Camera Models Review, Camera Calibration

Lecture 10-11

Perspective Camera Model

Canonical View

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & p_x \\ 0 & m_y f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

General View

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & p_x \\ 0 & m_y f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \mid \mathbf{T}]\mathbf{X}$$

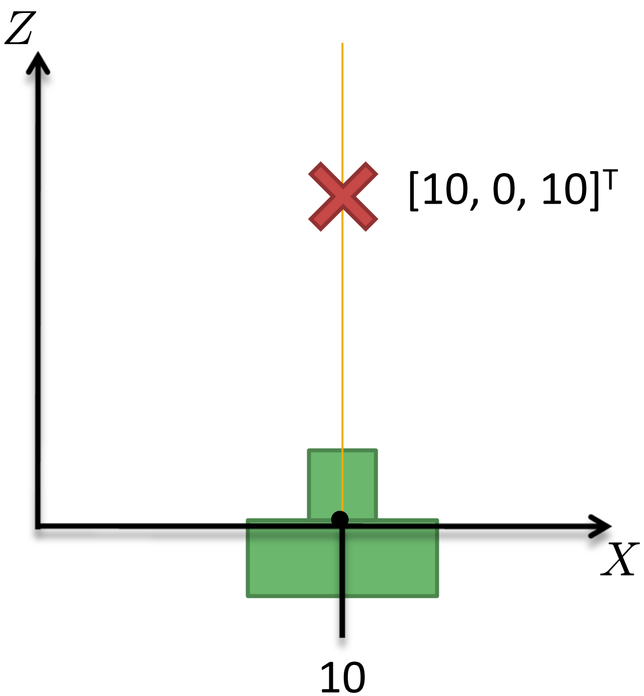
x – image point
X – world point
K – 3x3 matrix of internal camera parameters
[**R** | **T**] – 3x4 matrix of external camera parameters
R – rotation needed to align camera to world axes
T – Translation needed to bring camera to world origin
T = -RC where **C** is the vector to camera center

Camera Model

- If the camera is moved **C** from the origin, we should move the world point by $\mathbf{C}^{-1}$
- Then the perspective transform equation will be applicable
- Same holds for rotations

Example

- Translation by 10 units to the right

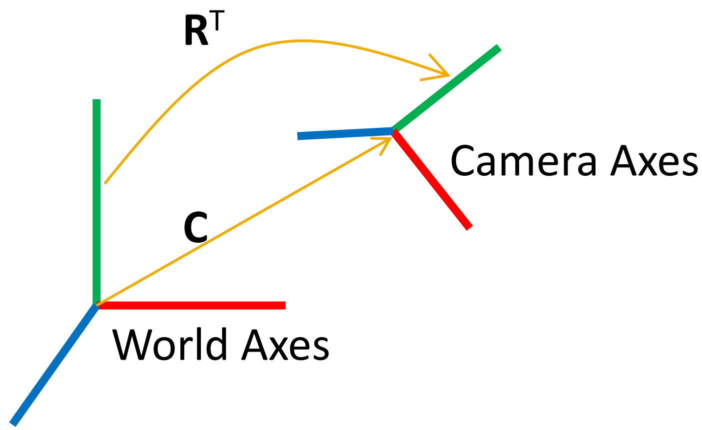


$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 0 \\ 10 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix}$$

Perspective Transform

- In general, the camera center is at a rotation of $\mathbf{R}^{\mathrm{T}}$, followed by a translation of $\mathbf{C}$ from the world origin
- Then



$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & p_x & 0 \\ 0 & m_y f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -C_X \\ 0 & 1 & 0 & -C_Y \\ 0 & 0 & 1 & -C_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right)$$

