EE565:Mobile Robotics
Lecture 7

Welcome

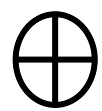
Dr. Ahmad Kamal Nasir

Today's Objectives

- Visual Odometry
  - Camera model
  - Calibration
- Feature detection
  - Harris corners
  - SIFT/SURF etc.
- Optical Flow
  - Kanade-Lucas-Tomasi Tracker

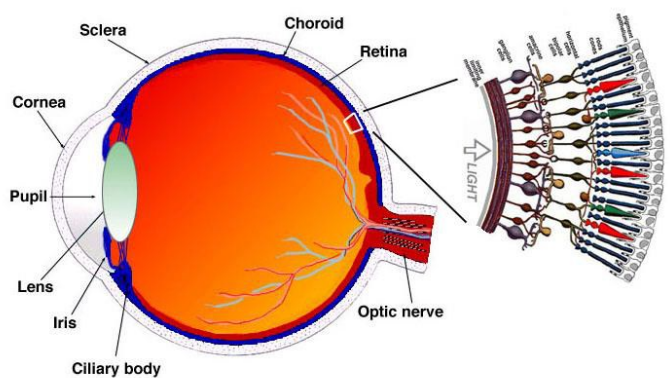
Vision

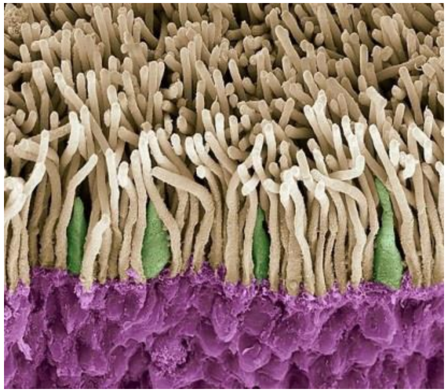
Use both eyes…at arm’s length, center target within finger OK sign Lock hand in position…see which eye is still aligned by closing the other. The eye with good alignment is your dominant eye!

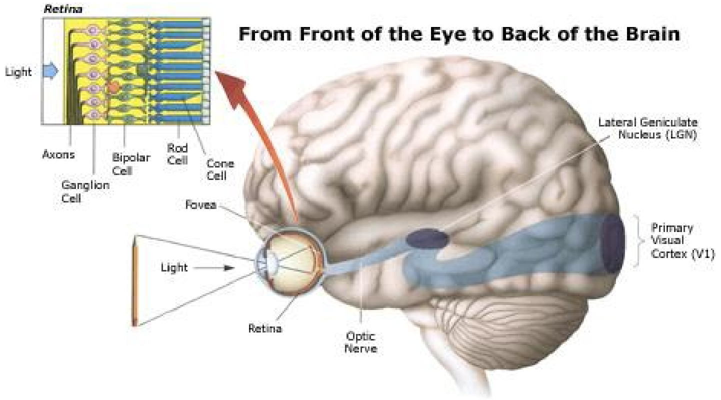


Human Vision

- Larger portion of our brain is used for vision
- Retina:$1000mm^2$ 120mills Rods, 7mills Cones
- Human Eye Resolution≈ 500 Megapixel
- Data rate≈ 3 GB/sec

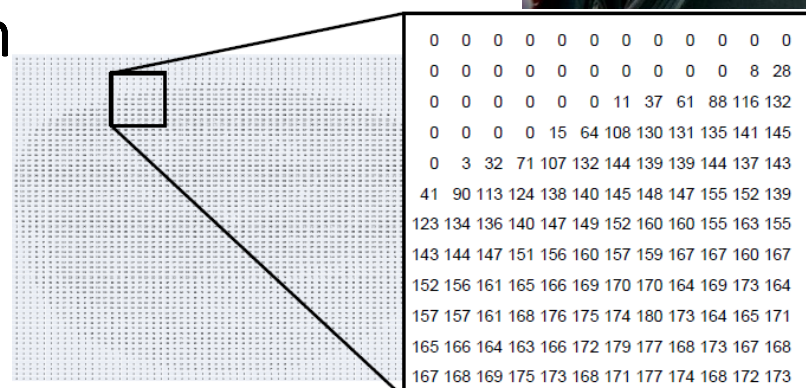






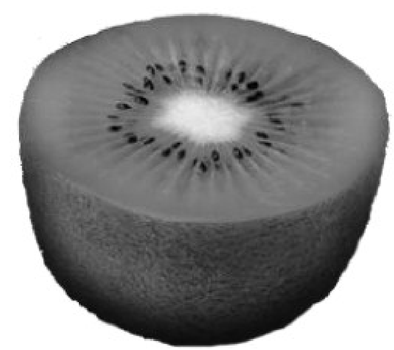
Computer Vision (Perception) is hard!

- Perception is hard because
  - A lot of data
  - Uncertainty
  - Model estimation
  - Contextual information
  - Cognitive reasoning



0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	8	28
0	0	0	0	0	0	11	37	61	88	116	132
0	0	0	0	15	64	108	130	131	135	141	145
0	3	32	71	107	132	144	139	139	144	137	143
41	90	113	124	138	140	145	148	147	155	152	139
123	134	136	140	147	149	152	160	160	155	163	155
143	144	147	151	156	160	157	159	167	167	160	167
152	156	161	165	166	169	170	170	164	169	173	164
157	157	161	168	176	175	174	180	173	164	165	171
165	166	164	163	166	172	179	177	168	173	167	168
167	168	169	175	173	168	171	177	174	168	172	173

Computer Perception



Human Perception

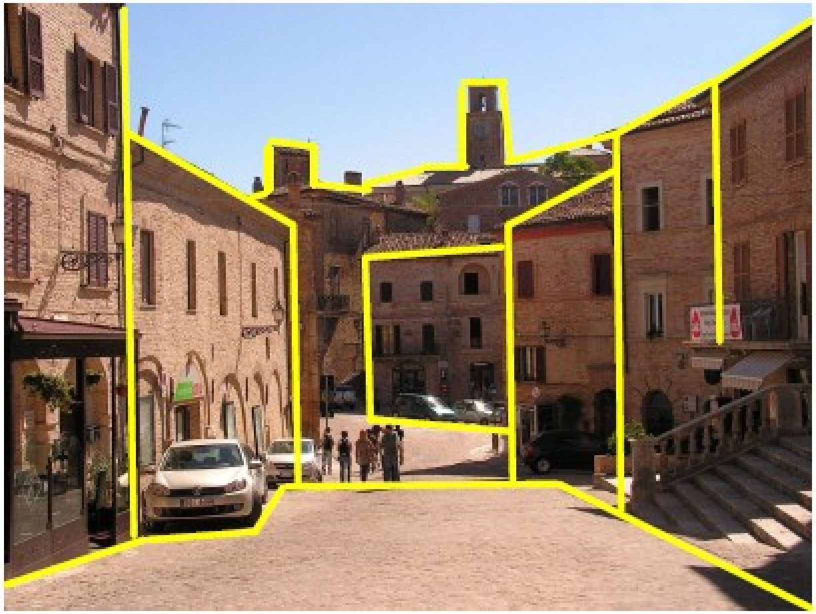
Image Processing Vs. Computer Vision

- Image processing we deals with the images and the outputs are also images
  - It deals with giving effects various effects to the image
- Computer vision also deals with images but the outputs are data.
  - It deals with extraction of meaningful information from images

Computer Vision

Automatic extraction of meaningful information from images and videos

Semantic Information



Geometric Information

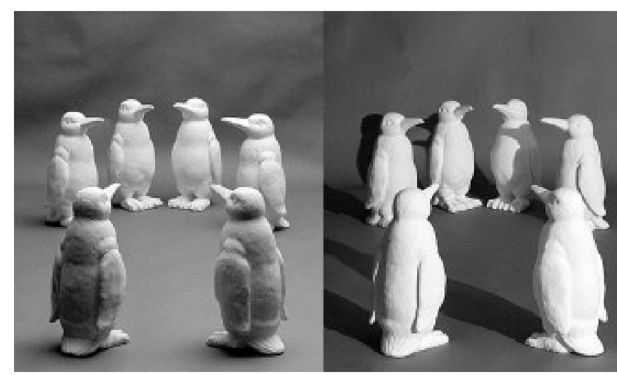
Challenges In Computer Vision

- Viewpoint changes
- Illumination changes
- Object intra-class variations
- Inherent ambiguities

Viewpoint changes

Inherent ambiguities

Object intra-class variations



Illumination changes

Applications

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

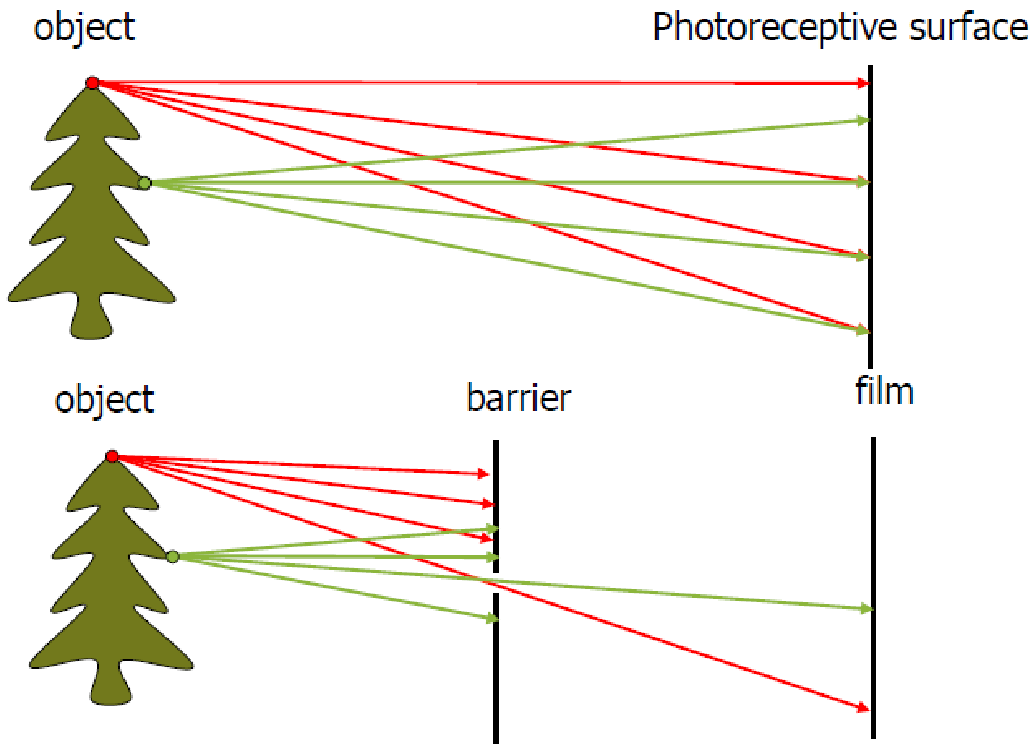
Visual Odometry

- Camera Model
- Calibration
- Feature Extraction
- Feature Tracking
- Camera Pose Estimation
- Triangulation

- Raw Data(Vision/Ranges)
- Clustering(Corners/Lines)
- Objects (Doors/Rooms)
- Semantics(Contextual Information, Place recogniation)

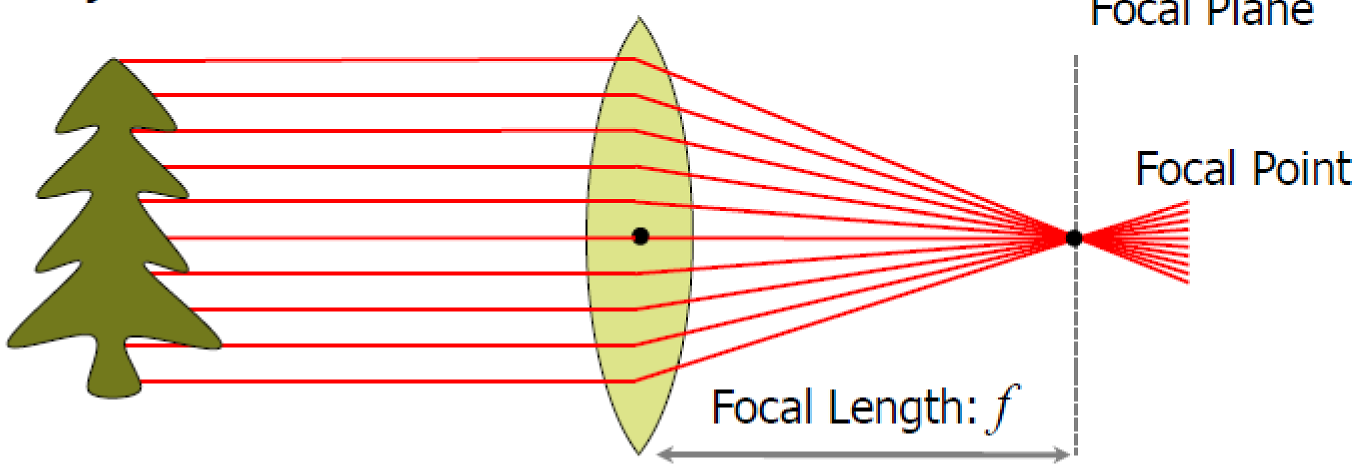
Image Formation

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

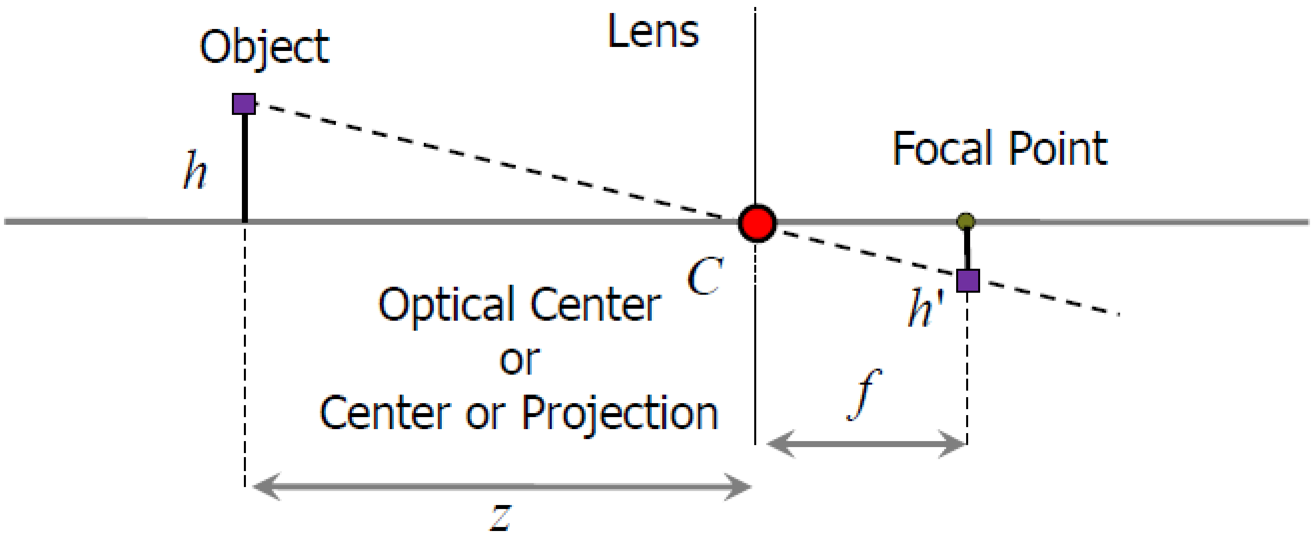


Why Use a Lens?

