CITS 4402 Computer Vision

Ajmal Mian
Mehdi Ravanbakhsh

Lecture 08 – Camera Calibration
Summary – Lecture 07

- Feature Detection
- Feature Extraction
- Harris Corner Detector
- Histogram of Oriented Gradients (HOG)
- Local Binary Patterns (LBP)
- Scale Invariant Feature Transform (SIFT)
Overview of this lecture

- What is camera calibration
- Why is it useful
- Pinhole camera model
- Perspective projection
- Camera calibration matrix derivation
- Camera calibration matrix estimation
- Calculating intrinsic and extrinsic camera parameters
- Calibration demo
What is camera calibration?

- The process of estimating camera parameters
- The 3D world coordinates are projected on the 2D image plane (film)
- The relationship between the world coordinates and image coordinates is defined by the camera parameters
- There are 5 intrinsic and 6 extrinsic camera parameters
Why do we need to calibrate cameras?

- To estimate the 3D geometry of the world
- To correct imaging artefacts caused by imperfect lenses

Examples include
- Stereo reconstruction
- Multiview reconstruction
- Single view measurements such as the height of a person
- Lens distortion correction
Camera parameters

- Intrinsic parameters (total 5)
  - Focal length in pixel coordinates (2 if the pixels are rectangular)
  - Principle point (2 coordinates)
  - Skew

- Extrinsic parameters (total 6)
  - 3 rotations
  - 3 translations
The pinhole camera model

- A box with a pinhole will make an inverted image of the object on its back plane

- The holes must be a point which is practically impossible

- Used as a simplified model of the real camera

- Why pinhole?
  - If the hole is not small, rays of light will not be focused
Camera calibration: Quick overview

- Imagine a pinhole camera with the imaging plane in front.

- C is the optical center, M is a point in 3D space, and m is its projection.

\[ sm = K[R \ t]M \]

\[ K = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \]

‘K’ contains the 5 intrinsic parameters and is called the camera intrinsic matrix.

‘R’ is the rotation matrix and ‘t’ the translation vector. R and t are the extrinsic parameters and have 3 degrees of freedom each.

\[ \gamma = \alpha \cot \theta \]

Axis skew causes shear distortion.
Camera calibration: Quick overview (cont’d)

- Let us expand this equation

\[ sm = K[R \ t]M \]

- Pixel coordinates \((u,v)\) start from the top left corner of the image
Camera calibration : Quick overview (cont’d)

\[
\begin{bmatrix}
  su \\
  sv \\
  s
\end{bmatrix}
= 
\begin{bmatrix}
  \alpha & \gamma & u_0 & 0 \\
  0 & \beta & v_0 & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
  r_{11} & r_{12} & r_{13} & t_x \\
  r_{21} & r_{22} & r_{23} & t_y \\
  r_{31} & r_{32} & r_{33} & t_z \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

3D location of the point in Camera coordinates
Camera calibration : Quick overview (cont’d)

\[
\begin{bmatrix}
    su \\
    sv \\
    s
\end{bmatrix}
= \begin{bmatrix}
    \alpha & \gamma & u_0 & 0 \\
    0 & \beta & v_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_x \\
    r_{21} & r_{22} & r_{23} & t_y \\
    r_{31} & r_{32} & r_{33} & t_z \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

3x4 Camera Calibration Matrix
Perspective projection

- The projection of a 3D point relative to the camera coordinate system is

\[ x = \frac{Xf}{Z}, \quad y = \frac{Yf}{Z} \]

- We can write this in matrix form using homogeneous coordinates

\[
\begin{bmatrix}
sx \\
 sy \\
 s \\
\end{bmatrix} = \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
Z \\
1 \\
\end{bmatrix}
\]

- Pixel coordinates are defined w.r.t. the top left corner of the image

\[
u = u_0 + x \\
v = v_0 + y
\]
Intrinsic camera matrix derivation

- Substituting these values
  \[ x = u - u_0 \]
  \[ y = v - v_0 \]

