3D Geometry and Camera Calibration
3D Coordinate Systems

- Right-handed vs. left-handed
2D Coordinate Systems

- y axis up vs. y axis down
- Origin at center vs. corner
- Will often write \((u, v)\) for image coordinates
3D Geometry Basics

- 3D points = column vectors

\[ \vec{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]

- Transformations = pre-multiplied matrices

\[ T\vec{p} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]
Rotation

- Rotation about the z axis

$$R_z = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Rotation about x, y axes similar (cyclically permute x, y, z)
Arbitrary Rotation

• Any rotation is a composition of rotations about $x$, $y$, and $z$

• Composition of transformations = matrix multiplication (watch the order!)

• Result: orthonormal matrix
  – Each row, column has unit length
  – Dot product of rows or columns $= 0$
  – Inverse of matrix $= \text{transpose}$
Arbitrary Rotation

- Rotate around $x, y, \text{then } z$:

$$R = \begin{bmatrix}
\sin \theta_y \cos \theta_z & -\cos \theta_x \sin \theta_z + \sin \theta_x \cos \theta_y \cos \theta_z & \sin \theta_x \sin \theta_z + \cos \theta_x \cos \theta_y \cos \theta_z \\
\sin \theta_y \sin \theta_z & \cos \theta_x \cos \theta_z + \sin \theta_x \cos \theta_y \sin \theta_z & -\sin \theta_x \cos \theta_y \cos \theta_z + \cos \theta_x \cos \theta_y \sin \theta_z \\
\cos \theta_y & -\sin \theta_x \sin \theta_y & -\cos \theta_x \sin \theta_y
\end{bmatrix}$$

- Don’t do this! Compute simple matrices and multiply them!
Scale

• Scale in $x$, $y$, $z$: 

$$
S = \begin{pmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & s_z \\
\end{pmatrix}
$$
Shear

- Shear parallel to xy plane:

\[
\sigma_{xy} = \begin{pmatrix}
1 & 0 & \sigma_x \\
0 & 1 & \sigma_y \\
0 & 0 & 1
\end{pmatrix}
\]
Translation

- Can translation be represented by multiplying by a $3 \times 3$ matrix?
- No.
- Proof:

\[ \forall A : A\vec{0} = \vec{0} \]
Homogeneous Coordinates

• Add a fourth dimension to each point:

\[
\begin{pmatrix}
x \\
y \\
z \\
w
\end{pmatrix} \rightarrow
\begin{pmatrix}
x \\
y \\
z \\
w
\end{pmatrix}
\]

• To get “real” (3D) coordinates, divide by \( w \):

\[
\begin{pmatrix}
x \\
y \\
z \\
w
\end{pmatrix} \rightarrow
\begin{pmatrix}
x/w \\
y/w \\
z/w \\
w/w
\end{pmatrix}
\]
Translation in Homogeneous Coordinates

\[
\begin{pmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
w
\end{pmatrix} =
\begin{pmatrix}
x + t_x w \\
y + t_y w \\
z + t_z w \\
w
\end{pmatrix}
\]

- After divide by \( w \), this is just a translation by \((t_x, t_y, t_z)\)
Perspective Projection

- What does 4th row of matrix do?

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
w \\
\end{pmatrix}
=
\begin{pmatrix}
x \\
y \\
z \\
w \\
\end{pmatrix}
\]

- After divide,

\[
\begin{pmatrix}
x \\
y \\
z \\
z \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
x/z \\
y/z \\
z \\
1 \\
\end{pmatrix}
\]
Perspective Projection

- This is projection onto the $z=1$ plane

- Add scaling, etc. $\Rightarrow$ pinhole camera model
Putting It All Together: A Camera Model

- Translate to image center
- Scale to pixel size
- Perspective projection
- Camera orientation
- Camera location
- 3D point (homogeneous coords)

\[ T_{img}, S_{pix}, P_{cam}, R_{cam}, T_{cam}, \tilde{x} \]

Then perform homogeneous divide, and get (u,v) coords
Putting It All Together:
A Camera Model

Intrinsics

Extrinsics

\[ T_{img}, S_{pix}, P_{cam}, R_{cam}, T_{cam}, \tilde{X} \]
Putting It All Together: A Camera Model

- Camera coordinates
- Normalized device coordinates
- Image coordinates
- Pixel coordinates
- Eye coordinates
- World coordinates

\[ T_{\text{img}}, S_{\text{pix}}, P_{\text{cam}}, R_{\text{cam}}, T_{\text{cam}}, \tilde{x} \]
More General Camera Model

- Multiply all these matrices together
- Don’t care about “z” after transformation

\[
\begin{pmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z \\
  1 \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  ax + by + cz + d \\
  ix + jy + kz + l \\
  ex + fy + gz + h \\
  ix + jy + kz + l \\
\end{pmatrix}
\]

- Scale ambiguity $\rightarrow$ 11 free parameters
Radial Distortion

• Radial distortion can not be represented by matrix

\[
\begin{align*}
    u_{\text{img}} & \rightarrow c_u + u_{\text{img}}^* \left(1 + \kappa(u_{\text{img}}^*^2 + v_{\text{img}}^*^2)\right) \\
    v_{\text{img}} & \rightarrow c_v + v_{\text{img}}^* \left(1 + \kappa(u_{\text{img}}^*^2 + v_{\text{img}}^*^2)\right)
\end{align*}
\]

