Camera Parameters, Calibration and Radiometry

Readings Forsyth & Ponce-
  Chap 1 & 2
  Chap 3.1.1 & 3.2
  Chap 4
Camera parameters

• From last time….

\[
\begin{pmatrix}
U \\
V \\
W
\end{pmatrix} = \left( \begin{array}{cccc}
\text{Transformation} & 1 & 0 & 0 & 0 \\
\text{representing} & 0 & 1 & 0 & 0 \\
\text{intrinsic parameters} & 0 & 0 & 1 & 0
\end{array} \right) \left( \begin{array}{c}
\text{Transformation} \\
\text{representing} \\
\text{extrinsic parameters}
\end{array} \right) \begin{pmatrix}
X \\
Y \\
Z \\
T
\end{pmatrix}
\]
Homogeneous Coordinates (Again)

\[
\begin{pmatrix}
U \\
V \\
W
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1/f & 0
\end{pmatrix} \begin{pmatrix}
X \\
Y \\
Z \\
T
\end{pmatrix}
\]

\((U, V, W) \rightarrow \left(\frac{U}{W}, \frac{V}{W}\right) = (u, v)\)
Extrinsic Parameters: Characterizing Camera position in *world* coordinates

**Chasles's theorem:**
Any motion of a solid body can be composed of a *translation* and a *rotation*.
3D Rotation Matrices

\[ R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix} \]

\[ R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix} \]

\[ R_z(\gamma) = \begin{bmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \]
Euler’s Theorem: An arbitrary rotation can be described by 3 rotation parameters.

For example: \[ R = R_x(\alpha) R_y(\beta) R_z(\gamma) \]

More Generally:

Most General:

\[
\begin{bmatrix}
    x'_1 \\
    x'_2 \\
    x'_3
\end{bmatrix} =
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3
\end{bmatrix}.
\]
Rodrique's Formula

Take any rotation axis $a$ and angle $\theta$: What is the matrix?

$$p'(t) = a \times p(t)$$

$$a \times v = \begin{bmatrix} a_y v_z - a_z v_y \\ a_z v_x - a_x v_z \\ a_x v_y - a_y v_x \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$= Av$$

with

$$A = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

$$e^{A\theta} = I + A \sin \theta + A^2 \left[1 - \cos \theta \right]$$

$$R = e^{A\theta}$$
Rotations can be represented as points in space (with care)

Turn vector length into angle, direction into axis:

Useful for generating random rotations, understanding angular errors, characterizing angular position, etc.

Problem: not unique
Not commutative
Other Properties of rotations

\[ R_1 \times R_2 \neq R_2 \times R_1 \]
Rotations

To form a rotation matrix, you can plug in the columns of new coordinate points.

For Example:
The unit x-vector goes to $x'$:

$$x' = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$
Other Properties of Rotations

**Inverse:** \( R^{-1} = R^T \)

Rows, columns are orthonormal

\[ r_i^T r_j = 0 \quad \text{if} \quad i \neq j, \quad \text{else} \quad r_i^T r_i = 1 \]

**Determinant:**

\[ \det( R ) = 1 \]

The effect of a coordinate rotation on a function:

\[ x' = R x \]

\[ F( x' ) = F( R^{-1} x ) \]
Extrinsic Parameters

\[ p' = R \ p + t \]

\( R = \) rotation matrix

\( t = \) translation vector

In Homogeneous coordinates,

\[ p' = R \ p + t \quad \Rightarrow \quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \]
2D Example

- Rotation $-90^\circ$

\[
\begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

- Scaling $\times 2$

\[
\begin{bmatrix}
2 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

- Translation

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{bmatrix}
\]

House Points

\[
p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]
Intrinsic Parameters

Differential Scaling
\[
\begin{bmatrix}
 u \times Su \\
 v \times Sv \\
 1
\end{bmatrix} =
\begin{bmatrix}
 Su & 0 & 0 \\
 0 & Sv & 0 \\
 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
 u \\
 v \\
 1
\end{bmatrix}
\]

Camera Origin Offset
\[
\begin{bmatrix}
 u + u_0 \\
 v + v_0 \\
 1
\end{bmatrix} =
\begin{bmatrix}
 1 & 0 & u_0 \\
 0 & 1 & v_0 \\
 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
 u \\
 v \\
 1
\end{bmatrix}
\]
The Whole (Linear) Transformation

\[
\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{intrinsic parameters} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}
\]

\[
\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} -f/s_u & 0 & u_0 & 0 \\ 0 & -f/s_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ T \end{bmatrix}
\]

