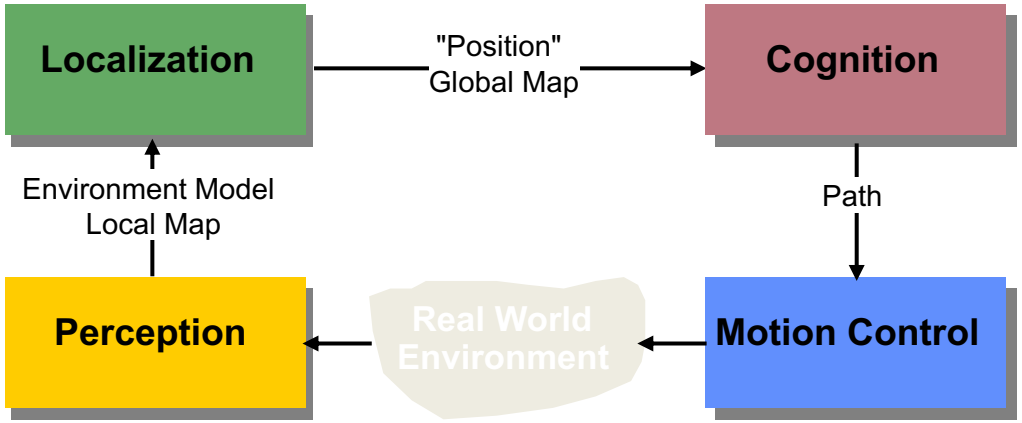
上海科技大学
ShanghaiTech University

CS283: **Robotics Spring 2025: Vision**

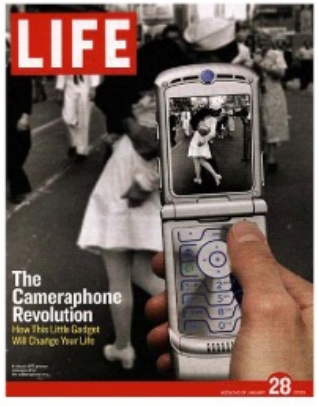
Sören Schwertfeger

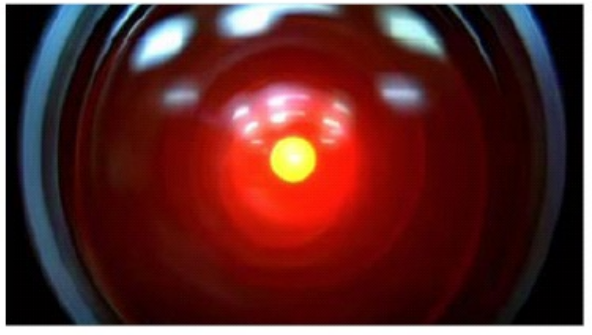
ShanghaiTech University

VISION

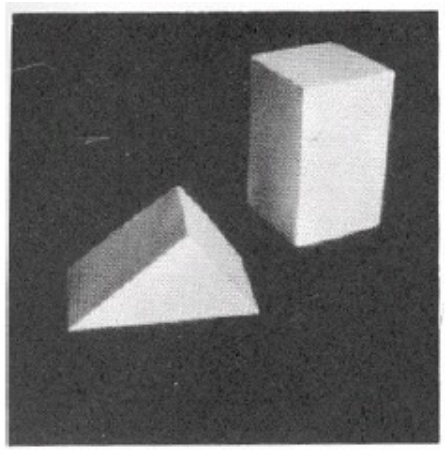

Localization
"Position"
Global Map
Cognition
Environment Model
Local Map
Path
Perception
Real World
Environment
Motion Control

Computer Vision

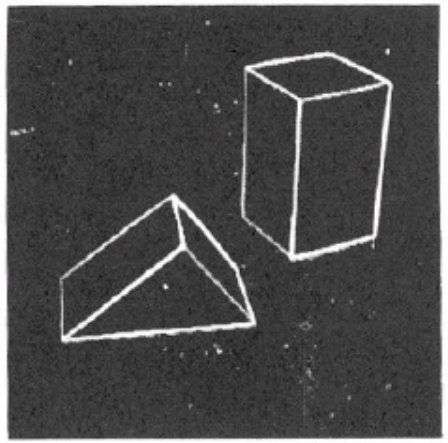




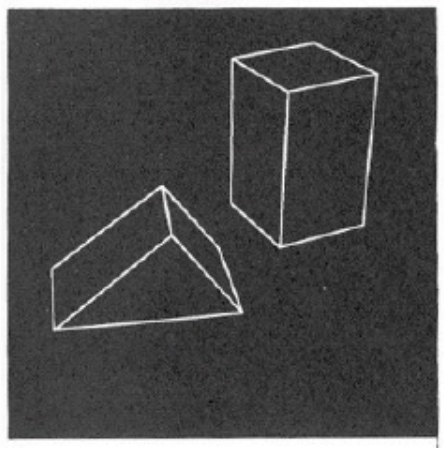
Origins of Computer Vision

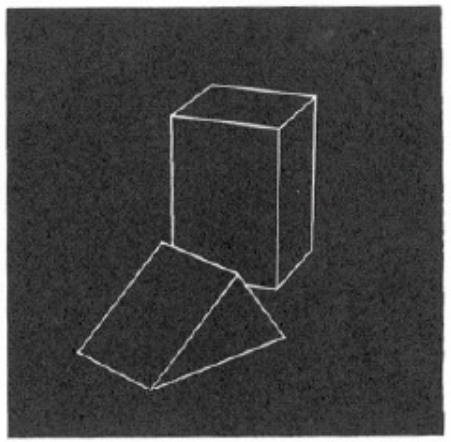


(a) Original picture.



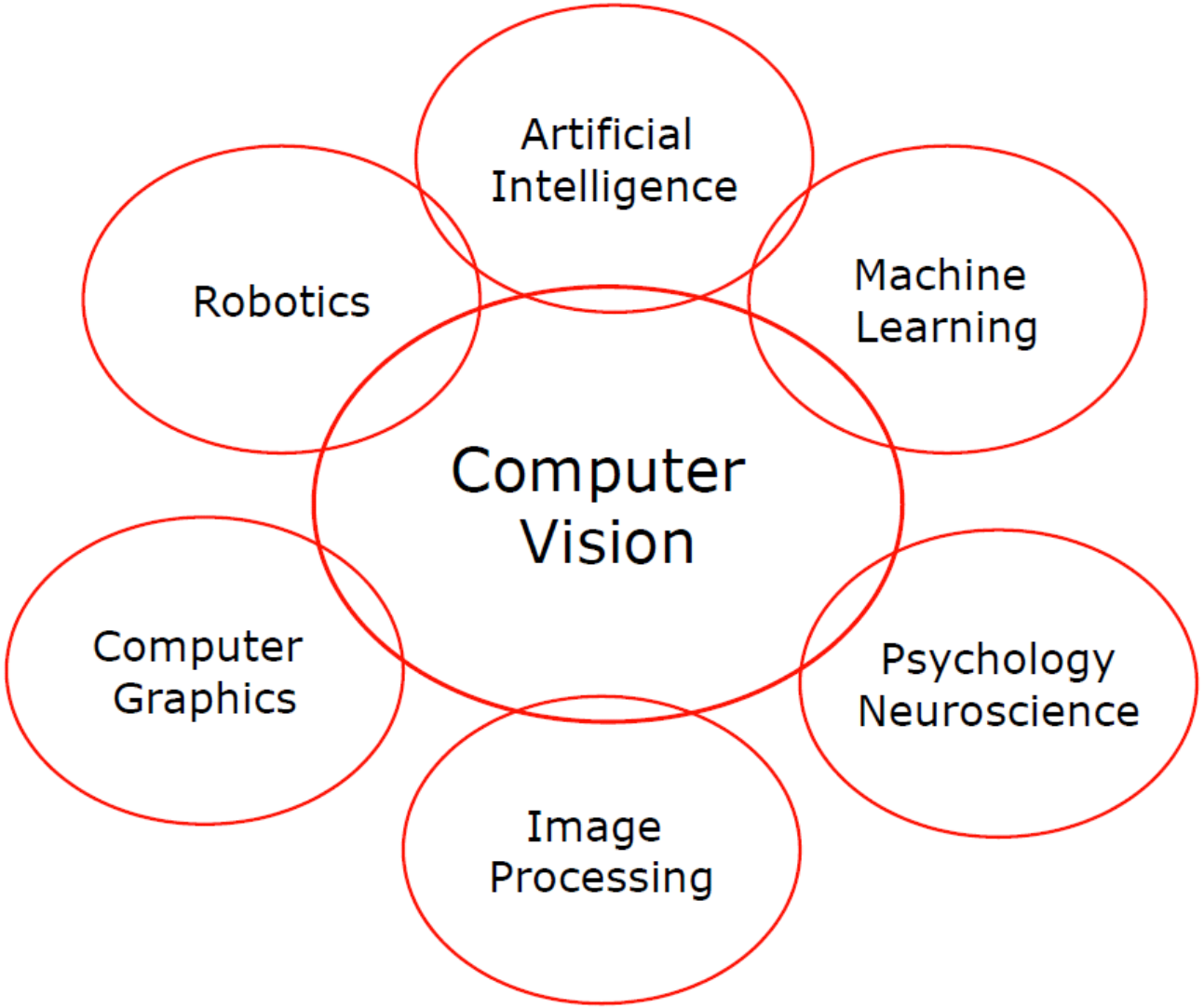
(b) Differentiated picture.





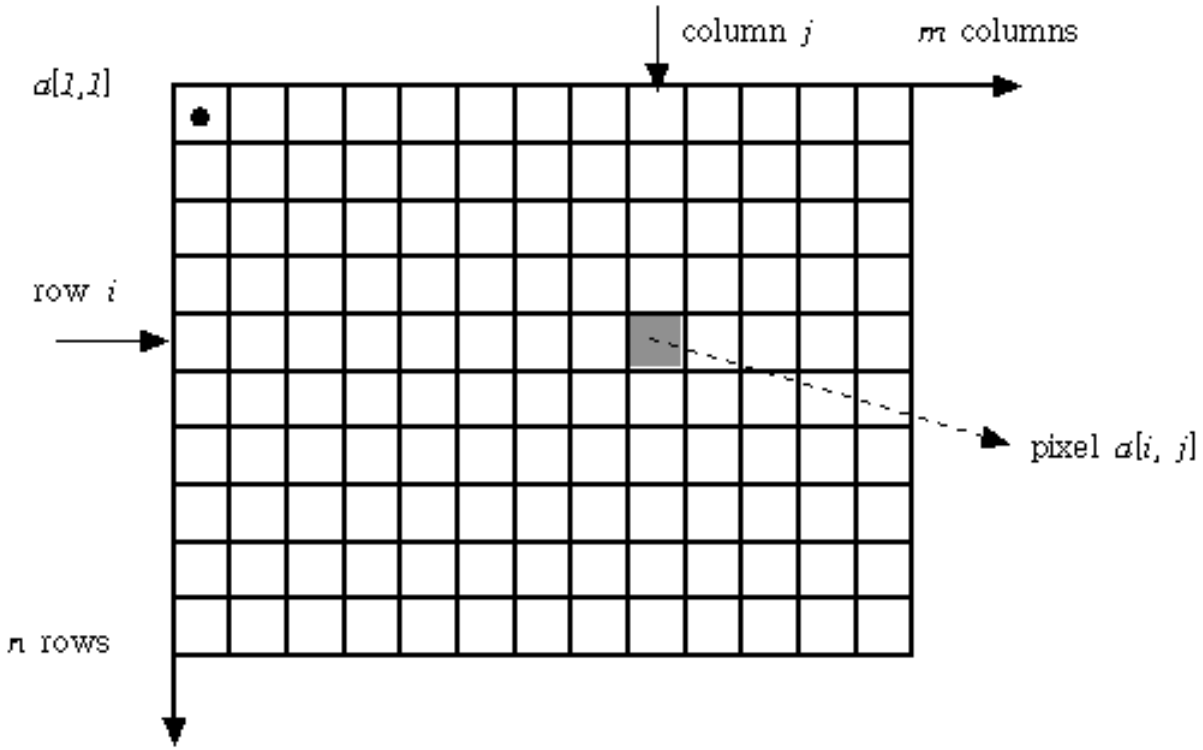
L. G. Roberts, *Machine Perception of Three Dimensional Solids,* Ph.D. thesis, MIT Department of Electrical Engineering, 1963.

Connection to other disciplines



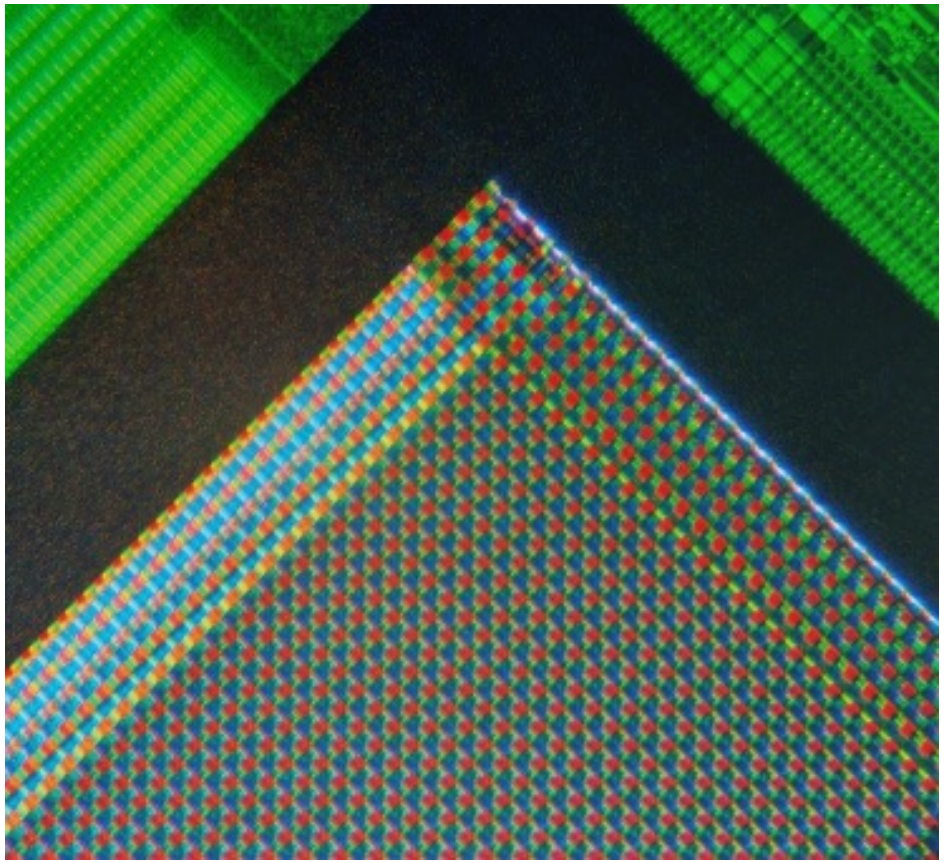
Image

- Image : a two-dimensional array of pixels
- The indices [i, j] of pixels : integer values that specify the rows and columns in pixel values

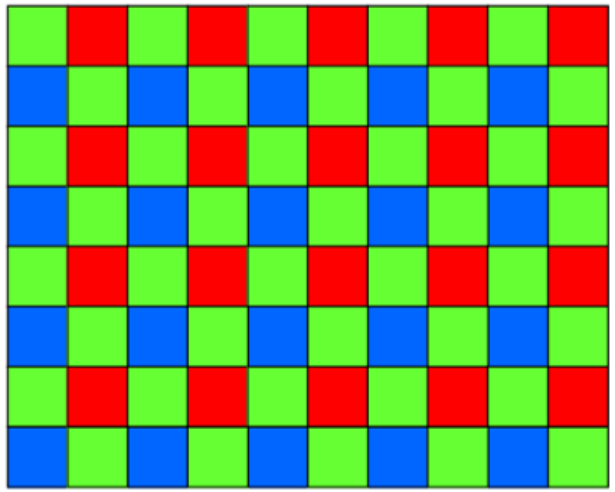


Digital Color Camera

- Bayer Pattern:
  - 50% green, 25% red and 25% blue =>
  - RGBG or GRGB or RGGB.
  - 1 Byte per square
  - 4 squared per 1 pixel
  - More green: eyes are more sensitive to green (nature!)

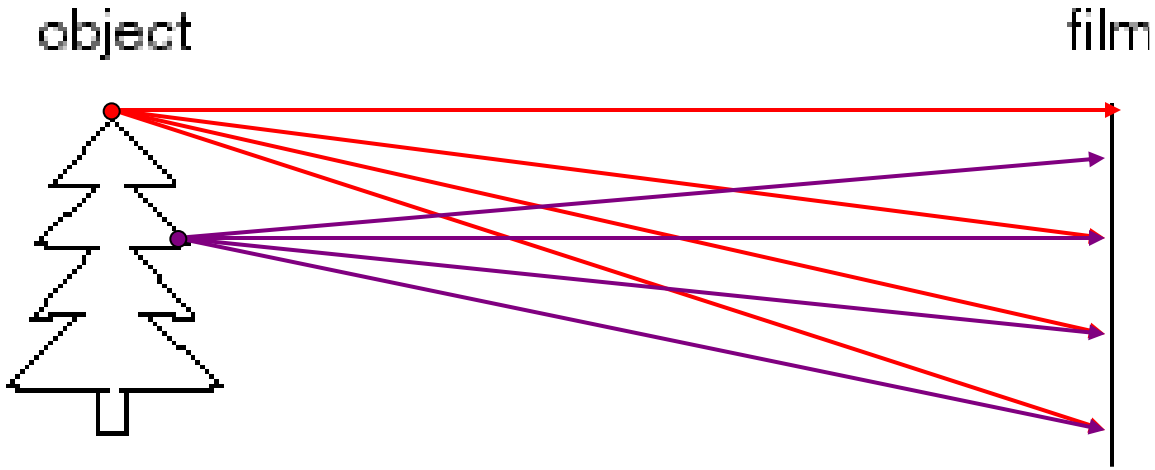


A micrograph of the corner of the photosensor array of a ‘webcam’ digital camera. (Wikimedia)



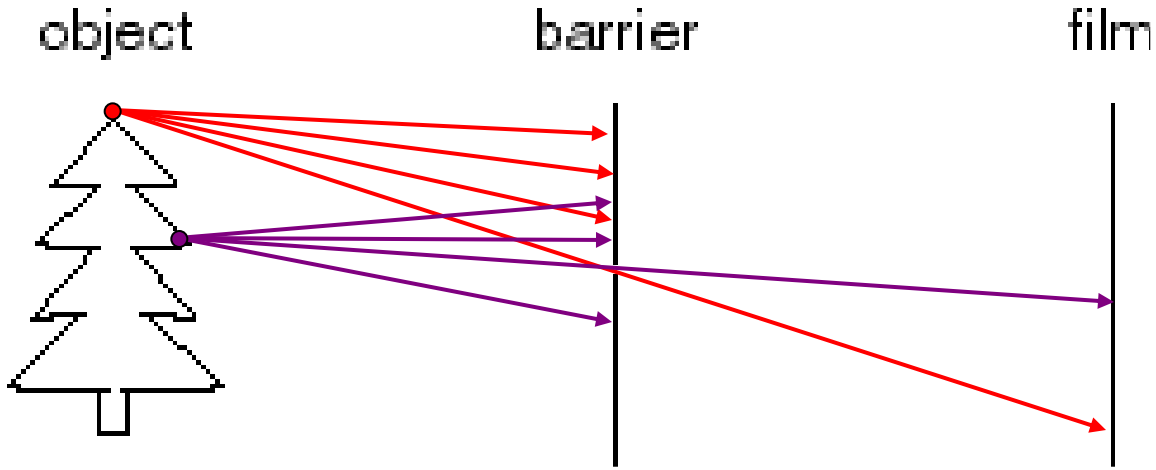
https://en.wikipedia.org/wiki/Bayer_filter

How do we see the world?



