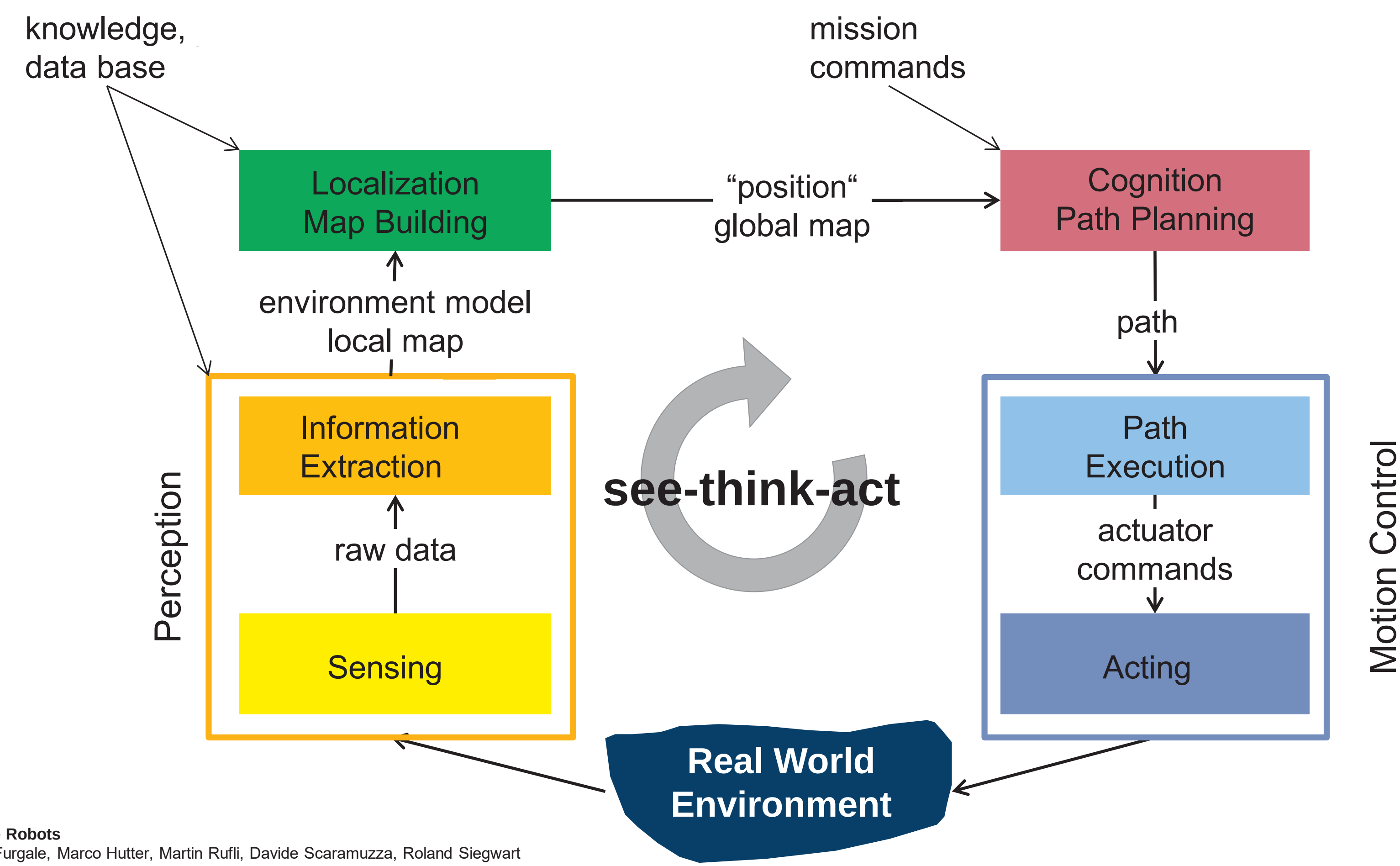
Perception II: Pinhole camera and Stereo Vision

Autonomous Mobile Robots

Davide Scaramuzza

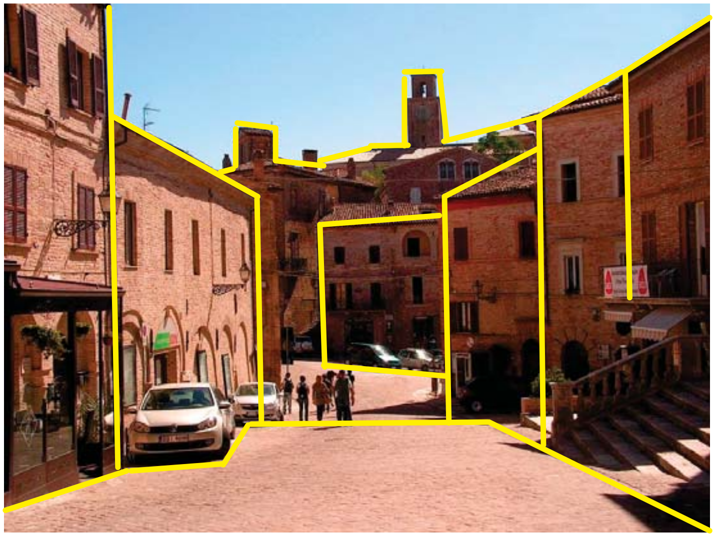
Margarita Chli, Paul Furgale, Marco Hutter, Roland Siegwart

Mobile Robot Control Scheme



Computer vision | definition

- Automatic extraction of “meaningful” information from images and videos



Semantic information

Geometric information

Computer vision | applications

- 3D reconstruction and modeling
- Recognition
- Motion capture
- Augmented reality:
- Video games and tele-operation
- Robot navigation and automotive
- Medical imaging

Google Earth, Microsoft's Bing Maps

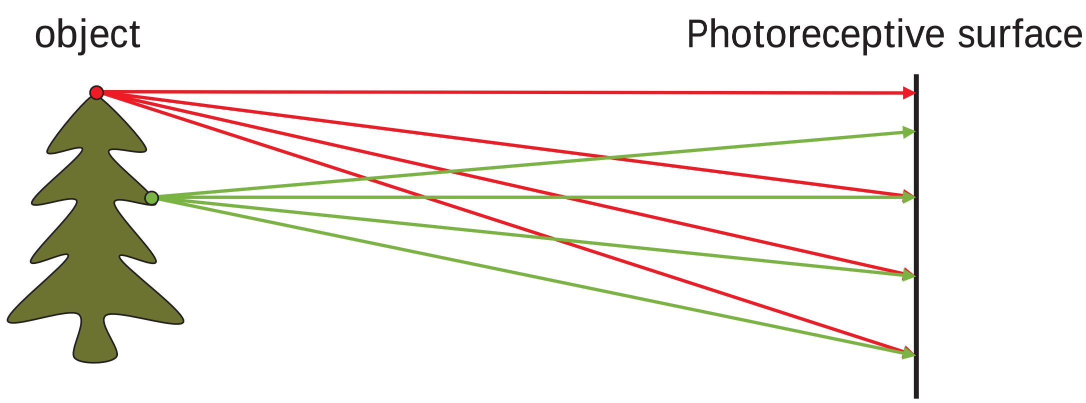
Mars rover Spirit used cameras for visual odometry

The camera

Sony Cybershot WX1

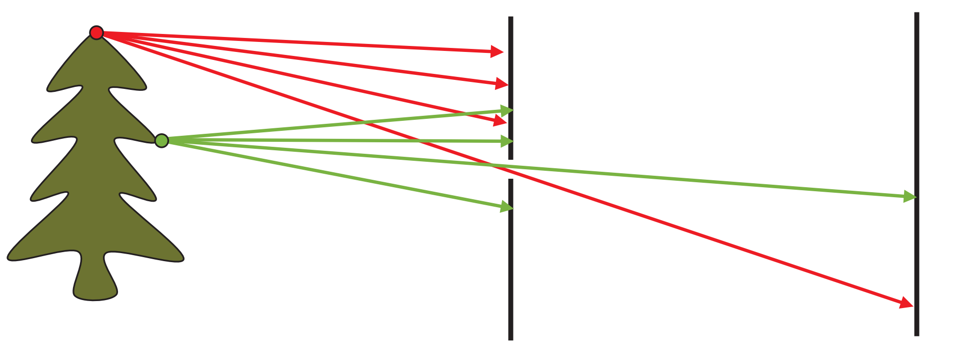
The camera | image formation

- If we place a piece of film in front of an object, do we get a reasonable image?



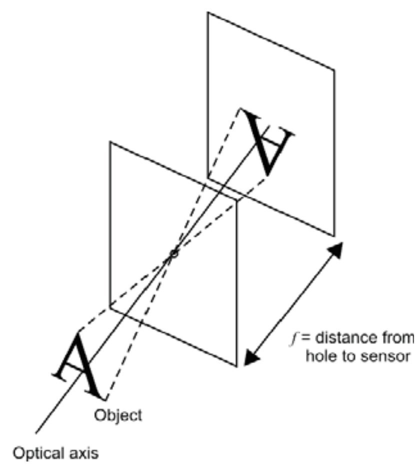
The camera | image formation

- If we place a piece of film in front of an object, do we get a reasonable image?
- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening is known as the **aperture**



The camera | camera obscura (pinhole camera)

- Pinhole model:
  - Captures **beam of rays** – all rays through a single point
  - The point is called **Center of Projection** or **Optical Center**
  - An “inverted” image is formed on the **Image Plane**
- We will use the pinhole camera model to describe how the image is formed



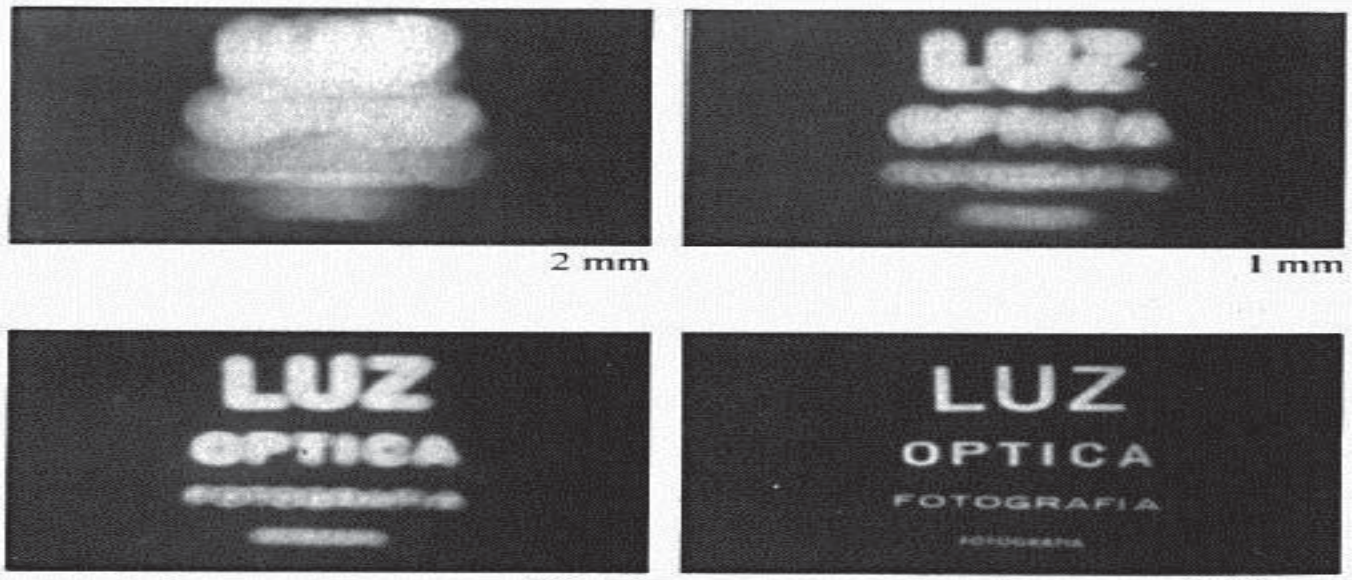
Gemma-Frisius (1508–1555)

Home-made pinhole camera

What can we do
to reduce the blur?

