Chapter 10
Imaging Model and Camera Calibration

Lin ZHANG
School of Computer Science and Technology
Tongji University

Outline

- What is Camera Calibration?
- Modeling for Imaging Pipeline
- General Framework for the Camera Calibration Algorithm
- Initial Rough Estimation of Calibration Parameters
- Nonlinear Least-squares
- Image Un-distortion

What is camera calibration?

- Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images
- It estimates the parameters of a lens and image sensor of the camera; you can use these parameters to correct for lens distortion, measure the size of an object in world units, or determine the location of the camera in the scene
- These tasks are used in applications such as machine vision to detect and measure objects. They are also used in robotics, for navigation systems, and 3-D scene reconstruction

What is camera calibration?

What is camera calibration?

- Camera parameters include
  - Intrinsics
  - Distortion coefficients
  - Extrinsics

To perform camera calibration, you need to know:

How to model the imaging process?

What is the general workflow for camera calibration?

How to get the initial estimation of parameters?

How to solve a nonlinear optimization problem?

Outline

- What is Camera Calibration?
- Modeling for Imaging Pipeline
- General Framework for the Camera Calibration Algorithm
- Initial Rough Estimation of Calibration Parameters
- Nonlinear Least-squares
- Image Un-distortion

Modeling for Imaging Pipeline

- The base model is the **pinhole camera model**

Modeling for Imaging Pipeline

Modeling for Imaging Pipeline

- To model the image formation process, 4 coordinate systems are required
  - World coordinate system (3D space)
  - Camera coordinate system (3D space)
  - Retinal coordinate system (2D space)
  - **Normalized retinal coordinate system (2D space)**
  - Pixel coordinate system (2D space)

Modeling for Imaging Pipeline

- From the world CS to the camera CS

$\left[x_w, y_w, z_w\right]^T$ is a 3D point represented in the WCS

In the camera CS, it is represented as,

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \mathbf{R}\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \mathbf{t} \quad \Rightarrow \quad \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \left[\mathbf{R} \;\; \mathbf{t}\right]_{3\times 4}\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (1)$$

a 3×1 translation vector

a 3×3 rotation matrix
(orthogonal, $\det(\mathbf{R})=1$)

(inhomogeneous) (normalized homogeneous)

Modeling for Imaging Pipeline

- From the camera CS to the retinal CS

We can use a pin-hole model to represent the mapping from the camera CS to the retinal CS

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f\dfrac{x_c}{z_c} \\ f\dfrac{y_c}{z_c} \end{bmatrix}$$

where f is the distance between the retinal plane and the camera optical center $\boldsymbol{O}$
The retinal plane is perpendicular to the optical axis

Note: From the view of the camera CS, the coordinates of the point (x, y) on the retinal plane are (x,y,f)

Modeling for Imaging Pipeline

- From the camera CS to the retinal CS

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f\dfrac{x_c}{z_c} \\ f\dfrac{y_c}{z_c} \end{bmatrix} \xrightarrow{\text{homogeneous form}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{z_c}\begin{bmatrix} fx_c \\ fy_c \\ z_c \end{bmatrix} = \frac{1}{z_c}\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \qquad (2)$$

normalized homogeneous

inhomogeneous

Modeling for Imaging Pipeline

- *From the camera CS to the normalized retinal CS

  Normalized retinal plane is a virtual retinal plane with a distance 1 to the optical center $\boldsymbol{O}$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \rightarrow \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \end{bmatrix} \xrightarrow{\text{homogeneous form}} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \quad (3)$$

normalized homogeneous

Lens' distortion is modeled on the normalized retinal plane and thus we will return to it shortly

Modeling for Imaging Pipeline

- From the retinal CS to the pixel CS

The unit for retinal CS $(x\text{-}y)$ is physical unit (e.g., mm, cm) while the unit for pixel CS $(u\text{-}v)$ is pixel

Suppose that one pixel represents dx physical units along the x-axis and represents dy physical units along the y-axis, and the image of the optical center is (c_x, c_y) (pixels)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & c_x \\ 0 & \frac{1}{dy} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

normalized homogeneous

Modeling for Imaging Pipeline

- From the retinal CS to the pixel CS

If the two axis, u and v, of the image plane (the CCD sensor) are not perpendicular,

$$\begin{cases} u = c_x + \dfrac{x + y\tan\alpha}{dx} \\ v = c_y + \dfrac{y}{dy} \end{cases}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{1}{dx} & \dfrac{\tan\alpha}{dx} & c_x \\ 0 & \dfrac{1}{dy} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

Modeling for Imaging Pipeline

From Eqs.1, 2, and 4, we can have

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & \frac{\tan\alpha}{dx} & c_x \\ 0 & \frac{1}{dy} & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{z_c} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} \frac{f}{dx} & \frac{f\tan\alpha}{dx} & c_x \\ 0 & \frac{f}{dy} & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{z_c} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \triangleq \begin{bmatrix} f_x & f_x\tan\alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{z_c} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

$$= \frac{1}{z_c} \underbrace{\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics matrix}} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3\times4}}_{\text{Extrinsics}} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Image formation model without considering lens distortions,

$$\mathbf{u} = \frac{1}{z_c} \cdot \mathbf{K}_{3\times3} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3\times4} \mathbf{p}_{4\times1} \qquad (5)$$

Note: $\mathbf{u}$ is the normalized homogeneous coordinates

Pay attention to the units of these symbols!

Modeling for Imaging Pipeline

- Some notes about the intrinsic matrix in practical use
  - In Matlab, the skew parameter s is explicitly modeled
  - In OpenCV, s is not modeled for both the ordinary cameras and fisheye cameras, meaning that OpenCV suppose that the two axes of the imaging plane are perfect perpendicular to each other

◆ calibrateCamera() [2/2]

```
double cv::calibrateCamera ( InputArrayOfArrays   objectPoints,
                             InputArrayOfArrays   imagePoints,
                             Size                 imageSize,
                             InputOutputArray     cameraMatrix,
                             InputOutputArray     distCoeffs,
                             OutputArrayOfArrays  rvecs,
                             OutputArrayOfArrays  tvecs,
                             OutputArray          stdDeviationsIntrinsics,
                             OutputArray          stdDeviationsExtrinsics,
                             OutputArray          perViewErrors,
                             int                  flags = 0,
                             TermCriteria         criteria = TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 30, DBL_EPSILON) )
```

Python:

```
cv.calibrateCamera(                objectPoints, imagePoints, imageSize, cameraMatrix, distCoeffs[, rvecs[, tvecs[, flags[, criteria]]]] ) -> retval, cameraMatrix, distCoeffs, rvecs, tvecs
cv.calibrateCameraExtended( objectPoints, imagePoints, imageSize, cameraMatrix, distCoeffs[, rvecs[, tvecs[, stdDeviationsIntrinsics[, stdDeviationsExtrinsics[, perViewErrors[, flags[, criteria]]]]]]] ) -> retval, cameraMatrix, distCoeffs, rvecs, tvecs, stdDeviationsIntrinsics, stdDeviationsExtrinsics, perViewErrors
```

`#include <opencv2/calib3d.hpp>`

Finds the camera intrinsic and extrinsic parameters from several views of a calibration pattern.

Parameters

objectPoints In the new interface it is a vector of vectors of calibration pattern points in the calibration pattern coordinate space (e.g. std::vector<std::vector<cv::Vec3f>>). The outer vector contains as many elements as the number of pattern views. If the same calibration pattern is shown in each view and it is fully visible, all the vectors will be the same. Although, it is possible to use partially occluded patterns or even different patterns in different views. Then, the vectors will be different. Although the points are 3D, they all lie in the calibration pattern's XY coordinate plane (thus 0 in the Z-coordinate), if the used calibration pattern is a planar rig. In the old interface all the vectors of object points from different views are concatenated together.

imagePoints In the new interface it is a vector of vectors of the projections of calibration pattern points (e.g. std::vector<std::vector<cv::Vec2f>>). imagePoints.size() and objectPoints.size(), and imagePoints[i].size() and objectPoints[i].size() for each i, must be equal, respectively. In the old interface all the vectors of object points from different views are concatenated together.

imageSize Size of the image used only to initialize the camera intrinsic matrix.

cameraMatrix Input/output 3x3 floating-point camera intrinsic matrix $A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$. If CALIB_USE_INTRINSIC_GUESS and/or CALIB_FIX_ASPECT_RATIO, CALIB_FIX_PRINCIPAL_POINT or CALIB_FIX_FOCAL_LENGTH are specified, some or all of fx, fy, cx, cy must be initialized before calling the function.

distCoeffs Input/output vector of distortion coefficients $(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6[, s_1, s_2, s_3, s_4[, \tau_x, \tau_y]]]])$ of $4, 5, 8, 12$ or 14 elements.

rvecs Output vector of rotation vectors (Rodrigues) estimated for each pattern view (e.g. std::vector<cv::Mat>>). That is, each i-th rotation vector together with the corresponding i-th translation vector (see the next output parameter description) brings the calibration pattern from the object coordinate space (in which object points are specified) to the camera coordinate space. In more technical terms, the tuple of the i-th rotation and translation vector performs a change of basis from object coordinate space to camera coordinate space. Due to its duality, this tuple is equivalent to the position of the calibration pattern with respect to the camera coordinate space.

tvecs Output vector of translation vectors estimated for each pattern view, see parameter describtion above.

