CSE 152A: Computer Vision

Manmohan Chandraker

Lecture 6: Edges and Corners

Overall goals for the course

- Introduce fundamental concepts in computer vision
- Enable one or all of several such outcomes
  - Pursue higher studies in computer vision
  - Join industry to do cutting-edge work in computer vision
  - Gain appreciation of modern computer vision technologies
- Engage in discussions and interaction
- This is a great time to study computer vision!

Course Details

Course details

- Class webpage:
  – https://cseweb.ucsd.edu/~mkchandraker/classes/CSE152A/Winter2024/
- Instructor email:
  – mkchandraker@ucsd.edu
- Grading
  – 35% final exam
  – 40% homework assignments
  – 20% mid-term
  – 5% self-study exercise
  – Ungraded quizzes
- Aim is to learn together, discuss and have fun!

Course details

- TAs
  - Nicholas Chua: nchua@ucsd.edu
  - Tarun Kalluri: sskallur@ucsd.edu
  - Sreyas Ravichandran: srravichandran@ucsd.edu

- Tutors
  - Kun Wang, Kevin Chan, Zixian Wang: {kuw010, tsc003, ziw081}@ucsd.edu

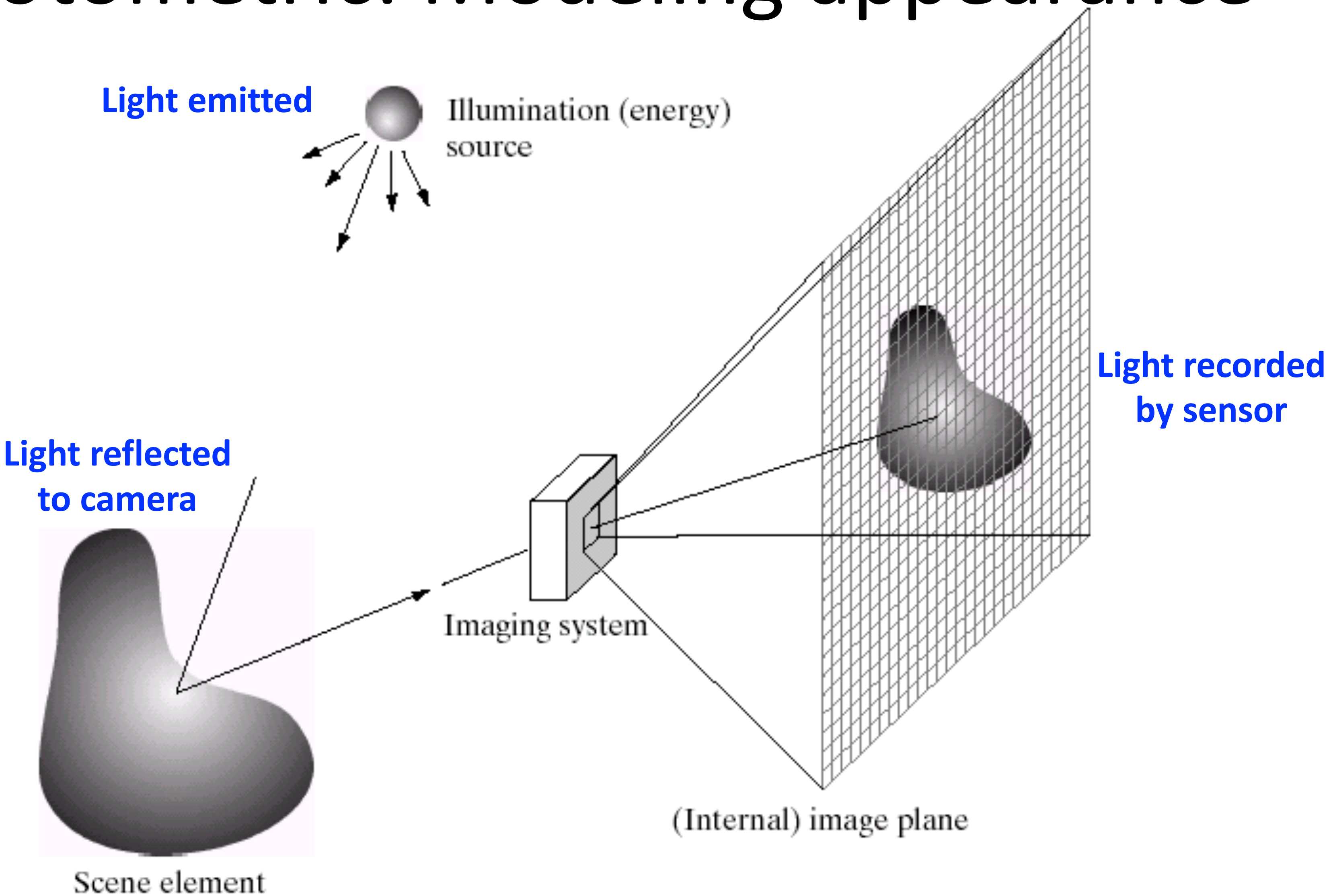
- Discussion section: M 3-3:50pm

- TA office hours and tutor hours to be posted on webpage

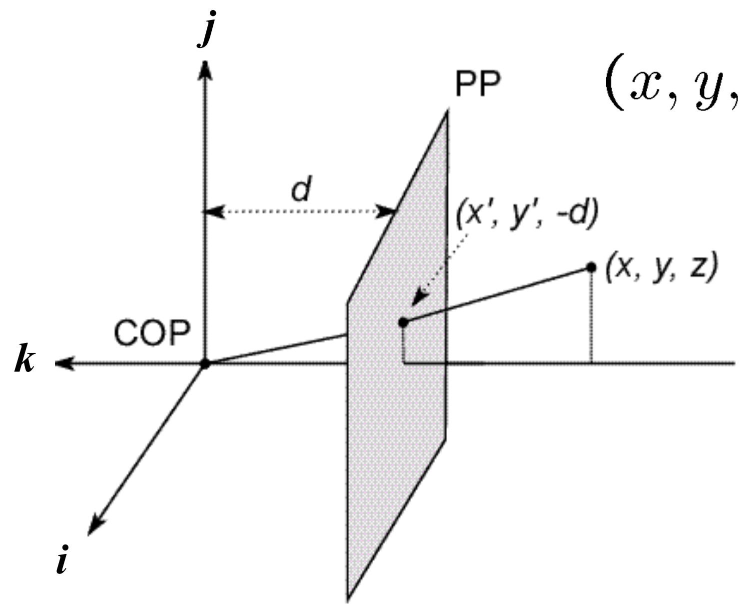
- Piazza for questions and discussions:
  - https://piazza.com/ucsd/winter2024/cse152a

Recap

Photometric: Modeling appearance



Geometric: Modeling projection



$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z})$$

- The coordinate system
  - We will use the pinhole model as an approximation
  - Put the optical center (**C**enter **O**f **P**rojection) at the origin
  - Put the image plane (**P**rojection **P**lane) in front of the COP
  - The camera looks down the negative z axis.

Geometric: Camera projection

- To project a point (*x*,*y*,*z*) in *world* coordinates into a camera
- First transform (*x*,*y*,*z*) into *camera* coordinates
- Need to know
  - Camera position (in world coordinates)
  - Camera orientation (in world coordinates)
- Then project into the image plane
  - Need to know camera *intrinsics*
- These can all be described with matrices.

Projection matrix

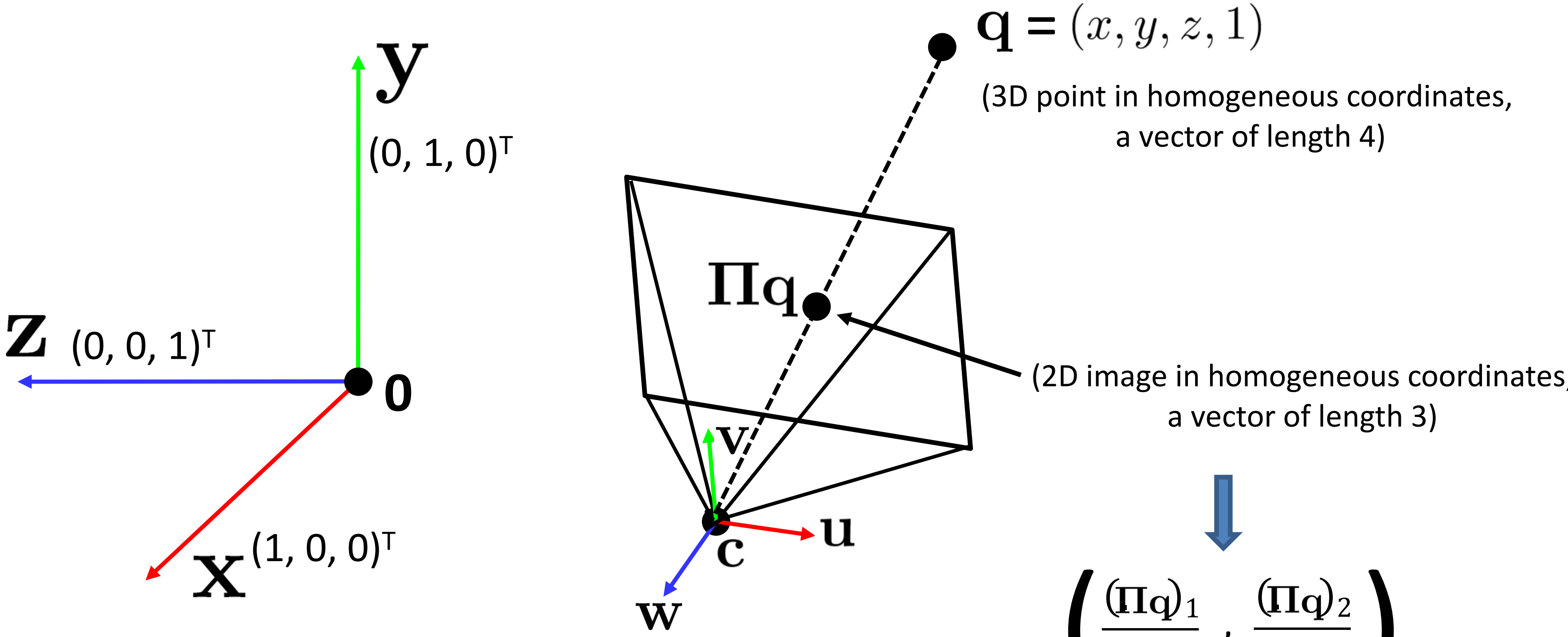
$$\mathbf{\Pi} = \underset{\text{intrinsics}}{\mathbf{K}} \underset{\text{projection}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}} \underset{\text{rotation}}{\begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}} \underset{\text{translation}}{\begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}}$$

$$\Downarrow$$

$$\mathbf{\Pi} = \mathbf{K} \left[\ \mathbf{R} \ \middle| \ -\mathbf{Rc} \ \right]$$