Based on slide by Steve Seitz

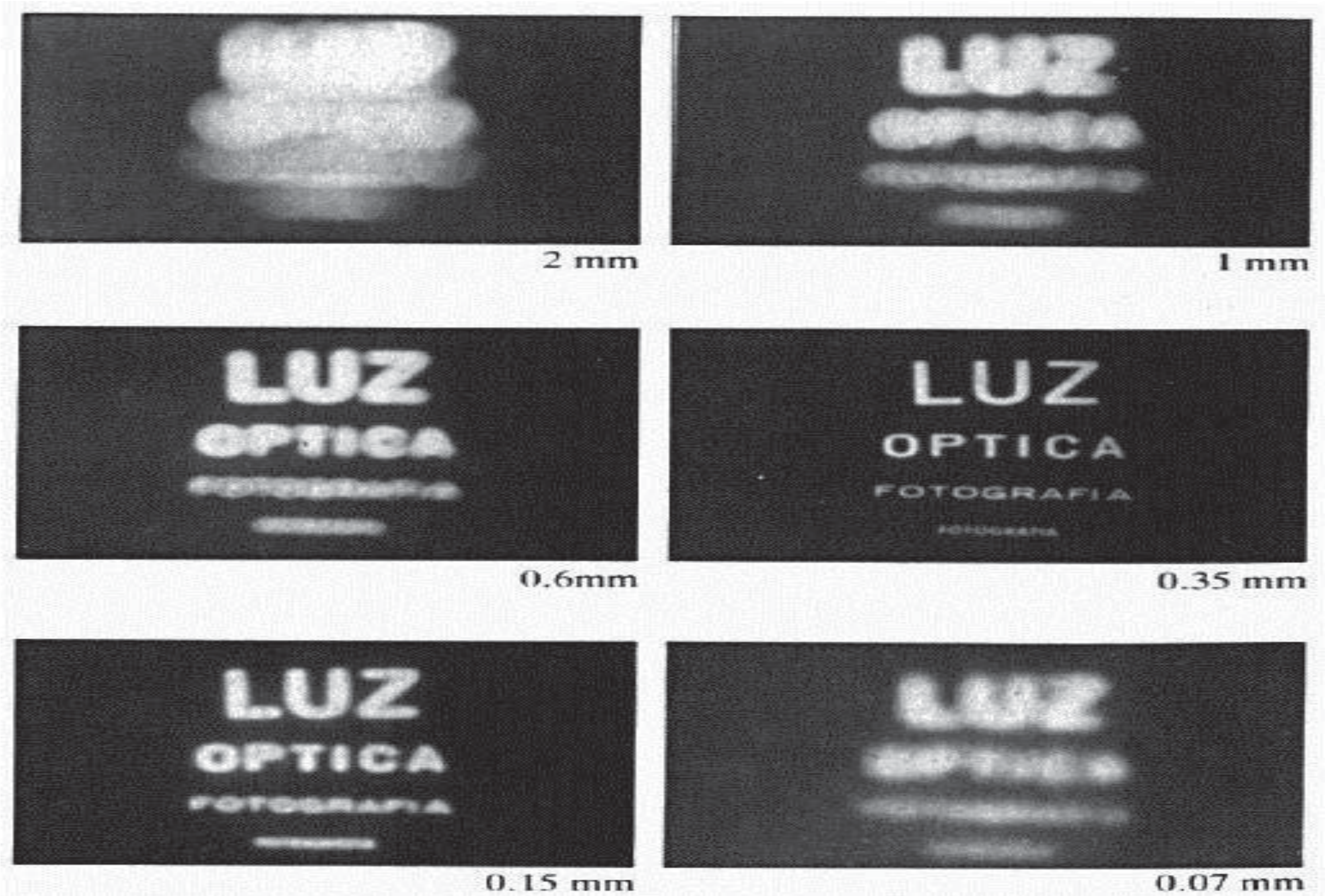
Shrinking the aperture



2 mm | 1 mm | 0.6mm | 0.35 mm

Why not make the aperture as small as possible?

Shrinking the aperture

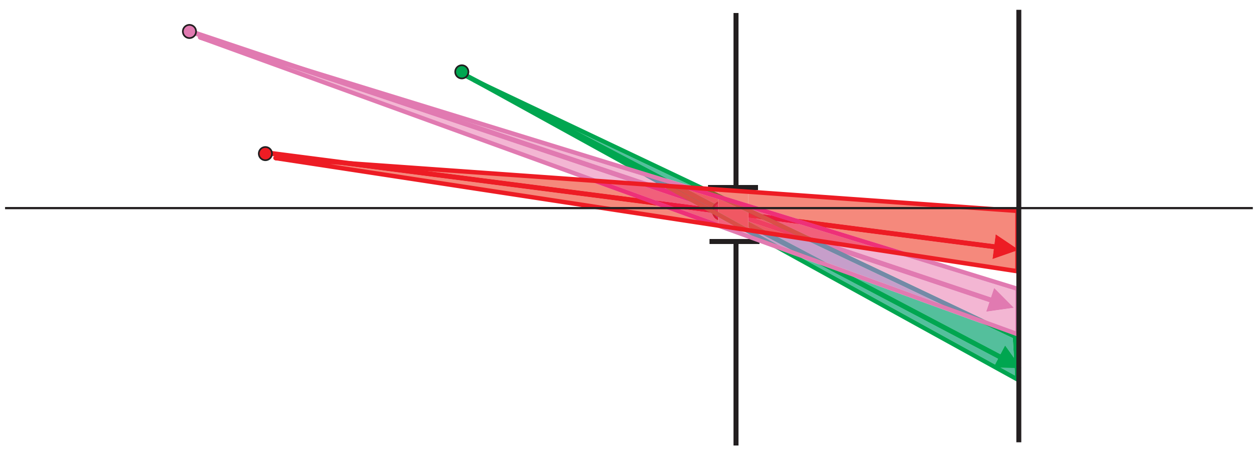


Why not make the aperture as small as possible?

- Less light gets through (must increase the exposure)
- Diffraction effects…

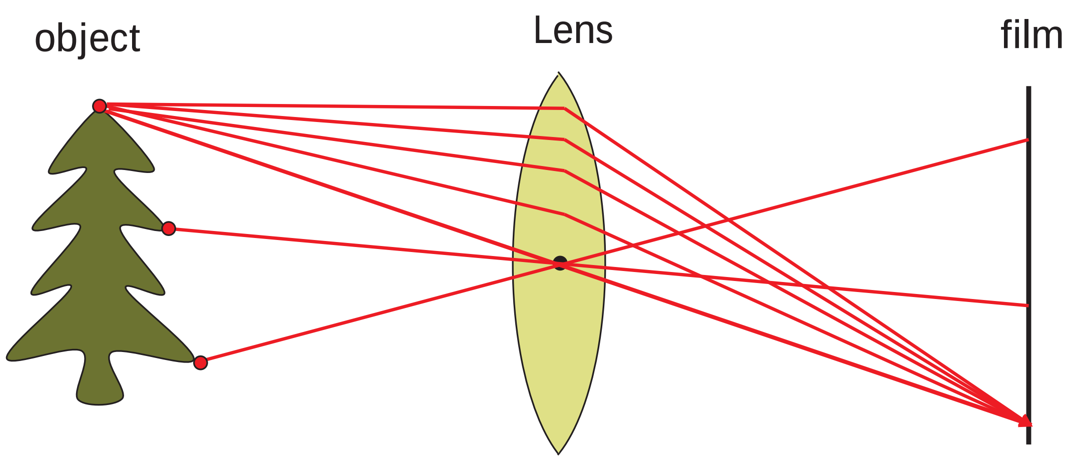
The camera | why use a lens?

- The ideal pinhole: only one ray of light reaches each point on the film
  - ⇨ image can be very dim; gives rise to diffraction effects
- Making the pinhole bigger (i.e. aperture) makes the image blurry



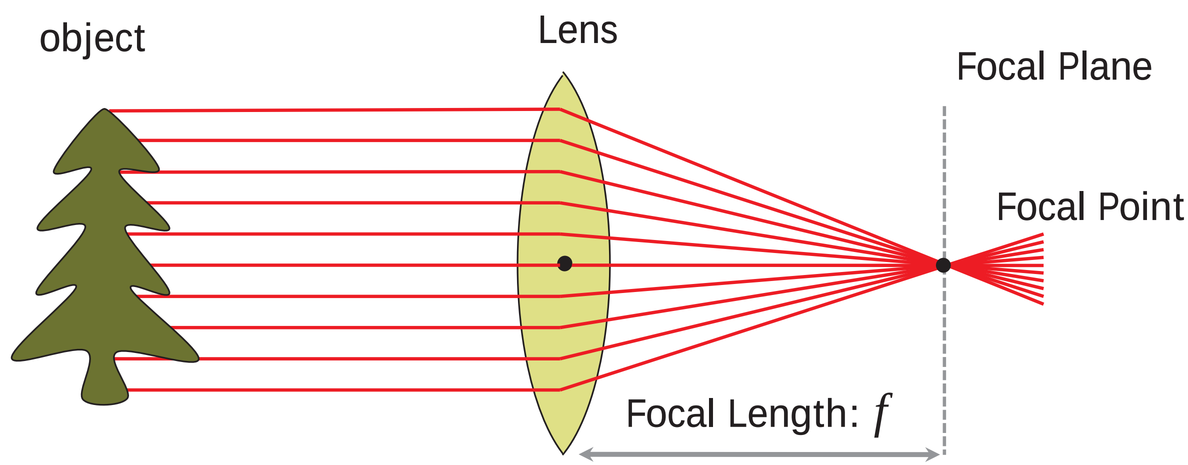
The camera | why use a lens?

- A lens focuses light onto the film
- Rays passing through the **optical center** are not deviated



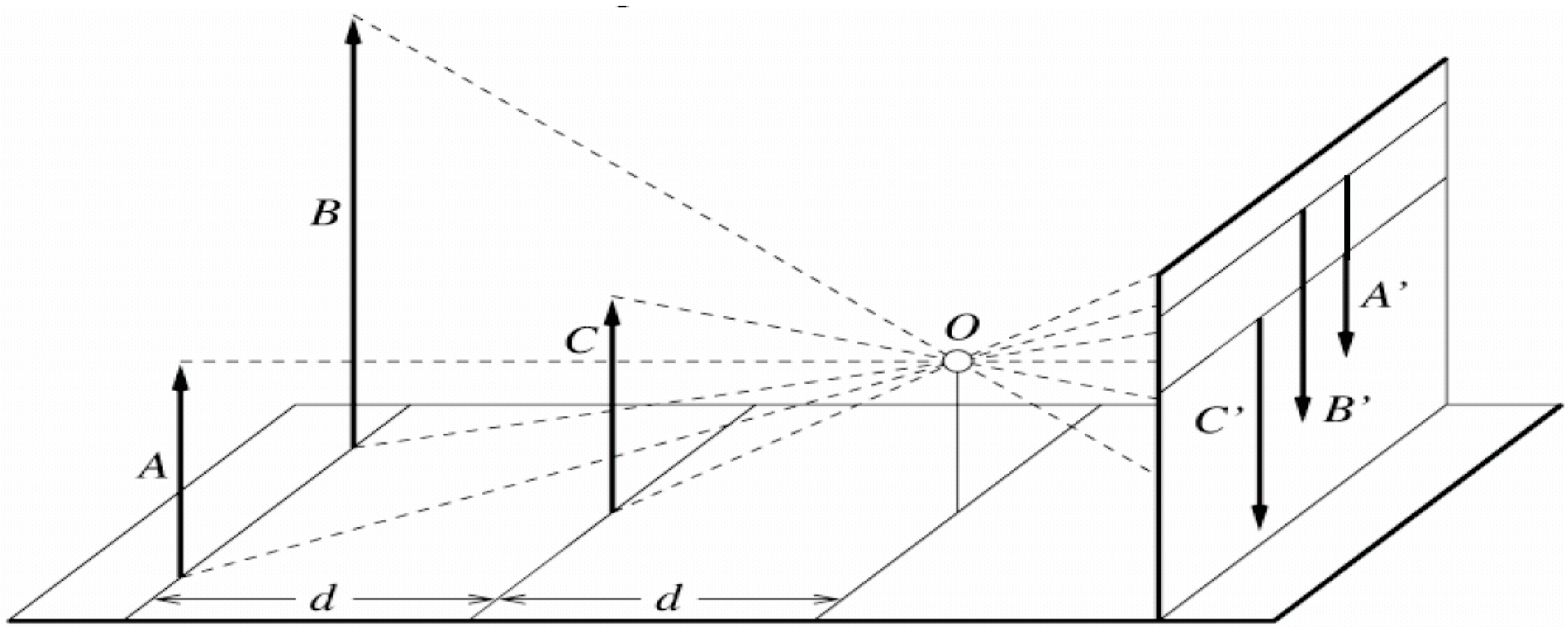
The camera | why use a lens?