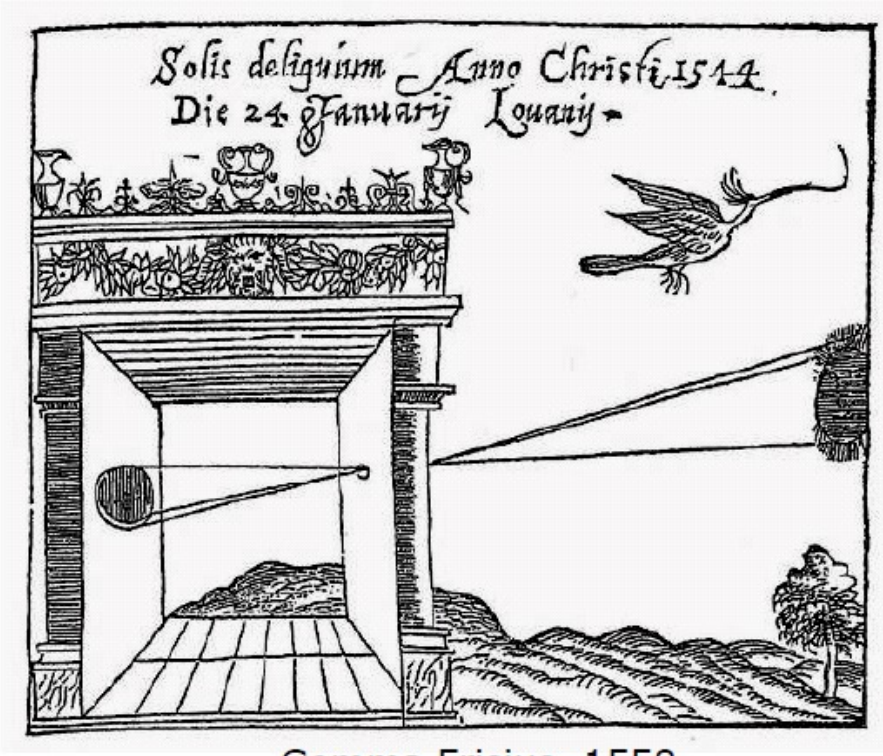
- Let’s design a camera
  - Idea 1: put a piece of film in front of an object
  - Do we get a reasonable image?

Pinhole camera



- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening known as the **aperture**

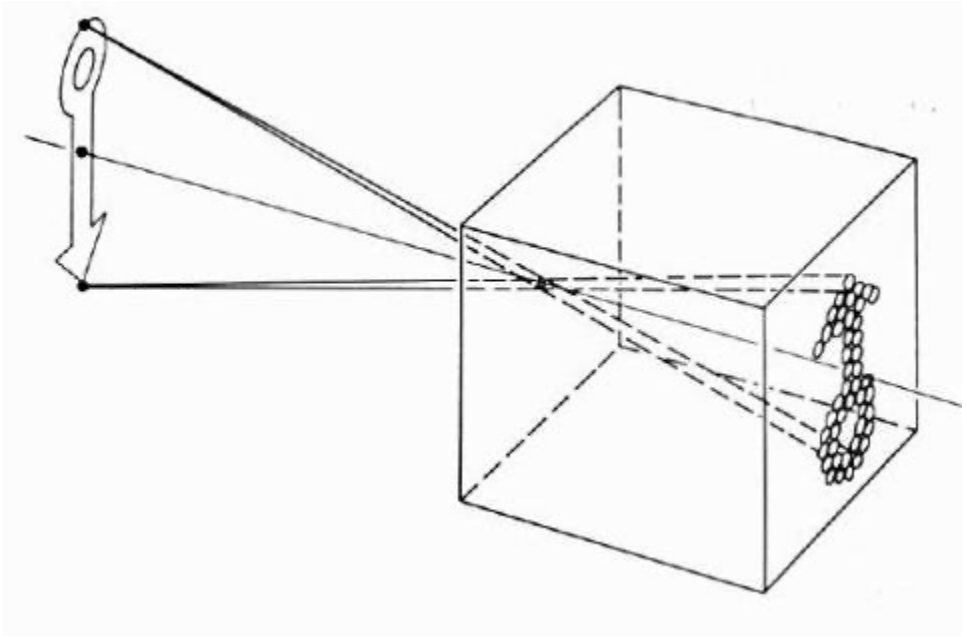
Camera obscura



Gemma Frisius, 1558

- Basic principle known to Mozi (470-390 BC), Aristotle (384-322 BC)
- Drawing aid for artists: described by Leonardo da Vinci (1452-1519)
- Depth of the room (box) is the effective focal length

Pinhole camera model



- Pinhole model:
  - Captures **pencil of rays** – all rays through a single point
  - The point is called **Center of Projection**
  - The image is formed on the **Image Plane**

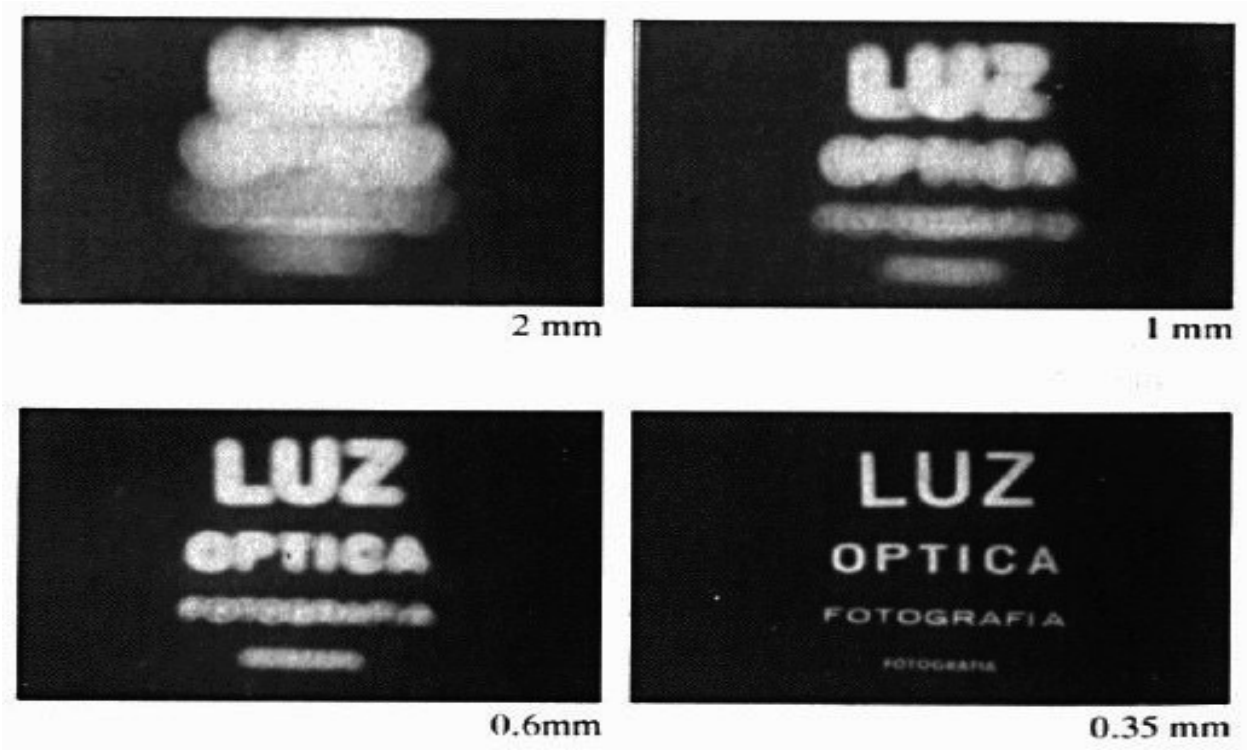
Slide by Steve Seitz

Home-made pinhole camera

Why so blurry?

http://www.debevec.org/Pinhole/

Shrinking the aperture

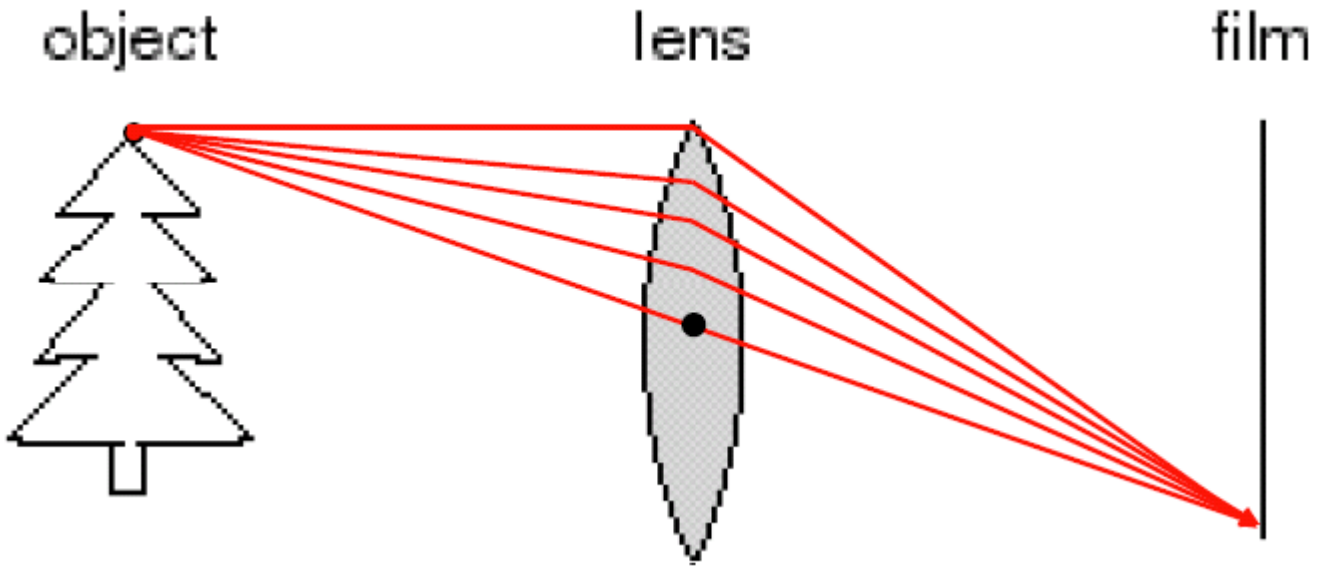


- Why not make the aperture as small as possible?
  - Less light gets through (must increase the exposure)
  - Diffraction effects…

Slide by Steve Seitz

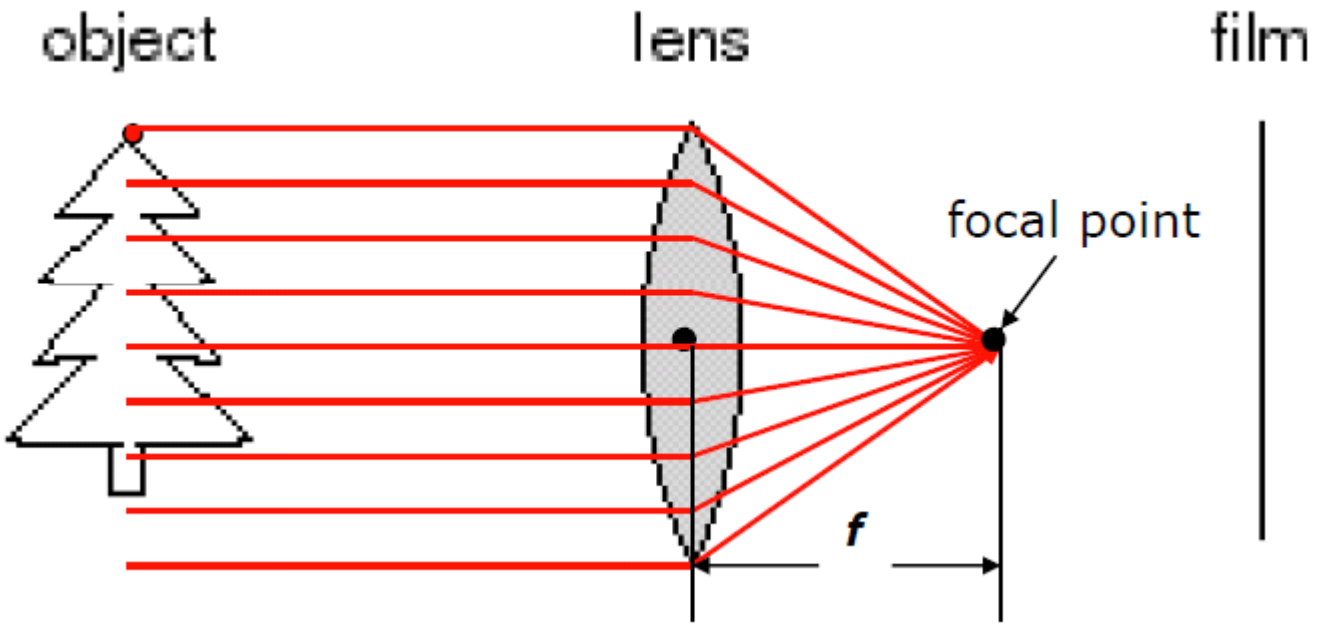
Shrinking the aperture

Solution: adding a lens



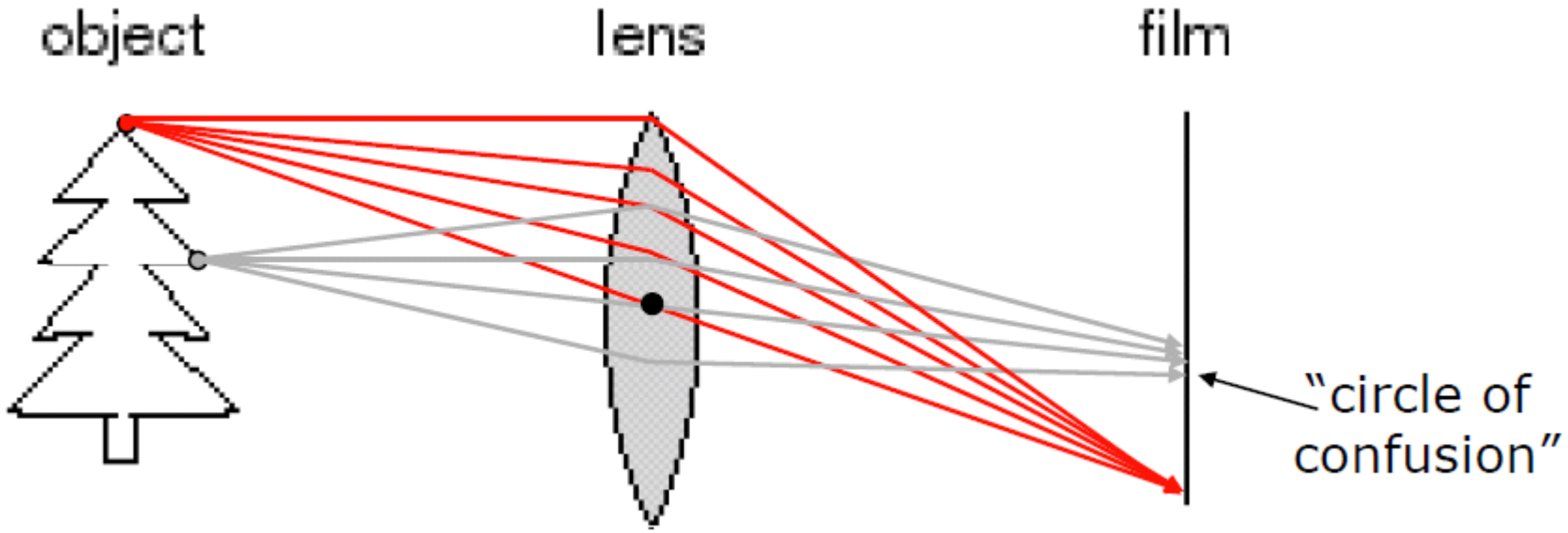
- A lens focuses light onto the film
  - Rays passing through the center are not deviated

Solution: adding a lens



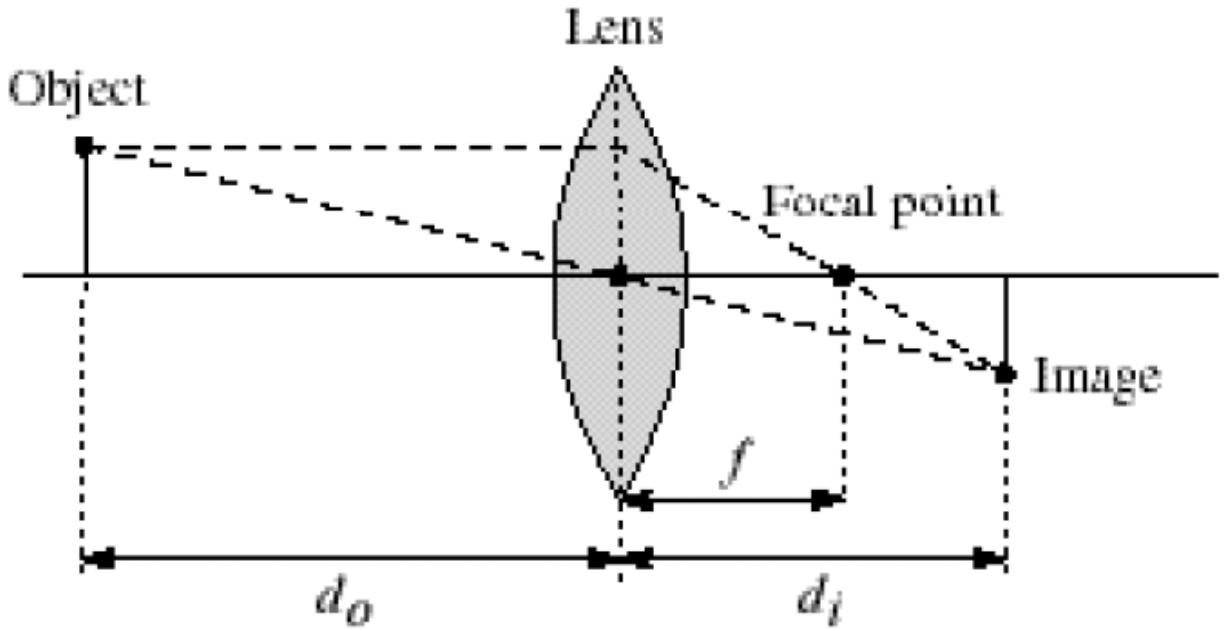
- A lens focuses light onto the film
  - Rays passing through the center are not deviated
  - All parallel rays converge to one point on a plane located at the *focal length f*

Solution: adding a lens



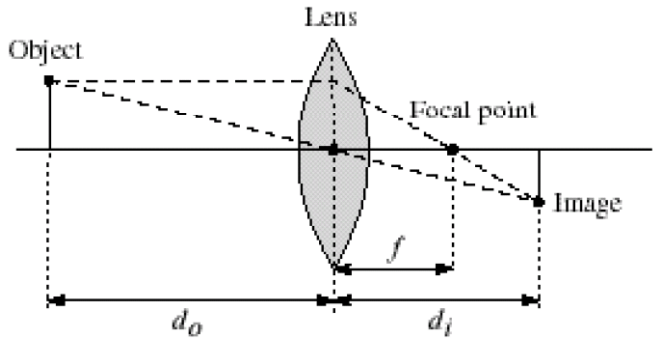
- A lens focuses light onto the film
  - There is a specific distance at which objects are "in focus"
    - other points project to a "circle of confusion" in the image

Thin lenses



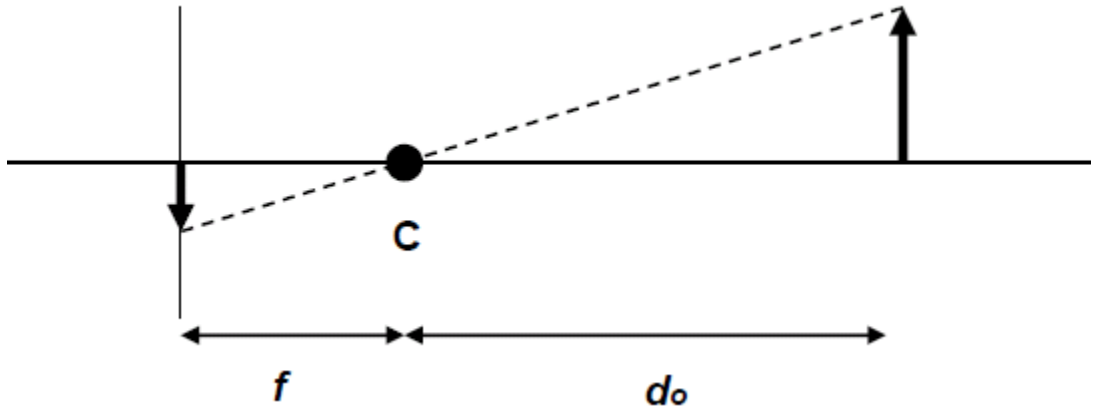
- Thin lens equation: $\frac{1}{d_0}+\frac{1}{d_i}=\frac{1}{f}$
  - Any object point satisfying this equation is in focus
  - This formula can also be used to estimate roughly the distance to the object (“Depth from Focus”)

