Course Notes

EECS 442

Computer Vision

Matthew Johnson-Roberson - Fall 2015

Contributors: Max Smith

Latest revision: September 23, 2015

Contents

Abstract

Computational methods for the recovery, representation and application of visual information. Topics from image formation, binary images, digital geometry, similarity and dissimilarity detection, matching, curve and surface fitting, constraint propagation relaxation labeling, stereo, shading texture, object representation and recognition, dynamic scene analysis and knowledge based techniques. Hardware, software techniques.

1 Introduction

– Computer vision studies the tools and theories that enable the design of machines that can extract useful information from imagery data (images and videos) toward the goal of interpreting the world.

  – **Information**: visual cues, 3D structure, motion flows, etc.
  – **Interpretation**: recognize objects, scenes, actions, events

1.1 Course overview

Geometry

– How to extract 3D information

– Which cues are useful

– What are the mathematical tools

– **Visual cues**: texture, shading, contours, shadows, reflections

– **Number of observers**: monocular, multiple views

– **Active lighting**: laser stripes, structured lighting patterns

Low & Mid-level vision

– Extract useful building blocks

– Region segmentation

– Motion flows

High level vision

– Recognition of objects and people

– Places

– Actions & events

2 Linear Algebra & Geometry

2.1 Basic definitions and properties

Vectors

- **Vectors (2D or 3D)**:
  - $v = (x_1, x_2)$
  - Magnitude: $||v|| = \sqrt{x_1^2 + x_2^2}$
  - **Unit vector**: iff $||v|| = \frac{v}{||v||} = 1$
  - Orientation: $\theta = \tan^{-1}\left(\frac{x_2}{x_1}\right)$
- Vector addition: $\vec{v} + \vec{w} = (x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y + 2)$
- Vector subtraction: $\vec{v} + \vec{w} = (x_1, x_2) - (y_1, y_2) = (x_1 - y_1, x_2 - y + 2)$
- Scalar product: $a\vec{v} = a(x_1, x_2) = (ax_1, ax_2)$
- Inner (dot) product: $\vec{v} + \vec{w} = (x_1, x_2) \cdot (y_1, y_2) = x_1y_1 + x_2y_2 = ||\vec{v}|| \cdot ||\vec{w}|| \cos(\alpha)$
  - Where α is the angle between the vectors
- Orthonormal basis:
  - $\vec{i} = (1, 0), \vec{j} = (0, 1)$
  - You can use them to represent vectors: $\vec{v} = (x_1, x_2) = x_1\vec{i} + x_2\vec{j}$
- Vector (cross) product:
  - $\vec{u} = \vec{v} \times \vec{w}$
  - Magnitude: $||\vec{u}|| = ||\vec{v} \times \vec{w}|| = ||\vec{v}||||\vec{w}|| \sin(\alpha)$
  - $(x_1, x_2, x_3) \times (y_1, y_2, y_3) = (x_2y_3 - x_3y_2, x_3y_1 - x_1y_3, x_1y_2 - x_2y_1)$

Matrices

- Matrices:
  - e.g.,
$$A_{n\times m} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$
  - Can be used to represent an image, where each entry is a pixel.
- Sum: $C_{n\times m} = A_{n\times m} + B_{n\times m}; c_{ij} = a_{ij} + b_{ij}$
- Product: $C_{n\times p} = A_{n\times m}B_{m\times p}$
$$\begin{bmatrix} 3 & 0 \\ 0 & 7 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 3 \\ 7 & 0 \end{bmatrix}$$

- Transpose: $C_{m\times n} = A^T_{n\times m}; c_{ij} = a_{ji}$
- Determinant:
  - Must be square
  - 2D: $det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11}a_{22} - a_{21}a_{12}$
  - 3D: $det \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} = \end{bmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{23} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$
- Inverse:
$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} = \frac{1}{a_{11}a_{22} - a_{21}a_{12}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$
  - Must be square
  - $A^{-1}A = I$
- Orthogonal: $Q_{n\times n}Q^T_{n\times n} = Q^T_{n\times n}Q_{n\times n} = I$
  - Rows and columns are unit vectors
- Block forms:
  - Represent submatrices as variables
  - e.g.,
$$P = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S & t \\ 0 & 1 \end{bmatrix}$$
$$S = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Eigenvalues and Eigenvectors

- **Eigen relation**: Matrix A acts on vector $\vec{u}$ and produces a scaled version of $\vec{u}$
$$A\vec{u} = \lambda\vec{u}$$
- $\vec{u}$: eigenvector
- λ: eigenvalue
- The eigenvalues of A are the roots of the characteristics equation:
$$p(\lambda) = det(\lambda I - A) = 0$$
- **Eigendecomposition**:
$$A = S\Lambda S^{-1}$$
  - Λ: diagonal matrix of eigenvalues
  - S: matrix with eigenvetors for columns

Singular Value decomposition

- $A = U\Sigma V^{-1}$
- U, V: orthogonal matrix
- Σ: diagonal matrix of singular values
- **Singular value**: $sigma_i = \sqrt{\lambda_i}$
  - λ: eigenvalue of $A^T A$

2.2 Geometrical transformations

- **Translation**:

$$P \to P'P = (x, y), t = (t_x, t_y)P' = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  - This will shift P by t

- **Homogeneous Coordinates**
  - Multipl;y the coordinate by a non-zero scalar and add an extra coordinate equal to that scalar. e.g.,

$$(x, y) \to (xz, yz, z), z \neq 0$$

  - To go back to cartesian: divide by last coordinate and eliminate it:

$$(x, y, z), z \neq 0 \to (x/z, y/z)$$

- Translation using homogeneous coordinates:

$$P' = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} i & T \\ 0 & 1 \end{bmatrix} P = TP$$

- **Scaling**:

$$P' = \begin{bmatrix} s_x t_x \\ x_y t_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} S & 0 \\ 0 & 1 \end{bmatrix} P = SP$$

- Scaling & Translating:

$$P' = TSP = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Translation and scaling is not communative: $STP \neq TSP$
- **Rotation**:
  - Rotation by angle θ

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

$$P' = RP$$

- Translation + Rotation + Scaling =

$$P' = (TRS)PP' = \begin{bmatrix} RS & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2.3 Transformation in 2D

Isometries

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H_e \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Preserve distance (areas)
- 3 DOF
- Regulate motion of a rigid object

Similarities

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} SR & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H_s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Preserve:
  - ratio of lengths
  - angles
- 4 DOF

