

Given $N \geq 6$ 3D-2D correspondences, $X \leftrightarrow x$, and linearly estimated camera pose, (C, R), refine the camera pose that minimizes reprojection error (implement the function `NonlinearPnP.m`). The linear PnP minimizes algebraic error. Reprojection error that is geometrically meaningful error is computed by measuring error between measurement and projected 3D point

$$\min_{C,R} \sum_{i=1}^{J} \left(\left(u_j - \frac{P_1^T \tilde{X}_j}{P_3^{jT} \tilde{X}_j}\right)^2 + \left(v_j - \frac{P_2^T \tilde{X}_j}{P_3^T \tilde{X}_j}\right)^2 \right)$$

here $\tilde{X}$ is the homogeneous representation of X. P_i^T is each row of camera projection matrix, P which is computed by $P = KR\,[I_{3\times 3} \quad -C]$. A compact representation of the rotation matrix using quaternion is a better choice to enforce orthogonality of the rotation matrix, $R = R(q)$ where q is four dimensional quaternion, i.e.,

$$\min_{C,q} \sum_{i=1}^{J} \left(\left(u_j - \frac{P_1^T \tilde{X}_j}{P_3^{jT} \tilde{X}_j}\right)^2 + \left(v_j - \frac{P_2^T \tilde{X}_j}{P_3^T \tilde{X}_j}\right)^2 \right)$$

This minimization is highly nonlinear because of the divisions and quaternion parameterization. The initial guess of the solution, $(C0, R0)$, estimated via the linear PnP is needed to minimize the cost function. This minimization can be solved using a nonlinear optimization toolbox such as `fminunc` or `lsqnonlin` in MATLAB.

8 Bundle Adjustment - 20Pts

In this section, you will refine all camera poses and 3D points together initialized by previous reconstruction by minimizing reprojection error.

8.1 Visibility Matrix

Find the relationship between a camera and point, construct a $I \times J$ binary matrix, V where V_{ij} is one if the j^{th} point is visible from the i^{th} camera and zero otherwise (implement the function `BuildVisibilityMatrix.m`).

8.2 Bundle Adjustment

Given initialized camera poses and 3D points, refine them by minimizing reprojection error (implement the function `BundleAdjustment.m`). The bundle adjustment refines camera poses and 3D points simultaneously by minimizing the following reprojection error over $C_{ii=1}^{I}$, $q_{ii=1}^{I}$ and $X_{jj=1}^{J}$.

$$\min_{\{C_i,q_i\}_{i=1}^{I},\{X\}_{j=1}^{J}} \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij}\left(\left(u_{ij}-\frac{P_{i1}^{T}\tilde{X}_j}{P_{i3}^{jT}\tilde{X}_j}\right)^2+\left(v_{ij}-\frac{P_{i2}^{T}\tilde{X}_j}{P_{i3}^{T}\tilde{X}_j}\right)^2\right)$$

This minimization can be solved using a nonlinear optimization toolbox such as `fminunc` and `lsqnonlin` in MATLAB but will be extremely slow due to a number of parameters. The Sparse Bundle Adjustment toolbox (`http://users.ics.forth.gr/~lourakis/sba/`) is designed to solve such optimization by exploiting sparsity of visibility matrix, V. Note that a small number of entries in V are one because a 3D point is visible from a small subset of images. Using the sparse bundle adjustment package is not trivial but it is much faster than MATLAB built-in optimizers. If you are going to use the package, please follow the following instructions:

1. Download the package from the website and compile the mex function in the matlab folder (See README.txt).
2. Refer to `SBA_example/sba_wrapper.m` in the `data` folder that shows an example of using the package. You need to modify the code to set up camera poses and 3D points.
3. Fill `SBA_example/projection.m` function that reprojects a 3D point to an image.

9 Putting All Things Together - 15Pts

Write a program that run the full pipeline of structure from motion based on Algorithm 1. Your outputs for 6 camera frames and 1456 3D points should look something like Fig. 2.