- in the perspective projection equation

\[
\begin{bmatrix}
  s(u - u_0) \\
  s(v - v_0) \\
  s
\end{bmatrix} =
\begin{bmatrix}
  f & 0 & 0 & 0 \\
  0 & f & 0 & 0 \\
  0 & 0 & 1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

- and rearranging

Note that for the simple case, we only have 3 intrinsic parameters but for the general case we have 5.
Looking back at the general case

The world coordinates are first rotated and translated to the camera coordinates.

\[
\begin{bmatrix}
    s u \\
    s v \\
    s
\end{bmatrix}
= \begin{bmatrix}
    \alpha & \gamma & u_0 & 0 \\
    0 & \beta & v_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_x \\
    r_{21} & r_{22} & r_{23} & t_y \\
    r_{31} & r_{32} & r_{33} & t_z \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

- The world coordinates are first rotated and translated to the camera coordinates.

- $\alpha$ and $\beta$ are the focal length expressed in pixels. Due to rectangular pixels, scale factors differ along the x and y dimensions.

- $\gamma$ is non-zero if there is skew in the image plane.
The camera calibration matrix $P$ is defined only up to an unknown scale $s$. Thus the last term in $P$ can be set to 1.

The aim of camera calibration is to find these unknowns given a set of known XYZ world points and their corresponding uv locations in an image.
For $i = 1 \ldots N$ points

\[
\begin{bmatrix}
sw_i \\
sv_i \\
s
\end{bmatrix} =
\begin{bmatrix}
q_{11} & q_{12} & q_{13} & q_{14} \\
q_{21} & q_{22} & q_{23} & q_{24} \\
q_{31} & q_{32} & q_{33} & 1
\end{bmatrix}
\begin{bmatrix}
X_i \\
Y_i \\
Z_i \\
1
\end{bmatrix}
\]

\[
s = X_i q_{31} + Y_i q_{32} + Z_i q_{33} + 1
\]

\[
u_i = \frac{X_i q_{11} + Y_i q_{12} + Z_i q_{13} + q_{14}}{X_i q_{31} + Y_i q_{32} + Z_i q_{33} + 1}
\]

\[
v_i = \frac{X_i q_{21} + Y_i q_{22} + Z_i q_{23} + q_{24}}{X_i q_{31} + Y_i q_{32} + Z_i q_{33} + 1}
\]
Writing as a system of linear equations \((Aq = b)\)

\[
\begin{align*}
X_i q_{11} + Y_i q_{12} + Z_i q_{13} + q_{14} - u_i X_i q_{31} - u_i Y_i q_{32} - u_i Z_i q_{33} &= u_i \\
X_i q_{21} + Y_i q_{22} + Z_i q_{23} + q_{24} - v_i X_i q_{31} - v_i Y_i q_{32} - v_i Z_i q_{33} &= v_i
\end{align*}
\]

How many equations do we need to solve this?
How many points do we need to solve this?

\[
\begin{bmatrix}
X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i X_i & -u_i Y_i & -u_i Z_i \\
0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_i X_i & -v_i Y_i & -v_i Z_i
\end{bmatrix}
\begin{bmatrix}
q_{11} \\
q_{12} \\
q_{13} \\
q_{14} \\
q_{21} \\
q_{22} \\
q_{23} \\
q_{24} \\
q_{31} \\
q_{32} \\
q_{33}
\end{bmatrix}
= 
\begin{bmatrix}
u_i \\
v_i
\end{bmatrix}
\]

2N x 11 Matrix

11 vector of unknowns

2N vector
Camera calibration targets

- 3D checker board makes a good calibration target

- Any 3D structure where precise XYZ locations of some points (>5) are known can be used

- 3D targets
  - Calibration is easy with single image
  - Most accurate approach
  - Expensive target
  - Visibility of points can be an issue in case of multiple cameras located at different view points

- 2D planar targets
  - Simpler target and better visibility for multiple cameras
  - Needs multiple images after rotating the target

- 1D line targets
  - Simplest target and can be seen from 360 degrees
  - Still very new
Solving for $q$

- Since every point gives two equations, we need at least 6 non-coplanar points to solve $Aq = b$.