Denote this by **t.**

Projection matrix



$$\left(\frac{(\Pi \mathbf{q})_1}{(\Pi \mathbf{q})_3}, \frac{(\Pi \mathbf{q})_2}{(\Pi \mathbf{q})_3} \right)$$

(Usual Cartesian coordinates,
a vector of length 2)

Ideal points and the line at infinity

- Consider two parallel lines in the 2D image:

$$ax + by + c = 0$$
$$ax + by + c' = 0$$

- In homogeneous coordinates, the lines are:

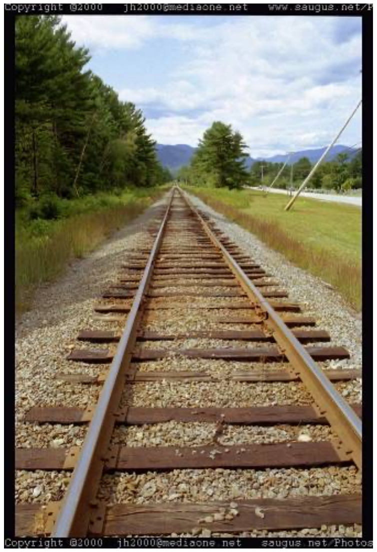
$$\mathbf{l} = (a, b, c)^\top \text{ and } \mathbf{l}' = (a, b, c')^\top$$

- Their point of intersection is given by:

$$\mathbf{x}_\infty = \mathbf{l} \times \mathbf{l}' = (c - c')(-b, a, 0)^\top \sim (-b, a, 0)^\top$$

- To de-homogeneize involves a division by 0
    - This is a point at “infinity”, called an ideal point

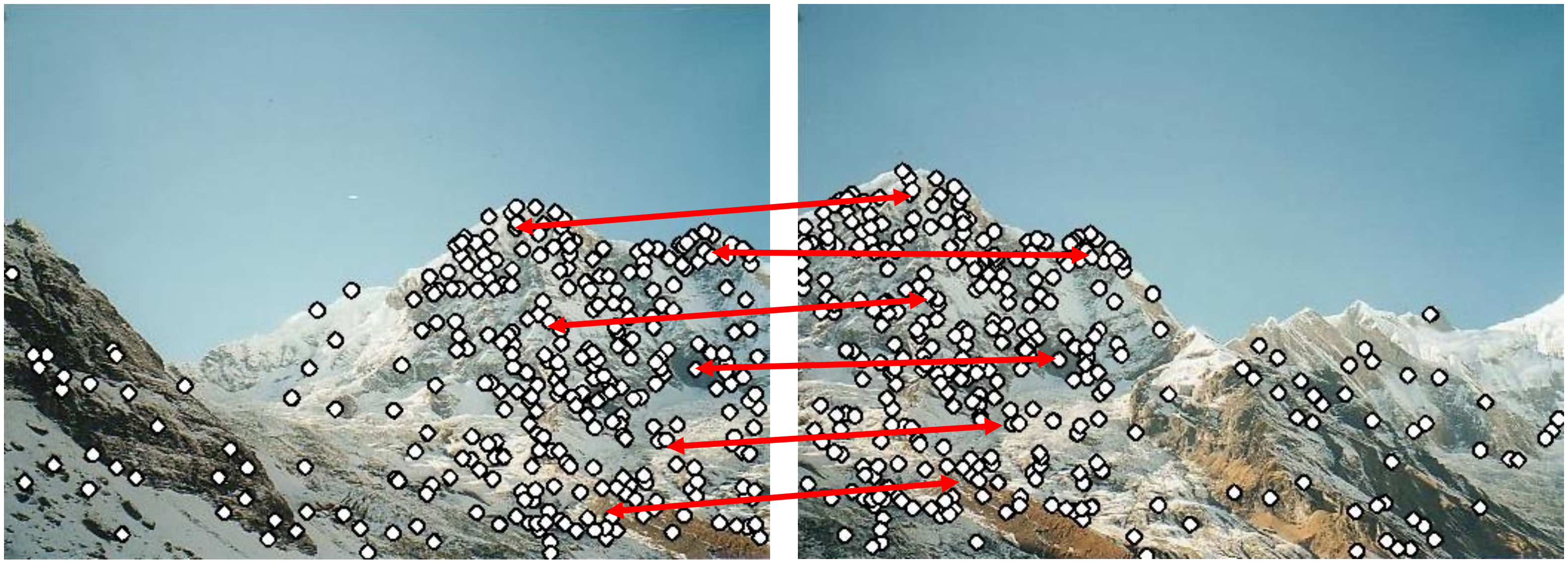
- Which line contains all ideal points $\mathbf{x}_\infty = (x, y, 0)^\top$?
    - Line at infinity: $\mathbf{l}_\infty = (0, 0, 1)^\top$.



This is a 2D image of 3D space, just for visualization. Equations here are for 2D space.

Correspondence estimation

- Motivation: panorama stitching



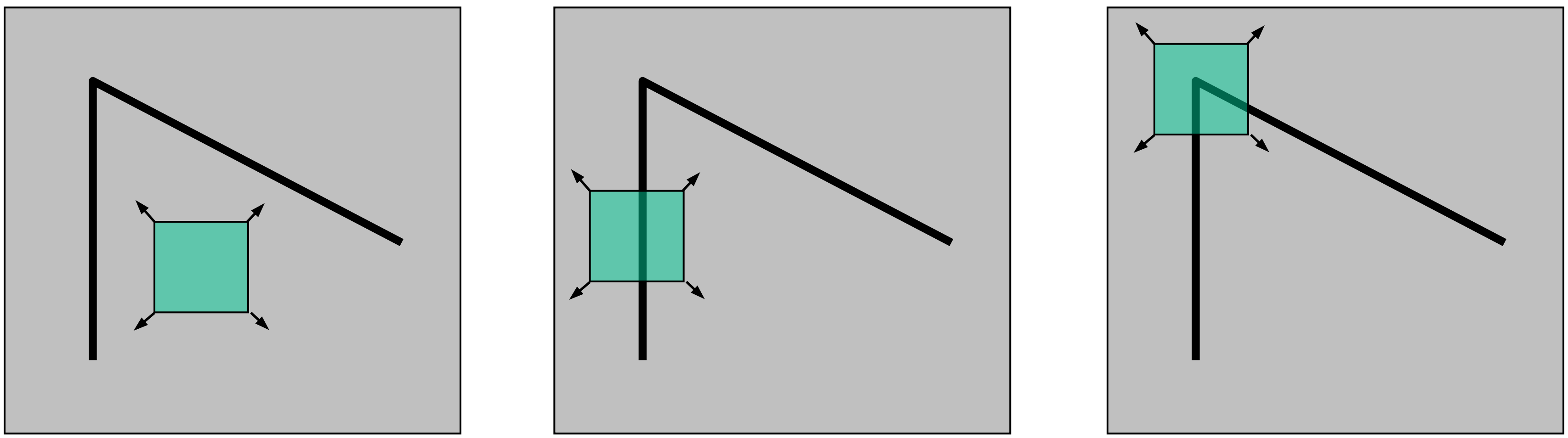
Extract features
Match features
Align images

[Images: Rick Szeliski]

Feature detection

Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in *some direction* causes a *big change*



"flat" region: no change in all directions

"edge": large change perpendicular to the edge direction

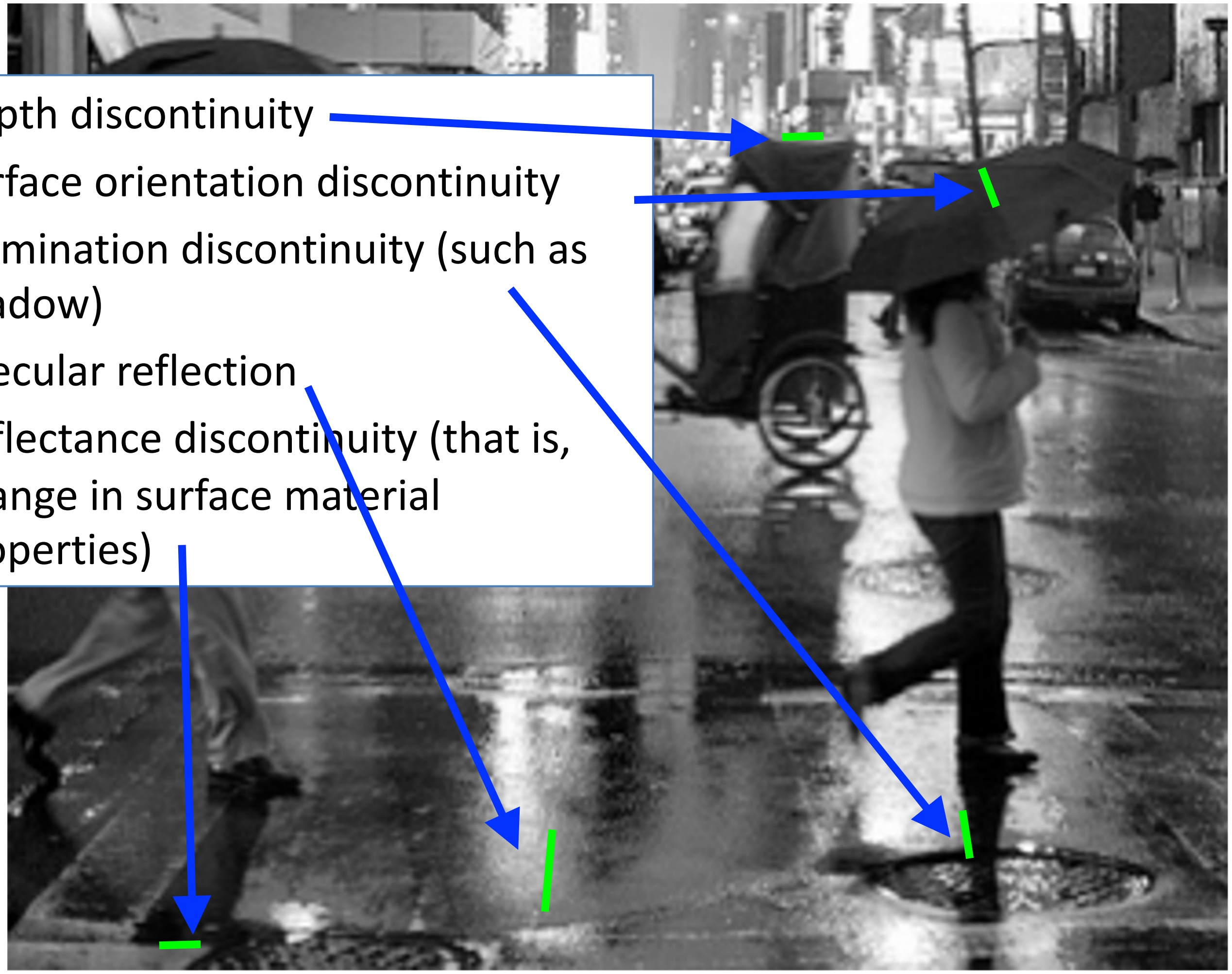
"corner": large change in all directions

