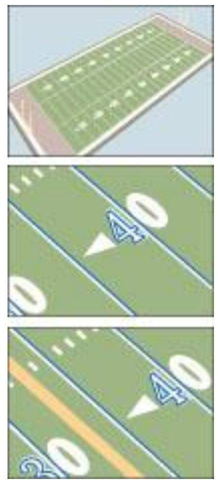
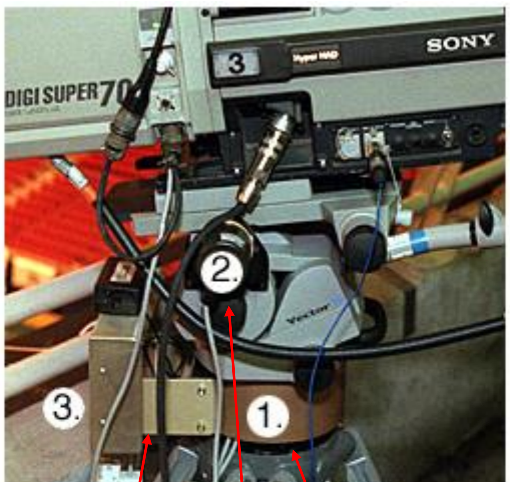
# *Camera Calibration*
# *Geometry and Radiometry*

# Motivation

- Think about the application of drawing the first-down line in football game
- Calibration steps:
  - Contour mapping
  - Camera calibration – position
- During the game:
  - Cameras: three, PTZ readings provided in real-time
  - People: spotter, line-position technician, 1st & ten operator, trouble-shooter
  - Computers: five
    - Gather PC: receive PTZ data
    - Tally: keep track of on-air camera and choose 3D map to use
    - F Ten: video display and overlay 3D map
    - Matte: pattern recognition (player/field classification)
    - Render: receives data from all other, draws the line

### Mapping Contours



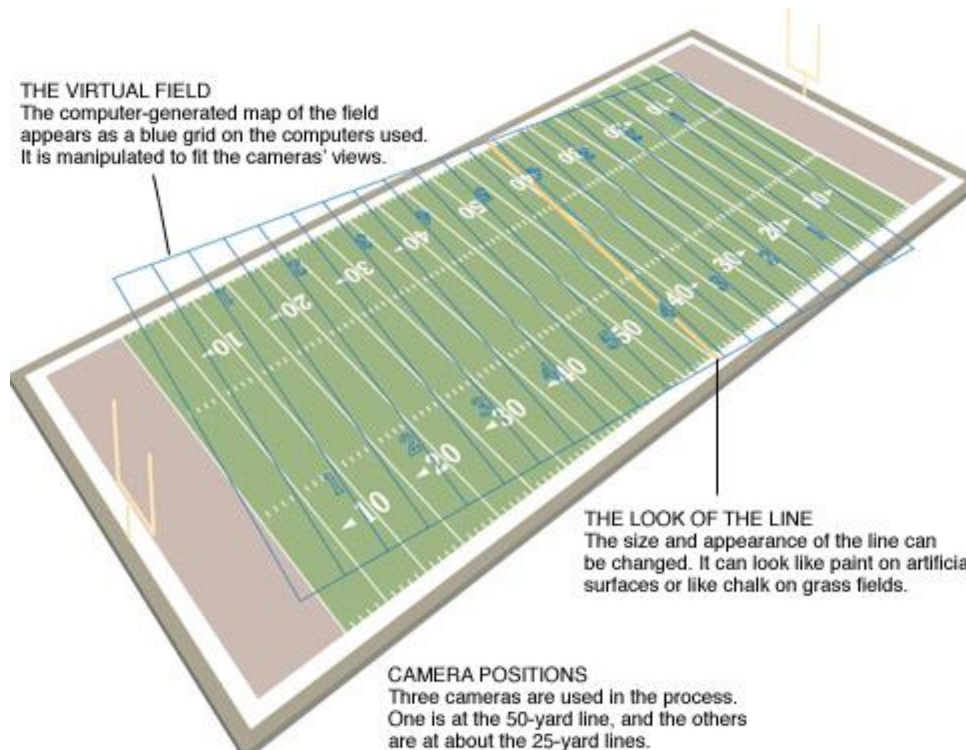



Remote

Sensor

Pan encoder

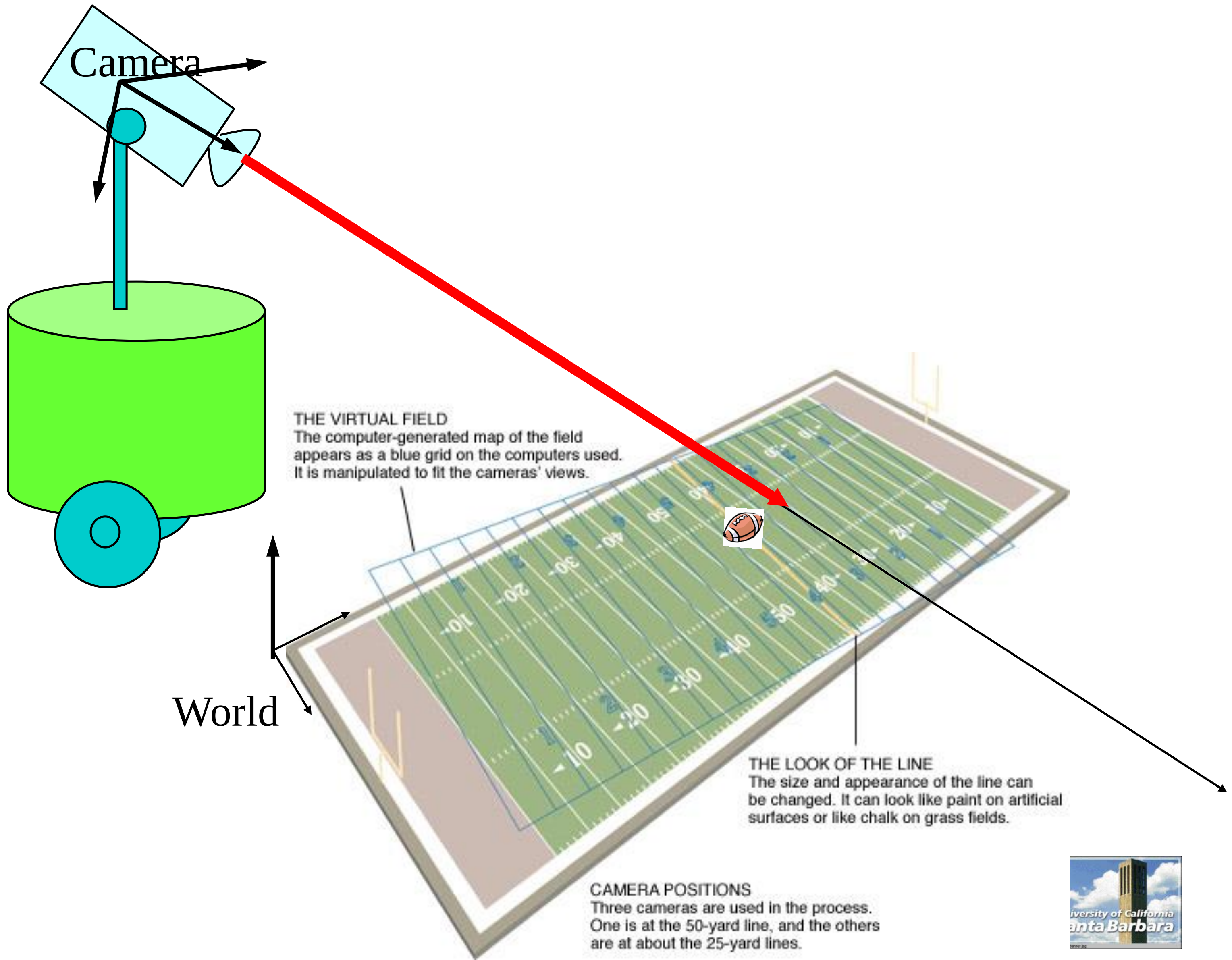
Tilt encoder



# Camera Calibration

- Static:
  - Where is a camera placed with respect to the field?
  - How is the camera aimed?
- Dynamic:
  - How is the camera aim's changed?
- In essence:
  - Camera coordinates (i,j) must be able to correlate into world coordinates (x,y,z)

# Coordinate systems



# *Other Applications*

- Autonomous navigation
- Photogrammetry and remote sensing
- Soldering and welding
- Inspection
- Almost all CV algorithms (except those deal entirely with 2D images) perform camera calibration

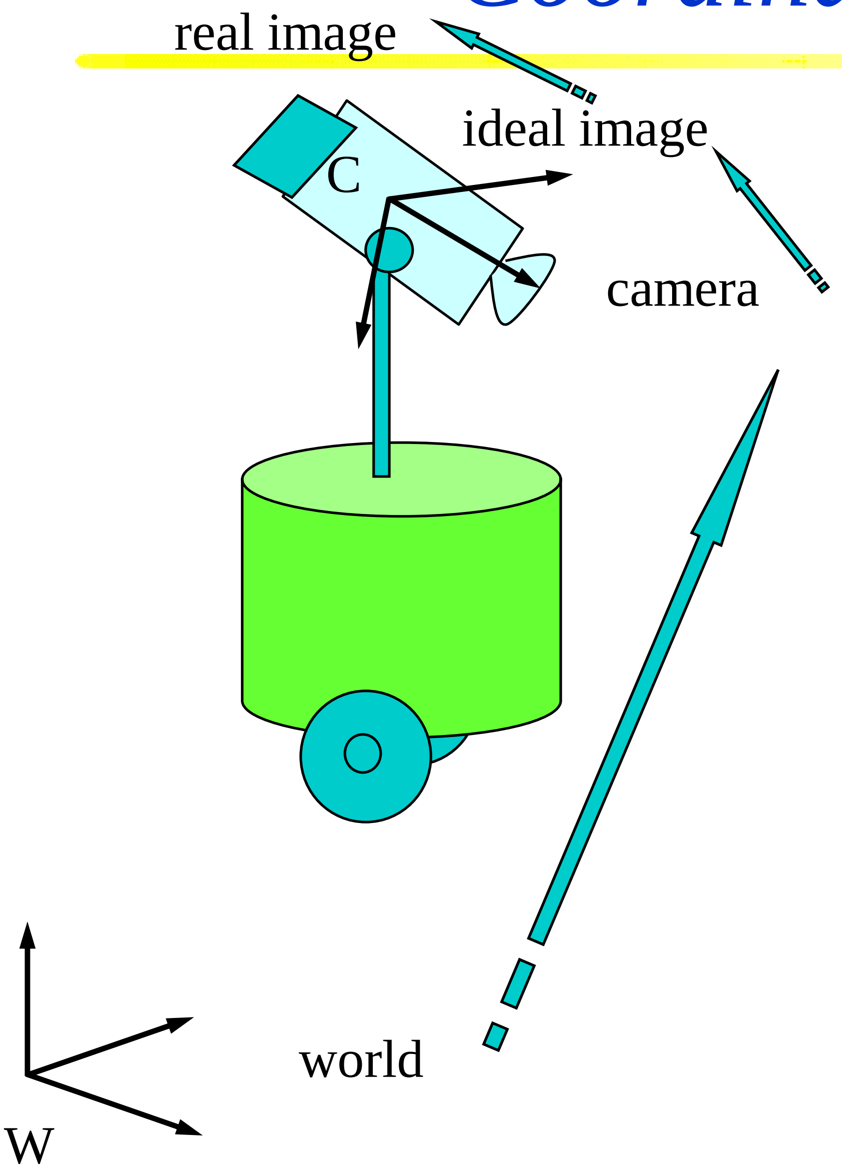
# *Calibration & Registration*

- Camera calibration
  - *Intrinsic* parameters (e.g., focal length, aspect ratio, lens distortion)
  - Independent of camera placement
- Pose registration
  - *Extrinsic* parameters
  - Independent of choice of cameras