Perspective Transform

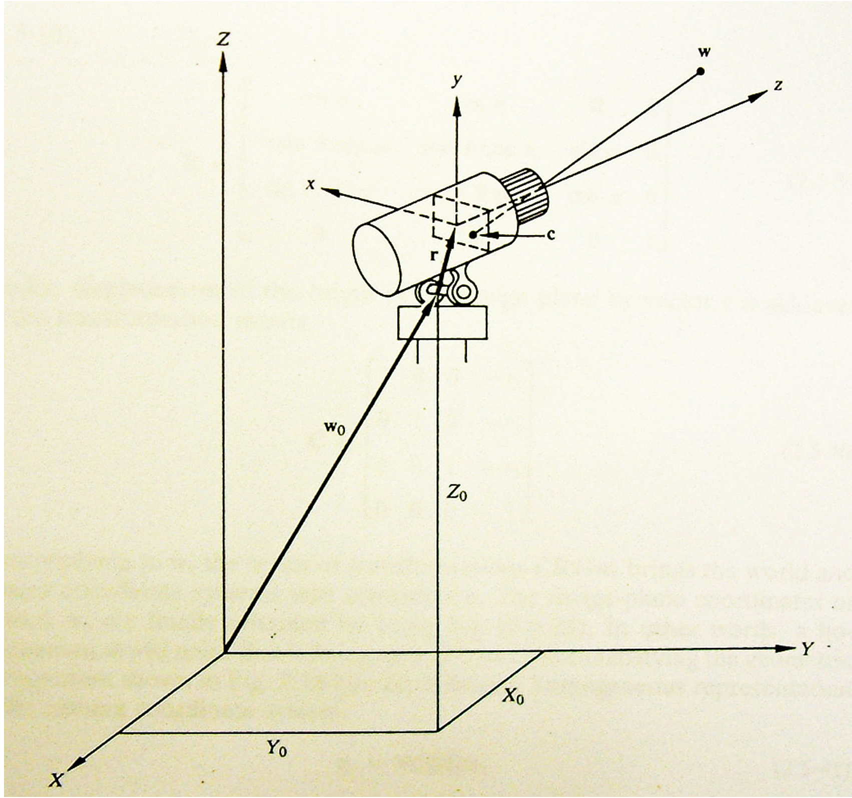
- Commonly used form for canonical view

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & p_x \\ 0 & m_y f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

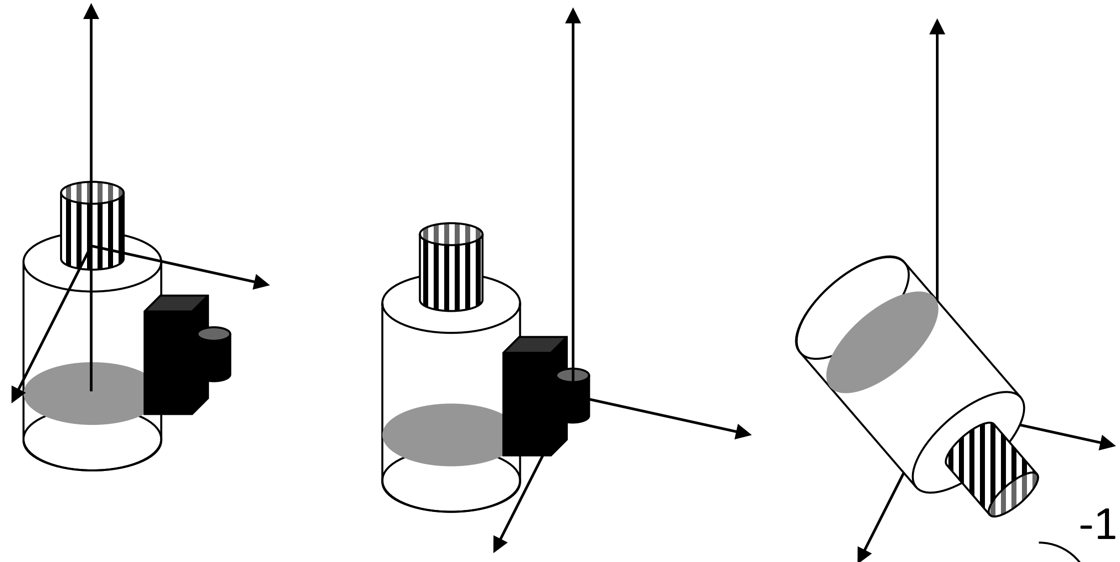
$$h\bar{\mathbf{x}} = \mathbf{K}\left[\mathbf{I}_{3\times 3} | \mathbf{0}_{3\times 1}\right]\mathbf{X}$$