Affinities

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H_a \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = R(\theta)R(-\phi)DR(\phi), D = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

- Preserve:
  - Parallel lines
  - Ratio of areas
  - Ratio of lengths on collinear lines
  - others...
- 6 DOF

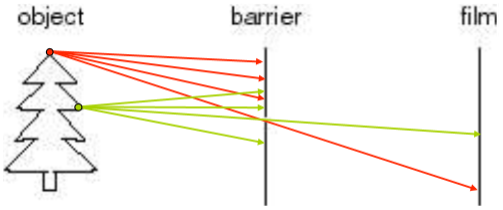
Projective

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} A & t \\ v & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H_p \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

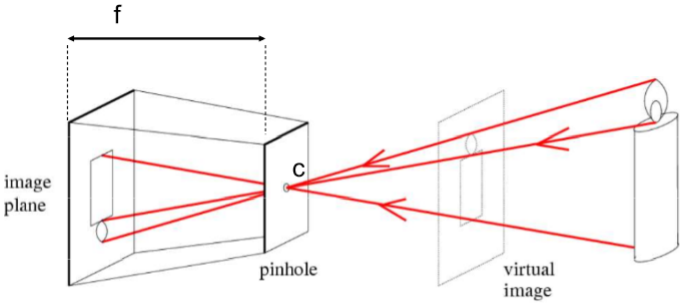
- 8 DOF
- Preserve:
  - cross ratio of 4 collinear points
  - collinearity
  - and a few others...

3 Cameras

3.1 Pinhole Camera

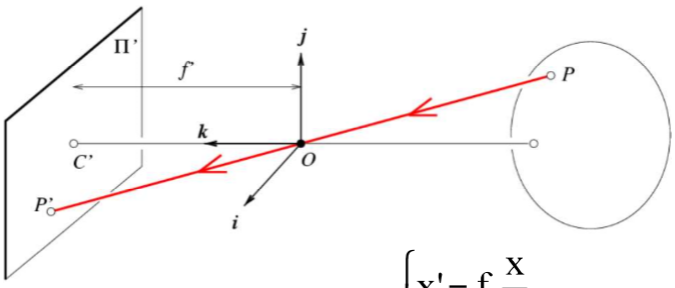


- A barrier is added to block off most of the rays
- This process reduces blurring resulting from multiple rays of light hitting the same spot
- **Aperature**: hole in the barrier that allows light to pass through.



- f (**focal length**): distance between the lens (pinhole) the image sensor (image plane)
- c: center of the camera/pinhole/aperature
- Say we have an object of interest P and want to understand how it'll register on our image plane as P'

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$



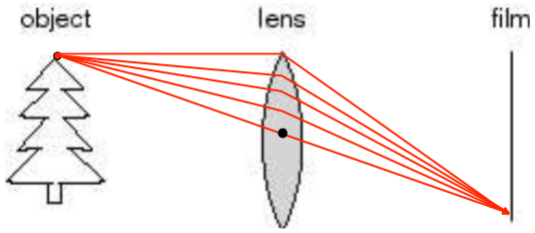
$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \qquad \begin{cases} x' = f\,\dfrac{x}{z} \\ y' = f\,\dfrac{y}{z} \end{cases}$$

- We can derive the tranformation using similar triangles:

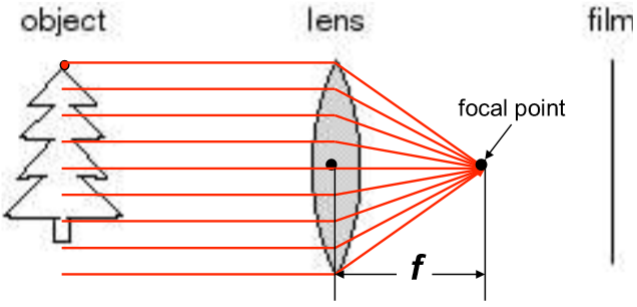
$$x' = f\frac{x}{z}, y' = f\frac{y}{z}$$

3.2 Cameras & Lenses

- If the aperature is too large, your image will blur
- If it's too small you won't have enough light going through to have an image (too dark)
- This can be fixed by using lenses

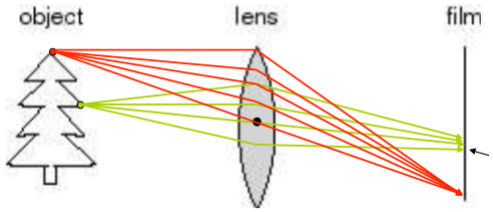


- A **lens** focuses light onto the film



- Rays passing through the center of the lens are not deviated

– All parallel rays converge to one point on a plan located at the **focal length** f



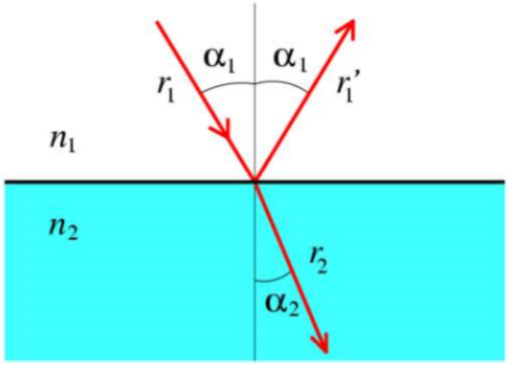
– There is a specific distance at which objects are "in focus"

– Related to the concept of depth of field ("range object is in focus")

– The area pointed at in the image refers to the "circle of confusion" an area where you can't resolve individual rays

3.3 Snell's Law

– Light bends at an interface as a result of travelling different speeds in different mediums.