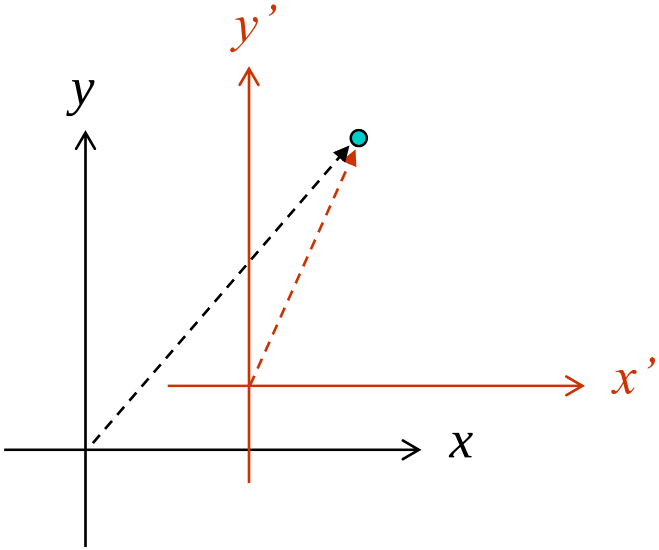
# *Mathematics of Image Formation*

- A sequence of coordinate transforms + projection (lens transform)
- 3D world coordinate
- 3D camera coordinate
- 2D ideal image coordinate
- 2D real image coordinate
- World  to camera: rigid body transform (**R** and **T**)
- Camera to ideal: ideal projection
- Ideal to real: real CCD and lens

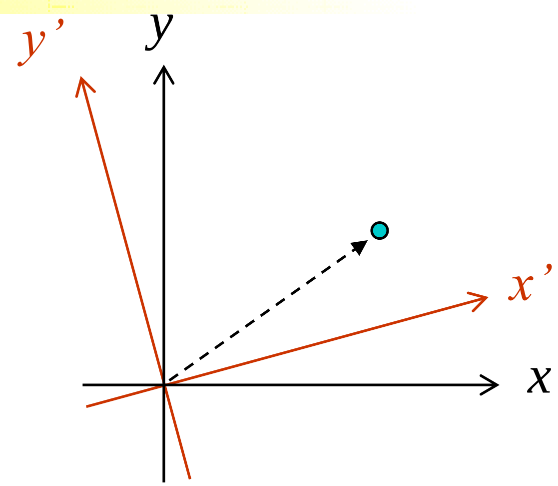
# *Coordinate systems*



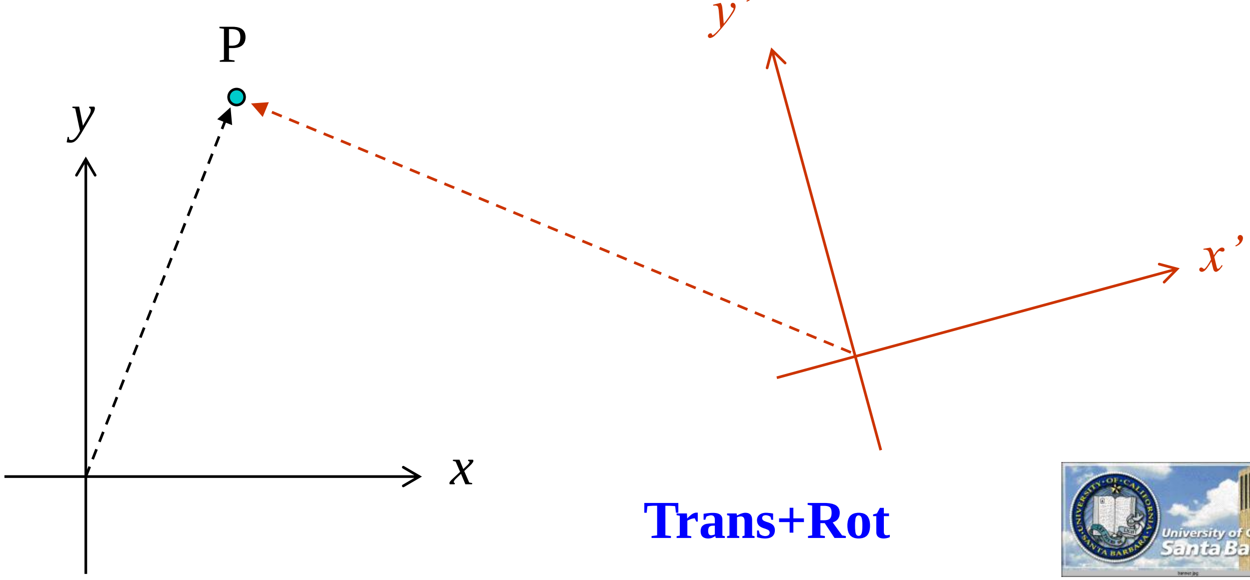

# 2D examples




**Trans**

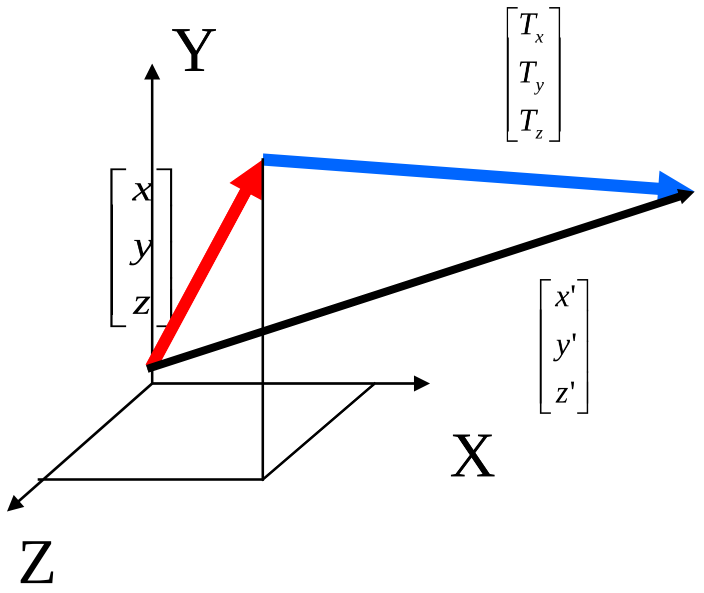



**Rot**



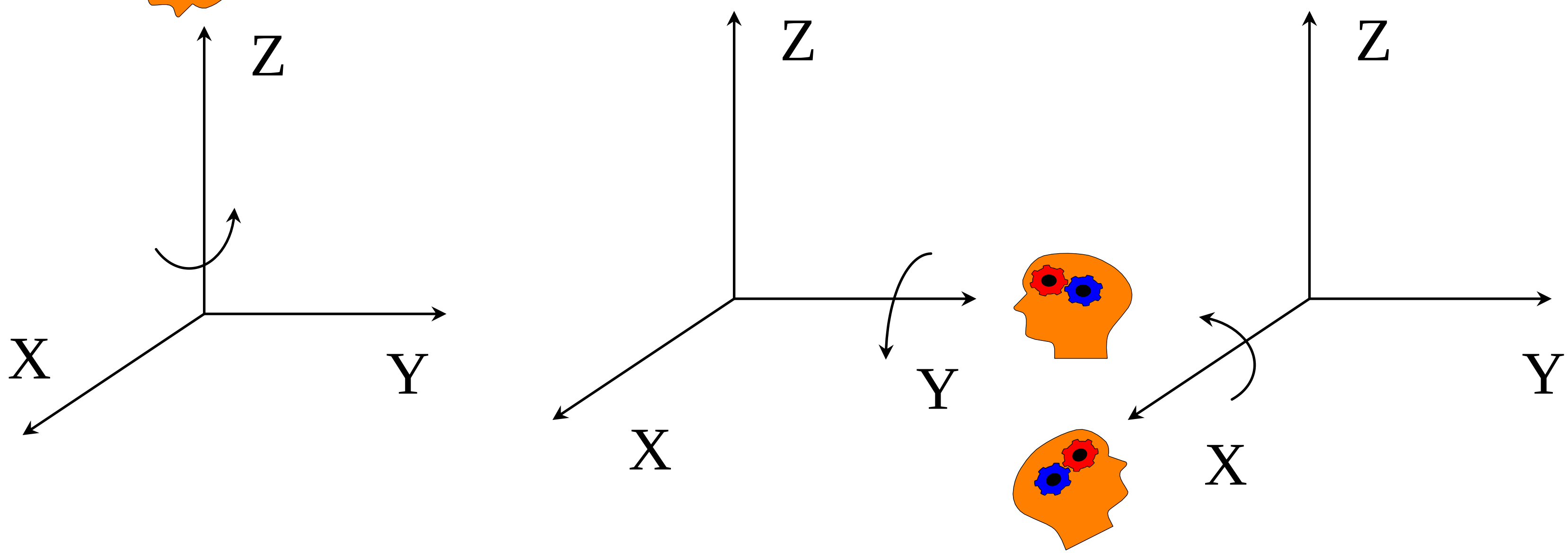

**Trans+Rot**

# 3D Translation




$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

# 3D: Euler Angle Rotation




$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

# *Generally*

- A translation
- Followed by a rotation around x
- Followed by another translation
- Followed by a rotation around y
- Followed by another translation
- Followed by a rotation around z
- Etc.

$$\cdots \mathbf{R}_z(\mathbf{R}_y(\mathbf{R}_x(\mathbf{P}+\mathbf{T}_1)+\mathbf{T}_2)+\mathbf{T}_3)\cdots$$

# *Sidebar: Representation of Coordinates*

- use $(X,Y,Z)^T$ for 3D and $(x,y)^T$ for 2D

  results in different representation for translation and rotation $\mathbf{P'}_{3x1}=\mathbf{R}_{3x3}\mathbf{P}_{3x1}+\mathbf{T}_{3x1}$