Pin-hole approximation



$$\frac{1}{d_0}+\frac{1}{d_i}=\frac{1}{f}$$

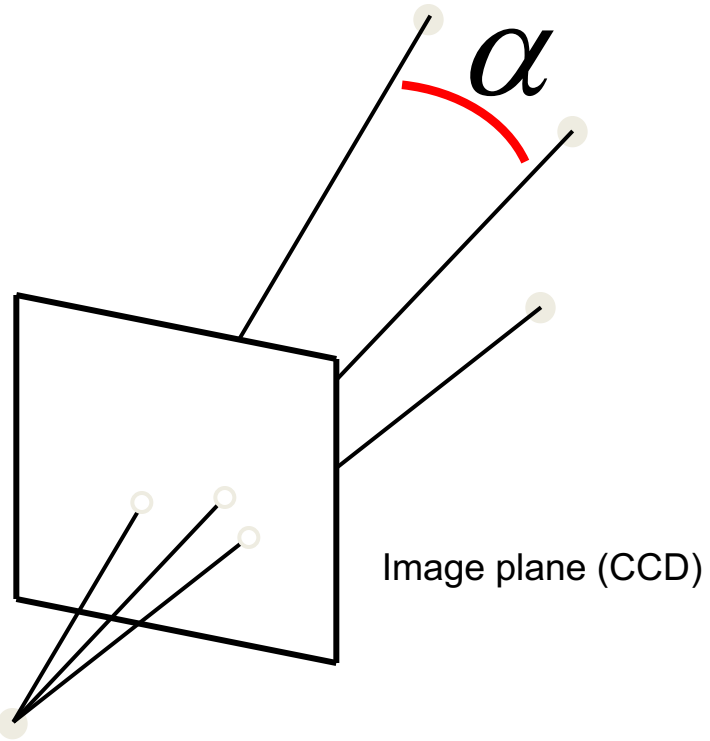
$$d_0 >> d_i \Rightarrow \frac{1}{d_i} \approx \frac{1}{f} \Rightarrow d_i \approx f$$



C = “optical center”

Pin-hole Model

Perspective camera



- For convenience, the image plane is usually represented in front so that the image preserves the same orientation (i.e. not flipped)
- **Notice: a camera does not measure distances but angles! Therefore it is a "bearing sensor"**

Perspective Projection onto the image plane

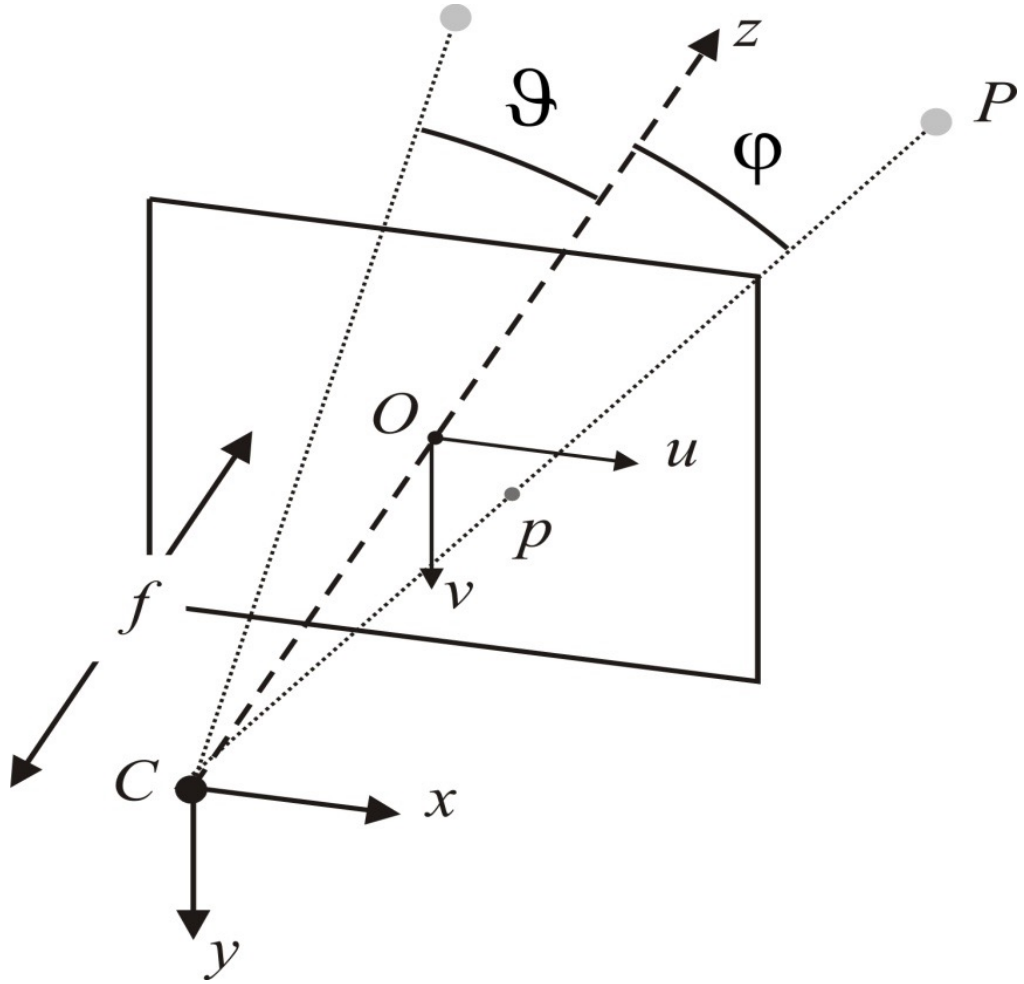
- To project a 3D scene point P = (x,y,z) [meters] onto the camera image plane p=(u,v) [pixels] we need to consider:
  - Pixelization: size of the pixel and position of the CCD with respect to the optical center
  - Rigid body transformation between camera and scene
- u = v = 0: where z-Axis passes trhough center of lens – z-Azis prependicular to lens (coincident with optical axis)

$$u = \frac{f}{z} \cdot x \qquad v = \frac{f}{z} \cdot y$$

Simple case
(without pixelization)

$$u = k_u \frac{f}{z} \cdot x + u_0 \qquad v = k_v \frac{f}{z} \cdot y + v_0$$

With pixelization
u_0, v_0 are the coordinates of the optical center
Ku and Kv are in [pxl/m]



Projection onto the image plane

- Observe that we can also rewrite this

$$u = k_u \frac{f}{z} \cdot x + u_0$$

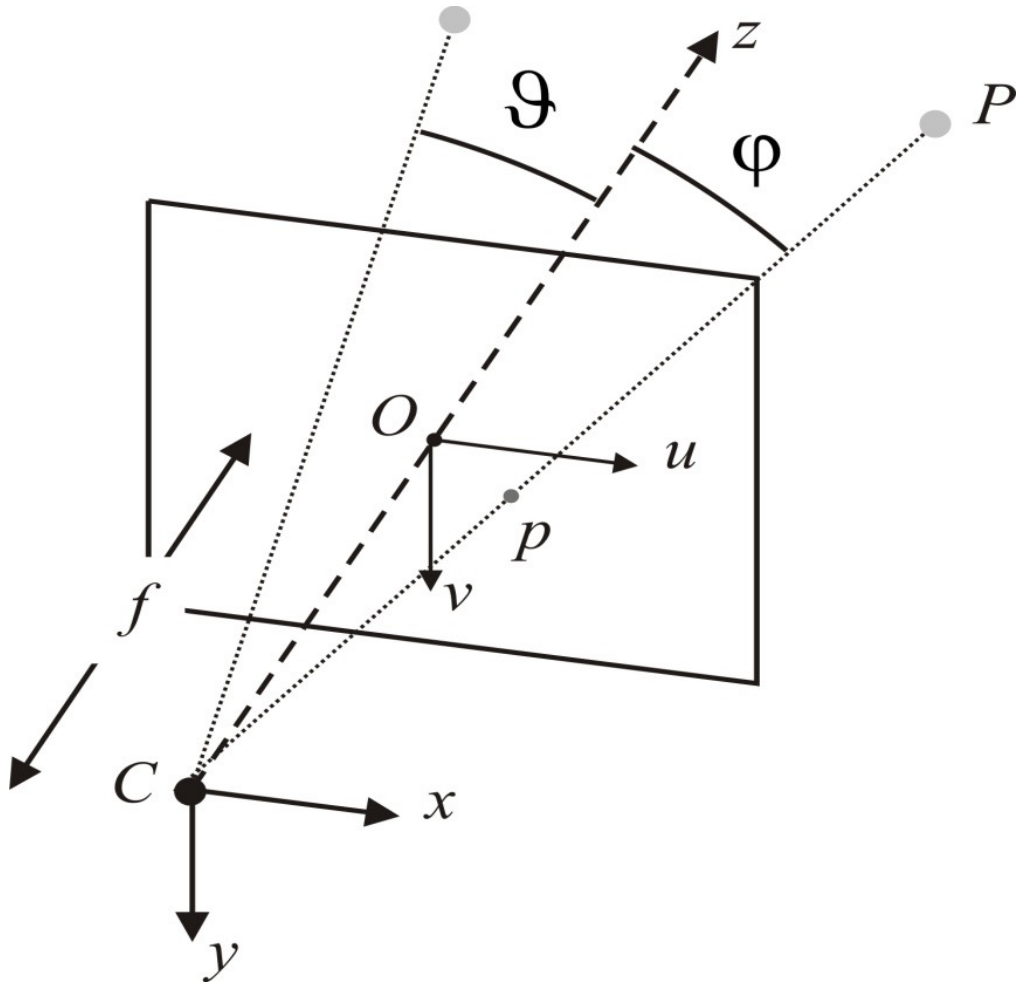
$$v = k_v \frac{f}{z} \cdot y + v_0$$

in matrix form (λ - homogeneous coordinates)

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Or alternatively

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



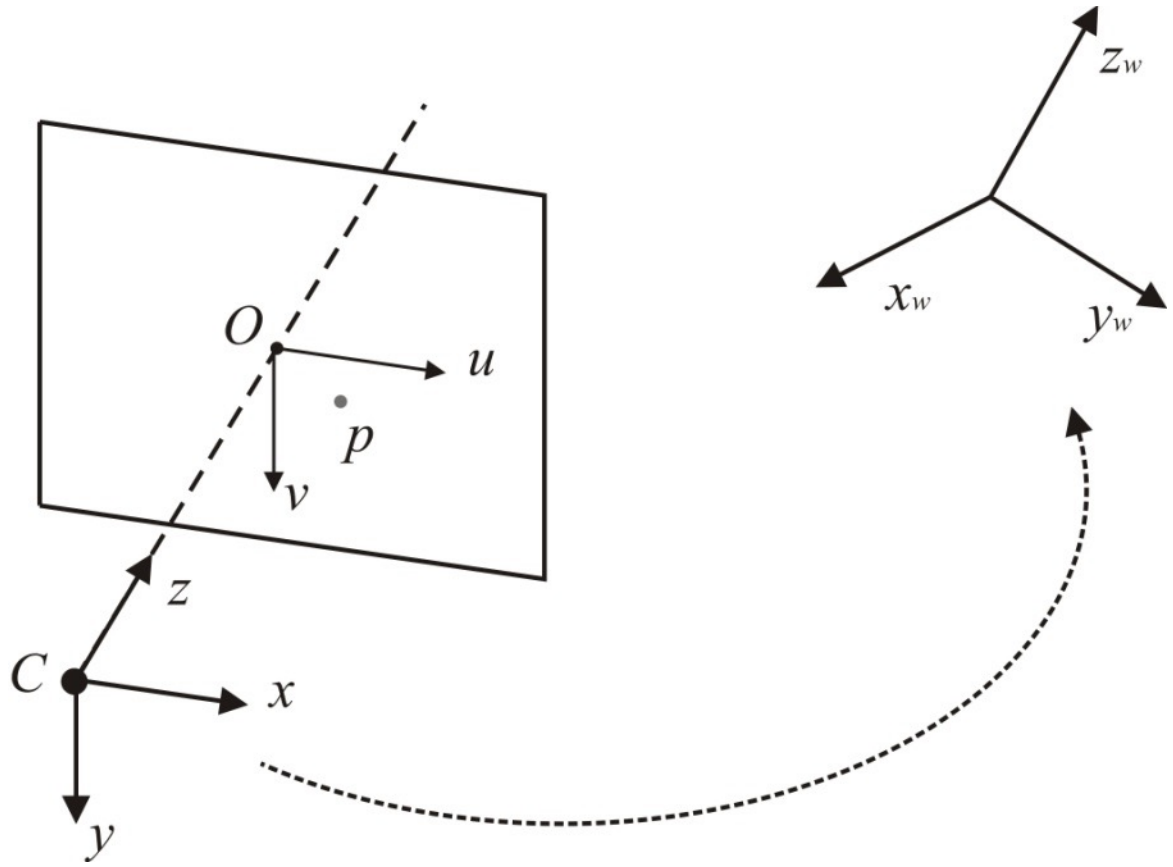
Projection onto the image plane

- Rigid body transformation from the World to the Camera reference frame

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = A\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T$$

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \cdot R\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T$$

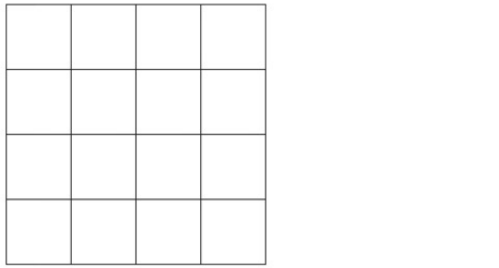


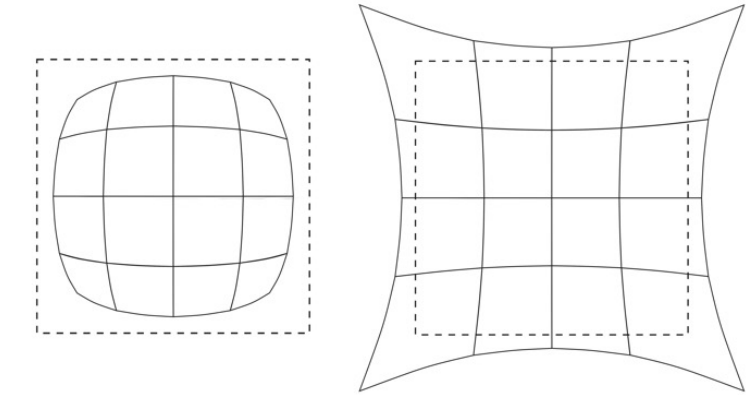
Radial distortion

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

where

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





Barrel distortion Pincushion distortion

Camera Calibration

- How many parameters do we need to model a camera?

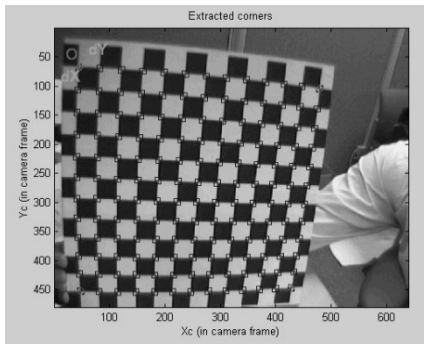
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \qquad \begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 \rho^2) \cdot \begin{bmatrix} u \\ v \end{bmatrix}$$

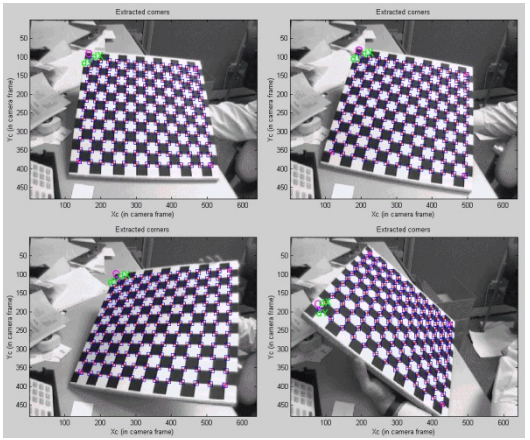
- 5 “intrinsic” parameters: α_u, α_v, u_0, v_0, k_1
- Camera pose?
- 6 “extrinsic” parameters (or 0 if the world and the camera frames coincide)

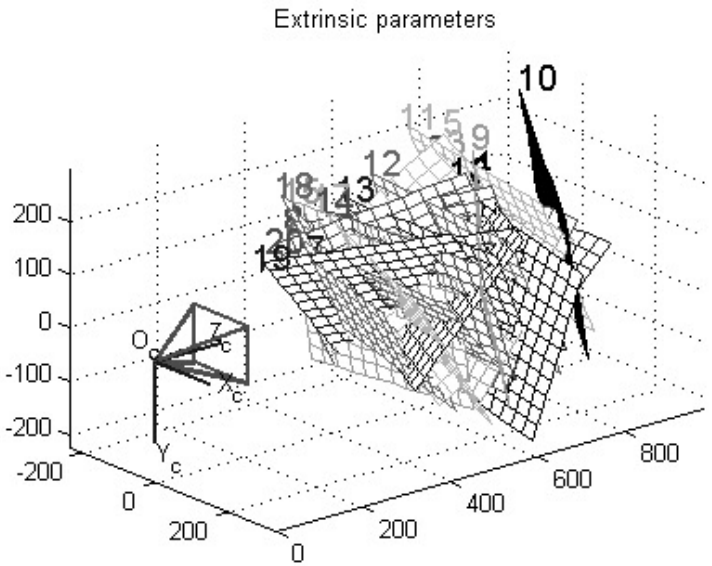
Camera Calibration: how does it work?

- Calibration: measuring accurately **intrinsic** + **extrinsic** parameters of the camera model.
- Parameters: govern mapping from scene points to image points
- Idea: known:
  - pixel coordinates of image points *p*
  - 3D coordinates of the corresponding scene points *P*
  - => compute the unknown parameters A, R, T by solving the perspective projection equation

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T$$







STEREO VISION

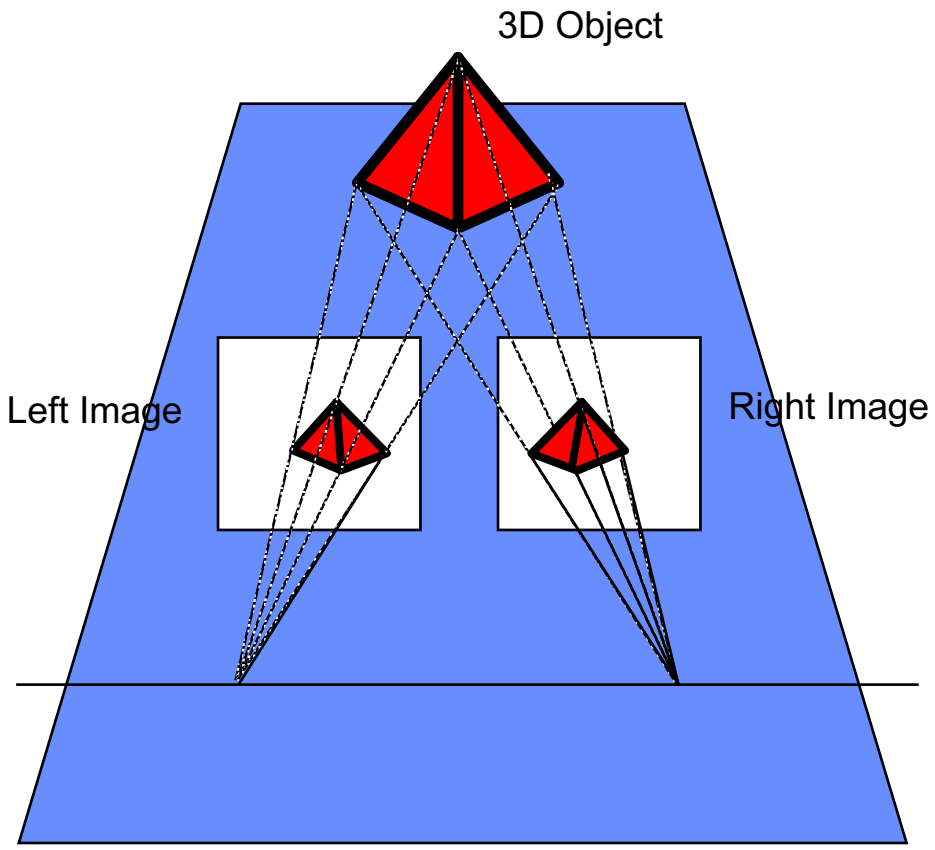
How do we measure distances with cameras?