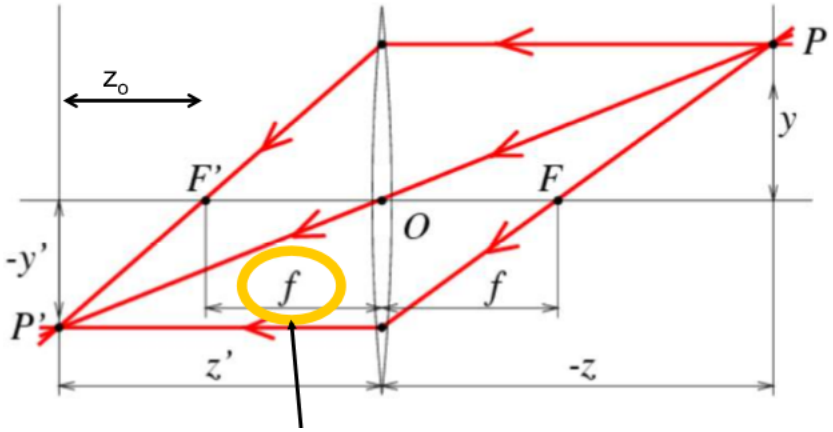
$$n_1 \sin \alpha_1 = n_2 \sin \alpha_2$$

– α_1: incident angle

– α_2: refraction angle

– n_i: index of refraction

– For small angles:



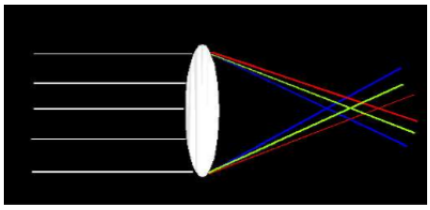
- $n_1\alpha_1 \approx n_2\alpha_2$
- $z' = f + z_0$
- $f = \frac{R}{2(n-1)}$

$$x' = z'\frac{x}{z}, y' = z'\frac{y}{z}$$

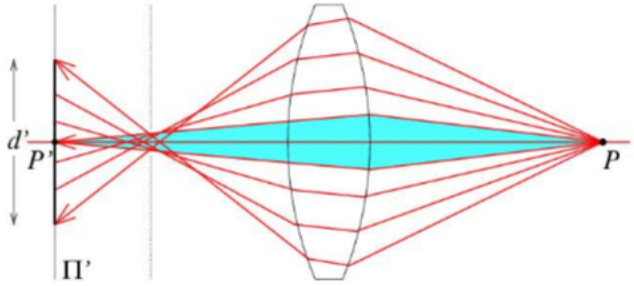
3.4 Issues with lenses

- **Chromatic aberration**: lenses have different refractive indices for different wavelengths which cause color fringing

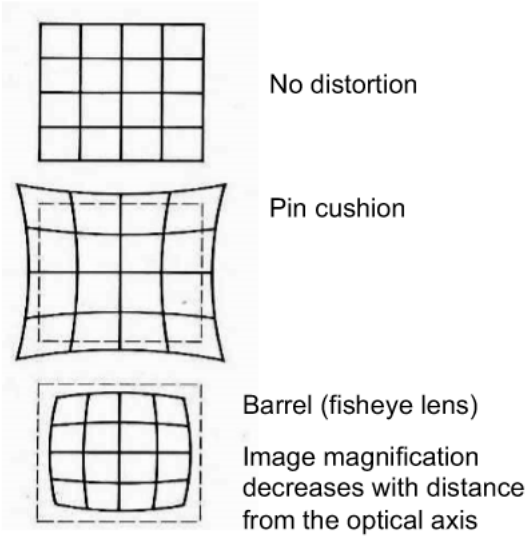
$$f = \frac{R}{2(n-1)}$$



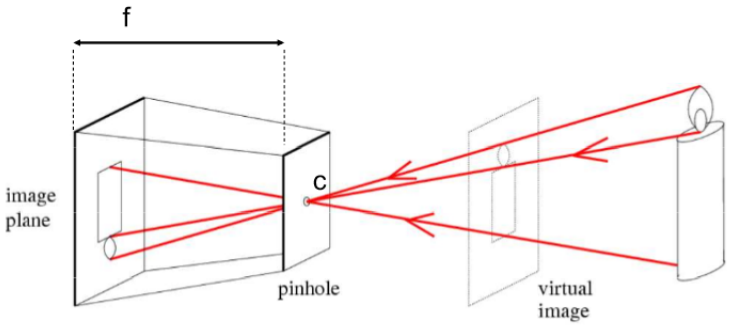
- **Spherical aberration**: rays farther from the optical axis focus closer

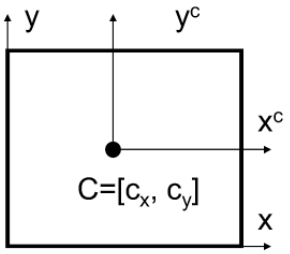


– **Radial distortion**: deviations are most noticeable for rays that pass through the edge of the lens



3.5 Converting to pixels





1. Offset
   - The center of the image plane is at coordinates $C = (c_x, c_y)$
   - This modifies our projection mapping to: $(f\frac{x}{z} + c_x, f\frac{y}{z} + c_y)$
2. From metric to pixels
   - If we let k, l be conversion metrics with units pixel/m, too allow for pixel length and width to be different
   - We can now make these modifications to the conversion to be in pixels:

$$(x, y, z) \rightarrow (fk\frac{x}{z} + c_x, fl\frac{y}{z} + c_y)(x, y, z) \rightarrow (\alpha\frac{x}{z} + c_x, \beta\frac{y}{z} + c_y)$$

   - We use α, β for ease of notation

3.6 Linear Transformation to pixels

- Our previous formula isn't linear because of the division by z
- If we use homogeneous coordinates we can make it linear:
  - Image coordinates: $(x, y, 1)$
  - Scene coordinates: $(x, y, z, 1)$
- We can now define a matrix M to represent our transformation:

$$x' = Mxx' = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- We can now use this idea for our pixel mapping:

$$x' = \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3.7 Camera Matrix

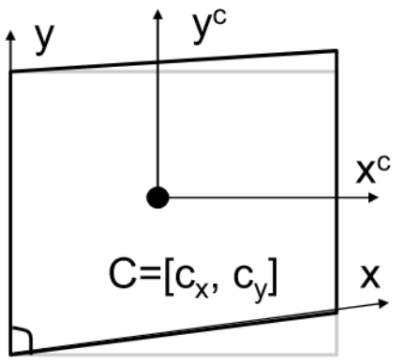
– The subsection of our previous matrix:

$$\begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} = K$$

– K is called the camera matrix

– It sometimes features a skew parameter (s), but in pratcie $s \approx 0$

$$\begin{bmatrix} \alpha & s & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix}$$



3.8 3D Rotation of Points

– Rotation around the coordinate axes, **counter-clockwise**:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3.9 Camera Matrix Homogeneous Coordinates

– In 4D homogeneous coordinates the mapping from world to image follows:

$$X = \begin{bmatrix} R & T \\ 0 & T \end{bmatrix}_{4\times4} X_{world}$$

$$X' = K_{3\times3}[R\ T]_{3\times4} X_{world,4\times1}$$

– Here K represents the internal camera parameters: $K = \begin{bmatrix} \alpha & s & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix}$