- Homogeneous coordinates

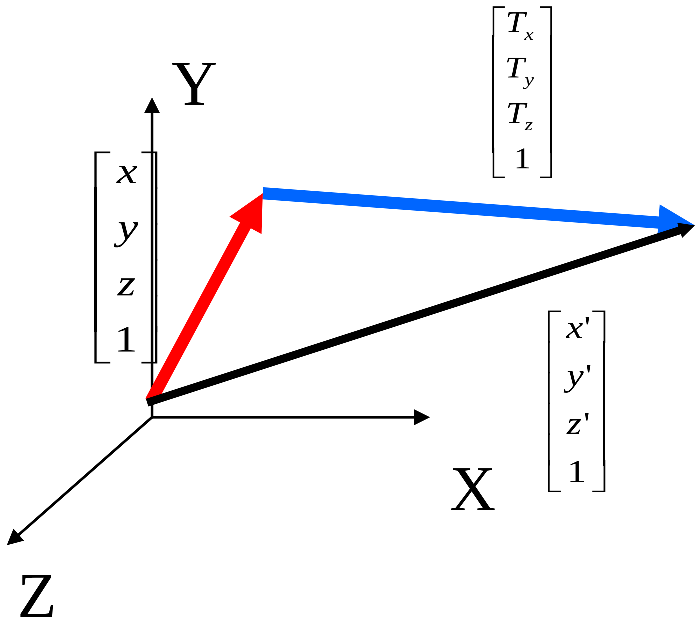
  $\mathbf{P'}_{4x1}=\mathbf{M}_{4x4}\mathbf{P}_{4x1}$

$$(x,y,z) \to (wx,wy,wz,w), w \neq 0$$

$$(wx,wy,wz,w) \to (wx/w, wy/w, wz/w)$$

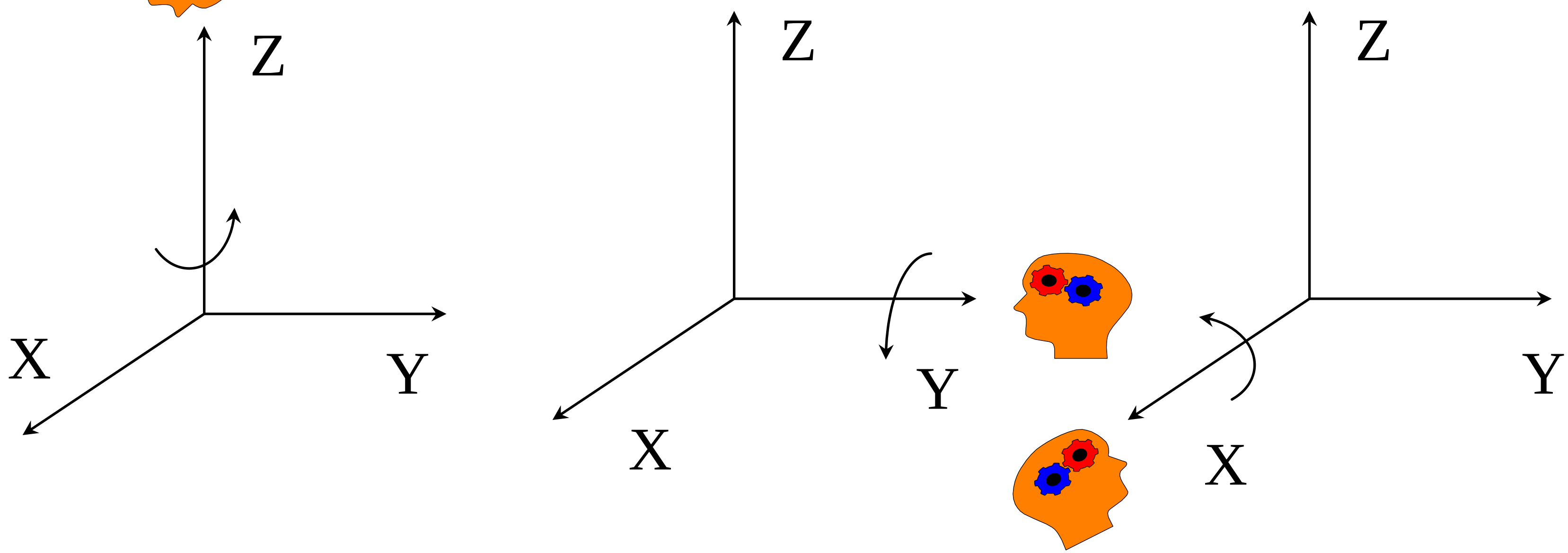
$$\mathbf{P}' = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \mathbf{P}$$

# 3D Translation




$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# 3D: Euler Angle Rotation




$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# *Generally*

- A translation
- Followed by a rotation around x
- Followed by another translation
- Followed by a rotation around y
- Followed by another translation
- Followed by a rotation around z
- Etc.

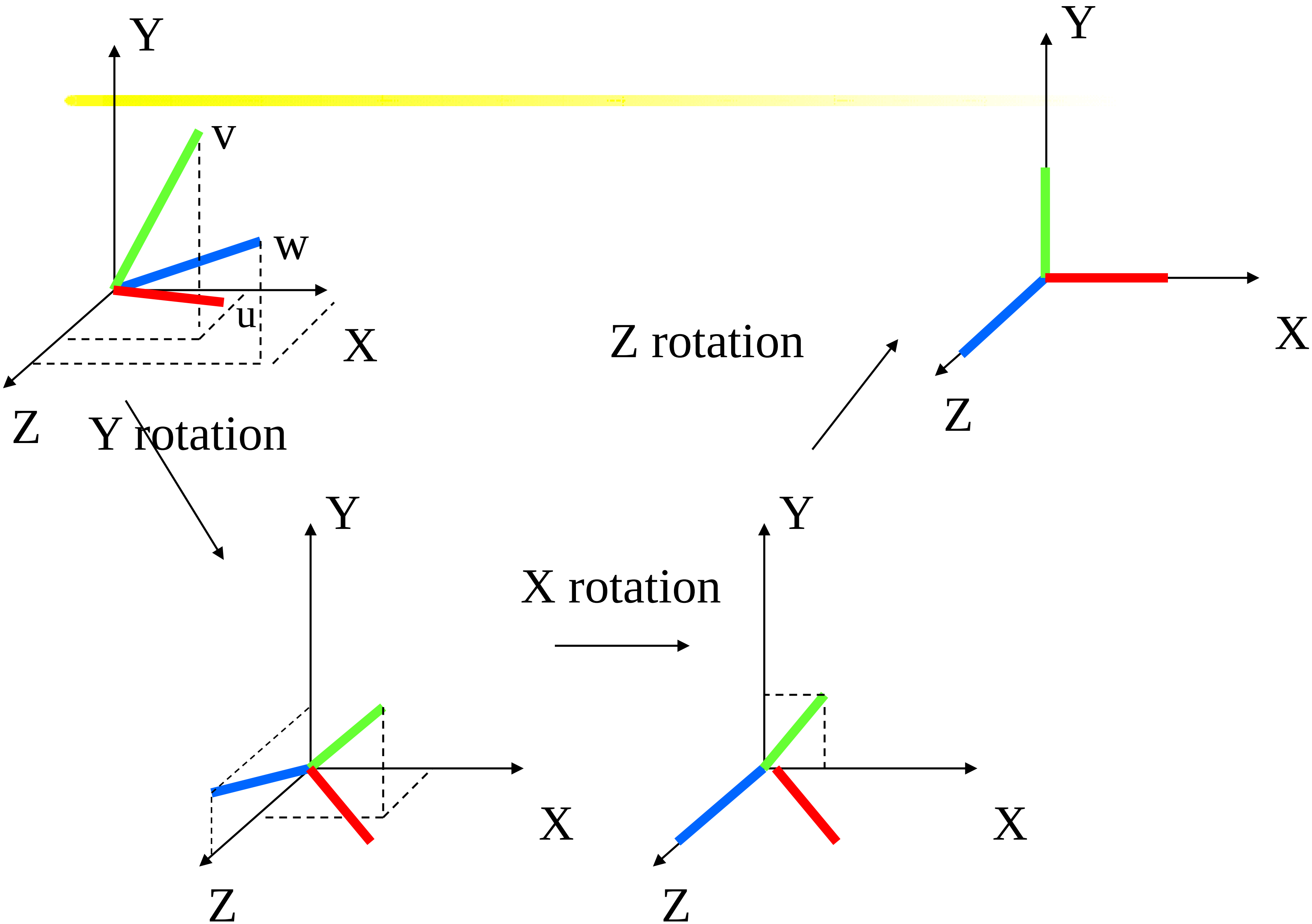
$$\cdots \mathbf{R}_z \mathbf{T}_3 \mathbf{R}_y \mathbf{T}_2 \mathbf{R}_x \mathbf{T}_1 \mathbf{P}$$

- A much elegant representation in terms of matrix operation
- Easy concatenation

# *View Normalization – Hard Way*

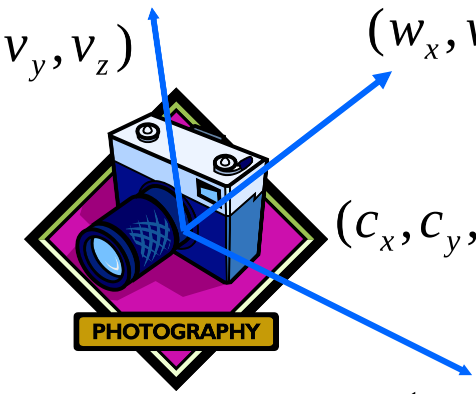
- Make world coordinates make sense to a camera
- Transform world coordinates into camera coordinates
- One translation
  - Zero out camera origin
- Three rotations –line up coordinate axes
  - Rotate about Y
  - Rotate about X
  - Rotate about Z

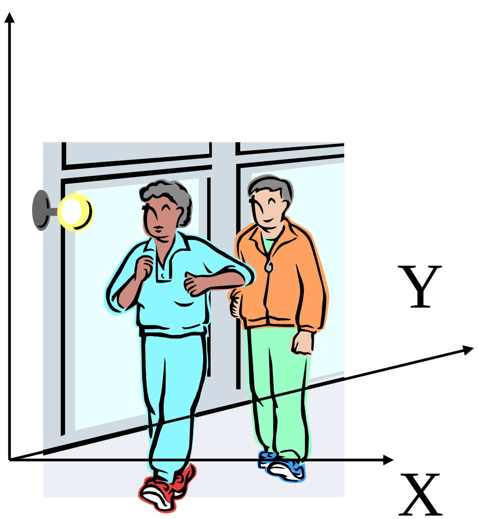
$$\mathbf{R}_z\mathbf{R}_x\mathbf{R}_y\mathbf{T}\mathbf{P}$$


Y
v
w
u
X
Z
Y rotation
Z rotation
X rotation

# World to Camera – Easy Way

$$\mathbf{P}_{camera} = \begin{bmatrix} \mathbf{R}_{camera\leftarrow world} & \mathbf{T}_{camera\leftarrow world} \\ 0 & 1 \end{bmatrix} \mathbf{P}_{world}$$