- Ideal pinhole: less amount of light, diffraction
  Bigger pinhole: blurry image
- Lens focuses light onto the film
  Rays passing through optical center are inert
- All rays parallel to the optical axis converge at the focal point



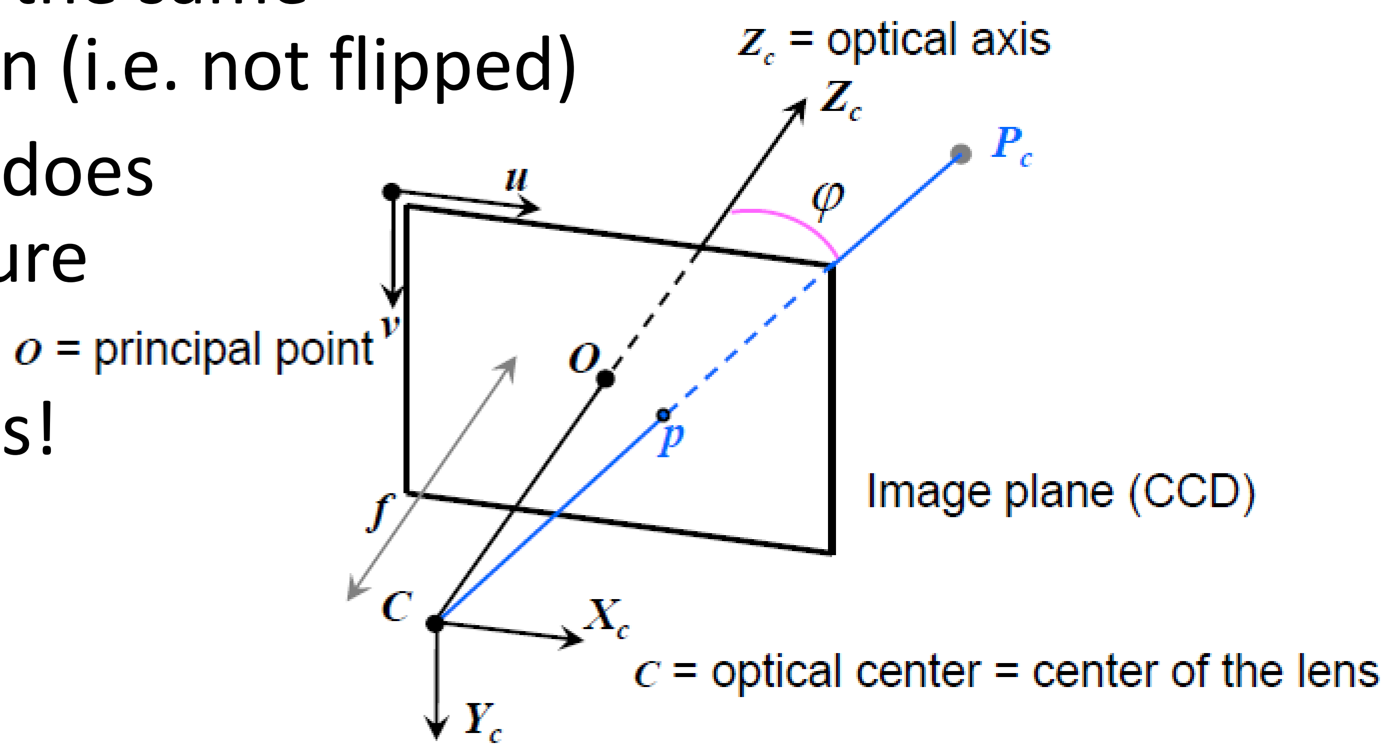
Pinhole camera model



$$\frac{h'}{h} = \frac{f}{z} \Rightarrow h' = \frac{f}{z} h$$

Perspective camera

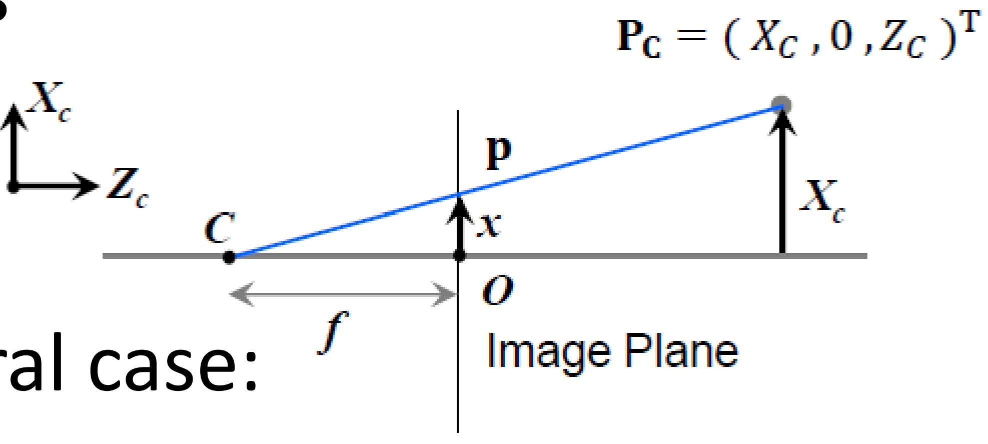
- For convenience, the image plane is usually represented in front of C such that the image preserves the same orientation (i.e. not flipped)
- A camera does not measure distances but angles!



Perspective Projection

- The Camera point $P_c = (X_c, 0, Z_c)$ projects to $p = (x, y)$ 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:

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

Scene Points into Pixels

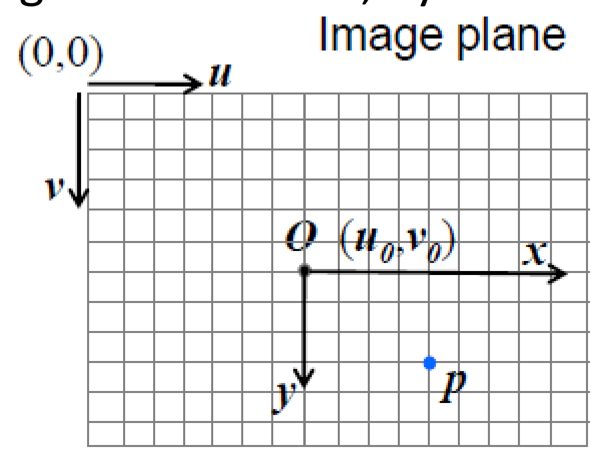
- To convert $\mathbf{p}$, from the local image plane coordinates (x, y) to the pixel coordinates (u, v), we need to account for optical center $O = (u0, v0)$ and scale factor k for the pixel-size

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

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

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

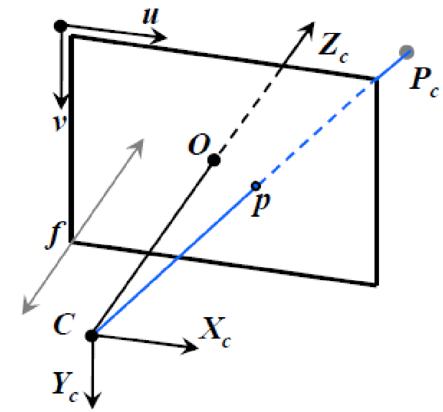
$$p = \begin{pmatrix} u \\ v \end{pmatrix} \Rightarrow \tilde{p} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$



Camera Model in Homogenous Form

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

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

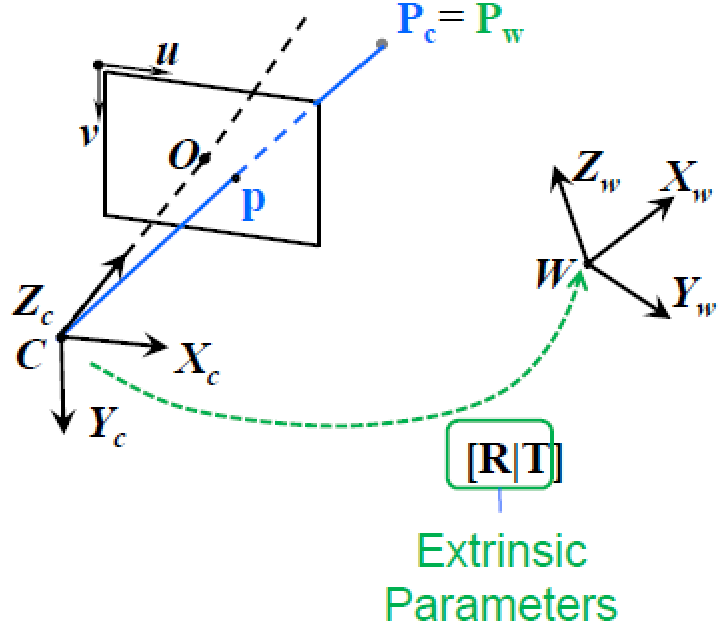


Expressed in matrix form and homogeneous coordinates

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

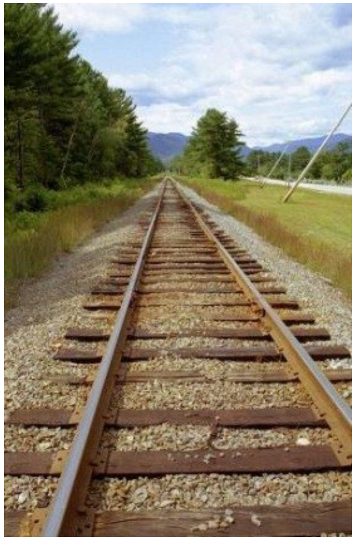
$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} R & | & T \end{bmatrix}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

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

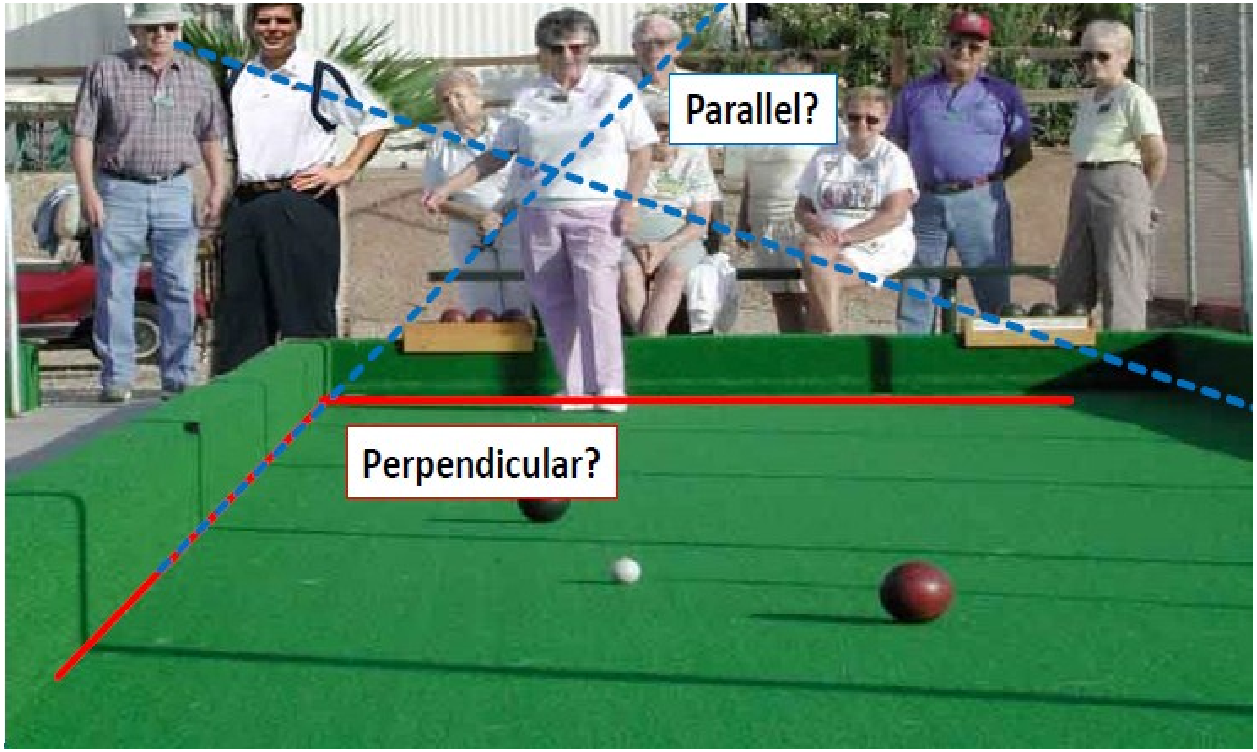


Perspective Effects

- What is lost?