– Our external camera parameters are represented by R, T

– If we represent $M = \begin{bmatrix}\vec{m}_1 & \vec{m}_2 & \vec{m}_3\end{bmatrix}^T$, then we may also write our transformation as:

$$(x, y, z)_w \rightarrow (\frac{\vec{m}_1 X_m}{\vec{m}_3 X_w}, \frac{\vec{m}_2 X_w}{\vec{m}_3 X_w})$$

Theorem 3.1 (Theorem (Faaugeras, 1993))**.**

$$M = K[R\ T] = [KR\ KT] = [A\ b]$$

– Perspective projection req:

– $det(A) \neq 0$

– Zero-skew conditions:

– $det(A) \neq 0$
– $(\vec{a}_1 \times \vec{a}_3) \cdot (\vec{a}_2 \times \vec{a}_3) = 0$

– Perspective proj & zero-skew conditons:

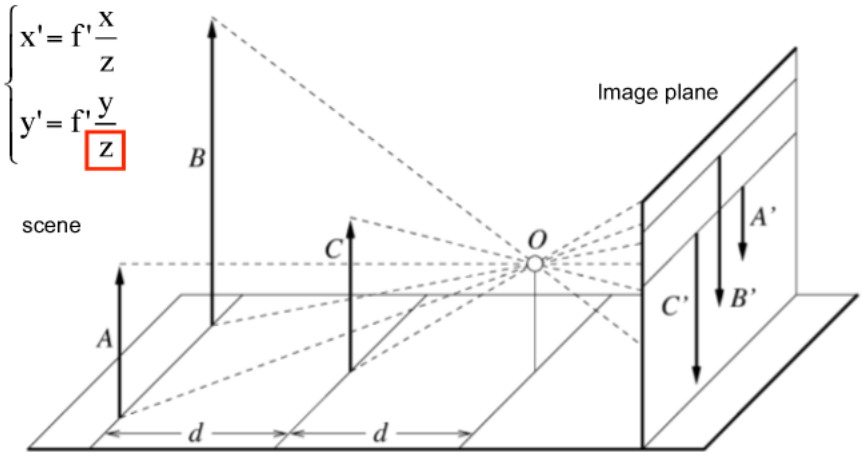
– $det(A) \neq 0$
– $(\vec{a}_1 \times \vec{a}_3) \cdot (\vec{a}_2 \times \vec{a}_3) = 0$
– $(\vec{a}_1 \times \vec{a}_3) \cdot (\vec{a}_1 \times \vec{a}_3) = (\vec{a}_2 \times \vec{a}_3) \cdot (\vec{a}_2 \times \vec{a}_3)$

3.10 Properties of Projection

– Points project to points

– Liens project to lines

– Distant objects look smaller



– Angles are not preserved

– Parallel lines will meet at a **vanishing point**

– Objects on the periphery are expanded

– Degenerate cases:

  – Line through focal point projects to a point
  – Plane though focal point projects to line
  – Plane perpendicular to image plane projects to part of the image (with horizon)

Vanishing points

– Sets of parallel lines on the same plan lead to collinear vanishing points (The line is called the **horizon** for that plane)

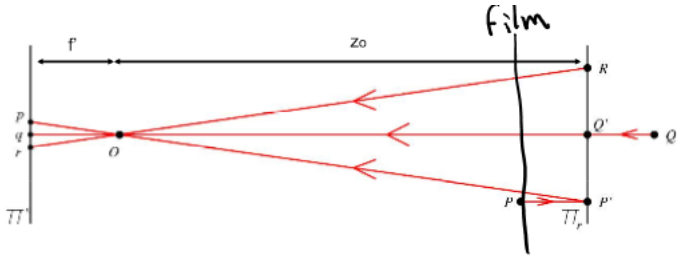
– Each set of parallel lines meets at a different point (The vanishing point for this direction)

3.11 Other camera models

Weak perspective projection

– When the relative scene depth is small compared to its distance from the camera

– Namely, if the relative distance (scene depth) between two points of a 3D object along the optical axis is much smaller than the average distance to the camera



$$x' = -\frac{f'}{z_0}x, y' = -\frac{f'}{z_0}y$$

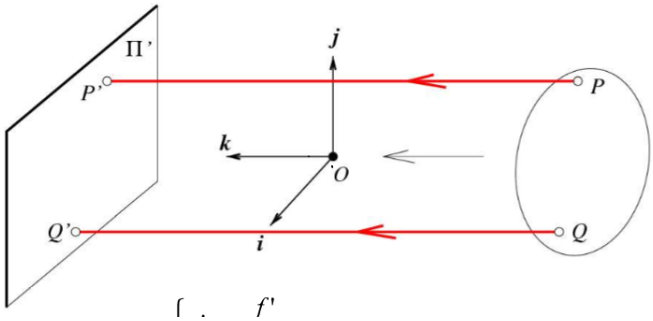
– In this perspective the fraction is the **magnification** (m)