- Structure from stereo (Stereo-vision):
  - use two cameras with known relative position and orientation

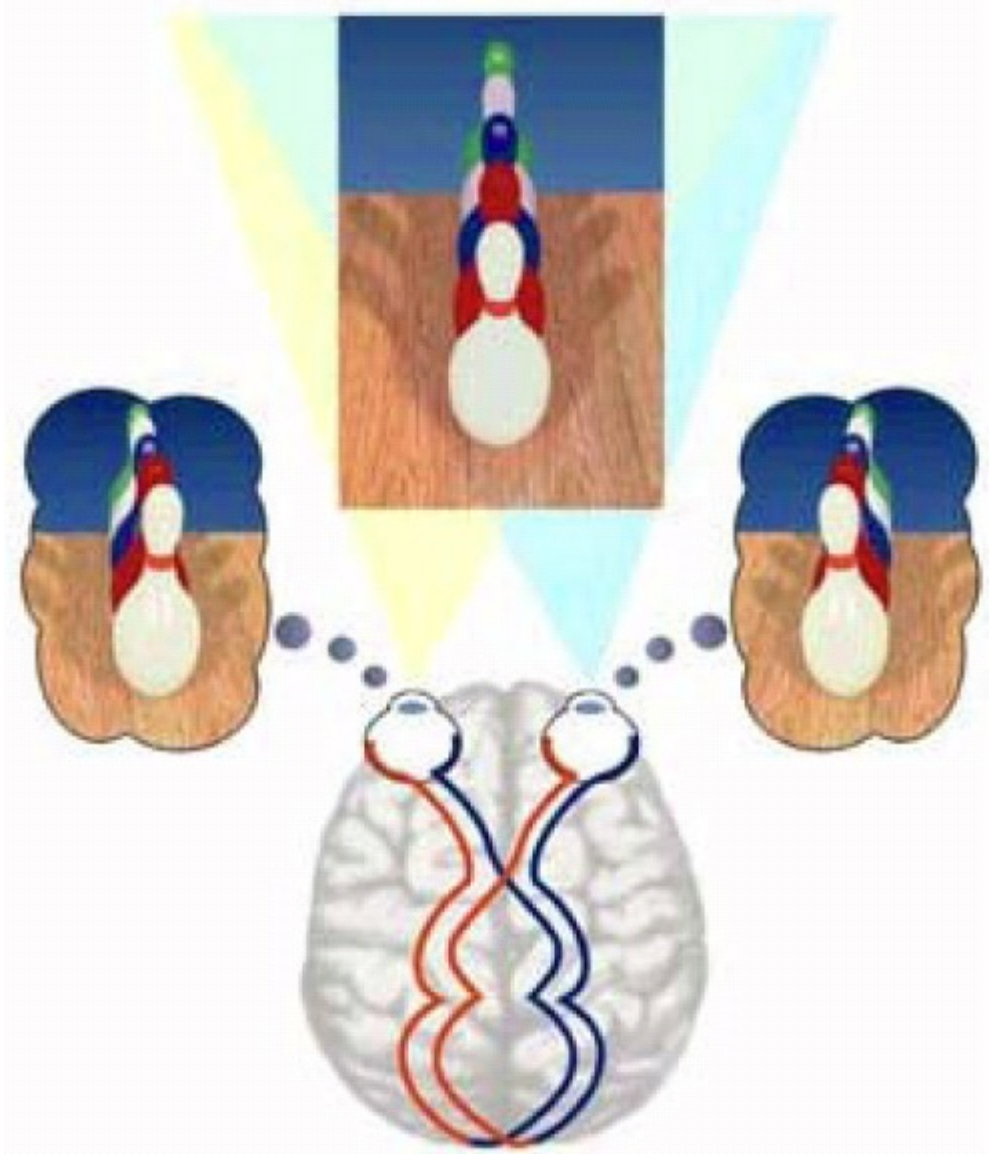
- Structure from motion:
  - use a single moving camera: both 3D structure and camera motion can be estimated up to a scale

Stereo Vision

- Allows to reconstruct a 3D object from two images taken at different locations

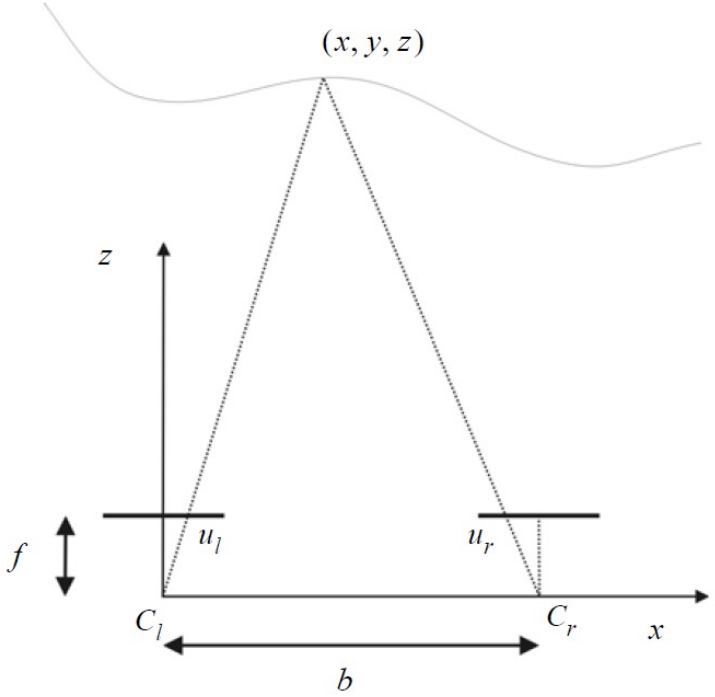


Disparity in the human retina



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



$$\frac{f}{z} = \frac{u_l}{x}, \qquad \frac{f}{z} = \frac{u_r}{b-x}$$

$$z = b\frac{f}{u_l - u_r}$$

Distance

- ***b*** = baseline, distance between the optical centers of the two cameras
- ***f*** = focal length
- u_l-u_r = disparity

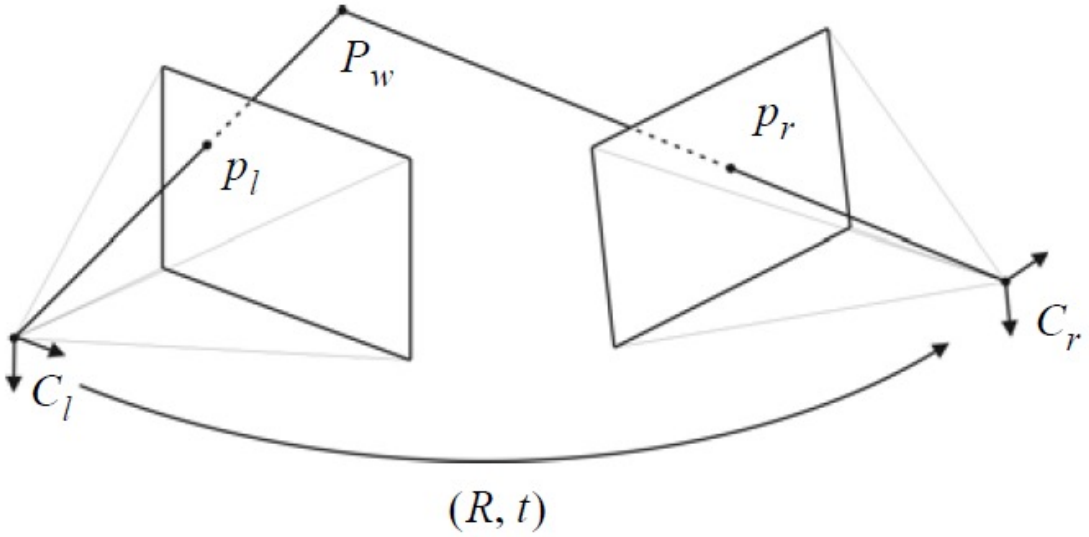
Stereo Vision: how to improve accuracy?

$$z = b\frac{f}{u_l - u_r}$$

1. Distance is inversely proportional to disparity ($\mathbf{u_l}$-$\mathbf{u_r}$)
   - closer objects can be measured more accurately
2. Disparity is proportional to ***b***
   - For a given disparity error, the accuracy of the depth estimate increases with increasing baseline ***b***.
   - However, as ***b*** is increased, some objects may appear in one camera, but not in the other.
3. Increasing image resolution improves accuracy

Stereo Vision – the general case

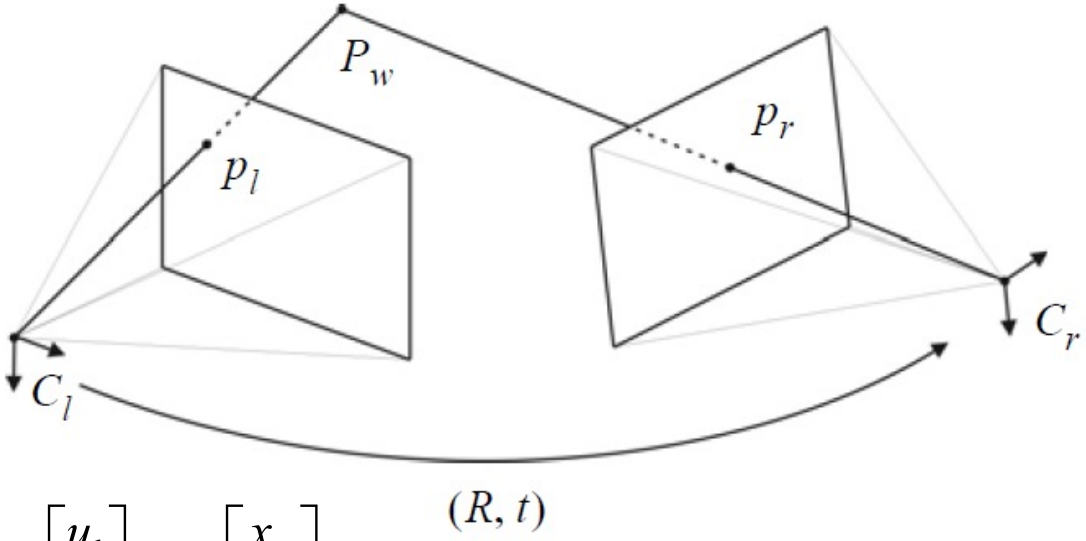
- Two identical cameras do not exist in nature!
- Aligning both cameras on a horizontal axis is very hard, also with the most expensive stereo cameras!



- In order to be able to use a stereo camera, we need first to estimate the relative pose between the cameras, that is, **Rotation** and **Translation**
- However, as the two cameras are not identical, we need to estimate:
  focal length, **image center**, **radial distortion**

Stereo Vision – the general case

- To estimate the 3D position we just construct the system of equations of the left and right camera



$$\lambda_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = A_l \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

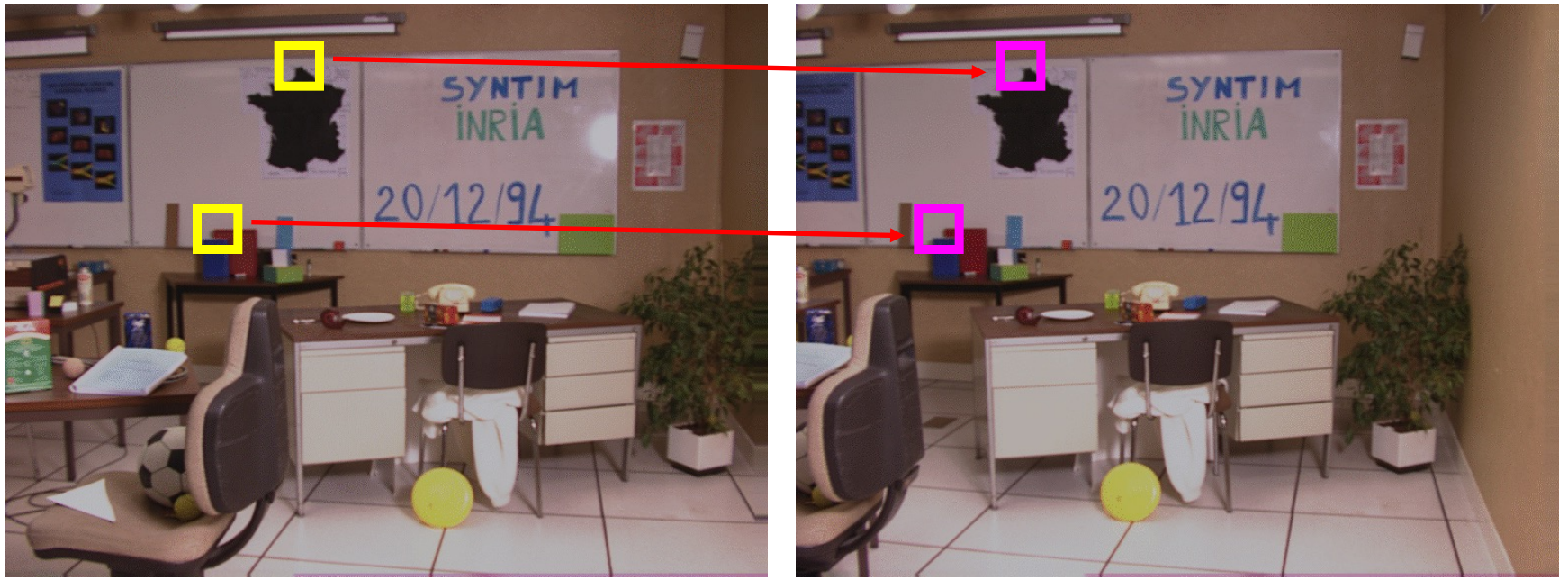
Left camera; world frame coincides with the left camera frame

$$\lambda_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = A_r \cdot R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T$$

Right camera

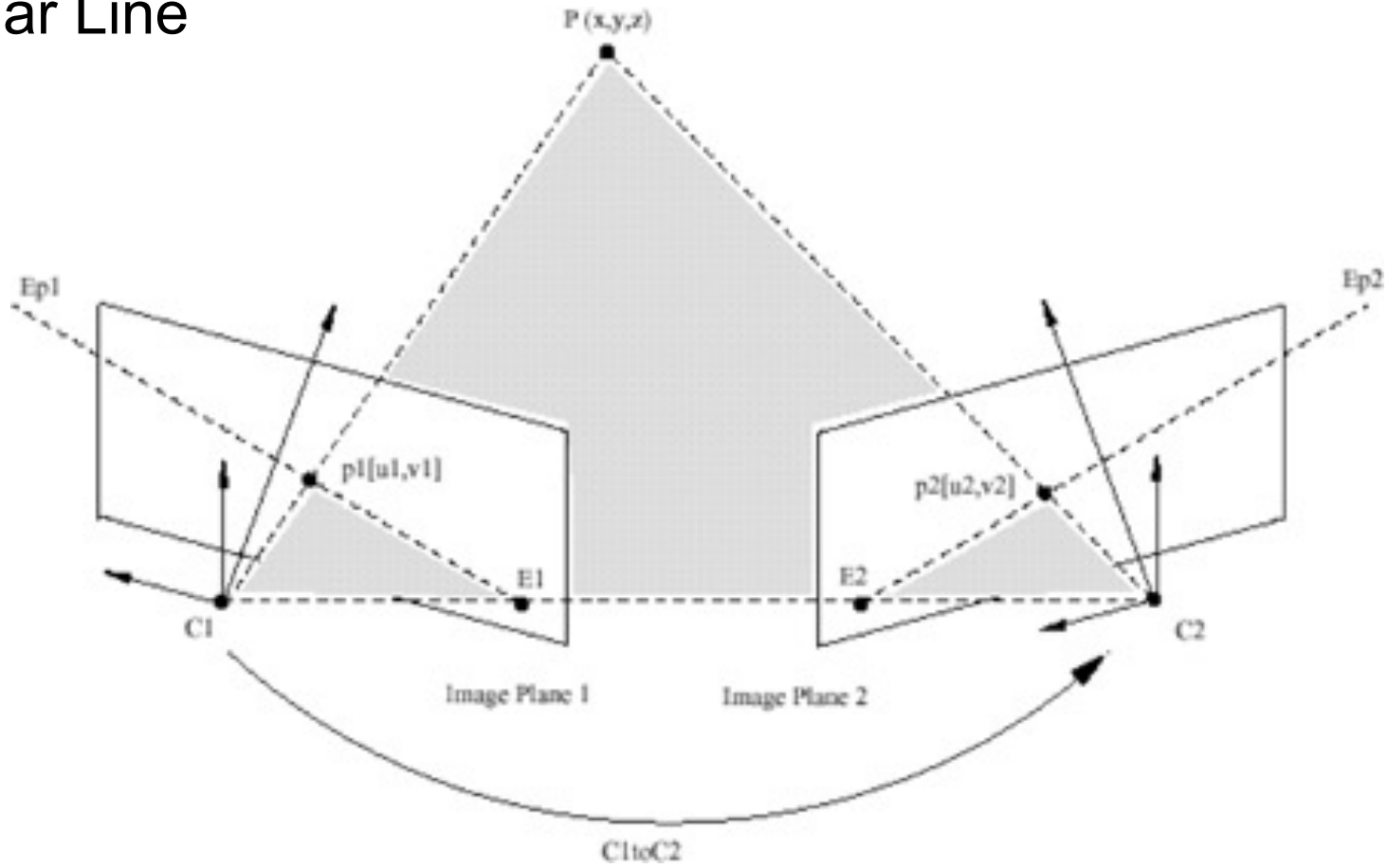
Stereo Vision: Correspondence Problem

- Matching between points in the two images which are projection of the same 3D real point
- Correspondence search could be done by comparing the observed points with all other points in the other image. Typical similarity measures are the Correlation and image Difference.
- This image search can be computationally very expensive! Is there a way to make the correspondence search 1 dimensional?



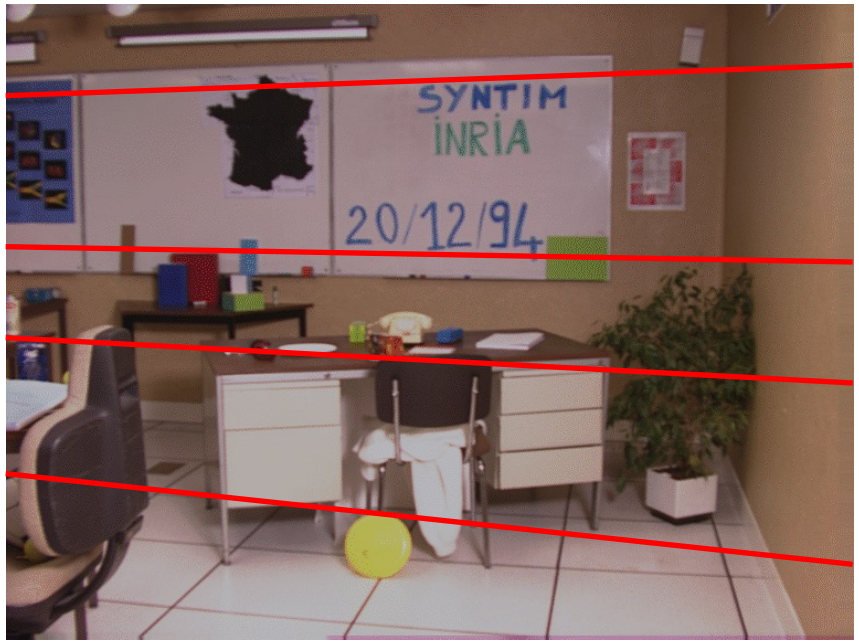
Correspondence Problem: Epipolar Constraint

- The correspondent of a point in an image must lie on a line in the other image, called Epipolar Line



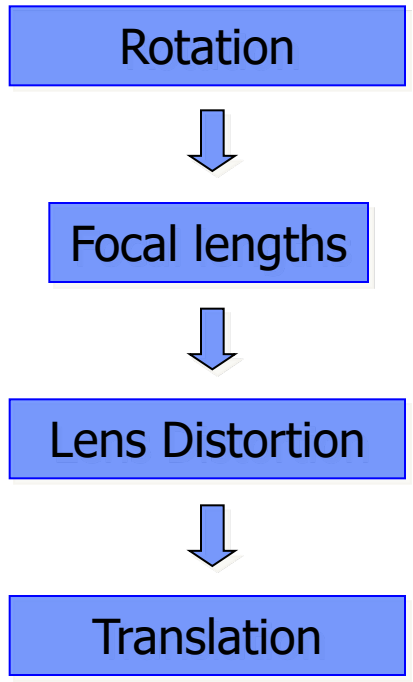
Correspondence Problem: Epipolar Constraint

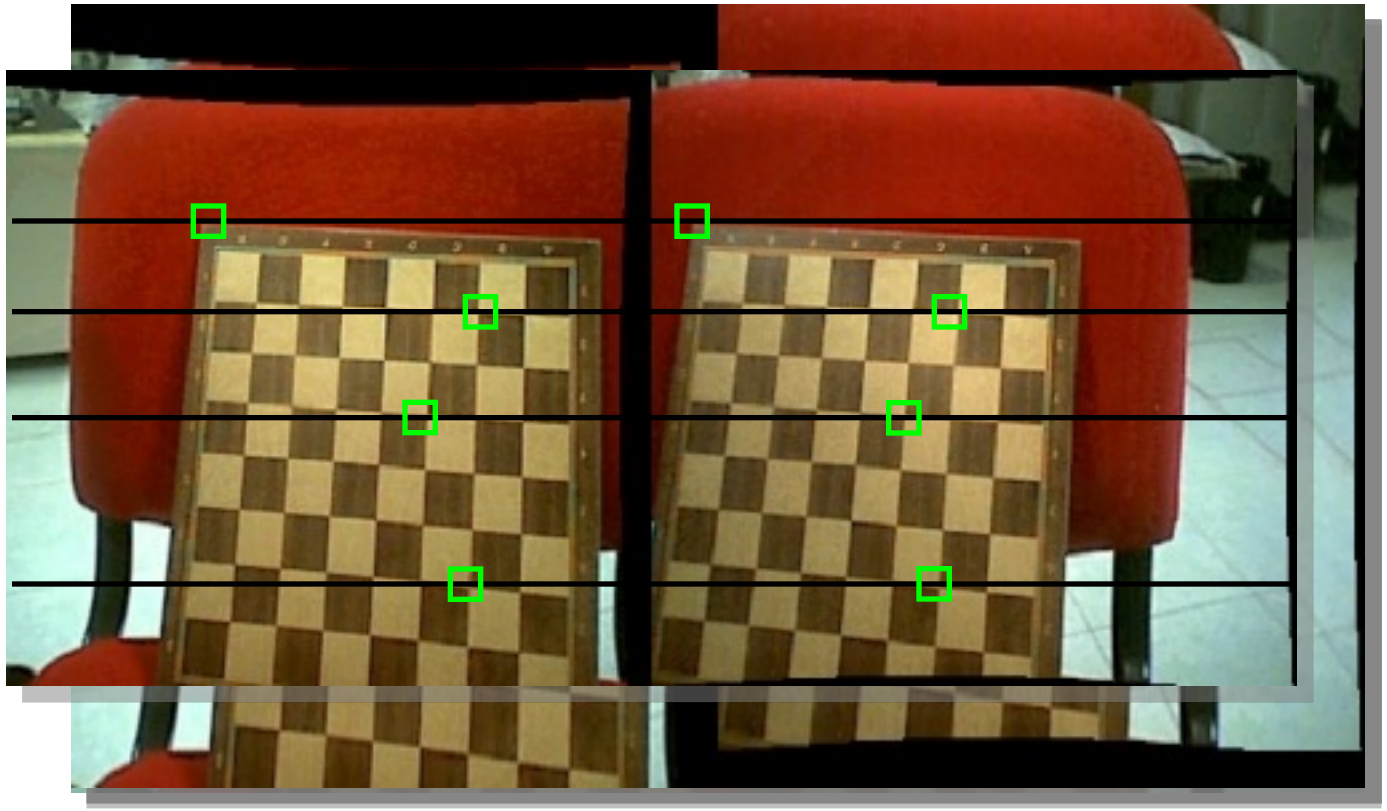
- Thanks to the epipolar constraint, conjugate points can be searched along epipolar lines: this reduces the computational cost to 1 dimension!



