CIS 580, Machine Perception, Spring 2015
Project 1
Due: 2015.03.16. 11:59AM

Instructions. Submit your complete code to `turnin`. 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`

You can find kit from http://cis.upenn.edu/~cis580/Spring2015/Projects/proj1/kit.zip.

The estimated calibration parameters by your method will be compared with one by the MATLAB Calibration Toolbox (http://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html). `Evaluate.m` script will allow you to evaluate the estimated parameters. Please refer to the following paper for more detail:

Zhengyou Zhang, *A flexible new technique for camera calibration*, PAMI, 2000.

1 Data Collection

In this section, you will capture multiple images of a 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.

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

   `K` is the 3×3 calibration matrix and `Hs` is $3 \times 3 \times N$ homographies from the world to images, where N is the number of the calibration images. `x` is the 2D points in a $n \times 2 \times N$ matrix and `X` is the 3D points in a $n \times 2$ matrix, where n is the number of corners in the checkerboard.

   Once you estimate $\mathbf{B}$, which is defined as

$$\mathbf{K}^{-T}\mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{f_x^2} & -\frac{s}{f_x^2 f_y} & \frac{p_y s - p_x f_y}{f_x^2 f_y} \\ -\frac{s}{f_x^2 f_y} & \frac{s^2}{f_x^2 f_y^2} + \frac{1}{f_y^2} & -\frac{s(p_y s - p_x f_y)}{f_x^2 f_y^2} - \frac{p_y}{f_y^2} \\ \frac{p_y s - p_x f_y}{f_x^2 f_y} & -\frac{s(p_y s - p_x f_y)}{f_x^2 f_y^2} - \frac{p_y}{f_y^2} & \frac{(p_y s - p_x f_y)^2}{f_x^2 f_y^2} + \frac{p_y^2}{f_y^2} + 1 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \mathbf{B},$$

   the intrinsic parameters can be computed as follows:

$$\begin{aligned} p_y &= \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \\ c &= B_{33} - \frac{B_{13}^2 + p_y(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \\ f_y &= \sqrt{\frac{cB_{11}}{B_{11}B_{22} - B_{12}^2}} \\ f_x &= \sqrt{\frac{c}{B_{11}}} \\ s &= -\frac{B_{12}f_x^2 f_y}{c} \\ p_x &= \frac{sp_y}{f_y} - \frac{B_{13}f_x^2}{c}. \end{aligned}$$

   To estimate $\mathbf{B}$, we need to solve the following linear equations:

$$\begin{bmatrix} {}^1\mathbf{v}_{11}^T - {}^1\mathbf{v}_{22}^T \\ {}^1\mathbf{v}_{12}^T \\ \vdots \\ {}^I\mathbf{v}_{11}^T - {}^I\mathbf{v}_{22}^T \\ {}^I\mathbf{v}_{12}^T \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix} = \mathbf{0},$$

where I is the number of calibration images and $\mathbf{v}_{i,j}$ is defined as follows:

$$\mathbf{v}_{i,j} = \begin{bmatrix} \mathbf{h}_{i,1}\mathbf{h}_{j,1} & \mathbf{h}_{i,1}\mathbf{h}_{j,2} + \mathbf{h}_{i,2}\mathbf{h}_{j,1} & \mathbf{h}_{i,2}\mathbf{h}_{j,2} & \mathbf{h}_{i,3}\mathbf{h}_{j,1} + \mathbf{h}_{i,1}\mathbf{h}_{j,3} & \mathbf{h}_{i,3}\mathbf{h}_{j,2} + \mathbf{h}_{i,2}\mathbf{h}_{j,3} & \mathbf{h}_{i,3}\mathbf{h}_{j,3} \end{bmatrix}^T,$$

where $\mathbf{h}_{i,k}$ is the kth element of ith column of a homograpphy $\mathbf{H}$. For each image, a homography $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$ can be estimated from 3D points and the corresponding 2D points. Use `est_homography.m` to estimate $\mathbf{H}$.

Check the lecture note for details.

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

   `Rs` is a set of rotation matrices ($3 \times 3 \times N$), `ts` is a set of translation vectors ($3 \times 1 \times N$), where N is the number of the calibration images. `Hs` and `K` are the output from `EstimateK_linear.m`.