Modeling for Imaging Pipeline

- Some notes about the intrinsic matrix in practical use
  - In Matlab, the skew parameter s is explicitly modeled
  - In OpenCV, s is not modeled for both the ordinary cameras and fisheye cameras, meaning that OpenCV suppose that the two axes of the imaging plane are perfect perpendicular to each other

◆ calibrate()

```
double cv::fisheye::calibrate ( InputArrayOfArrays   objectPoints,
                                InputArrayOfArrays   imagePoints,
                                const Size &         image_size,
                                InputOutputArray     K,
                                InputOutputArray     D,
                                OutputArrayOfArrays  rvecs,
                                OutputArrayOfArrays  tvecs,
                                int                  flags = 0,
                                TermCriteria         criteria = TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 100, DBL_EPSILON) )
```

Python:

cv.fisheye.calibrate(objectPoints, imagePoints, image_size, K, D[, rvecs[, tvecs[, flags[, criteria]]]]) -> retval, K, D, rvecs, tvecs

`#include <opencv2/calib3d.hpp>`

Performs camera calibration.

Parameters

objectPoints vector of vectors of calibration pattern points in the calibration pattern coordinate space.

imagePoints vector of vectors of the projections of calibration pattern points. imagePoints.size() and objectPoints.size() and imagePoints[i].size() must be equal to objectPoints[i].size() for each i.

image_size Size of the image used only to initialize the camera intrinsic matrix.

K Output 3x3 floating-point camera intrinsic matrix $A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$. If **fisheye::CALIB_USE_INTRINSIC_GUESS** is specified, some or all of fx, fy, cx, cy must be initialized before calling the function.

D Output vector of distortion coefficients (k_1, k_2, k_3, k_4).

rvecs Output vector of rotation vectors (see **Rodrigues**) estimated for each pattern view. That is, each k-th rotation vector together with the corresponding k-th translation vector (see the next output parameter description) brings the calibration pattern from the model coordinate space (in which object points are specified) to the world coordinate space, that is, a real position of the calibration pattern in the k-th pattern view (k=0.. *M* -1).

tvecs Output vector of translation vectors estimated for each pattern view.

Note: In this course, we do not consider s anymore

Modeling for Imaging Pipeline

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{z_c} [\mathbf{R} \;\; \mathbf{t}]_{3\times 4} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Point on the normalized retinal CS

$$\frac{1}{z_c} [\mathbf{R} \;\; \mathbf{t}]_{3\times 4} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}$$ (according to Eq.3)

Thus, we have a byproduct which states the relationship between the coordinates on the pixel CS and the coordinates on the normalized retinal CS,

Normalized homogeneous

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} \qquad (6)$$

Normalized homogeneous

Modeling for Imaging Pipeline

- To model the behavior of lens, we need to consider the distortion
  - Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center; the smaller the lens, the greater the distortion

Pincushion distortion
Positive radial displacement

No distortion

Barrel distortion
Negative radial displacement

Modeling for Imaging Pipeline

- To model the behavior of lens, we need to consider the distortion
  - Tangential distortion occurs when the lens and the image plane are not parallel

Modeling for Imaging Pipeline

- To model the behavior of lens, we need to consider the distortion
  - Both the two types of distortions are modeled on the **normalized retinal plane**

To model radial distortion

$$x_{dr} = x_n\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$

$$y_{dr} = y_n\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$

where $r^2 = x_n^2 + y_n^2$

k_1, k_2, k_3 are the radial distortion coefficients

To model tangential distortion

$$x_{dt} = x_n + \left(2\rho_1 x_n y_n + \rho_2\left(r^2 + 2x_n^2\right)\right)$$

$$y_{dt} = y_n + \left(2\rho_2 x_n y_n + \rho_1\left(r^2 + 2y_n^2\right)\right)$$

where $r^2 = x_n^2 + y_n^2$

ρ_1, ρ_2 are the tangential distortion coefficients

If they both need to be considered,

$$\begin{cases} x_d = x_n\left(1 + k_1 r^2 + k_2 r^4\right) + 2\rho_1 x_n y_n + \rho_2\left(r^2 + 2x_n^2\right) + x_n k_3 r^6 \\ y_d = y_n\left(1 + k_1 r^2 + k_2 r^4\right) + 2\rho_2 x_n y_n + \rho_1\left(r^2 + 2y_n^2\right) + y_n k_3 r^6 \end{cases} \quad (7)$$

Note: This step cannot be represented by matrix multiplication

Modeling for Imaging Pipeline

- To model the behavior of lens, we need to consider the distortion
  - Both the two types of distortions are modeled on the **normalized retinal plane**
  - If the FOV is extremely large (larger than 100 degrees), i.e. the camera is a fisheye camera, we need to use another model to characterize lens distortions

A typical image collected by a fisheye camera

Modeling for Imaging Pipeline

- To model the behavior of lens, we need to consider the distortion
  - Both the two types of distortions are modeled on the **normalized retinal plane**
  - If the FOV is extremely large (larger than 100 degrees), i.e. the camera is a fisheye camera, we need to use another model to characterize lens distortions

To model the fisheye distortion

$$\theta = \arctan(r)$$

$$r_d = \theta\left(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8\right)$$

$$x_d = \frac{r_d}{r} x_n$$

$$y_d = \frac{r_d}{r} y_n$$

where $r^2 = x_n^2 + y_n^2$

k_1, k_2, k_3, k_4 are the distortion coefficients

Modeling for Imaging Pipeline

By considering lens distortion, we get the complete imaging pipeline,

point on the normalized retinal plane

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathcal{D}\left\{ \frac{1}{z_c} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3\times 4} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \right\} \qquad (8)$$

distorted point on the normalized retinal plane

$f_x, f_y, c_x, c_y, k_1, k_2, \rho_1, \rho_2, k_3$ are the intrinsics of the camera (suppose it is an ordinary camera)

$\mathbf{R}$ (three DOFs) and $\mathbf{t}$ (three DOFs) are the extrinsics of the camera

Modeling for Imaging Pipeline

- The process to get the intrinsics and extrinsics of the camera is called camera calibration
  - For most cases of single camera calibration, only the intrinsics are what we really need
- To model radial and tangential distortions, we use 5 parameters; Actually, more complicated models can be used, but the modeling pipeline is the same
  - E.g. the thin prism model, the tilted model used in OpenCV

Outline

- What is Camera Calibration?
- Modeling for Imaging Pipeline
- General Framework for the Camera Calibration Algorithm
- Initial Rough Estimation of Calibration Parameters
- Nonlinear Least-squares
- Image Un-distortion

General Framework for the Camera Calibration Algorithm

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathcal{D}\left\{ \frac{1}{z_c} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3\times 4} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \right\}$$

(Eq. 8, the imaging pipeline)

- General idea
  - If we have a set of known points $\{\mathbf{p}_i\}_{i=1}^{n}$ in the WCS and their images $\{\mathbf{u}_i\}_{i=1}^{n}$, using Eq. 8, we could have $2n$ equations
  - If the number of valid constraints (equations) are enough, Eq. 8 could be solved
- All the calibration algorithms follow the above general rules and among them, Zhengyou Zhang's idea[1] is the most widely used

[1] Z. Zhang, A flexible new technique for camera calibration, IEEE Trans. Pattern Analysis and Machine Intelligence, 2000