Camera Model Example

- Think that the camera was originally at the origin looking down Z axis
- Then it was translated by $(r_1, r_2, r_3)^T$, rotated by ϕ along X, θ along Z, then translated by $(x_0, y_0, z_0)^T$
- This is the scenario in the figure on right



Camera Model Example



$$
\begin{bmatrix} m_x f & 0 & p_x & 0 \\ 0 & m_y f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\left(
\begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & r_1 \\ 0 & 1 & 0 & r_2 \\ 0 & 0 & 1 & r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\right)^{-1}
$$

$$
\boxed{
\begin{bmatrix} m_x f & 0 & p_x & 0 \\ 0 & m_y f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
}
$$

Camera Model Example

$$x = f\frac{(X - X_0)\cos\theta + (Y - Y_0)\sin\theta - r_1}{-(X - X_0)\sin\theta\sin\phi + (Y - Y_0)\cos\theta\sin\phi - (Z - Z_0)\cos\phi + r_3 + f}$$

$$y = f\frac{-(X - X_0)\sin\theta\cos\phi + (Y - Y_0)\cos\theta\cos\phi + (Z - Z_0)\sin\phi - r_2}{-(X - X_0)\sin\theta\sin\phi + (Y - Y_0)\cos\theta\sin\phi - (Z - Z_0)\cos\phi + r_3 + f}$$

- This camera model is applicable in many situations
- For example, this is the typical surveillance camera scenario

Aircraft Example

OTTER	system_id
TV	sensor_type
0001	serial_number
9.40000815266640300e+08	image_time
3.81319374646961220e+01	vehicle_latitude
-7.73452318519387770e+01	vehicle_longitude
9.94965840998765880e+02	vehicle_height
9.99517117444103990e-01	vehicle_pitch
1.70162641811320900e+00	vehicle_roll
1.20701055175302940e+02	vehicle_heading
1.65896873299097480e-02	camera_focal_length
-5.36131438955725910e+01	camera_elevation
-7.23296943354670500e+00	camera_scan_angle
480	number_image_lines
640	number_image_samples

cameraMat = perspective_transform * gimbal_rotation_y * gimbal_rotation_z * gimbal_translation * vehicle_rotation_x * vehicle_rotation_y * vehicle_rotation_z * vehicle_translation ;

$$\Pi_t = \begin{bmatrix} m_x f & 0 & p_x & 0 \\ 0 & m_y f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\omega & 0 & -\sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ \sin\omega & 0 & \cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\tau & \sin\tau & 0 & 0 \\ -\sin\tau & \cos\tau & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\Delta T_x \\ 0 & 1 & 0 & -\Delta T_y \\ 0 & 0 & 1 & -\Delta T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c(1,1) = (cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*sin(v_hdg);

c(1,2) = -(cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*cos(v_pch)*cos(v_hdg);

c(1,3) = -cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll);

c(1,4) = -((cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*sin(v_hdg))*vx-(-(cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*cos(v_pch)*cos(v_hdg))*vy-(-cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll))*vz;

c(2,1) = (-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(-sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg);

c(2,2) = -(-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(-sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*cos(v_hdg);

c(2,3) = sin(c_elv)*sin(c_scn)*sin(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll);

c(2,4) = -((-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(-sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(-sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(sin(c_elv)*sin(c_scn)*sin(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll))*vz;

c(3,1) = (cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-sin(c_elv)*sin(v_pch))*sin(v_hdg);

c(3,2) = -(cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-sin(c_elv)*sin(v_pch))*cos(v_hdg);

c(3,3) = -cos(c_elv)*sin(c_scn)*sin(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*cos(v_rll);

c(3,4) = -((cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-cos(c_elv)*sin(c_scn)*sin(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*cos(v_rll))*vz;

c(4,1) = (1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg);

c(4,2) = -(1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg);

c(4,3) = -1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll);

c(4,4) = -((1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll))*vz+1;

