CS 4495 Computer Vision

Calibration and Projective Geometry (1)

Aaron Bobick
School of Interactive Computing

Administrivia

- Problem set 2:
  - What is the issue with finding the PDF????
    http://www.cc.gatech.edu/~afb/classes/CS4495-Fall2013/
    or
    http://www.cc.gatech.edu/~afb/classes/CS4495-Fall2013/ProblemSets/PS2/ps2-descr.pdf
- Today: Really using homogeneous systems to represent projection. And how to do calibration.
- Forsyth and Ponce, 1.2 and 1.3

Last time…

What is an image?

- Last time: a function – a 2D pattern of intensity values
- This time: a 2D projection of 3D points

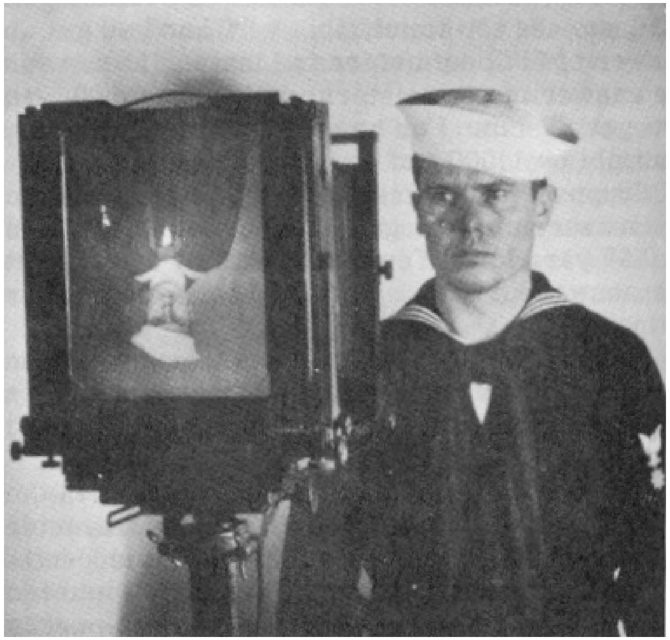
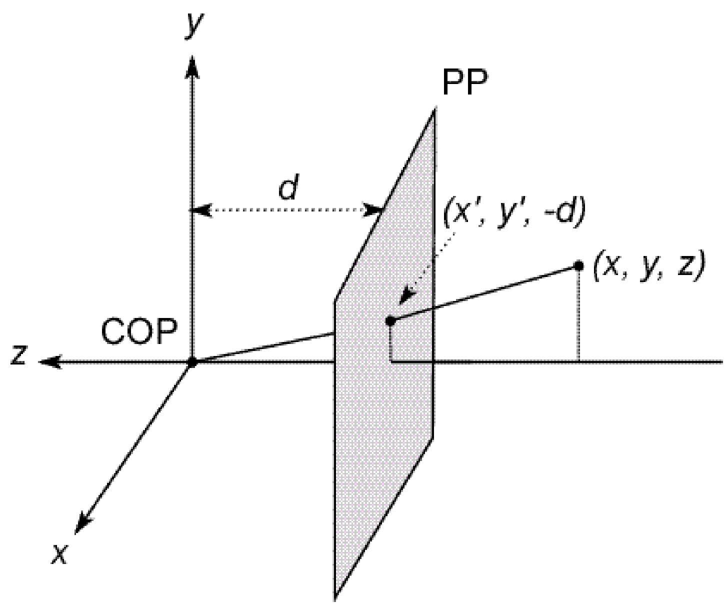


Figure from US Navy Manual of Basic Optics and Optical Instruments, prepared by Bureau of Naval Personnel. Reprinted by Dover Publications, Inc., 1969.

Modeling projection



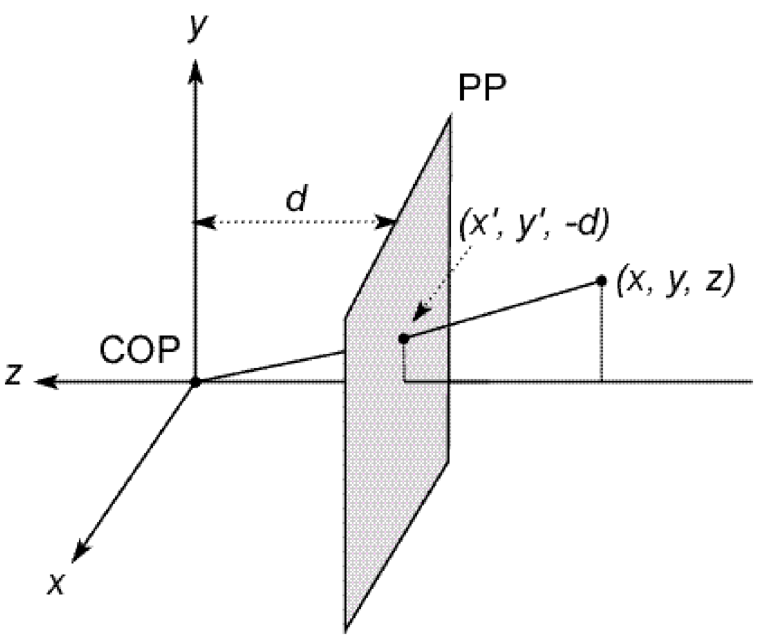
- The coordinate system
  - We will use the pin-hole model as an approximation
  - Put the optical center (**C**enter **O**f **P**rojection) at the origin
  - Put the image plane (**P**rojection **P**lane) *in front* of the COP
    - Why?
  - The camera looks down the *negative* z axis
    - we need this if we want right-handed-coordinates

Modeling projection

- Projection equations
  - Compute intersection with PP of ray from (x,y,z) to COP
  - Derived using similar triangles

Distant objects are smaller

$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z},\ -d)$$



- We get the projection by throwing out the last coordinate:

$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z})$$

Or…

- Assuming a positive focal length, and keeping *z* the distance:

$$x' = u = f\frac{x}{|z|}$$

$$y' = v = f\frac{y}{|z|}$$

Homogeneous coordinates

- Is this a linear transformation?
  - No – division by Z is non-linear

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image (2D) coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene (3D) coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Homogenous coordinates invariant under scale

Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} \quad \Rightarrow \left(f\frac{x}{z}, f\frac{y}{z} \right)$$

$$\Rightarrow (u, v)$$

This is known as perspective projection

- The matrix is the projection matrix
- The matrix is only defined up to a scale

S. Seitz

Geometric Camera calibration

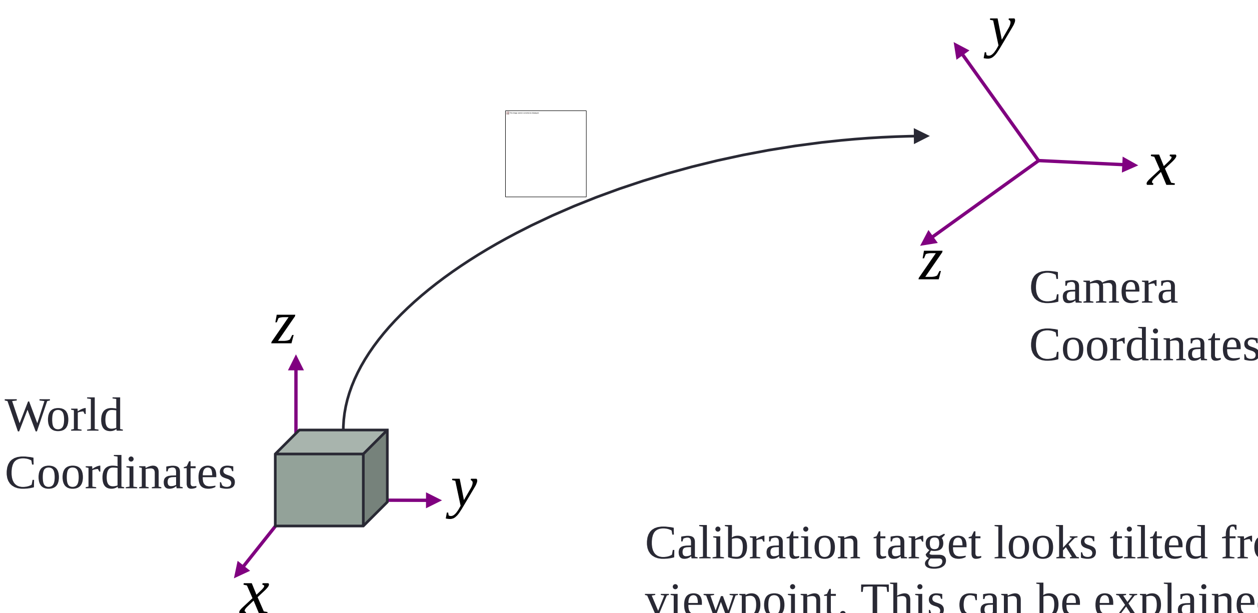
Use the camera to tell you things about the world:

- Relationship between coordinates in the world and coordinates in the image: *geometric camera calibration, see* Forsyth and Ponce, 1.2 and 1.3. Also, Szeliski section 5.2, 5.3 for references

- Made up of 2 transformations:
  - From some (arbitrary) world coordinate system to the camera's 3D coordinate system. ***Extrinisic parameters*** *(camera pose)*
  - From the 3D coordinates in the camera frame to the 2D image plane via projection. ***Intrinisic paramters***

Camera Pose

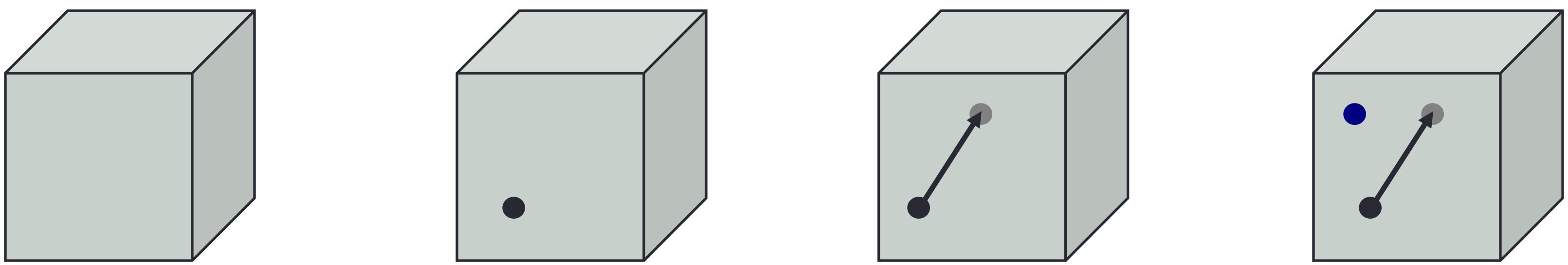
In order to apply the camera model, objects in the scene must be expressed in *camera coordinates*.



Calibration target looks tilted from camera viewpoint. This can be explained as a difference in coordinate systems.

Rigid Body Transformations

- Need a way to specify the six degrees-of-freedom of a rigid body.
- Why are their 6 DOF?



A rigid body is a collection of points whose positions relative to each other can't change

Fix one point, three DOF

Fix second point, two more DOF (must maintain distance constraint)

Third point adds one more DOF, for rotation around line

3

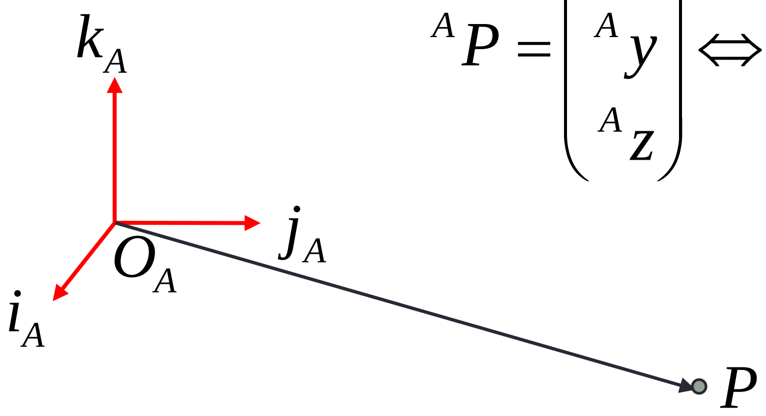
+2

+1

Notations (from F&P)

- Superscript references coordinate frame
- ${}^{A}P$ is coordinates of P in frame A
- ${}^{B}P$ is coordinates of P in frame B

$$ {}^{A}P = \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix} \Leftrightarrow \overline{OP} = \left({}^{A}x \bullet \overline{i_A}\right) + \left({}^{A}y \bullet \overline{j_A}\right) + \left({}^{A}z \bullet \overline{k_A}\right) $$

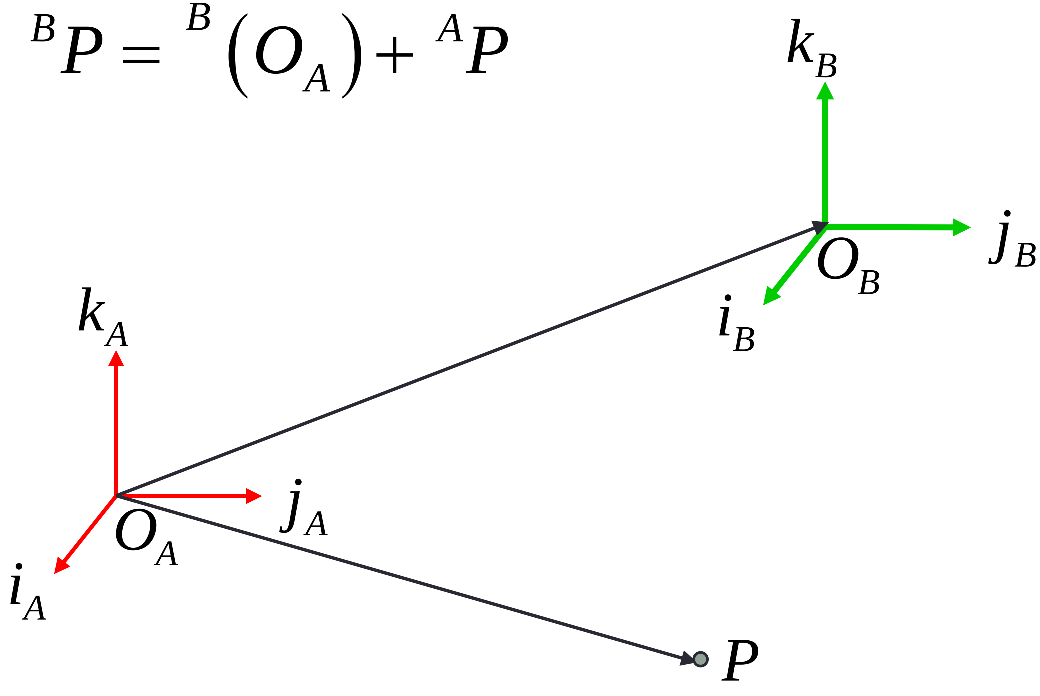


Translation Only

$$^{B}P = {}^{A}P + {}^{B}\left(O_A\right)$$

or

$$^{B}P = {}^{B}\left(O_A\right) + {}^{A}P$$



Translation

- Using homogeneous coordinates, translation can be expressed as a matrix multiplication.