[Darya Frolova, Denis Simakov, Weizmann Institute]

Edges

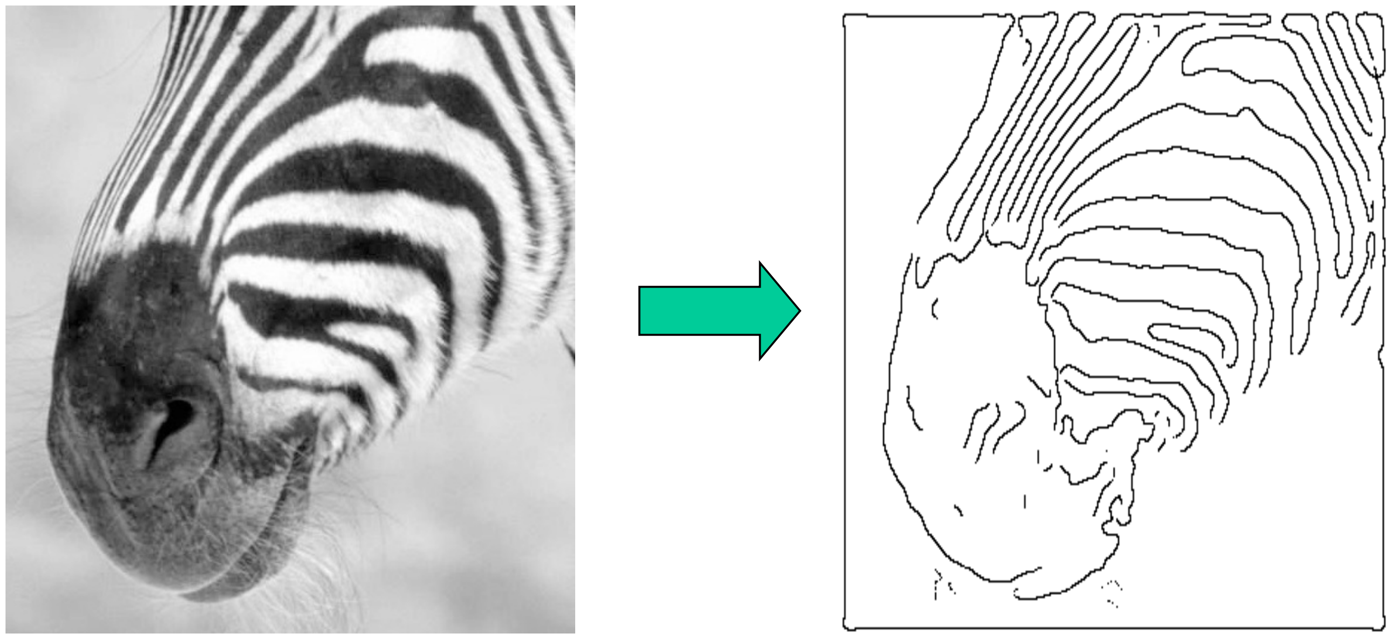
Edges in Natural Images

- Depth discontinuity
- Surface orientation discontinuity
- Illumination discontinuity (such as shadow)
- Specular reflection
- Reflectance discontinuity (that is, change in surface material properties)



Source: Photografr.com

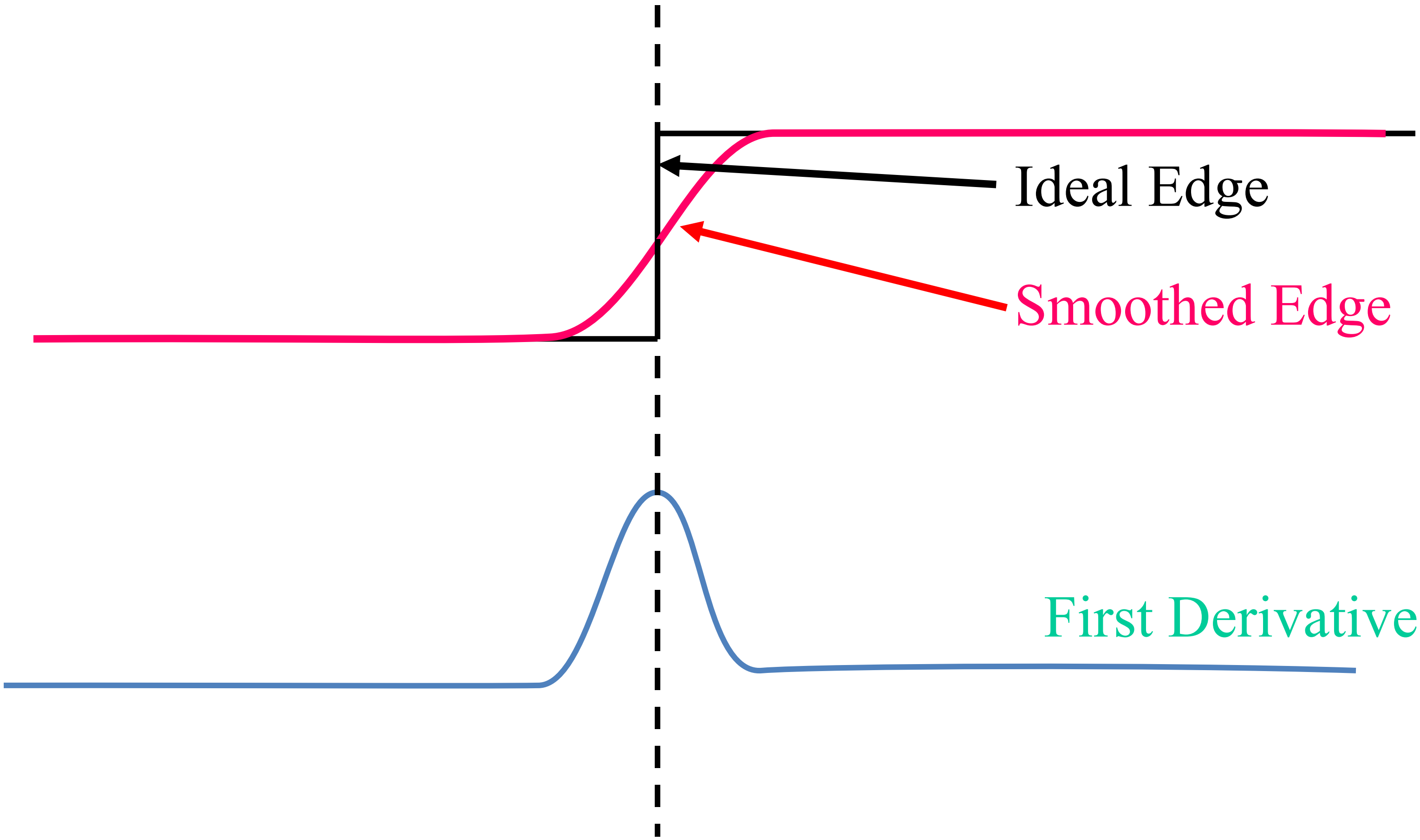
How Can We Find Edges?



Find regions where magnitude of gradient is large.

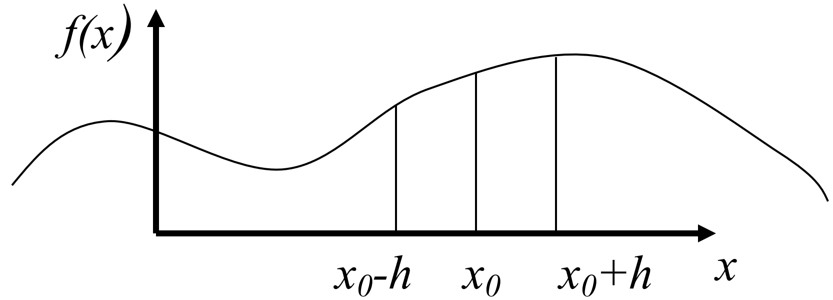
Edge is Where Change Occurs: 1-D

- Change is measured by derivative in 1D



- Biggest change: first derivative has maximum magnitude

Numerical Derivatives of Sampled Signal



Take Taylor series expansion of $f(x)$ about x_0

$$f(x) = f(x_0)+f'(x_0)(x-x_0) + \tfrac{1}{2} f''(x_0)(x-x_0)^2 + \dots$$

Consider samples taken at increments of h and first two terms

$$f(x_0+h) = f(x_0) + f'(x_0)h$$

$$f(x_0-h) = f(x_0) - f'(x_0)h$$

Rearranging the above two yields:

$$f'(x_0) = \frac{f(x_0+h) - f(x_0 - h)}{2h} = \begin{bmatrix} \frac{-1}{2h} & 0 & \frac{1}{2h} \end{bmatrix} . [\, f(x_0 - h) \;\; f(x_0) \;\; f(x_0+h)]$$

Numerical Derivatives

$$f'(x_0) = \frac{f(x_0+h) - f(x_0\ - h)}{2h} = \begin{bmatrix} \frac{-1}{2h} & 0 & \frac{1}{2h} \end{bmatrix} . [\ f(x_0 - h) \quad f(x_0) \quad f(x_0+h)]$$

- With images, units of *h* is pixels, so *h*=1
  - Compute x-derivative at (x_0, y_0):