• \((c_u, c_v)\) is image center,

\[ u^*_{\text{img}} = u_{\text{img}} - c_u, \quad v^*_{\text{img}} = v_{\text{img}} - c_v, \]

\(\kappa\) is first-order radial distortion coefficient
Camera Calibration

- Determining values for camera parameters
- Necessary for any algorithm that requires 3D ↔ 2D mapping
- Method used depends on:
  - What data is available
  - Intrinsics only vs. extrinsics only vs. both
  - Form of camera model
Camera Calibration – Example 1

• Given:
  – 3D ↔ 2D correspondences
  – General perspective camera model
    (11-parameter, no radial distortion)

• Write equations:

\[
\begin{align*}
\frac{ax_1 + by_1 + cz_1 + d}{ix_1 + jy_1 + kz_1 + l} &= u_1 \\
\frac{ex_1 + fy_1 + gz_1 + h}{ix_1 + jy_1 + kz_1 + l} &= v_1 \\
\vdots
\end{align*}
\]
Camera Calibration – Example 1

\[
\begin{bmatrix}
  x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 x_1 & -u_1 y_1 & -u_1 z_1 & -u_1 \\
  0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -u_1 x_1 & -u_1 y_1 & -u_1 z_1 & -u_1 \\
  x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2 x_2 & -u_2 y_2 & -u_2 z_2 & -u_2 \\
  0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -u_2 x_2 & -u_2 y_2 & -u_2 z_2 & -u_2 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\begin{bmatrix}
a \\ b \\ c \\ \vdots \\ l
\end{bmatrix}
= \begin{bmatrix} \hat{0} \end{bmatrix}
\]

- Linear equation
- Overconstrained (more equations than unknowns)
- Underconstrained (rank deficient matrix – any multiple of a solution, including 0, is also a solution)
Camera Calibration – Example 1

• Standard linear least squares methods for $Ax = 0$ will give the solution $x = 0$

• Instead, look for a solution with $|x| = 1$

• That is, minimize $|Ax|^2$ subject to $|x|^2 = 1$
Camera Calibration – Example 1

• Minimize $|Ax|^2$ subject to $|x|^2=1$

• $|Ax|^2 = (Ax)^T(Ax) = (x^T A^T)(Ax) = x^T(A^T A)x$

• Expand $x$ in terms of eigenvectors of $A^T A$:
  
  $x = \mu_1 e_1 + \mu_2 e_2 + \ldots$

  $x^T(A^T A)x = \lambda_1 \mu_1^2 + \lambda_2 \mu_2^2 + \ldots$

  $|x|^2 = \mu_1^2 + \mu_2^2 + \ldots$
Camera Calibration – Example 1

• To minimize

\[ \lambda_1 \mu_1^2 + \lambda_2 \mu_2^2 + \ldots \]

subject to

\[ \mu_1^2 + \mu_2^2 + \ldots = 1 \]

set \( \mu_{\text{min}} = 1 \) and all other \( \mu_i = 0 \)

• Thus, least squares solution is eigenvector of \( A^T A \)
corresponding to minimum eigenvalue
Camera Calibration – Example 2

- Incorporating additional constraints into camera model
  - No shear, no scale (rigid-body motion)
  - Square pixels
  - etc.

- These impose *nonlinear* constraints on camera parameters
Camera Calibration – Example 2

• Option 1: solve for general perspective model, then find closest solution that satisfies constraints

• Option 2: constrained nonlinear least squares
  – Usually “gradient descent” techniques
  – Common implementations available (e.g. Matlab optimization toolbox)
Camera Calibration – Example 3

• Incorporating radial distortion

• Option 1:
  – Find distortion first (straight lines in calibration target)
  – Warp image to eliminate distortion
  – Run (simpler) perspective calibration

• Option 2: nonlinear least squares
Camera Calibration – Example 4

• What if 3D points are not known?
• Structure from motion problem!
• As we saw, can often be solved since
  # of knowns > # of unknowns
Multi-Camera Geometry

• Epipolar geometry – relationship between observed positions of points in multiple cameras

• Assume:
  – 2 cameras
  – Known intrinsics and extrinsics
Epipolar Geometry

$C_1$ $p_1$ $C_2$ $p_2$
Epipolar Geometry
Epipolar Geometry

Epipolar line

Epipoles

C₁

C₂

p₁

p₂

l₂

P
Epipolar Geometry

- Goal: derive equation for $l_2$
- Observation: $P, C_1, C_2$ determine a plane
Epipolar Geometry

- Work in coordinate frame of $C_1$
- Normal of plane is $T \times Rp_2$, where $T$ is relative translation, $R$ is relative rotation
Epipolar Geometry

• $p_1$ is perpendicular to this normal:

$$p_1 \cdot (T \times R p_2) = 0$$
Epipolar Geometry

- Write cross product as matrix multiplication

\[ \vec{T} \times x = T^* x, \quad T^* = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix} \]
Epipolar Geometry

- $p_1 \cdot T^* R p_2 = 0 \Rightarrow p_1^T E p_2 = 0$
- $E$ is the essential matrix
Essential Matrix

- $E$ depends only on camera geometry
- Given $E$, can derive equation for line $l_2$
Fundamental Matrix

- Can define fundamental matrix \( F \) analogously, operating on pixel coordinates instead of camera coordinates
  \[
  u_1^T F u_2 = 0
  \]
- Advantage: can sometimes estimate \( F \) without knowing camera calibration