Epipolar Rectification

- Determines a transformation of each image plane so that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes (usually the horizontal one)





Stereo Vision Output 1 – Disparity map

- Find the correspondent points of all image pixels of the original images
- For each pair of conjugate points compute the disparity ***d = v-v'***
- ***d(x,y)*** is called Disparity map.

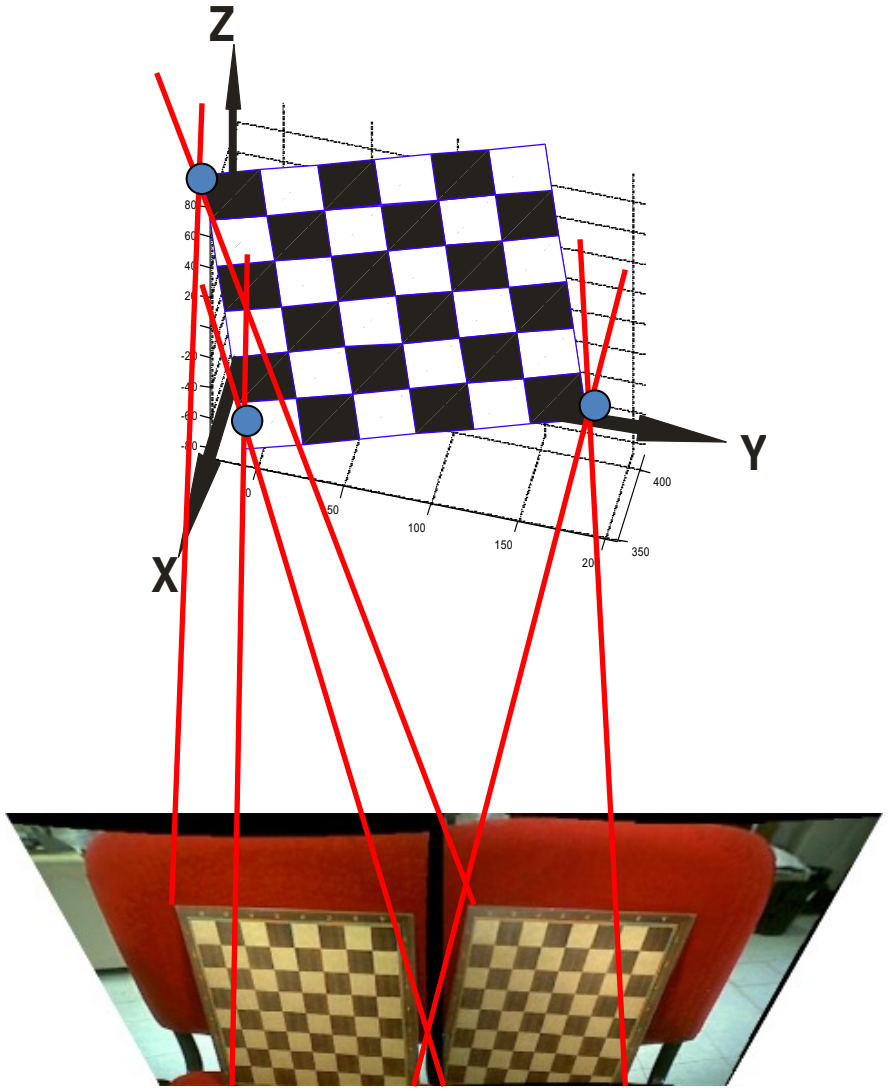
Left image

Right image

Disparity map

- Disparity maps are usually visualized as grey-scale images. Objects that are closer to the camera appear lighter, those who are further appear darker.

Stereo Vision Output 2 - 3D Reconstruction via triangulation



Stereo Camera Calibration

Estimates the parameters that manage the 3D – 2D transformation

$$\begin{cases} \begin{bmatrix} u_L \\ v_L \end{bmatrix} = f \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \\ \begin{bmatrix} u_R \\ v_R \end{bmatrix} = f \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \end{cases}$$

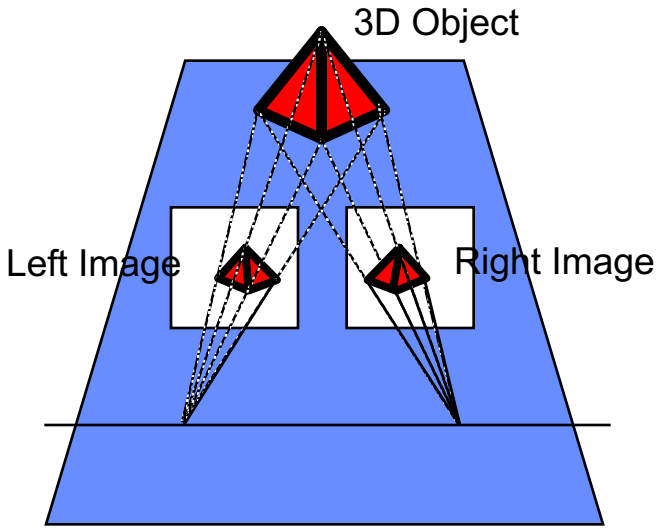
Stereo Camera Calibration

PLUS

5 parameters for each camera in order to compensate for lens distortion (radial & tangential distortion)

$$\begin{cases} \begin{bmatrix} u_{d,L} \\ v_{d,L} \end{bmatrix} = (1 + kc_{1,L}\rho^2 + kc_{2,L}\rho^4 + kc_{5,L}\rho^6) \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2kc_{3,L} \cdot u \cdot v + kc_{4,L}(\rho^2 + 2u^2) \\ kc_{3,L}(\rho^2 + 2v^2) + 2kc_{4,L} \cdot u \cdot v \end{bmatrix} \\ \begin{bmatrix} u_{d,R} \\ v_{d,R} \end{bmatrix} = (1 + kc_{1,R}\rho^2 + kc_{2,R}\rho^4 + kc_{5,R}\rho^6) \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2kc_{3,R} \cdot u \cdot v + kc_{4,R}(\rho^2 + 2u^2) \\ kc_{3,R}(\rho^2 + 2v^2) + 2kc_{4,R} \cdot u \cdot v \end{bmatrix} \end{cases}$$

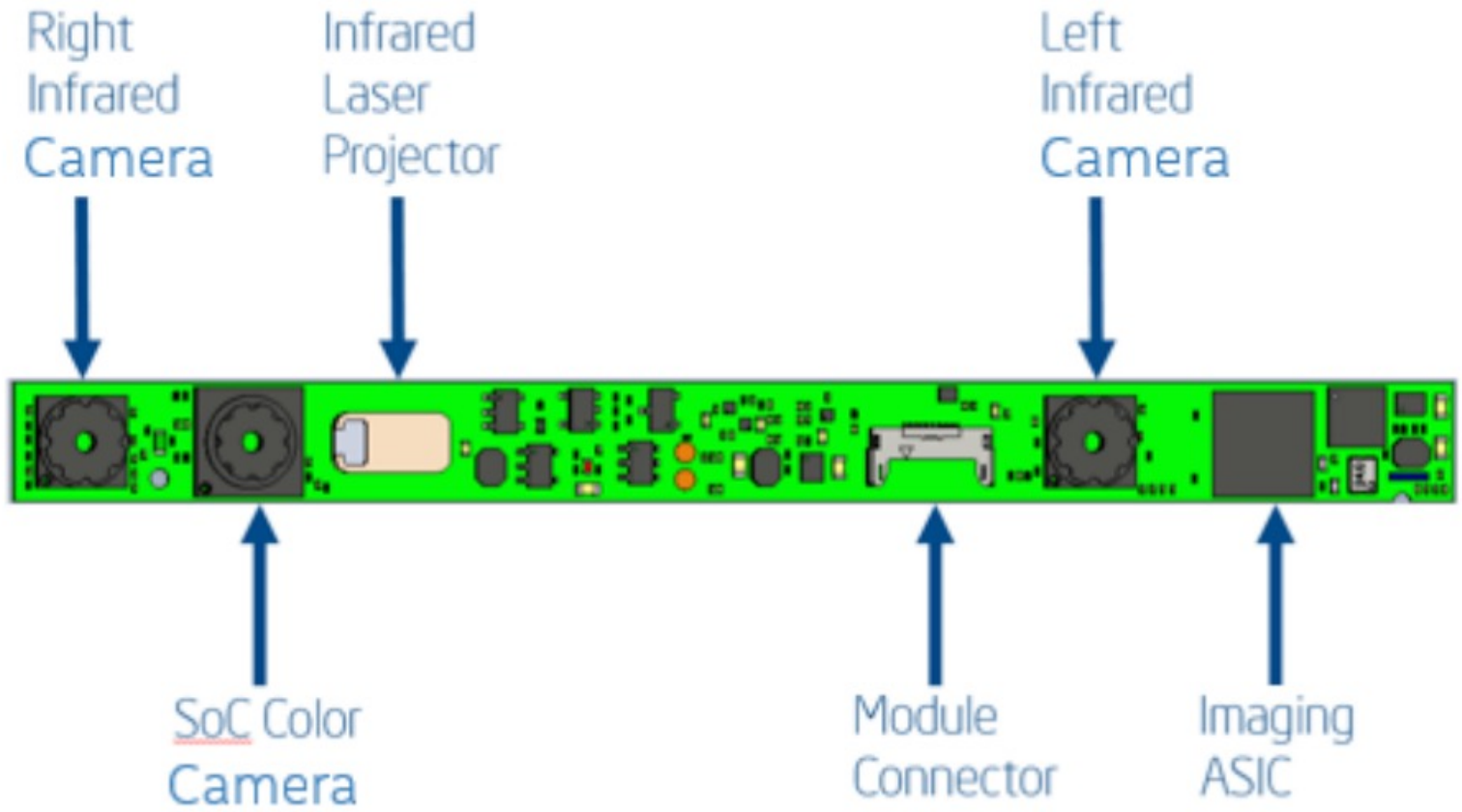
Stereo Vision - summary

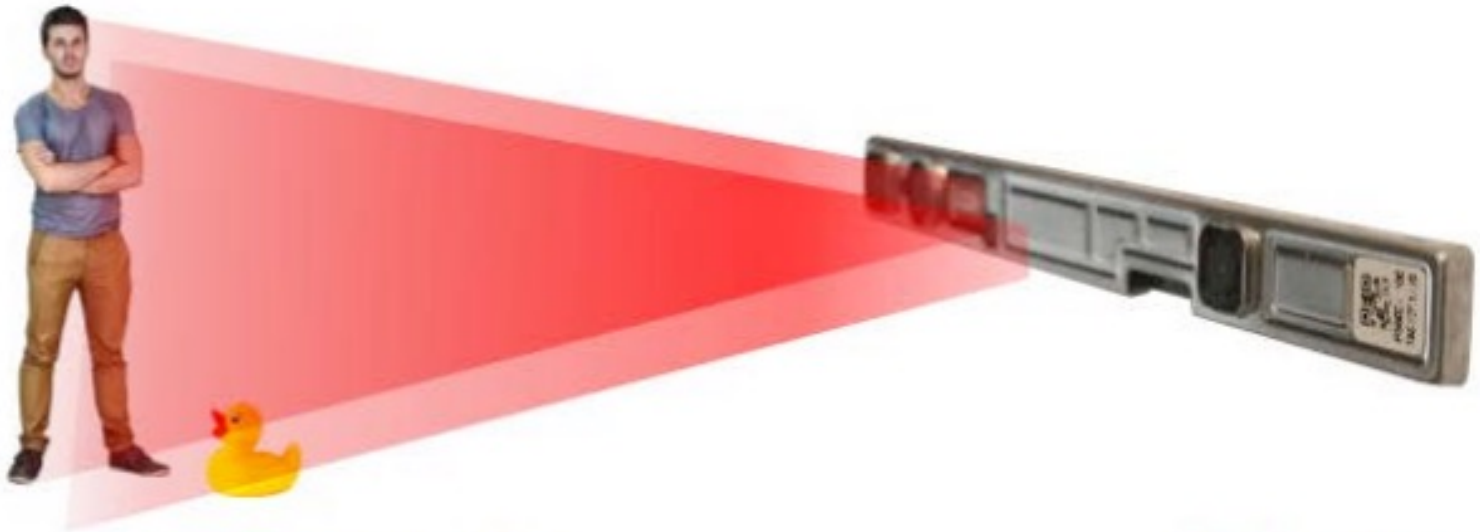


1. Stereo camera calibration -> compute camera relative pose
2. Epipolar rectification -> align images
3. Search correspondences
4. Output: compute stereo triangulation or disparity map
5. Consider baseline and image resolution to compute accuracy!

Device example:

- 640x480 stereo infrared cameras
- 1920x1080 HD RGB camera
- Infrared Laser Projector
- Up to 60Hz





1 CAPTURE

Project invisible light pattern on low texture surfaces like plain walls

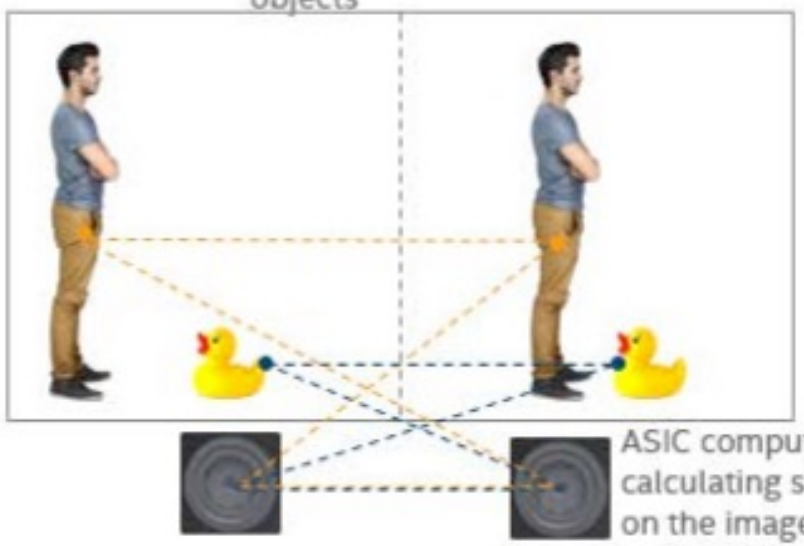
Left

Right

Each camera sees a slightly different viewpoint

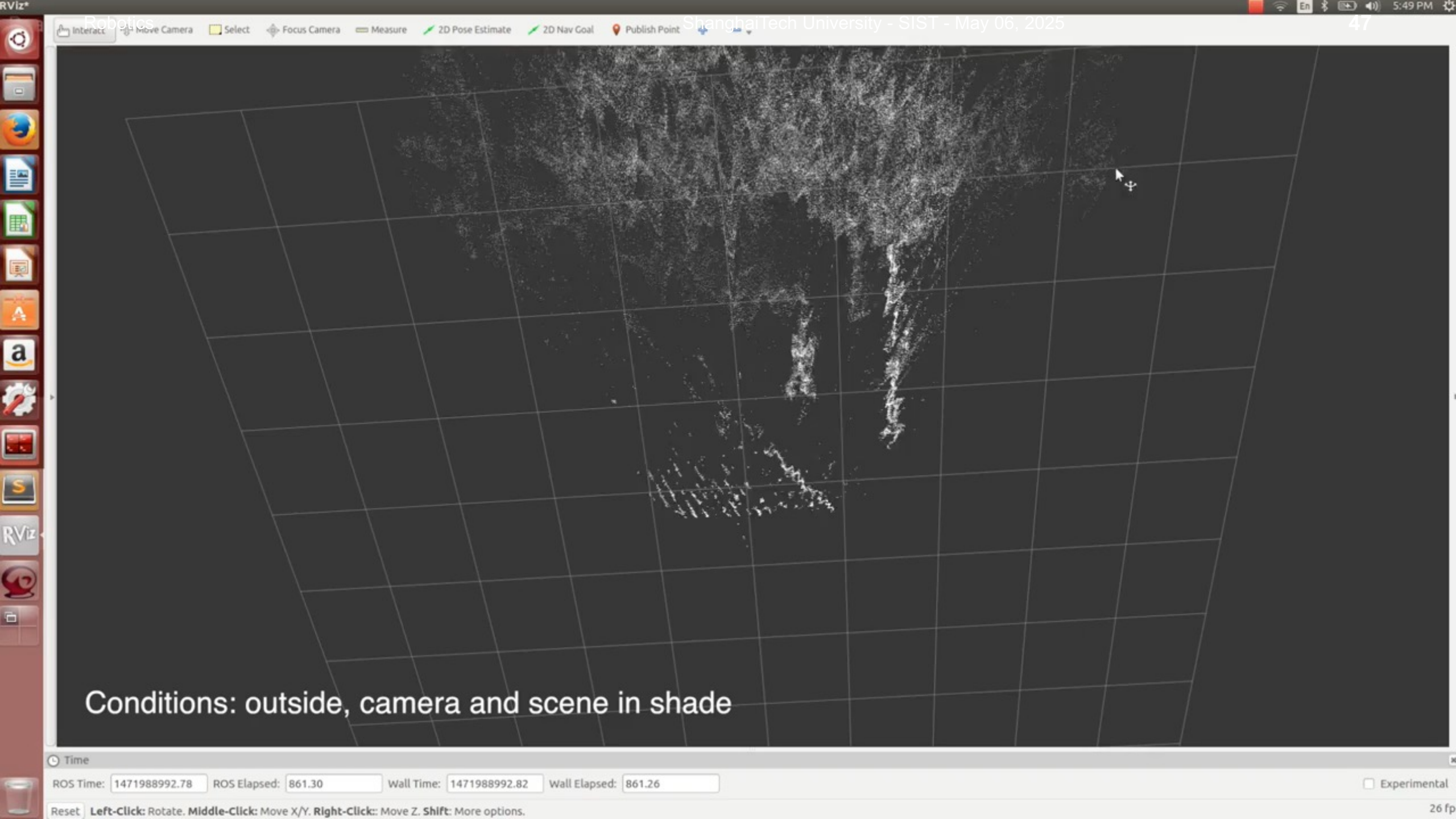
2 SEARCH

Closer objects shift more than further away objects



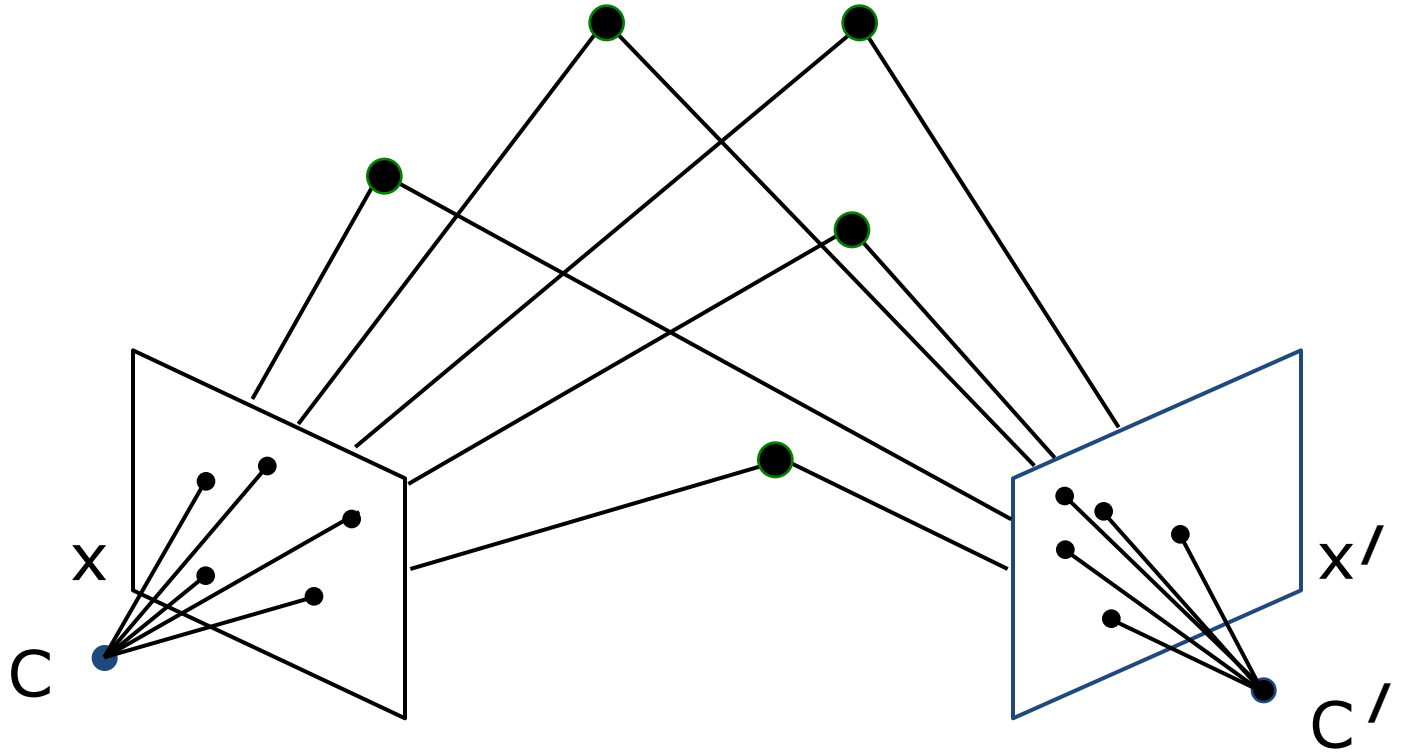
ASIC computes depth by calculating shift of every pixel on the image between the left and right images