- We can solve this using linear least squares:

\[
Aq = b \\
\Rightarrow A^T Aq = A^T b \\
\Rightarrow q = (A^T A)^{-1} A^T b
\]

- Can be solved in one line of Matlab $q = A \backslash b$;

- For a unique solution, $A^T A$ must be non-singular i.e. $\text{rank}(A^T A)$ or $\text{rank}(A)$ must be 11.

Need $2N \geq 11, \ N \geq 6$ non-coplanar points.
Calculating the camera calibration matrix (2\textsuperscript{nd} method)

Removing the condition \( q_{34} = 1 \)

\[
\begin{bmatrix}
    su_i \\
    sv_i \\
    s
\end{bmatrix} = 
\begin{bmatrix}
    q_{11} & q_{12} & q_{13} & q_{14} \\
    q_{21} & q_{22} & q_{23} & q_{24} \\
    q_{31} & q_{32} & q_{33} & q_{34}
\end{bmatrix}
\begin{bmatrix}
    X_i \\
    Y_i \\
    Z_i \\
    1
\end{bmatrix}
\]

\[ s = X_i q_{31} + Y_i q_{32} + Z_i q_{33} + q_{34} \]

\[ u_i = \frac{X_i q_{11} + Y_i q_{12} + Z_i q_{13} + q_{14}}{X_i q_{31} + Y_i q_{32} + Z_i q_{33} + q_{34}} \]

\[ v_i = \frac{X_i q_{21} + Y_i q_{22} + Z_i q_{23} + q_{24}}{X_i q_{31} + Y_i q_{32} + Z_i q_{33} + q_{34}} \]
Calculating the camera calibration matrix (2\textsuperscript{nd} method)

\[ X_i q_{11} + Y_i q_{12} + Z_i q_{13} + q_{14} - u_i X_i q_{31} - u_i Y_i q_{32} - u_i Z_i q_{33} - u_i q_{34} = 0 \]

\[ X_i q_{21} + Y_i q_{22} + Z_i q_{23} + q_{24} - v_i X_i q_{31} - v_i Y_i q_{32} - v_i Z_i q_{33} - v_i q_{34} = 0 \]

\[
\begin{bmatrix}
X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i X_i & -u_i Y_i & -u_i Z_i & -u_i \\
0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_i X_i & -v_i Y_i & -v_i Z_i & -v_i
\end{bmatrix}
\begin{bmatrix}
q_{11} \\
q_{12} \\
q_{13} \\
q_{14} \\
q_{21} \\
q_{22} \\
q_{23} \\
q_{24} \\
q_{31} \\
q_{32} \\
q_{33} \\
q_{34}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

\[ Aq = 0 \]
Solving for \( q \) using Eigen decomposition (Direct Linear Transformation or DLT)

- To find the solution of \( Aq = 0 \)

- We need to find the non-trivial null vector of \( A \).

- ‘\( A \)’ can have up to 12 eigenvalues.

- Case 1: If rank(\( A \)) is 12, its nullity is zero. There is no non-trivial null vector of \( A \).

- Case 2: If rank(\( A \)) is 11, it will have exactly one zero eigenvalue and the corresponding eigenvector will be the solution of \( Aq = 0 \)

- Case 3: If rank(\( A \)) < 11, there are infinite solutions to \( Aq = 0 \)
  Check if data is degenerate. Recalibrate.
Solving for q

- In practice, the smallest eigenvalue of ‘A’ will not be exactly equal to zero but will have a small value due to noise.

- The smallest eigenvector of ‘A’ is our solution i.e. q.

- Rule of thumb: Always check the smallest eigenvalue and/or the ratio between the largest and smallest values to estimate noise in the data.

- High levels of noise mean error in the construction of the matrix A.
Problem Statement:

**Given** \( n \) correspondences \( x_i \leftrightarrow X_i \), where \( X_i \) is a scene point and \( x_i \) its image:

**Compute**

\[
P = K [R \vert t]
\]
such that \( x_i = PX_i \).

The algorithm for camera calibration has two parts:

(i) Compute the matrix \( P \) from a set of point correspondences.