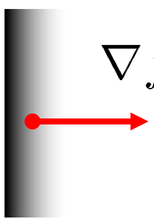
$$\frac{I(x_0+1, y_0) - I(x_0 - 1, y_0)}{2}$$

  - Compute x-derivative at (x_0, y_0):

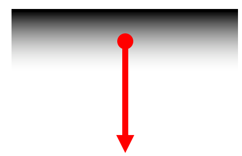
$$\frac{I(x_0, y_0+1) - I(x_0, y_0 - 1)}{2}$$

Edge Detection with Image Gradients

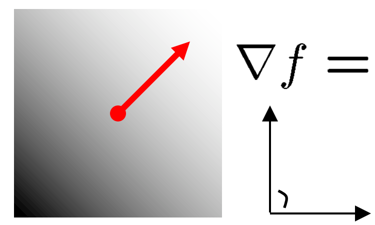
- Given a function f(x,y) , for example, image intensity is f
- Expression for gradient: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Represents direction of most rapid change in intensity



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



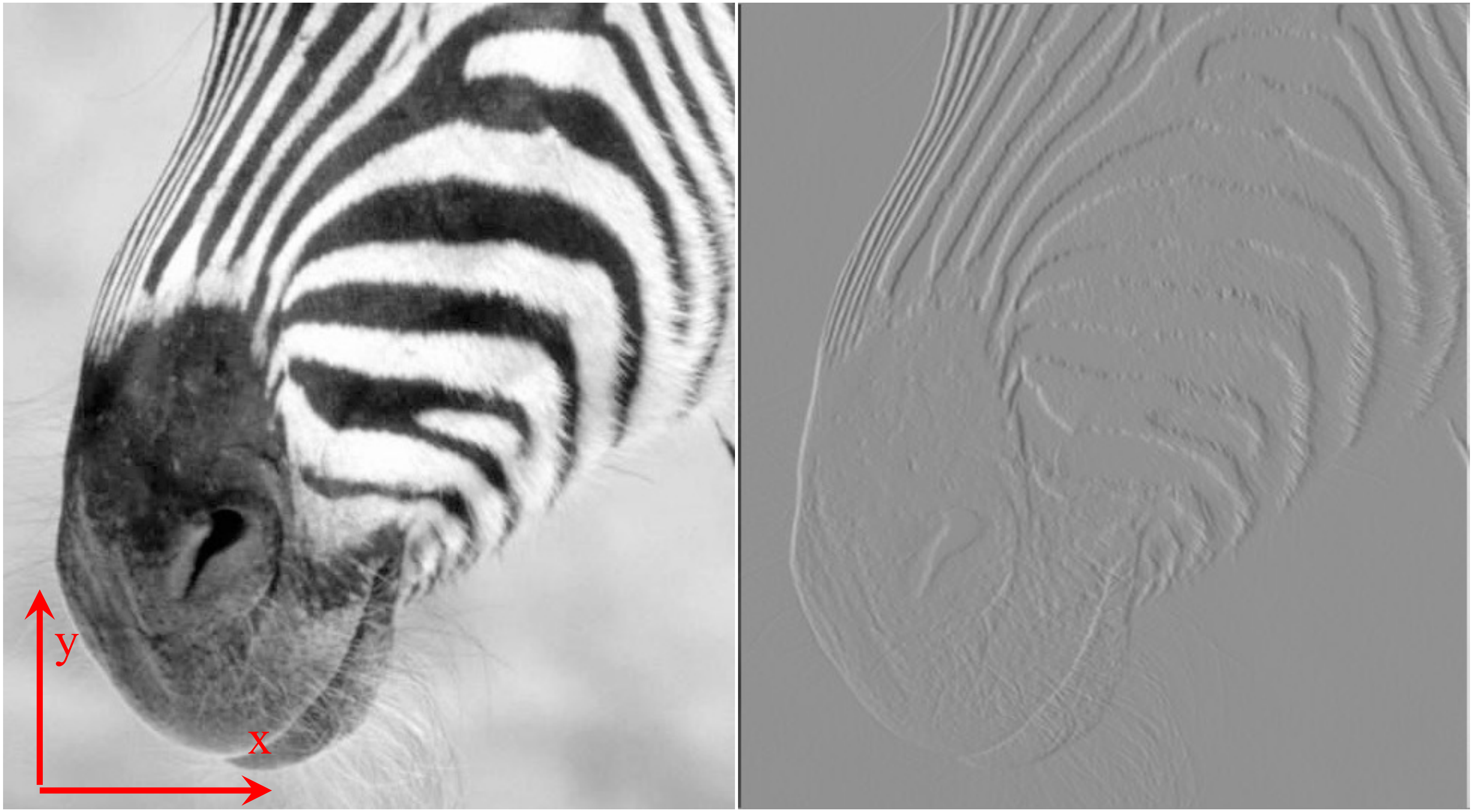
$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

- Gradient direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Finding derivatives

Is this dI/dx or dI/dy?



Convolution

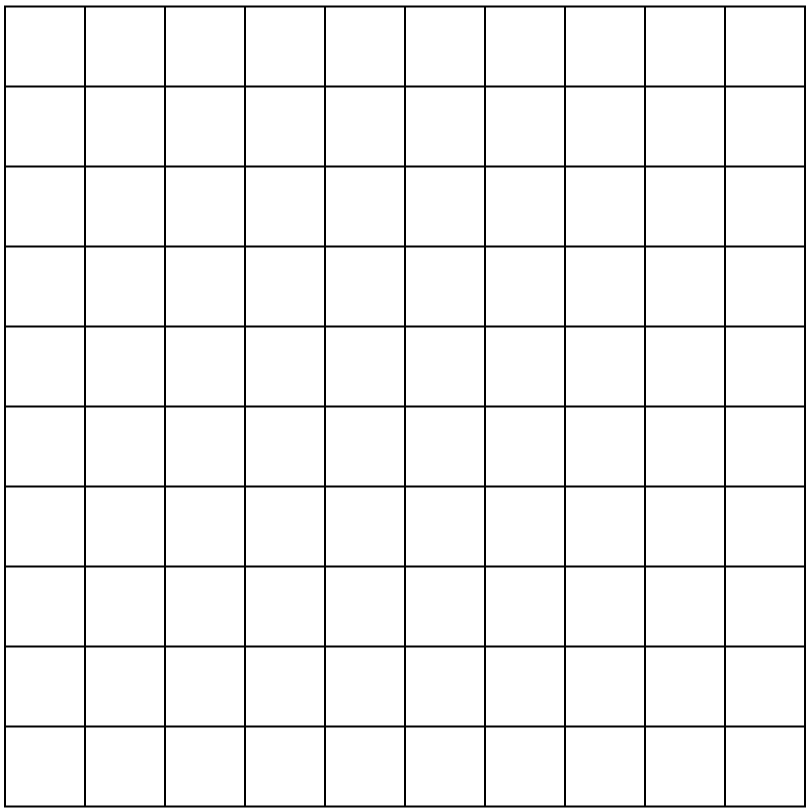


Image (I)

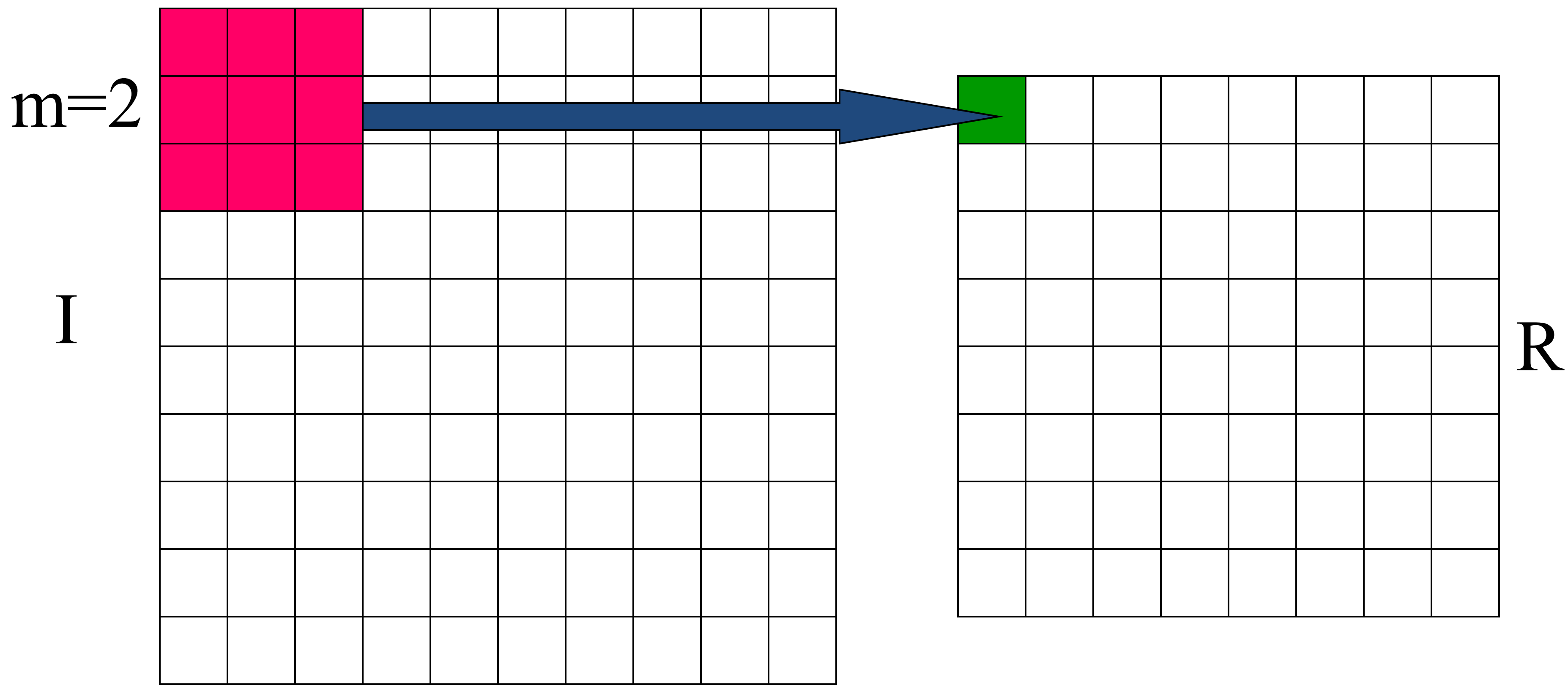
*

1	2	1
0	0	0
-1	-2	-1

Kernel (K)

Note: Typically, kernel is relatively small in vision applications.