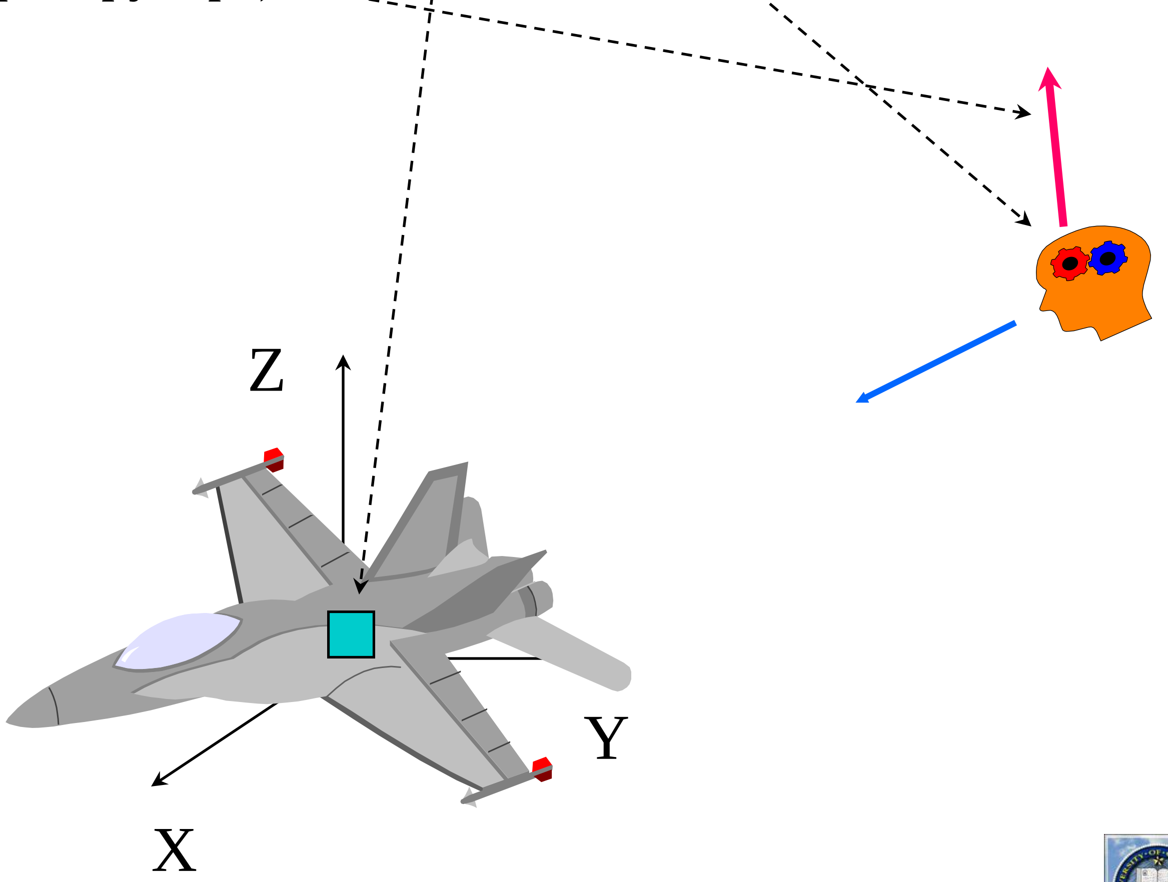




$$\mathbf{M}_{camera\leftarrow world} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} u_x & u_y & u_z & -\mathbf{r}_x \cdot \mathbf{c} \\ v_x & v_y & v_z & -\mathbf{r}_y \cdot \mathbf{c} \\ w_x & w_y & w_z & -\mathbf{r}_z \cdot \mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

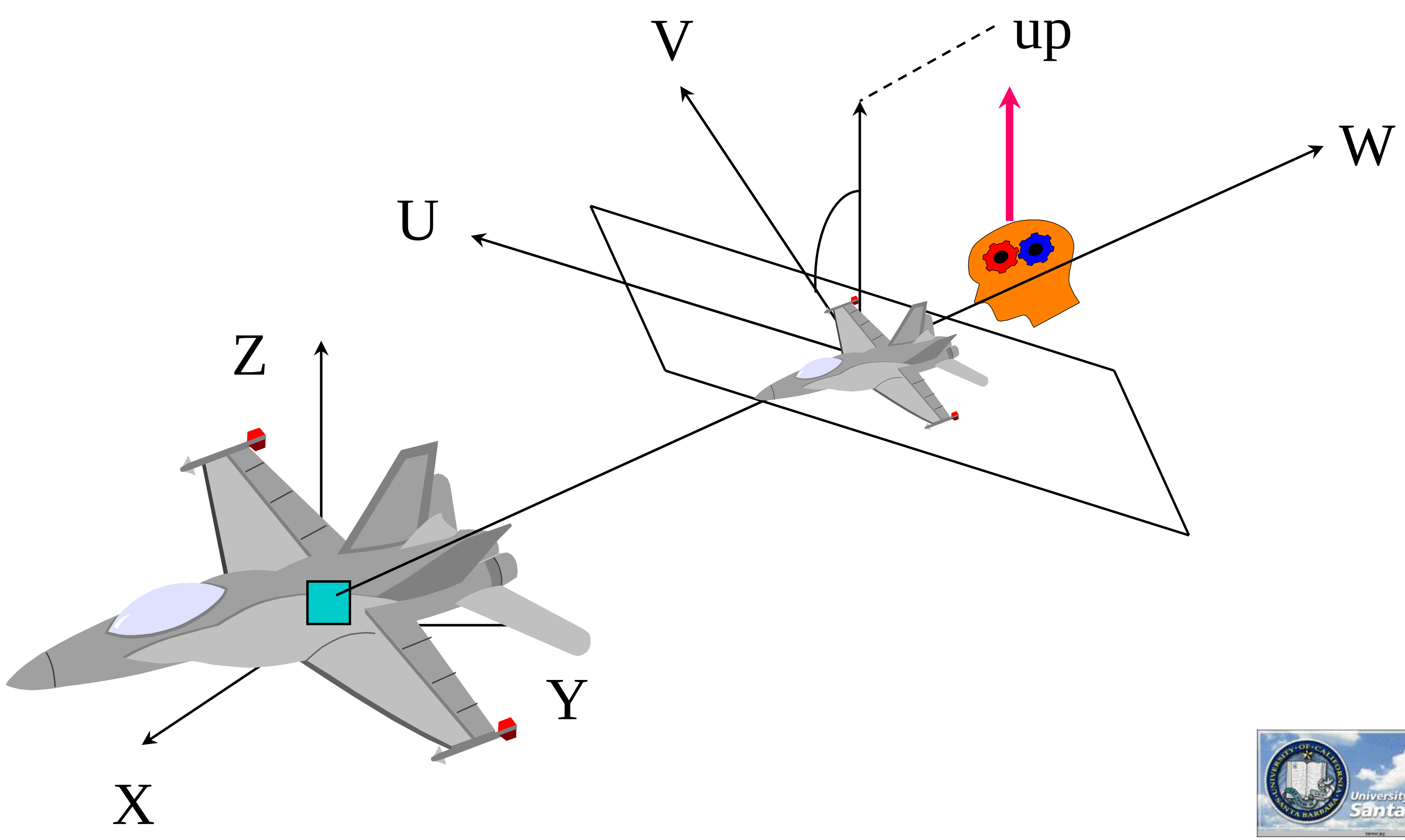
# *Viewing Transform in OpenGL*

void gluLookAt (GLdouble eyex, eyey, eyez,
GLdouble centerx, centery, centerz,
GLdouble upx, upy, upz)



# *Viewing Transform (cont.)*

- **eye** and **center: local** w(z) direction
- **up** and **local w(z):** local v(y) direction
- **local v(y)** and **w(z)** directions: local u(x) direction



# Viewing Normalization

- Figuring out **[u, v, w]** in **[x, y, z]** system
- Applying a rotation to transform **[x, y, z]** coordinates into **[u, v, w]** coordinates

$$w = \frac{e - c}{|e - c|}$$

$$u = \frac{up \times w}{|up \times w|}$$

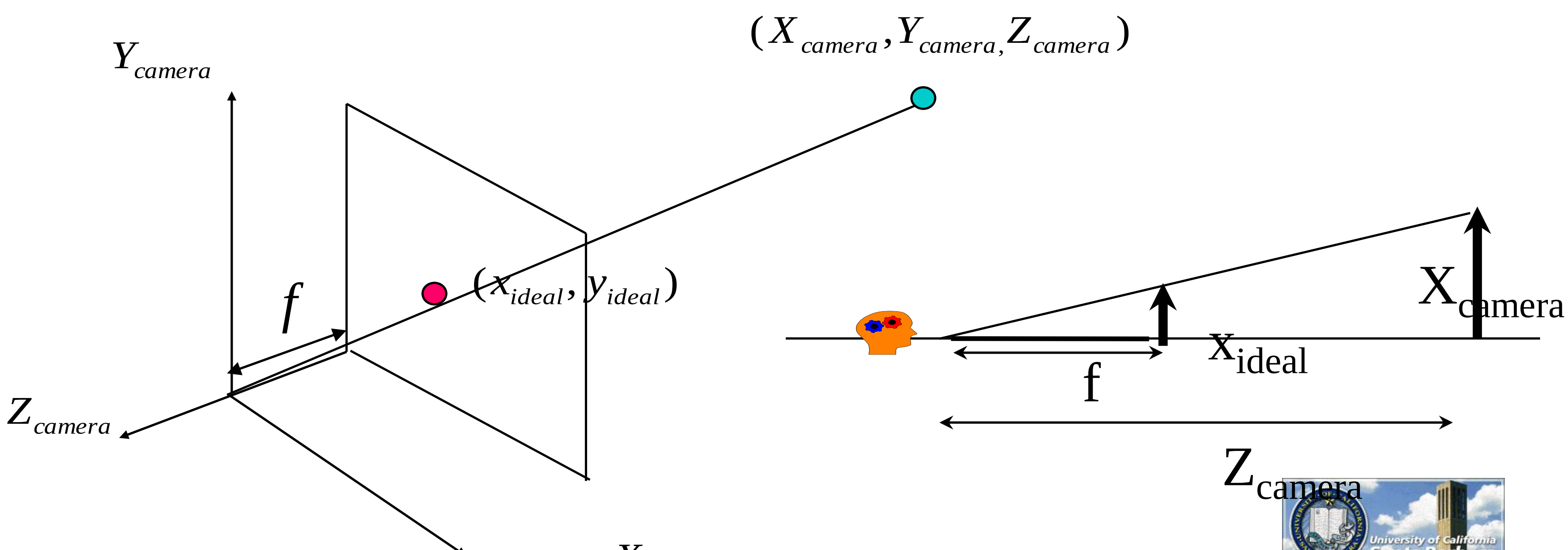
$$v = w \times u$$

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Camera to (Ideal) Image

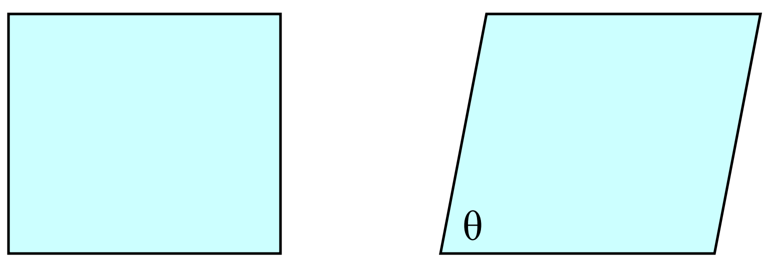
- Represented by a projection
  - focal length $f$, aspect ratio $1$

$$\begin{bmatrix} x_{ideal} \\ y_{ideal} \\ 1 \end{bmatrix} = \mathbf{M}_{ideal \leftarrow camera} \begin{bmatrix} X_{camera} \\ Y_{camera} \\ Z_{camera} \\ 1 \end{bmatrix} \quad \mathbf{p}_{ideal} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{P}_{camera} : \begin{cases} x_{ideal} = f \dfrac{X_{camrea}}{Z_{camera}} \\ y_{ideal} = f \dfrac{Y_{camera}}{Z_{camera}} \end{cases}$$