– Accurate when object is small and distant

– Most useful for recognition

Orthographic (affine) projection

– Distance form center of projection to image plane is infinite



$$\lceil \quad . \quad f'$$

$$x' = -x, y' = -y$$

– All rays are parallel

4 Camera Calibration

4.1 Goal of Calibration

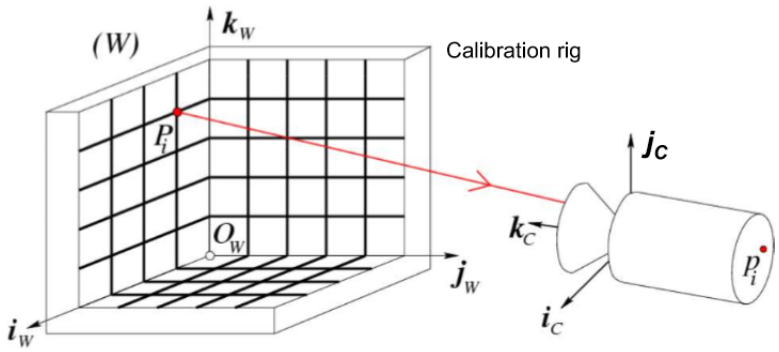
– Notation change: $P = P_w, p = P'$

– Goal: estimate intrinsic and extrinsic parameters from 1 or multiple images

$$P' = MP_w = K[R\ T]P_w$$

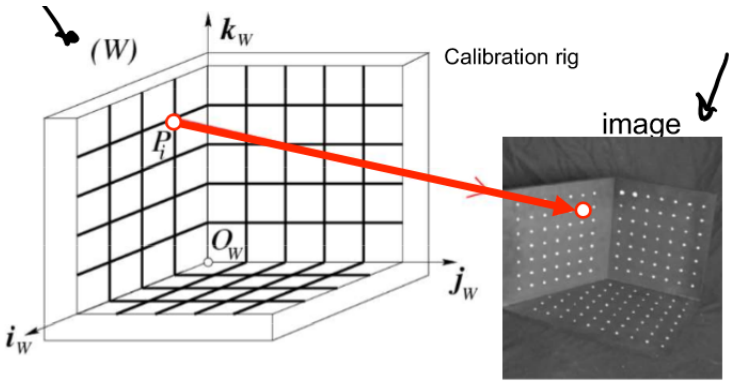
$$M = \begin{bmatrix} \alpha\vec{r_1}^T - \alpha\cot\theta\vec{r_2}^T + u_0\vec{r_3}^T & \alpha t_x - \alpha\cot\theta t_y + u_0 t_z \\ \frac{\beta}{\sin\theta}\vec{r_2}^T + v_0\vec{r_3}^T & \frac{\beta}{\sin\theta}t_y + v_0 t_z \\ \vec{r_3}^T & t_z \end{bmatrix}_{3\times 4}$$

4.2 Calibration Problem



– $P1, \ldots P_n$ with known position from rig in (O_w, i_w, j_w, k_w)

– $p_1, \ldots p_n$ with known positions in the image from camera



- This figure shows the mapping from a point P on the rig to p on the image.
- We need to solve for 11 unkowns in M with 11 equations, resulting in the need for 6 correspondences.

$$P_i \rightarrow MP_i \rightarrow p_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \frac{\vec{m}_1 P_i}{\vec{m}_3 P_i} \\ \frac{\vec{m}_2 P_i}{\vec{m}_3 P_i} \end{bmatrix}, \text{ where } M = \begin{bmatrix} \vec{m_1} \\ \vec{m_2} \\ \vec{m_3} \end{bmatrix}$$

- Simplifying the final equation for p_i, we receive:

$$u_i(\vec{m}_3 P_i) - \vec{m}_1 P_i = 0$$

$$v_i(\vec{m}_3 P_i) - \vec{m}_1 P_i = 0$$

- In the case of the calibration problem we may have n mappings so $i \in [1, n]$

4.3 Calibration matrices

- The resulting list of point mappings:

$$u_1(\vec{m}_3 P_1) - \vec{m}_1 P_1 = 0$$
$$v_1(\vec{m}_3 P_1) - \vec{m}_1 P_1 = 0$$
$$\vdots$$
$$u_n(\vec{m}_3 P_n) - \vec{m}_1 P_n = 0$$
$$v_n(\vec{m}_3 P_n) - \vec{m}_1 P_n = 0$$

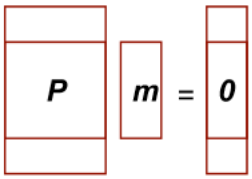
- Can be rewritten after considering an inverse block matrix multiplication:

$$\mathcal{P}\vec{m} = \vec{0}$$

$$\mathcal{P} := \begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \vdots & \vdots & \vdots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{bmatrix}_{2n \times 12}, \vec{m} = \begin{pmatrix} \vec{m}_1^T \\ \vec{m}_2^T \\ \vec{m}_3^T \end{pmatrix}_{12 \times 1}$$

4.4 Homogeneous $M \times N$ Linear Systems

- $M = number of equations = 2n$
- $N = number of unknown = 11$



- Rectangular system ($M > N$)
  - $\vec{0}$ is always a solution
  - To find non-zero solution:
    - Minimize $|\mathcal{P}\vec{m}|^2$
    - Under the constraint $|\vec{m}|^2 = 1$

4.5 Calibration Problem

$$\mathcal{P}\vec{m} = \vec{0}$$

- How do we solve this homogeneous linear system?
- Use **Direct Linear Transformation (DLT)** algorithm via SVD decomposition.
- Perform SVD decomposition of $\mathcal{P}$

$$\mathcal{P} \rightarrow U_{2n\times 12} D_{12\times 12} V^T_{12\times 12}$$

- The last column of V gives $\vec{m}$

$$MP_i \rightarrow p_i$$

- This representation of SVD is one of the possible decompositions that is typically used for efficiency
- If we rewrite our matrix $M = [A\ \vec{v}]$

$$\rho \begin{pmatrix} \vec{a}_1^T \\ \vec{a}_2^T \\ \vec{a}_3^T \end{pmatrix} = \begin{pmatrix} \alpha \vec{r}_1^T - \alpha \cot\theta \vec{r}_2^T + u_0 \vec{r}_3^T \\ \frac{\beta}{\sin\theta} \vec{r}_2^T + v_o \vec{r}_3^T \\ \vec{r}_3^T \end{pmatrix}$$

- Then using the fact that rows of a rotation matrix have unit length and are perpendicular this yields

(intrinsic):

$$\begin{aligned}
\rho &= \epsilon/|\vec{a}_3| \\
\vec{r}_3 &= \rho\vec{a}_3 \\
u_0 &= \rho^2(\vec{a}_2 \cdot \vec{a}_3) \\
v_0 &= \rho^2(\vec{a}_2 \cdot \vec{a}_3) \\
\cos\theta &= \frac{(\vec{a}_1 \times \vec{a}_3) \cdot (\vec{a}_2 \times \vec{a}_3)}{|\vec{a}_1 \times \vec{a}_3| \cdot |\vec{a}_2 \times \vec{a}_3|}
\end{aligned}$$

Can be used to solve for f:

$$\begin{aligned}
\alpha &= \rho^2|\vec{a}_\times\vec{a}_3|\sin\theta \\
\beta &= \rho^2|\vec{a}_2 \times \vec{a}_3|\sin\theta epsilon \quad = \mp 1
\end{aligned}$$

– You may also derive the following formulas for extrinsic parameters:

$$\begin{aligned}
\vec{r}_1 &= \frac{(\vec{a}_2 \times \vec{a}_3)}{|\vec{a}_2 \times \vec{a}_3|} \\
\vec{r}_2 &= \vec{r}_3 \times \vec{r}_1 \\
\vec{r}_3 &= \frac{\pm 1}{|\vec{a}_3|}T \qquad = \rho K^{-1}\vec{b}
\end{aligned}$$