Perspective Camera Model

Canonical View

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & p_x \\ 0 & m_y f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Surveillance Camera Example (Small gimbal translation ignored)

$$\begin{bmatrix} m_x f & 0 & p_x & 0 \\ 0 & m_y f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Aircraft Example

$$\Pi_t = \begin{bmatrix} m_x f & 0 & p_x & 0 \\ 0 & m_y f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\omega & 0 & -\sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ \sin\omega & 0 & \cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\tau & \sin\tau & 0 & 0 \\ -\sin\tau & \cos\tau & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\Delta T_x \\ 0 & 1 & 0 & -\Delta T_y \\ 0 & 0 & 1 & -\Delta T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Transform for Canonical View

Rotation needed to align camera with world axes

Translation by Inverse of Camera Center

Orthographic Projection

$$\mathbf{x} = \mathbf{K}\left[\mathbf{R}|\mathbf{T}\right]\mathbf{X}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r^1 & r^2 & r^3 & t^x \\ r^4 & r^5 & r^6 & t^y \\ r^7 & r^8 & r^9 & t^z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r^1 & r^2 & r^3 \\ r^4 & r^5 & r^6 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t^x \\ t^y \end{bmatrix}$$

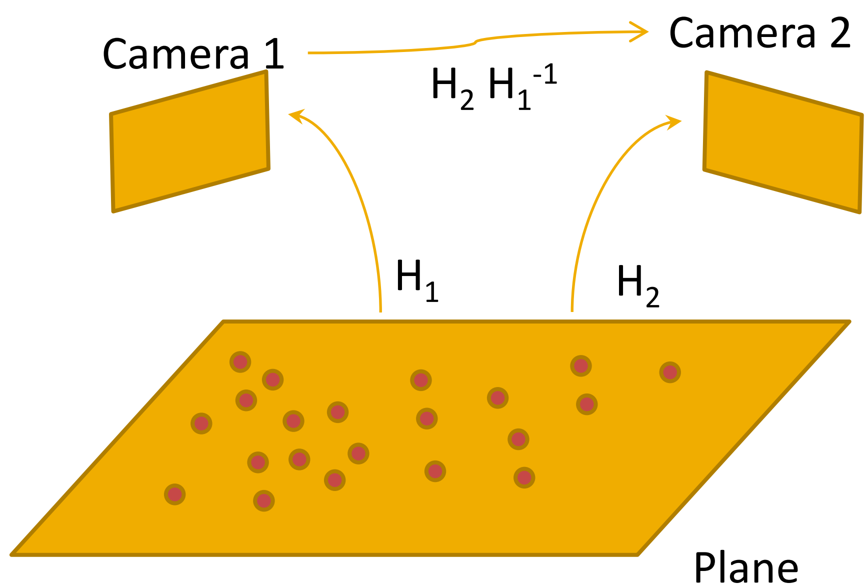
Plane + Perspective

- If plane is defined by $Z = 0$

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & p_x \\ 0 & m_y f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & -c_1 \\ r_{21} & r_{22} & r_{23} & -c_2 \\ r_{31} & r_{32} & r_{33} & -c_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad Z = 0$$