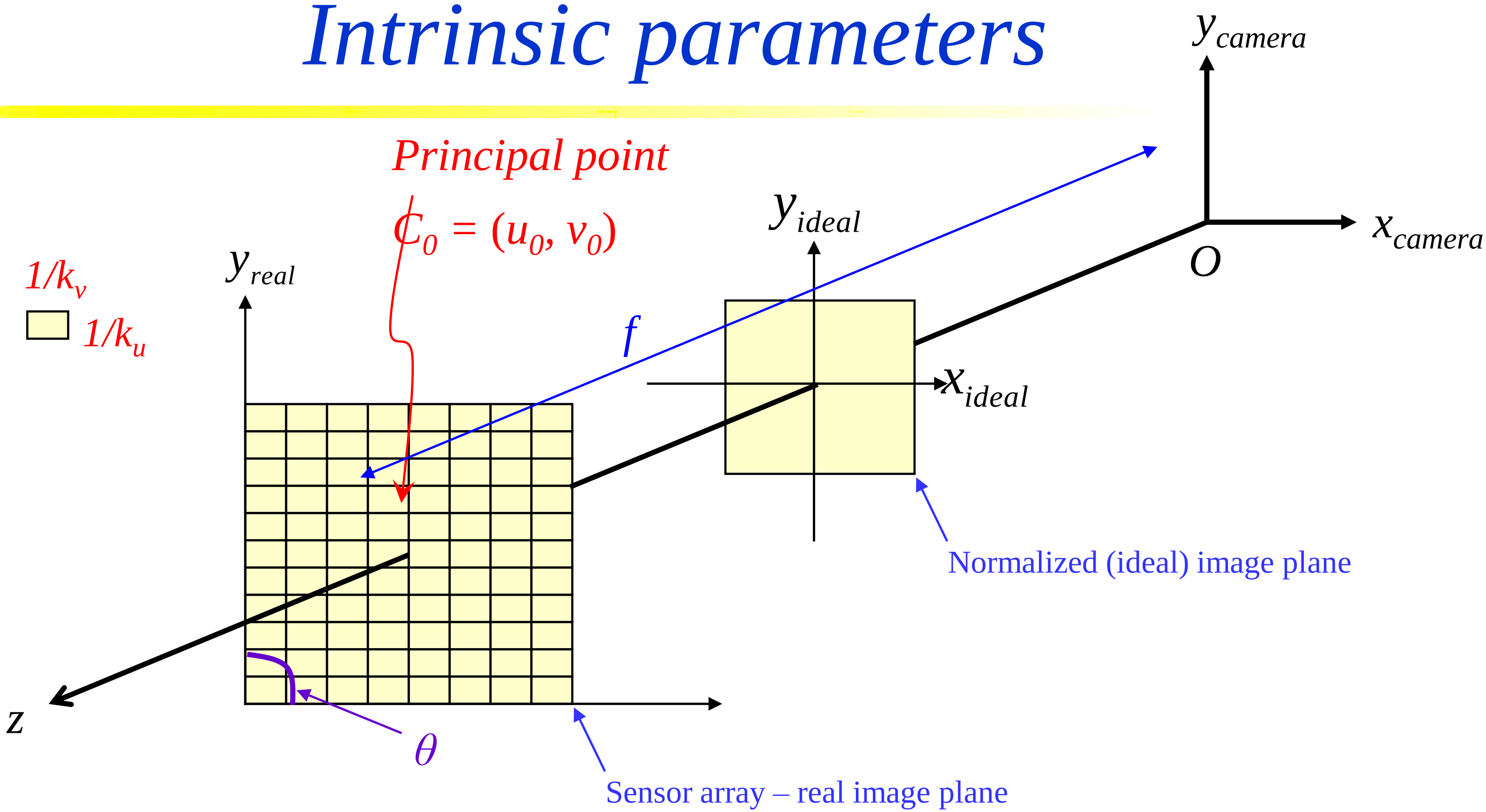


# *Intrinsic parameters*

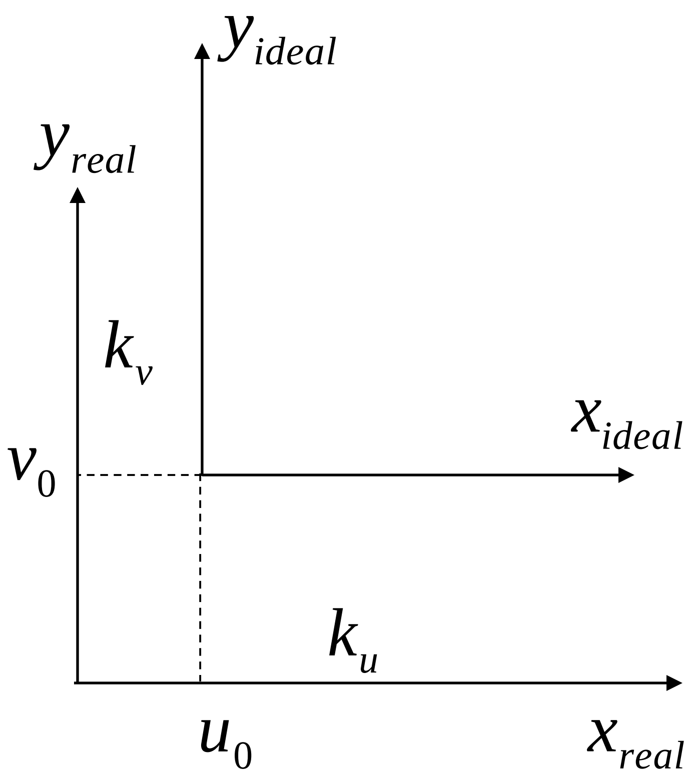
- 5 intrinsic parameters account for
  - The focal length ( $f$ )
  - The principal point $(C_0)=(u_0,v_0)$
    - Where the optical axis intersects the image plane
  - Pixel aspect ratio $(k_u, k_v)$
    - Pixels aren't necessarily square
  - Angle between the axes $(\theta)$
    - Skewness in manufacturing



# *Intrinsic parameters*



# Ideal Image to Real Image: Square Grid

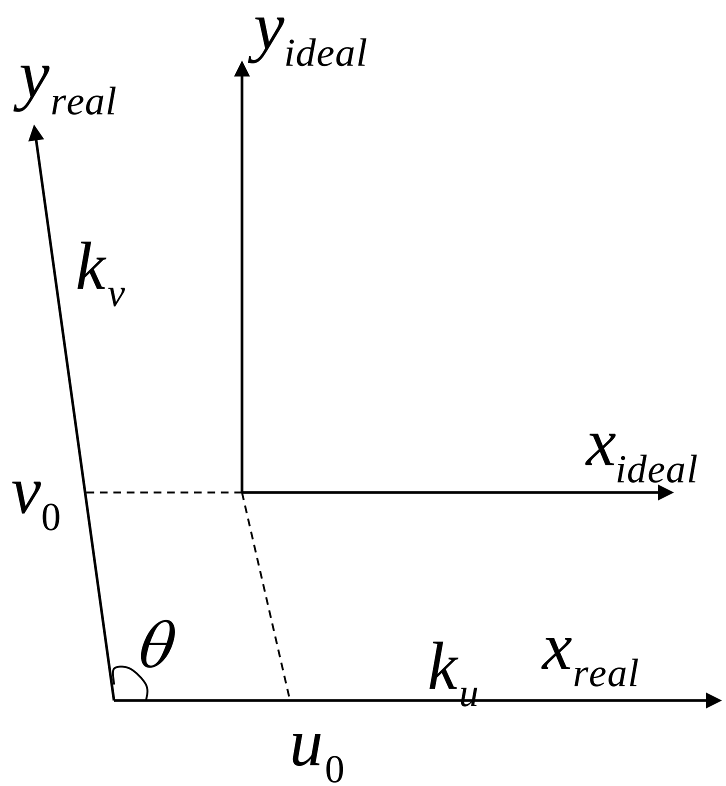



$$\mathbf{p}_{real} = \mathbf{M}_{real \leftarrow ideal} \mathbf{p}_{ideal}$$

$$= \begin{bmatrix} k_u & 0 & u_o \\ 0 & k_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{ideal}$$

$$\begin{cases} x_{real} = k_u x_{ideal} - u_o = f k_u \dfrac{X_{camrea}}{Z_{camera}} - u_o \\ y_{real} = k_v x_{ideal} - v_o = f k_v \dfrac{Y_{camera}}{Z_{camera}} - v_o \end{cases}$$

# *Ideal Image to Real Image: Non-square grid*




$$\mathbf{p}_{real} = \begin{bmatrix} k_u & k_u \cot\theta & u_o \\ 0 & k_v / \sin\theta & v_o \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{ideal}$$

# Putting It All Together

$$\mathbf{p}_{real} = \mathbf{M}_{real\leftarrow ideal}\mathbf{M}_{ideal\leftarrow camera}\mathbf{M}_{camera\leftarrow world}\mathbf{P}_{world}$$

$$= \mathbf{M}_{real\leftarrow world}\mathbf{P}_{world} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{bmatrix}\mathbf{P}_{world}$$

$$x_{real} = \frac{q_{11}X_{world} + q_{12}Y_{world} + q_{13}Z_{world} + q_{14}}{q_{31}X_{world} + q_{32}Y_{world} + q_{33}Z_{world} + q_{34}}$$

$$y_{real} = \frac{q_{21}X_{world} + q_{22}Y_{world} + q_{23}Z_{world} + q_{24}}{q_{31}X_{world} + q_{32}Y_{world} + q_{33}Z_{world} + q_{34}}$$

# Usage

- Governing equation

$$\mathbf{p}_{real} = \mathbf{M}_{real \leftarrow ideal} \mathbf{M}_{ideal \leftarrow camera} \mathbf{M}_{camera \leftarrow world} \mathbf{P}_{world} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{bmatrix} \mathbf{P}_{world}$$

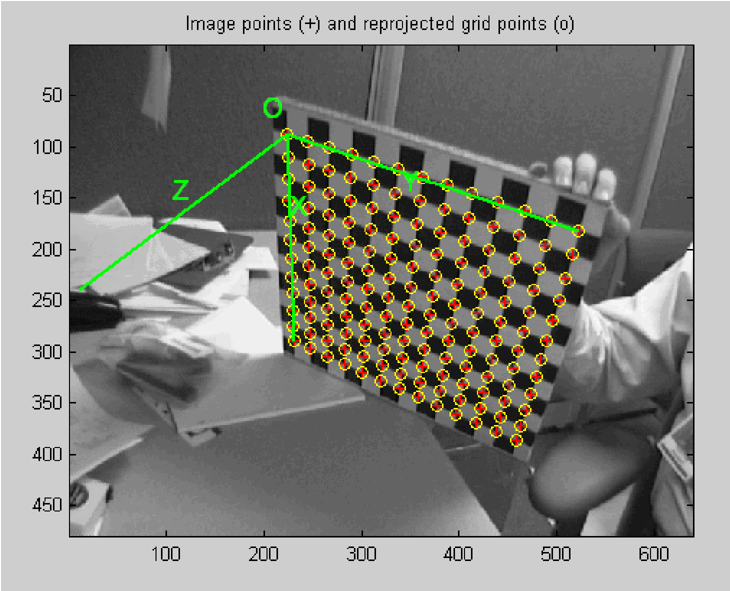
- Off-line process
  - Given known 3D coordinates (landmarks) and their 2D projections, calculate q's
- On-line process
  - Given arbitrary 3D coordinates (first down at 30 yards) and q's, calculate 2D coordinates (where to draw the first-and-ten line)
  - Given arbitrary 2D coordinates (images of a vehicle) and q's, calculate 3D coordinates (where to aim the gun to fire)

# *Camera Calibration and Registration*

- First step
  - Estimate the combined transformation matrix $\mathbf{M}_{real \leftarrow world}$
- Second step
  - Estimate intrinsic camera parameters
  - Estimate extrinsic camera parameters
- Solution
  - Using objects of known sizes and shapes (6 points at least)
  - Each point provides two constraints (x,y)
  - A checked board pattern placed at different depths

# *Calibration software available*

- Lots of it these days!
- E.g., Camera Calibration Toolkit for Matlab
  - From the Computational Vision Group at Caltech (Bouguet)
    - http://www.vision.caltech.edu/bouguetj/calib_doc/
    - Includes lots of links to calibration tools and research