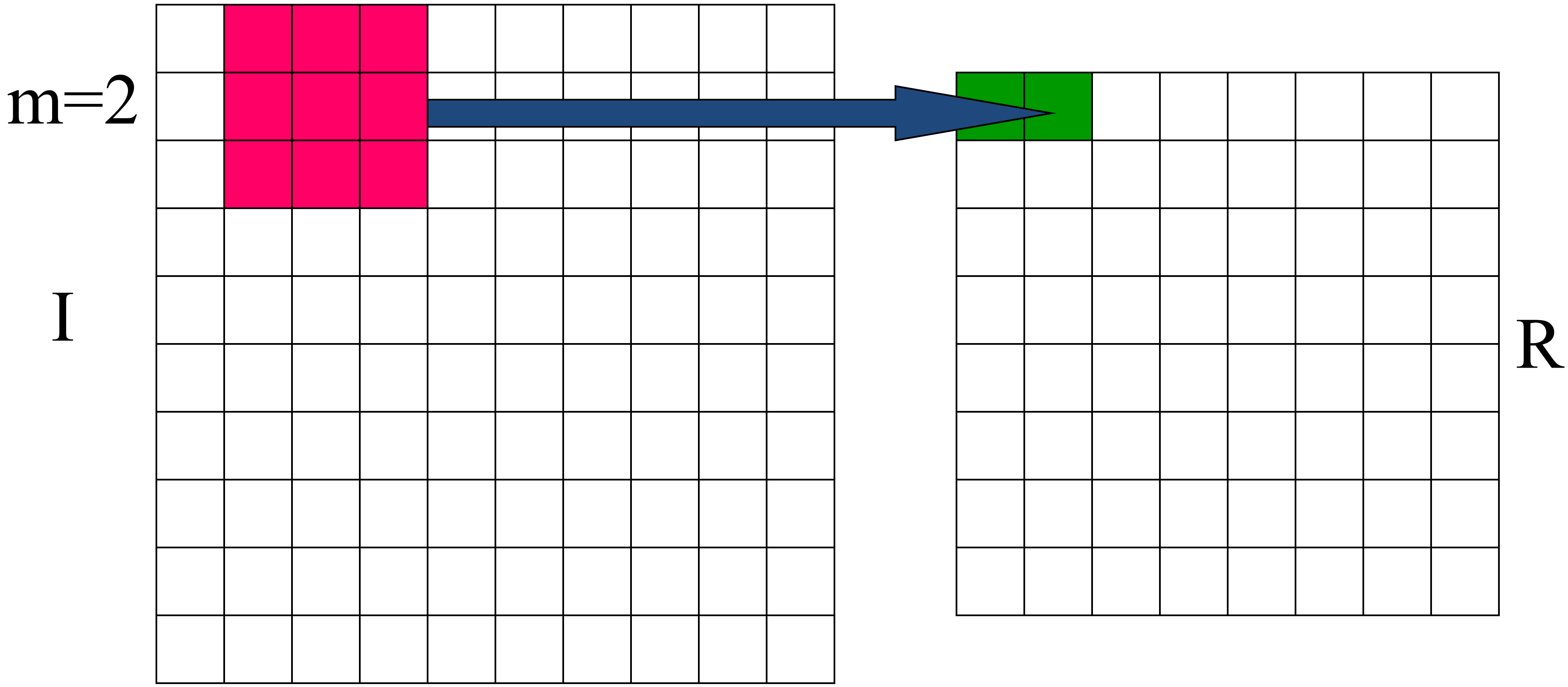
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

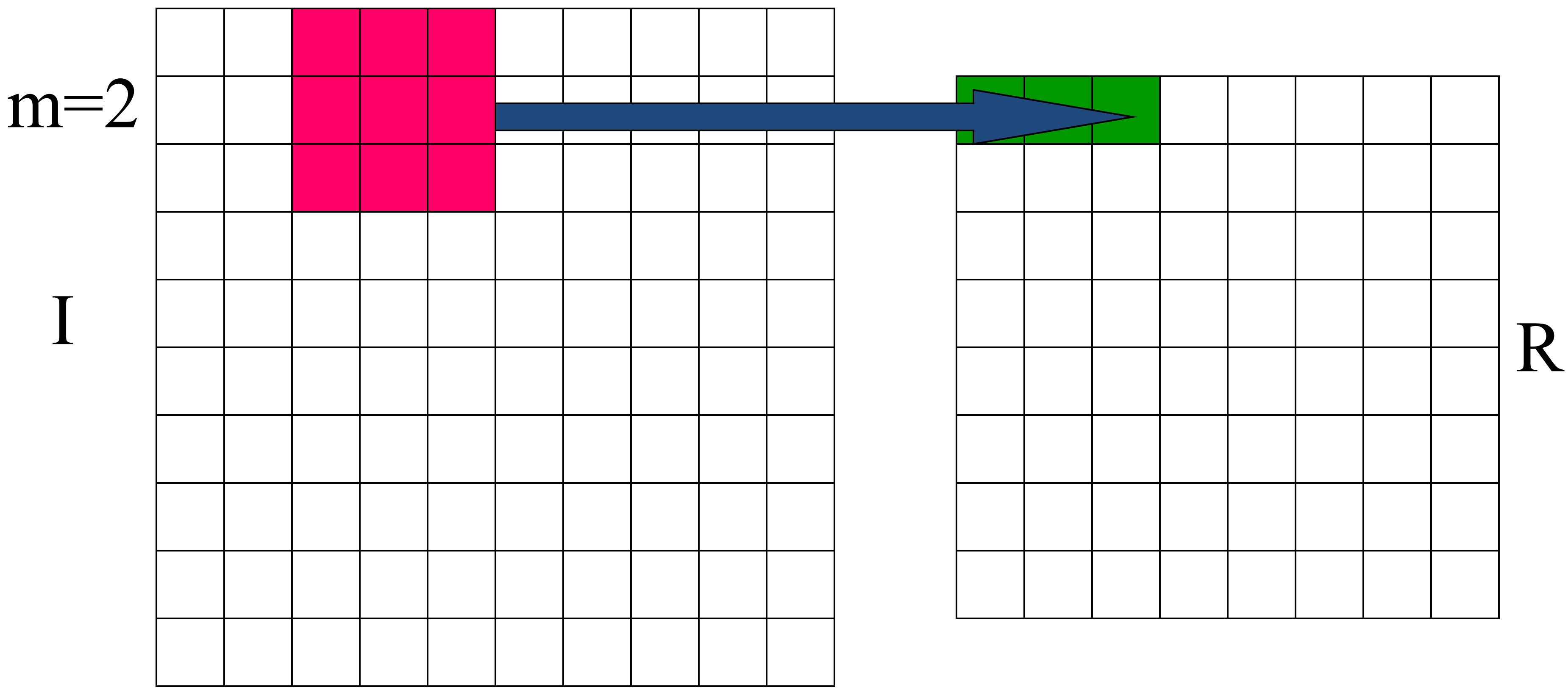
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

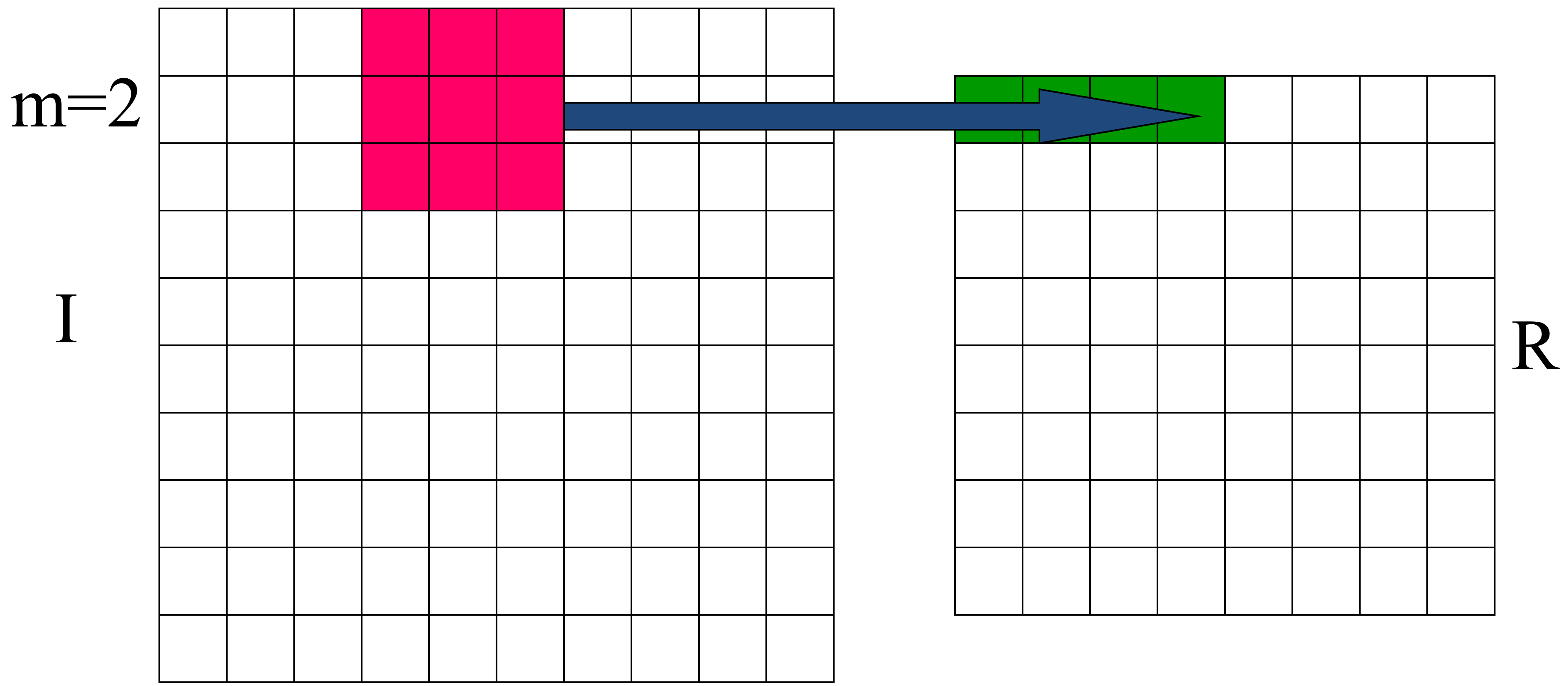
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