- What is preserved?



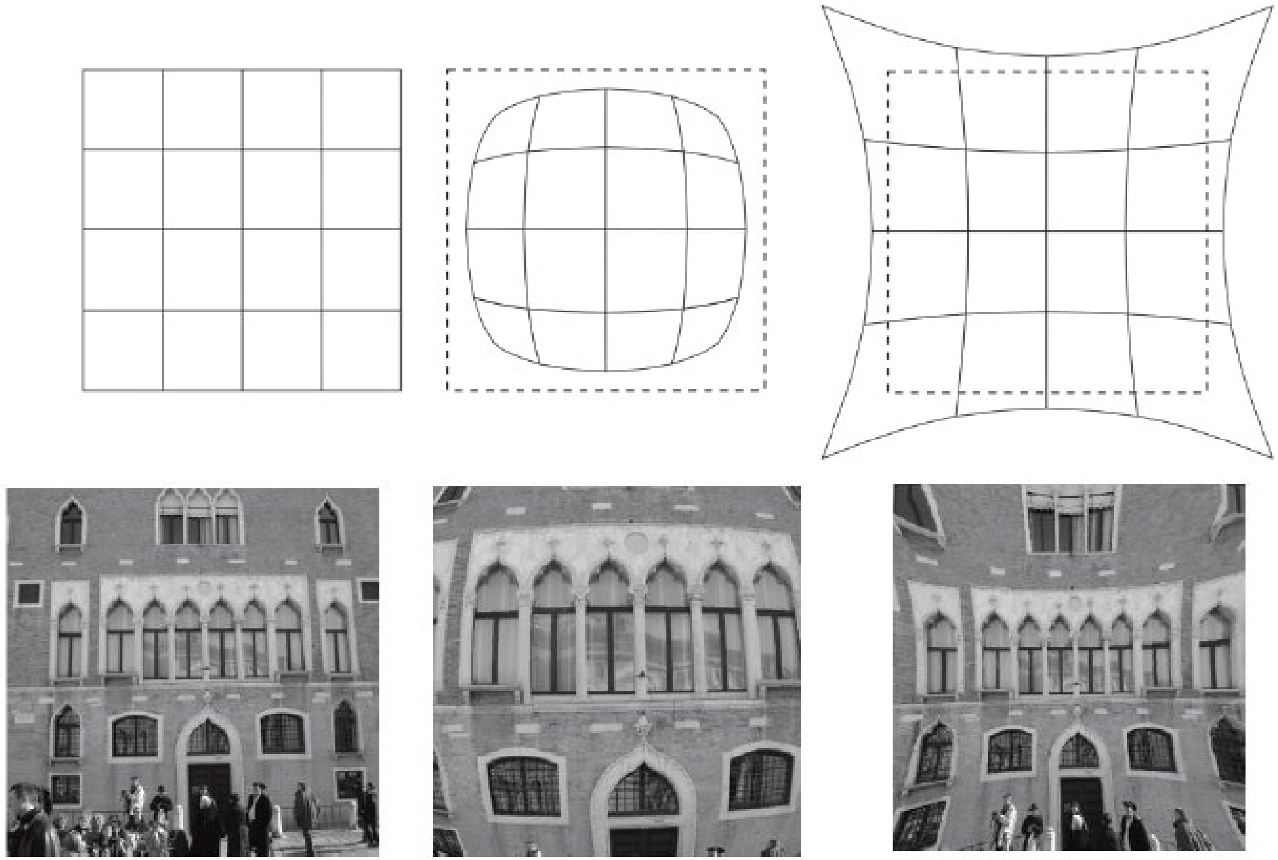
Lens Distortion

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

- The amount of distortion of the coordinates of the observed image is a nonlinear function of their radial distance.

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

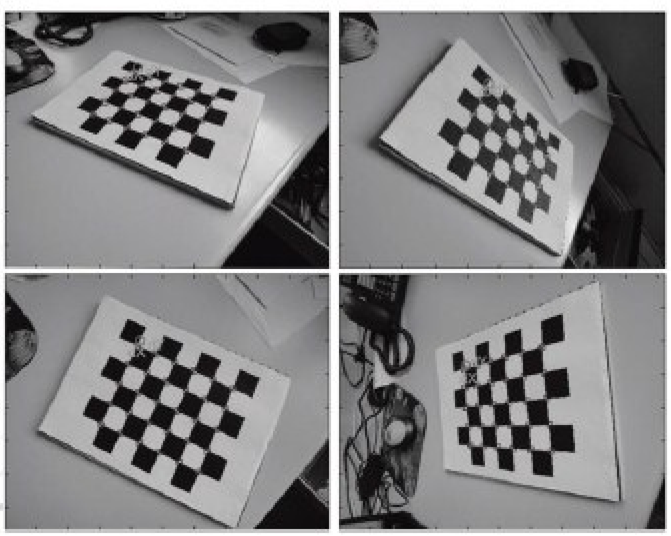
$$r^2 = (u - u_0)^2 + (v - v_0)^2$$



No distortion | Barrel distortion | Pincushion

Camera Calibration

- Goal: to determine the intrinsic parameters of the camera model
- The standard method consists of measuring the 3D positions of n control points on a calibration object and the 2D coordinates of their image projections
  - n ≥ 6 non-coplanar control points on a three-dimensional calibration target
  - n ≥ 4 non-collinear control points on a planar pattern



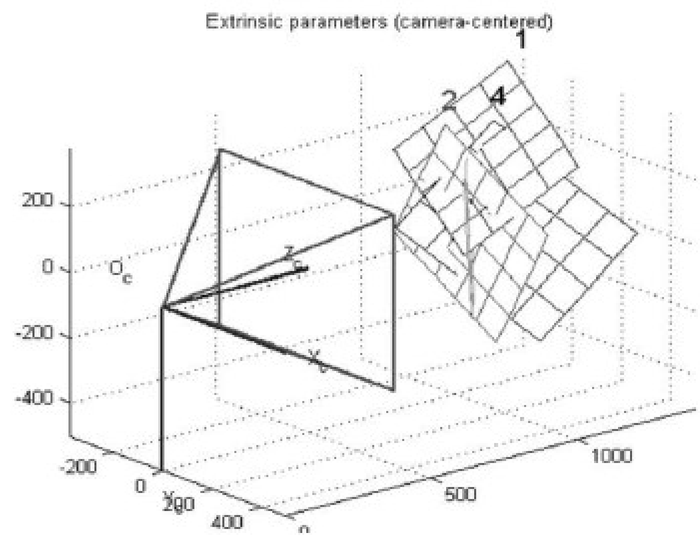
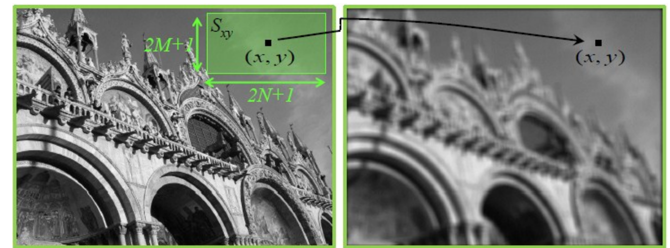


Image Filtering



- Averaging Filter

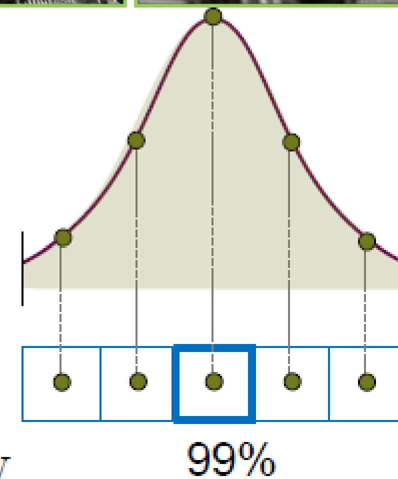
$$J(x,y) = \frac{\sum_{(r,c)\in S_{xy}} I(r,c)}{(2M+1)(2N+1)}$$

- Gaussian Filter

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

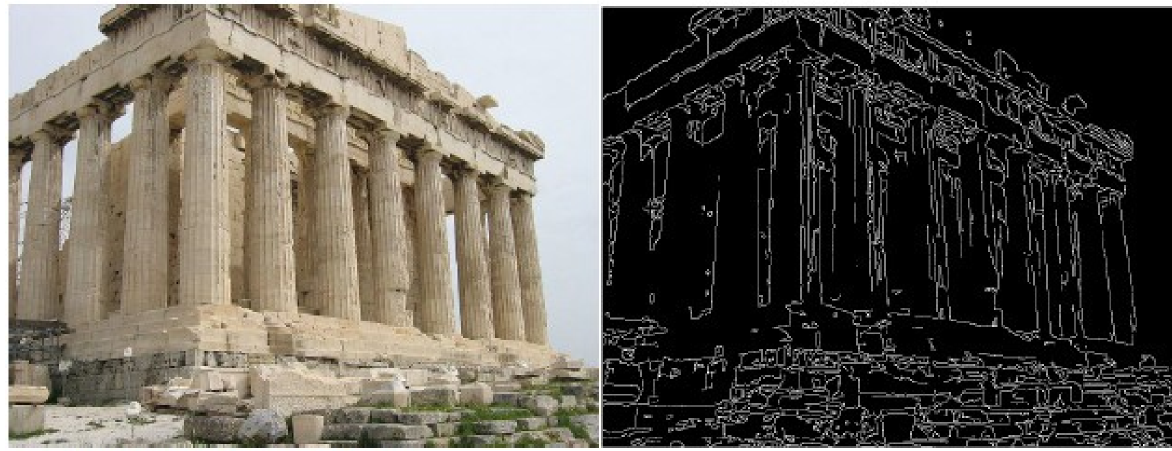
$\mu = 0$

σ : controls the amount of smoothing

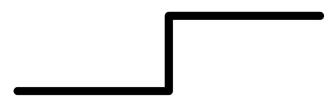


- Basic Filtering Operations
  - Convolution $$J(x) = F * I(x) = \sum_{i=-N}^{N} F(i) I(x-i)$$
  - Correlation $$J(x) = F \circ I(x) = \sum_{i=-N}^{N} F(i) I(x+i)$$

Edge Detection


Parthenon by Tim Bekaert, Wikimedia Commons

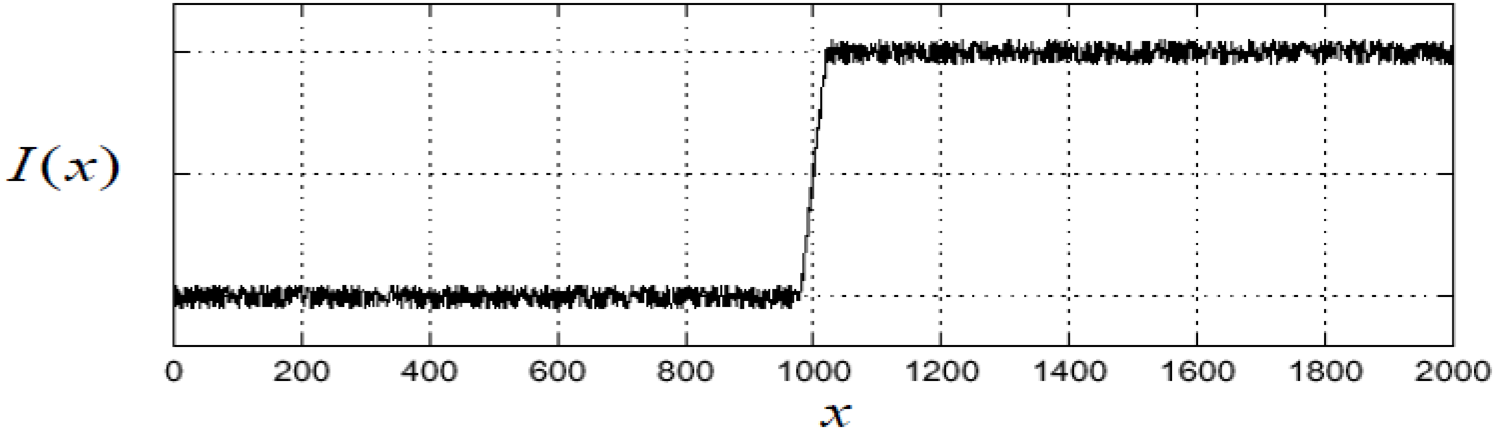
- Edge contours in the image correspond to important scene contours.
- Ultimate goal of edge detection: an idealized line drawing.
- Edges correspond to sharp changes of intensity
- Change is measured by 1st order derivative in 1D
- Or 2nd order derivative is zero.