4.6 Degenerate cases

– P_i's cannot lie on the same plane

– Points cannot lie on the intersection curve of two quadric surfaces

4.7 Radial Distortion

Radial Distortion

– TODO: Image

– Image magnification (in/de)creases with distance from the optical center

$$\begin{bmatrix} \frac{1}{\lambda} & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & 1 \end{bmatrix} MP_i \to \begin{pmatrix} u_i \\ v_i \end{pmatrix} = p_i$$

$$\lambda = 1 \pm \sum_{p=1}^{3} \kappa_p d^{2p}, \text{ where } \kappa \text{ is the Distortion coefficient}$$

– To model the radial behavior:

– TODO: radial disto image

$$d^2 = au^2 + bv^2 + cuv$$

– If we combine our two matrices to have a more concise notation:

$$\begin{bmatrix} \frac{1}{\lambda} & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & 1 \end{bmatrix} M = Q = \begin{bmatrix} \vec{q}_1 \\ \vec{q}_2 \\ \vec{q}_3 \end{bmatrix}$$

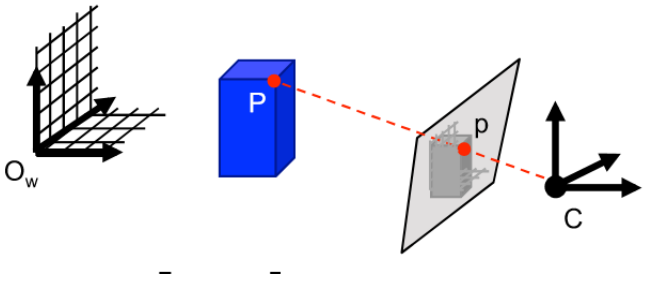
- This allows us to write our image coordinates into equations:

$$p_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \frac{\vec{q}_1 P_i}{\vec{q}_3 P_i} \\ \frac{\vec{q}_2 P_i}{\vec{q}_3 P_i} \end{bmatrix} \rightarrow u_i \vec{q}_3 P_i = \vec{q}_1 P_i, v_i \vec{q}_3 P_i = \vec{q}_2 P_i \rightarrow \frac{1}{\lambda} \begin{bmatrix} \frac{\vec{m}_1 P_i}{\vec{m}_3 P_i} \\ \frac{\vec{m}_2 P_i}{\vec{m}_3 P_i} \end{bmatrix}$$

- Simplifying it into this form allows us to ask the question: can we estimate $\vec{m}_1$ and $\vec{m}_2$ and ignore the radial distortion?
- Remember our distortion took the form of: TODO INSERT PICTURE
- The slope of our corner should be: slope $= \frac{u_i}{v_i} = \frac{\frac{\vec{m}_1 P_i}{\vec{m}_3 P_i}}{\frac{\vec{m}_2 P_i}{\vec{m}_3 P_i}}$

5 Single view metrology

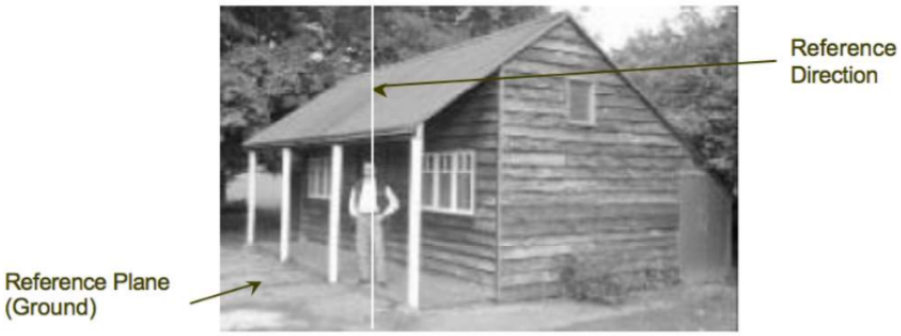
5.1 After calibration



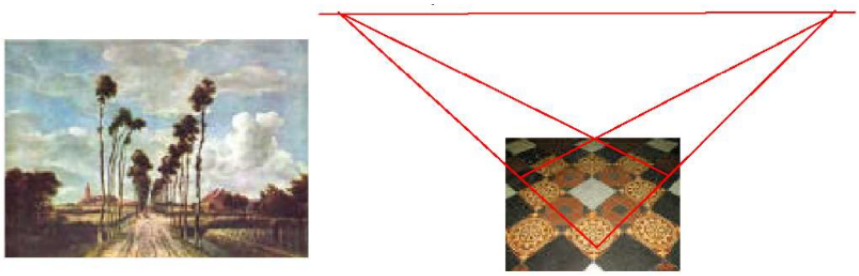
- Internal parameters K are known
- R, T are known - but these can only relate C to the calibraiton rig.
- You can't estimate P_i from the single image measurement of p_i because it may be anywhere along the line in space.
- We also don't have any information in the image to tell us the scale of anything in the world.

5.2 Recovering structure from a single view

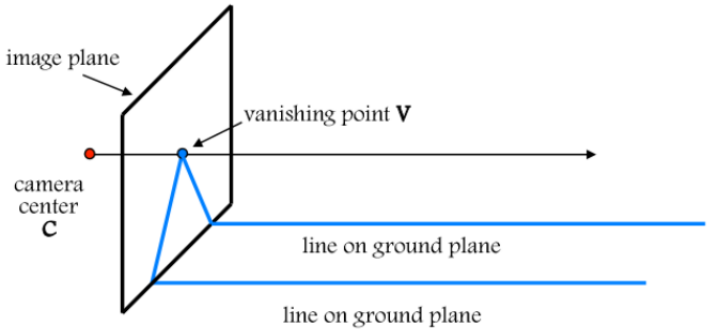
- You can make assumptions to get relative sizes of objects in images
- Pick a reference plane in the scene
- Pick a reference direction (not parallel to the reference plane) in the scene