- A lens focuses light onto the film
- Rays passing through the **optical center** are not deviated
- All rays parallel to the **optical axis** converge at the **focal point**



Perspective effects

- Far away objects appear smaller

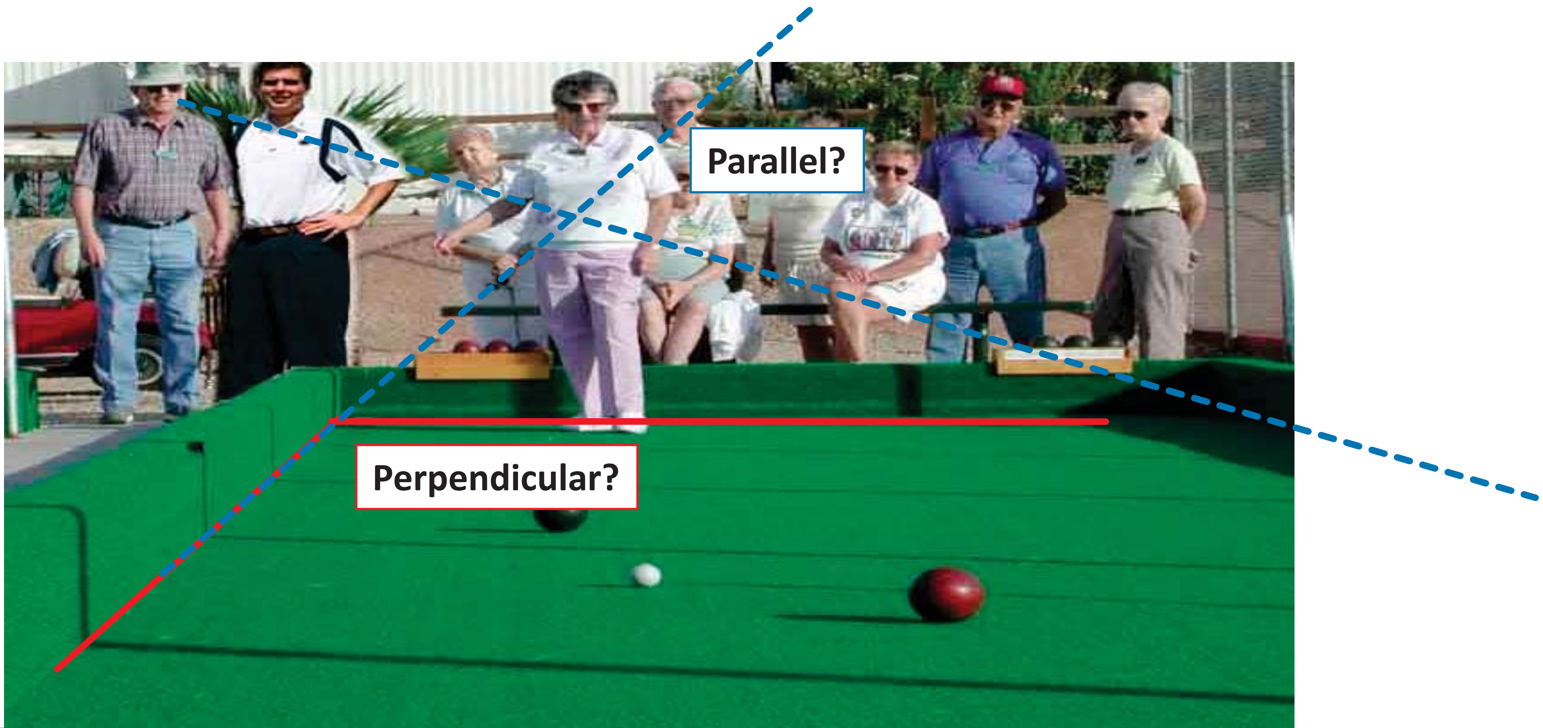


Perspective effects

Projective Geometry

What is lost?

- Length
- Angles



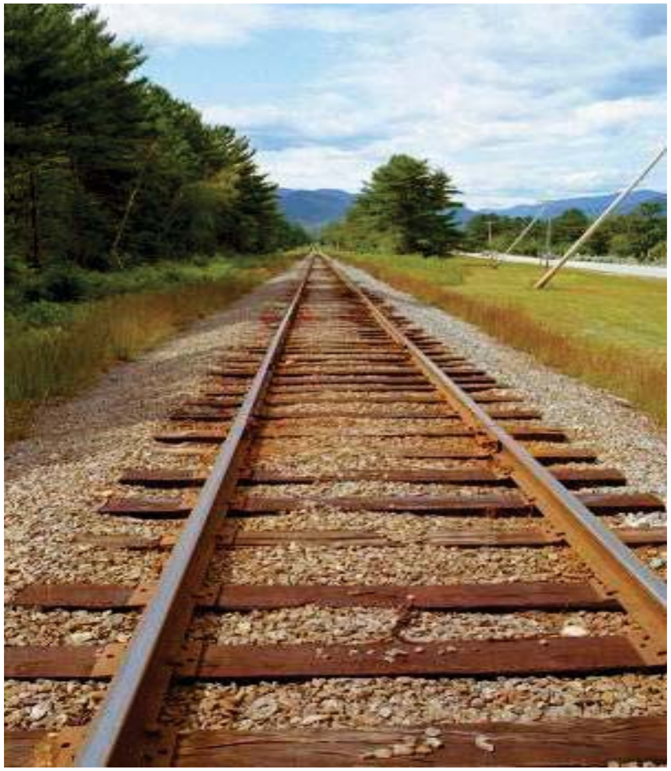
Projective Geometry

What is preserved?

- Straight lines are still straight

Vanishing points and lines

- Parallel lines in the world intersect in the image at a “vanishing point”



Vanishing points and lines

Perspective and art

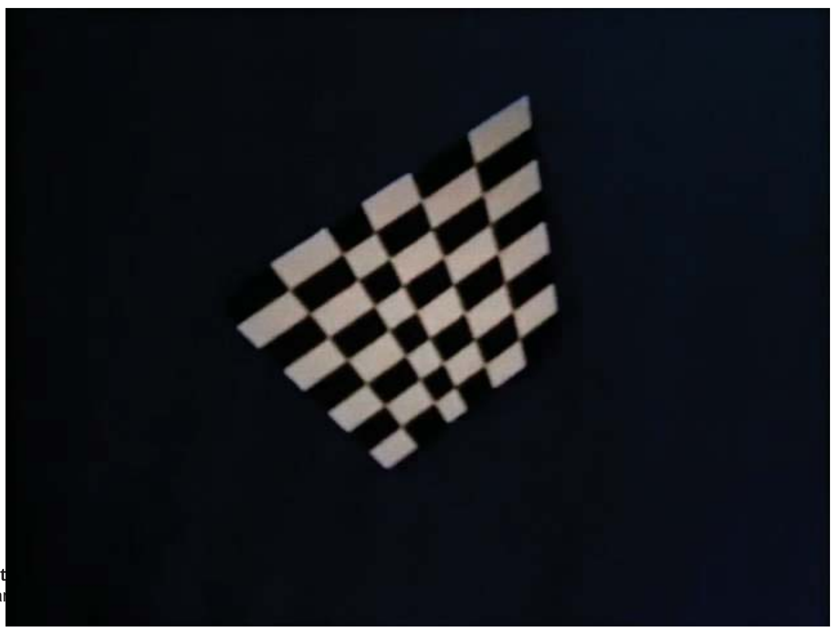
- Use of correct perspective projection indicated in 1st century B.C. frescoes
- Skill resurfaces in Renaissance: artists develop systematic methods to determine perspective projection (around 1480-1515)

Raphael

Durer, 1525

Playing with Perspective

- Perspective gives us very strong depth cues
  ⇨ hence we can perceive a 3D scene by viewing its 2D representation (i.e. image)
- An example where perception of 3D scenes is misleading:



"Ames room"

A clip from "The computer that ate Hollywood" documentary. Dr. Vilayanur S. Ramachandran.

Video

Outline of this lecture

- Perspective camera model
- Lens distortion
- Camera calibration
  - DLT algorithm

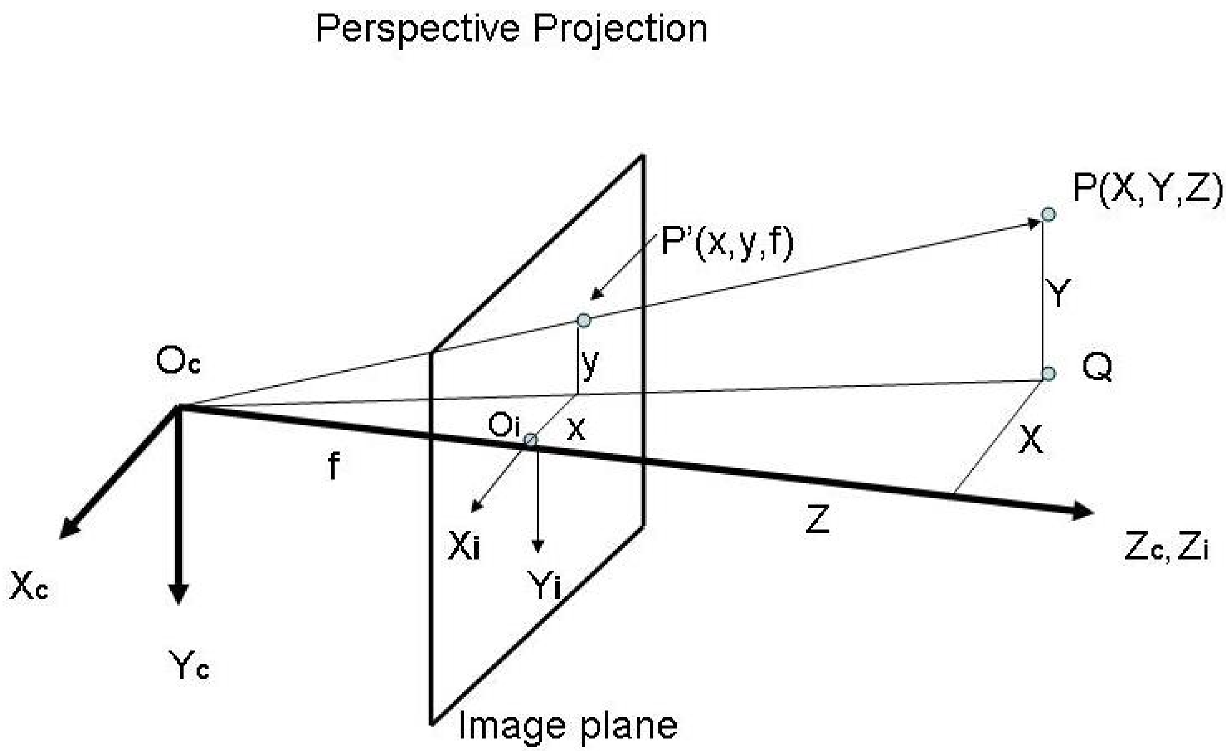


Figure 1: Perspective imaging geometry showing relationship between 3D points and image plane points.

1 Camera Model and Perspective Transform

We typically use a pinhole camera model that maps points in a 3-D camera frame to a 2-D projected image frame. In figure 1, we have a 3D camera coordinate frame X_c, Y_c, Z_c with origin O_c, and an image coordinate frame X_i, Y_i, Z_i with origin O_i. The focal length is f. Using similar triangles, we can relate image plane and world space coordinates. We have a 3D point $P = (X, Y, Z)$ which projects onto the image plane at $P' = (x, y, f)$. O_c is the origin of the camera coordinate system, known as the *center of projection* (COP) of the camera.

Using similar triangles, we can write down the folowing relationships:

$$\frac{X}{x} = \frac{Z}{f} \quad ; \quad \frac{Y}{y} = \frac{Z}{f} \quad ; \quad x = f \cdot \frac{X}{Z} \quad ; \quad y = f \cdot \frac{Y}{Z}$$

If $f = 1$, note that perspective projection is just scaling a world coordinate by its Z value. Also note that all 3D points along a line from the COP through a designated position (x, y) on the image plane will have the same image plane coordinates.

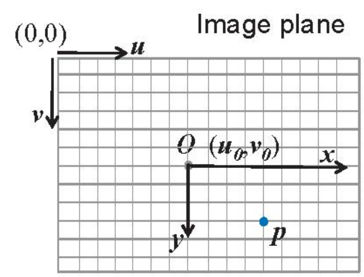
We can also describe perspective projection by the matrix equation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \triangleq \begin{bmatrix} s \cdot x \\ s \cdot y \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where s is a scaling factor and $[x, y, 1]^T$ are the projected coordinates in the image plane.

We can generate *image space* coordinates from the projected camera space coordinates. These are the actual pixels values that you use in image processing. Pixels values (u, v) are derived by scaling the camera image plane coordinates in the x and y directions (for example, converting mm to $pixels$), and adding a translation to the origin of the image space plane. We can call these scale factors D_x and D_y, and the translation to the origin of the image plane as (u_0, v_0).