3 DEPTH

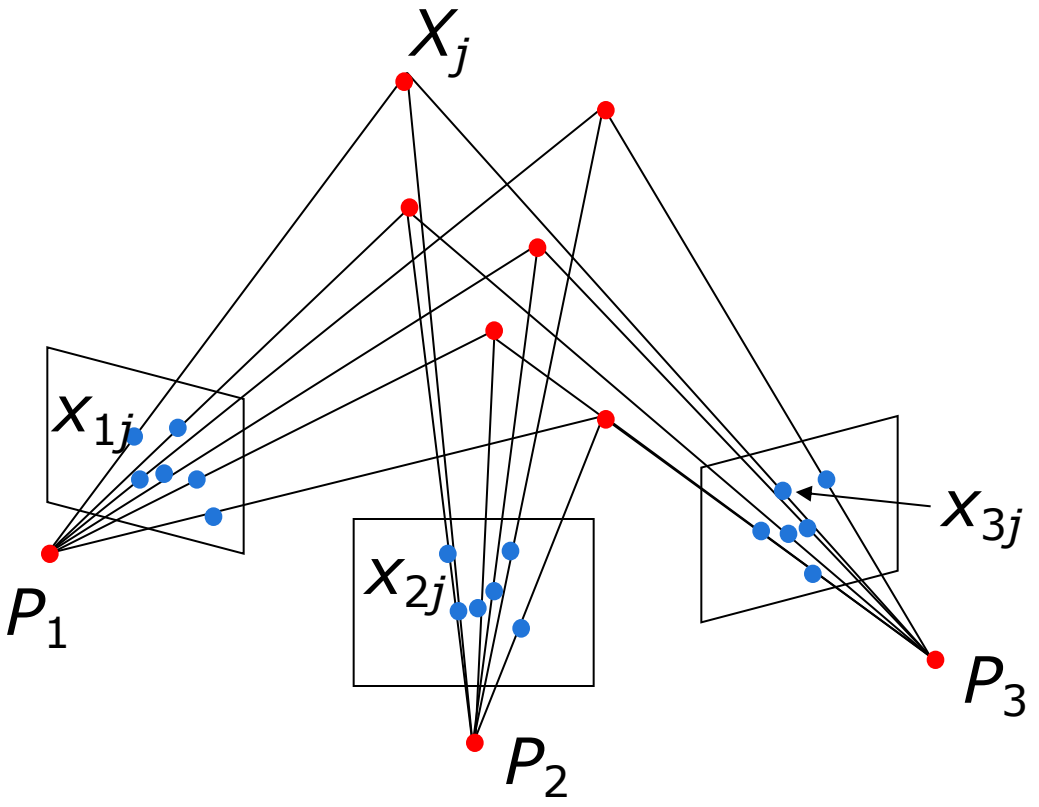

RViz*
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
5:49 PM
Conditions: outside, camera and scene in shade
Time
ROS Time: 1471988992.78
ROS Elapsed: 861.30
Wall Time: 1471988992.82
Wall Elapsed: 861.26
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.
26 fp

Structure from motion

- Given image point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$,determine R and T
- Rotate and translate camera until stars of rays intersect
- At least 5 point correspondences are needed



Multiple-view structure from motion

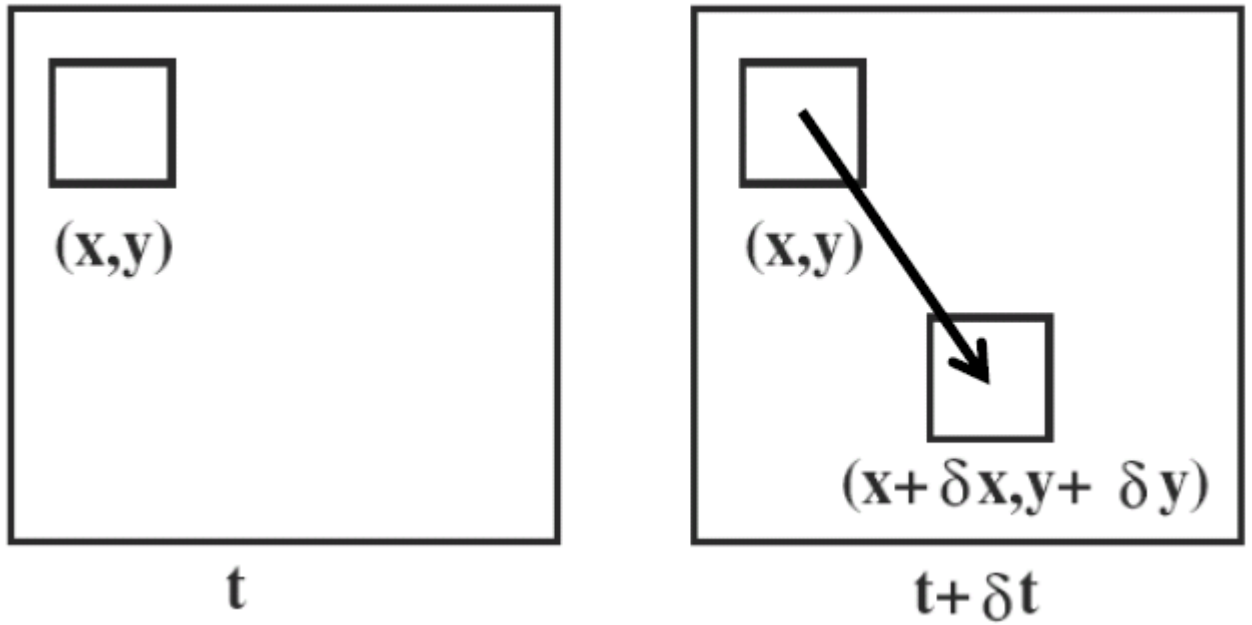


Multiple-view structure from motion

- Results of Structure from motion from 2 million user images from flickr.com - 2,106 images selected, 819,242 points

Optical Flow

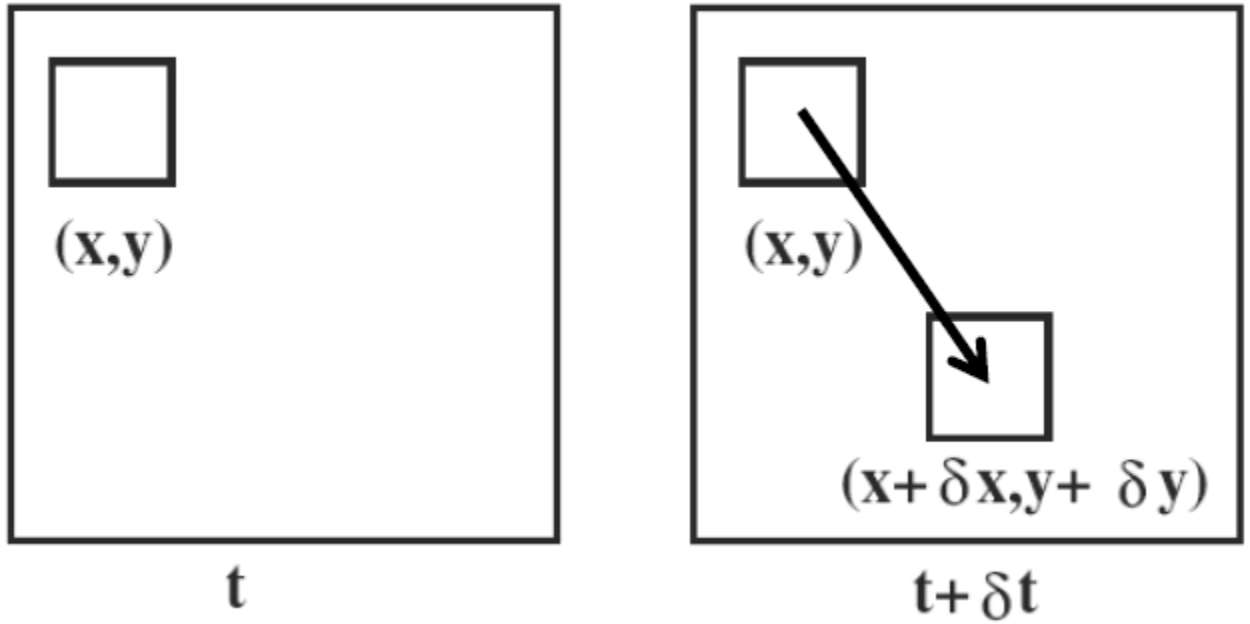
- It computes the motion vectors of all pixels in the image (or a subset of them to be faster)



$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

- Applications include collision avoidance

Optical Flow



$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0$$

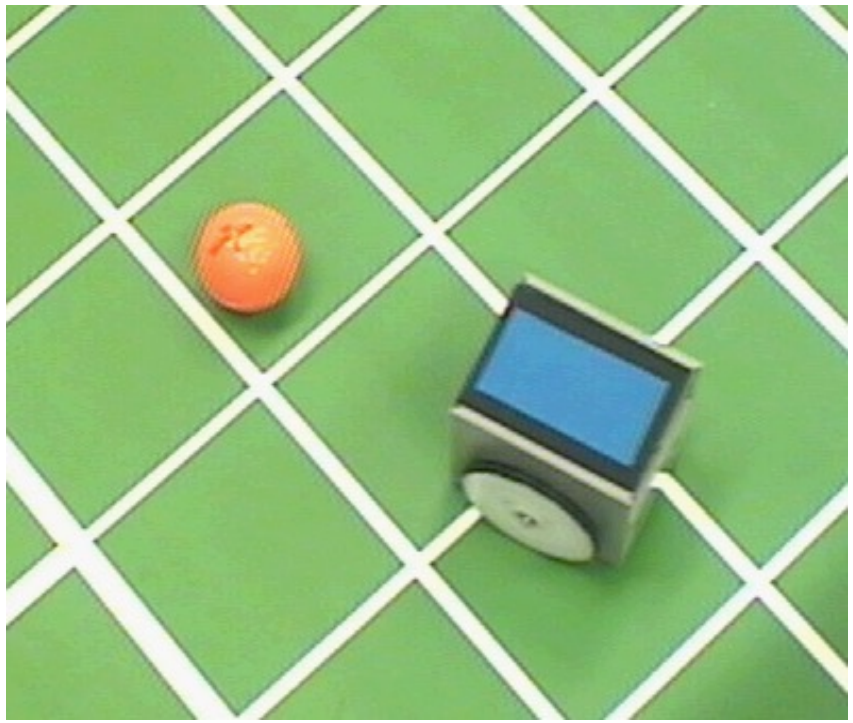
$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0$$

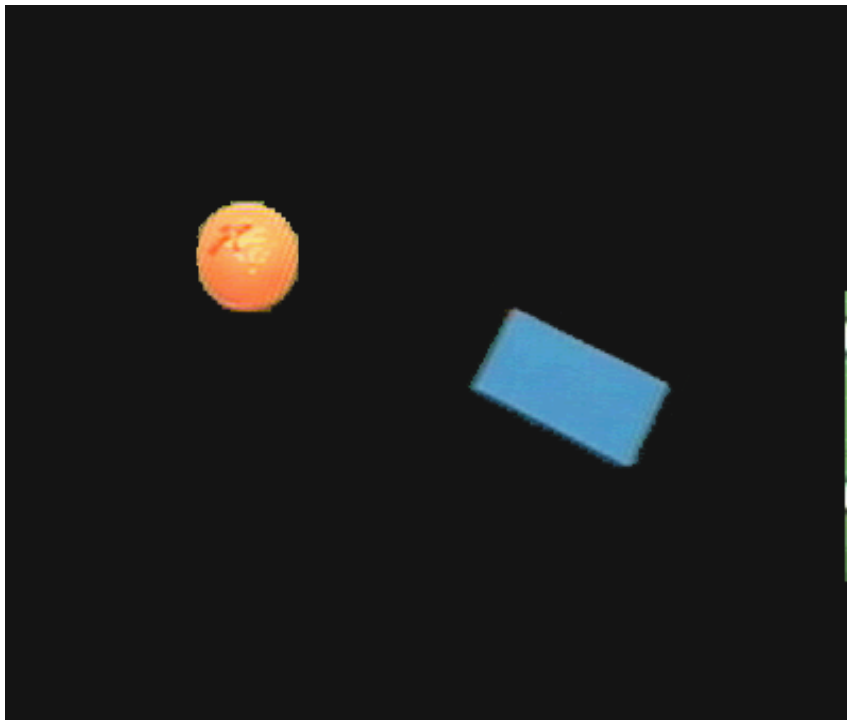
$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \cdots$$

Color Tracking

- Motion estimation of ball and robot for soccer playing using color tracking





Color segmentation with fixed thesholds

- Simple: constant thresholding:
  - selection only **iff** RGB values (r,g,b) simultaneously in R, G, and B ranges:
  - six thresholds [Rmin,Rmax], [Gmin,Gmax], [Bmin,Bmax]:

$$R_{min} < r < R_{max} \text{ and } G_{min} < g < G_{max} \text{ and } B_{min} < b < B_{max}$$

- Alternative: YUV color space
  - RGB values encode intensity of each color
  - YUV:
  - U and V together color (or chrominance)
  - Y brightness (or luminosity)
  - bounding box in YUV space => greater stability wrt. changes in illumination

Image filtering

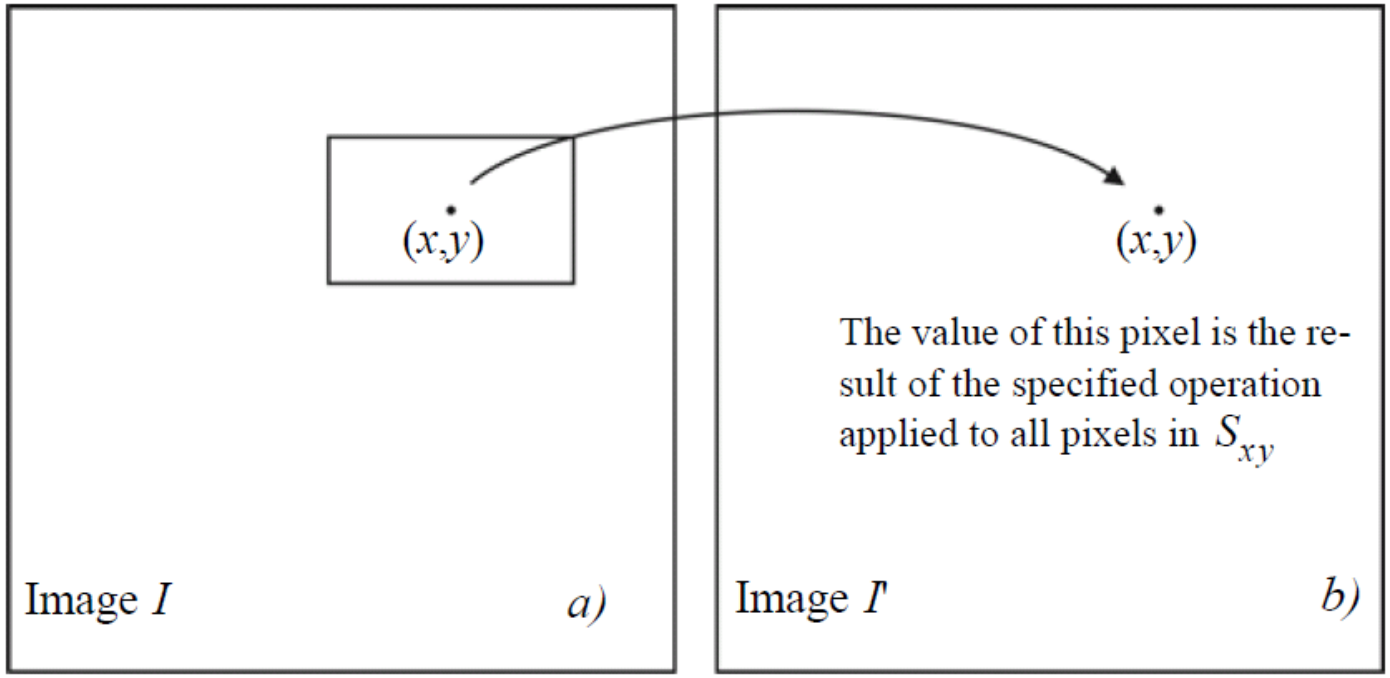
- Filer: frequency domain processing where “filtering” refers to the process of accepting or rejecting certain frequency components. E.g.:
  - Lowpass filer: pass only low frequencies => blur (smooth) an image
  - **spatial filters** (also called masks or kernels): same effect

Lowpass filtered image

Highpass filtered image

Spatial filters

- Let *Sxy* denote the set of coordinates of a neighborhood centered on an arbitrary point (x,y) in an image *I*
- Spatial filtering generates a corresponding pixel at the same coordinates in an output image *I'* where the value of that pixel is determined by a specified operation on the pixels in *Sxy*



- For example, an averaging filter is:

$$I' = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} I(r,c)$$

Linear filters

- In general, linear spatial filtering of an image with a filter *w* of size *m* x *n* is given by the expression

$$I'(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot I(x+s, y+t)$$

where *m=2a+t* and *n=2b+1* are usually assumed odd integers. The filter *w* is also called **kernel**, **mask**, or **window**.

- As observed in this formula, linear filtering is the process of moving a filter mask over the entire image and computing the sum of products at each location. In signal processing, this particular operation is also called *correlation with the kernel w* or alternatively, *convolution with the kernel w*

- The operation of convolution with the kernel *w* can be written in a more compact way as

$$I'(x,y) = w(x,y) * I(x,y)$$

where * denotes the operator of convolution

Smoothing filters (1)

- A constant averaging filter yields the standard average of all the pixels in the mask. For a 3x3 mask this writes:

$$w = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- where notice that all the coefficients sum to 1. This normalization is important to keep the same value as the original image if the region by which the filter is multiplied is uniform.