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

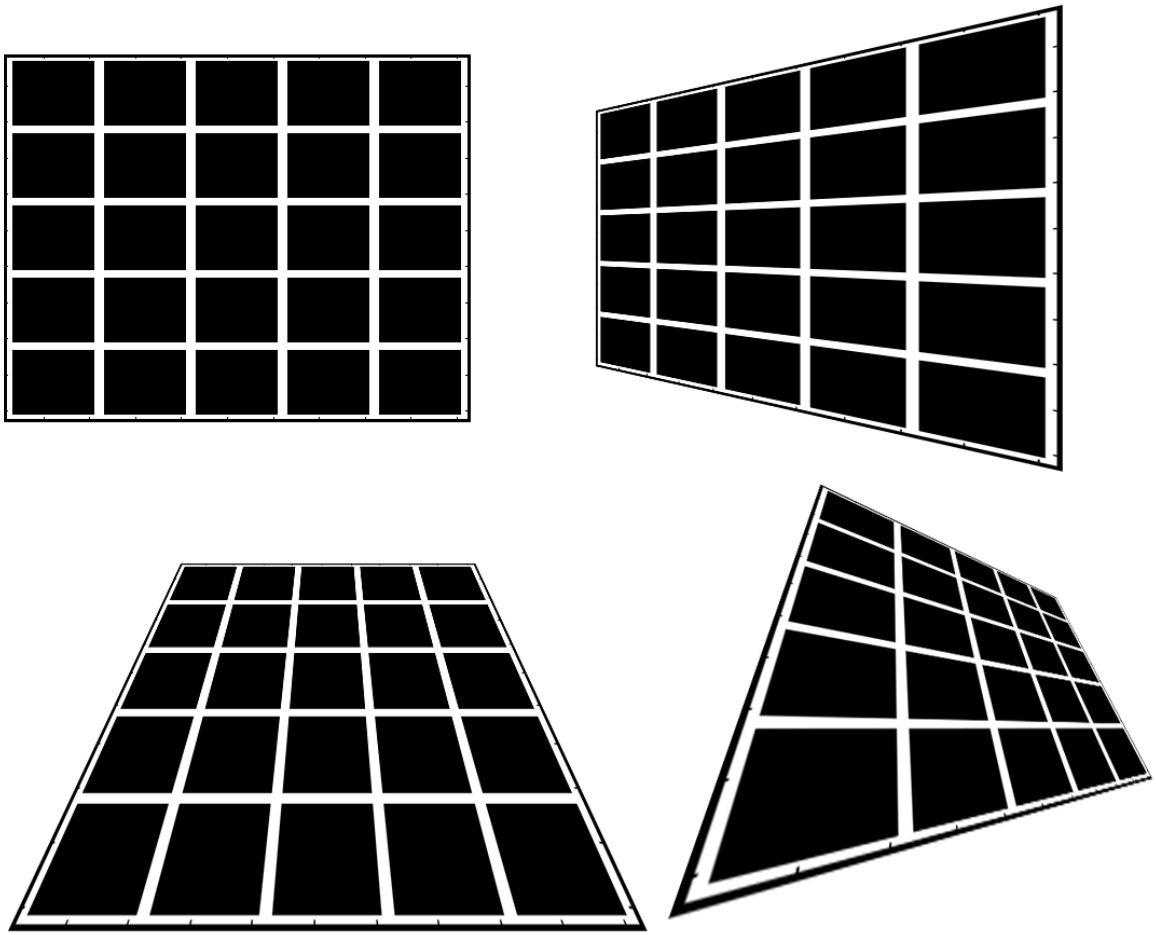
Projective Transformation between plane and image



Plane + Perspective Model

- Conclusion:
- Planar world and perspective camera yields **projective** relationship between the images
- Similarly, it can be shown that planar world and orthographic camera yields _________ relationship between images

Examples of Projective Transformations



Rotation about Camera Center (Pure Rotation)