If the pixel coordinates of a projected point (x,y) are (u,v) then we can write:



$$\frac{x}{D_x} = u - u_0; \quad \frac{y}{D_y} = v - v_0;$$

$$u = u_0 + \frac{x}{D_x}; \quad v = v_0 + \frac{y}{D_y}$$

where D_x, D_y are the physical dimensions of a pixel and (u_0, v_0) is the origin of the pixel coordinate system. $\frac{x}{D_x}$ and $\frac{y}{D_y}$ are simply the number of pixels, and we center them at the pixel coordinate origin. We can also put this into matrix form as:

$$\begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = \begin{bmatrix} \frac{1}{D_x} & 0 & u_0 \\ 0 & \frac{1}{D_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s \cdot x \\ s \cdot y \\ s \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \triangleq \begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = \begin{bmatrix} \frac{1}{D_x} & 0 & u_0 \\ 0 & \frac{1}{D_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$${}^{image}P = {}^{image}T_{persp} \, {}^{persp}T_{camera} \, {}^{camera}P$$

In the above, we assumed that the point to be imaged was in the camera coordinate system. If the point is in a previously defined world coordinate system, then we also have to add in a standard $4x4$ transform to express the world coordinate point in camera coordinates:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = \begin{bmatrix} \frac{1}{D_x} & 0 & u_0 \\ 0 & \frac{1}{D_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^wX \\ {}^wY \\ {}^wZ \\ 1 \end{bmatrix}$$

$$^{image}P \;=\; {}^{image}T_{persp}\,{}^{persp}T_{camera}\,{}^{camera}T_{world}\,{}^{world}P$$

Summing all this up, we can see that we need to find the following information to transform an arbitrary 3D world point to a designated pixel in a computer image:

- 6 parameters that relate the 3D world point to the 3D camera coordinate system (standard 3 translation and 3 rotation): (R, T)
- Focal Length of the camera: f
- Scaling factors in the x and y direcitons on the image plane: (D_x, D_y)
- Translation to the origin of the image plane: (u_0, v_0).

  This is 11 parameters in all. We can break these parameters down into *Extrinsic* parameters which are the 6-DOF transform between the camera coordinate system and the world coordinate system, and the *Intrinsic* parameters which are unique to the actual camera being used, and include the focal length, scaling factors, and location of the origin of the pixel coordinate system.

2 Camera Calibration

Camera calibration is used to find the mapping from 3D to 2D image space coordinates. There are 2 approaches:

- Method I: Find both extrinsic and intrinsic parameters of the camera system. However, this can be difficult to do. The instinsic parameters of the camera may be unknown (i.e. focal length, pixel dimension) and the 6-DOF transform also may be difficult to calculate directly.
- Method 2: An easier method is the "Lumped" or Direct Linear Transform (DLT) method. Rather than finding individual parameters, we find a composite matrix that relates 3D to 2D. Given the equation below:

  $$^{image}P \;=\; {}^{image}T_{persp}\,{}^{persp}T_{camera}\,{}^{camera}T_{world}\,{}^{world}P$$

  we can lump the 3 T matrices into a 3x4 calibration matrix C:

  $$^{image}P \;=\; C\,{}^{world}P$$

  $$C \;=\; {}^{image}T_{persp}\,{}^{persp}T_{camera}\,{}^{camera}T_{world}$$

- C is a <u>single</u> 3×4 transform that we can calculate empirically.

$$\overbrace{\begin{bmatrix} C \end{bmatrix}}^{3\times4} \quad \underbrace{\overbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}^{4\times1}}_{\text{3-D homo. vec}} = \underbrace{\overbrace{\begin{bmatrix} u \\ v \\ w \end{bmatrix}}^{3\times1}}_{\text{2-D homo. vec}} \triangleq \underbrace{\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}}_{\text{Pixels}} \quad \begin{matrix} where \\ u' = \frac{u}{w} \\ v' = \frac{v}{w} \end{matrix}$$

- Multiplying out the equations, we get:

$$c_{11}x + c_{12}y + c_{13}z + c_{14} = u$$

$$c_{21}x + c_{22}y + c_{23}z + c_{24} = v$$

$$c_{31}x + c_{32}y + c_{33}z + c_{34} = w$$

- Substituting $u = u'w$ and $v = v'w$, we get:

  1. $c_{11}x + c_{12}y + c_{13}z + c_{14} = u'(c_{31}x + c_{32}y + c_{33}z + c_{34})$
  2. $c_{21}x + c_{22}y + c_{23}z + c_{24} = v'(c_{31}x + c_{32}y + c_{33}z + c_{34})$

- How to interpret <u>1</u> and <u>2</u>:

  1. If we know all the c_{ij} and x, y, z, we can find u', v'. This means that if we know calibration matrix C and a 3-D point, we can predict its image space coordinates.
  2. If we know $x,\ y,\ z,\ u',\ v'$, we can find c_{ij}. Each 5-tuple gives 2 equations in c_{ij}. This is the basis for empirically finding the calibration matrix C (more on this later).
  3. If we know $c_{ij},\ u',\ v'$, we have 2 equations in x, y, z. They are the equations of 2 planes in 3-D. 2 planes form an intersecton which is a line. These are the equations of the line emanating from the center of projection of the camera, through the image pixel location u', v' and which contains point x, y, z.

- We can set up a linear system to solve for c_{ij}: $AC = B$

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u'_1x & -u'_1y & -u'_1z \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v'_1x & -v'_1y & -v'_1z \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u'_2x & -u'_2y & -u'_2z \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v'_2x & -v'_2y & -v'_2z \\ . \\ . \\ . \\ . \\ . \\ . \\ . \end{bmatrix} \underbrace{\begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{31} \\ c_{32} \\ c_{33} \end{bmatrix}}_{\text{We can assume } c_{34}=1} = \begin{bmatrix} u'_1 \\ v'_1 \\ u'_2 \\ v'_2 \\ u'_3 \\ v'_3 \\ . \\ . \\ . \\ u'_N \\ v'_N \end{bmatrix}$$

- Each set of points x, y, z, u', v' yields 2 equations in 11 unknowns (the c_{ij}'s).

- To solve for C, A needs to be invertible (square). We can overdetermine A and find a Least-Squares fit for C by using a pseudo-inverse solution.

  If A is $N \times 11$, where $N > 11$,

$$AC = B$$

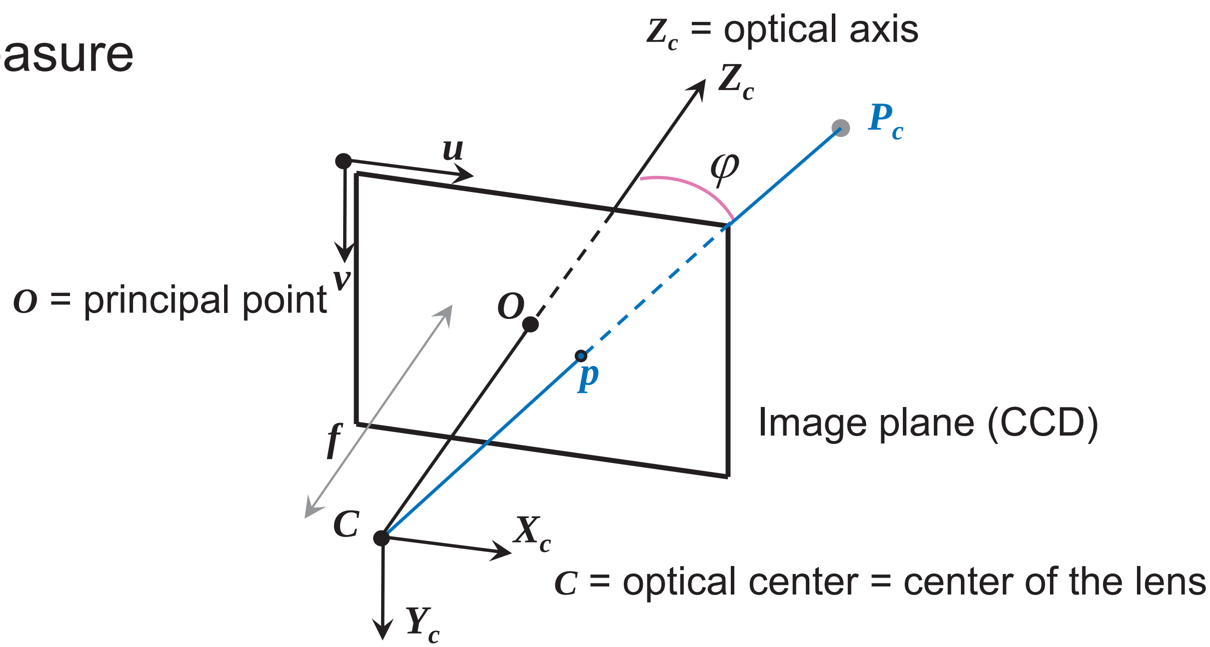
$$A^T AC = A^T B$$

$$C = \underbrace{(A^T A)^{-1}}_{\text{pseudo inverse}} A^T B$$

The camera | perspective camera

- For convenience, the image plane is usually represented in front of C such that the image preserves the same orientation (i.e. not flipped)

- A camera does not measure distances but angles!



Perspective projection| from scene points to pixels

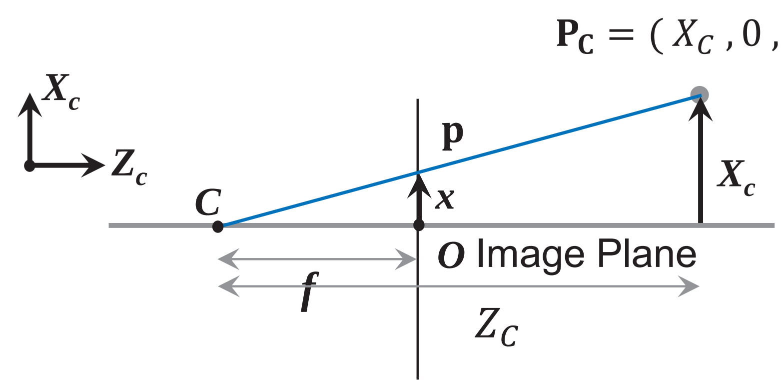
- The Camera point $\mathbf{P_C} = (X_C, 0, Z_C)^{\mathrm{T}}$ projects to $\mathbf{p} = (x, y)$ onto the image plane

- From similar triangles:

$$\frac{x}{f} = \frac{X_c}{Z_c} \Rightarrow x = \frac{fX_c}{Z_c}$$

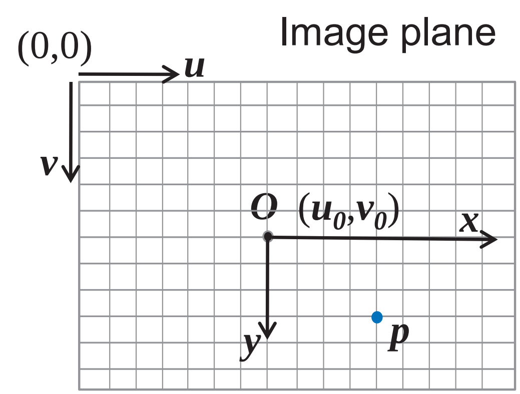
- Similarly, in the general case:

$$\frac{y}{f} = \frac{Y_c}{Z_c} \Rightarrow y = \frac{fY_c}{Z_c}$$



Perspective projection| from scene points to pixels

- To convert $\mathbf{p}$, from the local image plane coordinates (x, y) to the pixel coordinates (u, v), we need to account for:
  - The pixel coordinates of the camera optical center $O = (u_0, v_0)$
  - Scale factor k for the pixel-size



$$u = u_0 + kx \Rightarrow u_0 + k\frac{fX_C}{Z_C}$$

$$v = v_0 + ky \Rightarrow v_0 + k\frac{fY_C}{Z_C}$$

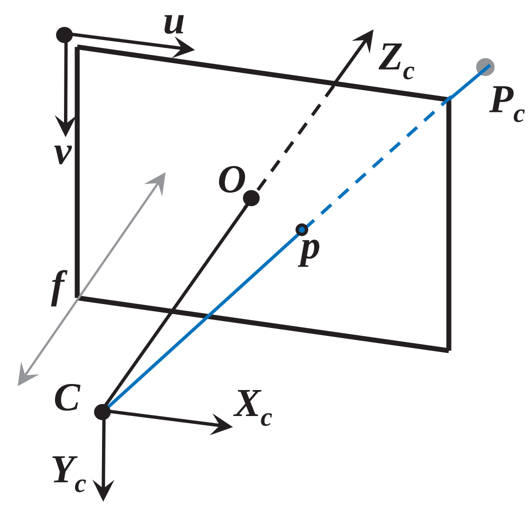
- Use Homogeneous Coordinates for linear mapping from 3D to 2D, by introducing an extra element (scale):

$$p = \begin{pmatrix} u \\ v \end{pmatrix} \Rightarrow \tilde{p} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Perspective projection| from scene points to pixels

$$u = u_0 + kx \Rightarrow u_0 + k\frac{fX_C}{Z_C}$$

$$v = v_0 + ky \Rightarrow v_0 + k\frac{fY_C}{Z_C}$$



- Expressed in matrix form and homogeneous coordinates:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} kf & 0 & u_0 \\ 0 & kf & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