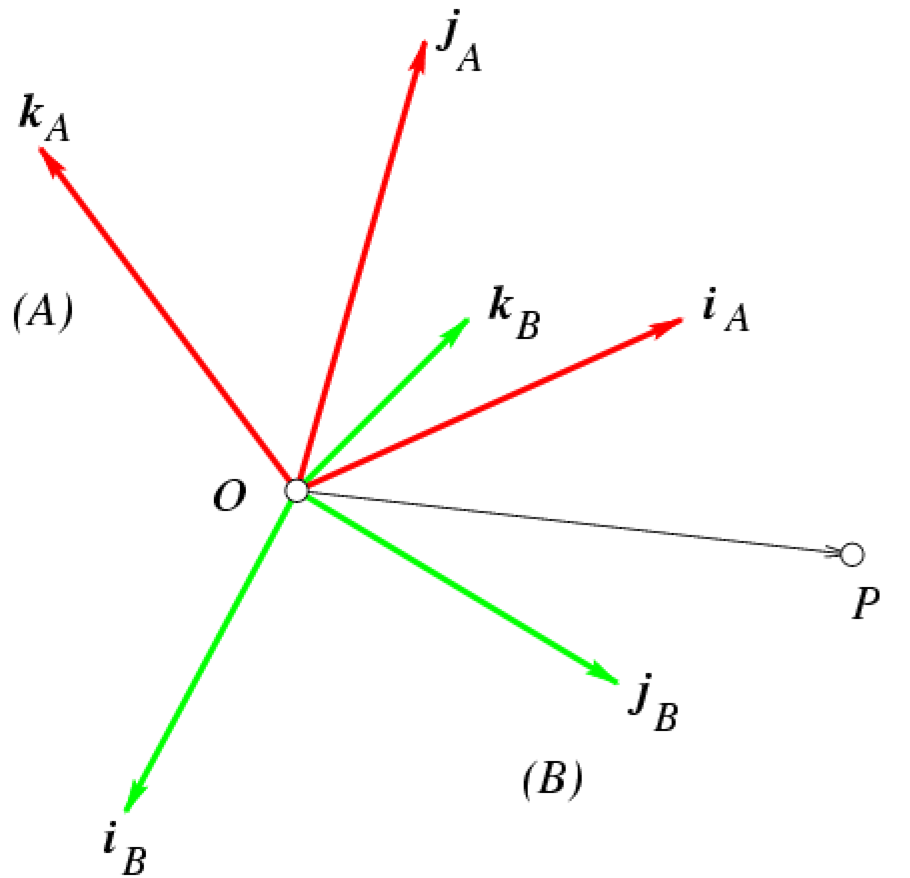
$$^{B}P = {}^{A}P + {}^{B}O_{A}$$

$$\begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} I & {}^{B}O_{A} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$$

- Translation is commutative

Rotation

$$\overrightarrow{OP} = \begin{pmatrix} i_A & j_A & k_A \end{pmatrix} \begin{pmatrix} {}^A x \\ {}^A y \\ {}^A z \end{pmatrix} = \begin{pmatrix} i_B & j_B & k_B \end{pmatrix} \begin{pmatrix} {}^B x \\ {}^B y \\ {}^B z \end{pmatrix}$$



$${}^B P = {}^B_A R \, {}^A P$$

${}^B_A R$ means describing frame A in The coordinate system of frame B

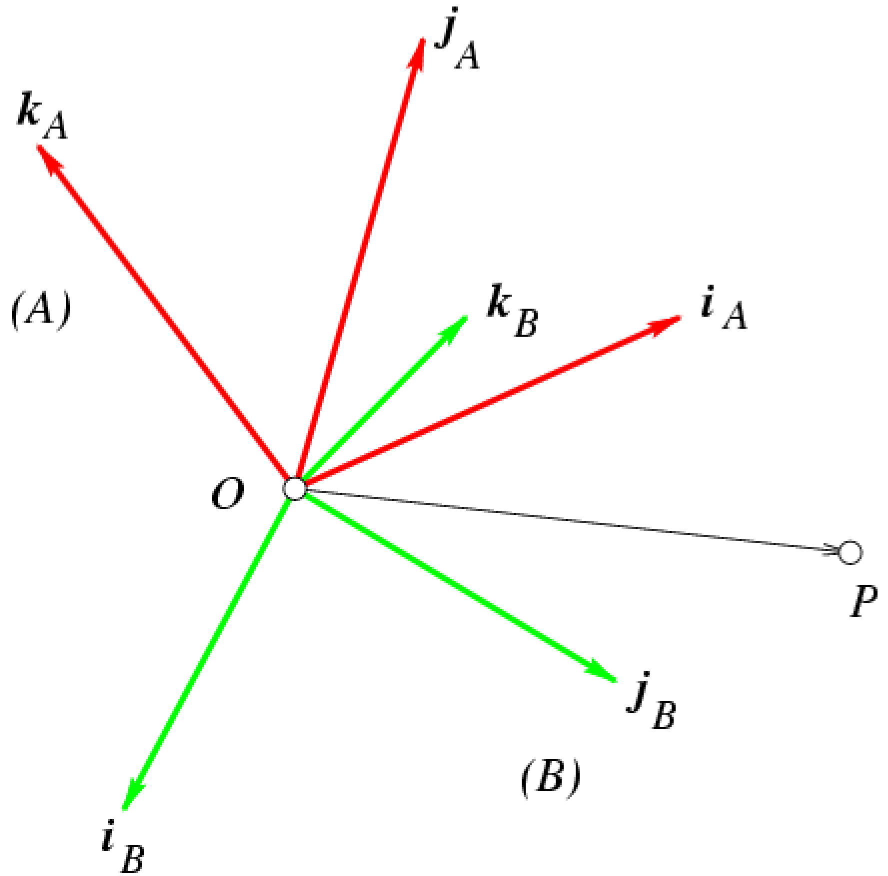
Rotation

$$ {}^{B}_{A}R = \begin{bmatrix} \mathbf{i}_A.\mathbf{i}_B & \mathbf{j}_A.\mathbf{i}_B & \mathbf{k}_A.\mathbf{i}_B \\ \mathbf{i}_A.\mathbf{j}_B & \mathbf{j}_A.\mathbf{j}_B & \mathbf{k}_A.\mathbf{j}_B \\ \mathbf{i}_A.\mathbf{k}_B & \mathbf{j}_A.\mathbf{k}_B & \mathbf{k}_A.\mathbf{k}_B \end{bmatrix} $$

$$ = \begin{bmatrix} {}^{B}\mathbf{i}_A & {}^{B}\mathbf{j}_A & {}^{B}\mathbf{k}_A \end{bmatrix} $$

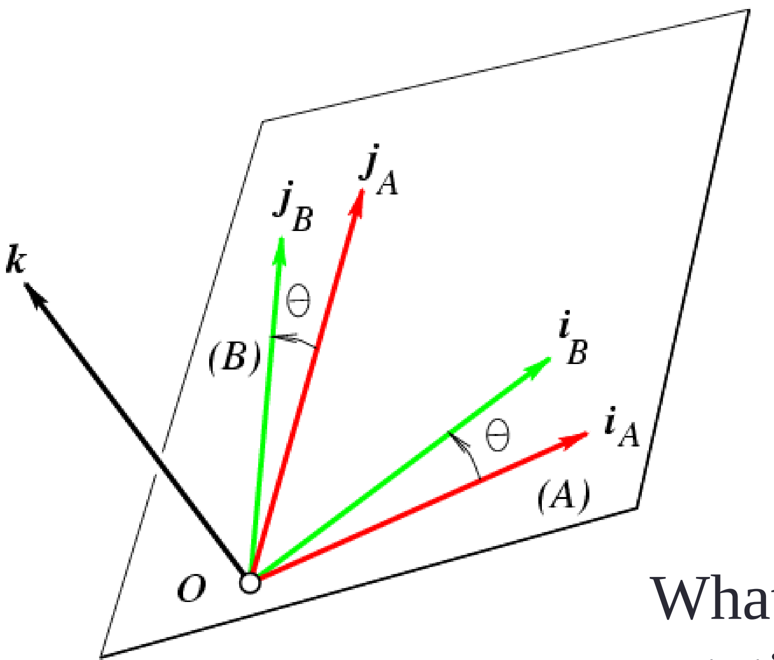
$$ = \begin{bmatrix} {}^{A}\mathbf{i}_B^T \\ {}^{A}\mathbf{j}_B^T \\ {}^{A}\mathbf{k}_B^T \end{bmatrix} $$

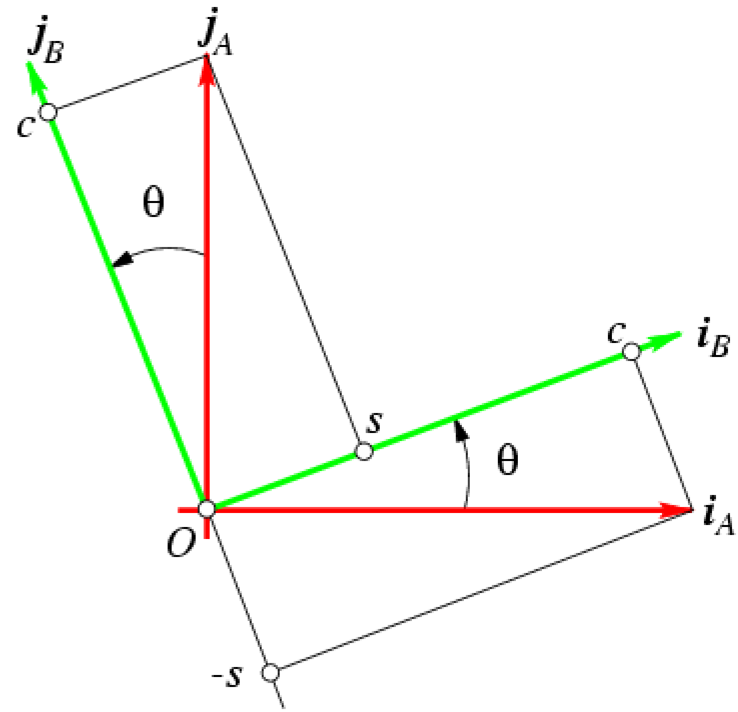
Orthogonal matrix!