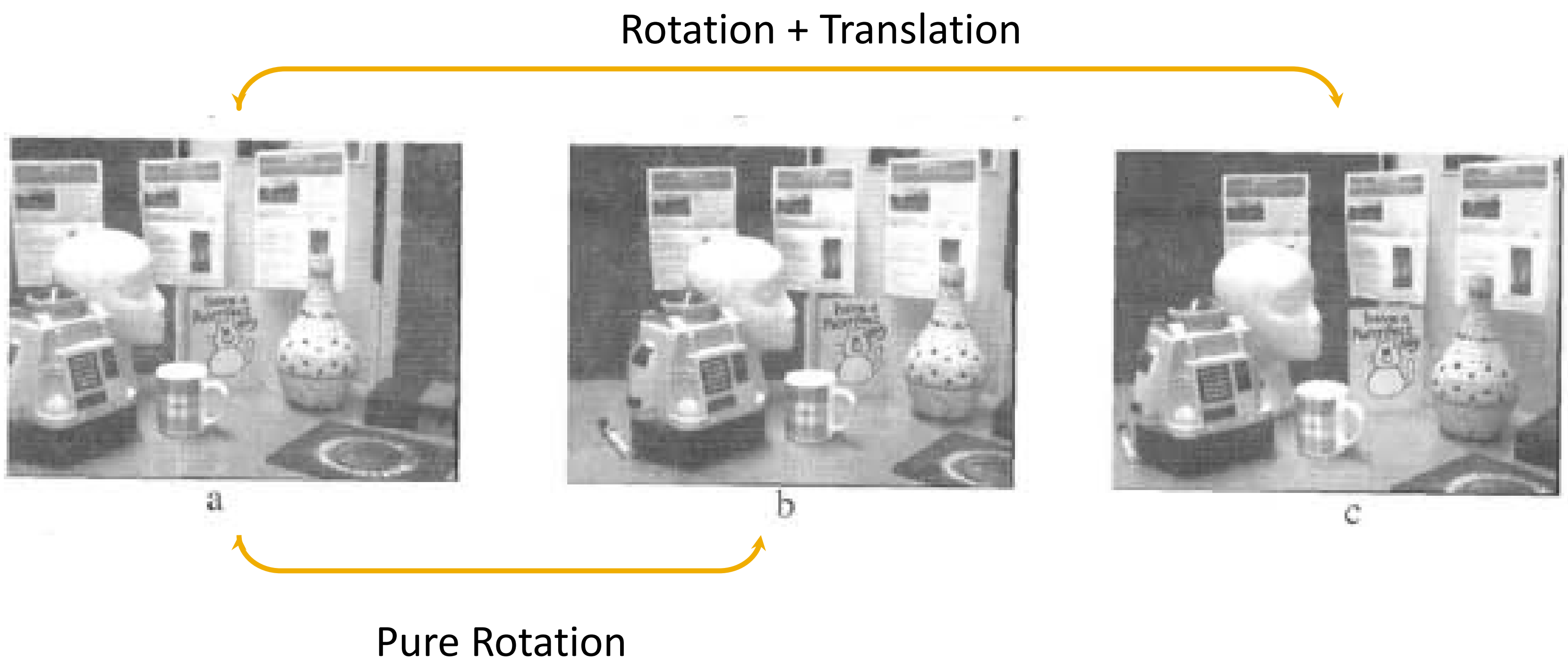
- $\mathbf{x}$ and $\mathbf{x}$' are images of a point $\mathbf{X}$ before and after rotation of the camera

$$\mathbf{x} = \mathbf{K}\,[\mathbf{I}|\mathbf{0}]\,\mathbf{X}$$

$$\mathbf{x}' \quad = \quad \mathbf{K}\,[\mathbf{R}|\mathbf{0}]\,\mathbf{X}$$

$$\mathbf{x}' = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{x} \qquad \mathbf{x}' = \mathbf{H}\mathbf{x}$$

Rotation about Camera Center (Pure Rotation)



Summary

- Pinhole Camera
  - Canonical View

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

    - Proof by similar triangles
  - General View

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & p_x \\ 0 & m_y f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \mid \mathbf{T}]\mathbf{X}$$

- Orthographic Camera

$$x = X \qquad y = Y$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

    - Parallel lines are preserved
    - Good approximation when perspective distortion is not visible (camera much far away than variation in depth)
- 2 images of a plane from pinhole camera are related by projective transformation
- 2 images of a plane from orthographic camera are related by affine transformation
- Images under pure rotation are related by a homography

Camera Calibration

- To relate 3D world points to 2D camera points, we need to know a lot of things about the camera
  - Camera Location X,Y,Z
  - Camera orientation α, β
  - Gimbal vector $(r_1, r_2, r_3)^T$
  - Focal length f
  - Size of CCD array
  - Center of projection
- Intrinsic parameters: internal to the camera.
  - Do not change when camera is moved
- Extrinsic parameters: External to the camera.
  - Change when camera is moved