(ii) Decompose \( P \) into \( K, R \) and \( t \) via the \( QR \) decomposition.
Decomposition of the P matrix

\[
\begin{bmatrix}
    su \\
    sv \\
    s
\end{bmatrix} = \begin{bmatrix}
    \alpha & \gamma & u_0 & 0 \\
    0 & \beta & v_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_x \\
    r_{21} & r_{22} & r_{23} & t_y \\
    r_{31} & r_{32} & r_{33} & t_z \\
    0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    su \\
    sv \\
    s
\end{bmatrix} = \begin{bmatrix}
    \alpha & \gamma & u_0 & 0 \\
    0 & \beta & v_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
    r_1^T & t_x \\
    r_2^T & t_y \\
    r_3^T & t_z \\
    0^T & 1
\end{bmatrix} \begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
    \alpha r_1^T + \gamma r_2^T + u_0 r_3^T & \alpha t_x + \gamma t_y + u_0 t_z \\
    \beta r_2^T + v_0 r_3^T & \beta t_y + v_0 t_z \\
    r_3^T & t_z
\end{bmatrix} \approx \begin{bmatrix}
    q_1^T & q_{14} \\
    q_2^T & q_{24} \\
    q_3^T & q_{34}
\end{bmatrix}
\]

- The ‘q’ matrix computed with DLT differs from ‘P’ by an unknown scale ‘s’
Properties of a rotation matrix \( R \)

- Can be used to decompose the \( P \) matrix into intrinsic and extrinsic parameters.

- Sum of squares of the elements in each row or column = 1
  \[
  \|r_1\| = \|r_2\| = \|r_3\| = 1
  \]
  \[
  r_1^T \cdot r_1^T = r_2^T \cdot r_2^T = r_3^T \cdot r_3^T = 1
  \]

- Dot product of any pair of rows or any pair of columns = 0
  \[
  r_1^T \cdot r_2^T = r_1^T \cdot r_3^T = r_2^T \cdot r_3^T = 0
  \]

- The rows of \( R \) represent the coordinates axes of the rotated space in the original space. (vice versa for columns of \( R \)).

- Determinant of \( R \) is +1.
Decomposing the P matrix

We can write the camera matrix $P$ as follows:

$$P = [M\ |\ -MC] = [M\ |\ b]$$

$M$ is a 3x3 invertible matrix and $C$ is the camera center position in world coordinates.
- Good to project 3D into 2D
- It does not tell you about the camera pose
- It does not tell you about the camera’s internal geometry

We can decompose this into intrinsic and extrinsic matrices as follows:

$$P = K\ [R\ |\ -RC] = K\ [R\ |\ t]$$

Where $K$ is the camera intrinsic matrix and $R$ is the rotation matrix. $t = -RC$ is the translation vector or the position of the world origin in camera coordinates.

$$M = KR$$
$$b = Kt$$
Recovering the intrinsic parameters

\[
M = KR
\]

\[
B = MM^T = KK^T = \begin{bmatrix}
\alpha^2 + \gamma^2 + u_0^2 & \gamma \beta + u_0 v_0 & u_0 \\
\gamma \beta + u_0 v_0 & \beta^2 + v_0^2 & v_0 \\
u_0 & v_0 & 1
\end{bmatrix}
\]

Since \( P \) is defined up to a scale factor, the last element is usually not 1. Therefore, we have to normalize \( B \) so that the last element is 1. Intrinsic parameters are then calculated as follows:

\[
u_0 = B_{13}
\]

\[
v_0 = B_{23}
\]

\[
\beta = \sqrt{B_{22} - v_0^2}
\]

\[
\gamma = \frac{B_{12} - u_0 v_0}{\beta}
\]

\[
\alpha = \sqrt{B_{11} - u_0^2 - \gamma^2}
\]

Both \( \alpha > 0 \) and \( \beta > 0 \), and thus unambiguous.
Recovering the extrinsic parameters

- Once intrinsic parameters are known, $K$ is known.
- Extrinsic parameters can be calculated as

$$R = K^{-1} M$$
$$t = K^{-1} b$$
Decomposing matrix \( P \) with RQ-factorization

- Can also recover intrinsic and extrinsic parameters with RQ-factorization

- Recall

\[
P = \begin{bmatrix} M & -MC \end{bmatrix} = \begin{bmatrix} M & b \end{bmatrix}
\]

\[
P = K \begin{bmatrix} R & -RC \end{bmatrix} = K \begin{bmatrix} R & t \end{bmatrix}
\]

\[
M = KR
\]

- Notice that \( K \) is an upper triangular matrix and \( R \) is an orthonormal matrix.