The columns of the rotation matrix are the axes of frame A expressed in frame B. Why?

Example: Rotation about z axis





What is the rotation matrix?

$$R_Z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Combine 3 to get arbitrary rotation

- Euler angles: Z, X', Z''
- Heading, pitch roll: world Z, new X, new Y
- Three basic matrices: order matters, but we'll not focus on that

$$R_Z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$R_Y(\kappa) = \begin{bmatrix} \cos(\kappa) & 0 & -\sin(\kappa) \\ 0 & 1 & 0 \\ \sin(\kappa) & 0 & \cos(\kappa) \end{bmatrix}$$

Rotation in homogeneous coordinates

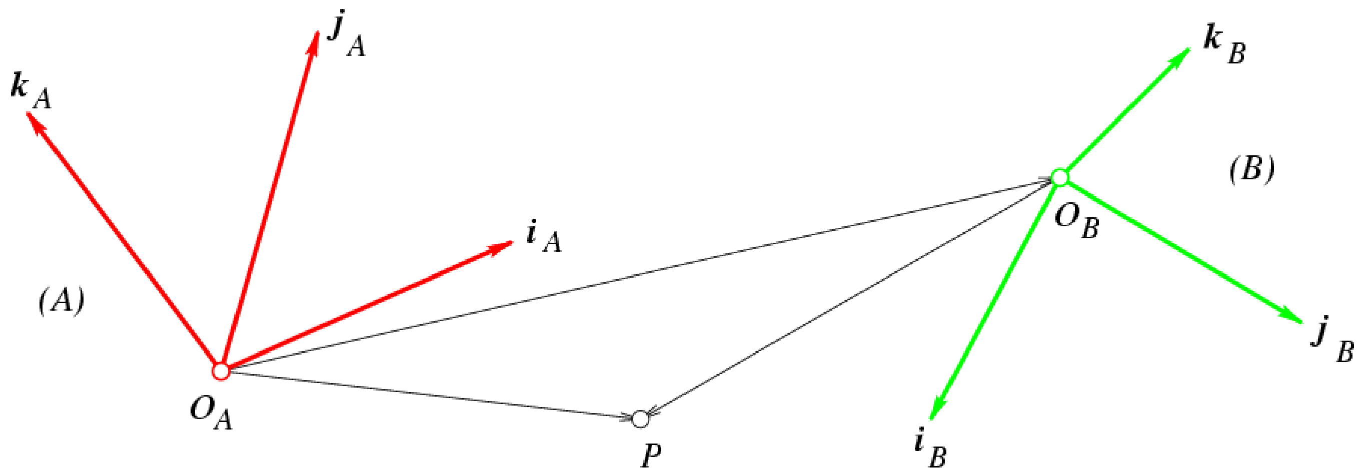
- Using homogeneous coordinates, rotation can be expressed as a matrix multiplication.

$$ {}^{B}P = {}^{B}_{A}R \, {}^{A}P $$

$$ \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{B}_{A}R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} $$

- Rotation is not commutative

Rigid transformations



$$^{B}P = {}^{B}_{A}R\,{}^{A}P + {}^{B}O_{A}$$

Rigid transformations (con't)

- Unified treatment using homogeneous coordinates.

$$\begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & {}^{B}O_{A} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}_{A}R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^{B}_{A}R & {}^{B}O_{A} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$$

Invertible!

$$\longrightarrow \quad \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = {}^{B}_{A}T \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$$

Translation and rotation

From frame A to B:

Non-homogeneous ("regular) coordinates

$$^{B}\vec{p} = {}^{B}_{A}R \; {}^{A}\vec{p} + {}^{B}_{A}\vec{t}$$

3x3 rotation matrix

Homogeneous coordinates

$$^{B}\vec{p} = \begin{pmatrix} \begin{pmatrix} & & \\ & {}^{B}_{A}R & \\ & & \end{pmatrix} & \begin{matrix} | \\ {}^{B}_{A}\vec{t} \\ | \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Homogenous coordinates allows us to write coordinate transforms as a single matrix!

From World to Camera

Rotation from world to camera frame

Translation from world to camera frame

$$ {}^{C}\vec{p} = {}^{C}_{W}R \; {}^{W}\vec{p} + {}^{C}_{W}\vec{t} $$

Point in camera frame

Point in world frame

Non-homogeneous coordinates

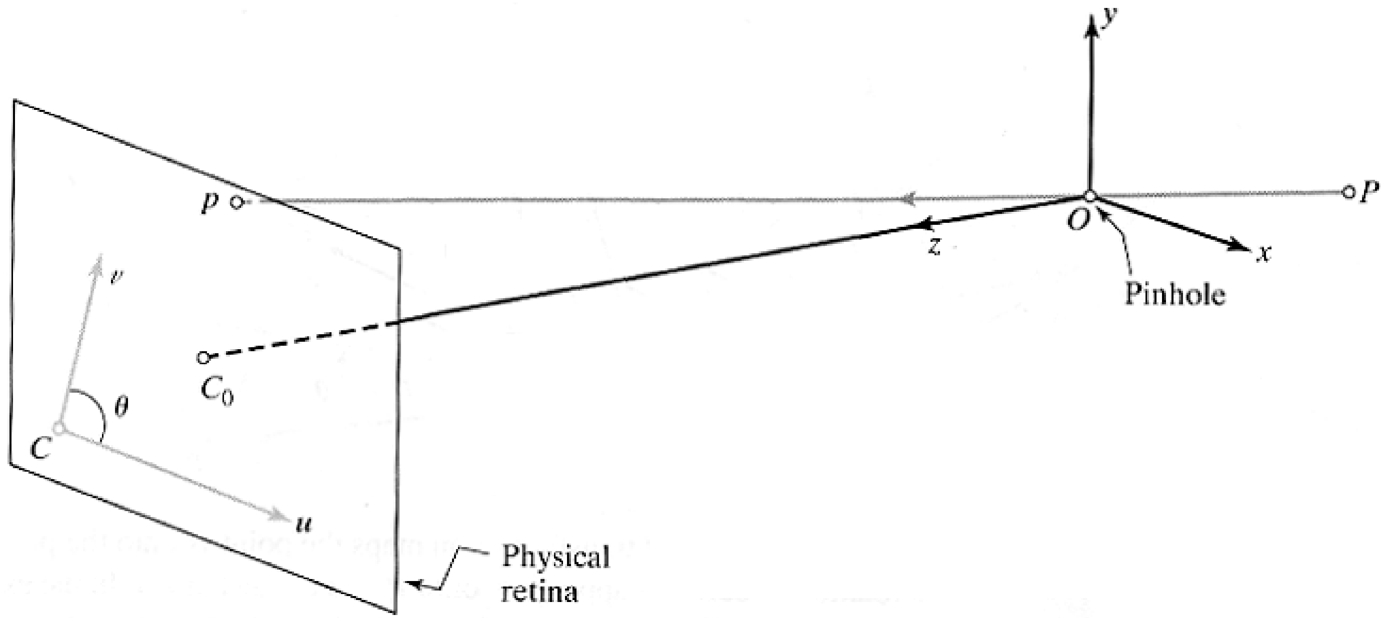
$$ \begin{pmatrix} \\ {}^{C}\vec{p} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} - & - & - & | \\ - & {}^{C}_{W}R & - & {}^{C}_{W}\vec{t} \\ - & - & - & | \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \\ {}^{W}\vec{p} \\ \\ \\ \end{pmatrix} $$

Homogeneous coordinates

From world to camera is the
extrinsic *parameter matrix (4x4)*
(sometimes 3x4 if using for next step in projection – not worrying about inversion)