Convolution: R= K*I



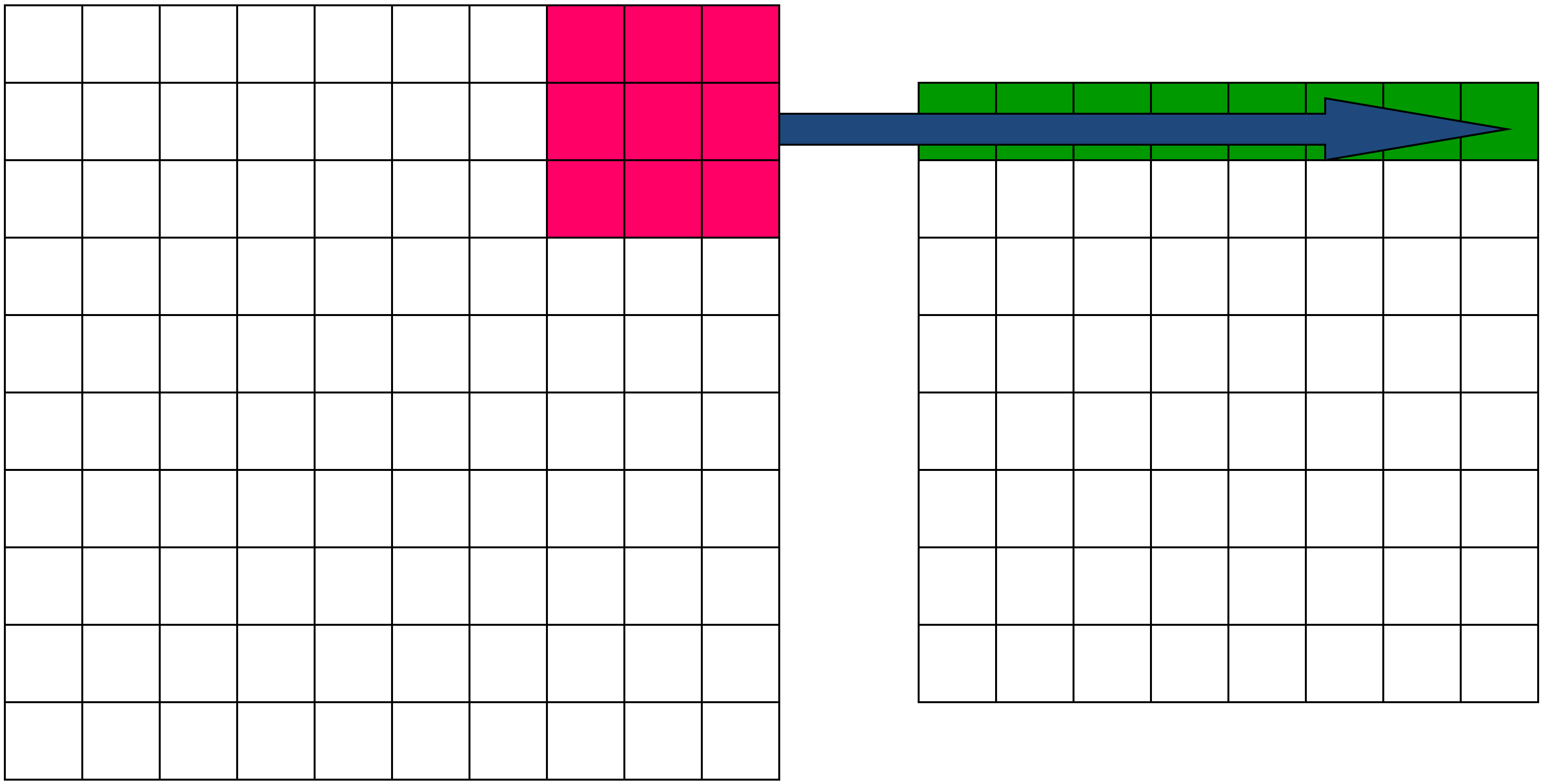
Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

Convolution: R= K*I

m=2

I

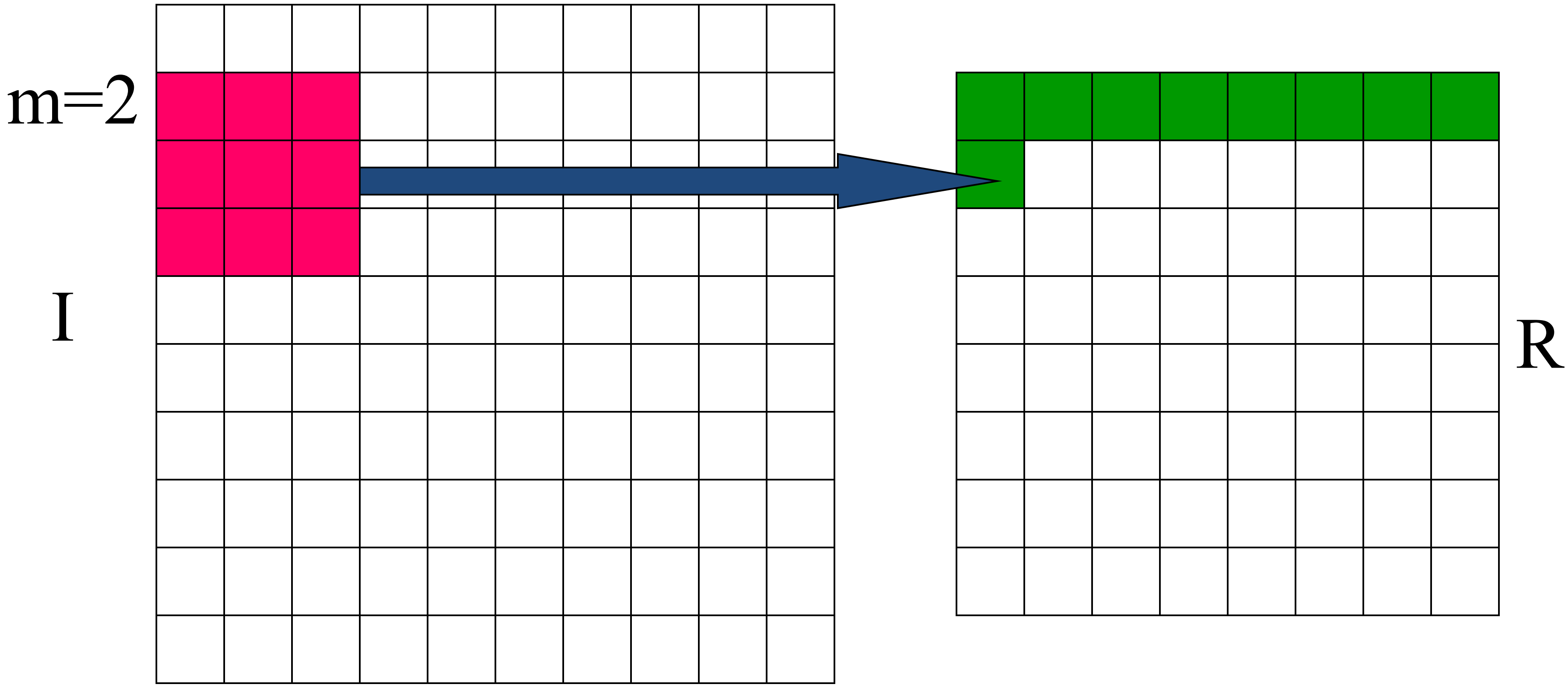


R

Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

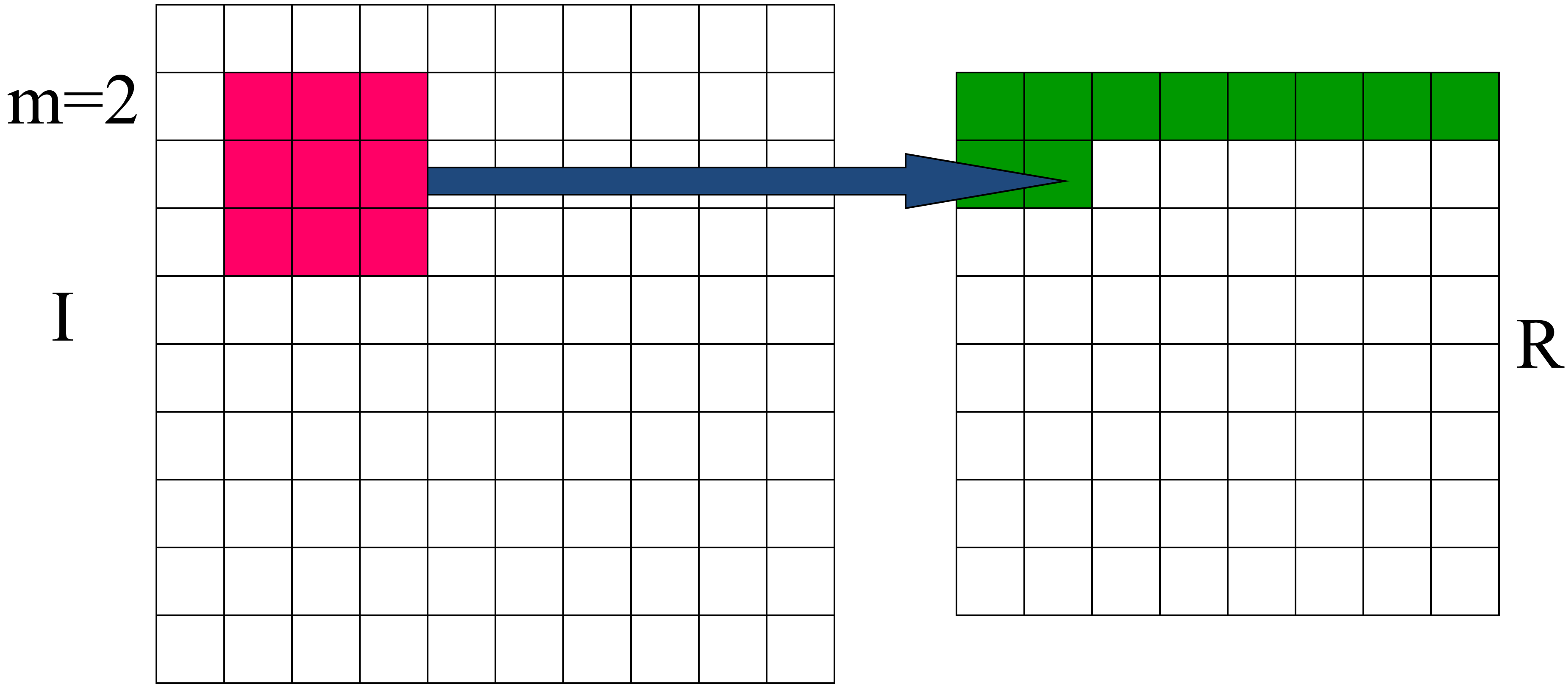
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