# Putting It All Together

$$\mathbf{p}_{real} = \begin{bmatrix} k_u & 0 & u_o \\ 0 & k_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \mathbf{P}_{world}$$

$$\begin{bmatrix} k_u & 0 & u_o \\ 0 & k_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_o & 0 \\ 0 & \alpha_v & v_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u \mathbf{r}_1 + u_o \mathbf{r}_3 & \alpha_u t_x + u_o t_z \\ \alpha_v \mathbf{r}_2 + v_o \mathbf{r}_3 & \alpha_v t_y + v_o t_z \\ \mathbf{r}_3 & t_z \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{q}_1^T & q_{14} \\ \mathbf{q}_2^T & q_{24} \\ \mathbf{q}_3^T & q_{34} \end{bmatrix}$$

$$|\mathbf{q}_3| = 1, (\mathbf{q}_1 \times \mathbf{q}_3) \cdot (\mathbf{q}_2 \times \mathbf{q}_3) = 0$$

# *Camera Calibration*

- Certainly, not all 3x4 matrices are like above
  - 3x4 matrices have 11 free parameters (with a scale factor that cannot be decided uniquely)
  - matrix in the previous slide has 10 parameters (2 scale, 2 camera center, 3 translation, 3 rotation)
  - additional constraints can be very useful
    - to solve for the matrix, and
    - to compute the parameters
  - Theorem: 3x4 matrices can be put in the form of the previous slide if and only if the following two constraints are satisfied

$$|\mathbf{q}_3| = 1, (\mathbf{q}_1 \times \mathbf{q}_3) \cdot (\mathbf{q}_2 \times \mathbf{q}_3) = 0$$

# *Finding the transform matrix*

$$\mathbf{p}_{real} = \begin{bmatrix} \mathbf{q_1}^T & q_{14} \\ \mathbf{q_2}^T & q_{24} \\ \mathbf{q_3}^T & q_{34} \end{bmatrix} \mathbf{P}_{world}$$

$$\begin{bmatrix} wx_{real} \\ wy_{real} \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{q_1}^T & q_{14} \\ \mathbf{q_2}^T & q_{24} \\ \mathbf{q_3}^T & q_{34} \end{bmatrix} \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{q_1}^T & q_{14} \\ \mathbf{q_2}^T & q_{24} \\ \mathbf{q_3}^T & q_{34} \end{bmatrix} \begin{bmatrix} \mathbf{P}^3_{world} \\ 1 \end{bmatrix}$$

$$\frac{\mathbf{q_1}^T \mathbf{P}^3_{world} + q_{14}}{\mathbf{q_3}^T \mathbf{P}^3_{world} + q_{34}} = x_{real} \Rightarrow \mathbf{q_1}^T \mathbf{P}^3_{world} - u\mathbf{q_3}^T \mathbf{P}^3_{world} + q_{14} - uq_{34} = 0$$

$$\frac{\mathbf{q_2}^T \mathbf{P}^3_{world} + q_{24}}{\mathbf{q_3}^T \mathbf{P}^3_{world} + q_{34}} = y_{real} \Rightarrow \mathbf{q_2}^T \mathbf{P}^3_{world} - v\mathbf{q_3}^T \mathbf{P}^3_{world} + q_{24} - vq_{34} = 0$$

$$\Rightarrow \mathbf{AQ} = \mathbf{0}$$

# *Finding the transform matrix (cont.)*

- Each data point provide two equations, with at least 6 points we will have 12 equations for solving 11 numbers up to a scale factor
- Lagrange multipliers can be used to incorporate other constraints
  - The usual constraint is $\mathbf{q}_3^2=1$
- Afterward, both intrinsic and extrinsic parameters can be recovered

# Details

$$\mathbf{AQ} = \mathbf{0}$$

$$\min \|\mathbf{AQ}\|^2 \text{ subject to } |\mathbf{q}_3| = 1$$

$$\min \|\mathbf{AQ}\|^2 + \lambda(1 - |\mathbf{q}_3|^2)$$

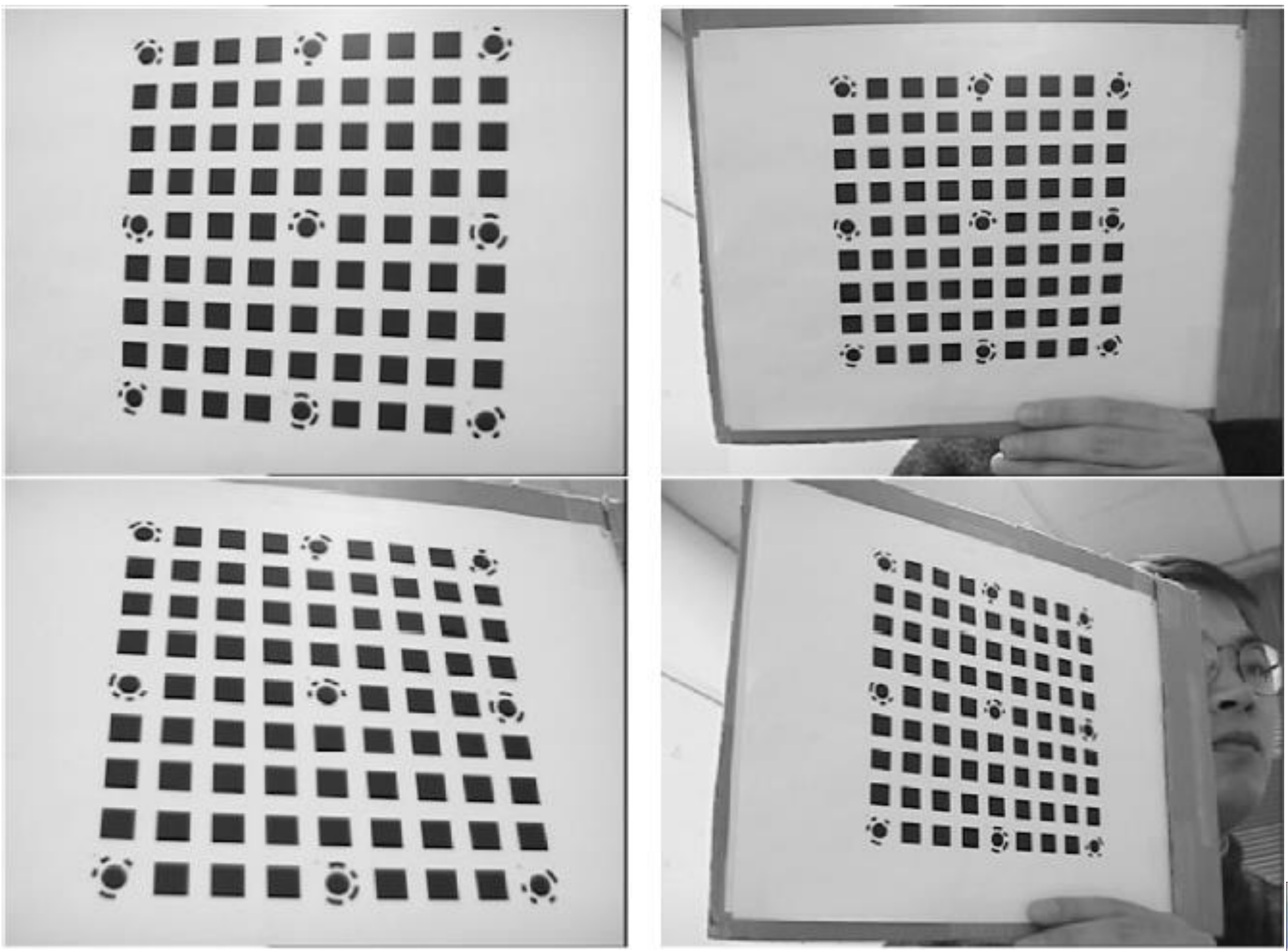
- Solved by Langrage multiplier

# *Other Formulations*

- With a calibration pattern
  - Flexible placement of the pattern
- With a calibration patterns
  - Use vanishing points and vanishing lines
- Calibrating against lens distortion
  - Fisheye and others
- Calibrating pan-tilt-zoom cameras
  - Pan and tilt axis direction, placement, angle
- Registration vs. calibration
  - Many times vs. once
  - On-line vs. off-line

# *Flexible Pattern Placement*

- http://research.microsoft.com/~zhang/calib/



# *Step 1: Intrinsic Parameters*

- $\mathbf{x}_{\text{image}} = \mathbf{H}\mathbf{x}_{\text{plane}} = \mathbf{K}[\mathbf{R}_1, \mathbf{R}_2, \mathbf{T}]\mathbf{x}_{\text{plane}}$

$$
\begin{aligned}
&\mathbf{H} = \mathbf{K}[\mathbf{R}_1 \mathbf{R}_2 \mathbf{T}] \\
&\Rightarrow \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \mathbf{K}[\mathbf{R}_1 \mathbf{R}_2 \mathbf{T}] \\
&\Rightarrow \begin{bmatrix} \mathbf{K}^{-1}\mathbf{h}_1 & \mathbf{K}^{-1}\mathbf{h}_2 & \mathbf{K}^{-1}\mathbf{h}_3 \end{bmatrix} = [\mathbf{R}_1 \mathbf{R}_2 \mathbf{T}] \\
&\Rightarrow \begin{matrix} (\mathbf{K}^{-1}\mathbf{h}_1)^T \mathbf{K}^{-1}\mathbf{h}_2 = 0 & (\mathbf{K}^{-1}\mathbf{h}_1)^T \mathbf{K}^{-1}\mathbf{h}_1 = (\mathbf{K}^{-1}\mathbf{h}_1)^T \mathbf{K}^{-1}\mathbf{h}_1 \\ \mathbf{h}_1{}^T \mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h}_2 = 0 & \mathbf{h}_1{}^T \mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h}_1 = \mathbf{h}_2{}^T \mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h}_2 \end{matrix}
\end{aligned}
$$

Image of absolute conic

- One homography
  - 8 DOFs
  - 6 extrinsic parameters (3 rotation + 3 translation)
  - 2 constraints on intrinsic parameters
  - 3 planes in general configuration

# *Step 2: Extrinsic Parameters*

$$\mathbf{H} = \mathbf{K}[\mathbf{R}_1 \mathbf{R}_2 \mathbf{T}]$$

$$\Rightarrow \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \mathbf{K}[\mathbf{R}_1\ \mathbf{R}_2\ \mathbf{T}]$$

$$\Rightarrow \mathbf{R}_1 = \lambda \mathbf{K}^{-1} \mathbf{h}_1$$

$$\mathbf{R}_2 = \lambda \mathbf{K}^{-1} \mathbf{h}_2$$

$$\mathbf{T} = \lambda \mathbf{K}^{-1} \mathbf{h}_3$$

$$\mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2$$

$$\lambda = \frac{1}{\left\| \mathbf{K}^{-1} \mathbf{h}_1 \right\|} = \frac{1}{\left\| \mathbf{K}^{-1} \mathbf{h}_2 \right\|}$$

# *Step 3: Intrinsic + Extrinsic Parameters*

- $\mathbf{x}_{\text{image}} = \mathbf{H}\mathbf{x}_{\text{plane}} = \mathbf{K}[\mathbf{R}_1, \mathbf{R}_2, \mathbf{T}]\mathbf{x}_{\text{plane}}$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(\mathbf{x}_{ij}^{\textbf{image}} - \mathbf{K}[\mathbf{R}_1\mathbf{R}_2\mathbf{T}]\mathbf{x}_{ij}^{\text{model}})^2$$