Smoothing filters (2)

- A Gaussian averaging write as

$$G_\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- To generate, say, a 3x3 filter mask from this function, we sample it about its center. For example, with σ=0.85, we get

$$G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Very popular: Such low-pass filters effectively removes high-frequency noise =>
- First derivative and especially the second derivative of intensity far more stable
- Gradients and derivatives very important in image processing =>
- Gaussian smoothing preprocessing popular first step in computer vision algorithms

Smoothing Example: SIST Lobby Scan

- Scanner: Faro Focus 3D X330
- 1 scan 20265 x 8533 pixel – about 173 million pixel
- Including color information
- Scan time: about 15 minutes

ShanghaiTech University

8 Scans of the lobby of the new SIST building

More than 400 million colored points

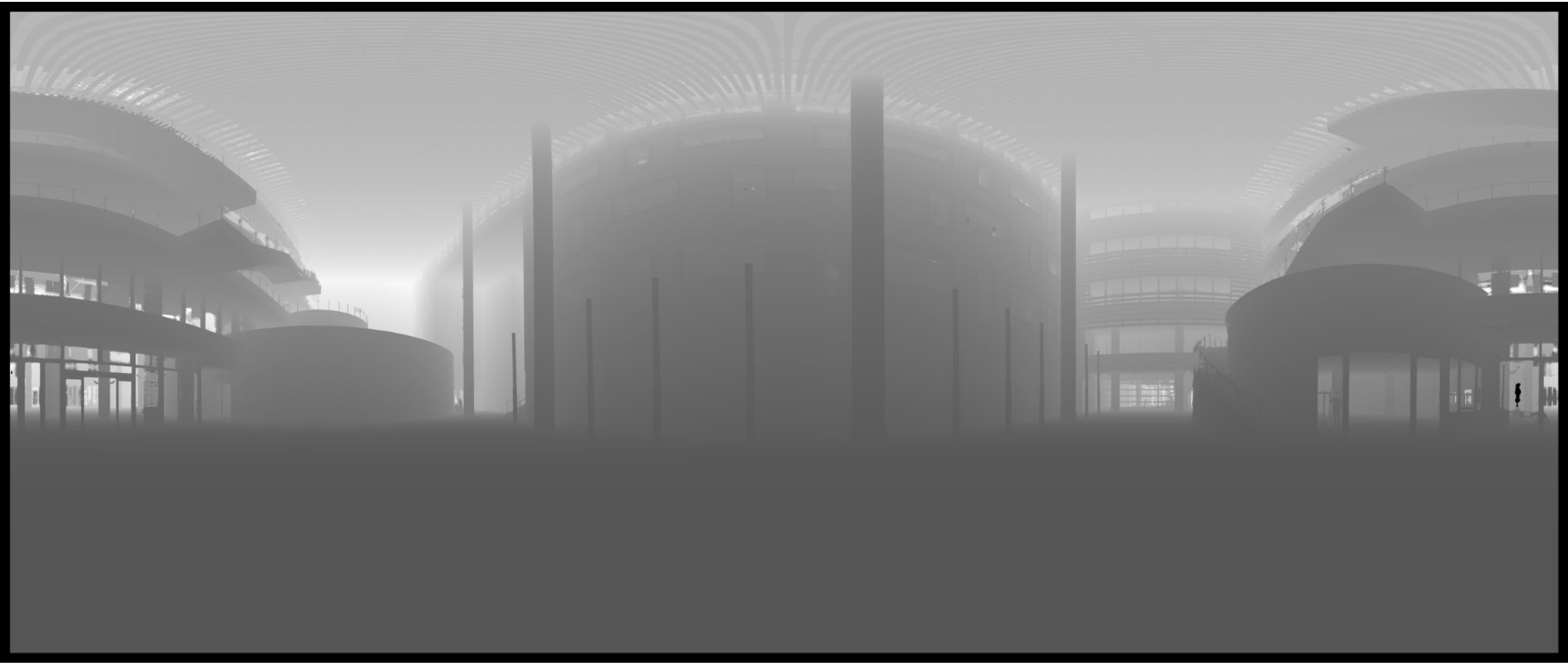
Rough alignment by hand and fine registration using ICP

SIST Lobby: RGB image 360 degrees

SIST Lobby: Depth image 360 degrees

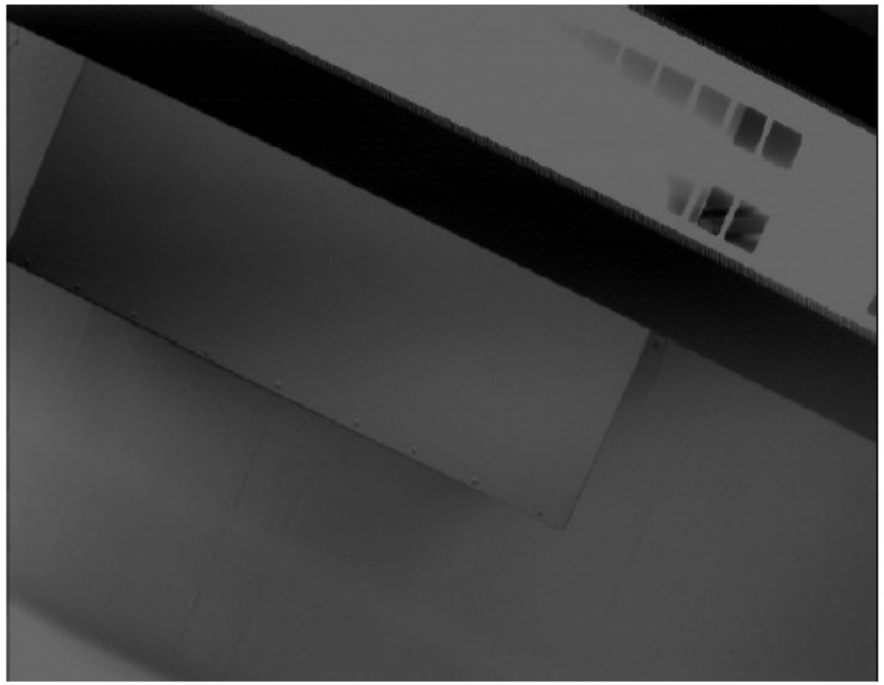
Depth image smoothing: kernel size 15, std. deviation 4

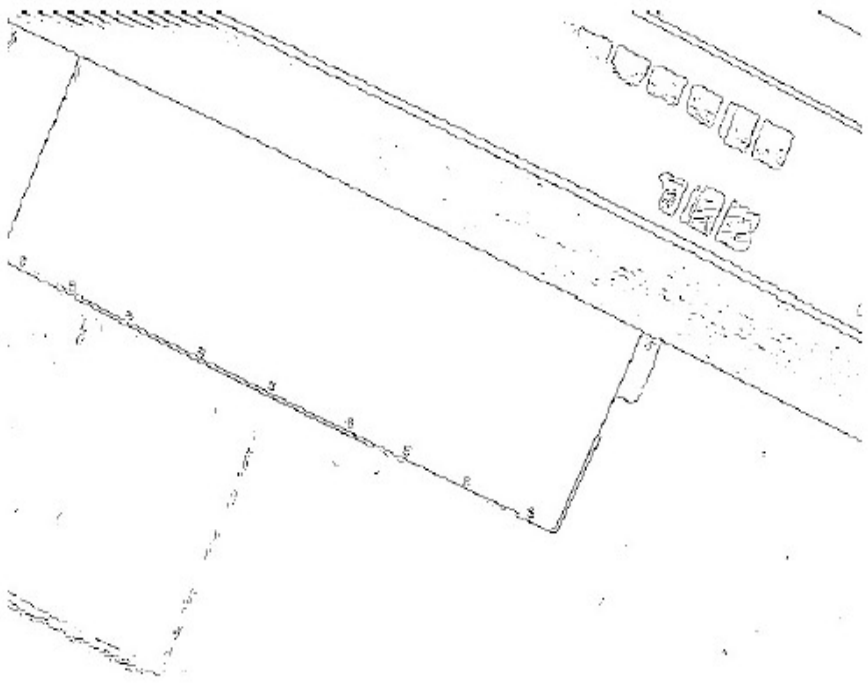
Depth image smoothing: kernel size 255, std. deviation 20



Edge Detection

- Ultimate goal of edge detection
  - an idealized line drawing.
- Edge contours in the image correspond to important scene contours.





Edge is Where Change Occurs

- Edges correspond to sharp changes of intensity
- Change is measured by 1st derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or 2nd derivative is zero.

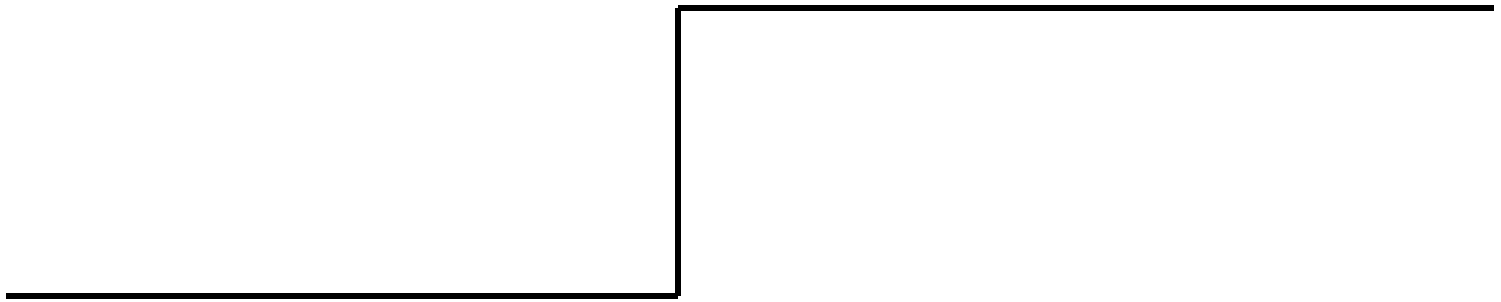
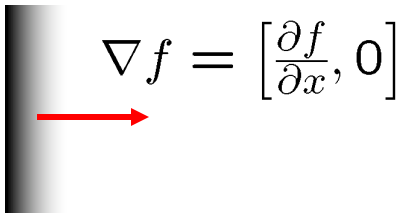
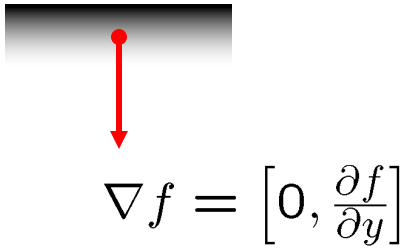


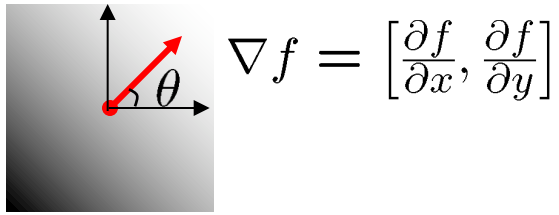
Image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

- The gradient points in the direction of most rapid change in intensity







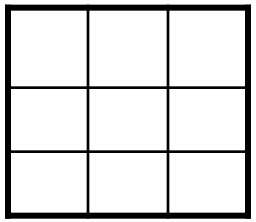
- The gradient direction is: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The gradient magnitude is: $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

The discrete gradient

- How can we differentiate a *digital* image f[x,y]?
  - Option 1: reconstruct a continuous image, then take gradient
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx f[x+1, y] - f[x, y]$$

- How to implement this as a spatial filter?



w

Gradient Edge Detectors

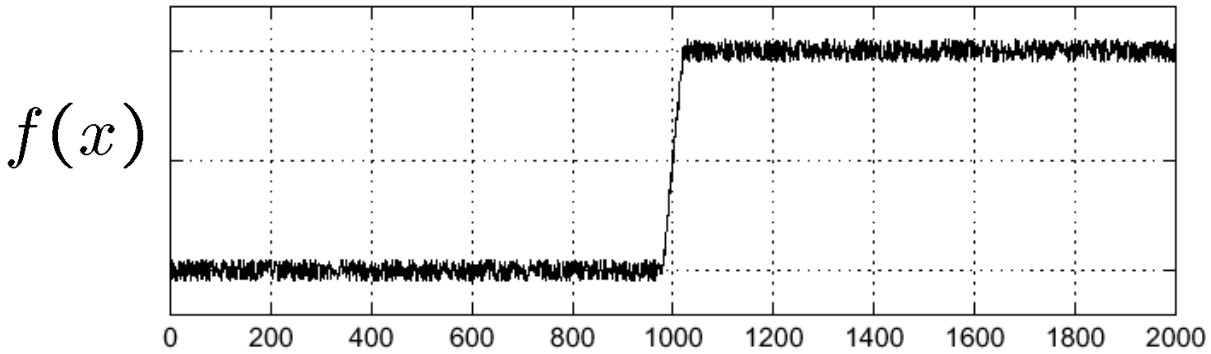
- **Roberts** $|G| \cong \sqrt{r_1^2 + r_2^2}$; $r_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$; $r_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

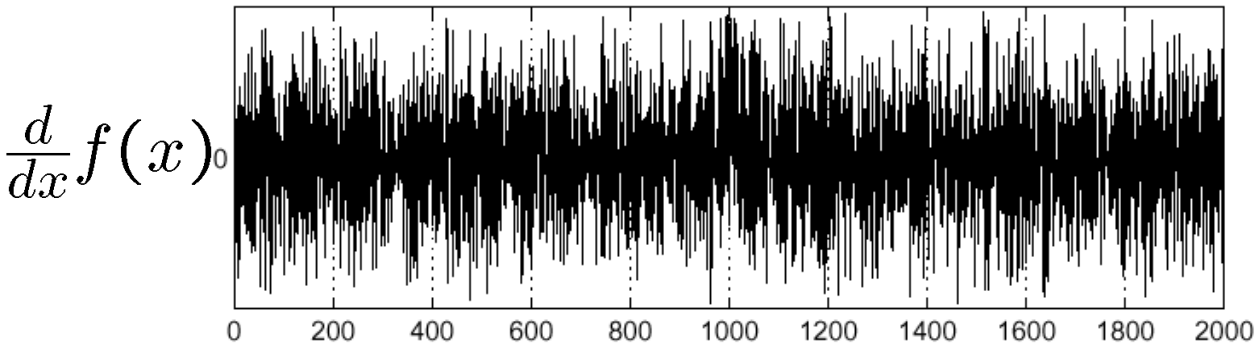
- **Prewitt** $|G| \cong \sqrt{p_1^2 + p_2^2}$; $\theta \cong \text{atan}\left(\frac{p_1}{p_2}\right)$; $p_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$; $p_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

- **Sobel** $|G| \cong \sqrt{s_1^2 + s_2^2}$; $\theta \cong \text{atan}\left(\frac{s_1}{s_2}\right)$; $s_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$; $s_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





- Where is the edge?

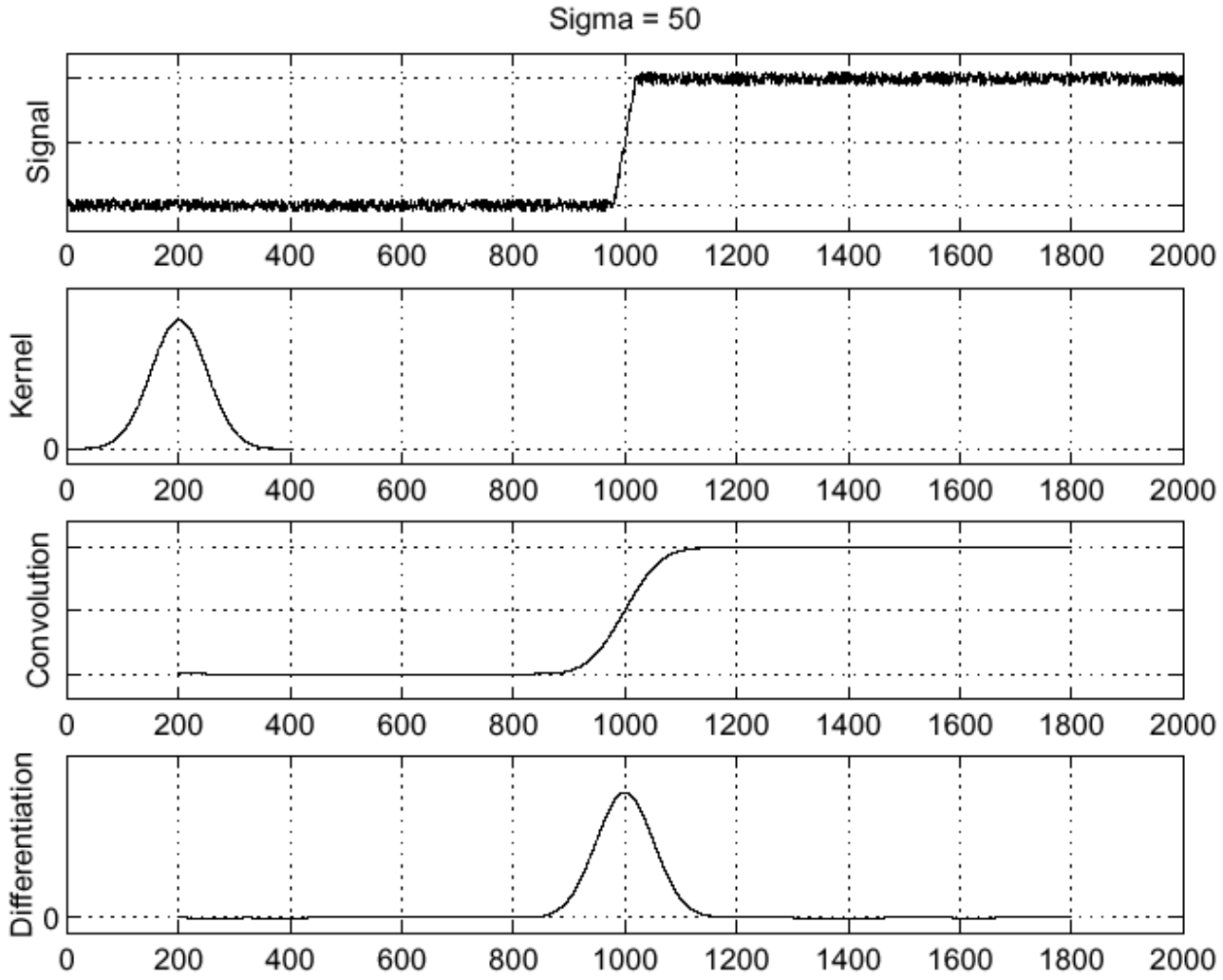
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



- Where is the edge?
- Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

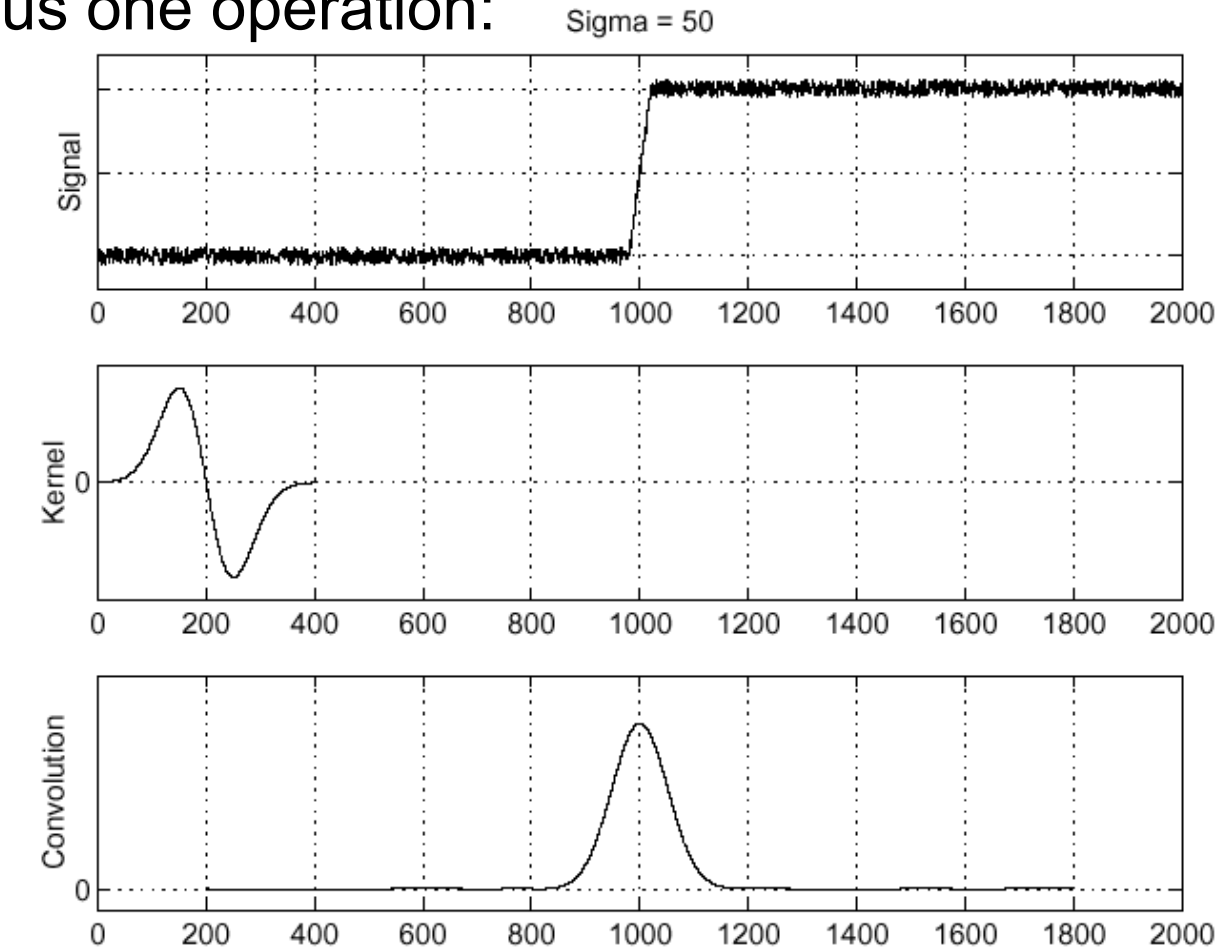
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- This saves us one operation:

f

$\frac{\partial}{\partial x}h$

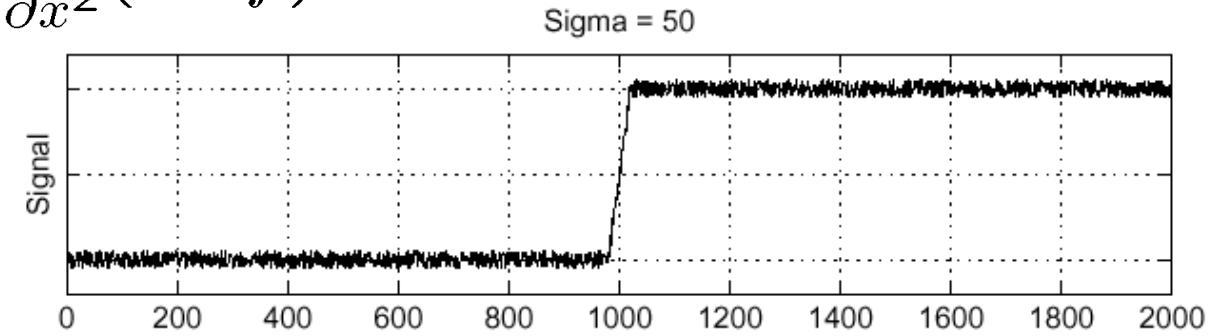
$(\frac{\partial}{\partial x}h) \star f$