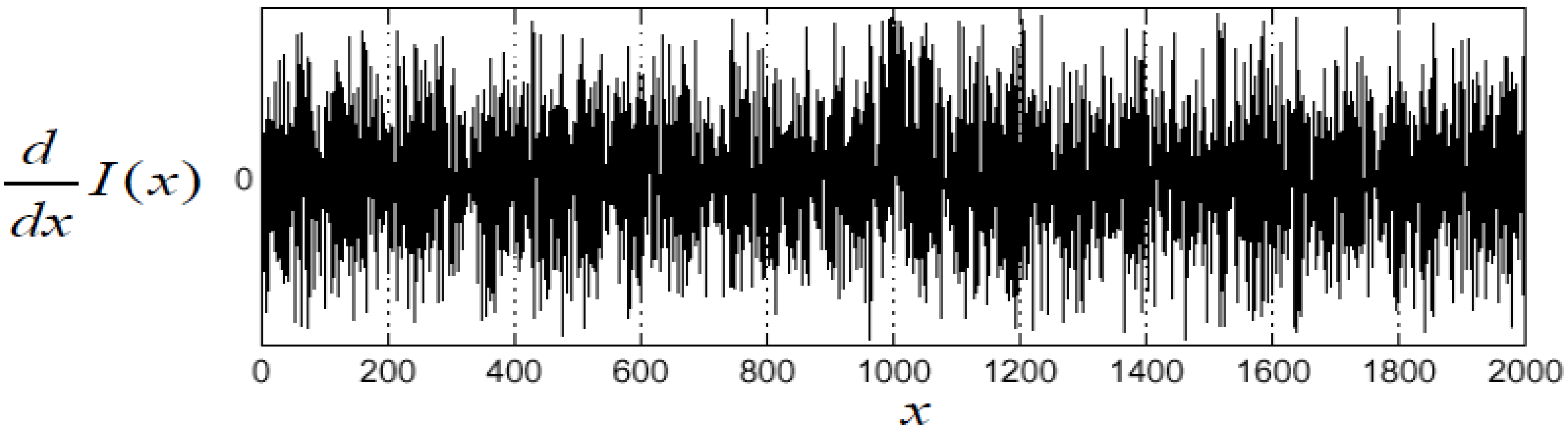


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \quad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \quad \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

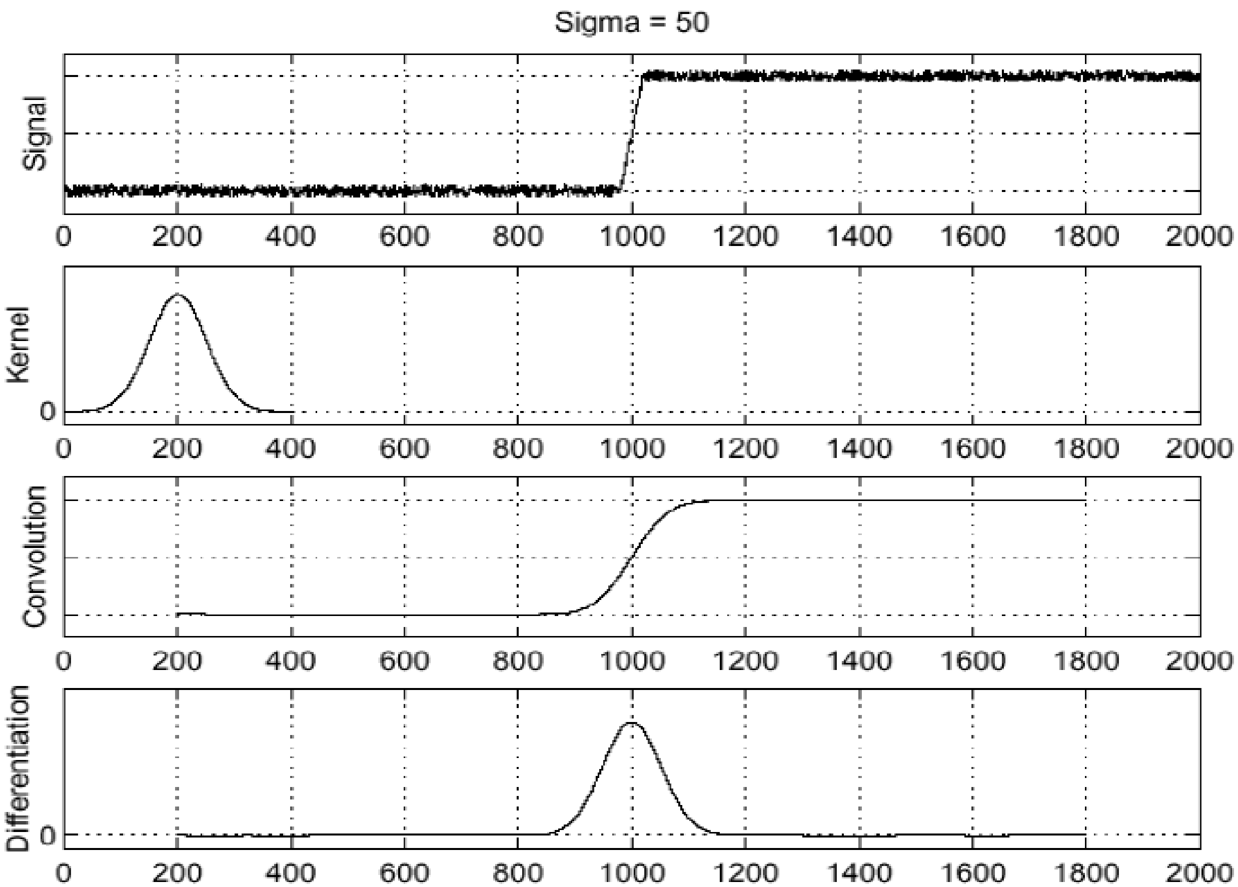
1D Edge Detection

- Image intensity shows an obvious change
- Where is an edge?





Solution: Smoothing



$$I(x)$$

$$G_\sigma(x)$$

$$s(x) = I(x) * G_\sigma(x)$$

$$s'(x) = \frac{d}{dx}(s(x))$$

Edges occur at maxima/minima of $s'(x)$

Drawback: Increased computation. Can we do something better?

Derivative Theorem of Convolution

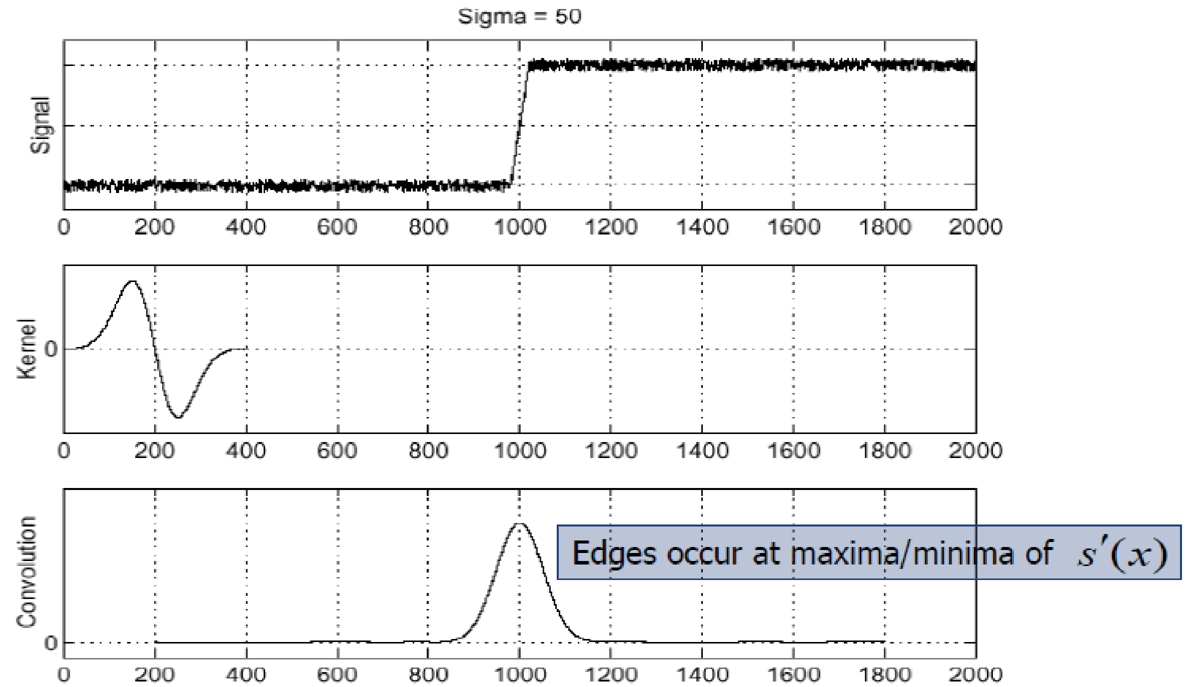
$$s'(x) = \frac{d}{dx}\big(G_\sigma(x) * I(x)\big) = G'_\sigma(x) * I(x)$$

This saves us one operation:

$$I(x)$$

$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x)$$

$$s'(x) = G'_\sigma(x) * I(x)$$



How to find edge rather than a maxima or minima?

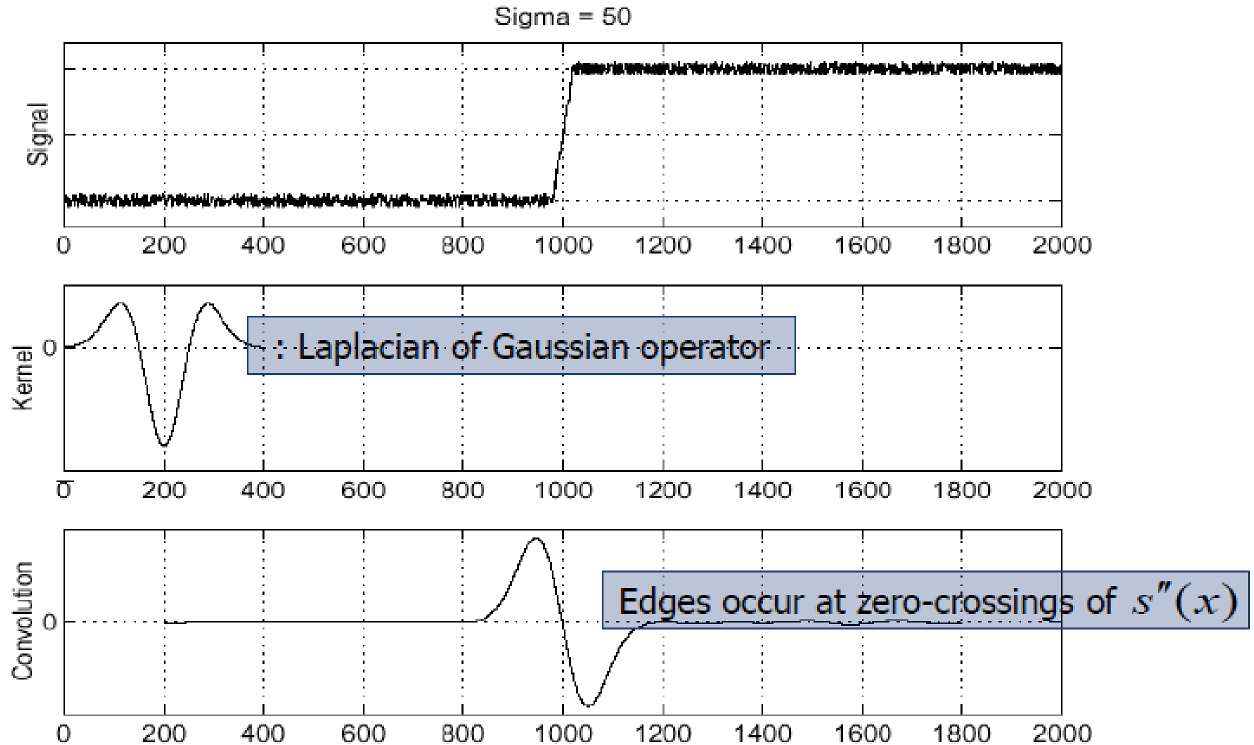
Zero Crossing

- Locations of Maxima/minima in $\dot{s}(x)$ are equivalent to $\ddot{s}(x)$

$$I(x)$$

$$G''_\sigma(x) = \frac{d^2}{dx^2} G_\sigma(x)$$

$$s''(x) = G''_\sigma(x) * I(x)$$



2D Edge Detection

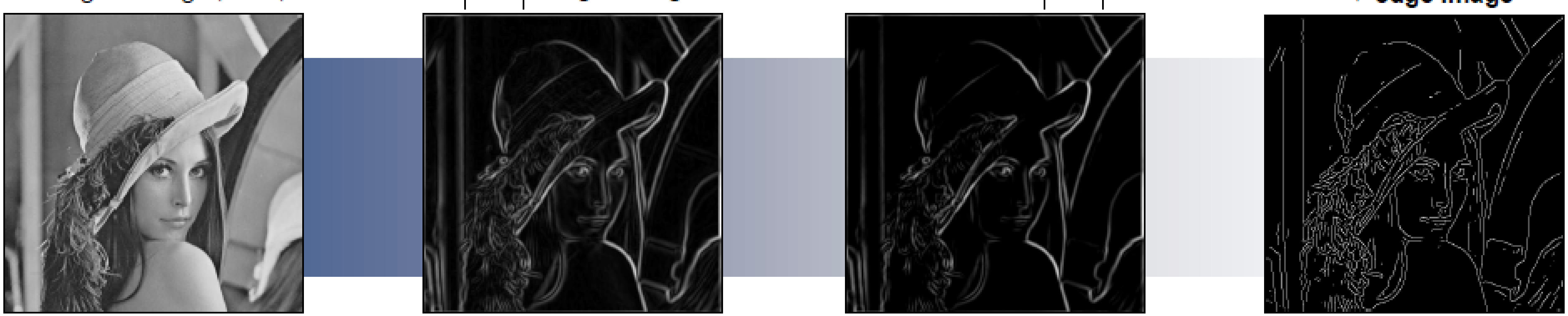
- Find gradient of smoothed image in both directions

Usually use a separable filter such that:

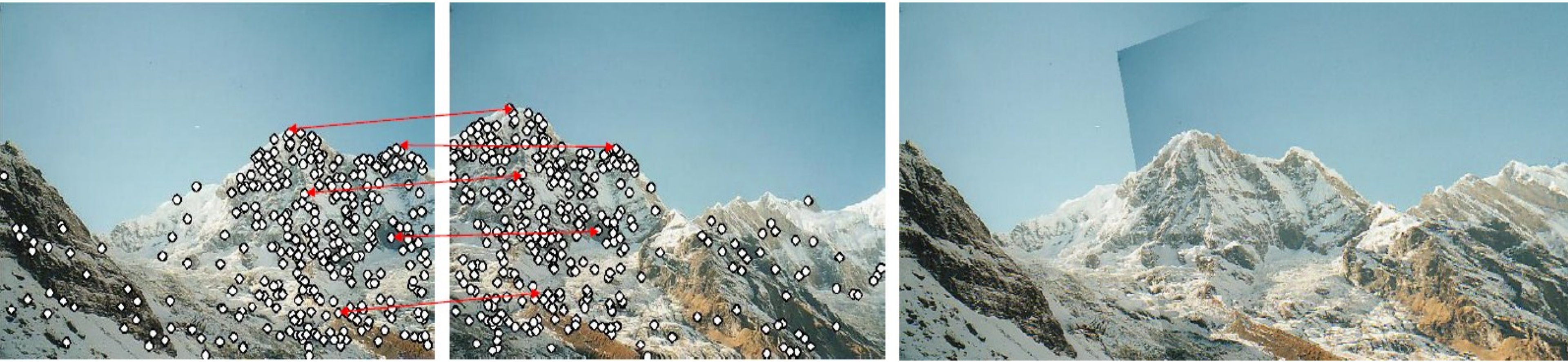
$$G_\sigma(x, y) = G_\sigma(x) G_\sigma(y)$$

$$\nabla S = \nabla (G_\sigma * I) = \begin{bmatrix} \frac{\partial (G_\sigma * I)}{\partial x} \\ \frac{\partial (G_\sigma * I)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial G_\sigma}{\partial x} * I \\ \frac{\partial G_\sigma}{\partial y} * I \end{bmatrix} = \begin{bmatrix} G'_\sigma(x) G_\sigma(y) * I \\ G_\sigma(x) G'_\sigma(y) * I \end{bmatrix}$$

- Discard pixels with $|\nabla S|$(i.e. edge strength), below a certain below
- **Non-maximal suppression:** identify local maxima of $|\nabla S|$ detected edges



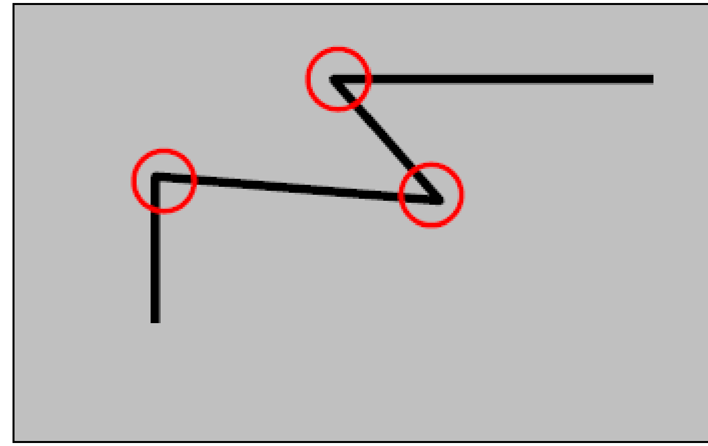
Point Features: Combining Images



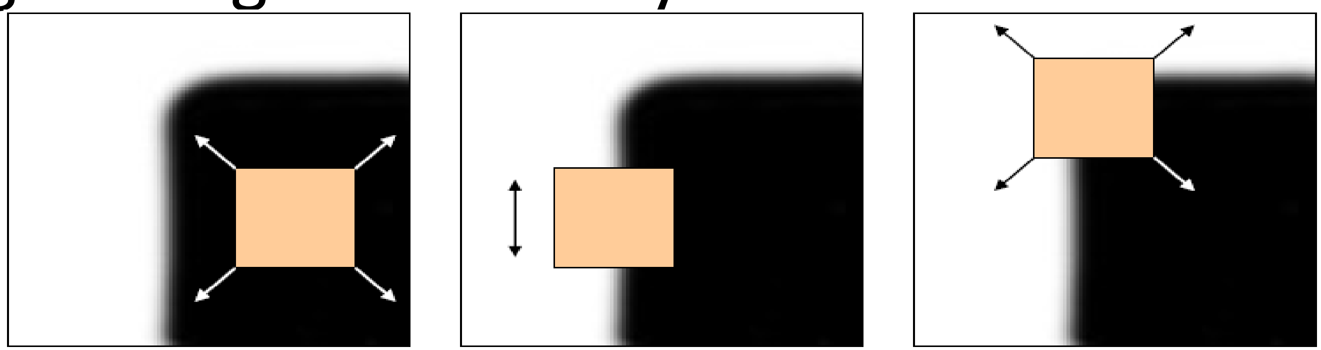
- Detect corresponding points across images in order to align them
  - Detect the same points independently in different images (Repeatable detector)
  - Identify the correct correspondence of each point (Reliable and Unique descriptor)
- Point features used in robot navigation, object/place recognition, 3D reconstruction

Harris corner detection

[Harris and Stephens, Alvey Vision Conference 1988]



- How do we identify corners?
- Key: around a corner, the image gradient has two or more dominant directions
- Shifting a window in any direction should give a large change in intensity in at least 2 directions



"flat" region:
no change in all directions

"edge":
no change along the edge direction

"corner":
significant change in all directions

Implementation

- Two image patches of size P one centered at (x, y) and one centered at $(x + \Delta x, y + \Delta y)$ the similarity measures between them is defined by sum squared error

$$SSD(\Delta x, \Delta y) = \sum_{x,y \in P} \left(I(x,y) - I(x + \Delta x, y + \Delta y) \right)^2$$

Let $I_x = \frac{\partial I(x,y)}{\partial x}$ and $I_y = \frac{\partial I(x,y)}{\partial y}$. Approximating $I(x + \Delta x, y + \Delta y)$

$$I(x + \Delta x, y + \Delta y) \approx I(x,y) + I_x(x,y)\Delta x + I_y(x,y)\Delta y$$

which results into

$$SSD(\Delta x, \Delta y) \approx \sum \left(I_x(x,y)\Delta x + I_y(x,y)\Delta y \right)^2 \approx [\Delta x \quad \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

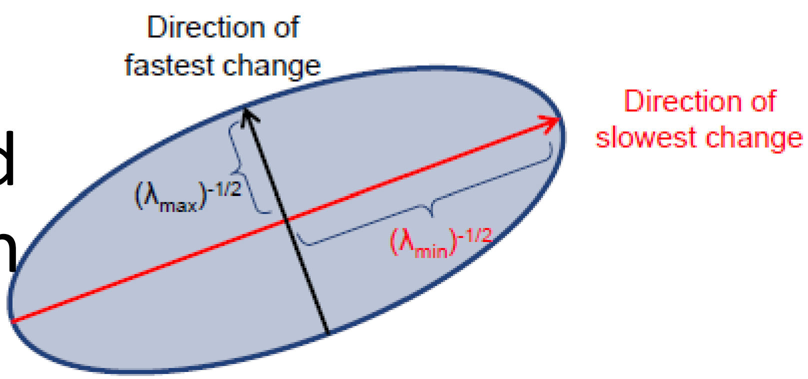
Implementation (Cont.)

- **M** is the "second moment matrix"
- Since **M** is symmetric with Eigen values λ_1 *and* λ_2
- The **Harris** detector analyses λ_1 *and* λ_2 to decide if we are in presence of a corner or not
- Visualize **M** as an ellipse with axis-lengths determined by λ_1 *and* λ_2 and orientation determined by **R**

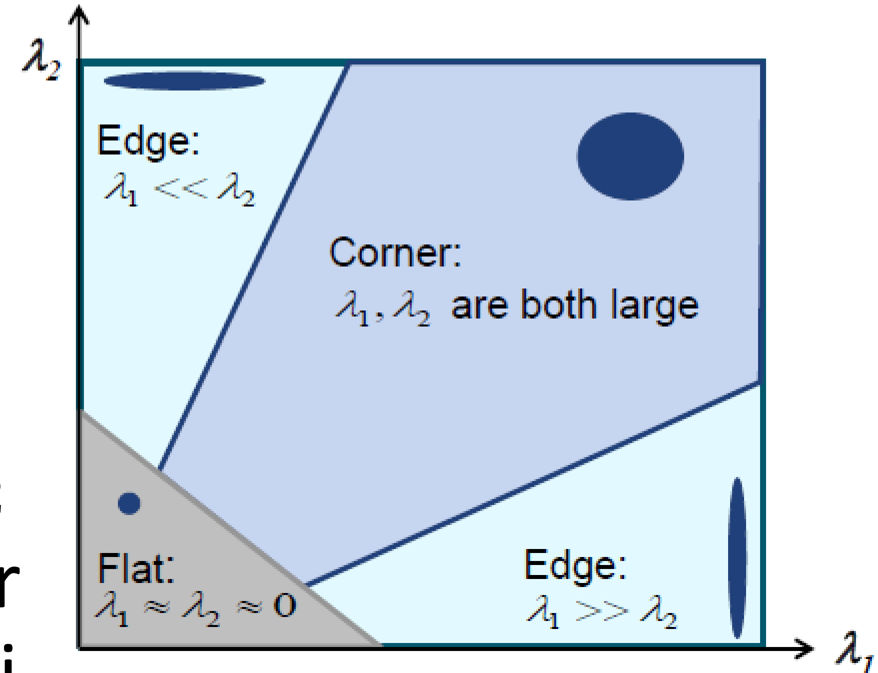
$$[\Delta x \quad \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$



Corner Response Function

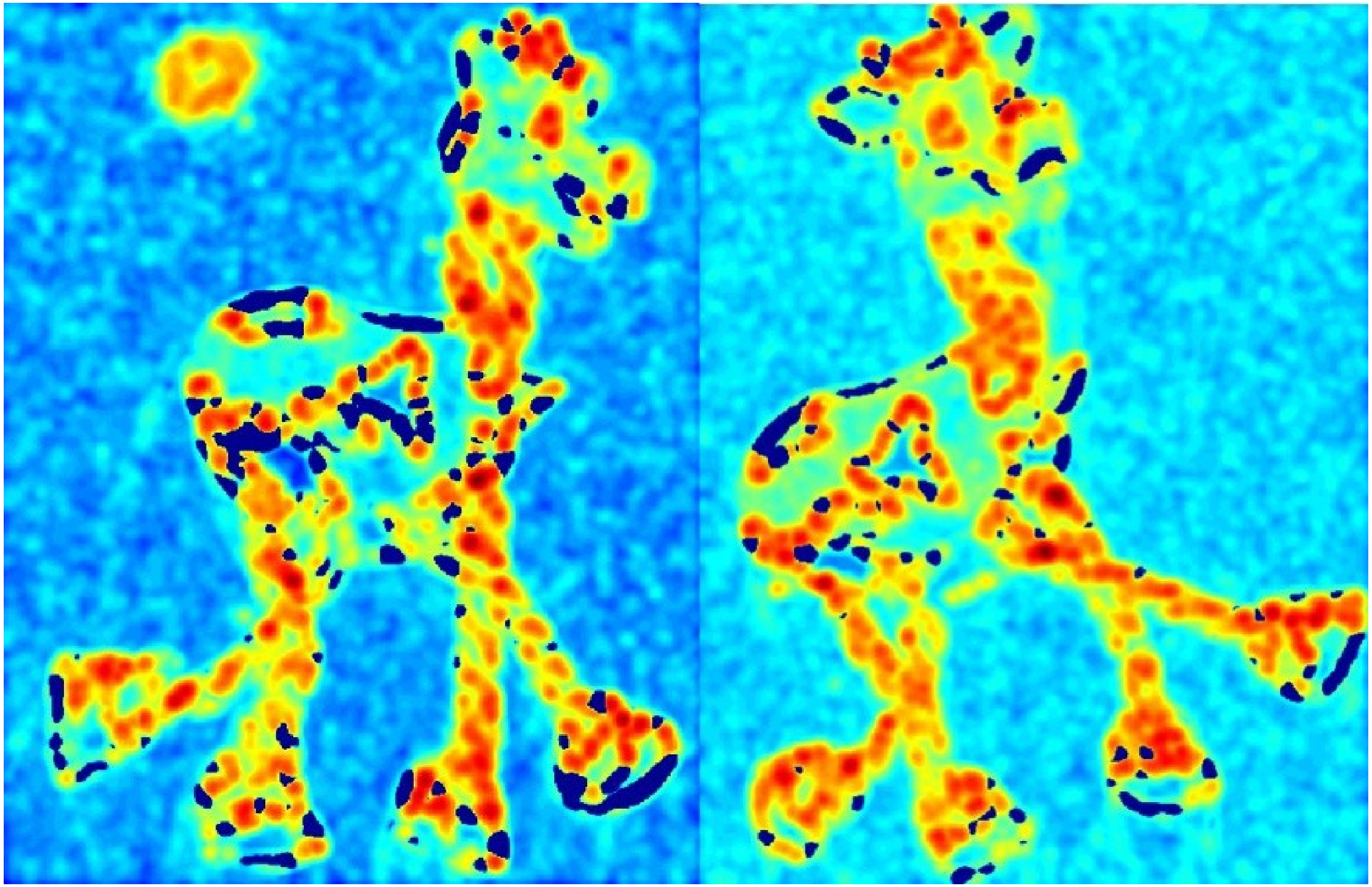


- Does the patch describe a corner or not?
  - No structure: $\lambda_1 = \lambda_2 = 0$
  - 1D structure: $\lambda_1 \gg \lambda_2$
  - 2D structure: Large λ_1, λ_2
- Last step of Harris corner detector: extract local minima of the cornerness function (Computation of λ_1, λ_2 *is expensive*) where $\kappa = [0.04 - 0.15]$