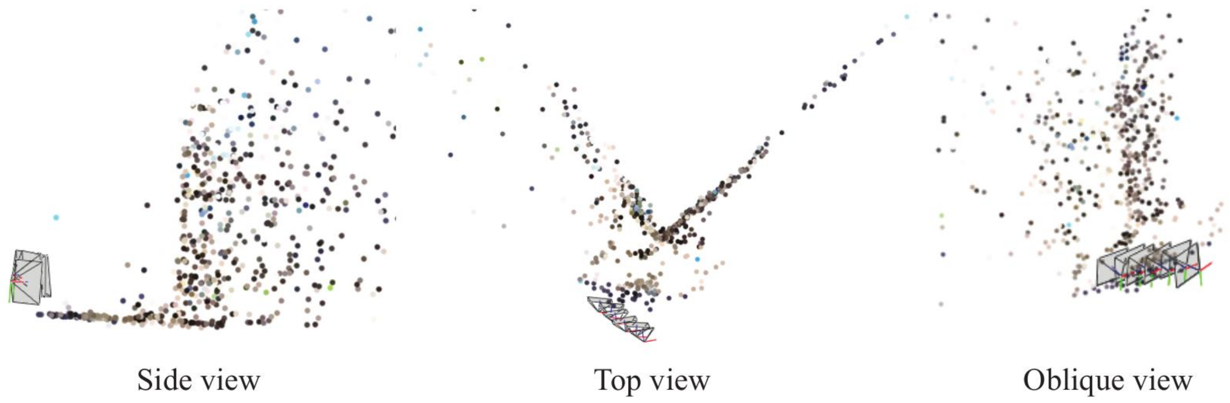

Figure 2: Input Images.

10 Compare your result against VSfM output

Compare your result with an off-the-shelf structure from motion software VisualSfM (`http://ccwu.me/vsfm/`).

11 Extra Credit

Run your SfM algorithm on the images you capture **15Pts**. Don't steal images from the internet. You need to capture images, calibrate them and undistrort them (Feel free to use MATLAB's built-in calibration tool for this).

12 Starter Code

You can find the starter code in `Code` folder. Feel free to edit any of these functions however you want. It is not mandatory to follow these function prototypes - feel free to adapt them as you desire.

13 Submission Guidelines

There will be no Test Set for this project. Please make your report extremely detailed with re-projection error after each step (Linear, Non-linear triangulation, Linear, Non-linear PnP before and after BA and so on). Describe all the steps (anything that is not obvious) and any other observations in your report. Your report should be in the IEEE double column format and should be typeset in LaTeXand **NOT** contain any code, feel free to include algorithms and math. Use the format provided in the `Draft` folder. Include outputs after each stage and talk about failure cases (if any)- we are interested in this. You should also include a detailed `README` file explaining how to run your code.

Submit your codes (`.m` files) with the naming convention `YourDirectoryID_P3.zip` onto ELMS/Canvas (**Please compress it to .zip and no other format**). If your e-mail ID is `ABCD@terpmail.umd.edu` or `ABCD@umd.edu` your Directory ID will be `ABCD`.
To summarize, you need to submit these things and in the following strcture: A zip file with the name `YourDirectoryID_P3.zip` onto ELMS/Canvas. A main folder with the name `YourDirectoryID_P3` and the following sub-folders:

- `Code` with all your code. Please include instructions on how to run your code in the `README` file.
- `Report` with your report in pdf format (typeset in LaTeXdouble-column IEEE format given to you in `Draft` folder).

If your submission does not comply with the above guidelines, you'll be given **ZERO** credit.

Also, please mention about all the extra credit you have done in your report.

Also make a presentation (need not be slides, you can use your report or the whiteboard or just talk) if you want to present it in the class **Note: Every student should present AT LEAST once in the class (you can volunteer to present for more than one project) and can choose to do for any of the Project except the last one. Good presentations will receive upto 10 bonus points.**

14 Disallowed Matlab functions

Any Matlab function which implements a part of the pipeline.

15 Project Hardness

This project is **HARD** and **TIME CONSUMING**, so please start early.

16 Collaboration Policy

You are restricted to discuss the ideas with at most one more group. But the code you turn-in should be your own and if you **DO USE** (try not it and it is not permitted) other external codes/codes from other students - do cite them. For other honor code refer to the CMSC733 Fall 2016 website.

DON'T FORGET TO HAVE FUN AND PLAY AROUND WITH IMAGES!.

Acknowledgements

This fun project was inspired from 'Machine Perception' (CIS 580) course of University of Pennsylvania (https://fling.seas.upenn.edu/~cis580/wiki/index.php?title=Machine_Perception_CIS_580_Spring_2017).