General Framework for the Camera Calibration Algorithm

- Zhengyou Zhang's calibration approach
  - A calibration board with a chessboard pattern is needed
  - Several images of the board need to be captured
  - Detect the feature points (cross points) in the images
  - Based on the correspondence pairs (pixel coordinate and world coordinate of a feature point), equation systems can be obtained
  - By solving the equation systems, parameters can be determined

Aug. 1, 1965~, now is the director of Tencent AI Lab

General Framework for the Camera Calibration Algorithm

- Zhengyou Zhang's calibration approach

Calibration board

The number of blocks of one side should be even and the number of blocks of the other side should be odd

General Framework for the Camera Calibration Algorithm

- Zhengyou Zhang's calibration approach

General Framework for the Camera Calibration Algorithm

- Zhengyou Zhang's calibration approach

A set of Calibration board images (20~30)

General Framework for the Camera Calibration Algorithm

Suppose we have M board images and for each image we have N cross points, then the calibration amounts to the following optimization problem,

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{1}{2} \left\| \mathbf{K} \cdot \mathcal{D} \left\{ \frac{1}{z_{cij}} \left[\mathbf{R}_i \ \mathbf{t}_i \right] \mathbf{p}_j \right\} - \mathbf{u}_{ij} \right\|_2^2 \qquad (9)$$

where $\Theta = \left\{ f_x, f_y, c_x, c_y, k_1, k_2, \rho_1, \rho_2, k_3, \{\mathbf{R}_i\}_{i=1}^{M}, \{\mathbf{t}_i\}_{i=1}^{M} \right\}$ denotes the parameters that needs to be optimized

$\mathbf{p}_j = (x_j, y_j, 0, 1)^{\mathrm{T}}$ is the WCS coordinates (determined by the physical calibration board) of the jth cross-point, and $\mathbf{u}_{ij}$ is its projection (pixel coordinate) on ith image

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$ denotes the intrinsics matrix

General Framework for the Camera Calibration Algorithm

- About the rotation
  - In 3D Euclidean space, a rotation has 3 DOFs (three Euler angles)
  - If we use a 3×3 matrix to denote the rotation, we must add extra constraints (the matrix should be orthonormal and its determinant should be 1) and that will make the optimization complicated
  - Thus, in all modern implementations, a rotation is finally represented by an **axis-angle (or rotation vector)**

$\mathbf{n} \in \mathbb{R}^{3\times1}, \|\mathbf{n}\|_2 = 1$ is a unit 3D vector describing an axis of rotation according to the right hand rule; $\theta \geq 0$ is the rotation angle $\mathbf{d}=\mathbf{n}\theta$, a 3D vector denoting the rotation is called axis-angle

For a vector $\mathbf{d} \in \mathbb{R}^{3\times1}$, the rotation represented by it is: the rotation axis is $\dfrac{\mathbf{d}}{\|\mathbf{d}\|_2}$ and the rotation angle is $\|\mathbf{d}\|_2$

General Framework for the Camera Calibration Algorithm

- About the rotation
  - An axis-angle vector can be uniquely converted to a rotation matrix and vice versa via **Rodrigues** formula

From axis-angle $\mathbf{d}=\mathbf{n}\theta$ to rotation matrix $\mathbf{R}$

$$\mathbf{R} = \cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta\mathbf{n}^\wedge \quad (10)$$

where $\mathbf{I}$ is the identity matrix and

$$\mathbf{n}^\wedge = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

From rotation matrix $\mathbf{R}$ to axis-angle $\mathbf{d}=\mathbf{n}\theta$

$$\theta = \arccos\left(\frac{tr(\mathbf{R})-1}{2}\right) \in [0,\pi]$$

If $\mathbf{n}$ is the axis of the rotation $\mathbf{R}$, applying $\mathbf{R}$ to $\mathbf{n}$ will not change $\mathbf{n}$. Thus, $\mathbf{n}$ is the solution of

$\mathbf{R}\boldsymbol{\alpha} = \boldsymbol{\alpha}$ ($\boldsymbol{\alpha}$ is the unit eigen-vector of $\mathbf{R}$ associated with the eigen-value 1)

If $\boldsymbol{\alpha}^*$ is a solution, $-\boldsymbol{\alpha}^*$ should be another solution. You need to check whether $\theta\boldsymbol{\alpha}^*$ or $-\theta\boldsymbol{\alpha}^*$ can be transformed back to $\mathbf{R}$ to determine whether $\mathbf{n}$ is $\boldsymbol{\alpha}^*$ or $-\boldsymbol{\alpha}^*$

General Framework for the Camera Calibration Algorithm

Suppose we have M board images and for each image we have N cross points, then the calibration amounts to the following optimization problem,

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{1}{2} \left\| \mathbf{K} \cdot \mathcal{D} \left\{ \frac{1}{z_{cij}} \left[\mathcal{R}\left(\mathbf{d}_i\right) \mathbf{t}_i \right] \mathbf{p}_j \right\} - \mathbf{u}_{ij} \right\|_2^2 \quad (9)$$

where $\mathcal{R}\left(\mathbf{d}_i\right) = \mathbf{R}_i$ and the parameters that need to be optimized are,

$$\Theta = \left\{ f_x, f_y, c_x, c_y, k_1, k_2, \rho_1, \rho_2, k_3, \left\{\mathbf{d}_i^T\right\}_{i=1}^{M}, \left\{\mathbf{t}_i^T\right\}_{i=1}^{M} \right\}^T$$ ($\mathbf{d}_i$ is the axis-angle representation of $\mathbf{R}_i$)

Altogether, we have $2 \times M \times N$ error terms and $9 + 6M$ unknown parameters

Eq. 9 is a nonlinear optimization problem and does not have a closed-form solution. It can be solved by iterative methods. But before that, we need to have a good starting point, i.e. we need to have a rough estimate to Θ

Outline

- What is Camera Calibration?
- Modeling for Imaging Pipeline
- General Framework for the Camera Calibration Algorithm
- Initial Rough Estimation of Calibration Parameters
- Nonlinear Least-squares
- Image Un-distortion

Initial Rough Estimation of Calibration Parameters

- The task at this step
  - Given us a set of M images of the planar calibration board, **roughly** estimate the intrinsics of the camera and the extrinsics of the camera poses when taking these images

Initial Rough Estimation of Calibration Parameters

- The task at this step
  - Given us a set of M images of the planar calibration board, **roughly** estimate the intrinsics of the camera and the extrinsics of the camera poses when taking these images

$$\Theta = \left\{ f_x, f_y, c_x, c_y, \underbrace{k_1, k_2, \rho_1, \rho_2, k_3}_{\text{Distortion coefficients can be safely initialized as zeros}}, \left\{ \mathbf{d}_i^T \right\}_{i=1}^{M}, \left\{ \mathbf{t}_i^T \right\}_{i=1}^{M} \right\}^T$$

Thus, in initial estimation of other parameters, we use the imaging model without considering distortions,

$$\mathbf{u} = \frac{1}{z_c} \cdot \mathbf{K}_{3\times3} \left[\mathbf{R} \;\; \mathbf{t} \right]_{3\times4} \mathbf{p}_{4\times1} \qquad \text{(Eq. 5)}$$

Given a calibration board, $\mathbf{p}$ is a cross-point on it, thus $\mathbf{p}$ has the form $\mathbf{p} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$

Initial Rough Estimation of Calibration Parameters

- The task at this step
  - Given us a set of M images of the planar calibration board, **roughly** estimate the intrinsics of the camera and the extrinsics of the camera poses when taking these images

$$\Theta = \left\{ f_x, f_y, \underbrace{c_x, c_y}, k_1, k_2, \rho_1, \rho_2, k_3, \left\{ \mathbf{d}_i^T \right\}_{i=1}^{M}, \left\{ \mathbf{t}_i^T \right\}_{i=1}^{M} \right\}^T$$