Final image coordinates

\[u = U/W\]
\[v = U/W\]
Intrinsic parameters

Perspective projection

\[ u = f \frac{x}{z} \]

\[ v = f \frac{y}{z} \]
Intrinsic parameters

But “pixels” are in some arbitrary spatial units

\[ u = \alpha \frac{x}{z} \]
\[ v = \alpha \frac{y}{z} \]
Intrinsic parameters

Maybe pixels are not square

\[ u = \alpha \frac{x}{z} \]
\[ v = \beta \frac{y}{z} \]
Intrinsic parameters

We don’t know the origin of our camera pixel coordinates

\[ u = \alpha \frac{x}{z} + u_0 \]

\[ v = \beta \frac{y}{z} + v_0 \]
Intrinsic parameters

May be skew between camera pixel axes

\[ u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0 + \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0 \]
Intrinsic parameters

\[ u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0 \]
\[ v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0 \]

Using homogenous coordinates, we can write this as:

\[
\begin{pmatrix}
  u \\
  v \\
  1
\end{pmatrix} = \frac{1}{z} \begin{pmatrix}
  \alpha & -\alpha \cot(\theta) & u_0 & 0 \\
  0 & \frac{\beta}{\sin(\theta)} & v_0 & 0 \\
  0 & 0 & 1 & 0
\end{pmatrix} \begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]

or:

\[
\vec{p} = \frac{1}{z} \begin{pmatrix}
  K & \vec{0} \\
  \vec{0} & 1
\end{pmatrix} \vec{P}
\]

\[ \vec{p} = \frac{1}{z} \begin{pmatrix}
  K & \vec{0} \\
  \vec{0} & 1
\end{pmatrix} \vec{P} \]
Combining extrinsic and intrinsic calibration parameters

\[ p = \frac{1}{z} \mathcal{M} p, \quad \text{where} \quad \mathcal{M} = \mathcal{K}(\mathcal{R}, t), \quad (2.15) \]

\( \mathcal{R} = C W \mathcal{R} \) is a rotation matrix, \( t = C W O_W \) is a translation vector, and \( p = (W_x, W_y, W_z, 1)^T \) denotes the homogeneous coordinate vector of \( P \) in the frame \( (W) \).

A projection matrix can be written explicitly as a function of its five intrinsic parameters (\( \alpha, \beta, u_0, v_0, \) and \( \theta \)) and its six extrinsic ones (the three angles defining \( \mathcal{R} \) and the three coordinates of \( t \)), namely,

\[
\mathcal{M} = \begin{pmatrix}
\frac{\alpha r_1^T - \alpha \cot \theta r_2^T + u_0 r_3^T}{\sin \theta} & \frac{\beta}{\sin \theta} r_2^T + v_0 r_3^T & \frac{\beta}{\sin \theta} t_y + v_0 t_z \\
\frac{\alpha t_x - \alpha \cot \theta t_y + u_0 t_z}{\sin \theta} & \frac{\beta}{\sin \theta} t_x + u_0 t_y & 1
\end{pmatrix}, \quad (2.17)
\]

where \( r_1^T, r_2^T, \) and \( r_3^T \) denote the three rows of the matrix \( \mathcal{R} \) and \( t_x, t_y, \) and \( t_z \) are the coordinates of the vector \( t \).
Other ways to write the same equation

$$\vec{p} = \frac{1}{z} M \vec{P}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z} \begin{pmatrix} \cdots \\ m_1^T \\ \cdots \\ m_2^T \\ \cdots \\ m_3^T \\ \cdots \end{pmatrix} \begin{pmatrix} W_x \\ W_y \\ W_z \\ 1 \end{pmatrix}$$

$$u = \frac{m_1 \cdot \vec{P}}{m_3 \cdot \vec{P}}$$
$$v = \frac{m_2 \cdot \vec{P}}{m_3 \cdot \vec{P}}$$

$z$ is in the *camera* coordinate system, but we can solve for that, since $1 = \frac{m_3 \cdot \vec{P}}{z}$, leading to:
Non-linear distortions
(not handled by our treatment)
Camera Calibration

- You need to know something beyond what you get out of the camera
  - Known World points (object)
    - Traditional camera calibration
    - Linear and non-linear parameter estimation
  - Known camera motion
    - Camera attached to robot
    - Execute several different motions
  - Multiple Cameras
Example: Known world points vs. known camera motion

**Augmented pin-hole camera model**
- Focal point, orientation
- Focal length, aspect ratio, center, lens distortion

2D ↔ 3D correspondence
“Classical” calibration

2D ↔ 2D correspondence
SFM, “Self-calibration”
Classical Calibration

Know 3D coords, 2D coords

- Find projection matrix $\Pi$

\[
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix} =
\begin{bmatrix}
  p_{11} & p_{12} & p_{13} & p_{14} \\
  p_{21} & p_{22} & p_{23} & p_{24} \\
  p_{31} & p_{32} & p_{33} & p_{34}
\end{bmatrix}
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

- $11$ unknowns (up to scale)
- $2$ equations per point (eliminate $d$)
- $6$ points is sufficient
Classical Camera Calibration

Take a known set of points.

