CIS 580, Machine Perception, Spring 2016
Project 1
Milestone 1 due: 2017.03.12. 11:59PM
Milestone 2 due: 2017.03.19. 11:59PM

Instructions. Submit your report and complete code to canvas. Note that this is an individual assignment.

In this project, you will estimate intrinsic parameters, i.e., focal length scale factors, $f_x$ and $f_y$, principal point, $p_x$ and $p_y$, skew factor, $s$, and radial lens distortion parameters, $k_1$ and $k_2$. You will use a checkerboard pattern to calibrate these parameters and recover camera poses with respect to the pattern.

You will complete the following *.m files in our project kit for the calibration:

- demo.m
- EstimateK_linear.m
- EstimateRt_linear.m
- EstimateDistort_linear.m
- GeoError.m
- MinGeoError.m

We have provided you the following additional functions, which you will not need to modify:

- EstimateHomography.m
- InitCalibration.m
- Evaluate.m

You can find the starter code on the course website.

For evaluation, you will be comparing your method with the one implemented in the MATLAB Calibration Toolbox (http://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html). Evaluate.m script will allow you to evaluate your estimated camera parameters. Please refer to the following paper for more detail on Matlab’s implementation:

1 Data Collection

In this section, you will capture multiple images of a planar checkerboard pattern (known 3D points) to calibrate intrinsic parameters of your cameras.

1. Print out the predefined checkerboard pattern (use open checkerboardPattern.pdf to show the pattern in MATLAB). Note that the checkerboard pattern is made of odd number of squares in one side and even number of squares on the other side to disambiguate the orientation of the pattern.

2. Measure the size of your checkerboard square in mm and fill in the size in demo.m script.
3. Capture multiple images of your checkerboard from different angles and specify the image folder path in `demo.m`. Follow the instructions below for data capture:

- Use 10-20 images for robust calibration.
- Disable auto-focus.
- Do not change the zoom settings while capturing.
- Do not modify your images, e.g., cropping, resizing, or rotating images.
- Capture the pattern from different angles with a sufficiently large baseline.
- Cover most parts of the images with the pattern.
- Do not use a camera lens with large distortion such as a GoPro camera.

**Tips:**

Note that you can also check the photographer side error by drawing "x" (corner detections). If they are far (1 pixels) from the actual corners, it’s better to recollect the calibration images by following the above instructions.

## 2 Linear Parameter Estimation

The camera intrinsic parameters are estimated linearly and refined by nonlinear optimization minimizing geometric (reprojection) error. In this section, we linearly compute camera parameters, \( \mathbf{K}, \mathbf{R}, \) and \( \mathbf{t} \) assuming no lens distortion and then, estimate lens parameters, \( k_1 \) and \( k_2 \), sequentially.

We use the projection relationship between known the 3D points on the checkerboard pattern and the corresponding 2D points in each image. The 2D points are detected by a corner detector in `InitCalibration.m` script. The script also provides the association between the 2D and 3D points. Note that we assume the 3D points are located at \( z = 0 \).

1. Complete \([\mathbf{K}, \mathbf{Hs}] = \text{EstimateK\_linear}(\mathbf{x}, \mathbf{X})\) in `EstimateK\_linear.m` that estimates a camera calibration matrix, \( \mathbf{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \).

- Output \( \mathbf{K} \) is the \( 3 \times 3 \) calibration matrix.
- Output \( \mathbf{Hs} \) is \( 3 \times 3 \times N \) homographies from the world to images, where \( N \) is the number of the calibration images.
- Input \( \mathbf{x} \) is \( 3 \times 2 \times N \) matrix.
- Input \( \mathbf{X} \) is \( 3 \times 2 \) matrix, where \( n \) is the number of corners in the checkerboard.

Define \( \mathbf{B} \) as:

\[
\mathbf{K}^{-T}\mathbf{K}^{-1} = \begin{bmatrix}
\frac{1}{f_x^2} & -\frac{s}{f_x f_y} & \frac{p_x}{f_x^2} \\
-\frac{s}{f_x f_y} & \frac{1}{f_y^2} + \frac{1}{f_x^2} & -\frac{p_y}{f_y^2} - \frac{p_x}{f_x f_y} \\
\frac{p_x s - p_y f_x}{f_x^2 f_y} & -\frac{s}{f_x f_y} (p_x s - p_y f_x) & \frac{p_x^2}{f_x^2} + \frac{p_y^2}{f_y^2} + 1
\end{bmatrix} = \begin{bmatrix}
B_{11} & B_{12} & B_{13} \\
B_{21} & B_{22} & B_{23} \\
B_{31} & B_{32} & B_{33}
\end{bmatrix} = \mathbf{B},
\]
If we had $B$, then we could compute the intrinsic parameters as follows:

$$
p_y = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2}$$

$$c = B_{33} - \frac{B_{13}^2 + p_y(B_{12}B_{13} - B_{11}B_{23})}{B_{11}}$$

$$f_y = \sqrt{\frac{cB_{11}}{B_{11}B_{22} - B_{12}^2}}$$

$$f_x = \sqrt{\frac{c}{B_{11}}}$$

$$s = -\frac{B_{12}f_x^2f_y}{c}$$

$$p_x = \frac{sp_y}{f_y} - \frac{B_{13}f_x^2}{c}.$$

To estimate $B$, we need to solve the following linear equations:

$$
\begin{bmatrix}
1 & v_{11}^T & v_{12}^T & \cdots & v_{1N}^T \\
1 & v_{12}^T & v_{22}^T & \cdots & v_{2N}^T \\
& \vdots & \vdots & \ddots & \vdots \\
1 & v_{1N}^T & v_{2N}^T & \cdots & v_{NN}^T
\end{bmatrix}
\begin{bmatrix}
B_{11} \\
B_{12} \\
B_{22} \\
B_{13} \\
B_{23} \\
B_{33}
\end{bmatrix} = 0,
\end{equation}

where $N$ is the number of calibration images and $v_{i,j}$ is defined as follows:

$$v_{i,j} = \begin{bmatrix}
h_{i,1}h_{j,1} & h_{i,1}h_{j,2} + h_{i,2}h_{j,1} & h_{i,1}h_{j,3} + h_{i,3}h_{j,1} + h_{i,1}h_{j,3} + h_{i,2}h_{j,2} + h_{i,2}h_{j,3} + h_{i,3}h_{j,2} + h_{i,3}h_{j,3}
\end{bmatrix}^T,$$

where $h_{i,k}$ is the $k$th element of $i$th column of a homography $H$. For each image, a homography $H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}$ can be estimated from 3D points and the corresponding 2D points. Use the provided EstimateHomography.m to estimate $H$.

Check the lecture note for details.

2. Complete $[Rs, ts]=\text{EstimateRt\_linear}(Hs, K)$ in EstimateRt\_linear.m that estimates the extrinsic parameters, $R$ and $t$.

- Output $Rs$ is a set of rotation matrices $(3 \times 3 \times N)$.
- Output $ts$ is a set of translation vectors $(3 \times 1 \times N)$.
- Inputs $Hs$ and $K$ are the output from EstimateK\_linear.m

Note that $N$ is the number of the calibration images.

Given the homographies and the calibration matrix, you can find rotations and translations for each image as follows:

$$r_1 = \frac{1}{z'}K^{-1}h_1$$

$$r_2 = \frac{1}{z'}K^{-1}h_2$$

$$r_3 = r_1 \times r_2$$

$$t = \frac{1}{z'}K^{-1}h_3,$$

where $H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}$ is a homography from the world to the image, and $z' = ||K^{-1}h_1||_2$. 

3
3. Complete \([ks]=\text{EstimateDistort}\_\text{linear}(x, X, K, Rs, ts)\) in \text{EstimateDistort}\_\text{linear.m} that estimates a radial distortion parameter, \(k = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}\).

- Output \(ks\) is the radial distortion parameter in a 2 \(\times\) 1 matrix.
- Inputs \(x, X, K, Rs, \) and \(ts\) are as defined above.

You can estimate \(k\) by solving the following linear equations.

\[
\begin{bmatrix}
(1u_{\text{ideal}} - p_x)^2_r \\
(1v_{\text{ideal}} - p_y)^2_r \\
\vdots \\
(mu_{\text{ideal}} - p_x)^2_r \\
(mv_{\text{ideal}} - p_y)^2_r \\
\end{bmatrix}
\begin{bmatrix}
1u_{\text{img}} - 1u_{\text{ideal}} \\
1v_{\text{img}} - 1v_{\text{ideal}} \\
\vdots \\
mu_{\text{img}} - mu_{\text{ideal}} \\
v_{\text{img}} - v_{\text{ideal}} \\
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
0 \\
\end{bmatrix}
\]

where:

- \(\begin{bmatrix} a \\ b \end{bmatrix}\) is an inhomogeneous representation of \([R \ t] [X \ 1]\) the location of a 3D world point in the camera’s coordinate system.
- \(r^2 = a^2 + b^2\)
- \(u_{\text{ideal}}\) and \(v_{\text{ideal}}\) is an inhomogeneous representation of \(K'[R \ t] [X \ 1]\), the 2D location of a world point projected into the image using \(K'\) as the camera intrinsics.
- \(K' = \begin{bmatrix} f_x & 0 & p_x \\
0 & f_y & p_y \\
0 & 0 & 1 \end{bmatrix}\)
- \(\begin{bmatrix} u_{\text{img}} \\ v_{\text{img}} \end{bmatrix}\) is the true corresponding 2D point for \(X\).
- \(m\) is the number of projected points in all images.

Check the lecture note for details.

To test that your code works, you should run \text{demo.m}, which will compute the calibration and then run the evaluation script.