- We can recover \( K \) and \( R \) by RQ-factorization of \( M \).

- ‘\( t \)’ can be recovered by \( t = -RC \) where \( C = -M^{-1}b \)

\[
[K, R, C, pp, pv] = \text{decomposecamera}(P); \ % \text{use Peter Kovesi’s Matlab code}
\]
Remarks

- Since RQ-factorization doesn’t have a unique solution, the diagonal entries are forced to be positive. (Negating a col in K and the corresponding row in R will give the same camera matrix P). This is a correct approach if
  - Your image’s X/Y axes point in the same direction as your camera’s X/Y axes
  - Your camera looks in the positive-z direction

- If the z-coordinates of the camera and world are pointing in the opposite direction, things can go wrong e.g. in OpenGL, the camera points in the negative z direction.

- For the general case some checks must be performed:
  - If camera and world x-axes are opposite, negate the 1st col of K and row of R.
  - If camera and world y-axes are opposite, negate the 2nd col of K and row of R.
  - If camera and world z-axes are opposite, negate the 3rd col of K and row of R.
  - If det(K) = -1, multiply K by -1.
Limitations of the linear approach

- Using least square minimization to get ‘q’ has little physical meaning.

- The method ignores constraints on the elements of ‘P’. The elements of ‘P’ are not arbitrary e.g. we may not be able to decompose it into an intrinsic and extrinsic parameter matrix.

- A more accurate approach is to use constrained non-linear optimization to find the calibration matrix.
Constrained non-linear optimization

- Estimate \( P \) using one of the linear methods.

- Use this \( P \) as an initial guess and reproject the points on the image plane.

- Minimize the distance between all measured and reprojected image points.

\[
\min_{\alpha, \beta, \gamma, u_0, v_0, R, t} \sum_{i=1}^{N} \left\{ \left( \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} - u_i \right)^2 + \left( \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} - v_i \right)^2 \right\}
\]

- Ensure that \( R \) remains a rotation matrix.

- Iterate until convergence.
Radial distortion

- Barrel distortion
  - Image magnification decreases with distance from the optical axis

- Pincussion distortion
  - Image magnification increases with distance from the optical axis

- Distortion is higher as we move away from the center of the image

- Thus distortion is a function of the radius ‘r’
Radial distortion correction

We can model radial distortion in the projection by applying a simple polynomial transformation.

Apply radial distortion to normalized camera coordinates

\[
x_n = \frac{X}{Z}, \quad y_n = \frac{Y}{Z} \quad \text{(normalized image coordinates)}
\]

\[
r^2 = x_n^2 + y_n^2
\]

\[
x_d = x_n(1 + \kappa_1 r^2 + \kappa_2 r^4 \cdots)
\]

\[
y_d = y_n(1 + \kappa_1 r^2 + \kappa_2 r^4 \cdots)
\]

Use distorted camera coordinates to calculate image coordinates

\[
u = fx_d + u_0
\]

\[
v = fy_d + v_0
\]
Barrel distortion correction example
Calibration Demo


Click on the four extreme corners of the rectangular pattern...
Calibration Demo

Calibration images
Calibration Demo

Click #1 (origin)

Click #2

Click #3

Click #4
Calibration Demo
Calibration Demo
Calibration Demo

Reprojection error (in pixel) - To exit: right button
Calibration Demo
Calibration Demo
Summary

- What is camera calibration
- Why is it useful
- Perspective projection
- Estimating the camera projection matrix
- Recovering the intrinsic and extrinsic parameters
- Radial distortion
- Calibration demo

Acknowledgements: Material for this lecture was taken from Zhang’s book chapter on camera calibration, Michigan University, Penn State University, Washington University and previous lectures delivered by Du Huynh and Peter Kovesi.