Lecture 16:
Camera geometry and calibration

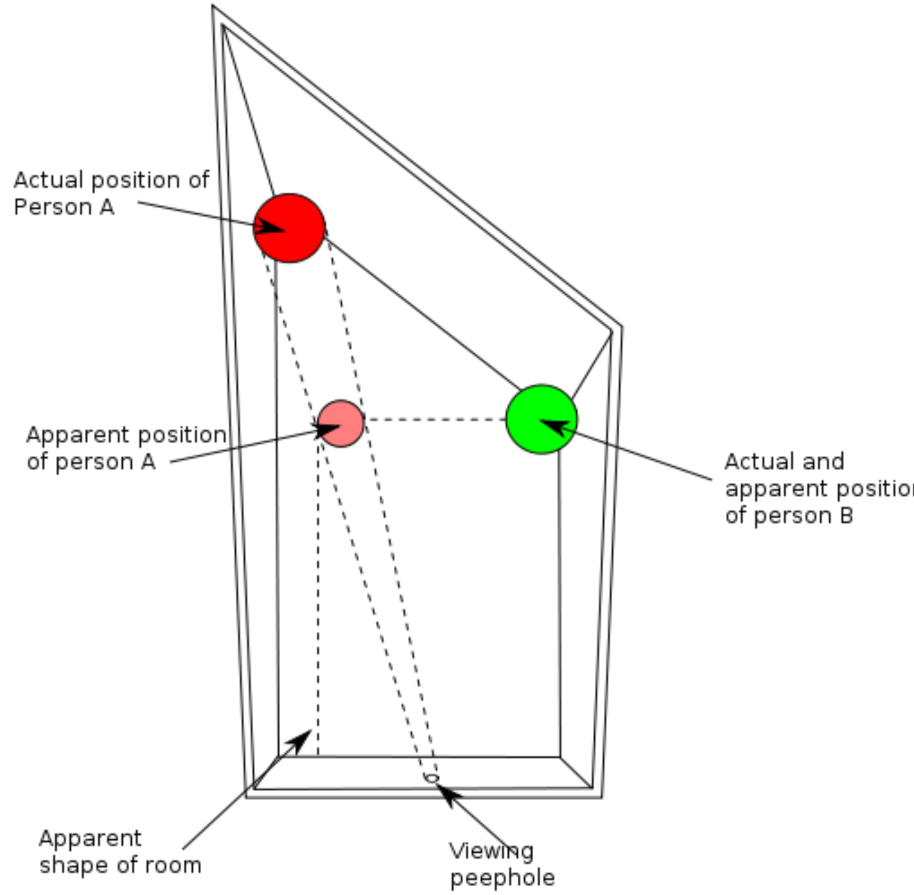
COS 429: Computer Vision

PRINCETON UNIVERSITY

Slides adapted from: Szymon Rusinkiewicz, Jia Deng

Single-view geometry

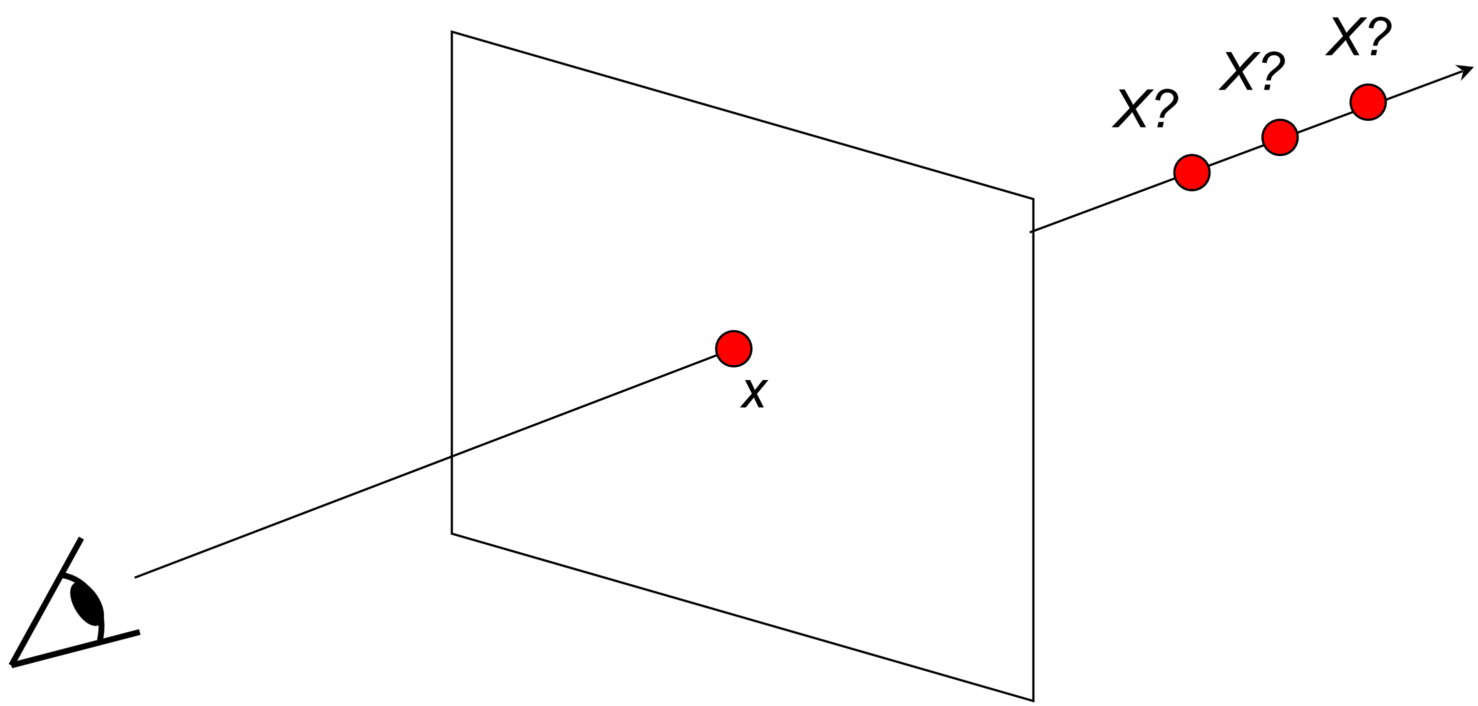
Ames Room



http://en.wikipedia.org/wiki/Ames_room
http://www.youtube.com/watch?v=gJhyu6nIGt8

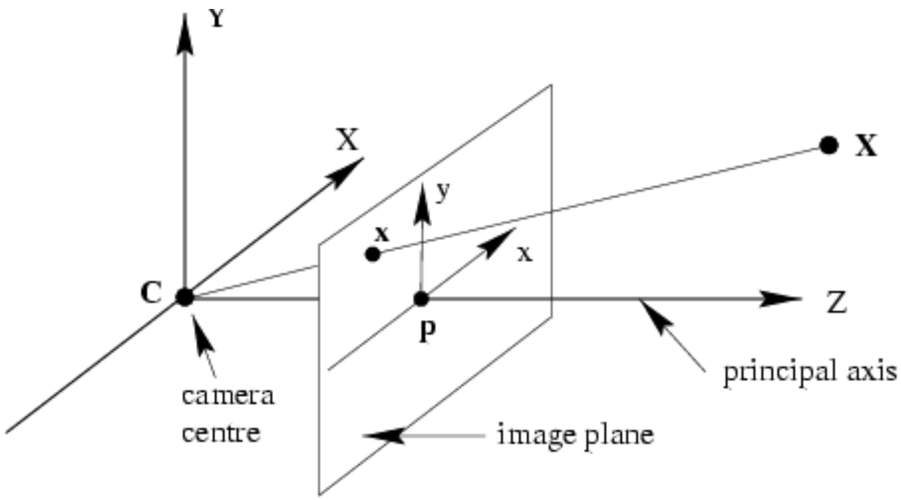
Source: S. Lazebnik

Our goal: Recovery of 3D structure

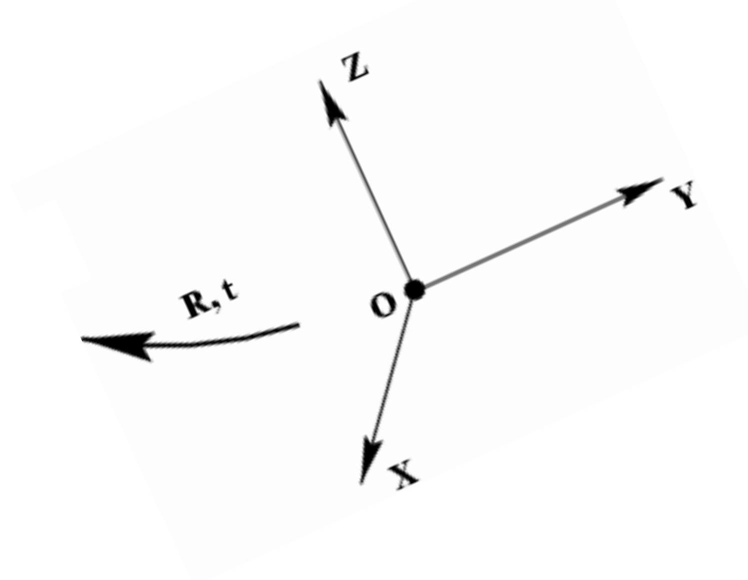


Source: S. Lazebnik

Review: Pinhole camera model



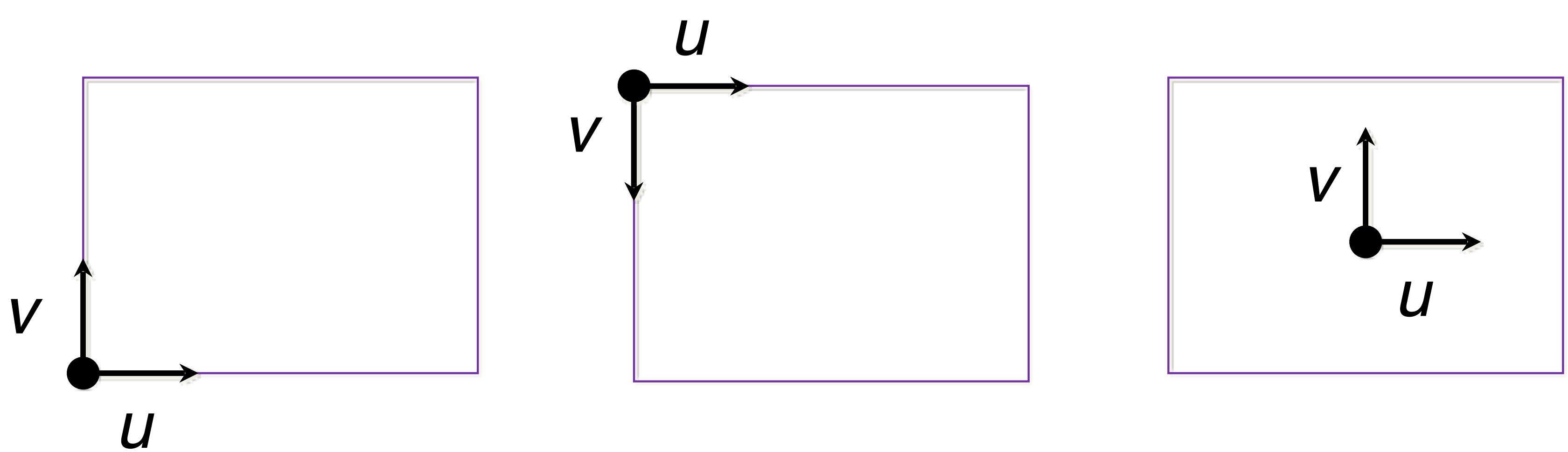
world coordinate system



Source: S. Lazebnik

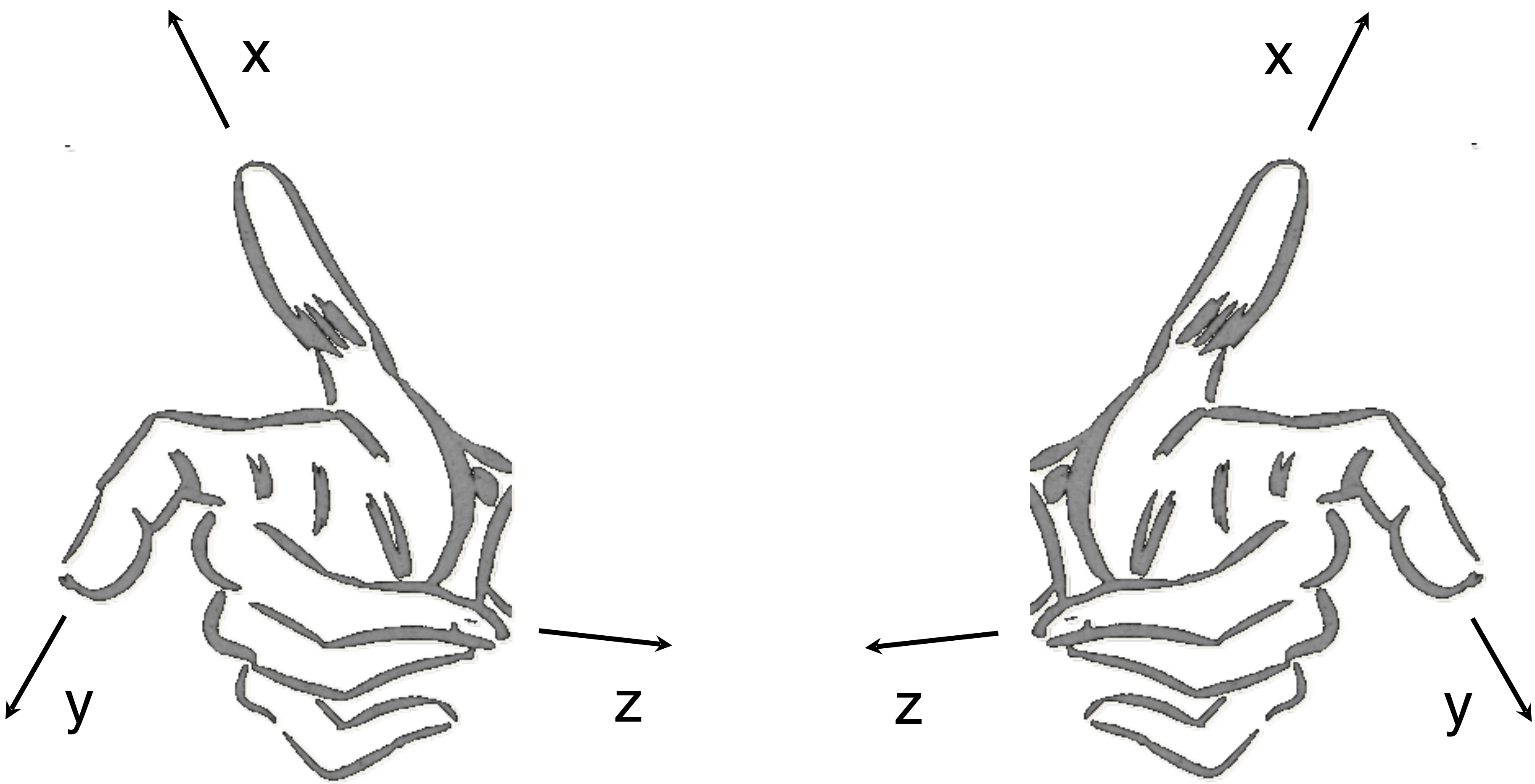
2D Coordinate Systems

- *y* axis up vs. *y* axis down
- Origin at center vs. corner
- Will often write (*u*, *v*) for image coordinates



3D Coordinate Systems

- Right-handed vs. left-handed



3D Geometry Basics

- 3D points = column vectors

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

- Transformations = pre-multiplied matrices

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

Rotation

- Rotation about the *z* axis

$$\mathbf{R}_z = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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

Arbitrary Rotation

- Rotate around *x*, *y*, then *z*:

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

- Don't do this! It's probably buggy!