- Or alternatively

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Focal length in pixels

Intrinsic parameters matrix

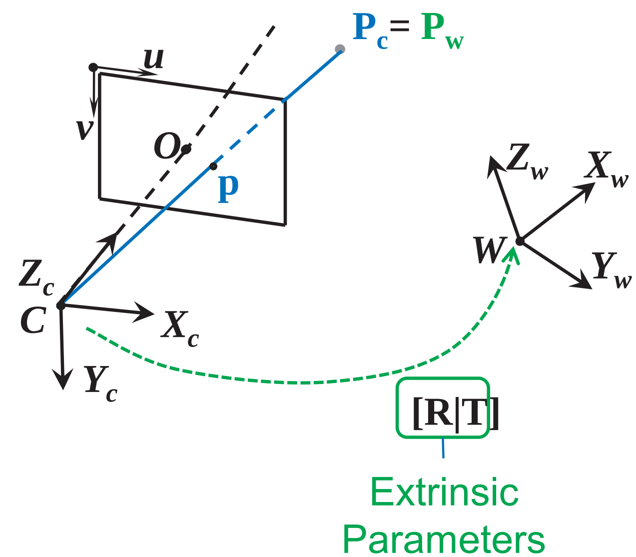
Perspective projection| from scene points to pixels

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} R & | & T \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Perspective Projection Matrix

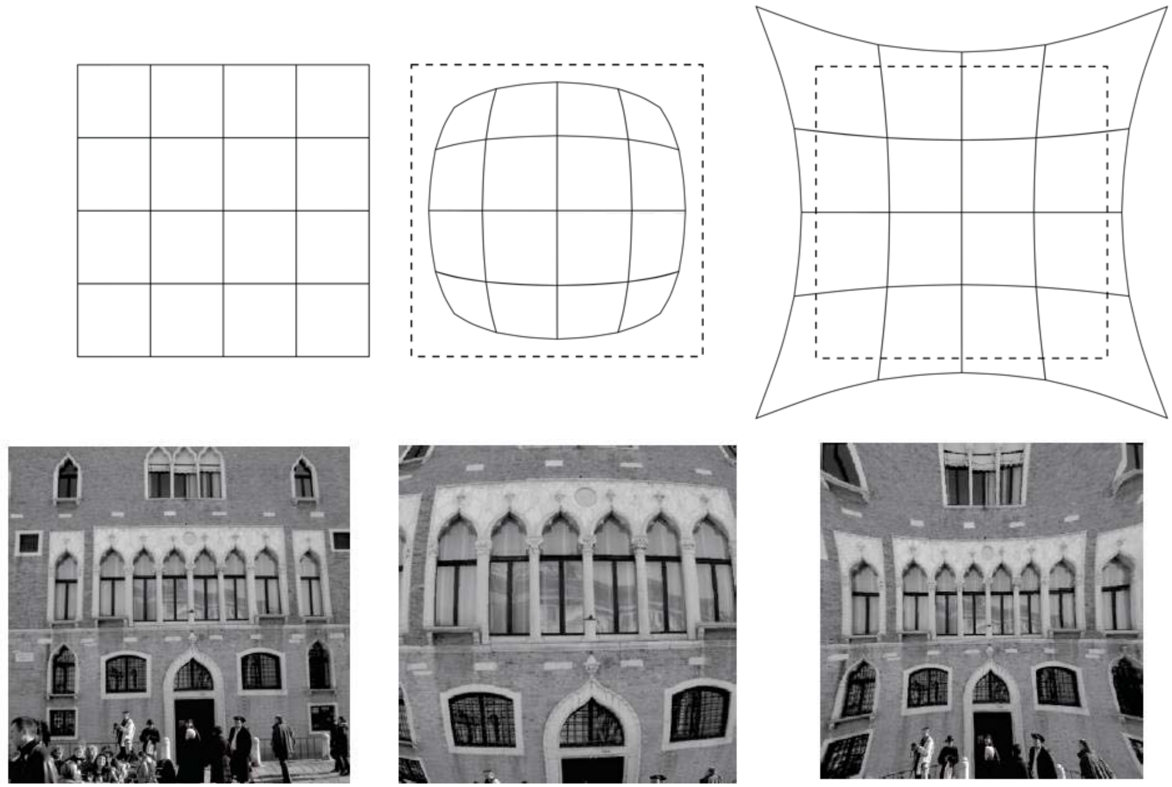
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boxed{K\left[R|T\right]} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$



Outline of this lecture

- Perspective camera model
- Lens distortion
- Camera calibration
  - DLT algorithm
- Stereo vision

Perspective projection| radial distortion



No distortion | Barrel distortion | Pincushion

Perspective projection| radial distortion

- The standard model of radial distortion is a transformation from the ideal coordinates (u, v) (i.e., undistorted) to the real observable coordinates (distorted) (u_d, v_d)

- The amount of distortion of the coordinates of the observed image is a nonlinear function of their radial distance. For most lenses, a simple quadratic model of distortion produces good results

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

where

$$r^2 = (u - u_0)^2 + (v - v_0)^2$$

Summary: Perspective projection equations

- To recap, a 3D world point $P = (X_w, Y_w, Z_w)$ projects into the image point $p = (u, v)$

$$\lambda p = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\left[R \mid T\right] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \qquad \text{where} \qquad K = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

and λ is the depth ($\lambda = Z_w$) of the scene point

- If we want to take into account for the radial distortion, then the distorted coordinates (u_d, v_d) (in pixels) can be obtained as

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

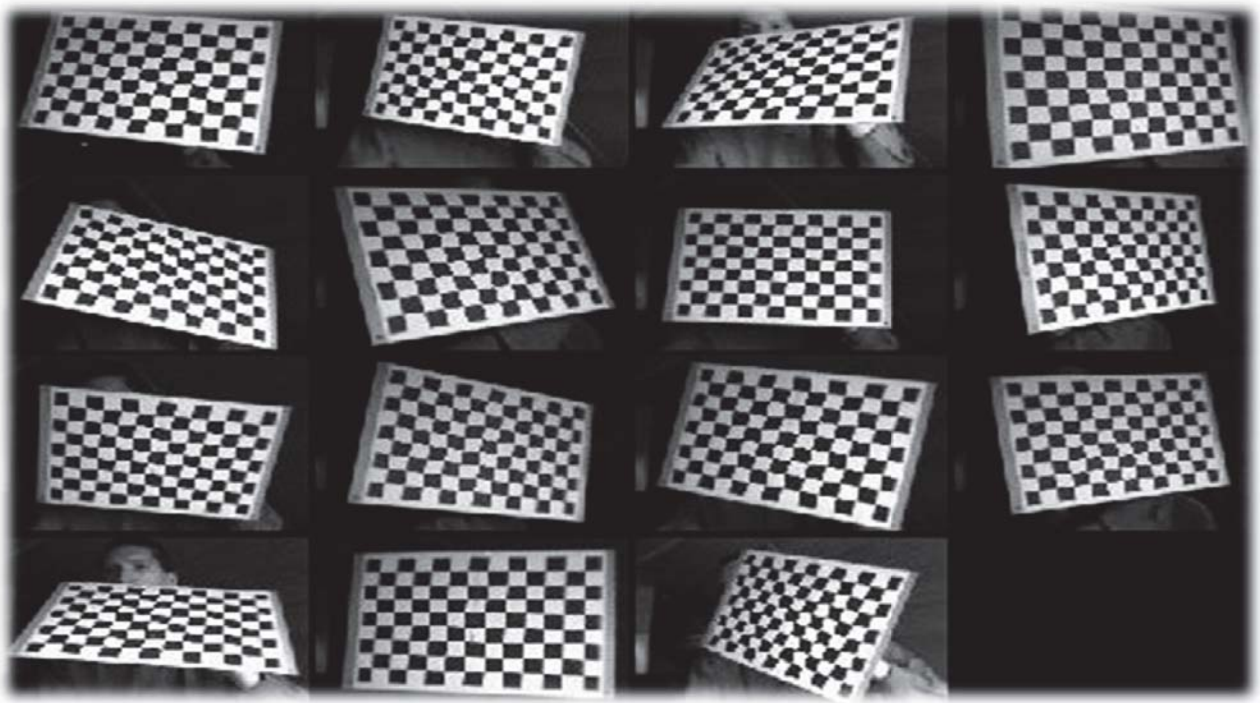
where $r^2 = (u - u_0)^2 + (v - v_0)^2$

Outline of this lecture

- Perspective camera model
- Lens distortion
- Camera calibration
  - DLT algorithm
- Stereo vision

Camera Calibration

- Procedure to determine the *intrinsic parameters* of a camera

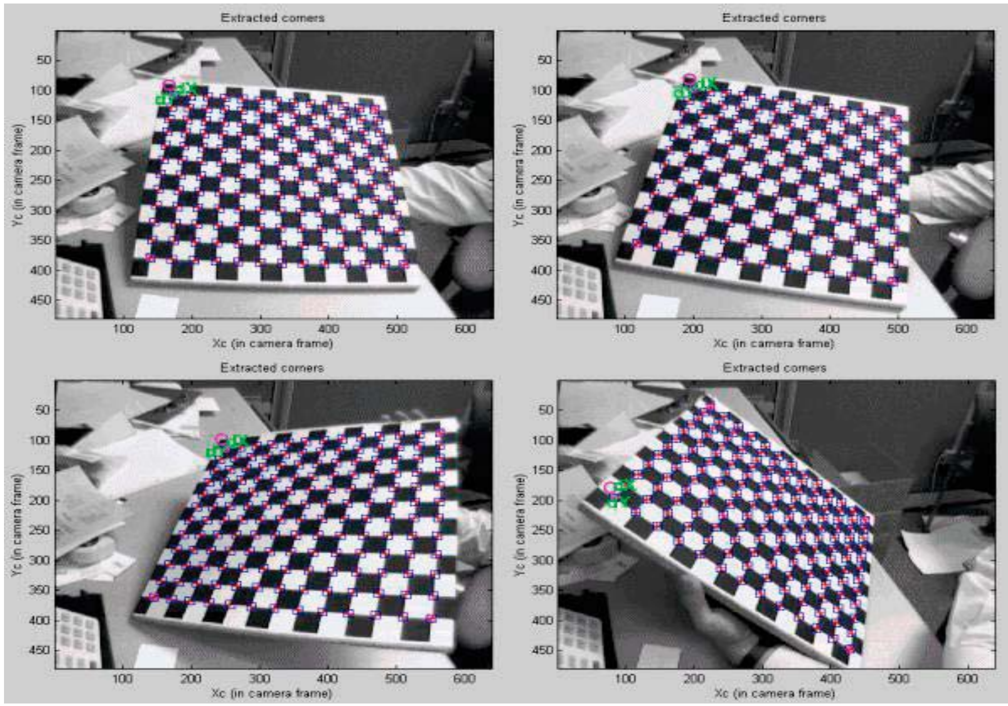


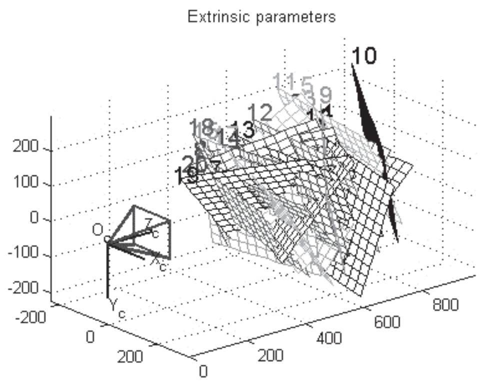
Camera Calibration

- Use camera model to interpret the projection from world to image plane
- Using known correspondences of $\boldsymbol{p} \Leftrightarrow \boldsymbol{P}$, we can compute the unknown parameters $\boldsymbol{K}$, $\boldsymbol{R}$, $\boldsymbol{T}$ by applying the perspective projection equation
- … so associate known, physical distances in the world to pixel-distances in image

Projection Matrix

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boxed{K[R|T]} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$





Camera Calibration (Direct Linear Transform (DLT) algorithm)

- We know that : $\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|T] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

- So there are 11 values to estimate: (the overall scale doesn't matter, so e.g. m_{34} could be set to 1)
- Each observed point gives us a pair of equations:

$$u_i = \frac{\lambda u_i}{\lambda} = \frac{m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14}}{m_{31} + m_{32} + m_{33} + m_{34}}$$

$$v_i = \frac{\lambda v_i}{\lambda} = \frac{m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24}}{m_{31} + m_{32} + m_{33} + m_{34}}$$

- To estimate 11 unknowns, we need **at least** ? points to calibrate the camera ⇨ solved using linear least squares

Camera Calibration (Direct Linear Transform (DLT) algorithm)

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R \mid T] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

- **what we obtained**: the 3x4 projection matrix,
  what we need: its decomposition into the camera calibration matrix $\boldsymbol{K}$, and the rotation $\boldsymbol{R}$ and position $\boldsymbol{T}$ of the camera.