Now from Camera 3D to Image…

Camera 3D (x,y,z) to 2D (u,v) or (x',y'): Ideal intrinsic parameters

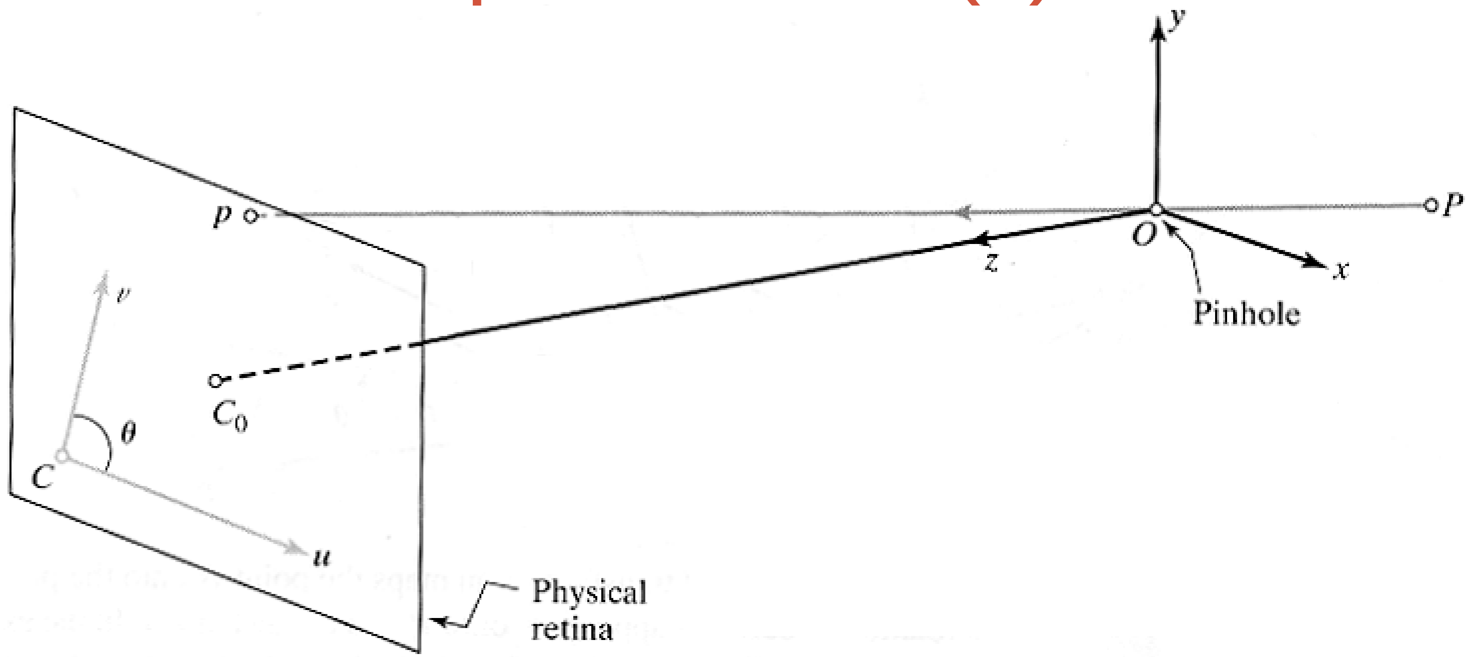


Ideal Perspective projection

$$u = f\,\frac{x}{z}$$

$$v = f\,\frac{y}{z}$$

Real intrinsic parameters (1)

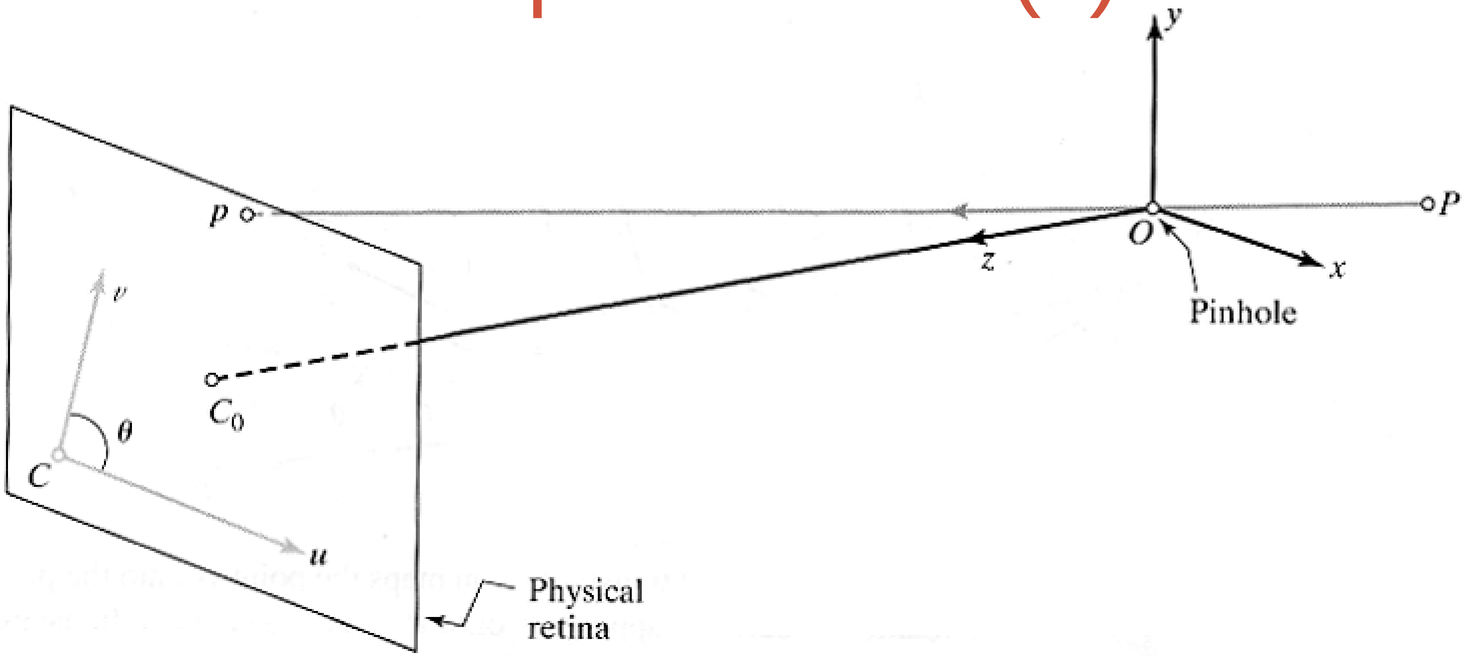


But "pixels" are in some arbitrary spatial units

$$u = \alpha \frac{x}{z}$$

$$v = \alpha \frac{y}{z}$$

Real intrinsic parameters (2)