  Compute simple matrices and multiply them…

Scale

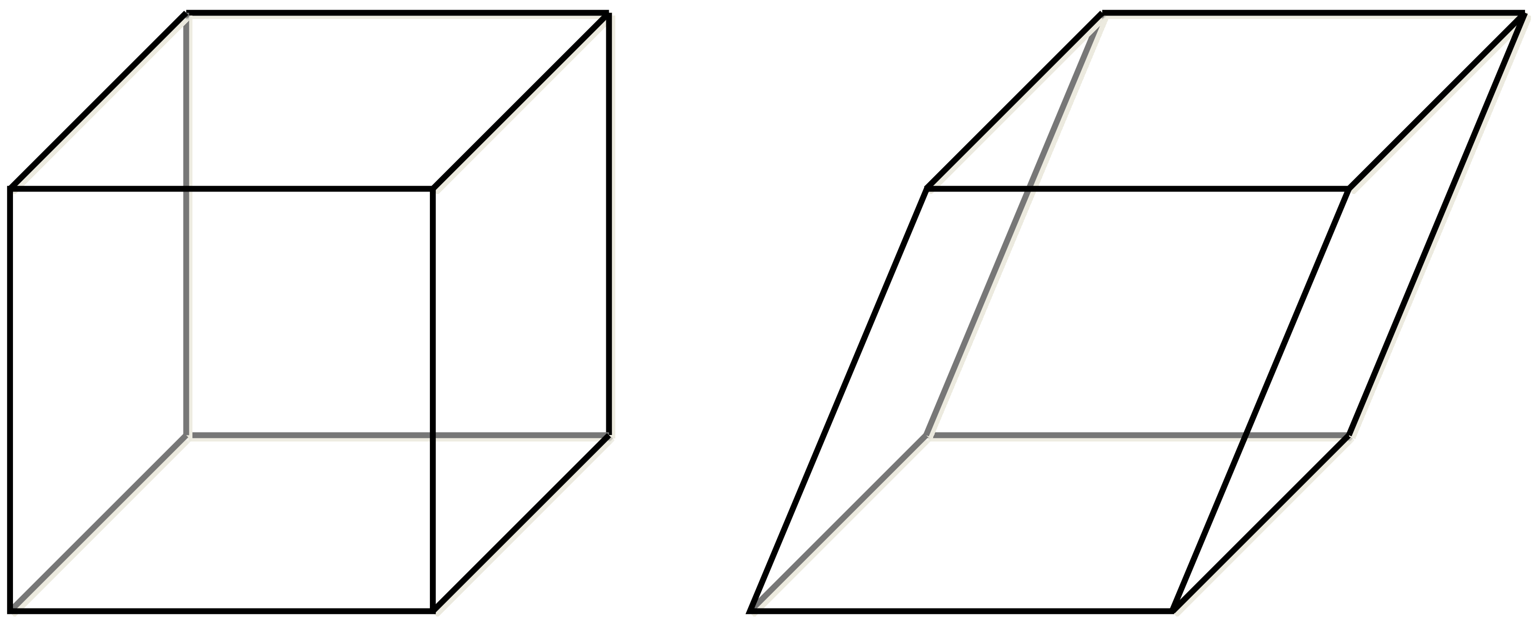
- Scale in *x*, *y*, *z*:

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

Shear

- Shear parallel to *xy* plane:

$$\boldsymbol{\sigma}_{xy} = \begin{pmatrix} 1 & 0 & \sigma_x \\ 0 & 1 & \sigma_y \\ 0 & 0 & 1 \end{pmatrix}$$



Translation

- Can translation be represented by multiplying by a 3×3 matrix?
- No.
- Proof:

$$\forall \mathbf{A}: \quad \mathbf{A}\vec{0} = \vec{0}$$

Homogeneous Coordinates

- Add a fourth dimension to each point:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

- To get "real" (3D) coordinates, divide by *w*:

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \to \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}$$

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?

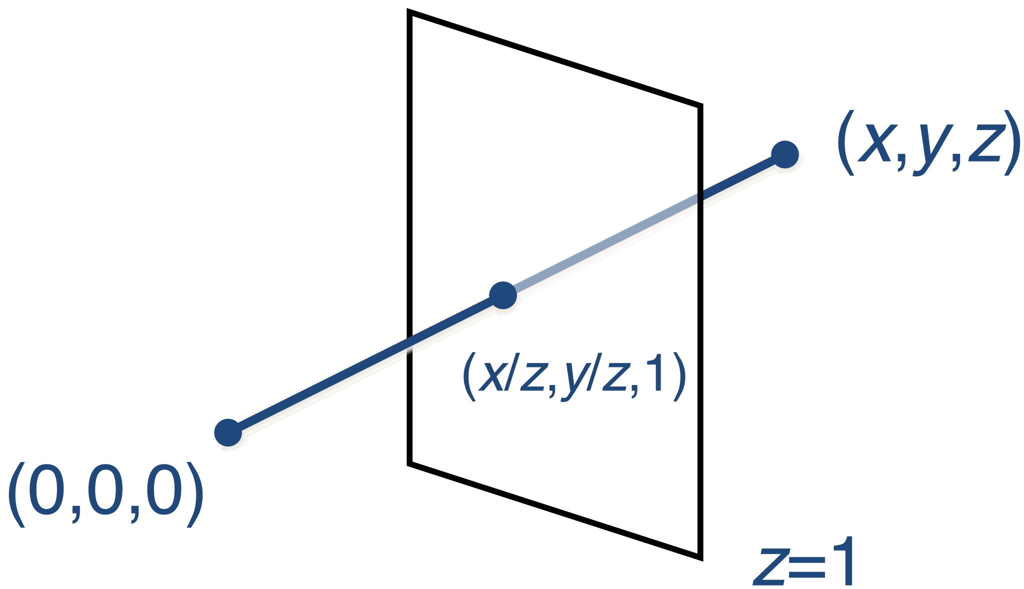
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z \end{pmatrix}$$

- After divide,

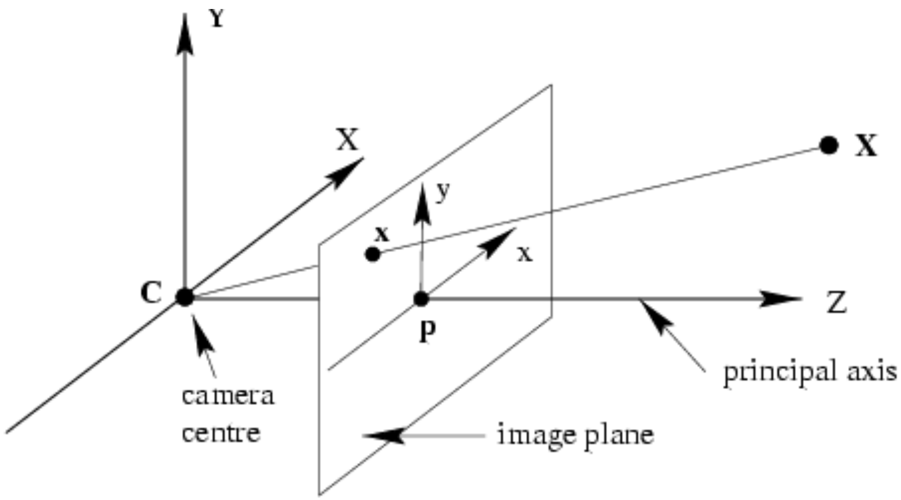
$$\begin{pmatrix} x \\ y \\ z \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix}$$

Perspective Projection

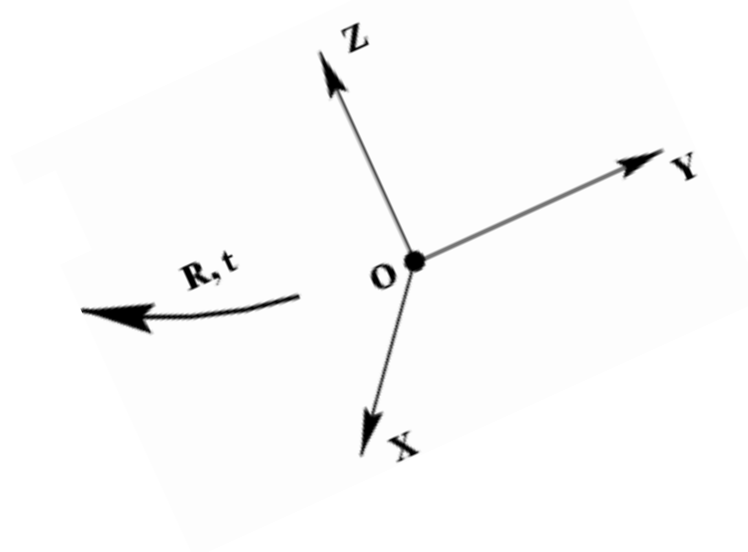
- This is projection onto the z=1 plane



Review: Pinhole camera model

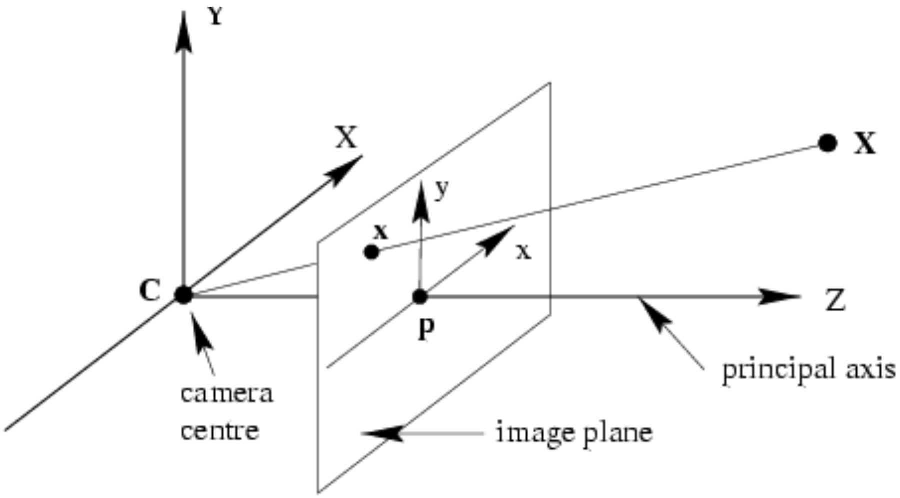


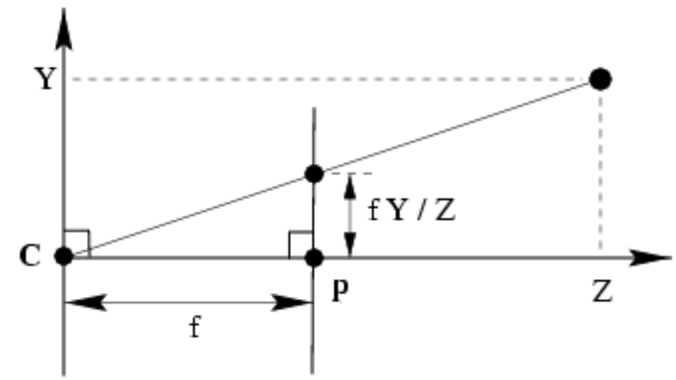
world coordinate system



Source: S. Lazebnik

Review: Pinhole camera model





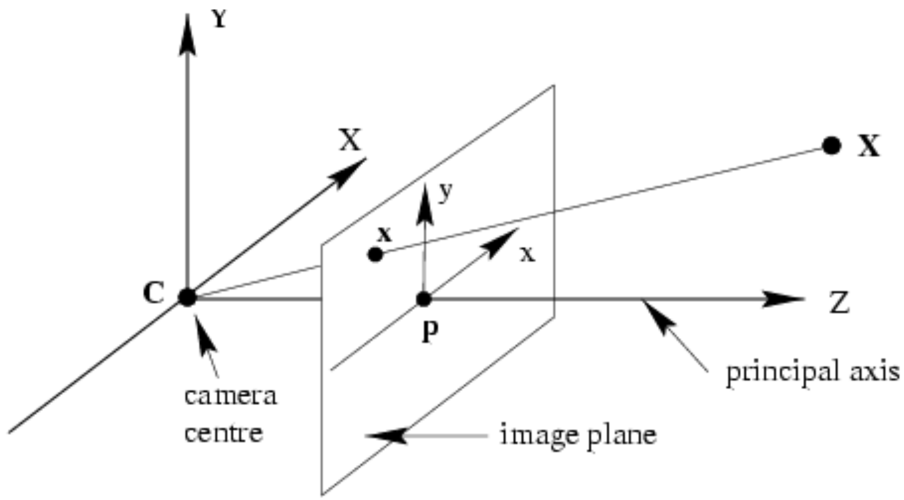
$$(X, Y, Z) \mapsto (f\,X/Z,\ f\,Y/Z)$$

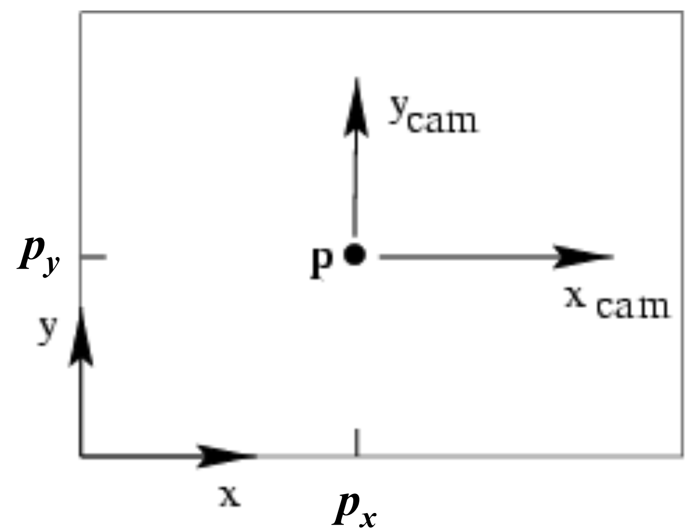
$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