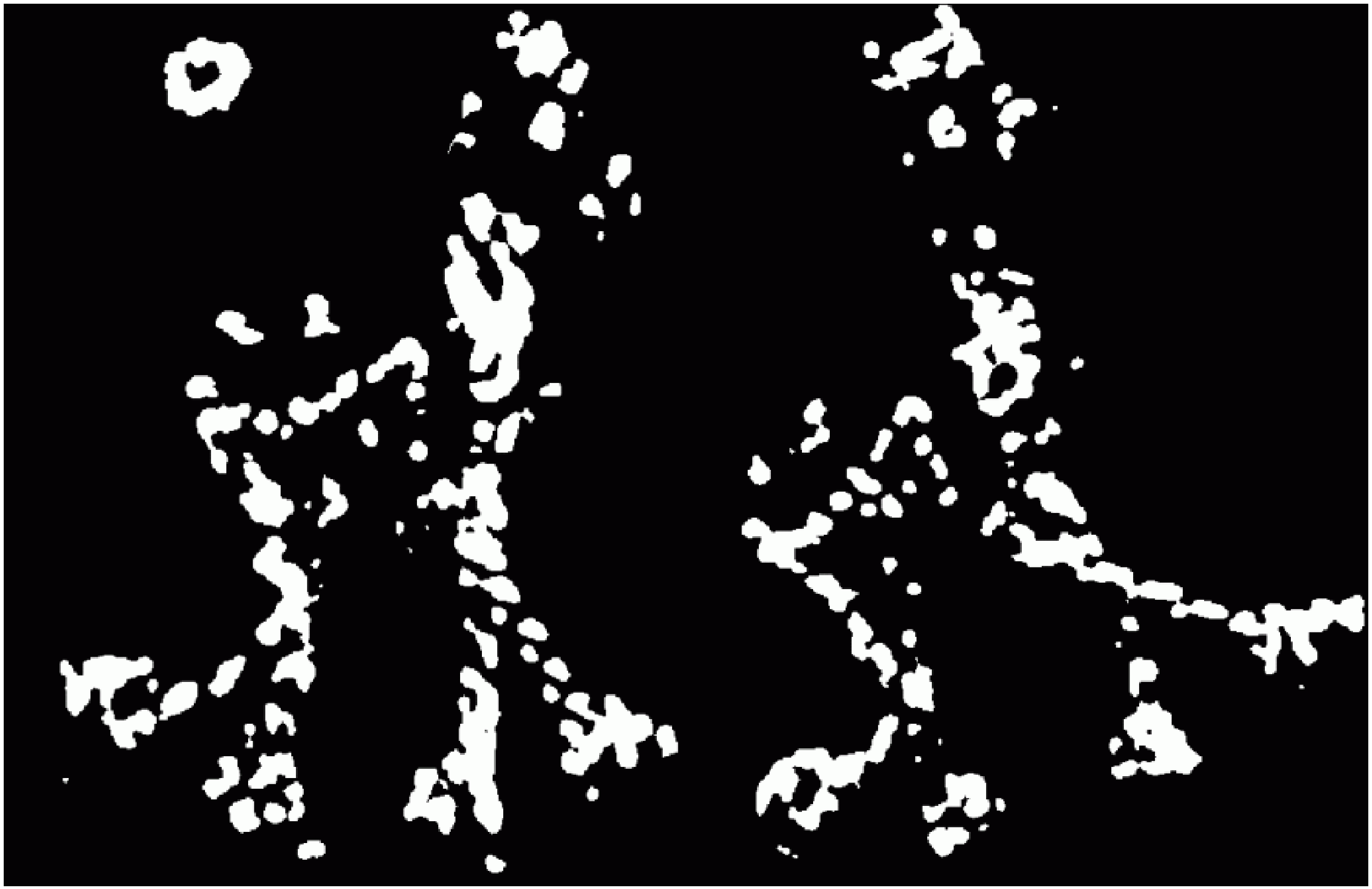
$$C = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = det(M) - \kappa \cdot trace^2(M)$$

Harris Corner Detector: Workflow

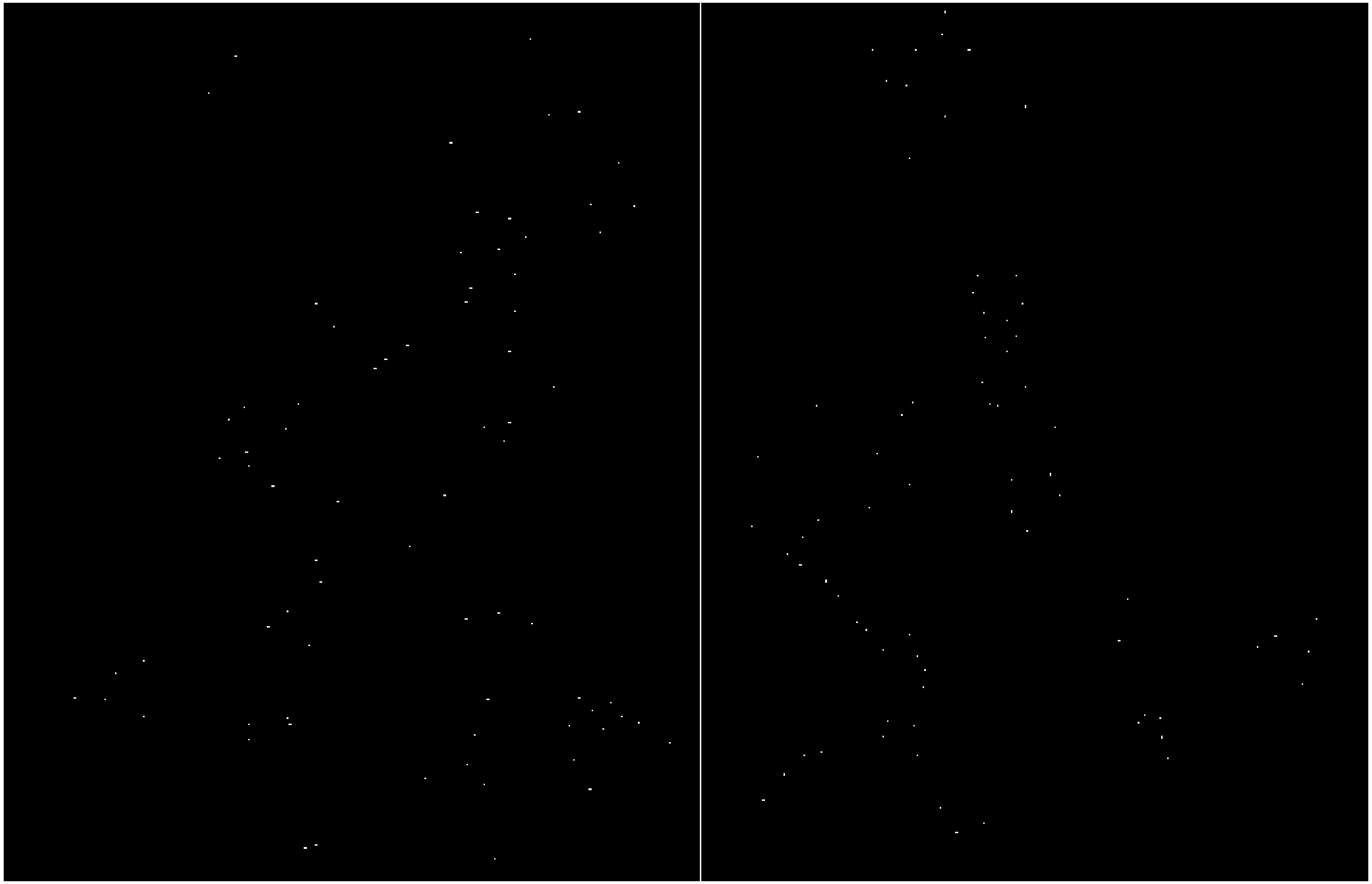
Workflow: Compute Corner Response **R**



Workflow:Find points with large corner response: *R*> threshold



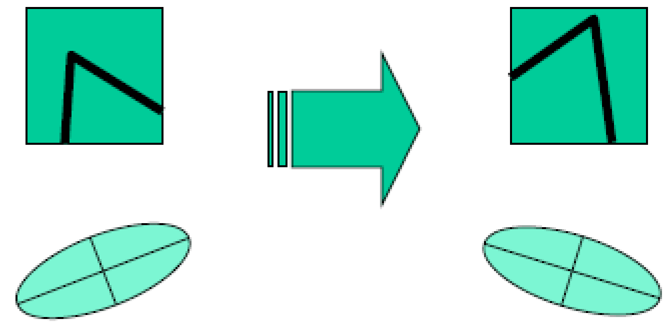
Workflow: Take only the points of local maxima of *R*



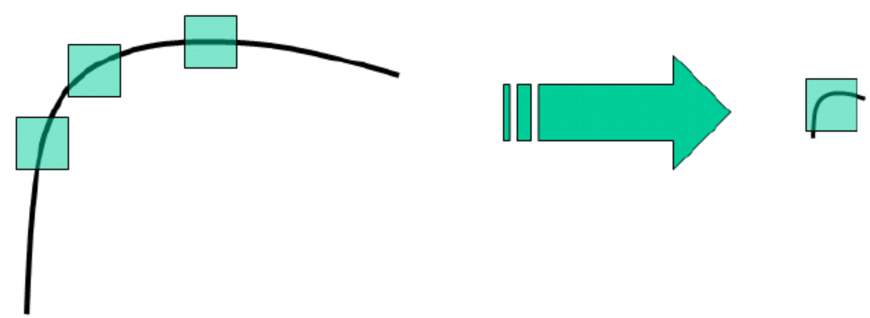
Detected Points

Properties of Harris Corner Detector

- Harris detector: probably the most widely used & known corner detect

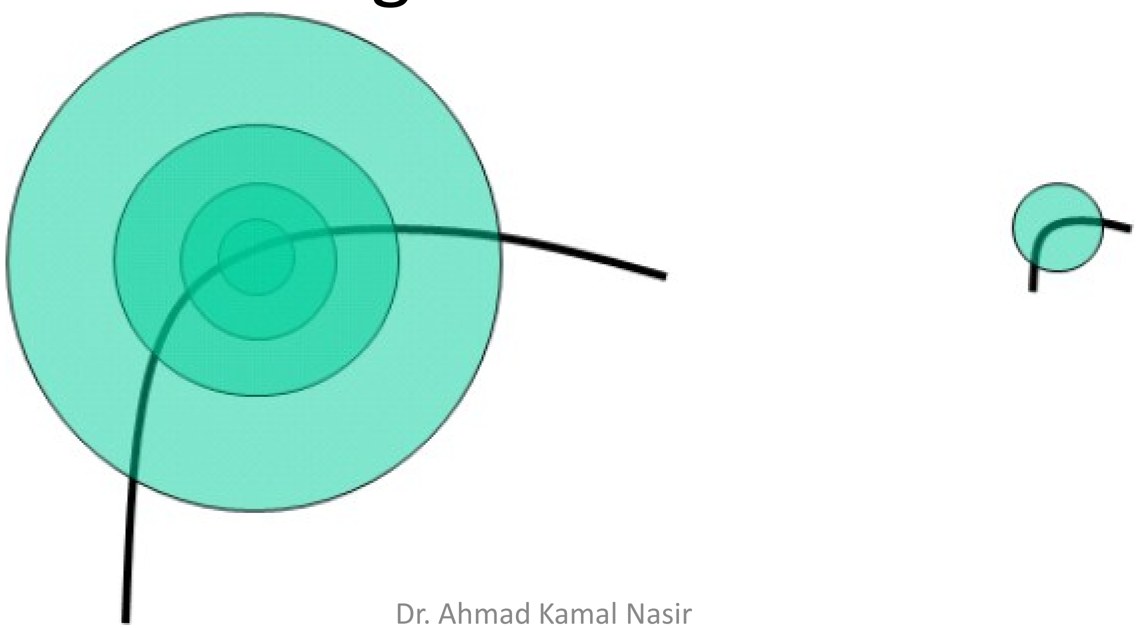


- The detection is invariant to
  - Rotation
  - Linear intensity changes
- The detection is NOT invariant to
  - **Scale changes**