(Typically 3 orthogonal planes)

Treat a point in the object as the World origin

Points x1, x2, x3, ….

Project to y1, y2, y3, ….
Calibration Patterns

Calibration grid
Z. Zhang, Microsoft Research

Chromaglyphs
Bruce Culbertson, HP-labs
Camera calibration

From before, we had these equations relating image positions, \( u,v \), to points at 3-d positions \( P \) (in homogeneous coordinates):

\[
\begin{align*}
    u &= \frac{m_1 \cdot \vec{P}}{m_3 \cdot \vec{P}} \\
    v &= \frac{m_2 \cdot \vec{P}}{m_3 \cdot \vec{P}}
\end{align*}
\]

So for each feature point, \( i \), we have:

\[
\begin{align*}
    (m_1 - u_i m_3) \cdot \vec{P}_i &= 0 \\
    (m_2 - v_i m_3) \cdot \vec{P}_i &= 0
\end{align*}
\]
Camera calibration

Stack all these measurements of $i = 1 \ldots n$ points

$$(m_1 - u_i m_3) \cdot \vec{P}_i = 0$$

$$(m_2 - v_i m_3) \cdot \vec{P}_i = 0$$

into a big matrix:

$$\begin{pmatrix}
P_1^T & 0^T & -u_1 P_1^T \\
0^T & P_1^T & -v_1 P_1^T \\
\vdots & \vdots & \vdots \\
P_n^T & 0^T & -u_n P_n^T \\
0^T & P_n^T & -v_n P_n^T
\end{pmatrix}\begin{pmatrix}
m_1 \\
m_2 \\
m_3
\end{pmatrix} = \begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix}$$
In vector form:
\[
\begin{pmatrix}
P_1^T & 0^T & -u_1P_1^T \\
0^T & P_1^T & -v_1P_1^T \\
\vdots & \vdots & \vdots \\
P_n^T & 0^T & -u_nP_n^T \\
0^T & P_n^T & -v_nP_n^T \\
\end{pmatrix}
\begin{pmatrix}
m_1 \\
m_2 \\
m_3 \\
0 \\
0 \\
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
0 \\
\vdots \\
0 \\
0 \\
\end{pmatrix}
\]

Camera calibration

Showing all the elements:

\[
\begin{pmatrix}
P_{lx} & P_{ly} & P_{lz} & 1 & 0 & 0 & 0 & 0 & -u_1P_{lx} & -u_1P_{ly} & -u_1P_{lz} & -u_1 \\
0 & 0 & 0 & 0 & P_{lx} & P_{ly} & P_{lz} & 1 & -v_1P_{lx} & -v_1P_{ly} & -v_1P_{lz} & -v_1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
P_{nx} & P_{ny} & P_{nz} & 1 & 0 & 0 & 0 & 0 & -u_nP_{nx} & -u_nP_{ny} & -u_nP_{nz} & -u_n \\
0 & 0 & 0 & 0 & P_{nx} & P_{ny} & P_{nz} & 1 & -v_nP_{nx} & -v_nP_{ny} & -v_nP_{nz} & -v_n \\
\end{pmatrix}
\begin{pmatrix}
m_{11} \\
m_{12} \\
m_{13} \\
m_{14} \\
m_{21} \\
m_{22} \\
m_{23} \\
m_{24} \\
m_{31} \\
m_{32} \\
m_{33} \\
m_{34} \\
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
0 \\
\vdots \\
0 \\
0 \\
\end{pmatrix}
\]
We want to solve for the unit vector $m$ (the stacked one) that minimizes $|Pm|^2$.