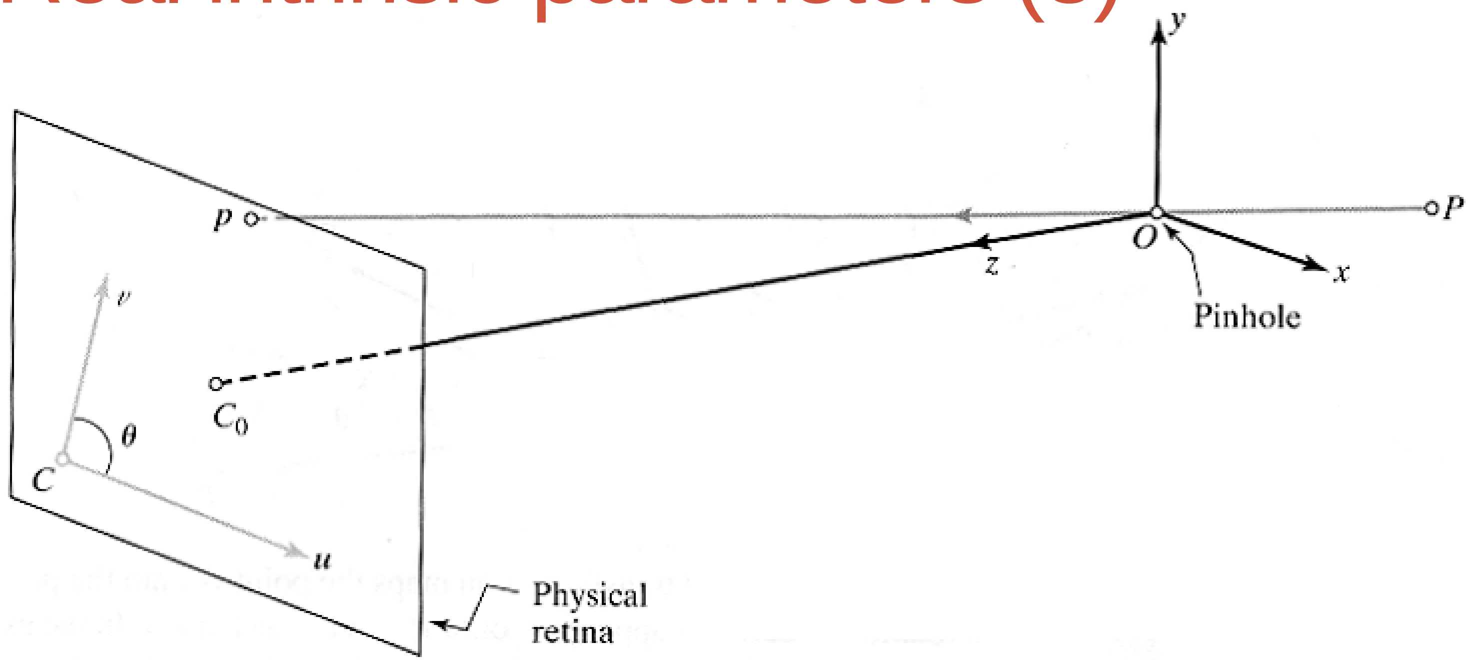


Maybe pixels are not square

$$u = \alpha \frac{x}{z}$$

$$v = \beta \frac{y}{z}$$

Real intrinsic parameters (3)

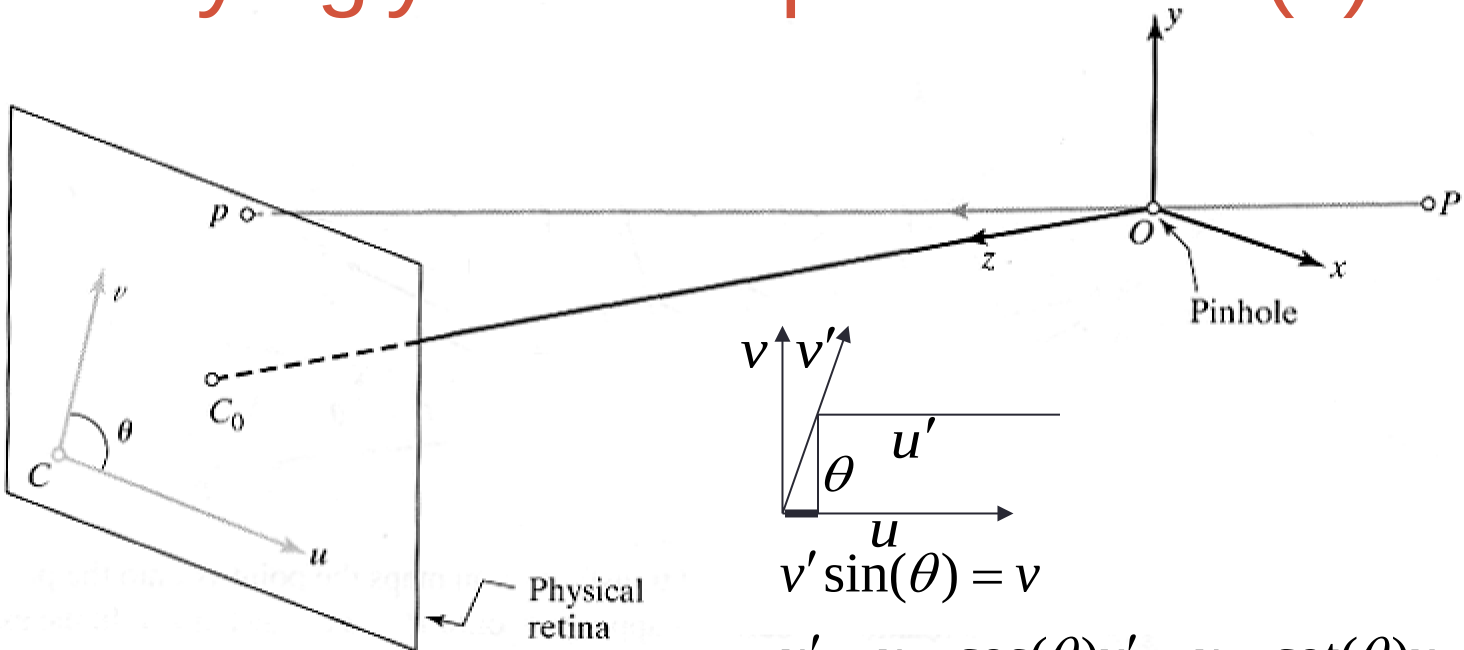


We don't know the origin of our camera pixel coordinates

$$u = \alpha \frac{x}{z} + u_0$$

$$v = \beta \frac{y}{z} + v_0$$

Really ugly intrinsic parameters (4)



$$v'\sin(\theta) = v$$

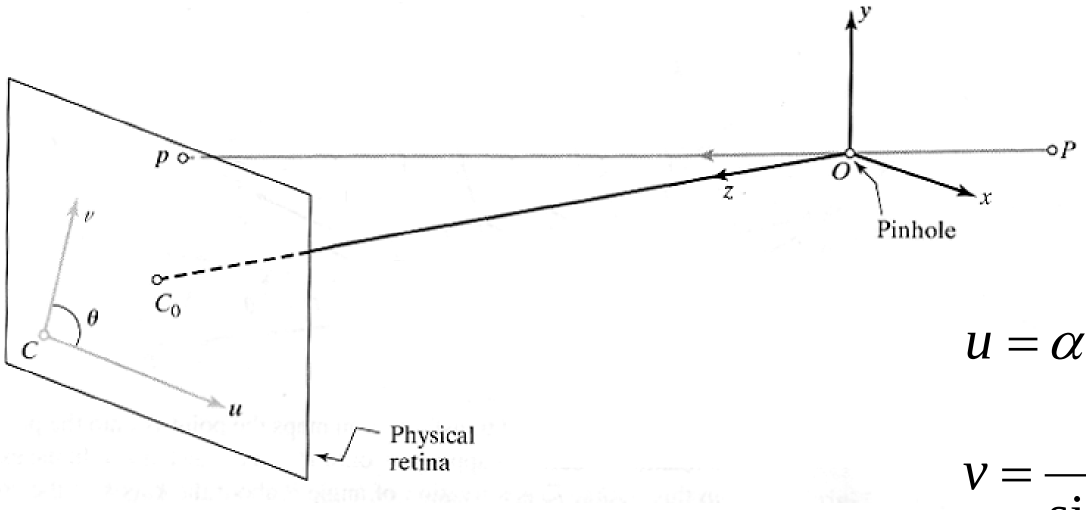
$$u' = u - \cos(\theta)v' = u - \cot(\theta)v$$

May be skew between camera pixel axes

$$u = \alpha\frac{x}{z} - \alpha\cot(\theta)\frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)}\frac{y}{z} + v_0$$

Intrinsic parameters, homogeneous coordinates



Notice division by z

$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$

Using homogenous coordinates we can write this as:

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

In homog pixels

$$\vec{p}\,' = \mathbf{K} \; {}^{C}\vec{p}$$

In camera-based 3D coords

Kinder, gentler intrinsics

- Can use simpler notation for intrinsics – last column is zero:

$$K = \begin{bmatrix} f & s & c_x \\ 0 & af & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

s – skew
a – aspect ratio
(5 DOF)

- If square pixels, no skew, and optical center is in the center (assume origin in the middle):