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

Source: S. Lazebnik

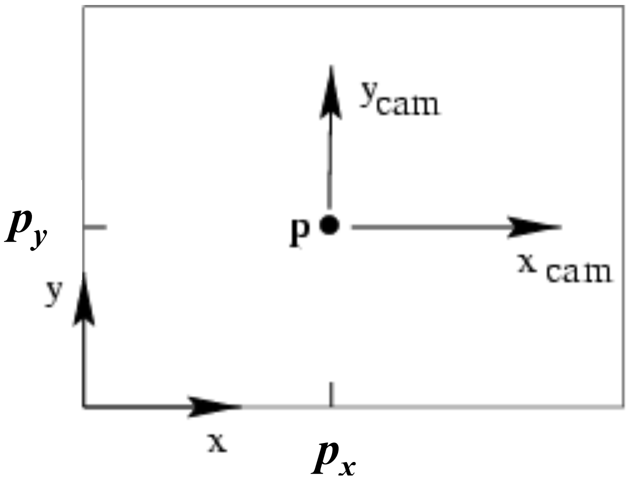
Change #1: Principal point offset





Source: S. Lazebnik

Change #1: Principal point offset



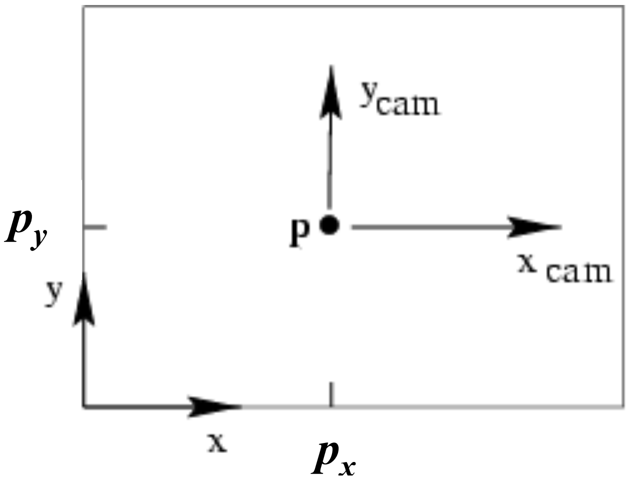
We want the principal point to map to (p_x, p_y) instead of (0,0)

$$(X, Y, Z) \mapsto (f X / Z + p_x, f Y / Z + p_y)$$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} f X + Z p_x \\ f Y + Z p_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Source: S. Lazebnik

Change #1: Principal point offset



principal point: (p_x, p_y)

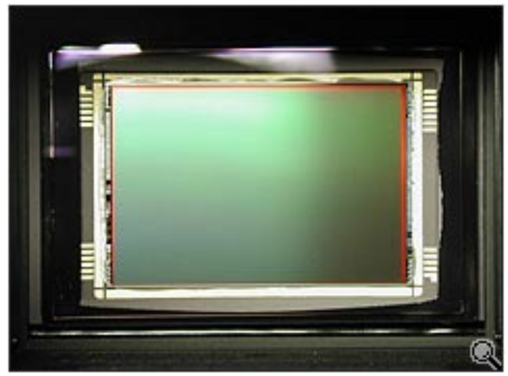
$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$ calibration matrix

$$\mathrm{P} = \mathrm{K}\left[\mathrm{I} \mid 0\right]$$

Source: S. Lazebnik

Change #2: Pixel coordinates



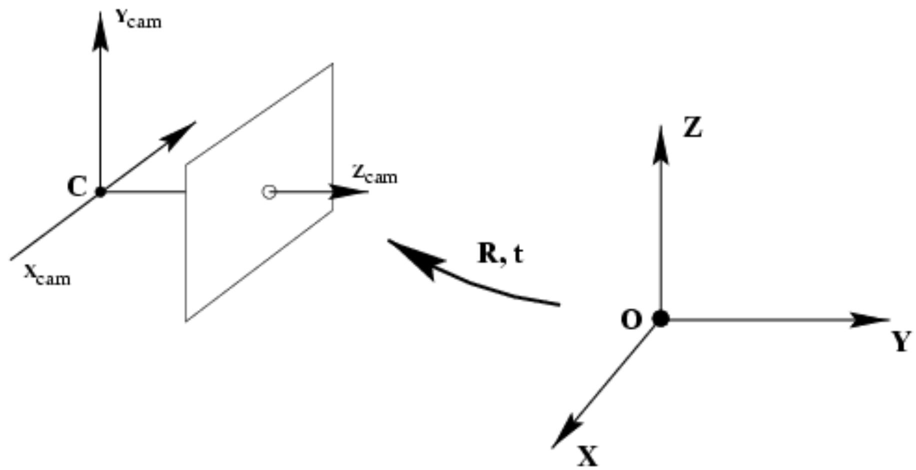
Pixel size: $\frac{1}{m_x} \times \frac{1}{m_y}$

$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}$$

pixels/m — m — pixels

Source: S. Lazebnik

Change #3: Camera rotation and translation



- Conversion from world to camera coordinate system (in non-homogeneous coordinates):

$$\tilde{X}_{cam} = R\left(\tilde{X} - \tilde{C}\right)$$

coords. of point in camera frame

coords. of a point in world frame

coords. of camera center in world frame

Source: S. Lazebnik

Camera projection matrix

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

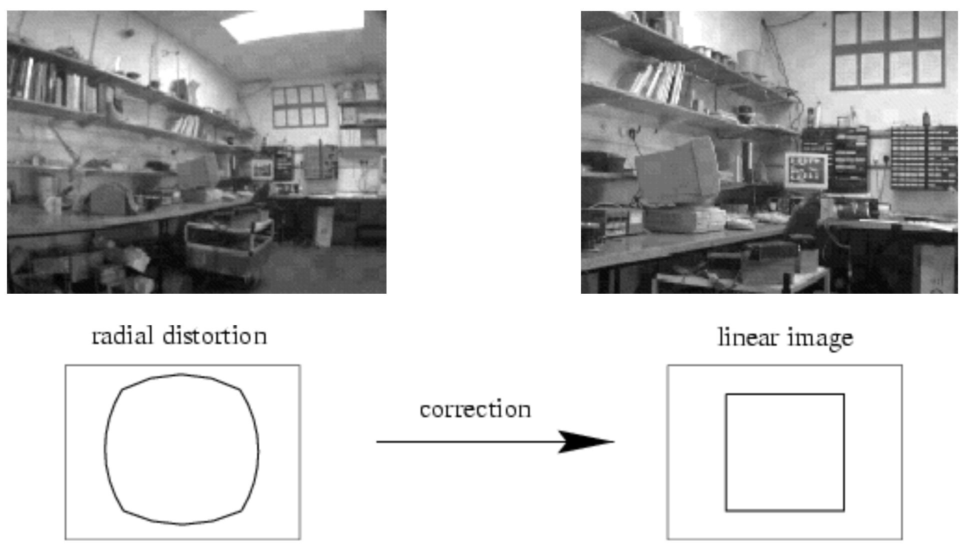
$$\begin{bmatrix}\lambda x \\ \lambda y \\ \lambda\end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1\end{bmatrix}$$

```
P = K[R t]
```

Camera parameters $P = K[R \ t]$

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - *Skew (non-rectangular pixels)*
  - *Radial distortion*

$$\mathbf{K} = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}$$



Source: S. Lazebnik

Camera parameters $P = K[R \ t]$

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - *Skew (non-rectangular pixels)*
  - *Radial distortion*
- Extrinsic parameters
  - Rotation and translation relative to world coordinate system

How many parameters here?

Source: S. Lazebnik

Camera calibration basics

Camera calibration

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\begin{bmatrix}\lambda x \\ \lambda y \\ \lambda\end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}\begin{bmatrix}X \\ Y \\ Z \\ 1\end{bmatrix}$$

General camera model

- Multiply all these matrices together
- Don't care about "*z*" after transformation

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

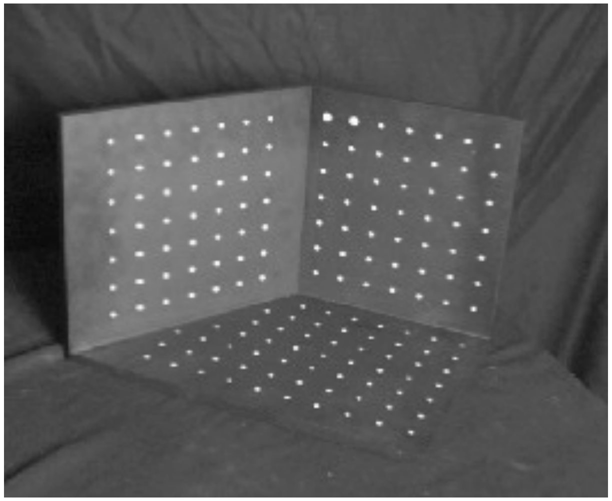
- Scale ambiguity → 11 free parameters
  - 6 extrinsic, 5 intrinsic

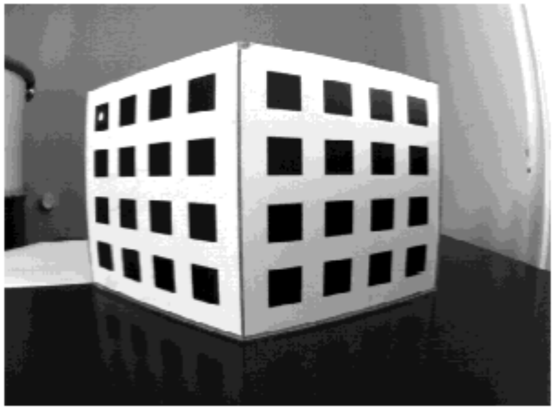
Camera Calibration

- 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

- General idea: place "calibration object" with known geometry in the scene
- Get correspondences
- Solve for mapping from scene to image





The Opti-CAL Calibration Target Image

Camera Calibration – linear system

- Given:
  - 3D ↔ 2D correspondences
  - General perspective camera model
- Write equations:

$$\frac{ax_1 + by_1 + cz_1 + d}{ix_1 + jy_1 + kz_1 + l} = u_1$$

$$\frac{ex_1 + fy_1 + gz_1 + h}{ix_1 + jy_1 + kz_1 + l} = v_1$$

$$\vdots$$

Camera Calibration – linear system

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & -u_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & -v_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 & -u_2 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_2x_2 & -v_2y_2 & -v_2z_2 & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ \vdots \\ l \end{pmatrix} = \vec{\mathbf{0}}$$

- Overconstrained (more equations than unknowns)
- Underconstrained (rank deficient matrix – any multiple of a solution, including 0, is also a solution)

Camera Calibration – linear system

- 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 – linear system

- Minimize $|Ax|^2$ subject to $|x|^2=1$
- $|Ax|^2 = (Ax)^T(Ax) = (x^TA^T)(Ax) = x^T(A^TA)x$
- Expand x in terms of eigenvectors of A^TA:

$$x = \mu_1 e_1 + \mu_2 e_2 + \ldots$$

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

$$|x|^2 = \mu_1^2+\mu_2^2+\ldots$$

Camera Calibration – linear system

- To minimize

$$\lambda_1\mu_1^2+\lambda_2\mu_2^2+\ldots$$

  subject to

$$\mu_1^2+\mu_2^2+\ldots = 1$$

  set $\mu_{min}= 1$ and all other $\mu_i=0$

- Thus, least squares solution is eigenvector of A^TA corresponding to minimum (nonzero) eigenvalue