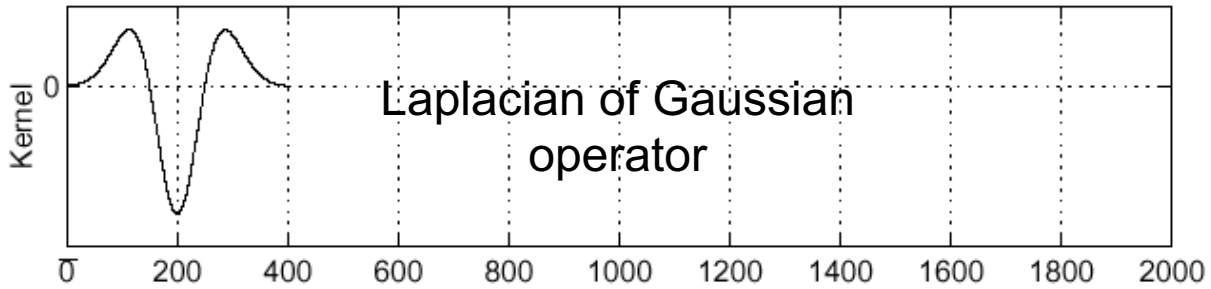
The Canny Edge Detector

- Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

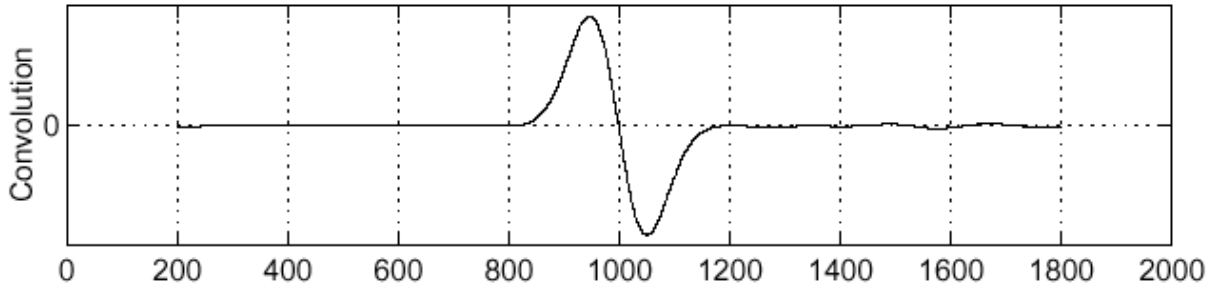
f



$\frac{\partial^2}{\partial x^2}h$

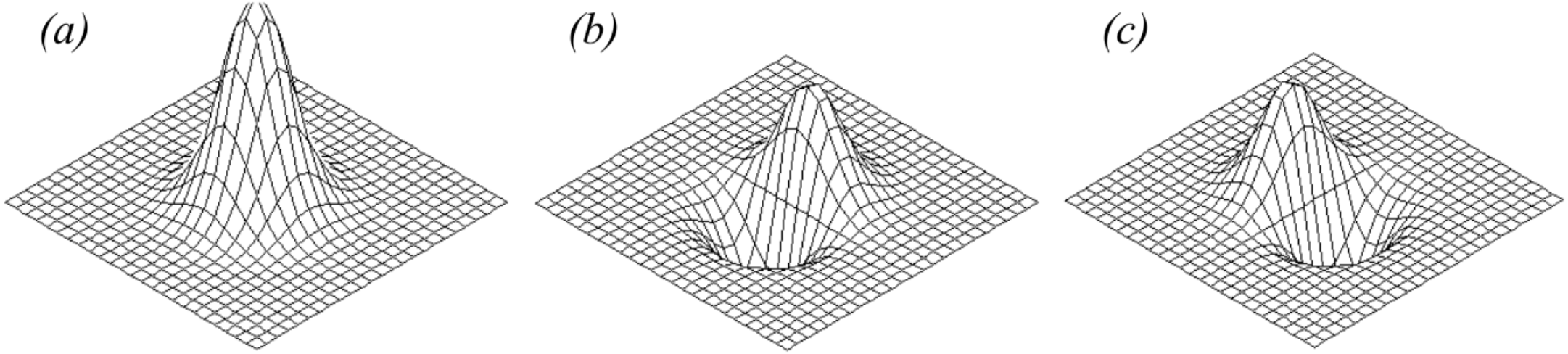


$(\frac{\partial^2}{\partial x^2}h) \star f$



- Where is the edge?
- Zero-crossings of bottom graph

2D Canny edge detector



$$G_\sigma(x,y) = G_\sigma(x)G_\sigma(y) \qquad f_V(x,y) = G'_\sigma(x)G_\sigma(y) \qquad f_H(x,y) = G'_\sigma(y)G_\sigma(x)$$

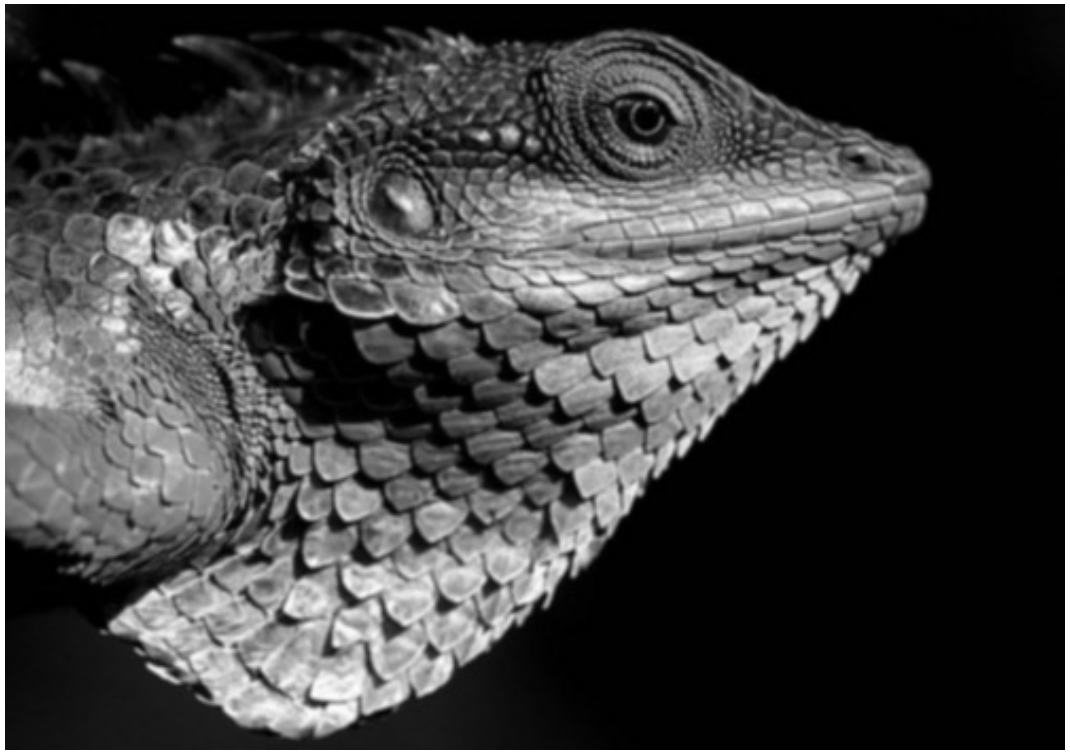
- Two perpendicular filters:
  - Convolve imge $I(x,y)$ with $f_V(x,y)$ and $f_H(x,y)$ – obtaining $R_V(x,y)$ and $R_H(x,y)$
  - Use square of gradient magnitude: $R(x,y) = R_V^2(x,y) + R_H^2(x,y)$
  - Mark peaks in $R(x,y)$ above a threshold

The Canny edge detector

https://en.wikipedia.org/wiki/Canny_edge_detector

The Canny edge detector

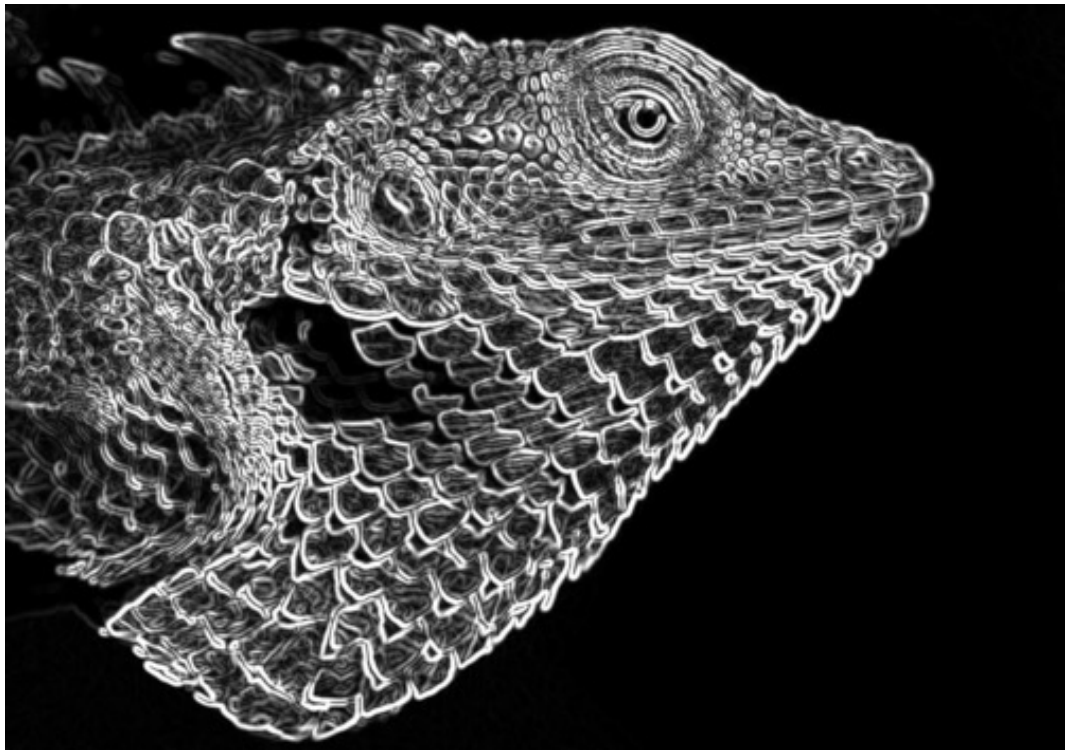
- Image has been reduced to grayscale, and a 5x5 Gaussian filter with σ=1.4 has been applied.



https://en.wikipedia.org/wiki/Canny_edge_detector

The Canny edge detector

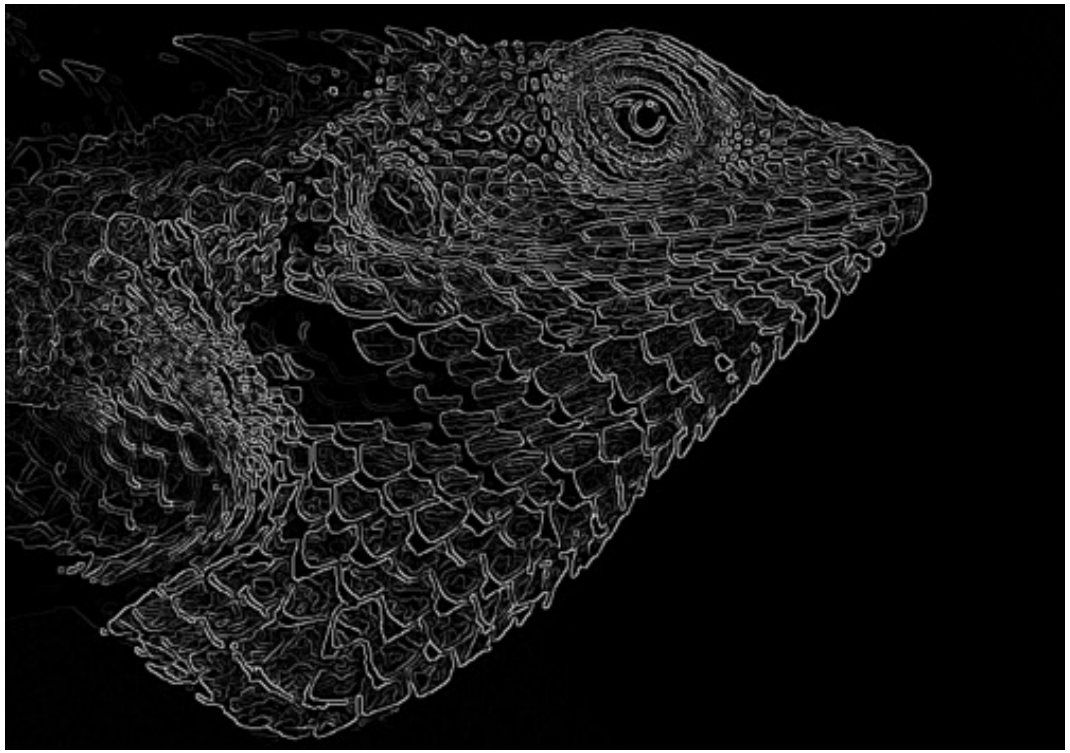
The intensity gradient of the previous image. The edges of the image have been handled by replicating.



https://en.wikipedia.org/wiki/Canny_edge_detector

The Canny edge detector

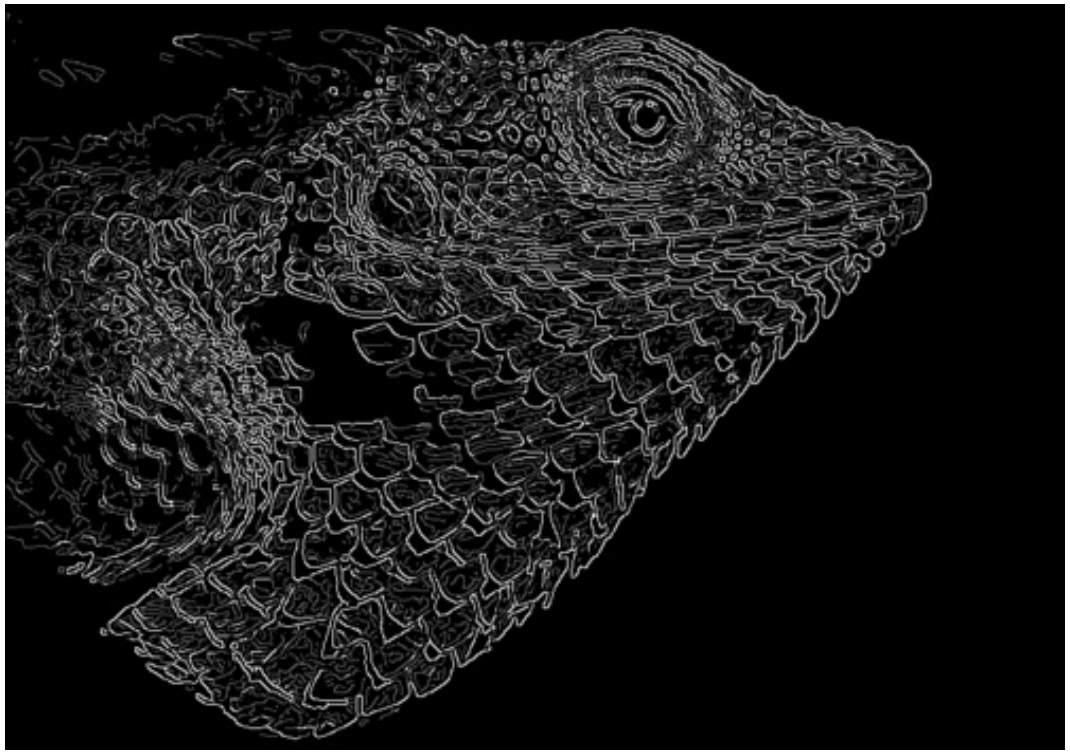
Non-maximum suppression applied to the previous image



https://en.wikipedia.org/wiki/Canny_edge_detector

The Canny edge detector

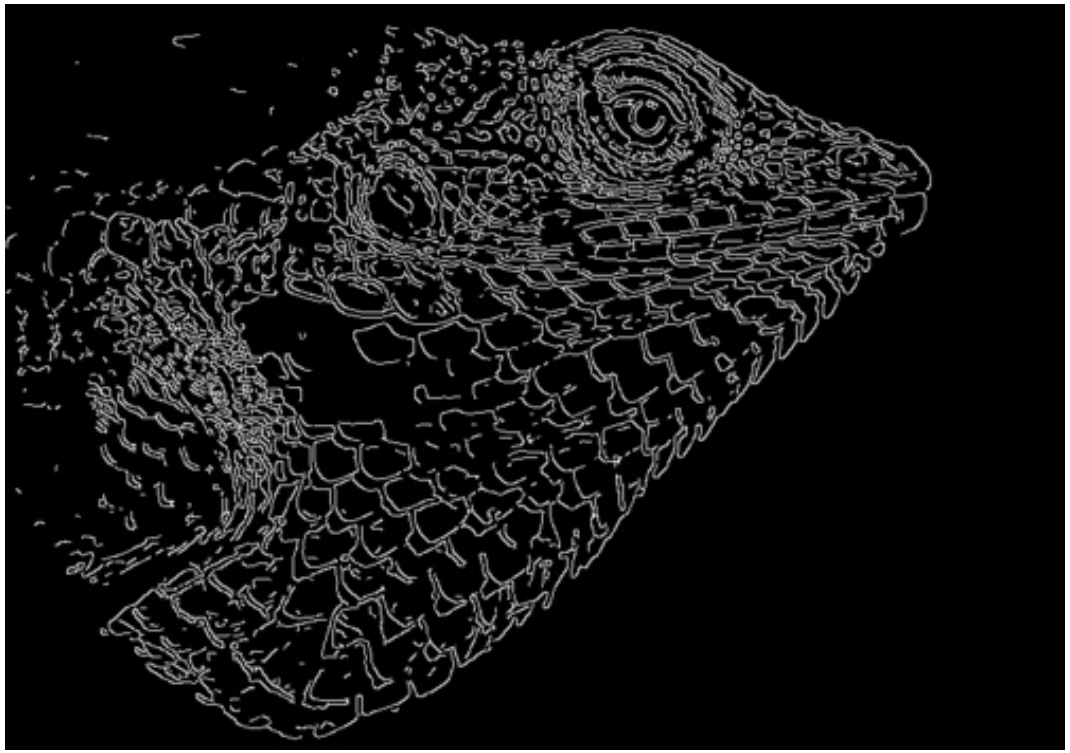
Double thresholding applied to the previous image. Weak pixels are those with a gradient value between 0.1 and 0.3. Strong pixels have a gradient value greater than 0.3.



https://en.wikipedia.org/wiki/Canny_edge_detector

The Canny edge detector

Hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges (8-neighborhood).



https://en.wikipedia.org/wiki/Canny_edge_detector