The principal point coordinates can be reasonably initialized as half of the image resolution,

$$c_x = \frac{\text{width}}{2}, c_y = \frac{\text{height}}{2}$$

where width and height are the image width and height taken by the camera

The estimation of f_x and f_y is a little bit complicated

Initial Rough Estimation of Calibration Parameters

Result 1: In the camera coordinate system, the direction of the ray pointing from the optical center $\boldsymbol{O}$ to the pixel $\mathbf{u}$ (in homogeneous form) on the imaging plane is

$$\mathbf{d} = \mathbf{K}^{-1}\mathbf{u}$$

where $\mathbf{K}$ is the intrinsics matrix of the camera

Let $\mathbf{u}'$ be the **normalized homogeneous** form of $\mathbf{u}$

According to Eq. 6, we have

$$\mathbf{K}^{-1}\mathbf{u}' = \mathbf{x}_n$$

Since $\boldsymbol{O}$, $\mathbf{x}_n$ and $\mathbf{u}$ are colinear,

$$\mathbf{d} = \overrightarrow{\boldsymbol{O}\mathbf{u}} = \overrightarrow{\boldsymbol{O}\mathbf{x}_n} = \mathbf{x}_n - \mathbf{0} = \mathbf{x}_n = \mathbf{K}^{-1}\mathbf{u}'$$

Actually, any $c\mathbf{K}^{-1}\mathbf{u}' = \mathbf{K}^{-1}(c\mathbf{u}')\ (c \neq 0)$ can represent the direction of $\mathbf{d}$

$c\mathbf{u}'\ (c \neq 0)$ is the general homogeneous form of $\mathbf{u}'$ ➡ $c\mathbf{u}' = \mathbf{u}$

⬇

$$\mathbf{d} = \mathbf{K}^{-1}\mathbf{u}$$

Initial Rough Estimation of Calibration Parameters

Result 2: In the camera coordinate system, the angle between two rays, pointing from $\boldsymbol{O}$ to $\mathbf{x}_1$ and $\mathbf{x}_2$ ($\mathbf{x}_1$ and $\mathbf{x}_2$ are the homogeneous coordinates of two pixels on the imaging plane), respectively, is determined as,

$$\cos\theta = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{\|\mathbf{d}_1\|\|\mathbf{d}_2\|}$$

$$= \frac{\left(\mathbf{K}^{-1}\mathbf{x}_1\right)^T \mathbf{K}^{-1}\mathbf{x}_2}{\sqrt{\left(\mathbf{K}^{-1}\mathbf{x}_1\right)^T\left(\mathbf{K}^{-1}\mathbf{x}_1\right)}\sqrt{\left(\mathbf{K}^{-1}\mathbf{x}_2\right)^T\left(\mathbf{K}^{-1}\mathbf{x}_2\right)}}$$

$$= \frac{\mathbf{x}_1^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{x}_2}{\sqrt{\mathbf{x}_1^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{x}_1}\sqrt{\mathbf{x}_2^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{x}_2}}$$

$$\Rightarrow \theta = \arccos\frac{\mathbf{x}_1^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{x}_2}{\sqrt{\mathbf{x}_1^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{x}_1}\sqrt{\mathbf{x}_2^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{x}_2}}$$

Initial Rough Estimation of Calibration Parameters

Definition 1: Vanishing point

With the pinhole camera model, the image of the infinity point of a line on the physical plane is called a **vanishing point**

The points $\boldsymbol{x}_i$, $i = 1, \ldots, 4, \ldots$ are equally spaced on the world line, but their spacing on the image line monotonically decreases. In the limit $\boldsymbol{x} \to \infty$, the world point is imaged at $\boldsymbol{v}$ on the imaging plane

another definition of the vanishing point

Initial Rough Estimation of Calibration Parameters

Definition 1: Vanishing point

With the pinhole camera model, the image of the infinity point of a line on the physical plane is called a **vanishing point**

Definition 2: Vanishing point (another definition)

The vanishing point of a world line l is obtained by intersecting the image plane with a ray parallel to l through the camera centre $\boldsymbol{O}$

Initial Rough Estimation of Calibration Parameters

Two parallel world lines should have the same infinity point; For each line, its vanishing point is the image of its infinity point; so the images of two parallel world lines would converge to the same vanishing point

Initial Rough Estimation of Calibration Parameters

Result 3: Denote $\boldsymbol{O}$ the camera center and $\mathbf{v}$ is one point on the imaging plane. The ray $\boldsymbol{O}\mathbf{v}$ is parallel to the world lines who share the same vanishing point $\mathbf{v}$

- Properties of vanishing points
  - A feature of perspective projection is that the image of an object that stretches off to infinity can have finite extent. E.g., an infinite scene line is imaged as a line terminating in a vanishing point
  - Parallel world lines, such as railway lines, are imaged as converging lines and their image intersection is the **vanishing point for the direction of the railway**

Initial Rough Estimation of Calibration Parameters

Initial Rough Estimation of Calibration Parameters

JingHu High-speed railway: rails will “meet” at the vanishing point

Initial Rough Estimation of Calibration Parameters

Result 4: l_1 and l_2 are two world lines on the same plane and $\mathbf{v}_1$ and $\mathbf{v}_2$ are their vanishing points on the imaging plane, respectively. $\boldsymbol{O}$ is the optical center. Let $\theta = \widehat{\overrightarrow{\boldsymbol{O}\mathbf{v}_1}\,\overrightarrow{\boldsymbol{O}\mathbf{v}_2}}$

$$\widehat{l_1 l_2} = \theta \text{, or } \widehat{l_1 l_2} = \pi - \theta$$

Proof:

$\mathbf{v}_1$ is l_1's vanishing point $\qquad$ $\mathbf{v}_2$ is l_2's vanishing point

based on Result 3

$$\left.\begin{array}{l} l_1 \parallel \overrightarrow{\boldsymbol{O}\boldsymbol{v}_1}, \quad l_2 \parallel \overrightarrow{\boldsymbol{O}\boldsymbol{v}_2} \\ \widehat{\overrightarrow{\boldsymbol{O}\mathbf{v}_1}\,\overrightarrow{\boldsymbol{O}\mathbf{v}_2}} = \theta \end{array}\right\} \Rightarrow \widehat{l_1 l_2} = \theta \text{, or } \widehat{l_1 l_2} = \pi - \theta$$

Initial Rough Estimation of Calibration Parameters

Result 5: l_1 and l_2 are two world lines on the same plane perpendicular to each other, and $\mathbf{v}_1$ and $\mathbf{v}_2$ are their vanishing points on the imaging plane, respectively. We have,

$$\mathbf{v}_1^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{v}_2 = 0$$

Proof:

$\mathbf{v}_1$ is l_1's vanishing point $\qquad$ $\mathbf{v}_2$ is l_2's vanishing point

based on Result 3

$$\left.\begin{array}{l} \boxed{l_1 \parallel \overrightarrow{\boldsymbol{Ov}_1}\,,\quad l_2 \parallel \overrightarrow{\boldsymbol{Ov}_2}} \\ l_1 \perp l_2 \end{array}\right\} \Rightarrow \overrightarrow{\boldsymbol{Ov}_1} \perp \overrightarrow{\boldsymbol{Ov}_2} \xrightarrow{\text{based on Result 2}} \mathbf{v}_1^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{v}_2 = 0$$

Note: This is a key result based on which camera calibration schemes roughly estimate camera's intrinsics f_x and f_y

Initial Rough Estimation of Calibration Parameters

On the projective plane defined by the calibration boards, consider the four lines,

l_1: X-axis, its infinity point is $\mathbf{p}_{\infty 1} = (1,0,0)^T$

l_2: Y-axis, its infinity point is $\mathbf{p}_{\infty 2} = (0,1,0)^T$

l_3: $Y=X$, its infinity point is $\mathbf{p}_{\infty 3} = (1,1,0)^T$

l_4: $Y=-X$, its infinity point is $\mathbf{p}_{\infty 4} = (1,-1,0)^T$

It can be verified: $l_1 \perp l_2$, $l_3 \perp l_4$

Initial Rough Estimation of Calibration Parameters

The plane of the calibration board and the image plane $\mathbf{I}_i$ is linked via a homography $\mathbf{H}_i$

$$c\mathbf{u}_{ij(3\times1)} = \mathbf{H}_{i(3\times3)}\mathbf{p}_{j(3\times1)}$$

projection of $\mathbf{p}_j$ on $\mathbf{I}_i$

homogeneous planar point on the calibration board

$\mathbf{H}_i$ can be estimated in advance using the techniques introduced in Theme 1

Denote $\mathbf{H}_i$ by,

$$\mathbf{H}_i = \left[\mathbf{h}_{i1}\ \mathbf{h}_{i2}\ \mathbf{h}_{i3}\right]$$