### 3 Geometric Error Minimization

Given initial estimates of \(K, R, t\) and \(k\), we will minimize geometric error, defined as follows:

\[
\sum_{j=1}^{N} \sum_{i=1}^{n} ||x_{i,j} - \hat{x}_{i,j}(k, K, R, t, X_j)||^2_2,
\]

where \(x_{i,j}\) and \(\hat{x}_{i,j}\) are an inhomogeneous representation, where \(x_{i,j}\) equals the detected 2D position of \(X_j\) in the \(i\)th image, and \(\hat{x}_{i,j}\) is the estimated 2D projection of \(X_j\) into the \(i\)th image.

1. Complete \([\text{error, f}] = \text{GeoError}(x, X, ks, K, Rs, ts)\) in \text{GeoError.m} that evaluates the geometric error defined in Eq. 1.

- Output \(\text{error}\) is the geometric error.
- Output \(f\) is a vectorized form of a 2 \(\times\) \(N\) \(\times\) \(n\) matrix which has \(x_{i,j,1} - \hat{x}_{i,j,1}\) or \(x_{i,j,2} - \hat{x}_{i,j,2}\) as its elements.
- \(x\) is the 2D points in a \(n \times 2 \times N\) matrix.
• $X$ is the 3D points in a $n \times 2$ matrix, where $n$ is the number of corners in the checkerboard and $N$ is the number of the calibration images.
• $k_s$ is the radial distortion parameter in a $2 \times 1$ matrix.
• $K$ is the $3 \times 3$ calibration matrix.
• $R_s$ is a set of rotation matrices ($3 \times 3 \times N$).
• $t_s$ is a set of translation vectors ($3 \times 1 \times N$).

Note that this function will be called by a MATLAB built-in optimizer, \texttt{lsqnonlin()}.

2. Complete $[k_s_{\text{opt}}, K_{\text{opt}}, R_s_{\text{opt}}, t_s_{\text{opt}}] = \text{MinGeoError}(x, X, k_s_{\text{init}}, K_{\text{init}}, R_s_{\text{init}}, t_s_{\text{init}})$ in \texttt{MinGeoError.m} that minimizes the geometric error.

$k_s_{\text{opt}}, K_{\text{opt}}, R_s_{\text{opt}}$ and $t_s_{\text{opt}}$ are optimized parameters. $k_s_{\text{init}}, K_{\text{init}}, R_s_{\text{init}}$, and $t_s_{\text{init}}$ are initial parameters obtained from Section 2. Use the MATLAB built-in optimizer, \texttt{lsqnonlin()}. Several helper functions are provided to you to help wrangle \texttt{GeoError} into the form used by the optimizer.

4 Submission

4.1 Milestone 1

Finish Part 1, Data Collection, and also answer the following questions.

1. Think about how to solve 3.1. Write down the equation or pseudo-code you will use.

2. Based on your answers to HW1, Problem 1, what do you expect your calibration results to be?

You should package the following files into a single zip file and submit to canvas:

• Data: images you collect for the camera calibration.

• Interim Report with answers to questions.

4.2 Milestone 2

Complete the whole project, and submit the following files:

• Source code: The complete functions in our project kit and any additional scripts you might use.

• Data: Same as Milestone 1, just in case you need to update your data.

• Report: Include your calibration parameters and minimum geometric error. Also specify how to use your code to generate the result. Note that points will be deducted if we cannot regenerate your results by following the instructions in your report.
FAQs

1. How can we solve linear system, \(Ax = b\)?
   In MATLAB, \(x = A\backslash b\)

2. How can we solve linear system, \(Ax = 0\)?
   In MATLAB, \([~, ~, V] = \text{svd}(A); x = V(:, \text{end});\)

3. Can we use \(r^2 = (u_{\text{ideal}} - p_x)^2 + (v_{\text{ideal}} - p_y)^2\) to linearly estimate the radial distortion parameters instead of \(r^2 = a^2 + b^2\)?
   No. Looks reasonable, but it makes the problem ill-conditioned. \(v_{\text{ideal}} - p_y\) in the image coordinates is usually big, e.g. 500. A change in \(p_y\) of 1 turns into a change in \((v_{\text{ideal}} - p_y)^4\), which is one term in \(r^4\), of nearly \(5 \times 10^8\).

4. How can I know my code works correctly?
   In \text{demo.m}, geometric error per measurement is printed out after estimations. If the error is within 1 or 2 pixel and decreases for each step, it means your code reasonably works.
   In \text{Evaluate.m}, camera locations and orientations are visualized. You can compare your results with actual locations and orientations when you captured images.

5. How can I disable the autofocus function of my cell phone?
   Usually, there is a way to disable the autofocus function. Mine is iPhone 5s. If I press the display for a long time, it disables the autofocus function.
   Also you can sample the video that records various poses of a checkerboard. In video mode, as far as I know, focus is not changed.
   In the worst case, you can ignore the focal length changes due to the autofocus if you can collect images in roughly fixed distance. It worked reasonably in my case.