- Use QR factorization to decompose the 3x3 submatrix ($m_{11:33}$) into the product of an upper triangular matrix $\boldsymbol{K}$ and a rotation matrix $\boldsymbol{R}$ (orthogonal matrix)

- The translation $\boldsymbol{T}$ can subsequently be obtained by:

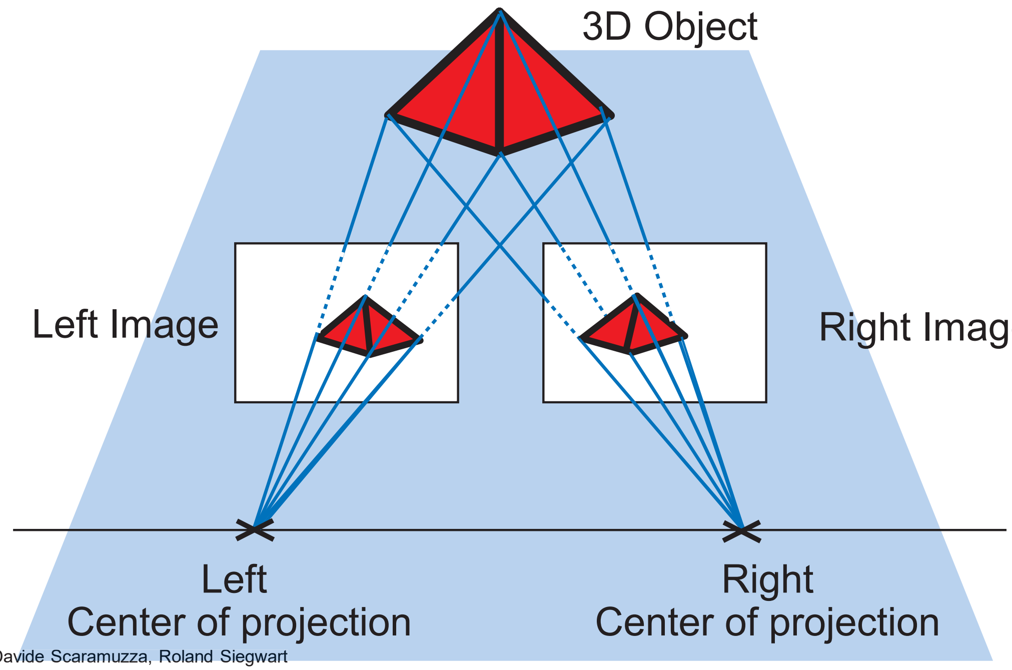
$$T = K^{-1} \begin{bmatrix} m_{14} \\ m_{24} \\ m_{34} \end{bmatrix}$$

Outline of this lecture

- Perspective camera model
- Lens distortion
- Camera calibration
  - DLT algorithm
- Stereo vision

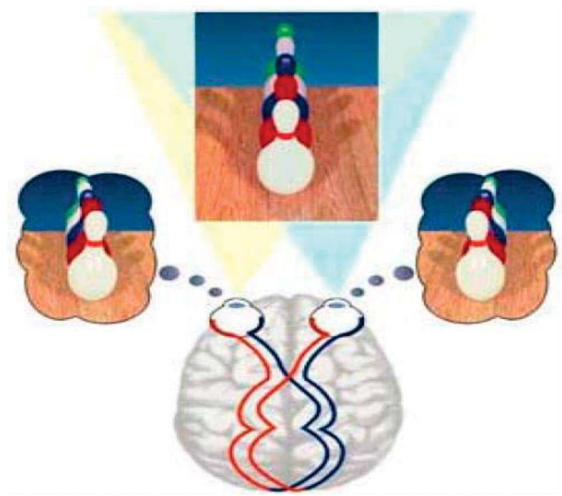
Depth from Stereo

- From a single camera, we can only deduct the **ray** on which each image point lies
- With a stereo camera (binocular), we can solve for the intersection of the rays and recover the 3D structure



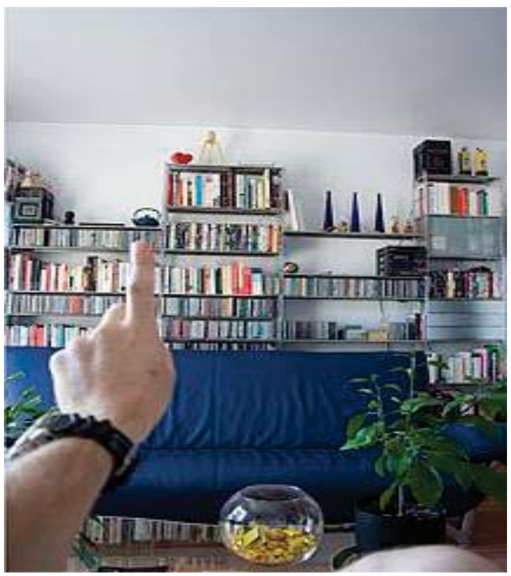
The “human” binocular system

- **Stereopsys:** the brain allows us to see the left and right retinal images as a single 3D image
- The images project on our retina up-side-down but our brains lets us perceive them as «straight». Radial disotion is also removed. This process is called «**rectification**»



The “human” binocular system

- **Stereopsys:** the brain allows us to see the left and right retinal images as a single 3D image
- The images project on our retina up-side-down but our brains lets us perceive them as «straight». Radial disotion is also removed. This process is called «**rectification**»



Make a simple test:
1. Fix an object
2. Open and close alternatively the left and right eyes.
- The horizontal displacement is called **disparity**
- The smaller the disparity, the farther the object

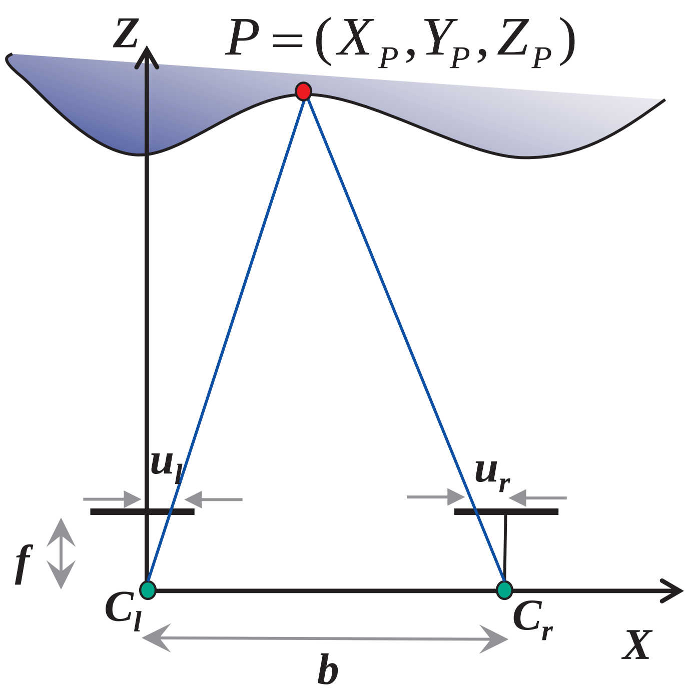
The “human” binocular system

- **Stereopsys:** the brain allows us to see the left and right retinal images as a single 3D image
- The images project on our retina up-side-down but our brains lets us perceive them as «straight». Radial disotion is also removed. This process is called «**rectification**»

Make a simple test:
1. Fix an object
2. Open and close alternatively the left and right eyes.
- The horizontal displacement is called **disparity**
- The smaller the disparity, the farther the object

Stereo Vision - The simplified case

- The simplified case is an ideal case. It assumes that both cameras are identical and are aligned on a horizontal axis



Baseline
distance between the optical centers of the two cameras

From Similar Triangles:

$$\frac{f}{Z_P} = \frac{u_l}{X_P}$$

$$\frac{f}{Z_P} = \frac{u_r}{b - X_P}$$ <---should be -Ur

$$Z_P = \frac{bf}{u_l - u_r}$$

Disparity
difference in image location of the projection of a 3D point in two image planes

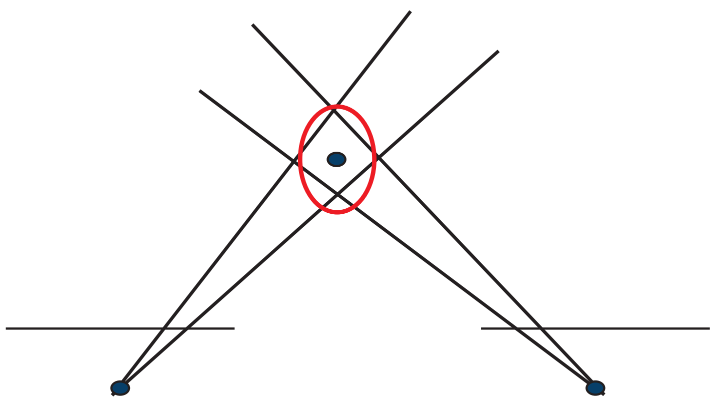
Stereo Vision facts

$$Z_P = \frac{bf}{u_l - u_r}$$

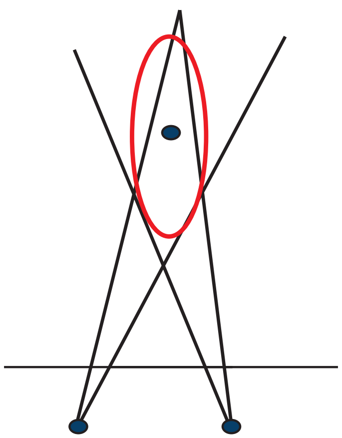
1. Depth is inversely proportional to disparity $(u_l - u_r)$
   - Foreground objects have bigger disparity than background objects
2. Disparity is proportional to stereo-baseline $\boldsymbol{b}$
   - The smaller the baseline $\boldsymbol{b}$ the more uncertain our estimate of depth
   - However, as $\boldsymbol{b}$ is increased, some objects may appear in one camera, but not in the other (remember both cameras have parallel optical axes)
3. The projections of a single 3D point onto the left and the right stereo images are called **'correspondence pair' or a 'stereo pair'**

Choosing the Baseline

- What's the optimal baseline?
  - **Too small:**
    - Large depth error
    - Can you quantify the error as a function of the disparity?
  - **Too large:**
    - Minimum measurable distance increases
    - Difficult search problem for close objects



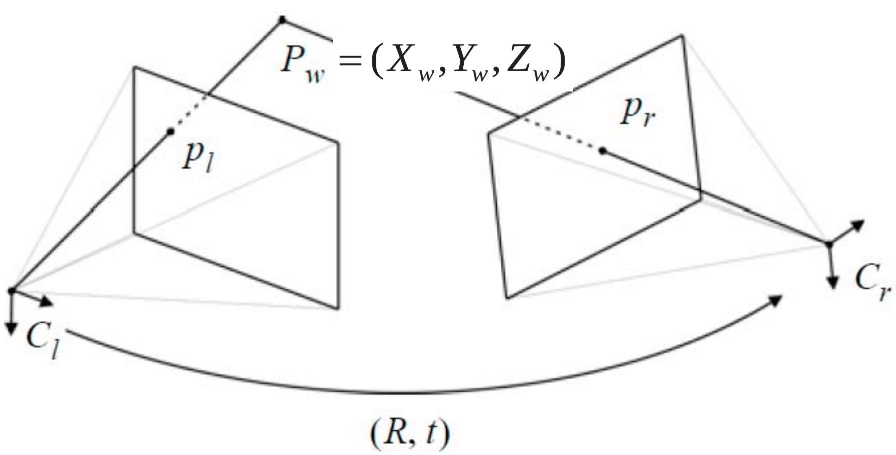
Large Baseline



Small Baseline

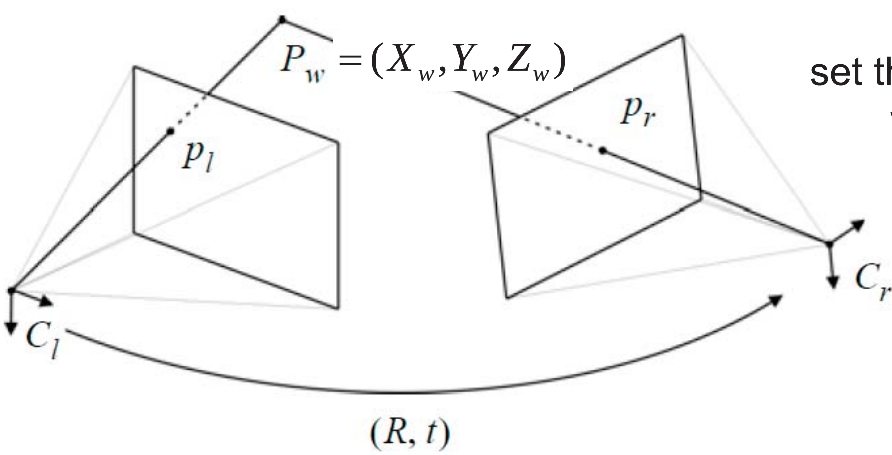
Stereo Vision | general case

- Two identical cameras do not exist in nature!
- Aligning both cameras on a horizontal axis is very difficult
- In order to use a stereo camera, we need to know the intrinsic extrinsic parameters of each camera, that is, the relative pose between the cameras (rotation, translation) ⇒ We can solve for this through camera calibration



Stereo Vision | general case

- To estimate the 3D position of P_W we can construct the system of equations of the left and right camera
- Triangulation is the problem of determining the 3D position of a point given a set of corresponding image locations and known camera poses.