Images of $\mathbf{p}_{\infty1}$, $\mathbf{p}_{\infty2}$, $\mathbf{p}_{\infty3}$, and $\mathbf{p}_{\infty4}$ on $\mathbf{I}_i$ are,

$$\mathbf{v}_{i1} = \left[\mathbf{h}_{i1}\ \mathbf{h}_{i2}\ \mathbf{h}_{i3}\right]\begin{bmatrix}1\\0\\0\end{bmatrix} = \mathbf{h}_{i1}, \mathbf{v}_{i2} = \left[\mathbf{h}_{i1}\ \mathbf{h}_{i2}\ \mathbf{h}_{i3}\right]\begin{bmatrix}0\\1\\0\end{bmatrix} = \mathbf{h}_{i2}$$

$$\mathbf{v}_{i3} = \left[\mathbf{h}_{i1}\ \mathbf{h}_{i2}\ \mathbf{h}_{i3}\right]\begin{bmatrix}1\\1\\0\end{bmatrix} = \mathbf{h}_{i1} + \mathbf{h}_{i2}$$

$$\mathbf{v}_{i4} = \left[\mathbf{h}_{i1}\ \mathbf{h}_{i2}\ \mathbf{h}_{i3}\right]\begin{bmatrix}1\\-1\\0\end{bmatrix} = \mathbf{h}_{i1} - \mathbf{h}_{i2}$$

$\mathbf{v}_{i1}$, $\mathbf{v}_{i2}$, $\mathbf{v}_{i3}$ and $\mathbf{v}_{i4}$ are actually the vanishing points of l_1~ l_4 on $\mathbf{I}_i$

Initial Rough Estimation of Calibration Parameters

Based on Result 5, we have $\begin{cases} \mathbf{v}_{i1}^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{v}_{i2}=0 \\ \mathbf{v}_{i3}^T\left(\mathbf{K}^{-T}\mathbf{K}^{-1}\right)\mathbf{v}_{i4}=0 \end{cases}$ (11)

$$\mathbf{K}=\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}=\underbrace{\begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{P}}\underbrace{\begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Q}} \Rightarrow \mathbf{K}=\mathbf{PQ} \Rightarrow \mathbf{K}^{-T}\mathbf{K}^{-1}=\left(\mathbf{PQ}\right)^{-T}\left(\mathbf{PQ}\right)^{-1}=\mathbf{P}^{-T}\left(\mathbf{Q}^{-T}\mathbf{Q}^{-1}\right)\mathbf{P}^{-1}$$

$$\begin{cases} \left(\mathbf{P}^{-1}\mathbf{v}_{i1}\right)^T\left(\mathbf{Q}^{-T}\mathbf{Q}^{-1}\right)\mathbf{P}^{-1}\mathbf{v}_{i2}=0 \\ \left(\mathbf{P}^{-1}\mathbf{v}_{i3}\right)^T\left(\mathbf{Q}^{-T}\mathbf{Q}^{-1}\right)\mathbf{P}^{-1}\mathbf{v}_{i4}=0 \end{cases} \quad (12)$$

$$\mathbf{P}^{-1}\mathbf{v}_{i1}\triangleq\begin{pmatrix} a_{i1} \\ b_{i1} \\ c_{i1} \end{pmatrix},\mathbf{P}^{-1}\mathbf{v}_{i2}\triangleq\begin{pmatrix} a_{i2} \\ b_{i2} \\ c_{i2} \end{pmatrix},\mathbf{P}^{-1}\mathbf{v}_{i3}\triangleq\begin{pmatrix} a_{i3} \\ b_{i3} \\ c_{i3} \end{pmatrix},\mathbf{P}^{-1}\mathbf{v}_{i4}\triangleq\begin{pmatrix} a_{i4} \\ b_{i4} \\ c_{i4} \end{pmatrix} \qquad \mathbf{Q}^{-T}\mathbf{Q}^{-1}=\begin{bmatrix} \frac{1}{f_x^2} & 0 & 0 \\ 0 & \frac{1}{f_y^2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Eq. 12 becomes ➡

Initial Rough Estimation of Calibration Parameters

$$\begin{cases} \dfrac{a_{i1}a_{i2}}{f_x^2} + \dfrac{b_{i1}b_{i2}}{f_y^2} = -c_{i1}c_{i2} \\ \dfrac{a_{i3}a_{i4}}{f_x^2} + \dfrac{b_{i3}b_{i4}}{f_y^2} = -c_{i3}c_{i4} \end{cases} \Rightarrow \begin{bmatrix} a_{i1}a_{i2} & b_{i1}b_{i2} \\ a_{i3}a_{i4} & b_{i3}b_{i4} \end{bmatrix} \begin{bmatrix} \dfrac{1}{f_x^2} \\ \dfrac{1}{f_y^2} \end{bmatrix} = \begin{bmatrix} -c_{i1}c_{i2} \\ -c_{i3}c_{i4} \end{bmatrix}$$

If we have M calibration board images, we can finally have $2M$ such equations,

$$\begin{bmatrix} a_{11}a_{12} & b_{11}b_{12} \\ a_{13}a_{14} & b_{13}b_{14} \\ a_{21}a_{22} & b_{21}b_{22} \\ a_{23}a_{24} & b_{23}b_{24} \\ \vdots & \vdots \\ a_{M1}a_{M2} & b_{M1}b_{M2} \\ a_{M3}a_{M4} & b_{M3}b_{M4} \end{bmatrix}_{2M\times 2} \begin{bmatrix} \dfrac{1}{f_x^2} \\ \dfrac{1}{f_y^2} \end{bmatrix} = \begin{bmatrix} -c_{11}c_{12} \\ -c_{13}c_{14} \\ -c_{21}c_{22} \\ -c_{23}c_{24} \\ \vdots \\ -c_{M1}c_{M2} \\ -c_{M3}c_{M4} \end{bmatrix}_{2M\times 1}$$

We can solve $\left[\dfrac{1}{f_x^2}, \dfrac{1}{f_y^2}\right]$ using the least squares technique, and at last f_x and f_y are obtained

Initial Rough Estimation of Calibration Parameters

- Initial estimation of extrinsics $\mathbf{d}$ and $\mathbf{t}$ when taking image $\mathbf{I}$

The calibration board plane and $\mathbf{I}$'s **normalized retinal plane** is linked via a homography $\mathbf{H}$ ($\mathbf{H}$ can be estimated in advance)

$$c_j \begin{bmatrix} x_{nj} \\ y_{nj} \\ 1 \end{bmatrix} = \mathbf{H}_{3\times3} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix}$$

The planar point on the calibration board

On the other hand, based on the imaging model,

$$z_{cj} \begin{bmatrix} x_{nj} \\ y_{nj} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ z_j \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix}$$

$\mathbf{H}$ and $\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$ map $(x_j, y_j, 1)^T$ to the same point on the normalized retinal plane

$\mathbf{H}$ and $\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$ actually represent the same homography

$$\begin{bmatrix} \mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3 \end{bmatrix} = \mathbf{H} = \lambda \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

Initial Rough Estimation of Calibration Parameters

- Initial estimation of extrinsics $\mathbf{d}$ and $\mathbf{t}$ when taking image $\mathbf{I}$

$$\lambda\mathbf{r}_1 = \mathbf{h}_1, \lambda\mathbf{r}_2 = \mathbf{h}_2, \lambda\mathbf{t} = \mathbf{h}_3$$

$$\Downarrow$$

$$\mathbf{r}_1 = \frac{1}{\lambda}\mathbf{h}_1, \mathbf{r}_2 = \frac{1}{\lambda}\mathbf{h}_2, \mathbf{t} = \frac{1}{\lambda}\mathbf{h}_3$$

Since $\|\mathbf{r}_1\| = \|\mathbf{r}_2\| = 1 \Rightarrow |\lambda| = \|\mathbf{h}_1\| = \|\mathbf{h}_2\|$

Note: In OpenCV, λ is estimated as $\lambda = \frac{1}{2}\left(\|\mathbf{h}_1\| + \|\mathbf{h}_2\|\right)$

Since $\mathbf{r}_3 \perp \mathbf{r}_1, \mathbf{r}_3 \perp \mathbf{r}_2, \|\mathbf{r}_3\| = 1, \det\left([\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]\right) = 1 \Rightarrow \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$