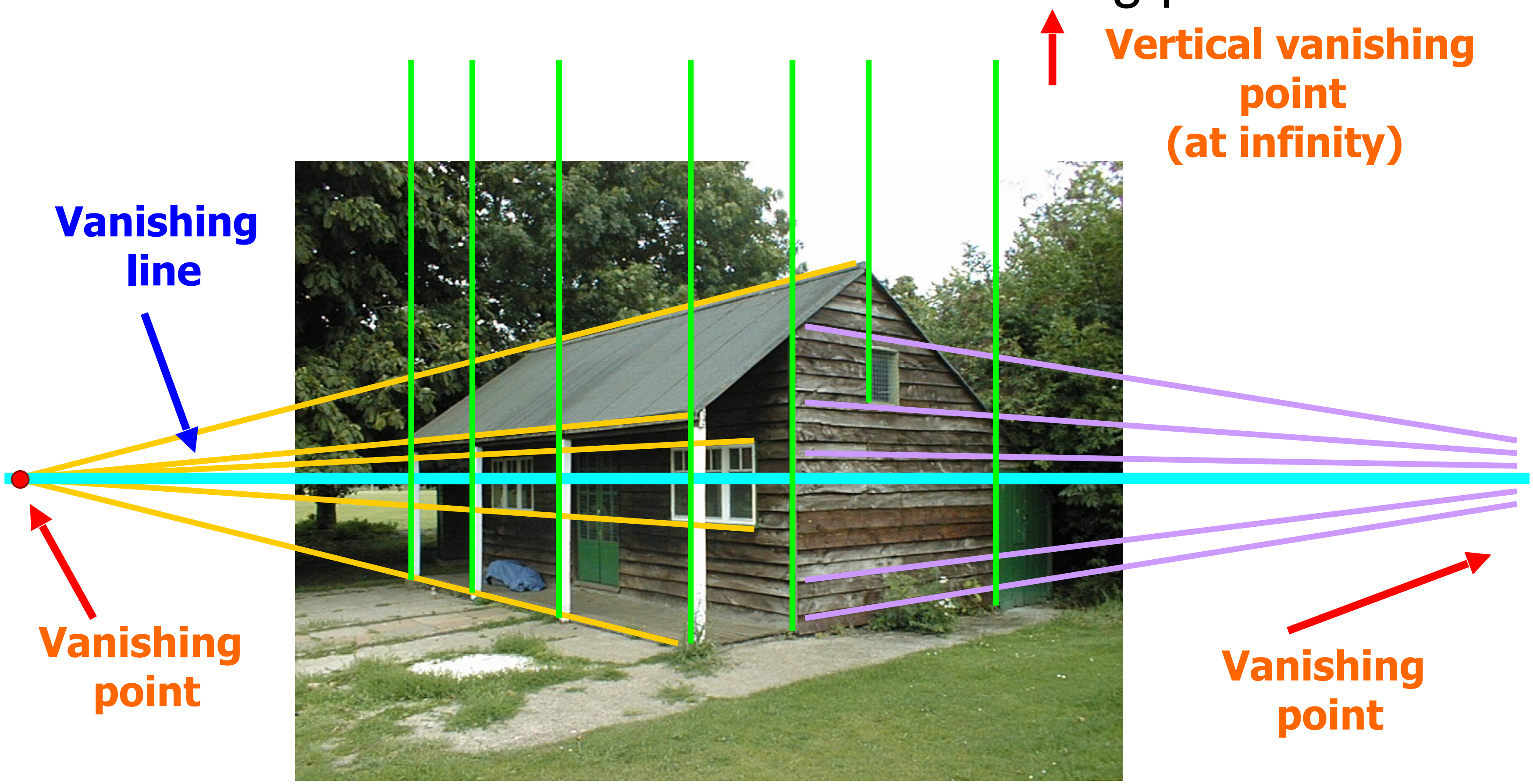
Camera calibration: Linear method

- Advantages: easy to formulate and solve
- Disadvantages
  - Doesn't directly tell you camera parameters
  - Doesn't model radial distortion
  - Can't impose constraints, such as known focal length and orthogonality

- Non-linear methods are preferred
  - Define error as squared distance between projected points and measured points
  - Minimize error using Newton's method or other non-linear optimization

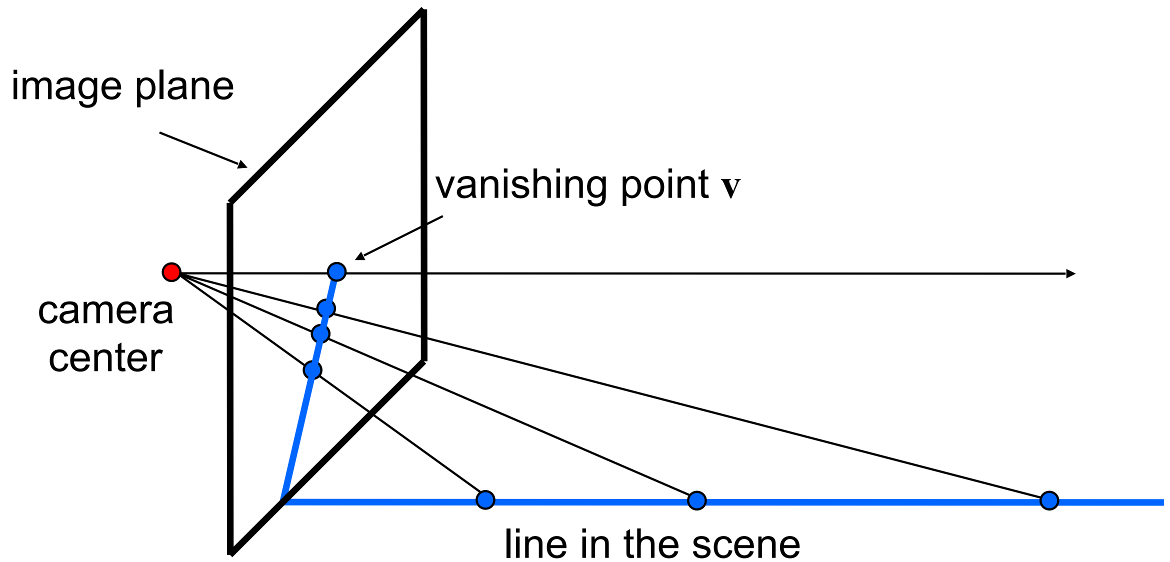
Camera calibration without known coordinates

- What if world coordinates of reference 3D points are not known?
- We can use scene features such as vanishing points



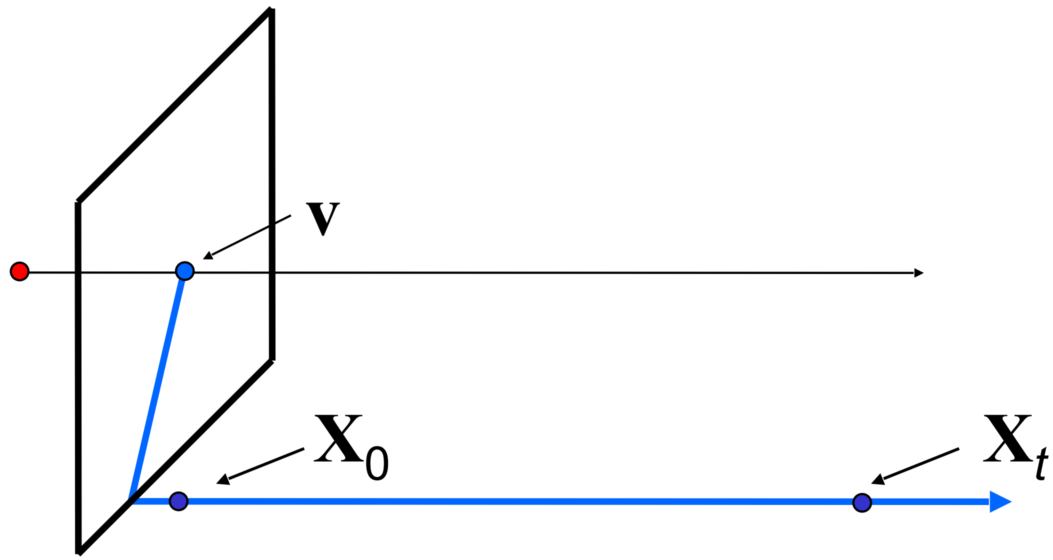
Slide from Efros, Photo from Criminisi

Recall: Vanishing points



- All lines having the same direction share the same vanishing point

Computing vanishing points

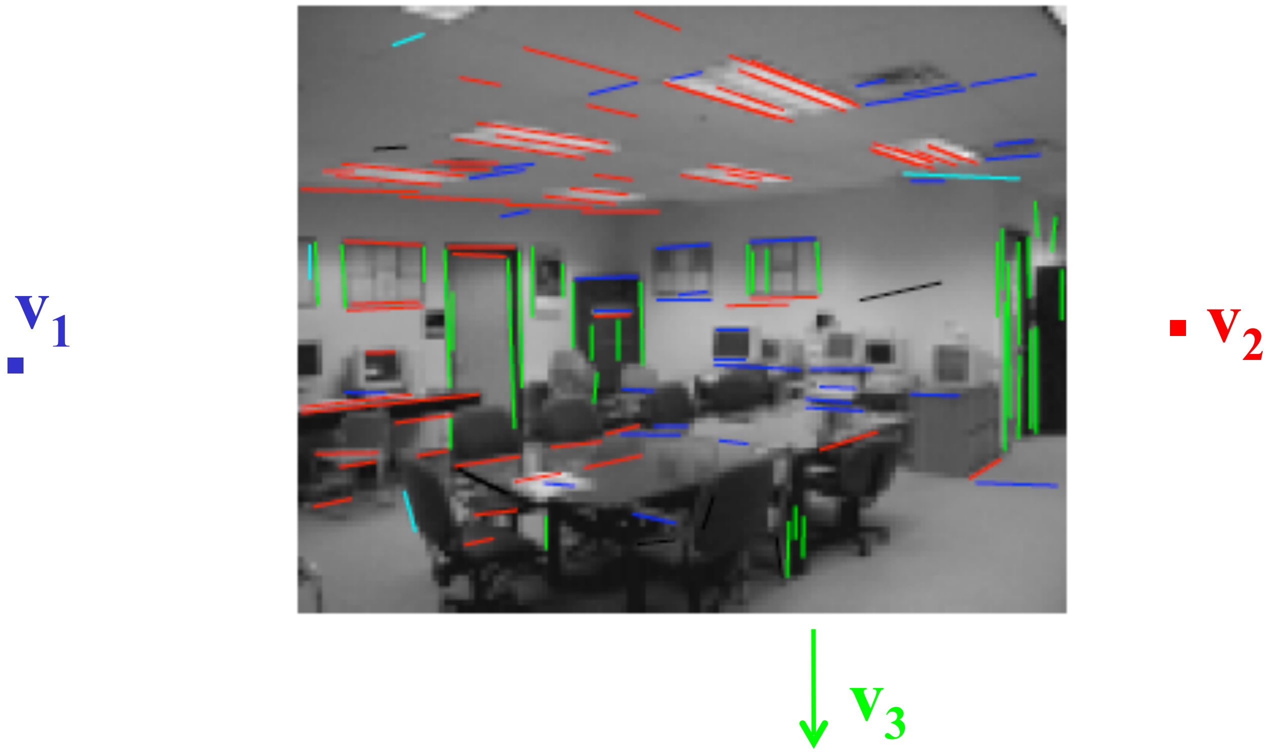


$$\mathbf{X}_t = \begin{bmatrix} x_0 + td_1 \\ y_0 + td_2 \\ z_0 + td_3 \\ 1 \end{bmatrix} = \begin{bmatrix} x_0/t + d_1 \\ y_0/t + d_2 \\ z_0/t + d_3 \\ 1/t \end{bmatrix} \qquad \mathbf{X}_\infty = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ 0 \end{bmatrix}$$

- $\mathbf{X}_\infty$ is a *point at infinity,* $\mathbf{v}$ is its projection: $\mathbf{v} = \mathbf{P}\mathbf{X}_\infty$
- The vanishing point depends only on *line direction*
- All lines having direction $\mathbf{D}$ intersect at $\mathbf{X}_\infty$

Calibration from vanishing points

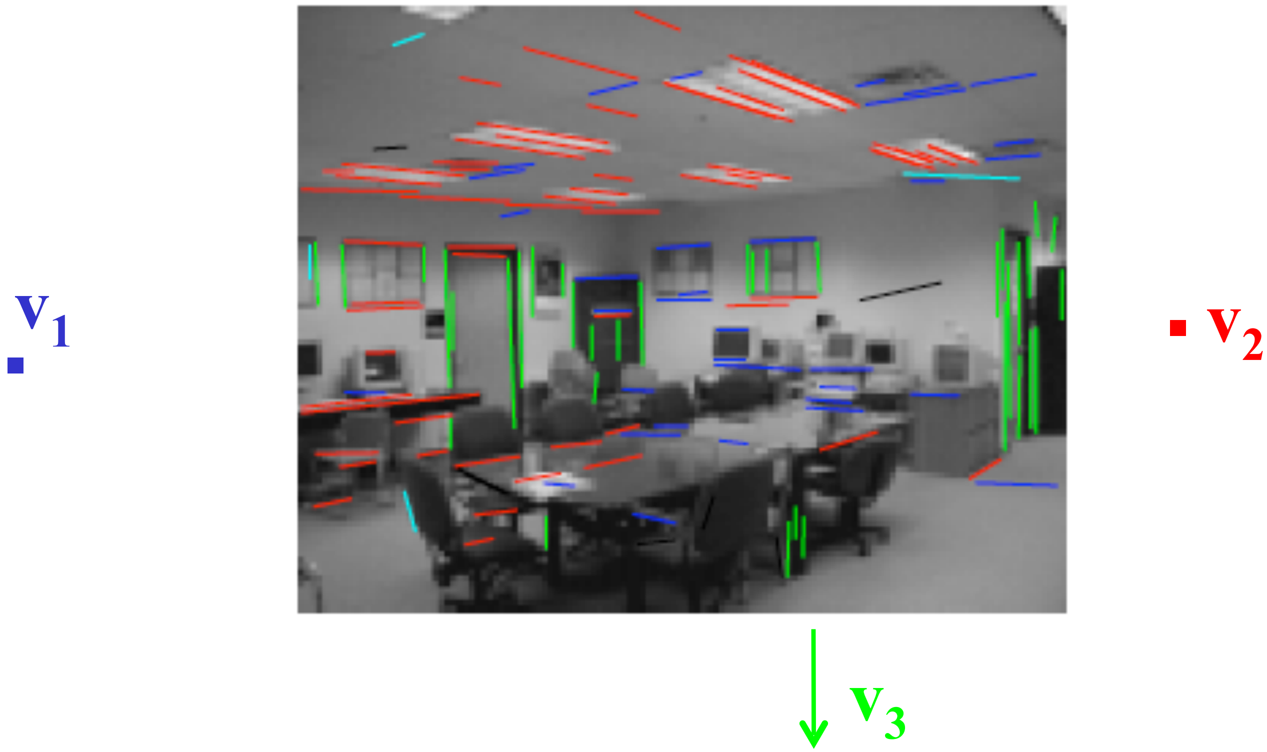
- Consider a scene with three orthogonal vanishing directions:



- Note: $\mathbf{v}_1$, $\mathbf{v}_2$ are *finite* vanishing points and $\mathbf{v}_3$ is an *infinite* vanishing point

Calibration from vanishing points

- Consider a scene with three orthogonal vanishing directions:



- We can align the world coordinate system with these directions

Calibration from vanishing points

$$\mathbf{P} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix}$$

- $\mathbf{p}_1 = \mathbf{P}(1,0,0,0)^T$ – the vanishing point in the x direction
- Similarly, $\mathbf{p}_2$ and $\mathbf{p}_3$ are the vanishing points in the y and z directions
- $\mathbf{p}_4 = \mathbf{P}(0,0,0,1)^T$ – projection of the origin of the world coordinate system
- Problem: we can only know the four columns up to independent scale factors, additional constraints needed to solve for them

Calibration from vanishing points

- Let us align the world coordinate system with three orthogonal vanishing directions in the scene:

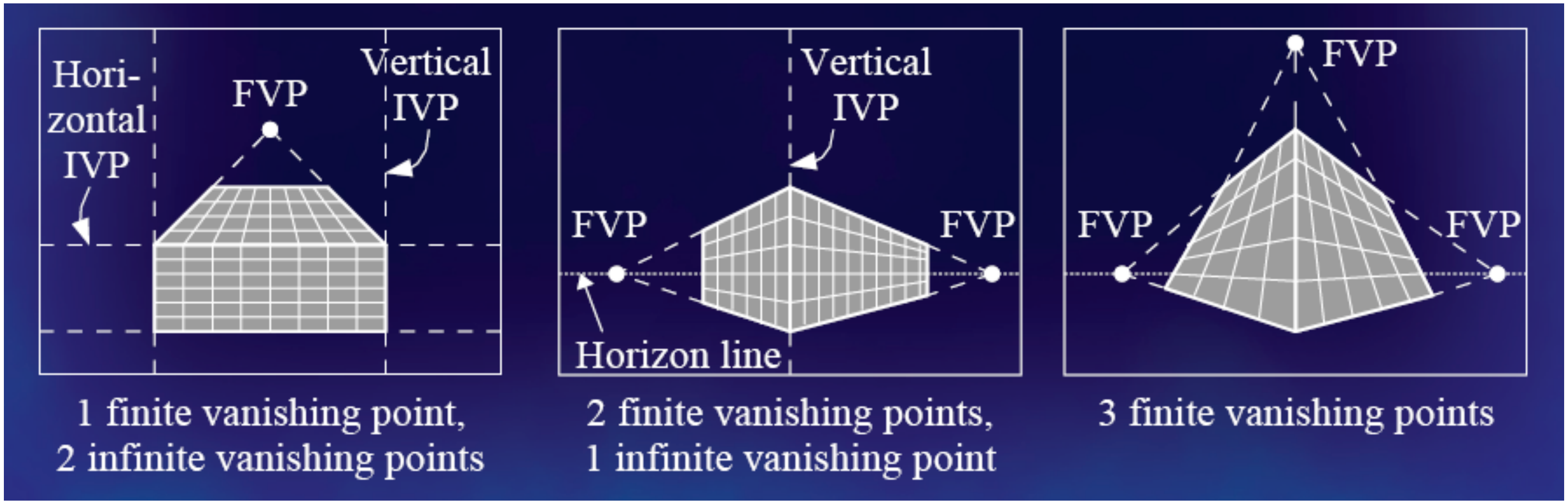
$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \lambda_i \mathbf{v}_i = \mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right] \begin{bmatrix} \mathbf{e}_i \\ 0 \end{bmatrix} = \mathbf{K}\mathbf{R}\mathbf{e}_i$$

$$\mathbf{e}_i = \lambda_i \mathbf{R}^T \mathbf{K}^{-1} \mathbf{v}_i, \quad \mathbf{e}_i^T \mathbf{e}_j = 0$$

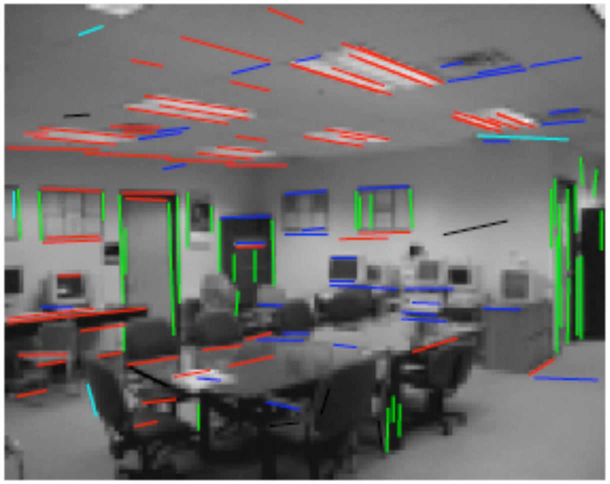
$$\mathbf{v}_i^T \mathbf{K}^{-T} \mathbf{R}\mathbf{R}^T \mathbf{K}^{-1} \mathbf{v}_j = \mathbf{v}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{v}_j = 0$$

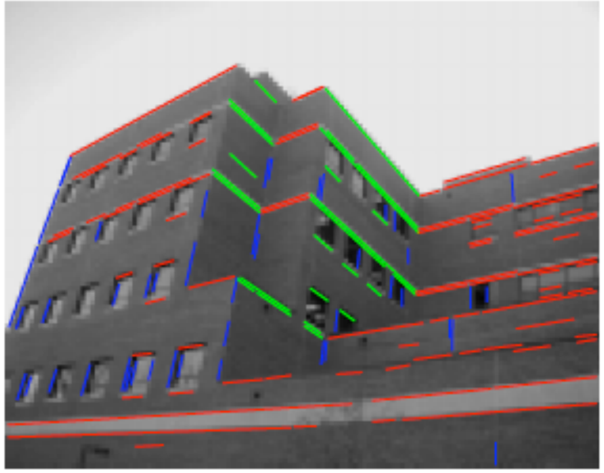
- Each pair of vanishing points gives us a constraint on the focal length and principal point (assuming zero skew and unit aspect ratio).

Calibration from vanishing points



Cannot recover focal length, principal point is the third vanishing point





Can solve for focal length, principal point

Rotation from vanishing points

$$\lambda_i \mathbf{v}_i = \mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right]\begin{bmatrix} \mathbf{e}_i \\ 0 \end{bmatrix} = \mathbf{K}\mathbf{R}\mathbf{e}_i$$

$$\lambda_i \mathbf{K}^{-1}\mathbf{v}_1 = \mathbf{R}\mathbf{e}_1 = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3]\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{r}_1$$

Thus,

$$\lambda_i \mathbf{K}^{-1}\mathbf{v}_i = \mathbf{r}_i.$$

Get λ_i by using the constraint $\|\mathbf{r}_i\|^2 = 1$.

Calibration from vanishing points: Summary

- Solve for K (focal length, principal point) using three orthogonal vanishing points
- Get rotation directly from vanishing points once calibration matrix is known

- Advantages
  - No need for calibration chart, 2D-3D correspondences
  - Could be completely automatic
- Disadvantages
  - Only applies to certain kinds of scenes
  - Inaccuracies in computation of vanishing points
  - Problems due to infinite vanishing points

Hold this thought…

- Will come back to calibration when discussing structure from motion
- But first, let’s talk about 2+ cameras

Recall: epipolar geometry from last time

Binocular stereo

- Given a calibrated binocular stereo pair, fuse it to produce a depth image

image 1

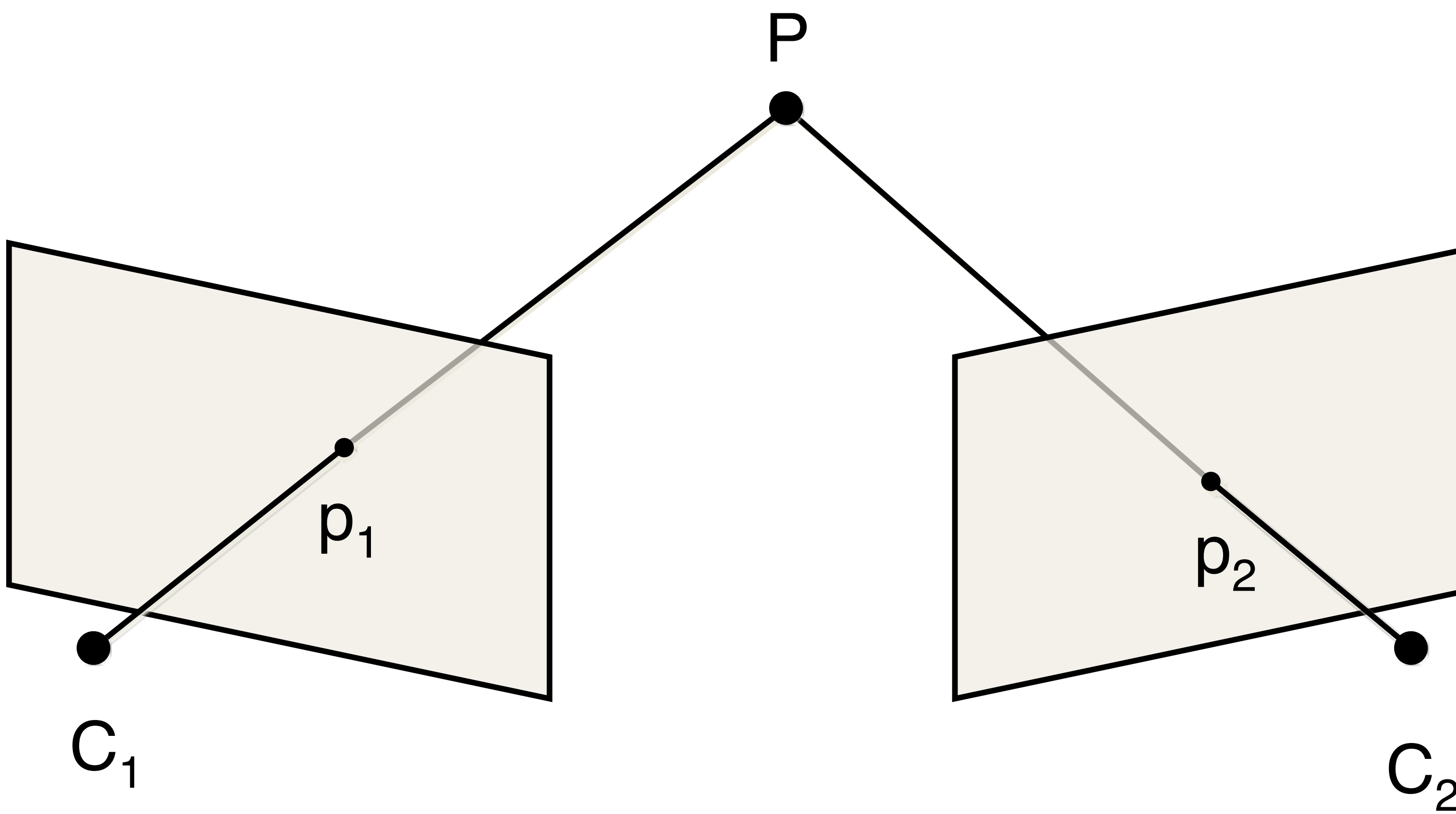
image 2

Dense depth map

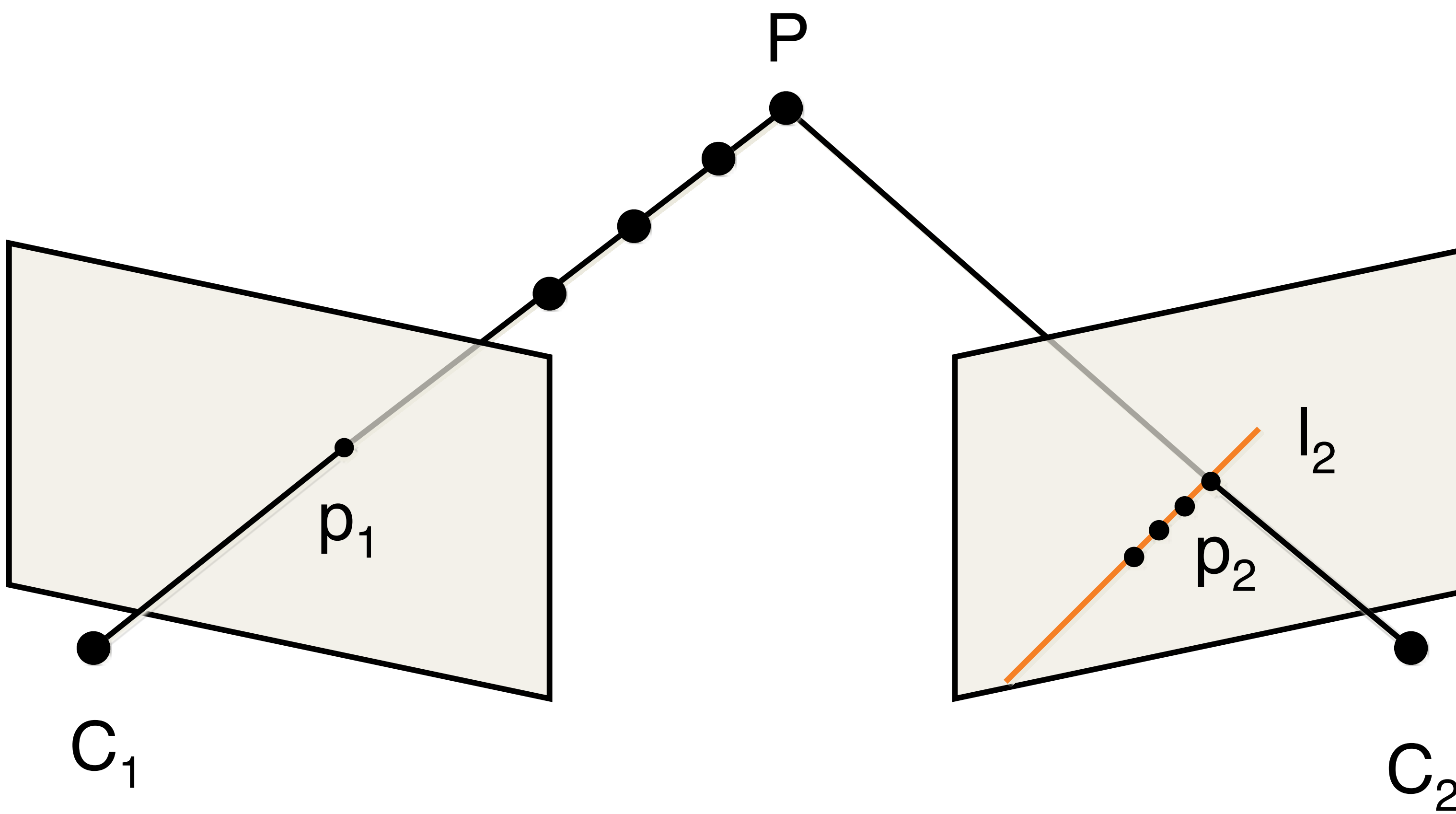
Multi-Camera Geometry

- Epipolar geometry – relationship between observed positions of points in multiple cameras
- Assume:
  - 2 cameras
  - Known intrinsics and extrinsics