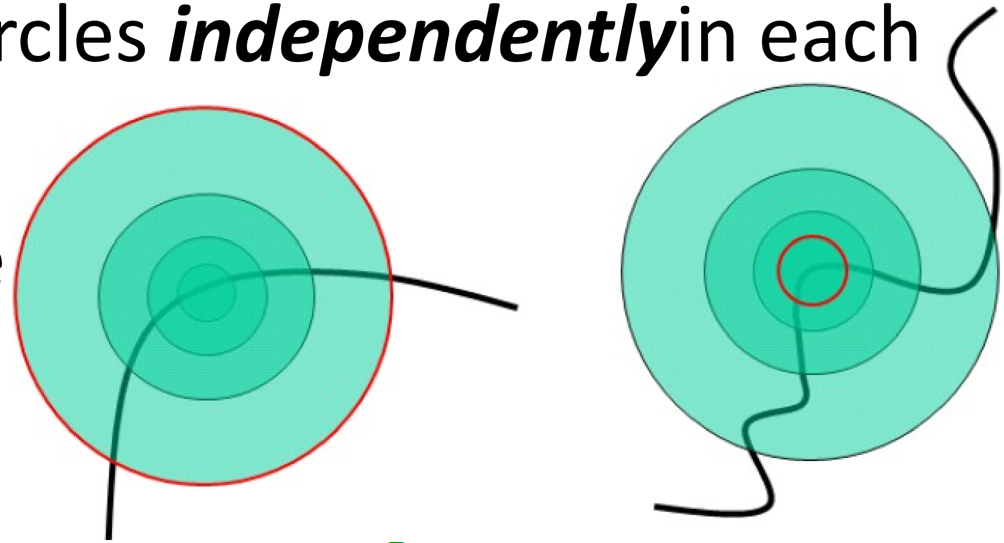
Scale Invariant Detection

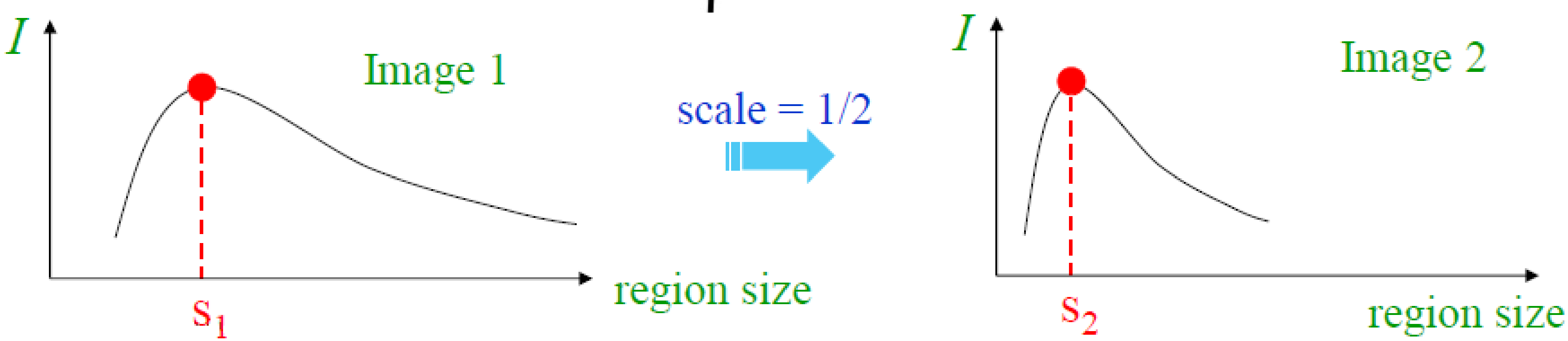
- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

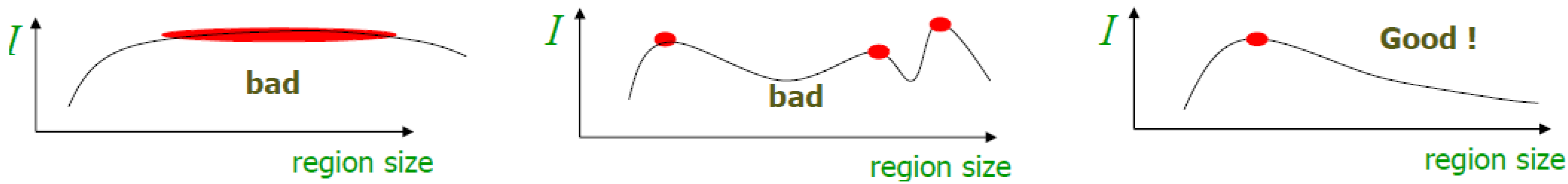
- The problem: how do we choose corresponding circles ***independently*** in each image?
- Intensity average of region





Scale Invariant Detection

- Design a function on the region (circle), which is “scale invariant” (the same for corresponding regions, even if they are at different scales)



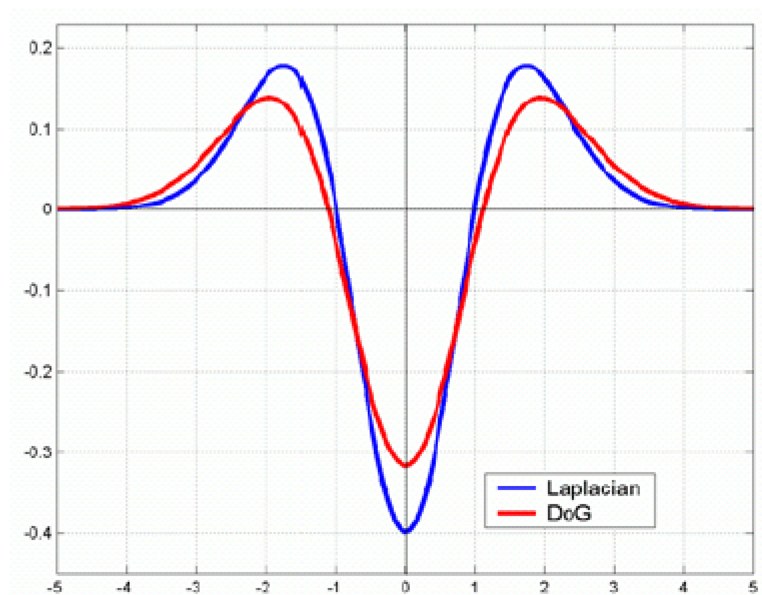
Kernels:

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

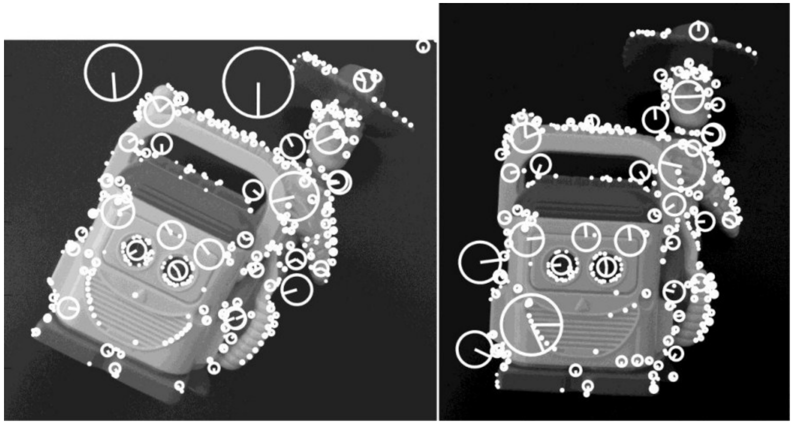
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



SIFT Features

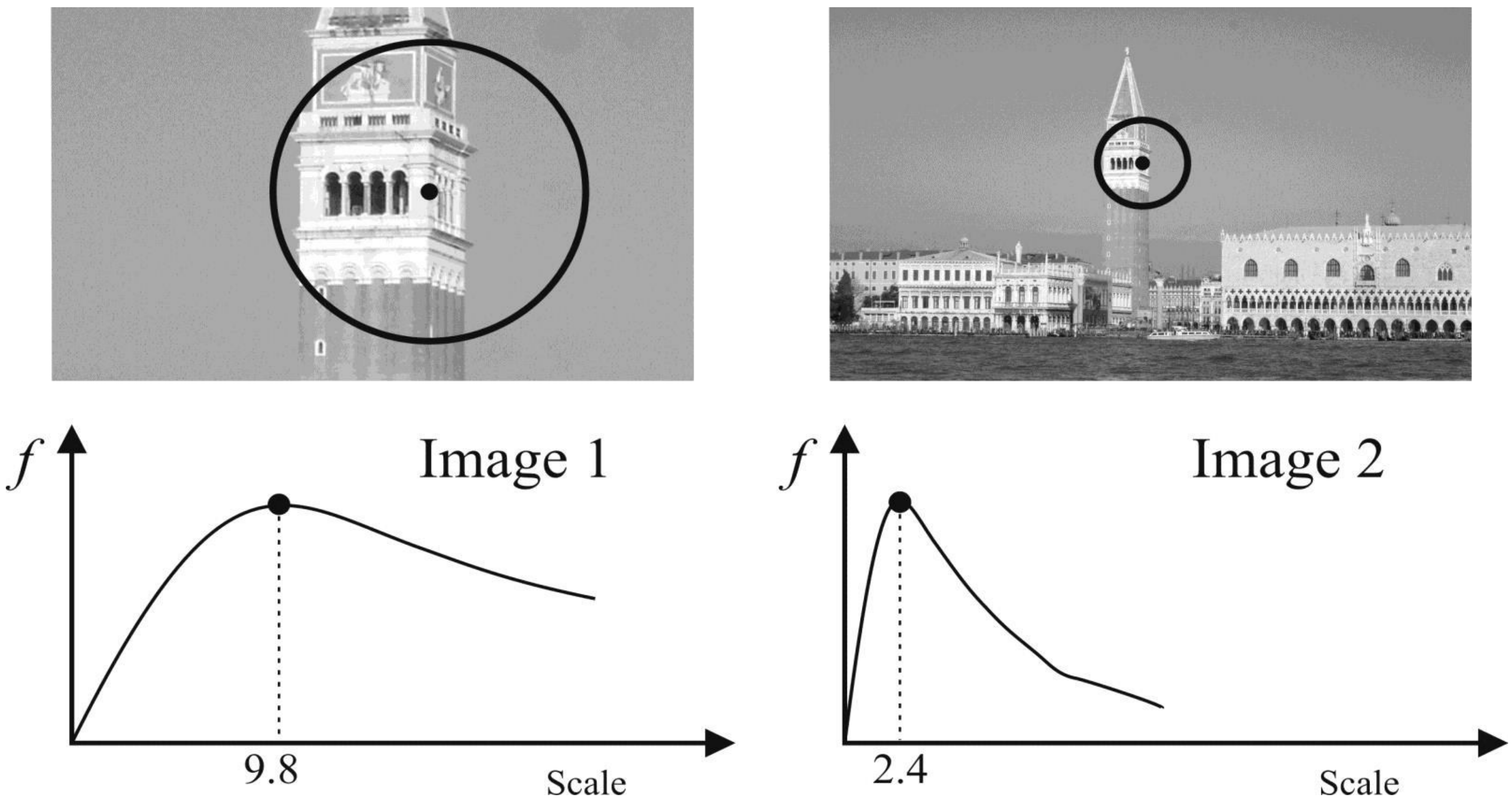
[Lowe, IJCV 2004]

- SIFT: Scale Invariant Feature Transform
- SIFT features are reasonably invariant to changes in: **rotation**, **scaling**, small changes in **viewpoint**, **illumination**
- Very powerful in capturing + describing **distinctive** structure, but also **computationally demanding**
- **Main SIFT stages:**
  - Extract keypoints + scale
  - Assign keypoint orientation
  - Generate keypoint descriptor



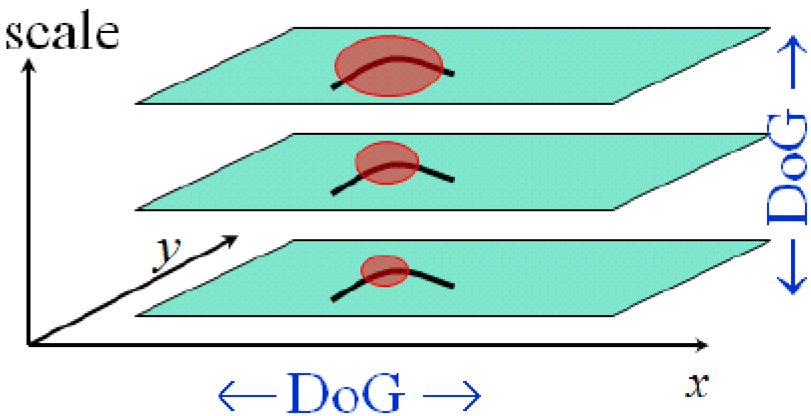
SIFT

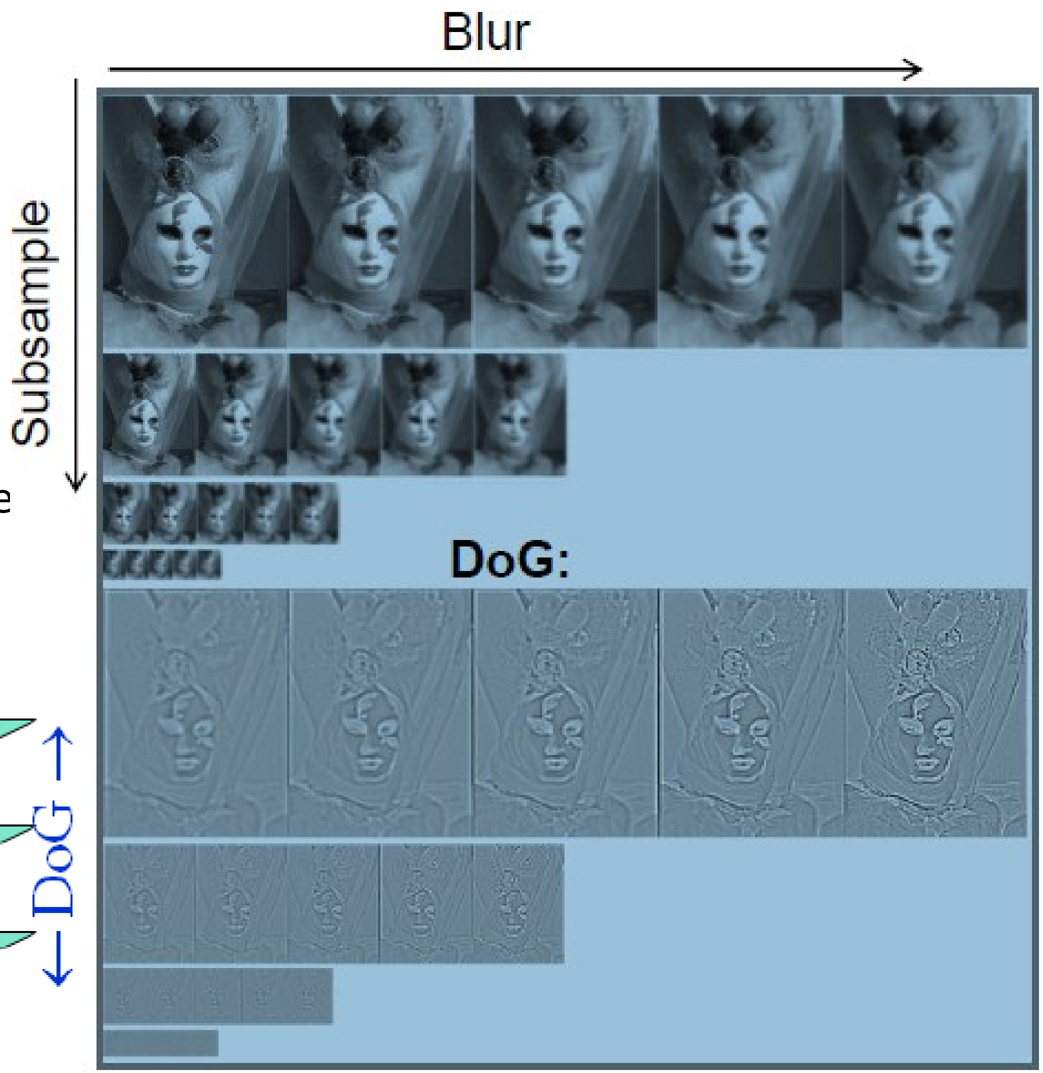
- Response of LoG for corresponding regions