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

$$\begin{aligned} \mathbf{r}_1 &= \frac{1}{z'}\mathbf{K}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \frac{1}{z'}\mathbf{K}^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \frac{1}{z'}\mathbf{K}^{-1}\mathbf{h}_3, \end{aligned}$$

   where $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$ is a homography from the world to the image, and $z' = \|\mathbf{K}^{-1}\mathbf{h}_1\|_2$.

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

   `ks` is the radial distortion parameter in a 2×1 matrix. The definition of `x, X, K, Rs` and `ts` are defined above. You can estimate $\mathbf{k}$ by solving the following linear equations.

$$\begin{bmatrix} ({}^1u_{\text{ideal}} - p_x)r_1^2 & ({}^1u_{\text{ideal}} - p_x)r_1^4 \\ ({}^1v_{\text{ideal}} - p_y)r_1^2 & ({}^1v_{\text{ideal}} - p_y)r_1^4 \\ \vdots & \vdots \\ ({}^mu_{\text{ideal}} - p_x)r_m^2 & ({}^mu_{\text{ideal}} - p_x)r_m^4 \\ ({}^mv_{\text{ideal}} - p_y)r_m^2 & ({}^mv_{\text{ideal}} - p_y)r_m^4 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} {}^1u_{\text{img}} - {}^1u_{\text{ideal}} \\ {}^1v_{\text{img}} - {}^1v_{\text{ideal}} \\ \vdots \\ {}^mu_{\text{img}} - {}^mu_{\text{ideal}} \\ {}^mv_{\text{img}} - {}^mv_{\text{ideal}} \end{bmatrix},$$

   where $\begin{bmatrix} a \\ b \end{bmatrix}$ is an inhomogeneous representation of $\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$, $r^2 = a^2 + b^2$, $\begin{bmatrix} u_{\text{ideal}} \\ v_{\text{ideal}} \end{bmatrix}$ is an inhomogeneous representation of $\mathbf{K}' \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$, where $\mathbf{K}' = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$, $\begin{bmatrix} u_{\text{img}} \\ v_{\text{img}} \end{bmatrix}$ is the corresponding 2D point $\mathbf{x}$, and m is the number of projected points in all images.

   Check the lecture note for details.

3 Geometric Error Minimization

Given initial estimates of $\mathbf{K}, \mathbf{R}, \mathbf{t}$ and $\mathbf{k}$, we will minimize geometric error, defined as follows:

$$\sum_{i=1}^{I} \sum_{j=1}^{J} \|\mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}(\mathbf{k}, \mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)\|_2^2, \tag{1}$$

where I is the number of calibration images, J is the number of 3D points, and $\mathbf{x}_{i,j}$ and $\hat{\mathbf{x}}_{i,j}$ are an inhomogeneous representation.

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

   `error` is the geometric error and `f` is a vectorized form of a $2 \times J \times I$ matrix which has $\mathbf{x}_{i,j,1} - \hat{\mathbf{x}}_{i,j,1}$ or $\mathbf{x}_{i,j,2} - \hat{\mathbf{x}}_{i,j,2}$ as its elements. Note that this function can be called by a MATLAB built-in optimizer, `lsqnonlin()`.

2. Complete `[ks_opt, K_opt, Rs_opt, ts_opt]=MinGeoError(x, X, ks_init, K_init, Rs_init, ts_init)` in `MinGeoError.m` that minimizes the geometric error.

   `ks_opt, K_opt, Rs_opt` and `ts_opt` are optimized parameters. `ks_init, K_init, Rs_init`, and `ts_init` are initial parameters obtained from Section 2. Use the MATLAB built-in optimizer, `lsqnonlin()`.

FAQs

1. How can we solve linear system, $\mathbf{Ax} = \mathbf{b}$?

   In MATLAB, `x = A\b`

2. How can we solve linear system, $\mathbf{Ax} = \mathbf{0}$?

   In MATLAB, `[~, ~, V] = svd(A); x = V(:, end);`

3. Can we use $r^2 = (u_{img} - p_x)^2 + (v_{img} - p_y)^4$ to linearly estimate the radial distortion parameters instead of $r^2 = u_{ideal}^2 + v_{ideal}^4$?

   No. Looks reasonable, but it makes the problem ill-conditioned. $v_{img} - p_y$ in the image coordinates is usually big, e.g. 500. A change in p_y of 1 turns into a change in $(v_{img} - p_y)^4$ of nearly 5×10^8.

4. How can I know my code works correctly?

   In `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 `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.