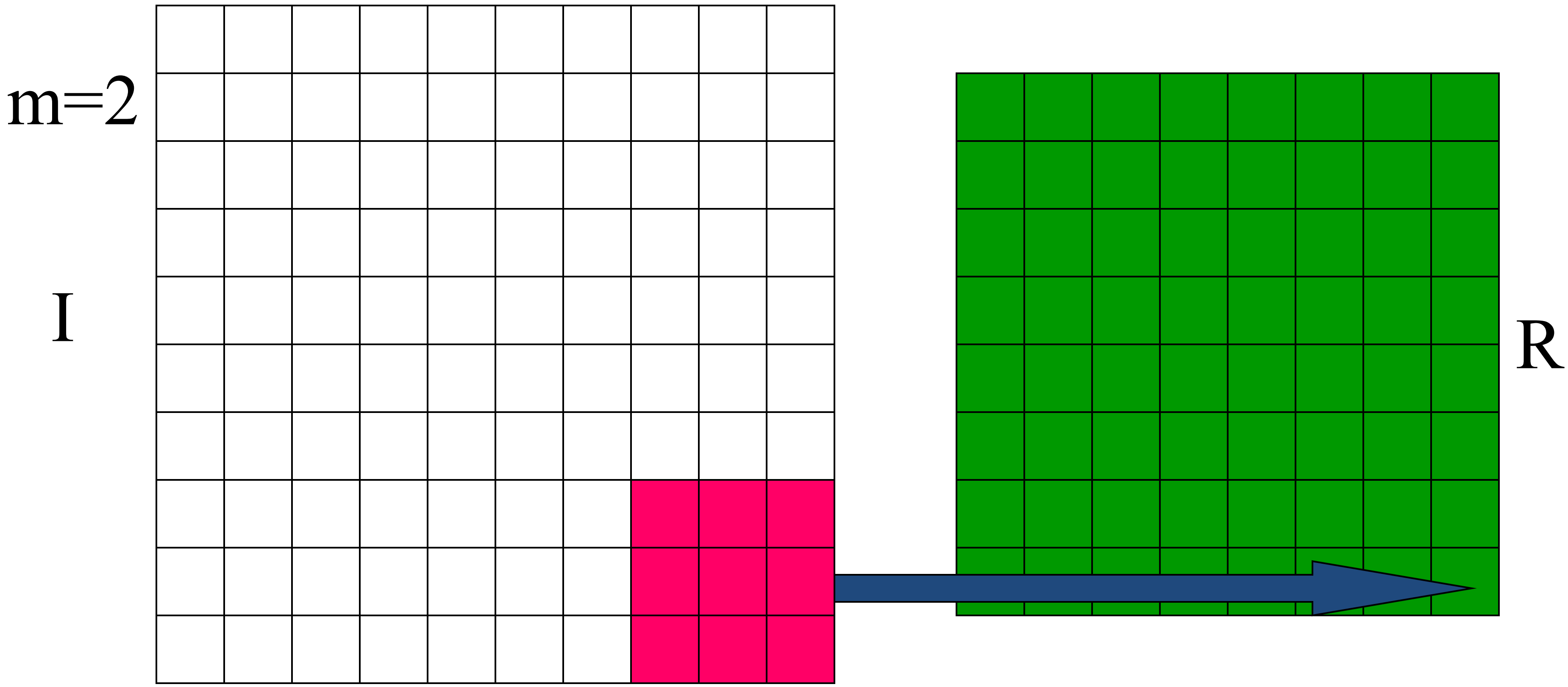
Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

Convolution: R= K*I



Kernel size
is m+1 by m+1

$$R(i,j)=\sum_{h=-m/2}^{m/2}\sum_{k=-m/2}^{m/2}K(h,k)I(i-h,j-k)$$

Numerical Derivatives

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} = \begin{bmatrix} \frac{-1}{2h} & 0 & \frac{1}{2h} \end{bmatrix} . [\, f(x_0 - h) \quad f(x_0) \quad f(x_0 + h)]$$

- With images, units of *h* is pixels, so *h*=1
  - Operator for derivative: [-1/2 0 1/2]
- When computing derivatives in the x and y directions, use these operators:

$$\frac{d}{dx} = [-1/2 \quad 0 \quad 1/2] \qquad \frac{d}{dy} = \begin{bmatrix} -1/2 \\ 0 \\ 1/2 \end{bmatrix}$$

- For convolution, can use square kernels

$$\boldsymbol{K}_x = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{K}_y = \frac{1}{2}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Numerical Derivatives

- In practice, Sobel operator is often used
- For image **A**, derivative images $\mathbf{G}_x$ and $\mathbf{G}_y$ are

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

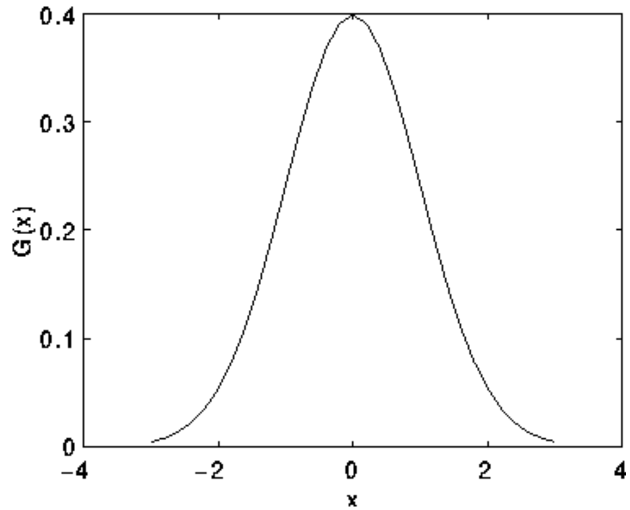
- Can be viewed as smoothing and derivative

$$\mathbf{G}_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \left(\begin{bmatrix} +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \right) \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} * \left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \mathbf{A} \right)$$

Aside: Gaussian Smoothing

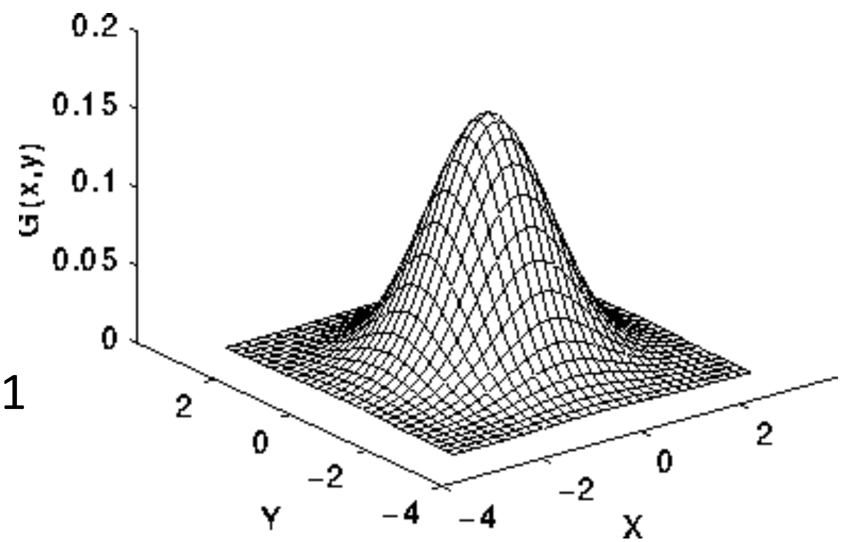
$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