# of images      # of points/images

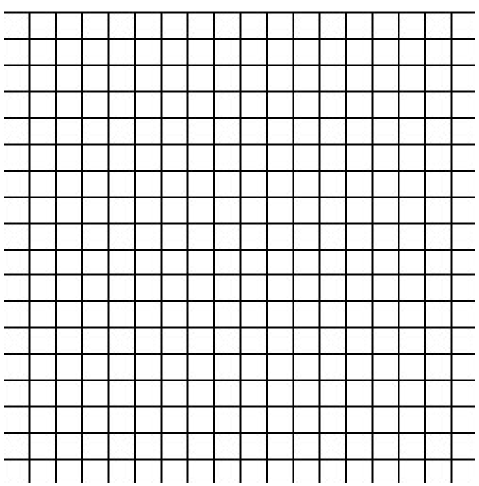
- Nonlinear optimization
- Using Lenvenberg-Marquardt in Minpack
- **K** from the previous step as initial guess

# *Calibrating for radial distortion*

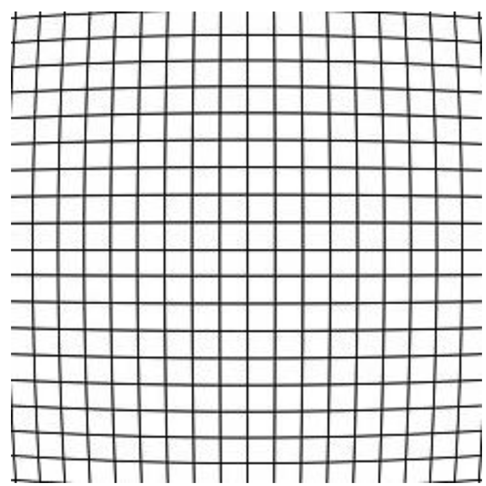
- The camera lens also introduces errors of several type (as we've already discussed):
  - Spherical aberration
  - Coma
  - Chromatic aberration
  - Vignetting
  - Astigmatism
  - Misfocus
  - Radial distortion
- Of these, *radial distortion* is the most significant in most systems, and it can be corrected for

# *Radial distortion*

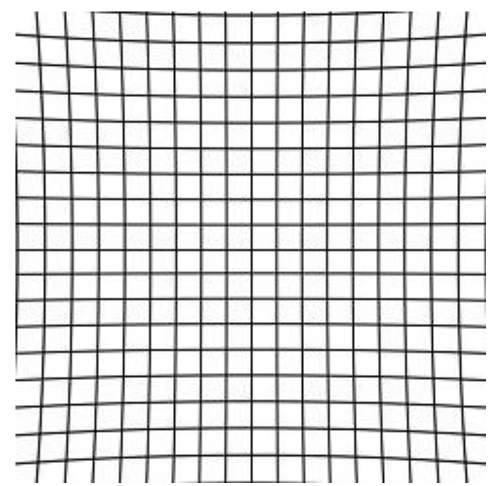
- Variation in the magnification for object points at different distances from the optical axis
- Effect increases with distance from the optical axis
- Straight lines become bent!
- Two main descriptions: *barrel* distortion and *pincushion* distortion
- Can be modeled and corrected for



Correct



Barrel



Pincushion

University of California Santa Barbara

# Correcting for radial distortion

Original

Corrected

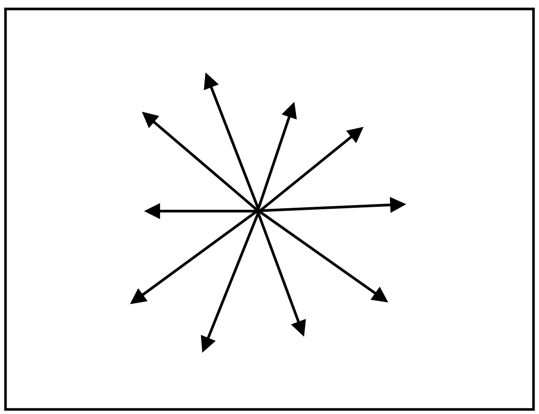
# *Modeling Lens Distortion*

- Radial, Barrel, Pincushion, etc.
  - Modeled as bi-cubic (or bi-linear) with more parameters to compensate for
  - Very hard to solve

$$x_{real} = \begin{bmatrix} x_{ideal}^3 & x_{ideal}^2 & x_{ideal} & 1 \end{bmatrix} \begin{bmatrix} a_{11}^x & a_{12}^x & \cdots & a_{14}^x \\ a_{21}^x & a_{22}^x & \cdots & a_{24}^x \\ \cdots & \cdots & \cdots & \cdots \\ a_{41}^x & a_{42}^x & \cdots & a_{44}^x \end{bmatrix} \begin{bmatrix} y_{ideal}^3 \\ y_{ideal}^2 \\ y_{ideal} \\ 1 \end{bmatrix}$$

# *Modeling radial distortion*

- The radial distortion can be modeled as a polynomial function ($\lambda$) of $d^2$, where $d$ is the distance between the image center and the image point
  - Called the *radial alignment constraint*



$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \lambda(d) \begin{bmatrix} u \\ v \end{bmatrix}$$

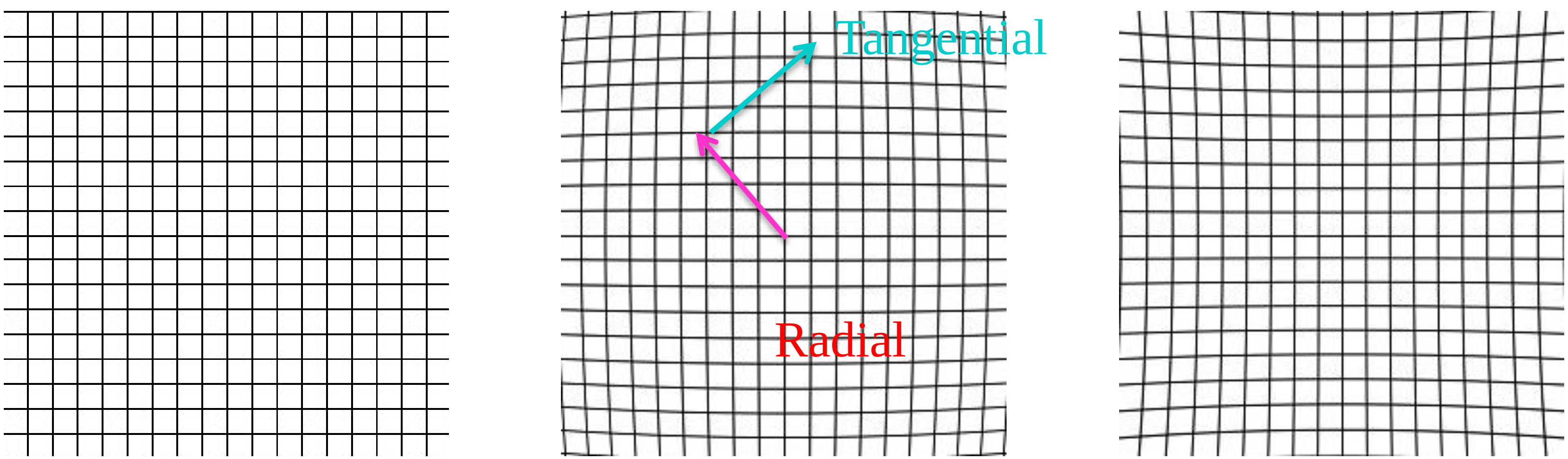
e.g., $\lambda(d) = 1 + \kappa_1 d^2 + \kappa_2 d^4 + \kappa_3 d^6$

$q$ distortion coefficients ($q$ <4)

# *Less Frequently Used – Tangential Distortion*

- Less common

$$dx = \begin{bmatrix} 2\,kc(3)\;x\;y + kc(4)\left(r^2 + 2x^2\right) \\ kc(3)\left(r^2 + 2y^2\right) + 2\,kc(4)\;x\;y \end{bmatrix}$$



# *Modeling distortion*

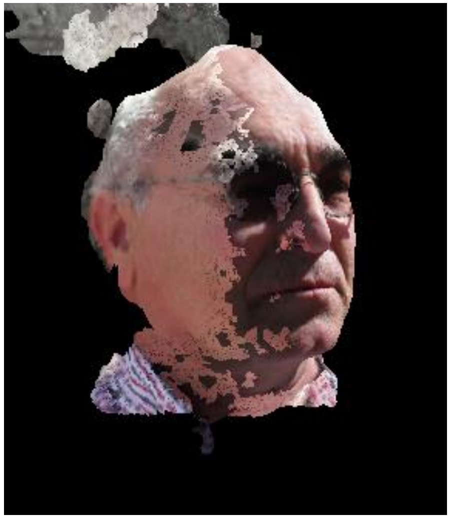
- Since $d$ is a function of $u$ and $v$, we could also write $\lambda$ as $\lambda(u, v)$

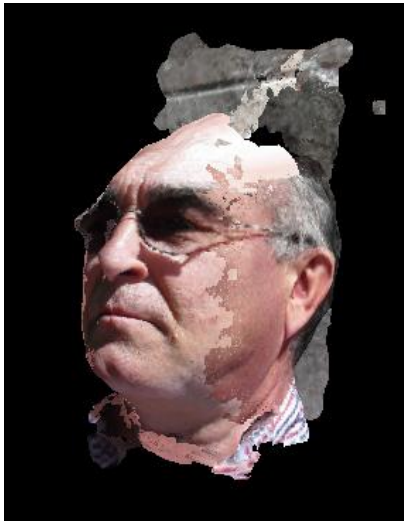
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \frac{1}{\lambda(u',v')} & 0 & 0 \\ 0 & \frac{1}{\lambda(u',v')} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{P}$$

where $(u', v')$ comes from **K[R|T]P**

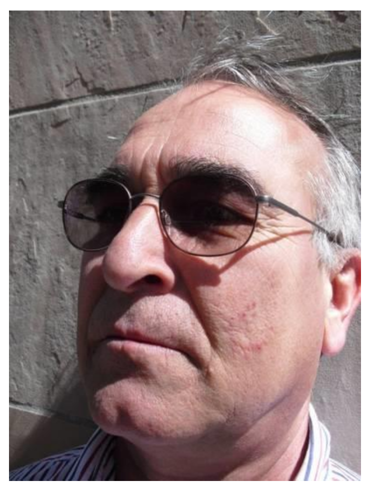
- Now do calibration by estimating the **11+*q*** parameters