Then, $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, and $\mathbf{t}$ are all initialized

Finally, $\mathbf{R}=[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]$ is converted to its axis-angle representation $\mathbf{d} \in \mathbb{R}^{3\times1}$

Note: Initial estimation of extrinsics needs to be performed to every calibration board image

Outline

- What is Camera Calibration?
- Modeling for Imaging Pipeline
- General Framework for the Camera Calibration Algorithm
- Initial Rough Estimation of Calibration Parameters
- Nonlinear Least-squares
- Image Un-distortion

Nonlinear Least-squares

- For nonlinear least-square solutions, please refer to Chapter 9

The camera calibration problem is to solve,

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{1}{2}\left\| \mathbf{K}\cdot\mathcal{D}\left\{\frac{1}{z_{cij}}\left[\mathcal{R}(\mathbf{d}_i)\,\mathbf{t}_i\right]\mathbf{p}_j\right\} - \mathbf{u}_{ij}\right\|_2^2 \qquad (9)$$

$$\mathbf{f}_{ij}(\Theta) = \mathbf{K}\cdot\mathcal{D}\left\{\frac{1}{z_{cij}}\left[\mathcal{R}(\mathbf{d}_i)\,\mathbf{t}_i\right]\mathbf{p}_j\right\} - \mathbf{u}_{ij}$$

$$\Theta^* = \arg\min_{\Theta} \frac{1}{2}\mathbf{f}^T(\Theta)\mathbf{f}(\Theta)$$

$$\mathbf{f}(\Theta) = \begin{bmatrix} \mathbf{f}_{11}(\Theta) \\ \mathbf{f}_{12}(\Theta) \\ \vdots \\ \mathbf{f}_{1N}(\Theta) \\ \mathbf{f}_{21}(\Theta) \\ \vdots \\ \mathbf{f}_{MN}(\Theta) \end{bmatrix}_{2MN\times 1}$$

The update step in L-M method,

$$\mathbf{h}_{\mathrm{lm}} = -\left[\left(\mathbf{J}^T(\Theta_k)\mathbf{J}(\Theta_k)\right) + \mu\mathbf{I}\right]^{-1}\mathbf{J}^T(\Theta_k)\mathbf{f}(\Theta_k)$$

The core problem is to derive $\mathbf{J}(\Theta)$

Nonlinear Least-squares

$$\mathbf{f}_{ij}(\Theta) = \overbrace{\mathbf{K}\cdot\mathcal{D}\left\{\frac{1}{z_{cij}}\left[\mathcal{R}(\mathbf{d}_i)\,\mathbf{t}_i\right]\mathbf{p}_j\right\}}^{\hat{\mathbf{u}}_{ij}} - \mathbf{u}_{ij}$$

$$\Theta = \left\{ f_x, f_y, c_x, c_y, \underbrace{k_1, k_2, \rho_1, \rho_2, k_3}_{\mathbf{k}^T}, \left\{\mathbf{d}_i^T\right\}_{i=1}^{M}, \left\{\mathbf{t}_i^T\right\}_{i=1}^{M} \right\}^T$$

$$\mathbf{J}(\Theta) = \begin{bmatrix} \frac{d\mathbf{f}_{11}(\Theta)}{d\Theta^T} \\ \frac{d\mathbf{f}_{12}(\Theta)}{d\Theta^T} \\ \vdots \\ \frac{d\mathbf{f}_{1N}(\Theta)}{d\Theta^T} \\ \frac{d\mathbf{f}_{21}(\Theta)}{d\Theta^T} \\ \vdots \\ \frac{d\mathbf{f}_{MN}(\Theta)}{d\Theta^T} \end{bmatrix}_{2MN\times(9+6M)} = \begin{bmatrix} \frac{d\hat{\mathbf{u}}_{11}}{d\Theta^T} \\ \frac{d\hat{\mathbf{u}}_{12}}{d\Theta^T} \\ \vdots \\ \frac{d\hat{\mathbf{u}}_{1N}}{d\Theta^T} \\ \frac{d\hat{\mathbf{u}}_{21}}{d\Theta^T} \\ \vdots \\ \frac{d\hat{\mathbf{u}}_{MN}}{d\Theta^T} \end{bmatrix} = \begin{bmatrix} \frac{d\hat{\mathbf{u}}_{11}}{df_x} & \frac{d\hat{\mathbf{u}}_{11}}{df_y} & \frac{d\hat{\mathbf{u}}_{11}}{dc_x} & \frac{d\hat{\mathbf{u}}_{11}}{dc_y} & \frac{d\hat{\mathbf{u}}_{11}}{d\mathbf{k}^T} & \frac{d\hat{\mathbf{u}}_{11}}{d\mathbf{d}_1^T} & (\mathbf{0}\cdots\cdots\mathbf{0})_{2\times(3\times(M-1))} & \frac{d\hat{\mathbf{u}}_{11}}{d\mathbf{t}_1^T} & (\mathbf{0}\cdots\mathbf{0})_{2\times(3\times(M-1))} \\ \frac{d\hat{\mathbf{u}}_{12}}{df_x} & \frac{d\hat{\mathbf{u}}_{12}}{df_y} & \frac{d\hat{\mathbf{u}}_{12}}{dc_x} & \frac{d\hat{\mathbf{u}}_{12}}{dc_y} & \frac{d\hat{\mathbf{u}}_{12}}{d\mathbf{k}^T} & \frac{d\hat{\mathbf{u}}_{12}}{d\mathbf{d}_1^T} & (\mathbf{0}\cdots\cdots\mathbf{0})_{2\times(3\times(M-1))} & \frac{d\hat{\mathbf{u}}_{12}}{d\mathbf{t}_1^T} & (\mathbf{0}\cdots\mathbf{0})_{2\times(3\times(M-1))} \\ \vdots \\ \frac{d\hat{\mathbf{u}}_{1N}}{df_x} & \frac{d\hat{\mathbf{u}}_{1N}}{df_y} & \frac{d\hat{\mathbf{u}}_{1N}}{dc_x} & \frac{d\hat{\mathbf{u}}_{1N}}{dc_y} & \frac{d\hat{\mathbf{u}}_{1N}}{d\mathbf{k}^T} & \frac{d\hat{\mathbf{u}}_{1N}}{d\mathbf{d}_1^T} & (\mathbf{0}\cdots\cdots\mathbf{0})_{2\times(3\times(M-1))} & \frac{d\hat{\mathbf{u}}_{1N}}{d\mathbf{t}_1^T} & (\mathbf{0}\cdots\mathbf{0})_{2\times(3\times(M-1))} \\ \frac{d\hat{\mathbf{u}}_{21}}{df_x} & \frac{d\hat{\mathbf{u}}_{21}}{df_y} & \frac{d\hat{\mathbf{u}}_{21}}{dc_x} & \frac{d\hat{\mathbf{u}}_{21}}{dc_y} & \frac{d\hat{\mathbf{u}}_{21}}{d\mathbf{k}^T} & \mathbf{0}_{2\times3} & \frac{d\hat{\mathbf{u}}_{21}}{d\mathbf{d}_2^T} & (\mathbf{0}\cdots\mathbf{0})_{2\times(3\times(M-2))} & \mathbf{0}_{2\times3} & \frac{d\hat{\mathbf{u}}_{21}}{d\mathbf{t}_2^T} & (\mathbf{0}\cdots\mathbf{0})_{2\times(3\times(M-2))} \\ \vdots \\ \frac{d\hat{\mathbf{u}}_{MN}}{df_x} & \frac{d\hat{\mathbf{u}}_{MN}}{df_y} & \frac{d\hat{\mathbf{u}}_{MN}}{dc_x} & \frac{d\hat{\mathbf{u}}_{MN}}{dc_y} & \frac{d\hat{\mathbf{u}}_{MN}}{d\mathbf{k}^T} & (\mathbf{0}\cdots\mathbf{0})_{2\times(3\times(M-1))} & \frac{d\hat{\mathbf{u}}_{MN}}{d\mathbf{d}_M^T} & (\mathbf{0}\cdots\mathbf{0})_{2\times(3\times(M-1))} & \frac{d\hat{\mathbf{u}}_{MN}}{d\mathbf{t}_M^T} \end{bmatrix}$$