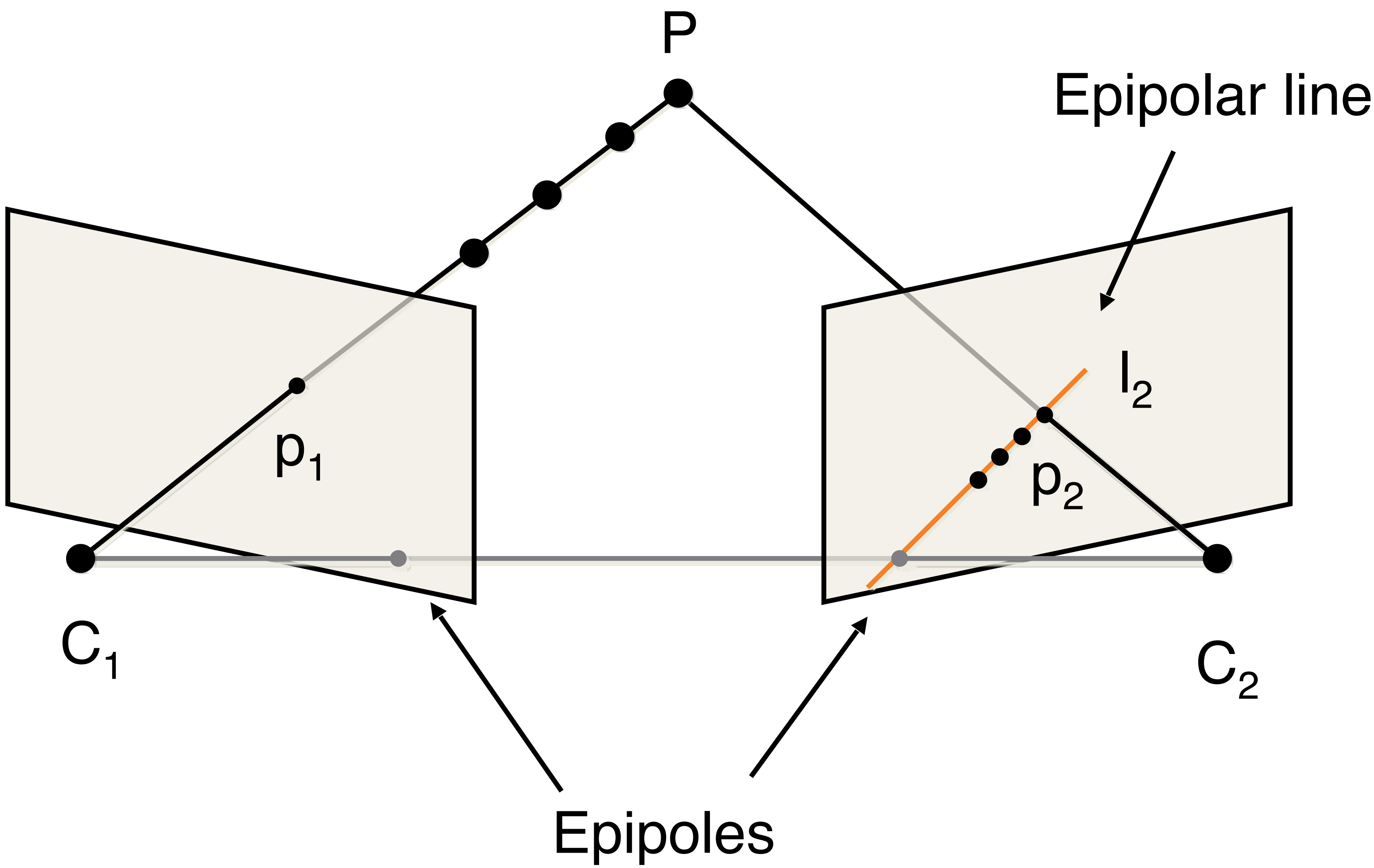
Epipolar Geometry



Epipolar Geometry

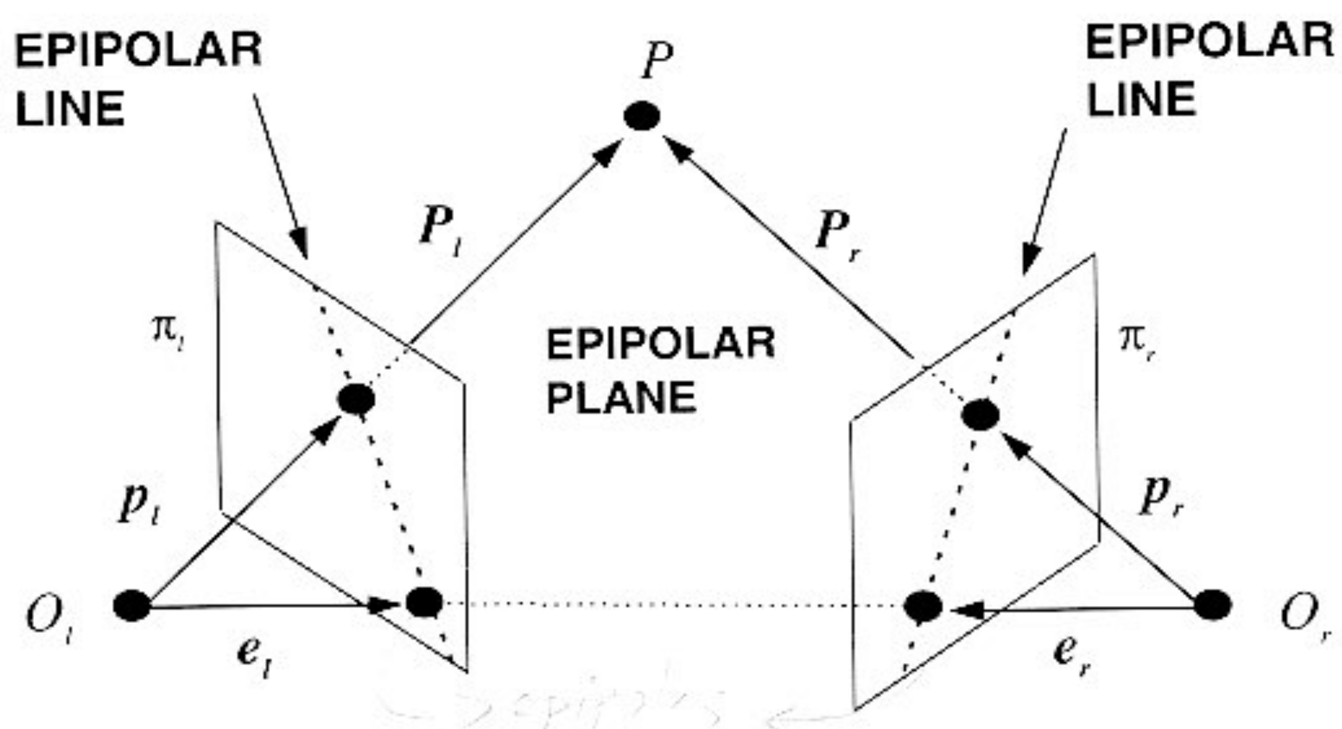


Epipolar Geometry



Epipolar Geometry

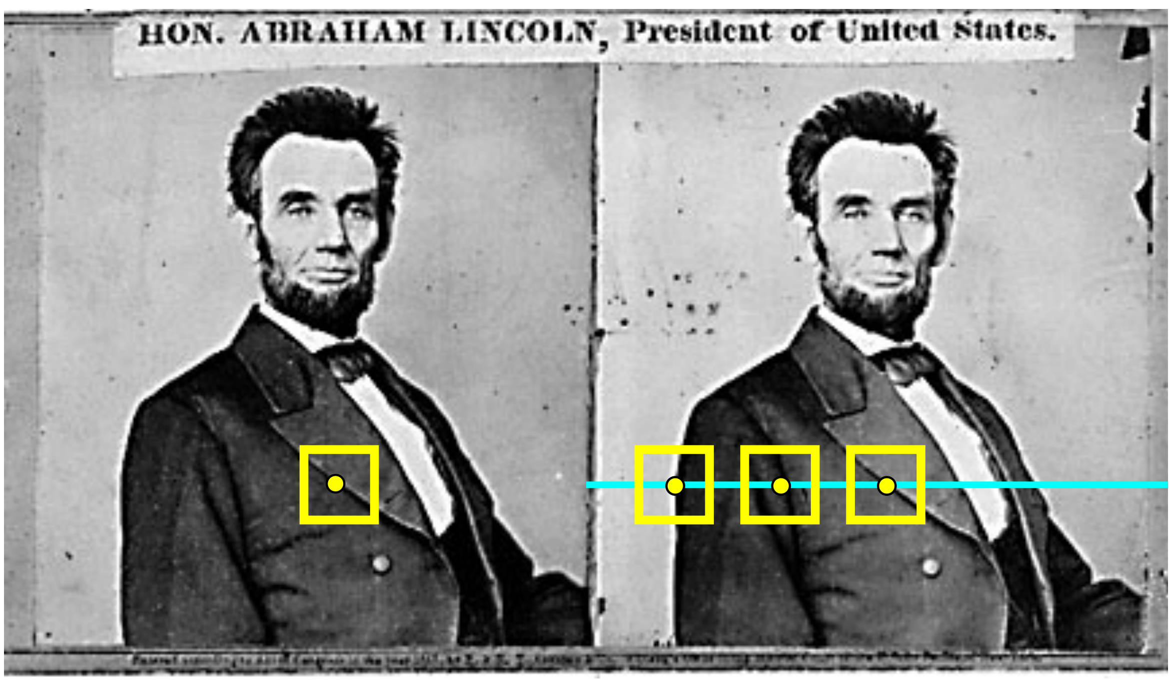
- Epipolar constraint: corresponding points must lie on conjugate epipolar lines
  - Search for correspondences becomes a 1-D problem



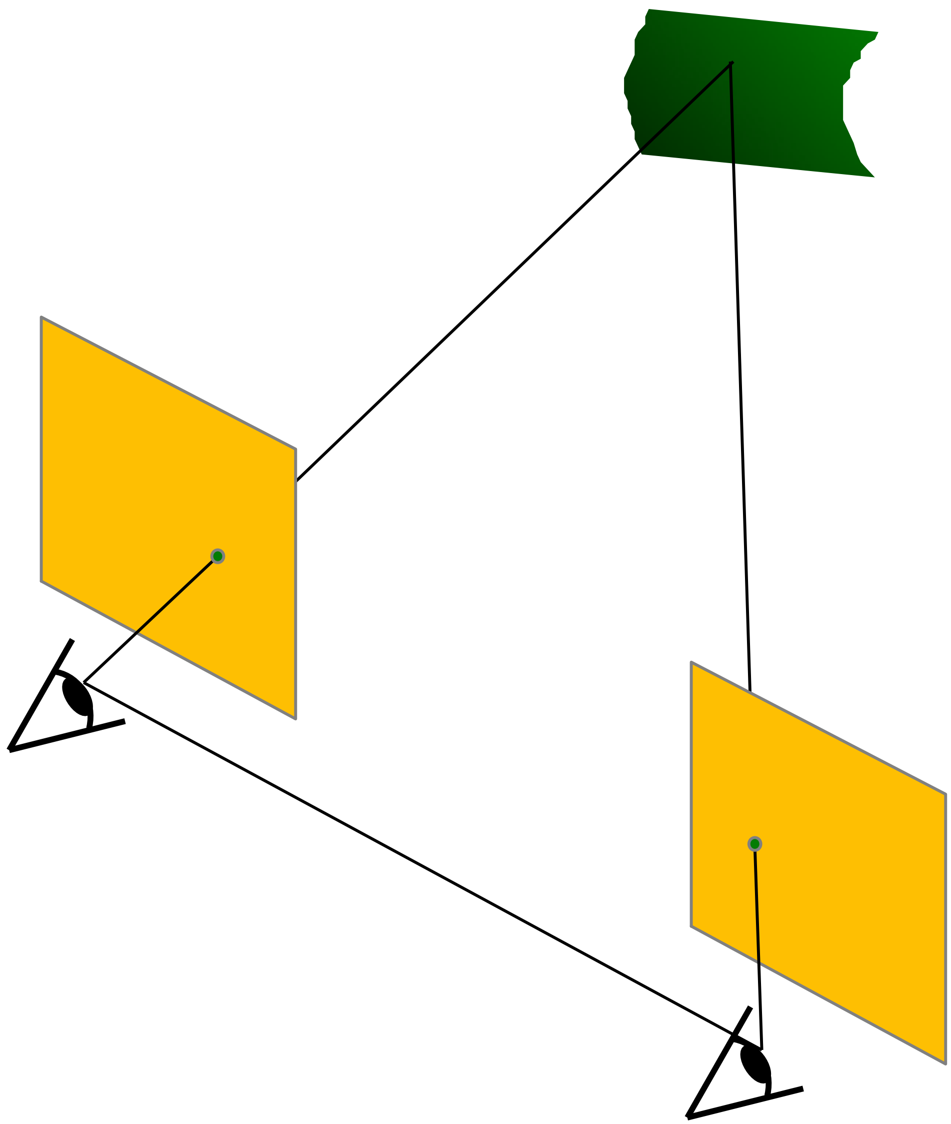
Basic stereo matching algorithm

- For each pixel in the first image
  - Find corresponding epipolar line in the right image
  - Examine all pixels on the epipolar line and pick the best match
  - Triangulate the matches to get depth information

- Simplest case: epipolar lines are corresponding scanlines
  - When does this happen?