Left camera:
set the world frame to coincide with the left camera frame

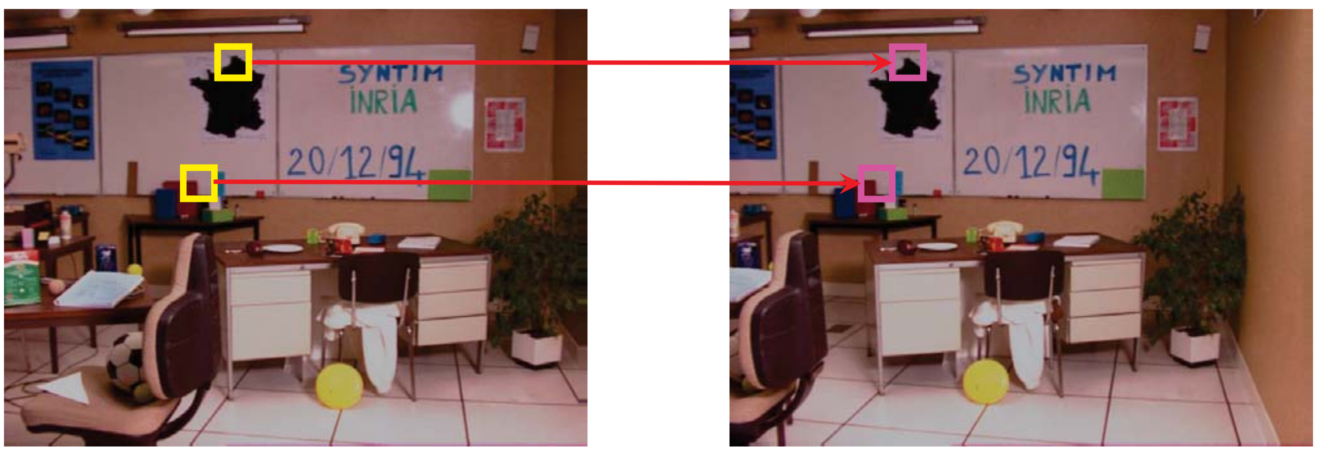
$$\tilde{p}_l = \lambda_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = K_l \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$

Right camera:

$$\tilde{p}_r = \lambda_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = K_r R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T$$

Correspondence Search | the problem

- Goal: identify corresponding points in the left and right images, which are the reprojection of the same 3D scene point
  - Typical similarity measures: Normalized Cross-Correlation (NCC) , Sum of Squared Differences (SSD), Sum of Absolute Differences (SAD), Census Transform
  - Exhaustive image search can be computationally very expensive! Can we make the correspondence search in 1D?



Similarity measures

- Sum of Squared Differences (**SSD**)

$$SSD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \left(H(u,v) - F(u,v) \right)^2$$

- Sum of Absolute Differences (**SAD**)

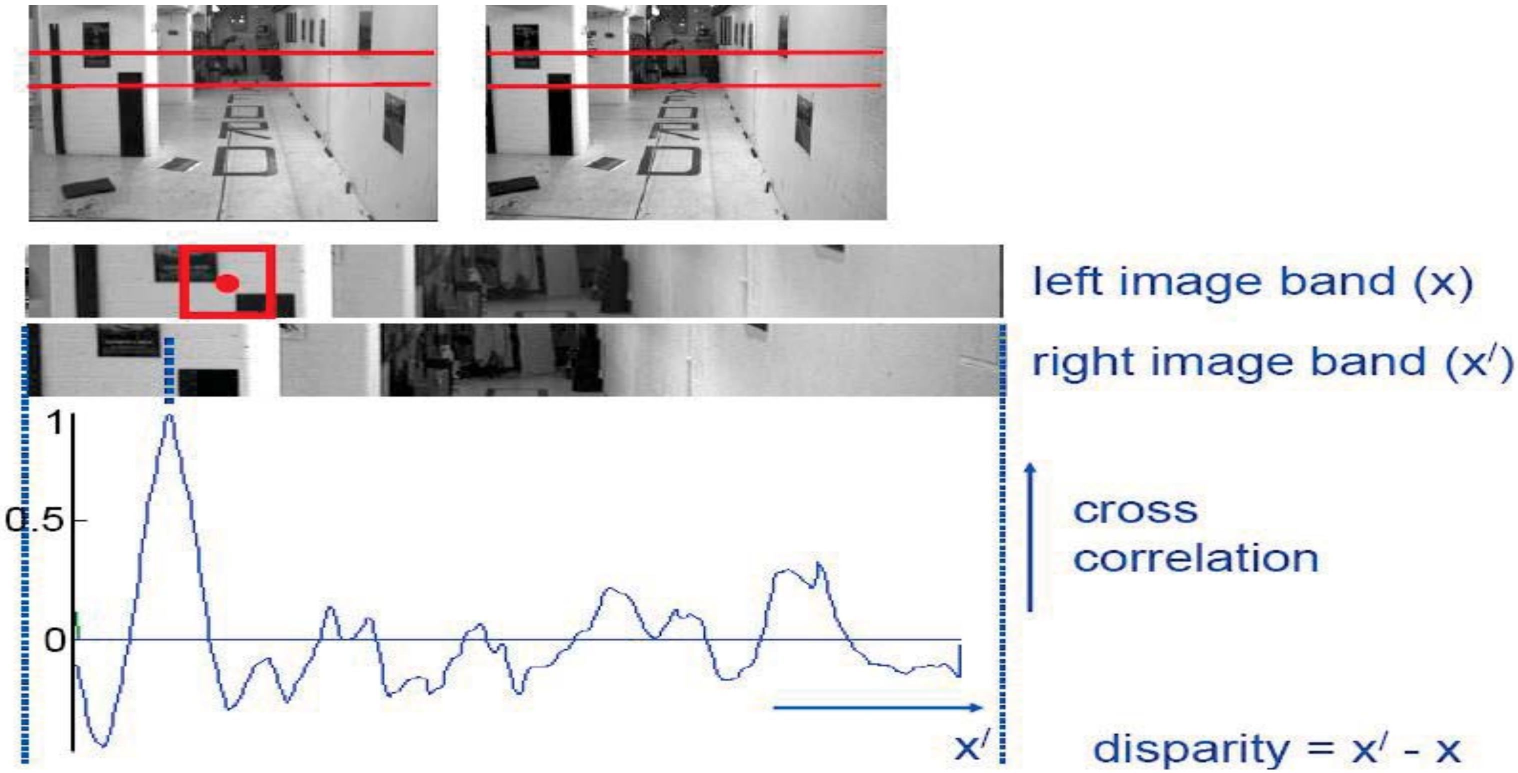
$$SAD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \left| H(u,v) - F(u,v) \right|$$

Similarity measures

- For ***slight* invariance to intensity changes**, the Zero-mean Normalized Cross Correlation (**ZNCC**) is widely used

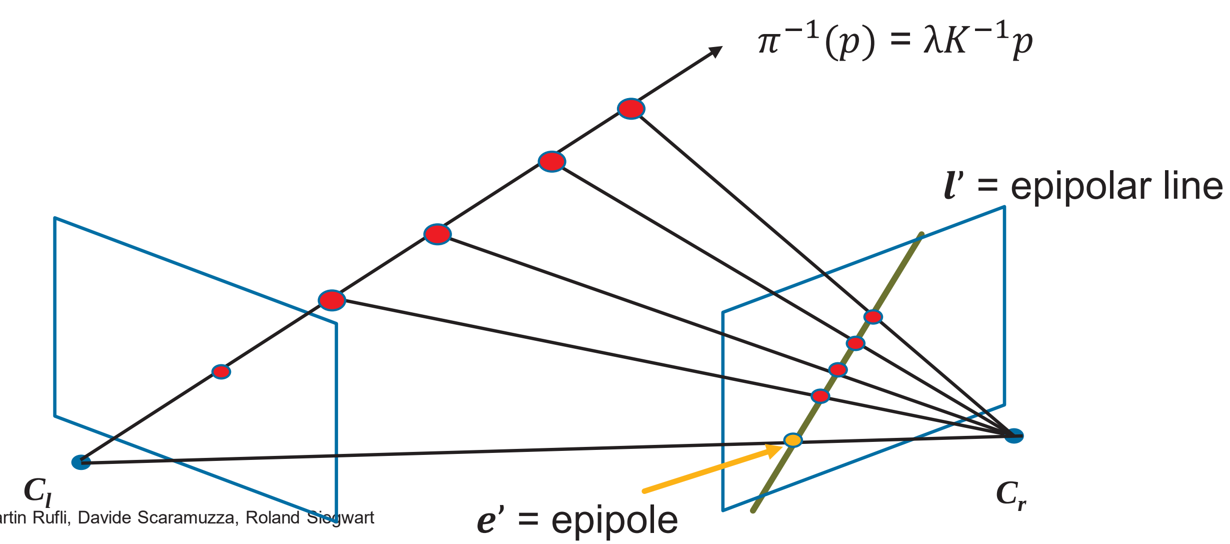
$$ZNCC = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v)-\mu_H\right)\left(F(u,v)-\mu_F\right)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v)-\mu_H\right)^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(F(u,v)-\mu_F\right)^2}} \quad \begin{cases} \mu_H = \dfrac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)}{(2N+1)^2} \\ \mu_F = \dfrac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)}{(2N+1)^2} \end{cases}$$

Correlation-based window matching



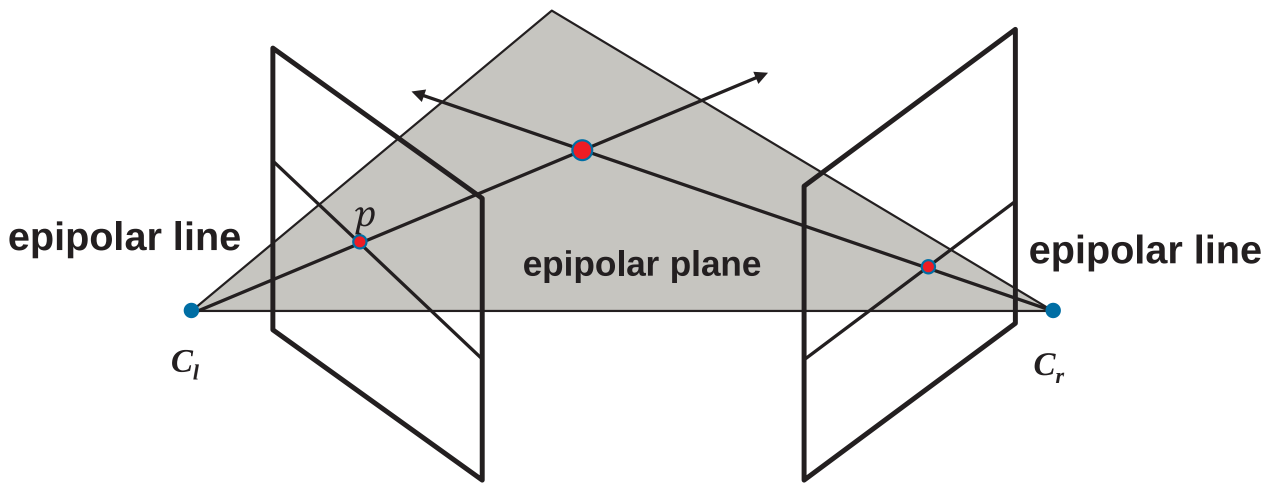
Correspondence Problem

- **Exhaustive** image search can be computationally very expensive!
- Can we make the correspondence search in 1D?
- Potential matches for $\boldsymbol{p}$ have to lie on the corresponding epipolar line $\boldsymbol{l'}$
  - The **epipolar line** is the projection of the infinite ray $\pi^{-1}(p)$ corresponding to $\boldsymbol{p}$ in the other camera image
  - The **epipole** $\boldsymbol{e'}$ is the projection of the optical center in in the other camera image