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

*In this case only one DOF, focal length **f***

Combining extrinsic and intrinsic calibration parameters, in homogeneous coordinates

pixels

Intrinsic

$$\vec{p}\,' = \mathrm{K}\ {}^{C}\vec{p}$$

Camera 3D coordinates

World 3D coordinates

Extrinsic

$$\begin{pmatrix} \\ {}^{C}\vec{p} \\ \\ \end{pmatrix} = \begin{pmatrix} \begin{array}{|ccc|}\hline - & - & - \\ - & {}^{C}_{W}R & - \\ - & - & - \\ \hline \end{array} & \begin{array}{|c|}\hline | \\ {}^{C}_{W}\vec{t} \\ | \\ \hline \end{array} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} \begin{pmatrix} \\ {}^{W}\vec{p} \\ \\ \end{pmatrix}$$

$$\vec{p}\,' = \underbrace{K \begin{pmatrix} {}^{C}_{W}R & {}^{C}_{W}\vec{t} \\ 0\ 0\ 0 & 1 \end{pmatrix}}\ {}^{W}\vec{p}$$

$$\vec{p}\,' = M\ {}^{W}\vec{p}$$

(If K is 3x4)

Other ways to write the same equation

pixel coordinates

world coordinates

$$\vec{p}\,' = M\ {}^{W}\vec{p}$$

Conversion back from homogeneous coordinates leads to:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \simeq \begin{pmatrix} s*u \\ s*v \\ s \end{pmatrix} = \begin{pmatrix} . & m_1^T & . & . \\ . & m_2^T & . & . \\ . & m_3^T & . & . \end{pmatrix} \begin{pmatrix} {}^{W}p_x \\ {}^{W}p_y \\ {}^{W}p_z \\ 1 \end{pmatrix}$$

projectively similar

$$\begin{cases} u = \dfrac{m_1 \cdot \vec{P}}{m_3 \cdot \vec{P}} \\[2ex] v = \dfrac{m_2 \cdot \vec{P}}{m_3 \cdot \vec{P}} \end{cases}$$

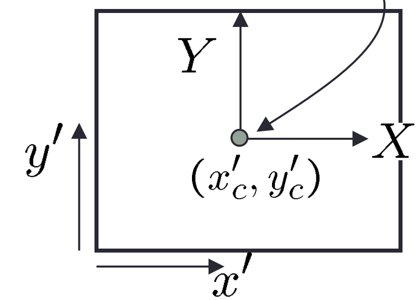
Finally: Camera parameters

A camera (and its matrix) $\mathbf{M}$ (or $\mathbf{\Pi}$) is described by several parameters

- Translation **T** of the optical center from the origin of world coords
- Rotation **R** of the image plane
- focal length f, principle point (x'_c, y'_c), pixel size (s_x, s_y)
- **blue** parameters are called "**extrinsics**," red are "intrinsics"

Projection equation

$$\mathbf{x} \simeq \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{MX}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

identity matrix

$$\mathbf{M} = \underbrace{\begin{bmatrix} f & s & x'_c \\ 0 & af & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{translation}}$$

DoFs: 5+0+3+3 = 11

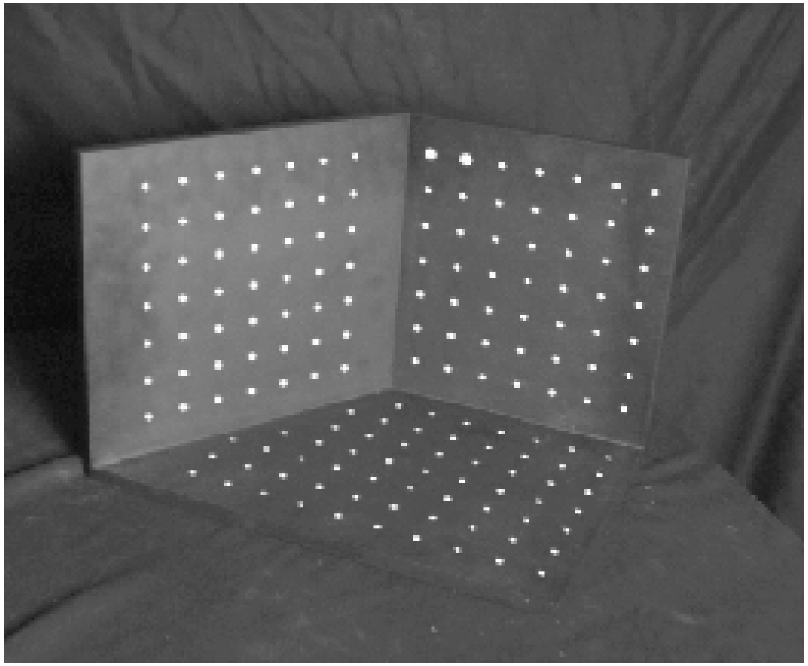
- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another

Calibration

- How to determine **M** (or $\mathbf{\Pi}$)?

Calibration using a reference object

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image

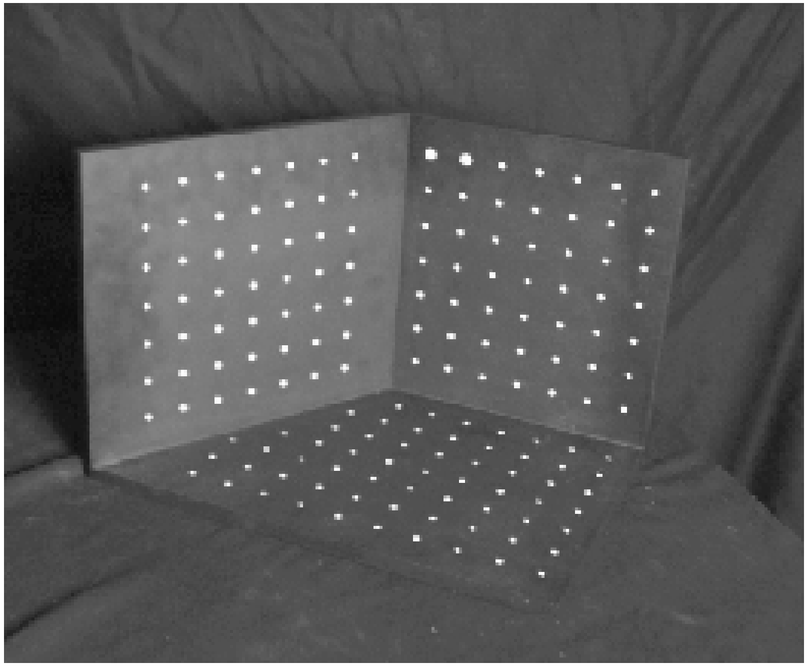


Issues

- must know geometry very accurately
- must know 3D->2D correspondence

Estimating the projection matrix

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image



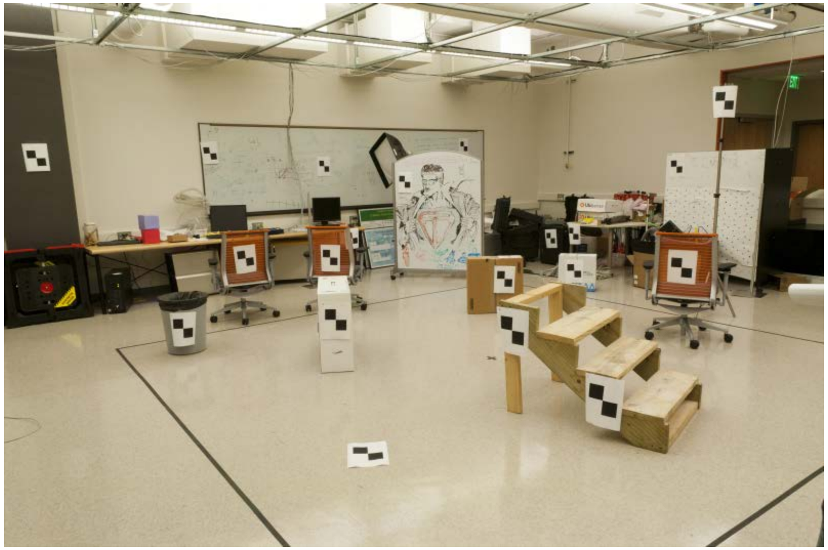
$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Resectioning – estimating the camera matrix from known 3D points

- Projective Camera Matrix:

$$p = K\begin{bmatrix} R & t \end{bmatrix} P = MP$$

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



- Only up to a scale, so 11 DOFs.

Direct linear calibration - homogeneous

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$
$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

One pair of equations for each point

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Direct linear calibration - homogeneous

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{10} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$\mathbf{A}$ 2n × 12 $\mathbf{m}$ 12 $\mathbf{0}$ 2n

This is a homogenous set of equations.