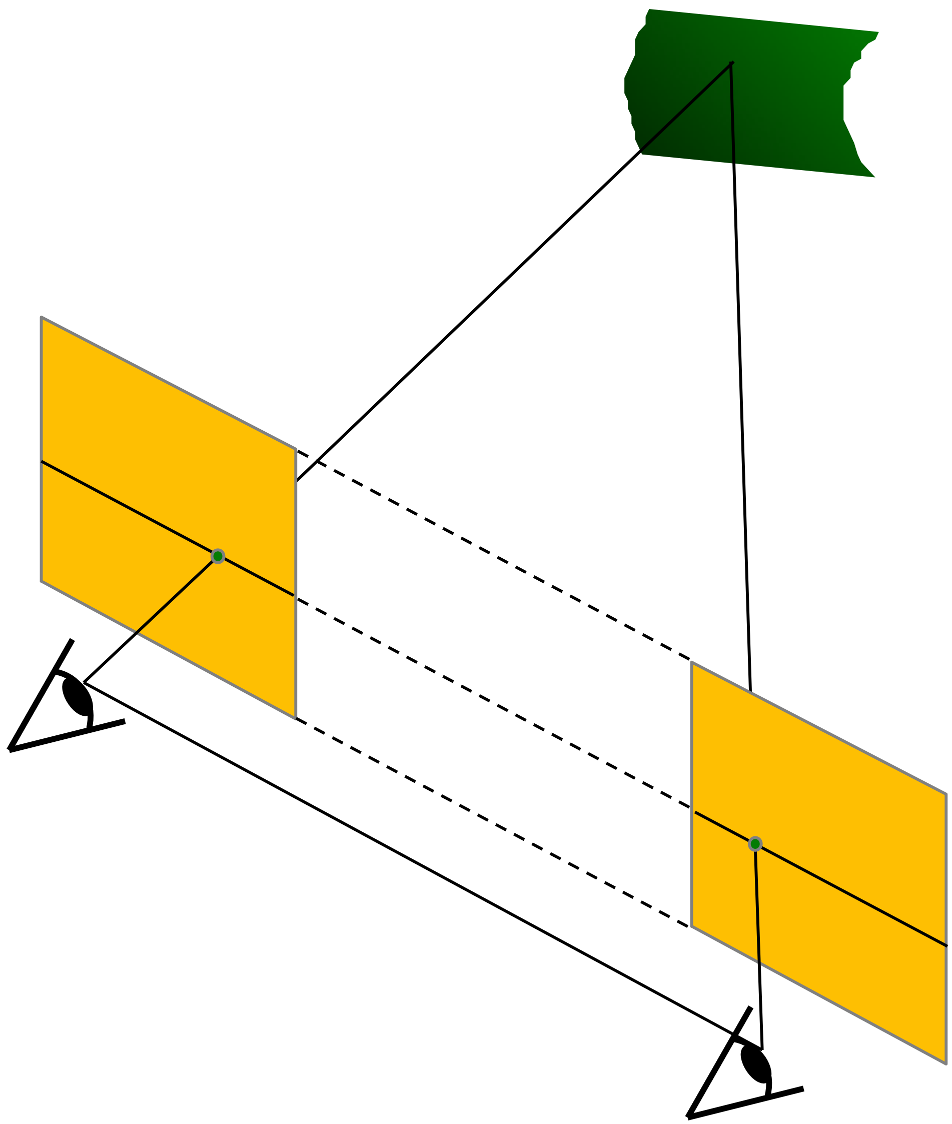


Simplest Case: Parallel images



- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same

Simplest Case: Parallel images

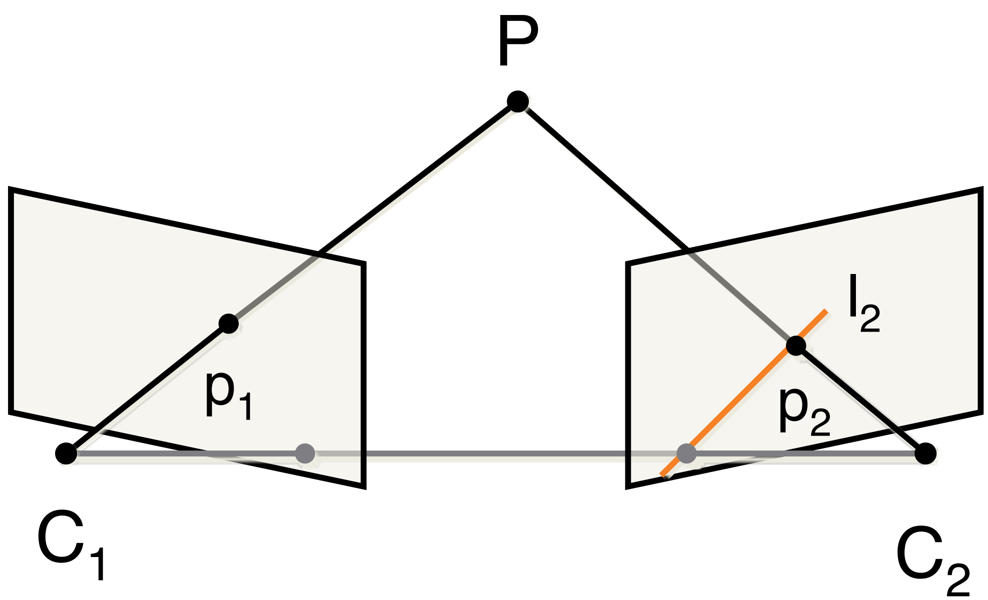


- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the horizontal scan lines of the images

What if images are not aligned?

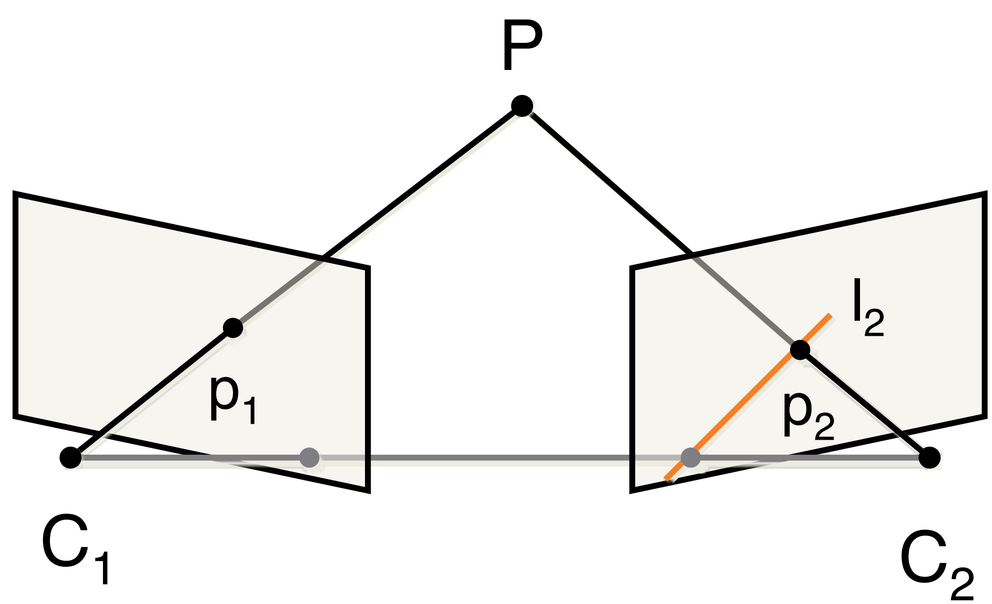
Epipolar Geometry

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



Epipolar Geometry

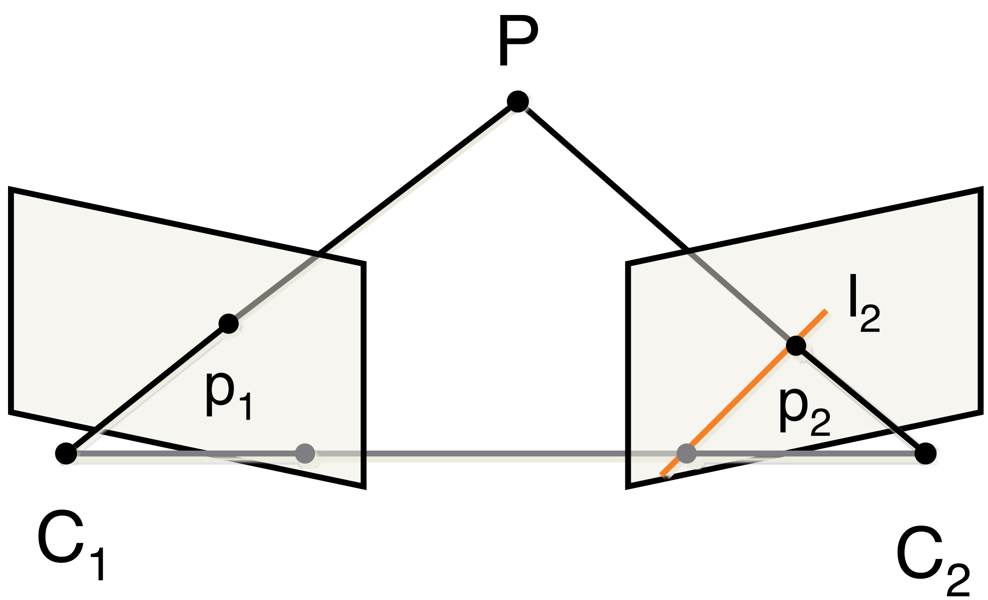
- Work in coordinate frame of C_1
- Normal of plane is $T \times Rp_2$, where T is relative translation, R is relative rotation



Epipolar Geometry

- p_1 is perpendicular to this normal:

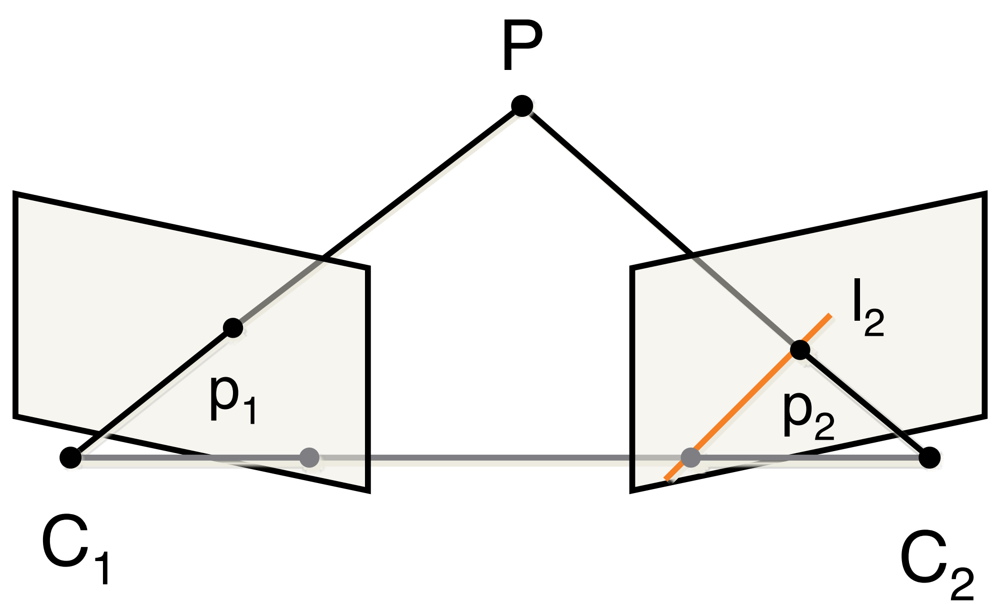
$$p_1 \bullet (T \times Rp_2) = 0$$



Epipolar Geometry

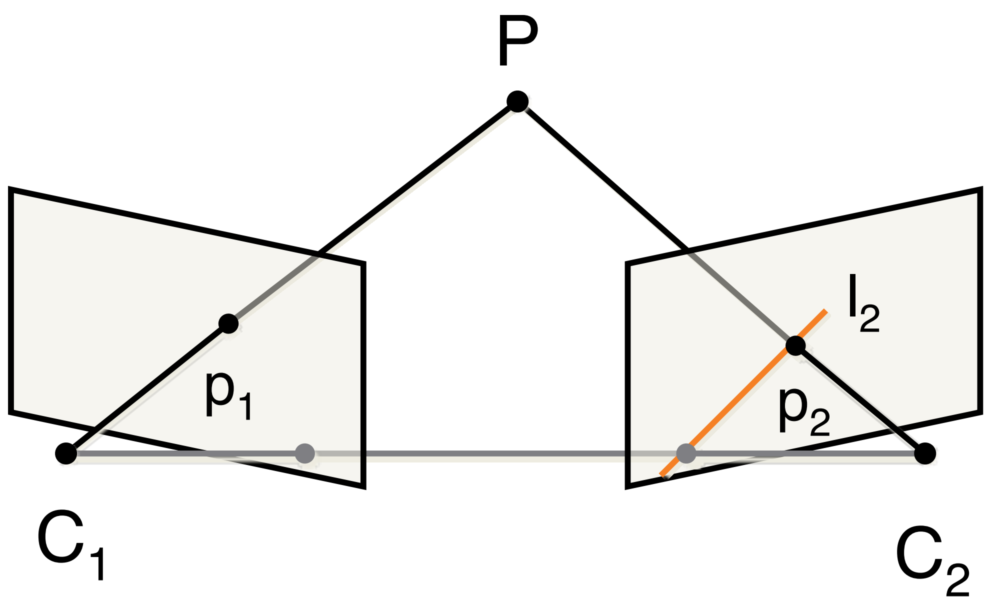
- Write cross product as matrix multiplication

$$\vec{T} \times x = \mathbf{T}^{\times} x, \qquad \mathbf{T}^{\times} = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$