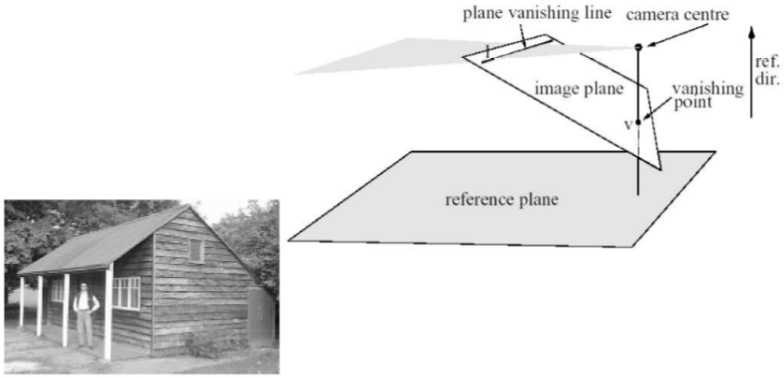
Geometry



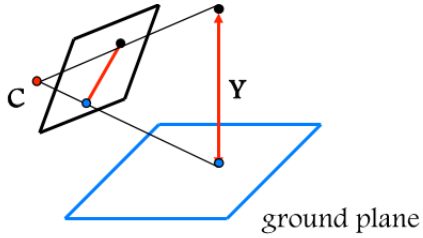
- Under perspective projection, parallel lines in three-space project to convering lines in the image plane. The common point of intersection, perhaps at infinity, is called the **vanishing point**
  - projection of a point at infinity (goes infinity far)



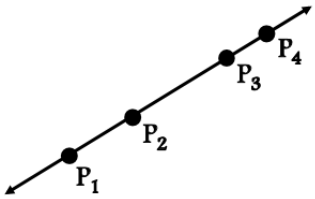
  - Any two parallel lines have the same vanishing point
  - The ray from C through v point is parallel to the lines
  - AN image may have more than one vanishing point
- Two or more vanishing points from lines known to lie in a single 3D plane establish a **vanishing line**, which completely determines the orientation of the plane



5.3 The Cross Ratio



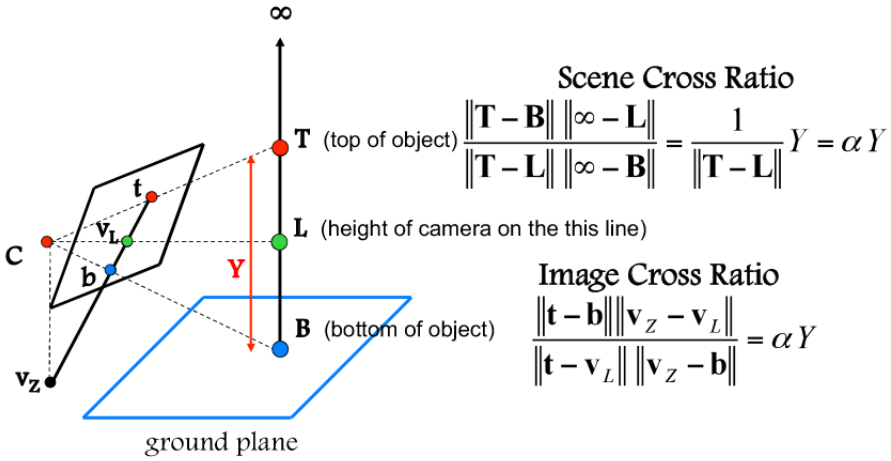
- Y is desired height to measure
- Compute Y from image measurements
  - You'll need more than vanishing points to compute this
- **Projective Invariant**: something that does not change under projective transformations (including perspective projection)



- The **cross ratio** of 4 collinear points is projective invariant

$$\frac{||P_3 - P_1|| \cdot ||P_4 - P_2||}{||P_3 - P_2|| \cdot ||P_4 - P_1||}$$

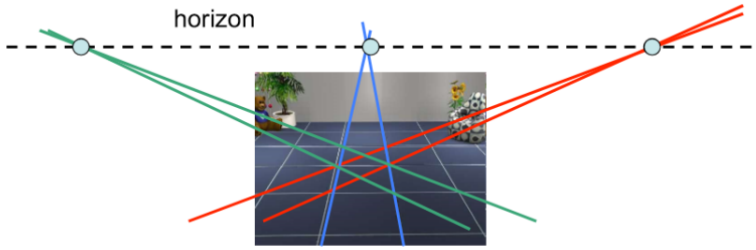
- You can permute the point ordering and it will remain true.
- Often called the fundamental invariant of projective geometry



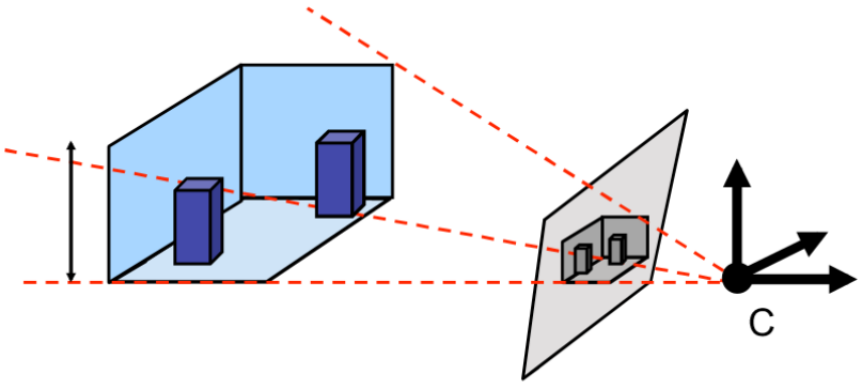
- When you consider the points at ∞, and the point where the camera would meet the ground plane v_z you have enough points to use the cross ratio on our camera model.

- You need the same points in the world and camera system (can't mix and match)
- You must make some assumption to determine the relative sizes, typically assume L

5.4 Horizon line



- Sets of parallel lines on the same plane lead to collinear vanishing points the line is called the **horizon**

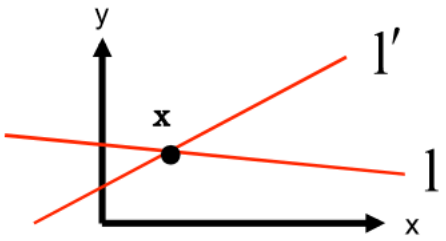


- When trying to recover the structure within the camera reference system, we can check if two linesare parallel or not
  - If they do, recognize the horizon line
  - Measure if hte 2 lines meet the horizon
  - If they do, they are parallel in 3D
  - Actual scale of scene is not recovered, only relative distances