Camera Calibration

- In general, the camera model looks like:

$$\begin{bmatrix} x_h \\ y_h \\ h \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{AX}$$

- Calibration is the process of finding the parameters $[a_{11} \ldots a_{34}]$
- If **x** and **X** are known, then we can solve for the unknown parameters in **A**

Camera Calibration

$$\begin{bmatrix} x_h \\ y_h \\ h \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$x = \frac{x_h}{h} = \frac{a_{11}X + a_{12}Y + a_{13}Z + a_{14}}{a_{31}X + a_{32}Y + a_{33}Z + a_{34}} \qquad y = \frac{y_h}{h} = \frac{a_{21}X + a_{22}Y + a_{23}Z + a_{24}}{a_{31}X + a_{32}Y + a_{33}Z + a_{34}}$$

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - a_{31}Xx - a_{32}Yx - a_{33}Zx - a_{34}x = 0$$

$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - a_{31}Xy - a_{32}Yy - a_{33}Zy - a_{34}y = 0$$

Camera Calibration

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - a_{31}Xx - a_{32}Yx - a_{33}Zx - a_{34}x = 0$$

$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - a_{31}Xy - a_{32}Yy - a_{33}Zy - a_{34}y = 0$$

- These equations have 12 unknowns
- Each correspondence between a world point and an image point yields two equations
- If 6 correspondences are known, we can solve for the unknowns

Camera Calibration

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - a_{31}Xx - a_{32}Yx - a_{33}Zx - a_{34}x = 0$$

$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - a_{31}Xy - a_{32}Yy - a_{33}Zy - a_{34}y = 0$$

- Separating out the knowns and the unknowns

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{31} \\ a_{32} \\ a_{33} \\ a_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Camera Calibration

- Given *n* correspondences…

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{31} \\ a_{32} \\ a_{33} \\ a_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{Ca} = \mathbf{0}$$

Camera Calibration

- This system **Ca** = **0** is a homogeneous system.
- **C** is rank deficient: rank(**C**) = 11
- Has multiple solutions (other than the trivial solution)…
  Can be solved uniquely only up to a scale factor

- Solution?

Solving for P

- The null space of **C** represents the **a** which are the solutions to the system **Ca** = 0
- How to find null space?
  1. null(C) in MATLAB
  2. Take SVD of C, as svd(C) = USV^T. The column of V corresponding to the singular value of zero represents the solution

  (in practice, you will have to take the smallest eigen value)

Camera Calibration: Summary

- Given a set of world points (in 3D coordinates) and their corresponding image points, we solve for the 3x4 camera matrix that relates them
- This transforms into a problem of the form $\mathbf{Ca} = \mathbf{0}$, which can be solved by finding the null vector of **C**.

Camera Calibration: Solving for Extrinsic and Intrinsic Parameters

- After finding **a**, we end up with a 3x4 camera matrix relating world points to image points

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

- Can I find camera rotation, translation and intrinsic parameters?

$$\mathbf{x} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]\mathbf{X}$$

- We want to decompose **P** (that we have solved through calibration) into its components

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}|-\mathbf{C}]$$

Camera Calibration: Solving for Extrinsic and Intrinsic Parameters

- Solving for Camera Center $\mathbf{C}$:
  - Note that $\mathbf{PC} = \mathbf{KR[I|\text{-}C]C} = \mathbf{0}$
  - Hence $\mathbf{C}$ is a null vector of $\mathbf{P}$ (solve by SVD)
- Solving for $\mathbf{K}$ and $\mathbf{R}$
  - Note that $\mathbf{P = [KR \mid \text{-}KRC]}$
  - Hence first 3x3 block of $\mathbf{P}$ i.e. $\mathbf{P}_{3x3} = \mathbf{KR}$
  - $\mathbf{K}$ is an upper triangular matrix, $\mathbf{R}$ is an orthonormal matrix
  - Solved through RQ decomposition [RQ decomposition decomposes a matrix into an upper triangular matrix times an orthonormal matrix