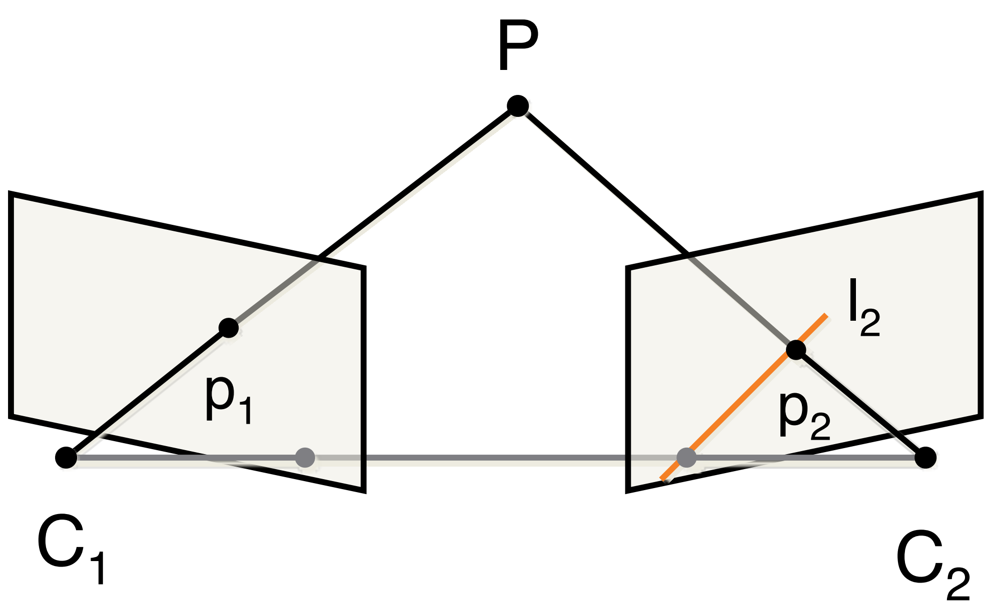
Epipolar Geometry

- $p_1 \bullet T_{\times} R\, p_2 = 0 \quad \Rightarrow \quad p_1^T E\, p_2 = 0$
- E is the essential matrix

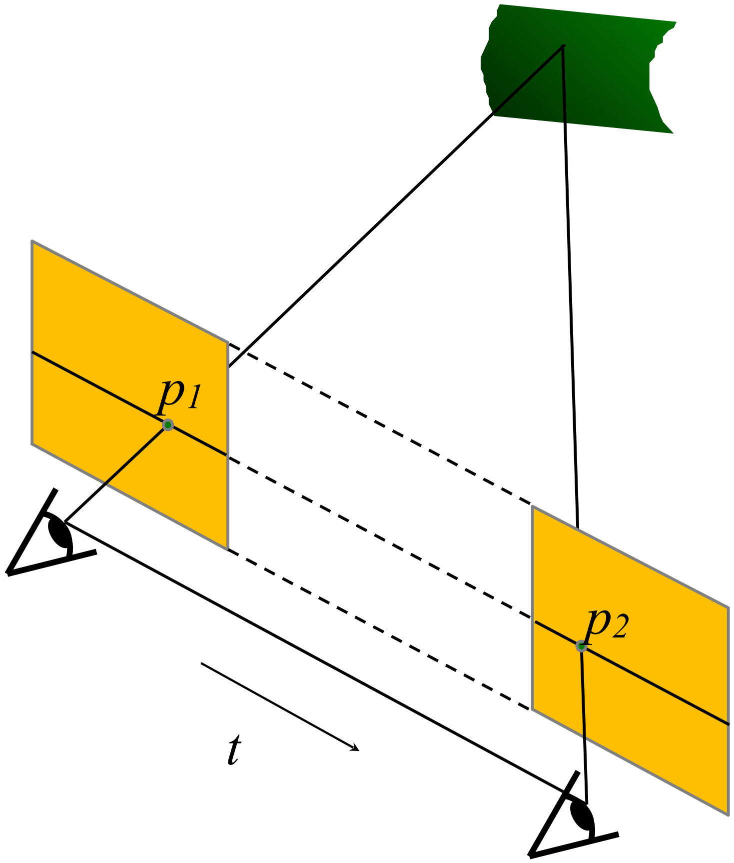


Essential Matrix

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



Concrete example: parallel images

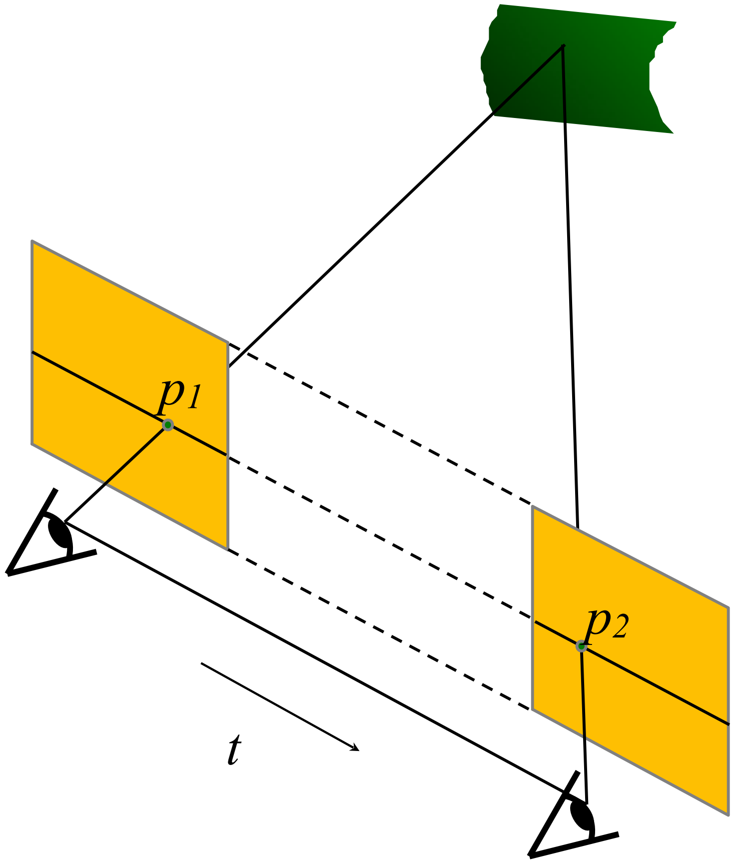


- Rotation?
  - Identity
- Translation?

$$T = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \text{-t} \\ 0 & \text{t} & 0 \end{bmatrix}$$

Concrete example: parallel images



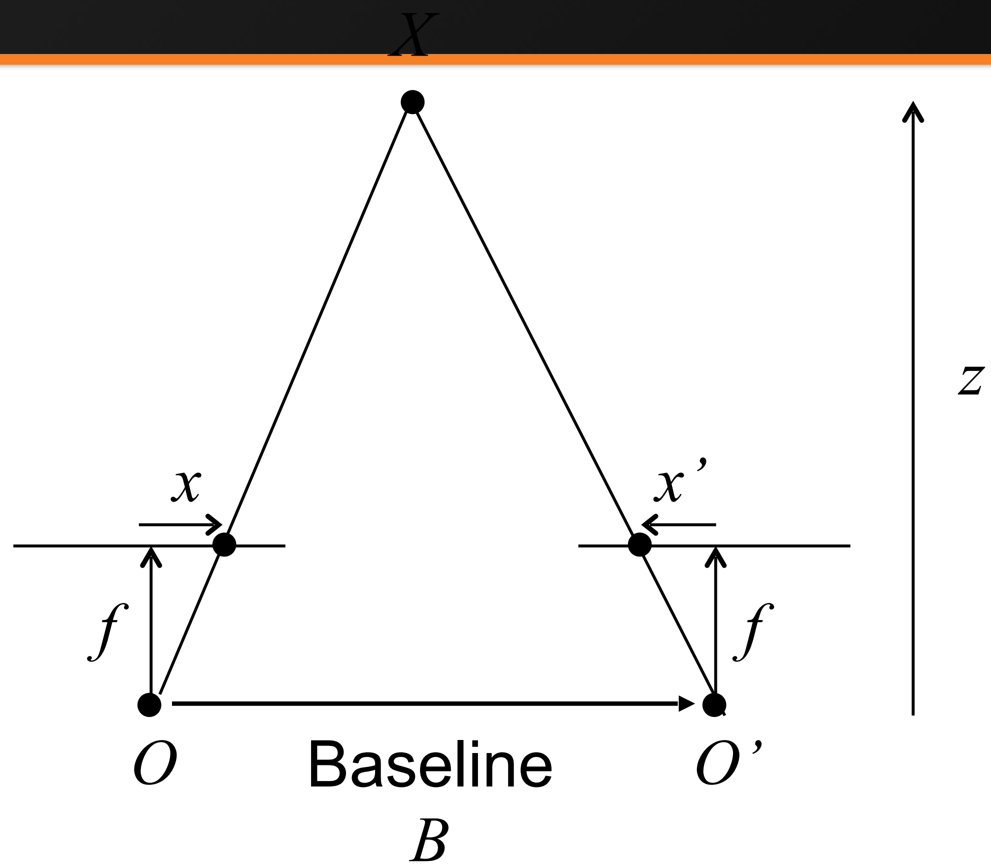
$$[u_1 \ v_1 \ 1] \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t \\ 0 & t & 0 \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

$$[u_1 \ v_1 \ 1] \begin{bmatrix} 0 \\ -t \\ tv_2 \end{bmatrix} = 0$$

$$-tv_1 + tv_2 = 0$$

The y-coordinates of corresponding points are the same!

Giving the consequence from last time that:



$$disparity = x - x' = \frac{B \cdot f}{z}$$

Disparity is inversely proportional to depth!

Fundamental Matrix

- Can define fundamental matrix F analogously to essential matrix, operating on pixel coordinates instead of camera coordinates

$$u_1^T F u_2 = 0$$

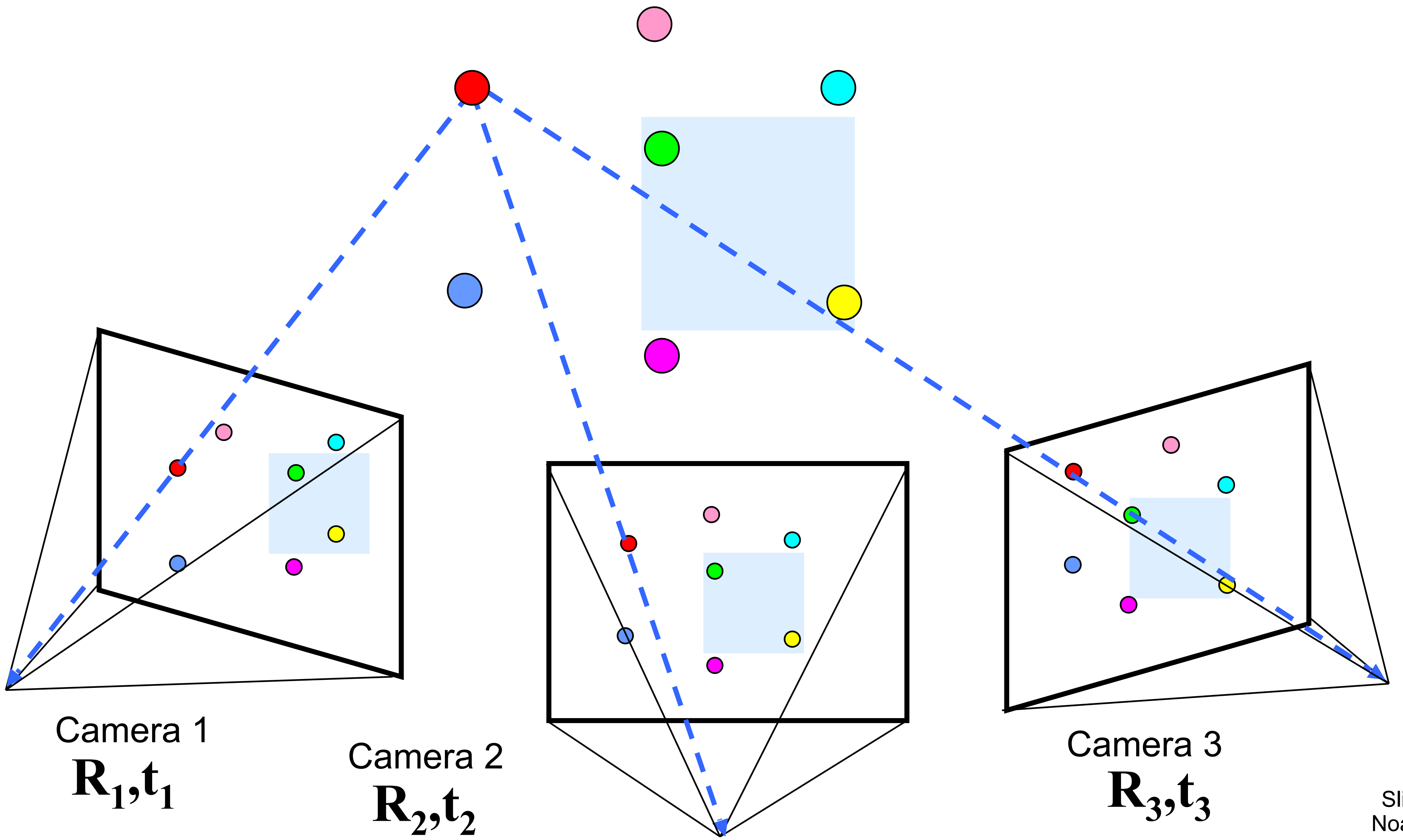
- Advantage: can sometimes estimate F without knowing camera calibration
  - Given a few good correspondences, can get epipolar lines and estimate more correspondences, all without calibrating cameras

From epipolar geometry to camera calibration

- Estimating the fundamental matrix is known as "weak calibration"
- If we know the calibration matrices of the two cameras, we can estimate the essential matrix: $\boldsymbol{E} = \boldsymbol{K}'^{T}\boldsymbol{F}\boldsymbol{K}$
- The essential matrix gives us the relative rotation and translation between the cameras, or their extrinsic parameters

Source: S. Lazebnik

Next time: multi-view geometry problems



Slide credit:
Noah Snavely