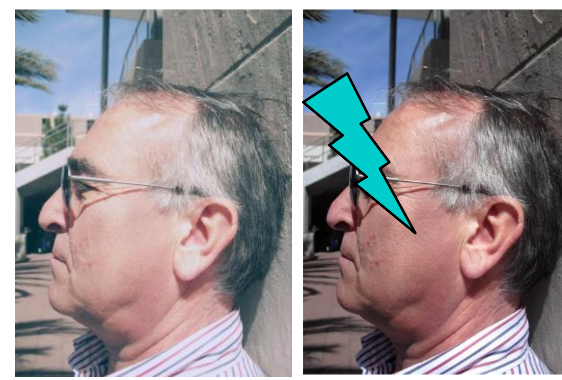
# *Radiometry Calibration*

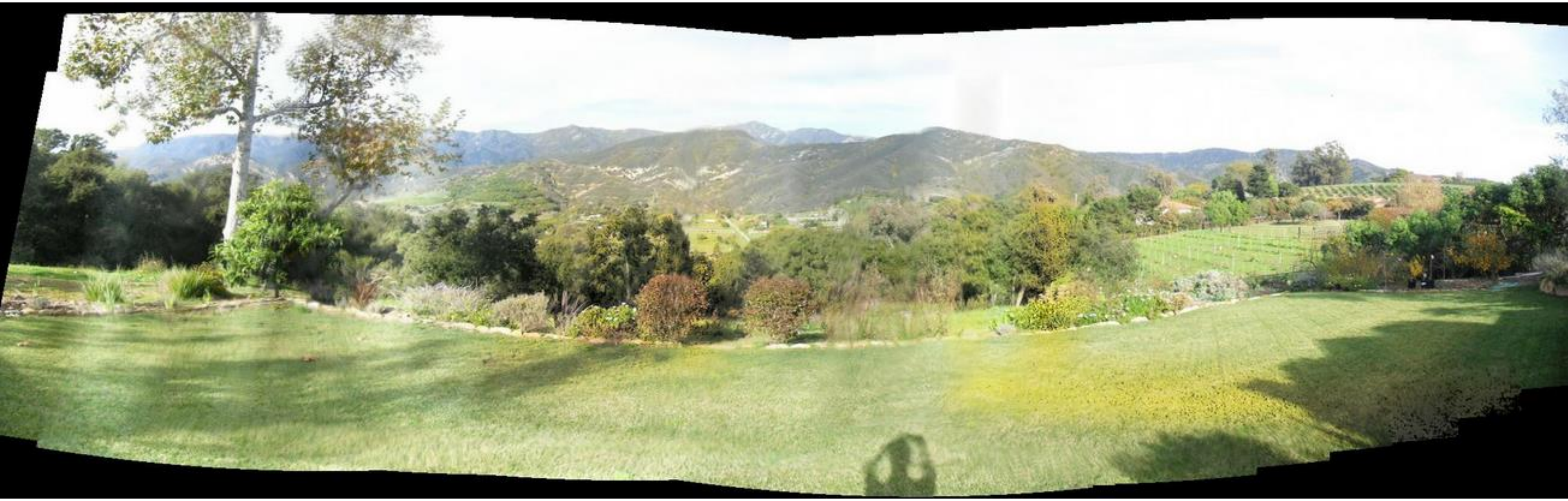




# *Radiometry Calibration*



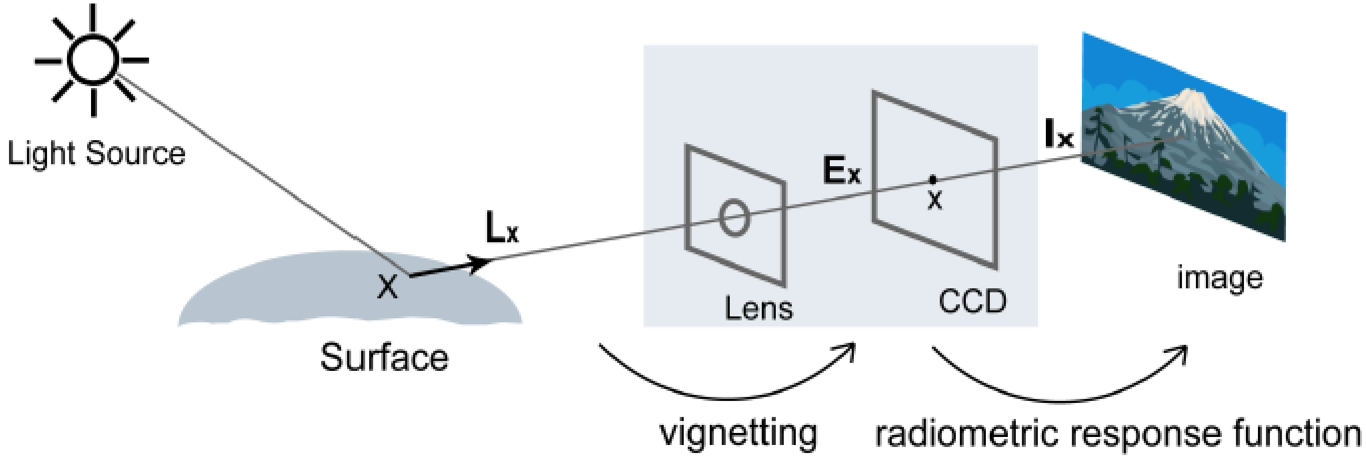




University of California
Santa Barbara

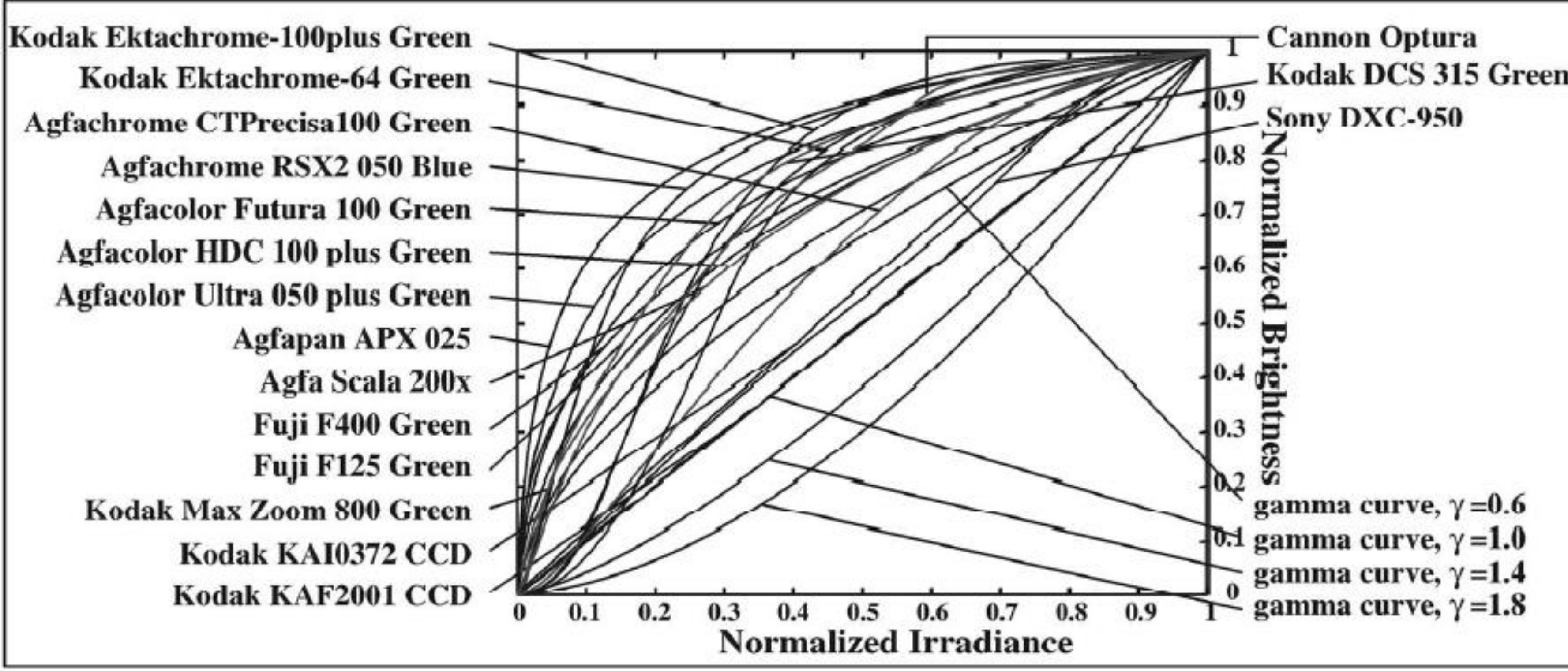
University of California Santa Barbara

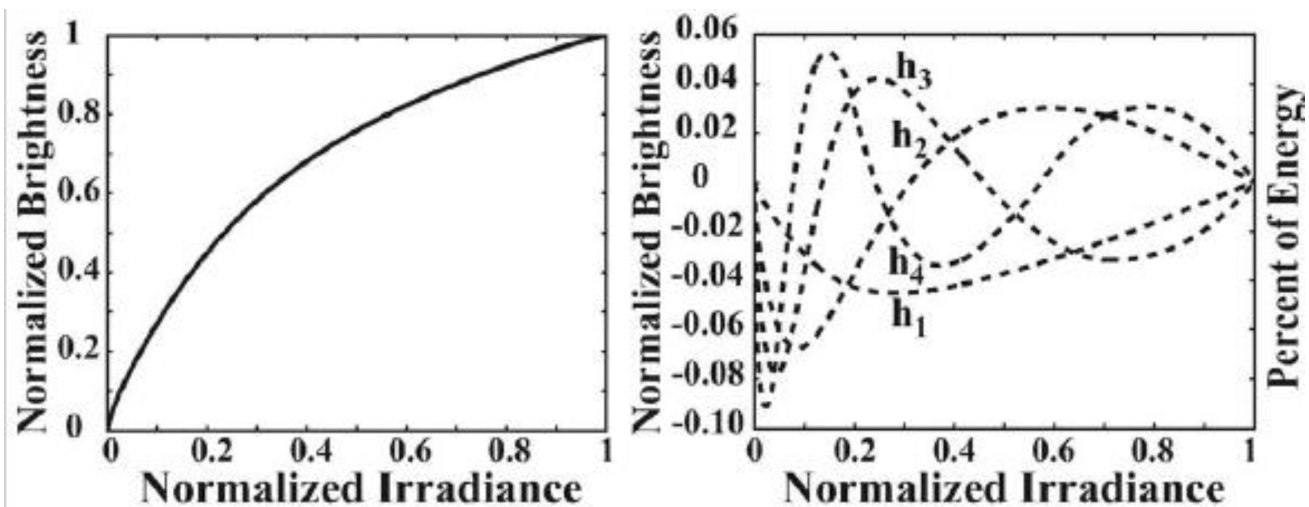
# *Formulation*




$$I_{\mathbf{x}} = \boxed{f}(k\boxed{V}(r_{\mathbf{x}})L_{\mathbf{x}})$$

$$f(E) = f_0(E) + \sum_{n=1}^{M} c_n h_n(E), \qquad V(r) = 1 + \sum_{n=1}^{D} \beta_n r^{2n}$$


Kodak Ektachrome-100plus Green
Kodak Ektachrome-64 Green
Agfachrome CTPrecisa100 Green
Agfachrome RSX2 050 Blue
Agfacolor Futura 100 Green
Agfacolor HDC 100 plus Green
Agfacolor Ultra 050 plus Green
Agfapan APX 025
Agfa Scala 200x
Fuji F400 Green
Fuji F125 Green
Kodak Max Zoom 800 Green
Kodak KAI0372 CCD
Kodak KAF2001 CCD
Cannon Optura
Kodak DCS 315 Green
Sony DXC-950
gamma curve, γ =0.6
gamma curve, γ =1.0
gamma curve, γ =1.4
gamma curve, γ =1.8
Normalized Brightness
Normalized Irradiance
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Normalized Brightness
Normalized Irradiance
Normalized Brightness
Percent of Energy
Normalized Irradiance
h3
h2
h4
h1
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.10
0
0.2
0.4
0.6
0.8
1

(a) (b)

# *Where is the = Sign?*

- Assume overlapped area contains diffuse surface
- It doesn't matter where the camera is, Lx is constant (the surface gives out light equally in all directions)
- Procedure: match the intensities of points that are seen in both images 1 and 2

$$I_{\mathbf{x}} = f(kV(r_{\mathbf{x}})L_{\mathbf{X}})$$

$$I_x^1 = f(k_1 V(r_x^1) L_x) \qquad I_x^2 = f(k_2 V(r_x^2) L_x)$$

$$L_x = f^{-1}(I_x^1)/k_1 V(r_x^1) \qquad L_x = f^{-1}(I_x^2)/k_2 V(r_x^2)$$

$$\Rightarrow f^{-1}(I_x^1)/k_1 V(r_x^1) = f^{-1}(I_x^2)/k_2 V(r_x^2)$$