When over constrained, defines a least squares problem

– minimize $\|\mathbf{Am}\|$

- Since **m** is only defined up to scale, solve for unit vector $\mathbf{m^*}$
- Solution: $\mathbf{m^*}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with *smallest* eigenvalue
- Works with 6 or more points

The SVD (singular value decomposition) trick…

Find the $\mathbf{x}$ that minimizes $||\mathbf{Ax}||$ subject to $||\mathbf{x}|| = 1$.

Let $\mathbf{A} = \mathbf{UDV}^T$ (singular value decomposition, $\mathbf{D}$ diagonal, $\mathbf{U}$ and $\mathbf{V}$ orthogonal)

Therefor minimizing $||\mathbf{UDV}^T\mathbf{x}||$

But, $||\mathbf{UDV}^T\mathbf{x}|| = ||\mathbf{DV}^T\mathbf{x}||$ and $||\mathbf{x}|| = ||\mathbf{V}^T\mathbf{x}||$

Thus minimize $||\mathbf{DV}^T\mathbf{x}||$ subject to $||\mathbf{V}^T\mathbf{x}|| = 1$

Let $\mathbf{y} = \mathbf{V}^T\mathbf{x}$: Minimize $||\mathbf{Dy}||$ subject to $||\mathbf{y}||=1$.

But $\mathbf{D}$ is diagonal, with decreasing values. So $||\mathbf{Dy}||$ min is when

$$\mathbf{y} = (0,0,0\ldots,0,1)^T$$

Thus $\mathbf{x} = \mathbf{Vy}$ is the last column in $\mathbf{V}$. [ortho: $\mathbf{V}^T = \mathbf{V}^{-1}$]

And, the singular values of $\mathbf{A}$ are square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ and the columns of $\mathbf{V}$ are the eigenvectors. (Show this?)

Direct linear calibration - inhomogeneous

- Another approach: 1 in lower r.h. corner for 11 d.o.f

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \simeq \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Now "regular" least squares since there is a non-variable term in the equations:

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

Dangerous if m_{23} is really zero!

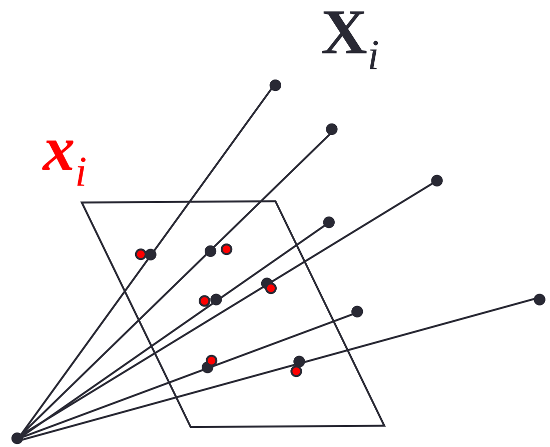
Direct linear calibration (transformation)

- Advantage:
  - Very simple to formulate and solve. Can be done, say, on a problem set
  - These methods are referred to as "algebraic error" minimization.
- Disadvantages:
  - Doesn't directly tell you the camera parameters (more in a bit)
  - Doesn't model radial distortion
  - Hard to impose constraints (e.g., known focal length)
  - Doesn't minimize the right error function

For these reasons, *nonlinear methods* are preferred

- Define error function E between projected 3D points and image positions
  - E is nonlinear function of intrinsics, extrinsics, radial distortion
- Minimize E using nonlinear optimization techniques
  - e.g., variants of Newton's method (e.g., Levenberg Marquart)

Geometric Error



Predicted Image locations

$$\text{minimize } E = \sum_i d(x_i', \hat{x}_i')$$

$$\min_{\mathbf{M}} \sum_i d(x_i', \mathbf{M}\mathbf{X}_i)$$

"Gold Standard" algorithm *(Hartley and Zisserman)*

Objective

Given n≥6 3D to 2D point correspondences $\{X_i \leftrightarrow x_i'\}$, determine the "Maximum Likelihood Estimation" of **M**

Algorithm

(i) Linear solution:

(a) (Optional) Normalization: $\tilde{\mathbf{X}}_i = \mathbf{U}\mathbf{X}_i \quad \tilde{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i$

(b) Direct Linear Transformation Minimization of geometric error: using the linear estimate as a starting point minimize the geometric error:

$$\min_{\mathbf{M}} \sum_i d(x_i', \mathbf{M}\mathbf{X}_i)$$

(ii) Denormalization: $\mathbf{M} = \mathbf{T}^{-1}\tilde{\mathbf{M}}\mathbf{U}$

Finding the 3D Camera Center from P-matrix

- Slight change in notation. Let $\mathbf{M} = [\mathbf{Q} \mid \mathbf{b}]$ (3x4) – $\mathbf{b}$ is last column of $\mathbf{M}$
- Null-space camera of projection matrix. Find **C** such that:

$$\mathbf{MC} = \mathbf{0}$$

- Proof: Let **X** be somewhere between any point **P** and **C**

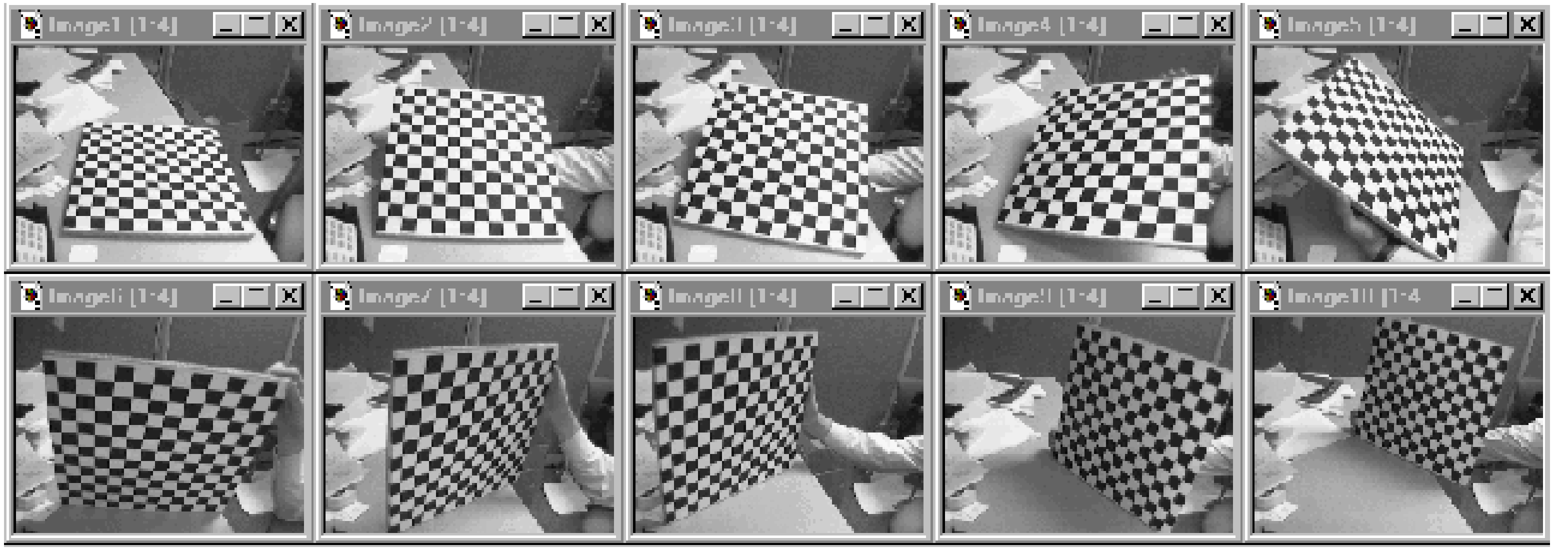
$$\mathbf{X} = \lambda\mathbf{P} + (1-\lambda)\mathbf{C}$$

$$\mathbf{x} = \mathbf{MX} = \lambda\mathbf{MP} + (1-\lambda)\mathbf{MC}$$

  - For all P, all points on PC projects on image of P,
  - Therefore C the camera center has to be in null space
- Can also be found by:

$$\mathbf{C} = \begin{pmatrix} -\mathbf{Q}^{-1}\mathbf{b} \\ 1 \end{pmatrix}$$

Alternative: multi-plane calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
  - Intel's OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang's web site: http://research.microsoft.com/~zhang/Calib/