Correspondence Search | the epipolar constraint

- The epipolar plane is defined by the image point $\mathbf{p}$ and the optical centers
- Impose the epipolar constraint to aid matching: search for a correspondence along the epipolar line

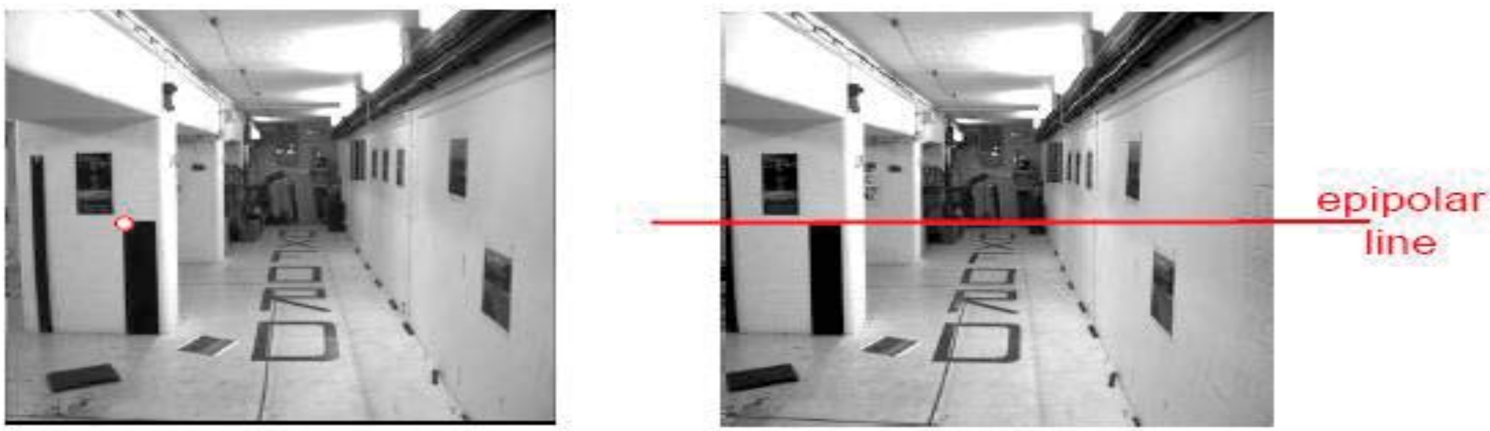


Correspondence Search | the epipolar constraint

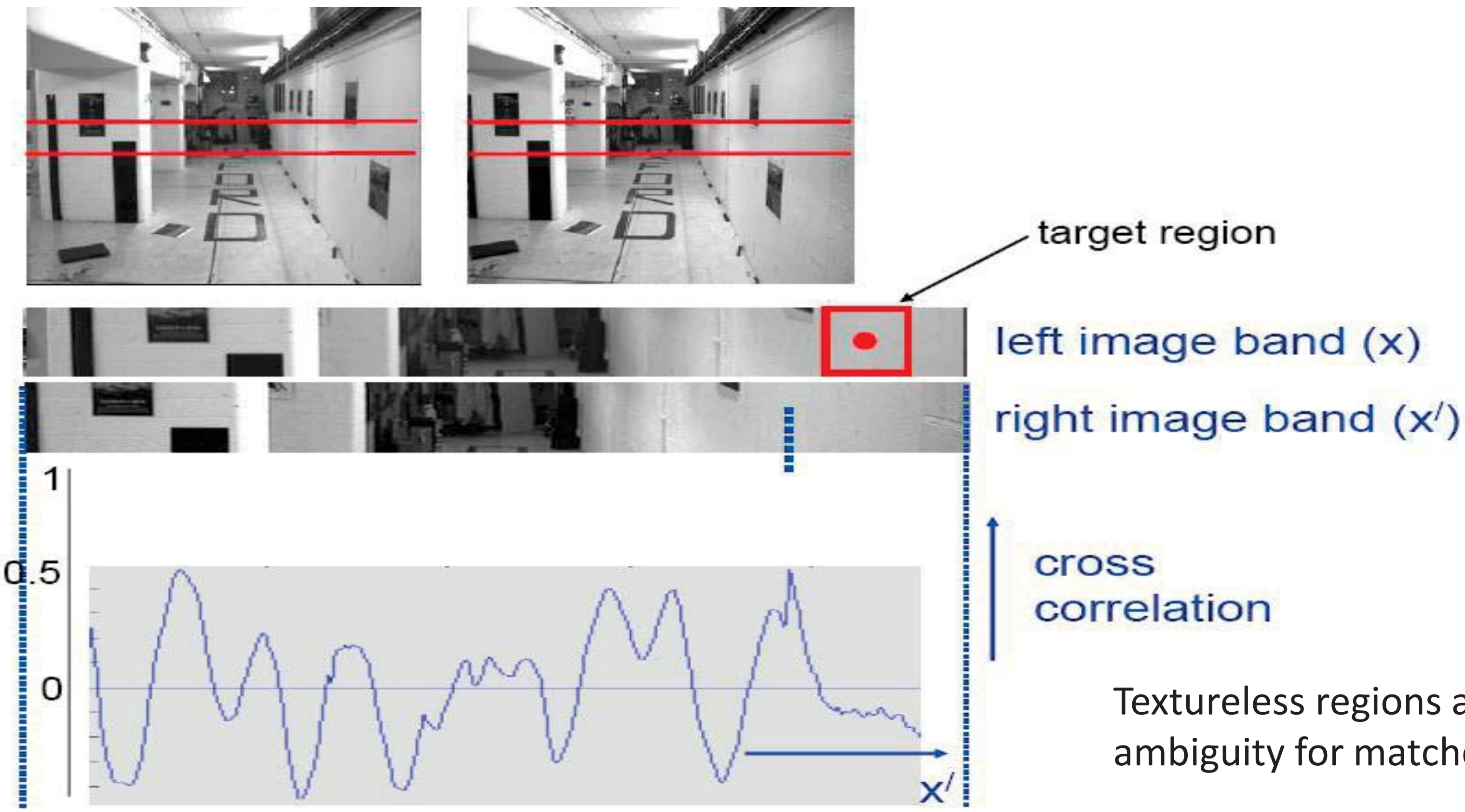
- Thanks to the epipolar constraint, corresponding points can be searched for, along epipolar lines ⇒ computational cost reduced to 1 dimension!

Correspondence problem

- Now that the left and right images are rectified, the correspondence search can be done along the same scanlines

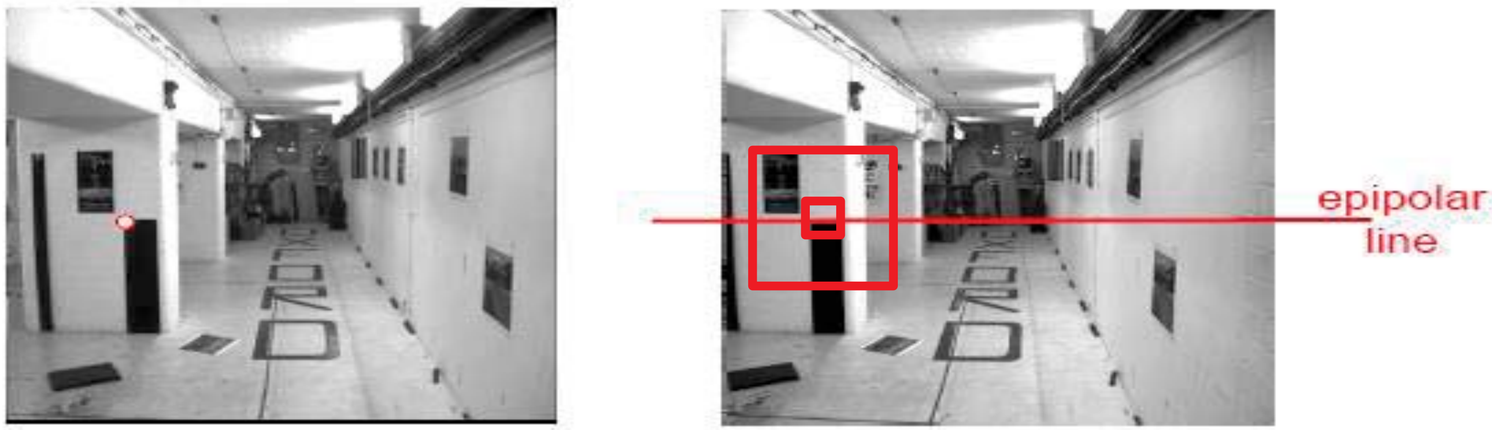


Correspondence Problems:
Textureless regions (the aperture problem)



Textureless regions are non-distinct; high ambiguity for matches.

Solution: increase window size



How can we improve window-based matching?

- Beyond the epipolar constraint, there are “soft” constraints to help identify corresponding points
  - Uniqueness
    - Only one match in right image for every point in left image
  - Ordering
    - Points on **same surface** will be in same order in both views
  - Disparity gradient
    - Disparity changes smoothly between points on the same surface

Grauman

Stereo Vision | disparity map

- The disparity map holds the disparity value at every pixel:
  - Identify correspondent points of all image pixels in the original images
  - Compute the disparity $(u_l - u_r)$ for each pair of correspondences
- Usually visualized in gray-scale images
- Close objects experience bigger disparity; thus, they appear brighter in disparity map

Left image

Right image

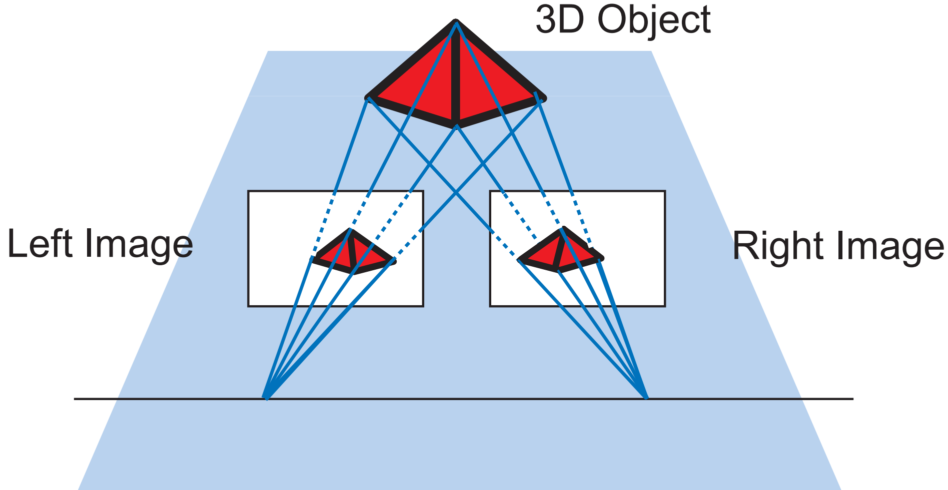
Disparity Map

Stereo Vision | disparity map

- The disparity map holds the disparity value at every pixel:
  - Identify correspondent points of all image pixels in the original images
  - Compute the disparity $(u_l - u_r)$ for each pair of correspondences
- Usually visualized in gray-scale images
- Close objects experience bigger disparity; thus, they appear brighter in disparity map
- From the disparity, we can compute the depth Z as:

$$Z = \frac{bf}{u_l - u_r}$$

Stereo Vision - summary



1. Stereo camera calibration ⇨ compute camera relative pose
2. Epipolar rectification ⇨ align images & epipolar lines
3. Search for correspondences
4. Output: compute stereo triangulation or disparity map
5. Consider how baseline & image resolution affect accuracy of depth estimates

SFM: Structure From Motion (watch video segment)

- Given image point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$, determine R and T
- Keep track of point trajectories over multiple views to reconstruct scene structure and motion