The minimum eigenvector of the matrix $P^TP$ gives us that (see Forsyth&Ponce, 3.1)
Matlab code is simple

- Form the matrix $P$ by appending each 3D point and the corresponding pixel locations in the arrangement given shown in the previous slide:
  - $P = \begin{bmatrix} P_{x1} & P_{y1} & P_{z1} & 1 & 0 & 0 & 0 & 0 & -u_{1}*P_{x1} & -u_{1}*P_{y1} & -u_{1}*P_{z1} & -u_{1}; \\
 0 & 0 & 0 & 0 & P_{x1} & P_{y1} & P_{z1} & 1 & -v_{1}*P_{x1} & -v_{1}*P_{y1} & -v_{1}*P_{z1} & -v_{1}; \\
 0 & 0 & 0 & 0 & P_{x1} & P_{y1} & P_{z1} & 1 & -v_{1}*P_{x1} & -v_{1}*P_{y1} & -v_{1}*P_{z1} & -v_{1}; \\
 0 & 0 & 0 & 0 & P_{x1} & P_{y1} & P_{z1} & 1 & -v_{1}*P_{x1} & -v_{1}*P_{y1} & -v_{1}*P_{z1} & -v_{1}; \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots; \\
0 & 0 & 0 & 0 & P_{x1} & P_{y1} & P_{z1} & 1 & -v_{1}*P_{x1} & -v_{1}*P_{y1} & -v_{1}*P_{z1} & -v_{1}; \\
0 & 0 & 0 & 0 & P_{x1} & P_{y1} & P_{z1} & 1 & -v_{1}*P_{x1} & -v_{1}*P_{y1} & -v_{1}*P_{z1} & -v_{1}; \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots; \\
P_{x_n} & P_{y_n} & P_{z_n} & 1 & 0 & 0 & 0 & 0 & -u_{n}*P_{x_n} & -u_{n}*P_{y_n} & -u_{n}*P_{z_n} & -u_{n}; \\
0 & 0 & 0 & 0 & P_{x_n} & P_{y_n} & P_{z_n} & 1 & -v_{n}*P_{x_n} & -v_{n}*P_{y_n} & -v_{n}*P_{z_n} & -v_{n}; \\
0 & 0 & 0 & 0 & P_{x_n} & P_{y_n} & P_{z_n} & 1 & -v_{n}*P_{x_n} & -v_{n}*P_{y_n} & -v_{n}*P_{z_n} & -v_{n}\end{bmatrix}$

Compute eigenanalysis on $P^\prime*P$

$[V,D]=\text{eig}(P^\prime*P);$

$\text{eigenvalues} = \text{diag}(D);$

$j = \text{find(eigenvalues} == \text{min(eigenvalues))};$

$m=V(:,j);$
Camera calibration

Once you have the $M$ matrix, can recover the intrinsic and extrinsic parameters as in Forsyth&Ponce, sect. 3.2.2.

$$
M = \begin{pmatrix}
\alpha r_1^T - \alpha \cot \theta r_2^T + u_0 r_3^T & \alpha t_x - \alpha \cot \theta t_y + u_0 t_z \\
\frac{\beta}{\sin \theta} r_2^T + v_0 r_3^T & \frac{\beta}{\sin \theta} t_y + v_0 t_z \\
\frac{\beta}{\sin \theta} r_3^T & t_z
\end{pmatrix}
$$
Real Calibration

• Real calibration procedures look quite different
  – Weight points by correspondence quality
  – Nonlinear optimization
  – Linearizations
  – Non-linear distortions
  – Etc.
Camera Motion

\[ n_1 X' = d_1 \]
\[ n_2 X' = d_2 \]

Planar scene

4 degree-of-freedom gantry
Calibration Example
(Zhang, Microsoft)

8 Point matches manually picked
Motion algorithm used to calibrate camera
Applications of Calibrated Cameras

Image based rendering: Light field -- Hanrahan (Stanford)
Virtualized Reality
Projector-based VR
UNC Chapel Hill
Shader Lamps

Figure 1. An enthusiastic young user (Miriam Mintzer Fuchs, age 9 years) demonstrates our easy to use system for 3D painting on movable objects. Note that the color palette on the table and the color on the spherical tip of the “paint brush” are projected.
Is classical camera calibration necessary to to computer vision?

**NO. For example, Shape recovery without calibration (Fitzgibbons, Zisserman)**

**Fully automatic procedure:** Video in, VRML out.
- **Allows for irregular motion:** angle between views can vary, and it doesn't have to be known. Recovery of the angle is automatic, and accuracy is about 40 millidegrees standard deviation.
- **No calibration targets:** features on the objects themselves are used to determine where the camera is, relative to the turntable.

Right shows a shape model automatically extracted from a dinosaur image sequence without any additional information. Camera parameters are determined as well.
NEXT: Measuring light in images

• Geometric: (what we’ve been briefly covering)
  how positions in the image relate to 3-d positions in the world.

• Photometric/Radiometric:
  how the intensities in the image relate surface and lighting properties in the world.