Nonlinear Least-squares

The core problem is to determine $\dfrac{d\hat{\mathbf{u}}_{ij}}{d\Theta^T}$,

i.e, to determine

$$\frac{d\hat{\mathbf{u}}_{ij}}{df_x},\frac{d\hat{\mathbf{u}}_{ij}}{df_y},\frac{d\hat{\mathbf{u}}_{ij}}{dc_x},\frac{d\hat{\mathbf{u}}_{ij}}{dc_y},\frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{k}^T},\frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{d}_i^T},\frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{t}_i^T}$$

Note that: $\dfrac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{d}_m^T}=\mathbf{0},\dfrac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{t}_m^T}=\mathbf{0},\forall m\neq i$

For derivation simplicity, in the following, we denote

$$\hat{\mathbf{u}}_{ij}\triangleq\begin{bmatrix}u\\v\end{bmatrix},\mathbf{d}_i\triangleq\begin{pmatrix}d_1\\d_2\\d_3\end{pmatrix},\mathbf{t}_i\triangleq\begin{pmatrix}t_1\\t_2\\t_3\end{pmatrix}$$

Denote $\mathbf{d}_i$'s rotation matrix representation by $\mathcal{R}(\mathbf{d}_i)=\begin{bmatrix}r_{11}&r_{12}&r_{13}\\r_{21}&r_{22}&r_{23}\\r_{31}&r_{32}&r_{33}\end{bmatrix}$ and its vector form by $\mathbf{r}=\begin{bmatrix}r_{11}\\r_{12}\\r_{13}\\r_{21}\\r_{22}\\r_{23}\\r_{31}\\r_{32}\\r_{33}\end{bmatrix}$

Nonlinear Least-squares

$\mathbf{p}_j \triangleq (x, y, z, 1)^T$

$\mathbf{p}_j$'s position w.r.t the camera coordinate system of camera i, $\mathbf{p}_c=[x_c,y_c,z_c]^T$

$\mathbf{p}_j$'s ideal projection on the normalized retinal plane of camera i, $\mathbf{p}_n=[x_n, y_n]^T$

$\mathbf{p}_j$'s distorted projection on the normalized retinal plane of camera i, $\mathbf{p}_d=[x_d, y_d]^T$

Note: $\mathbf{p}_j$ is known beforehand and is fixed during iterations; $\mathbf{p}_c$, $\mathbf{p}_n$, $\mathbf{p}_d$, and $\hat{\mathbf{u}}_{ij}$ can be computed using the imaging model with the current iteration point Θ_k as parameters. Thus, for the current iteration point Θ_k, Θ_k, $\mathbf{p}_c$, $\mathbf{p}_n$, $\mathbf{p}_d$, and $\hat{\mathbf{u}}_{ij}$ are all known

Let's derive the above-mentioned derivatives one by one……

Nonlinear Least-squares

According to Eq. 8 (from the projection on the normalized retinal plane to the final pixel position), we have

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \Rightarrow \begin{cases} u = f_x x_d + c_x \\ v = f_y y_d + c_y \end{cases}$$

$$\boxed{\frac{d\hat{\mathbf{u}}_{ij}}{df_x} = \begin{bmatrix} \frac{\partial u}{\partial f_x} \\ \frac{\partial v}{\partial f_x} \end{bmatrix} = \begin{bmatrix} x_d \\ 0 \end{bmatrix}, \frac{d\hat{\mathbf{u}}_{ij}}{df_y} = \begin{bmatrix} \frac{\partial u}{\partial f_y} \\ \frac{\partial v}{\partial f_y} \end{bmatrix} = \begin{bmatrix} 0 \\ y_d \end{bmatrix}, \frac{d\hat{\mathbf{u}}_{ij}}{dc_x} = \begin{bmatrix} \frac{\partial u}{\partial c_x} \\ \frac{\partial v}{\partial c_x} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \frac{d\hat{\mathbf{u}}_{ij}}{dc_y} = \begin{bmatrix} \frac{\partial u}{\partial c_y} \\ \frac{\partial v}{\partial c_y} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}}$$

We also have a byproduct which will be used later,

$$\frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{p}_d^T} = \begin{bmatrix} \frac{\partial u}{\partial x_d} & \frac{\partial u}{\partial y_d} \\ \frac{\partial v}{\partial x_d} & \frac{\partial v}{\partial y_d} \end{bmatrix} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix}$$

Nonlinear Least-squares

According to Eq. 7 and the notation $\mathbf{k} \triangleq \left[k_1\ k_2\ \rho_1\ \rho_2\ k_3\right]^T$ we have

$$\frac{d\mathbf{p}_d}{d\mathbf{k}^T} = \begin{bmatrix} \frac{\partial x_d}{\partial k_1} & \frac{\partial x_d}{\partial k_2} & \frac{\partial x_d}{\partial \rho_1} & \frac{\partial x_d}{\partial \rho_2} & \frac{\partial x_d}{\partial k_3} \\ \frac{\partial y_d}{\partial k_1} & \frac{\partial y_d}{\partial k_2} & \frac{\partial y_d}{\partial \rho_1} & \frac{\partial y_d}{\partial \rho_2} & \frac{\partial y_d}{\partial k_3} \end{bmatrix} = \begin{bmatrix} x_n r^2 & x_n r^4 & 2x_n y_n & r^2 + 2x_n^2 & x_n r^6 \\ y_n r^2 & y_n r^4 & r^2 + 2y_n^2 & 2x_n y_n & y_n r^6 \end{bmatrix}$$

Then we have,

$$\boxed{\frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{k}^T} = \frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{p}_d^T} \cdot \frac{d\mathbf{p}_d}{d\mathbf{k}^T} = \begin{bmatrix} f_x x_n r^2 & f_x x_n r^4 & 2 f_x x_n y_n & f_x (r^2 + 2x_n^2) & f_x x_n r^6 \\ f_y y_n r^2 & f_y y_n r^4 & f_y (r^2 + 2y_n^2) & 2 f_y x_n y_n & f_y y_n r^6 \end{bmatrix}}$$

Also based on Eq. 7, we can have

$$\frac{\partial \mathbf{p}_d}{\partial \mathbf{p}_n^T} = \begin{bmatrix} \frac{\partial x_d}{\partial x_n} & \frac{\partial x_d}{\partial y_n} \\ \frac{\partial y_d}{\partial x_n} & \frac{\partial y_d}{\partial y_n} \end{bmatrix} = \begin{bmatrix} 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + 2x_n^2\left(k_1 + 2k_2 r^2 + 3k_3 r^4\right) + 2\rho_1 y_n + 6\rho_2 x_n & 2x_n y_n\left(k_1 + 2k_2 r^2 + 3k_3 r^4\right) + 2\left(\rho_1 x_n + \rho_2 y_n\right) \\ 2x_n y_n\left(k_1 + 2k_2 r^2 + 3k_3 r^4\right) + 2\left(\rho_1 x_n + \rho_2 y_n\right) & 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + 2y_n^2\left(k_1 + 2k_2 r^2 + 3k_3 r^4\right) + 2\rho_2 x_n + 6\rho_1 y_n \end{bmatrix}$$

Nonlinear Least-squares

According to Eq. 3, we have

$$\frac{d\mathbf{p}_n}{d\mathbf{p}_c^T} = \begin{bmatrix} \frac{\partial x_n}{\partial x_c} & \frac{\partial x_n}{\partial y_c} & \frac{\partial x_n}{\partial z_c} \\ \frac{\partial y_n}{\partial x_c} & \frac{\partial y_n}{\partial y_c} & \frac{\partial y_n}{\partial z_c} \end{bmatrix} = \begin{bmatrix} \frac{1}{z_c} & 0 & \frac{-x_c}{z_c^2} \\ 0 & \frac{1}{z_c} & \frac{-y_c}{z_c^2} \end{bmatrix}$$