5.5 Lines in a 2D plane

$$ax + by + c = 0; l = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

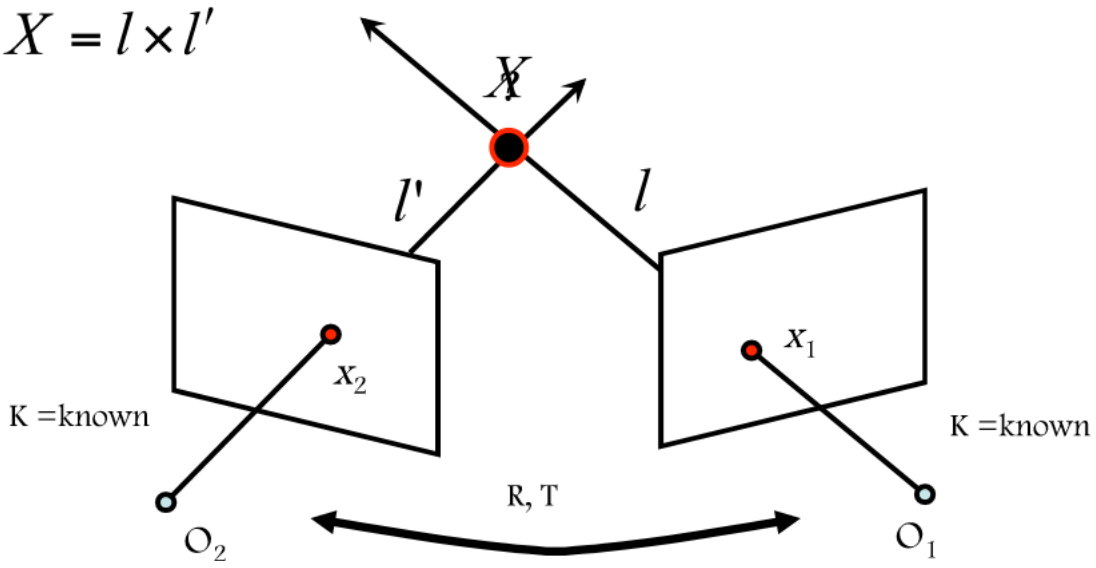
Intersecting lines



- The intersection of lines can be calculated with:

$$x = l \times l'$$

5.6 Stereo-view geometry

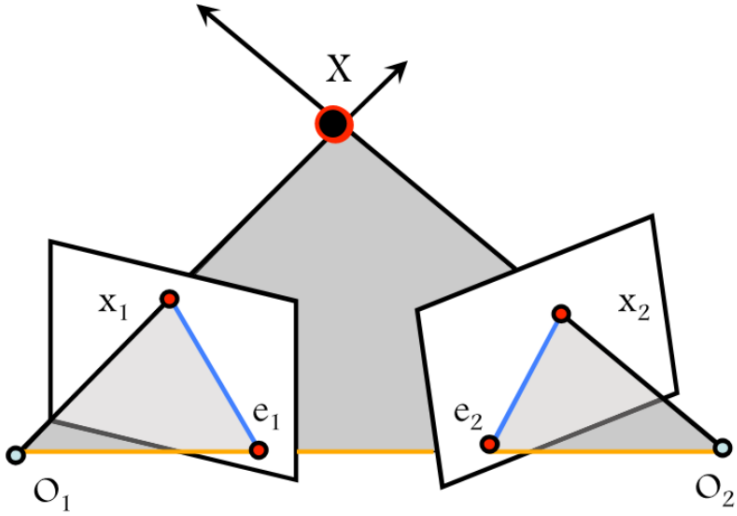


- Two camera perspectives allow us to find position of objects through **triangulation**
  - This requires a few knowns, K_1, K_2, R, T
- Small inaccuracies with knowns can lead to situations where the lines may not intersect
- Instead, we'll find where they came close enough

$$d^2(x_1, M_1X) + d^2(x_2, M_2X)$$

$d :=$ distance between two lines

5.7 Epipolar Geometry

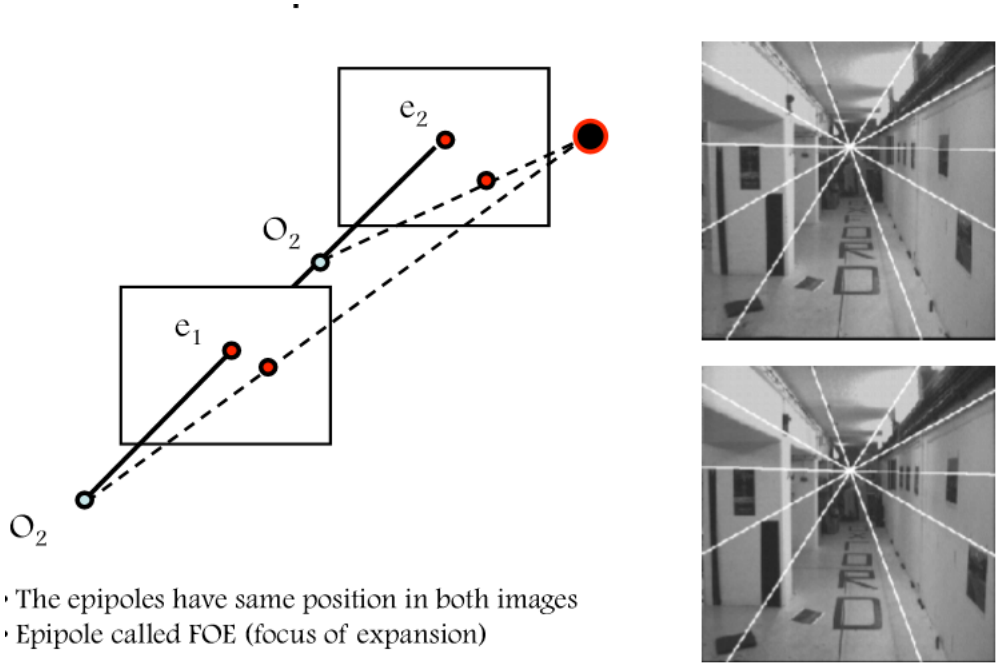


- **Epipolar plane** (grey): intersections of baseline with image planes
- **Baseline** (orange): projectison of other camera center
- **Epipolar lines** (blue): vanishing points of camera motion direction
- This framepoint, allows us to consider if two pointsare related by epipolar geometry
- Use epipolar lines to find sharing points will greatly save computation cost as a 2D search becomes 1D

Parallel image planes

- Baseline intersects the image plane at infinity
- Epipoles are at infinity
- Epipolar lines are parallel to x-axis

Forward translation



– When a camera moves forward the lines turn into a spiral