Extract keypoints + scale

- Keypoint detection
  - Scale-space pyramid: subsample and blur original image
  - Difference of Gaussians (DoG) pyramid: subtract successive smoothed image
  - Keypoints: local extrema in the DoG pyramid





SIFT orientation and descriptor

- **Keypoint orientation** (to achieve **rotation invariance**)
  - Sample intensities around the keypoint
  - Compute a histogram of orientations of intensity gradients
  - **Keypoint orientation = histogram peak**

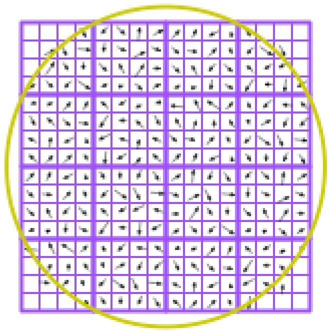
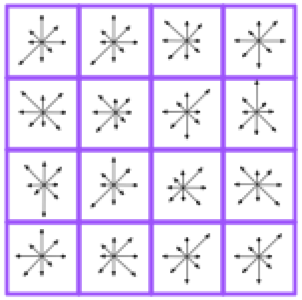


Image gradients

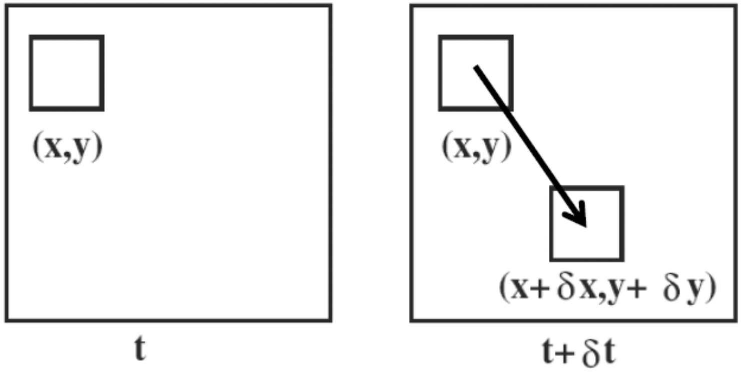
- **Keypoint descriptor**
  - SIFT descriptor: 128-long vector
  - Describe all gradient orientations **relative to the Keypoint Orientati**
  - Divide keypoint neighborhood in 4×4 regions & compute orientatio histograms along 8 directions
  - SIFT descriptor: concatenation of all 4×4×8 (=128) values



Keypoint descriptor

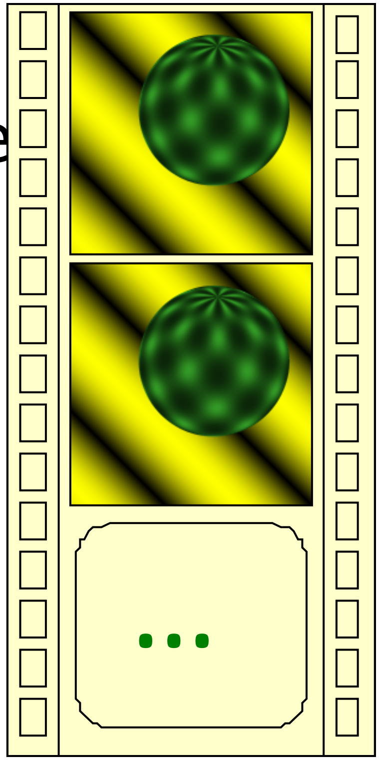
Optical Flow

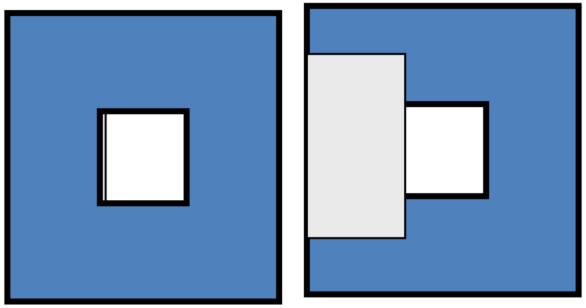
- Optical flow is an approximation of the apparent motion of objects within an image.
- Algorithms used to calculate optical flow attempt to find correlations between near frames in a video, generating a vector field showing where each pixel or region in the original image moved to in the second image.
- Typically the motion is represented as vectors originating or terminating at pixels in a digital image sequence.
- Estimating the optical flow is useful in pattern recognition, computer vision, and other image processing applications
- It computes the motion vectors of all pixels in the image (or a subset of them to be faster)



Apparent Motion

- Apparent motion of objects on the image plane
- Caution required!!
  - Consider a perfectly uniform sphere that is rotating but no change in the light direction
    - Optic flow is zero
  - Perfectly uniform sphere that is stationary but the light is changing
    - Optic flow exists
- Aperture problem





Optic Flow Computation

- Two strategies for computing motion
  - Differential Methods
    - Spatio temporal derivatives for estimation of flow at every position
    - Multi-scale analysis required if motion not constrained within a small range
      - Dense flow measurements
  - Matching Methods
    - Feature extraction(Image edges, corners)
    - Feature/Block Matching and error minimization
      - Sparse flow measurements

Optic Flow Computation (Cont.)

- Image Brightness Constancy assumption
  - Let I be the image intensity as captured by the camera
  - Using Taylor series to expand I

$$I\left(x+\Delta x, y+\Delta y, t+\Delta t\right)=I\left(x, y, t\right)+\frac{\partial I}{\partial x}\Delta x+\frac{\partial I}{\partial y}\Delta y+\frac{\partial I}{\partial t}\Delta t$$

$$\underset{\Delta t\to 0}{Lt}\frac{I\left(x+\Delta x, y+\Delta y, t+\Delta t\right)-I\left(x, y, t\right)}{\Delta t}=\underset{\Delta t\to 0}{Lt}\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t}+\frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t}+\frac{\partial I}{\partial t}$$

  - Apparent brightness of moving objects remains constant

$$\frac{\partial I}{\partial x}\frac{dx}{dt}+\frac{\partial I}{\partial y}\frac{dy}{dt}+\frac{\partial I}{\partial t}=\frac{dI}{dt}=0$$

Optic Flow Computation (Cont.)

- Image Brightness Constancy assumption
  - Apparent brightness of moving objects remains constant

$$\frac{\partial I}{\partial x}\frac{dx}{dt}+\frac{\partial I}{\partial y}\frac{dy}{dt}+\frac{\partial I}{\partial t}=0$$

  - The $\left(\partial I/\partial x, \partial I/\partial y\right)=\nabla I$ are the image gradient while the $\left(dx/dt, dy/dt\right)=\mathbf{v}$ are the components of the motion field

$$\boxed{\left(\nabla I\right)^T \mathbf{v}+I_t=0}$$

Optic Flow Constraint

- How to get more equations for a pixel?
  - Basic idea: impose additional constraints
    - Most common is to assume that the flow field is smooth locally
  - One method: pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel!

$$\nabla I(\mathbf{p}_i) . [u \quad v] + I_t(\mathbf{p}_i) = 0$$

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$A_{25\times 2} d_{2\times 1} = b_{25\times 1}$$

Lucas-Kanade Optic Flow

- We now have more equations than unknowns

$$A_{25\times 2} d_{2\times 1} = b_{25\times 1} \Rightarrow \min \| Ad - b \|$$

- Solve the least squares problem
  - Minimum least squares solution (in d) is given by

$$\left(A^T A\right)_{2\times 2} d_{2\times 1} = \left(A^T\right)_{2\times 25} b_{25\times 1}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

  - First proposed by Lucas-Kanade in 1981
  - Summation performed over all the pixels in the window

Lucas-Kanade Optic Flow

- Lucas-Kanade Optic flow

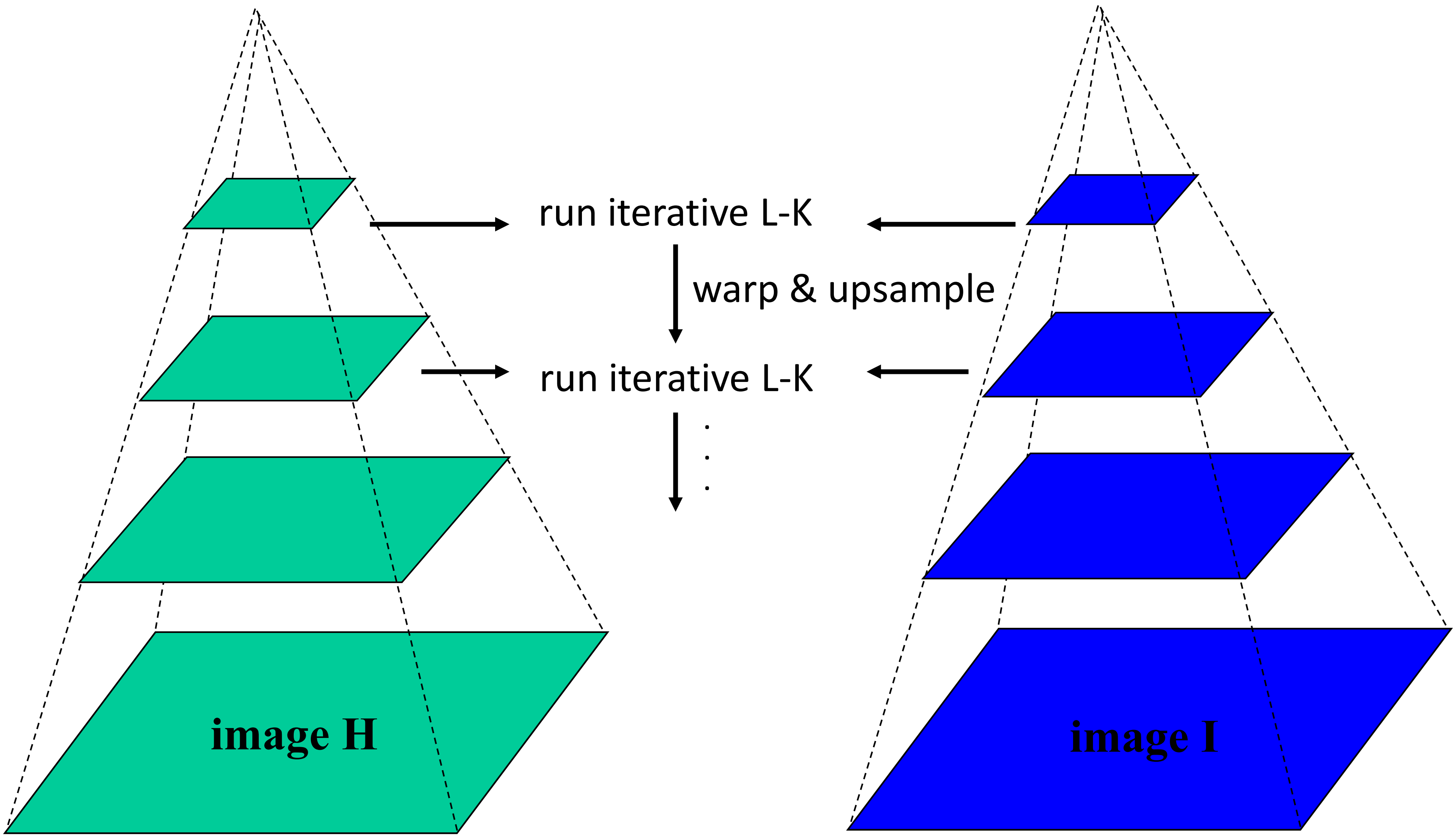
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- When is the Lucas-Kanade equations solvable
  - $A^T A$ should be invertible
  - $A^T A$ should not be too small (effects of noise)
    - Eigenvalues of $A^T A$, λ_1 and λ_2 should not be small
  - $A^T A$ should be well conditioned
    - λ_1/λ_2 should not be large (λ_1 = larger eigenvalue)

Improving the Lucas-Kanade method

- When our assumptions are violated
  - Brightness constancy is not satisfied
  - The motion is not small
  - A point does not move like its neighbors
- Iterative Lucas-Kanade Algorithm
  - Estimate velocity at each pixel by solving Lucas-Kanade equations
  - Warp H towards I using the estimated flow field
    - use image warping techniques
  - Repeat until convergence

Iterative Lucas-Kanade method



Gaussian pyramid of image H **Gaussian pyramid of image I**

Summary

- Visual Odometry
  - Camera model
  - Calibration
- Feature detection
  - Harris corners
  - SIFT/SURF etc.
- Optical Flow
  - Kanade-Lucas-Tomasi Tracker

Questions