According to Eq. 1, we have

$$\mathbf{p}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11}x + r_{12}z + r_{13}z + t_1 \\ r_{21}x + r_{22}z + r_{23}z + t_2 \\ r_{31}x + r_{32}z + r_{33}z + t_3 \end{bmatrix} \Rightarrow \frac{d\mathbf{p}_c}{d\mathbf{r}^T} = \begin{bmatrix} \frac{\partial x_c}{\partial r_{11}} & \frac{\partial x_c}{\partial r_{12}} & \frac{\partial x_c}{\partial r_{13}} & \frac{\partial x_c}{\partial r_{21}} & \frac{\partial x_c}{\partial r_{22}} & \frac{\partial x_c}{\partial r_{23}} & \frac{\partial x_c}{\partial r_{31}} & \frac{\partial x_c}{\partial r_{32}} & \frac{\partial x_c}{\partial r_{33}} \\ \frac{\partial y_c}{\partial r_{11}} & \frac{\partial y_c}{\partial r_{12}} & \frac{\partial y_c}{\partial r_{13}} & \frac{\partial y_c}{\partial r_{21}} & \frac{\partial y_c}{\partial r_{22}} & \frac{\partial y_c}{\partial r_{23}} & \frac{\partial y_c}{\partial r_{31}} & \frac{\partial y_c}{\partial r_{32}} & \frac{\partial y_c}{\partial r_{33}} \\ \frac{\partial z_c}{\partial r_{11}} & \frac{\partial z_c}{\partial r_{12}} & \frac{\partial z_c}{\partial r_{13}} & \frac{\partial z_c}{\partial r_{21}} & \frac{\partial z_c}{\partial r_{22}} & \frac{\partial z_c}{\partial r_{23}} & \frac{\partial z_c}{\partial r_{31}} & \frac{\partial z_c}{\partial r_{32}} & \frac{\partial z_c}{\partial r_{33}} \end{bmatrix} = \begin{bmatrix} x & y & z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x & y & z \end{bmatrix}$$

$$\frac{d\mathbf{p}_c}{d\mathbf{t}_i^T} = \begin{bmatrix} \frac{\partial x_c}{\partial t_1} & \frac{\partial x_c}{\partial t_2} & \frac{\partial x_c}{\partial t_3} \\ \frac{\partial y_c}{\partial t_1} & \frac{\partial y_c}{\partial t_2} & \frac{\partial y_c}{\partial t_3} \\ \frac{\partial z_c}{\partial t_1} & \frac{\partial z_c}{\partial t_2} & \frac{\partial z_c}{\partial t_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nonlinear Least-squares

Let $\theta = \|\mathbf{d}_i\|, \mathbf{n} = (n_1, n_2, n_3)^T = \dfrac{\mathbf{d}_i}{\|\mathbf{d}_i\|}$ $\qquad \alpha \triangleq \sin\theta, \beta \triangleq \cos\theta, \gamma \triangleq 1-\cos\theta$

According to Rodrigues formula (Eq. 10), we can have,

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \beta \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} + \gamma \begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix} + \alpha \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

$$\Rightarrow \quad \frac{\partial \mathbf{r}}{\partial \mathbf{d}_i^T} = \begin{bmatrix}
\frac{2\gamma n_1(1-n_1^2)}{\theta} + \alpha n_1(n_1^2-1) & -\frac{2\gamma n_1^2 n_2}{\theta} + \alpha n_2(n_1^2-1) & -\frac{2\gamma n_1^2 n_3}{\theta} + \alpha n_3(n_1^2-1) \\
n_1(\alpha n_1 n_2 - \beta n_3) + \frac{\gamma n_2(1-2n_1^2) + \alpha n_1 n_3}{\theta} & n_2(\alpha n_1 n_2 - \beta n_3) + \frac{\gamma n_1(1-2n_2^2) + \alpha n_2 n_3}{\theta} & n_3(\alpha n_1 n_2 - \beta n_3) + \frac{\alpha(n_3^2-1) - 2\gamma n_1 n_2 n_3}{\theta} \\
n_1(\alpha n_1 n_3 + \beta n_2) + \frac{\gamma n_3(1-2n_1^2) - \alpha n_1 n_2}{\theta} & n_2(\alpha n_1 n_3 + \beta n_2) + \frac{\alpha(1-n_2^2) - 2\gamma n_1 n_2 n_3}{\theta} & n_3(\alpha n_1 n_3 + \beta n_2) + \frac{\gamma n_1(1-2n_3^2) - \alpha n_2 n_3}{\theta} \\
n_1(\alpha n_1 n_2 + \beta n_3) + \frac{\gamma n_2(1-2n_1^2) - \alpha n_1 n_3}{\theta} & n_2(\alpha n_1 n_2 + \beta n_3) + \frac{\gamma n_1(1-2n_2^2) - \alpha n_2 n_3}{\theta} & n_3(\alpha n_1 n_2 + \beta n_3) + \frac{\alpha(1-n_3^2) - 2\gamma n_1 n_2 n_3}{\theta} \\
\frac{-2\gamma n_1 n_2^2}{\theta} + \alpha n_1(n_2^2-1) & \frac{2\gamma n_2(1-n_2^2)}{\theta} + \alpha n_2(n_2^2-1) & -\frac{2\gamma n_2^2 n_3}{\theta} + \alpha n_3(n_2^2-1) \\
n_1(\alpha n_2 n_3 - \beta n_1) - \frac{\alpha(1-n_1^2) + 2\gamma n_1 n_2 n_3}{\theta} & n_2(\alpha n_2 n_3 - \beta n_1) + \frac{\gamma n_3(1-2n_2^2) + \alpha n_1 n_2}{\theta} & n_3(\alpha n_2 n_3 - \beta n_1) + \frac{\alpha n_1 n_3 + \gamma n_2(1-2n_3^2)}{\theta} \\
n_1(\alpha n_1 n_3 - \beta n_2) + \frac{\alpha n_1 n_2 + \gamma n_3(1-2n_1^2)}{\theta} & n_2(\alpha n_1 n_3 - \beta n_2) - \frac{\alpha(1-n_2^2) + 2\gamma n_1 n_2 n_3}{\theta} & n_3(\alpha n_1 n_3 - \beta n_2) + \frac{\alpha n_2 n_3 + \gamma n_1(1-2n_3^2)}{\theta} \\
n_1(\alpha n_2 n_3 + \beta n_1) + \frac{\alpha(1-n_1^2) - 2\gamma n_1 n_2 n_3}{\theta} & n_2(\alpha n_2 n_3 + \beta n_1) + \frac{\gamma n_3(1-2n_2^2) - \alpha n_1 n_2}{\theta} & n_3(\alpha n_2 n_3 + \beta n_1) + \frac{\gamma n_2(1-2n_3^2) - \alpha n_1 n_3}{\theta} \\
\frac{-2\gamma n_1 n_3^2}{\theta} + \alpha n_1(n_3^2-1) & -\frac{2\gamma n_2 n_3^2}{\theta} + \alpha n_2(n_3^2-1) & \frac{2\gamma n_3(1-n_3^2)}{\theta} + \alpha n_3(n_3^2-1)
\end{bmatrix}$$

Nonlinear Least-squares

Then, we can compute,

$$\frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{d}_i^T} = \frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{p}_d^T} \cdot \frac{d\mathbf{p}_d}{d\mathbf{p}_n^T} \cdot \frac{d\mathbf{p}_n}{d\mathbf{p}_c^T} \cdot \frac{d\mathbf{p}_c}{d\mathbf{r}^T} \cdot \frac{d\mathbf{r}}{d\mathbf{d}_i^T}$$

$$\frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{t}_i^T} = \frac{d\hat{\mathbf{u}}_{ij}}{d\mathbf{p}_d^T} \cdot \frac{d\mathbf{p}_d}{d\mathbf{p}_n^T} \cdot \frac{d\mathbf{p}_n}{d\mathbf{p}_c^T} \cdot \frac{d\mathbf{p}_c}{d\mathbf{t}_i^T}$$

Till now, we have obtained all the derivatives required by $\mathbf{J}(\Theta)$

Outline

- What is Camera Calibration?
- Modeling for Imaging Pipeline
- General Framework for the Camera Calibration Algorithm
- Initial Rough Estimation of Calibration Parameters
- Nonlinear Least-squares
- Image Un-distortion

Image Un-distortion

With the calibrated camera, many amazing applications can be performed….

One naive example, the distorted image can be undistorted

Original fish-eye image

Undistorted image

How?

Image Un-distortion

$$\mathbf{u}_d = \mathbf{K}\mathcal{D}\left(\mathbf{K}^{-1}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}\right)$$

$$\mathbf{I}(\mathbf{u}) \leftarrow \mathbf{I}_d(\mathbf{u}_d)$$

Image Un-distortion

Q+A