A 1D Gaussian with mean 0, variance 1



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

A 2D Gaussian with mean (0,0), variance 1

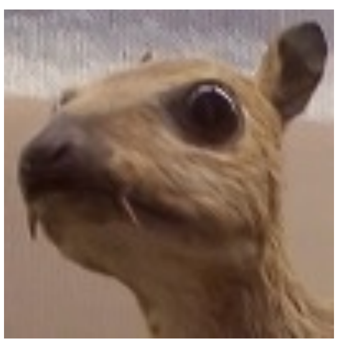


Aside: Gaussian Smoothing

A discrete approximation to a Gaussian kernel with variance 1

$\frac{1}{273}$ ×

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

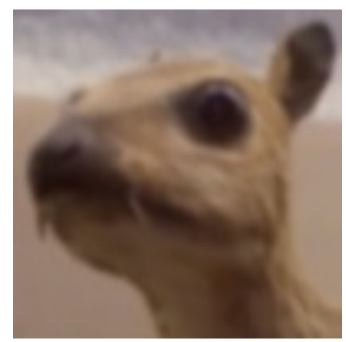


Convolution

$\circledast$ $\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

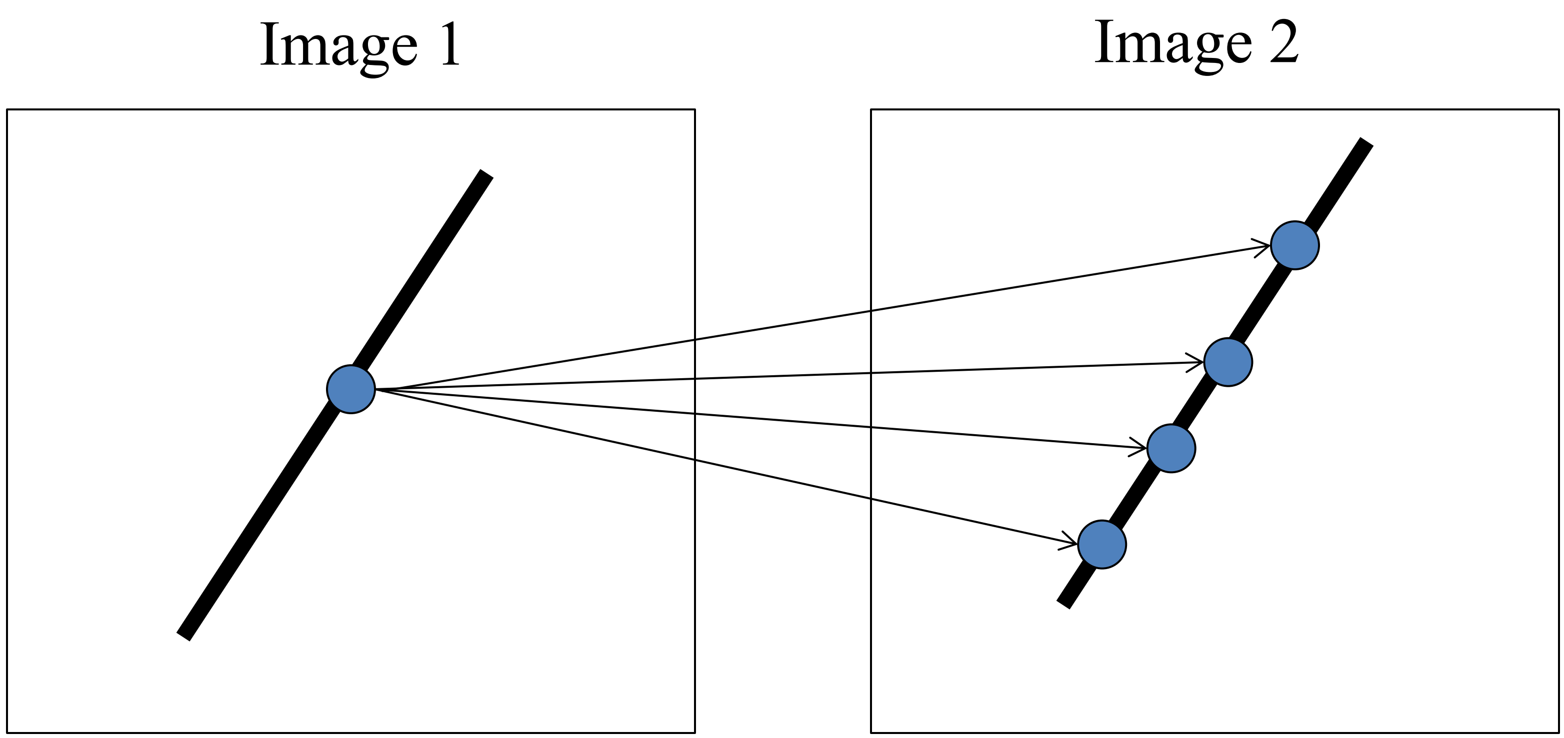
$\Rightarrow$



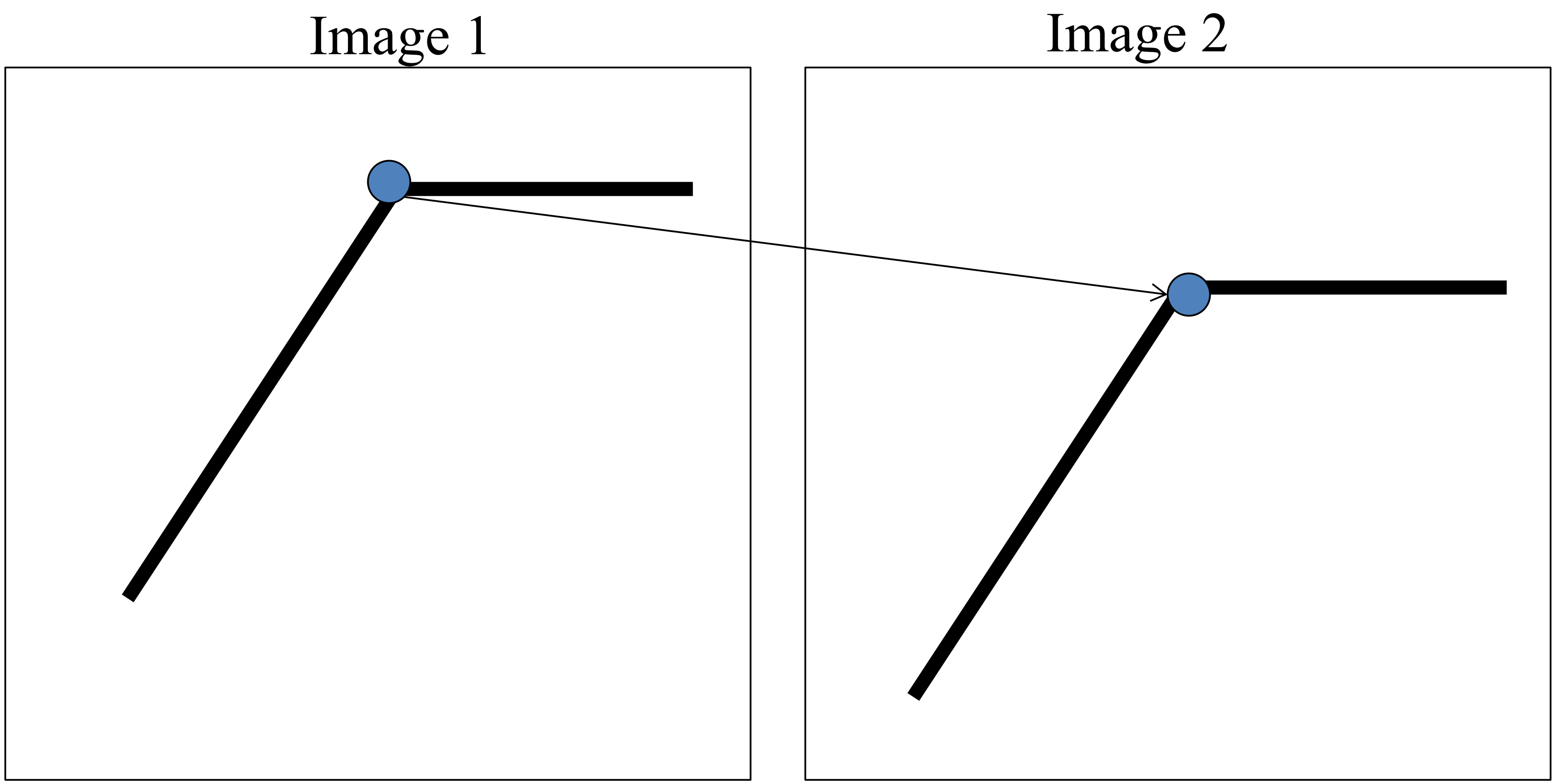
Corners

Corners more informative than lines

- A point on a line is hard to match

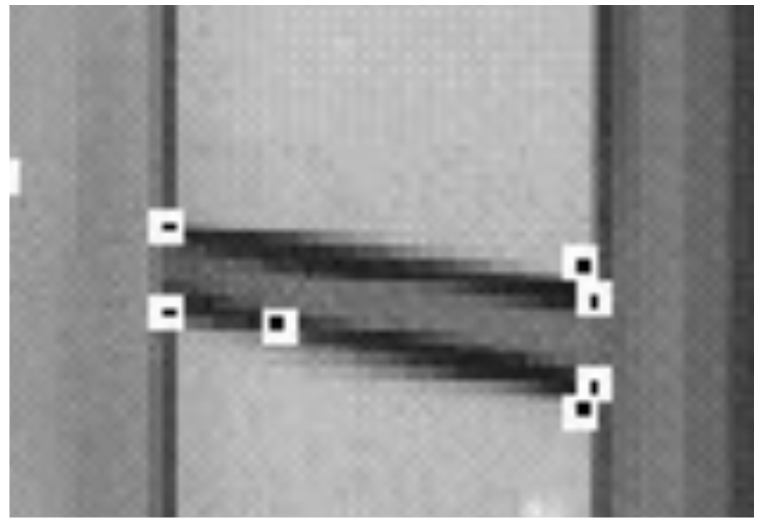


Corners more informative than lines



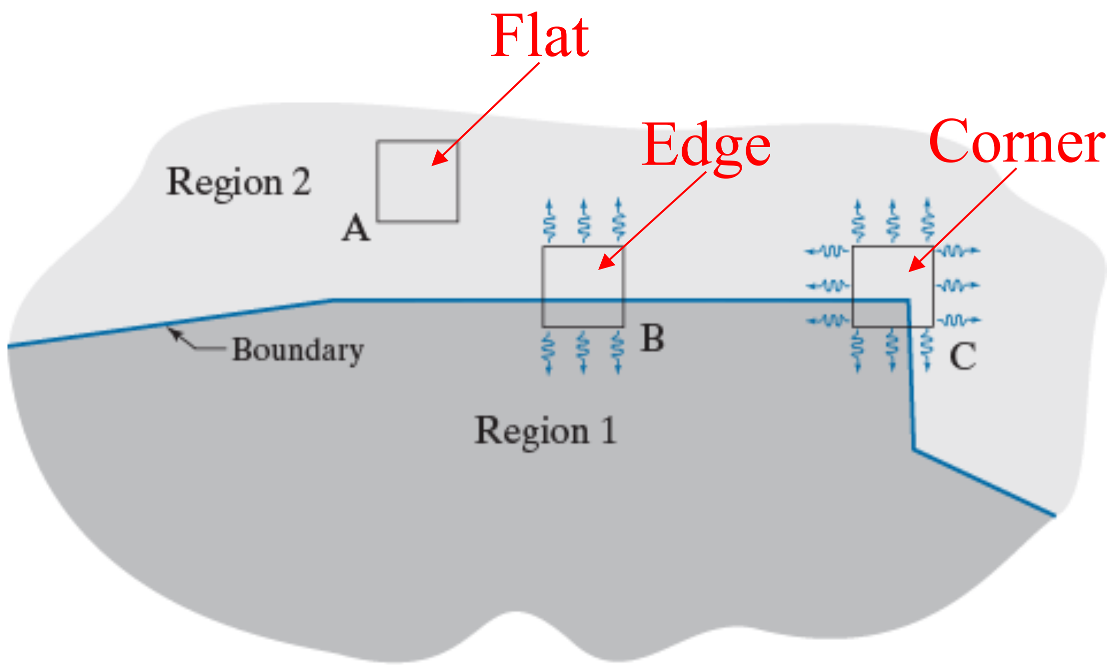
Corners

- A rapid change of direction in a curve
- A highly effective feature
  - Distinctive, reasonably invariant to viewpoint



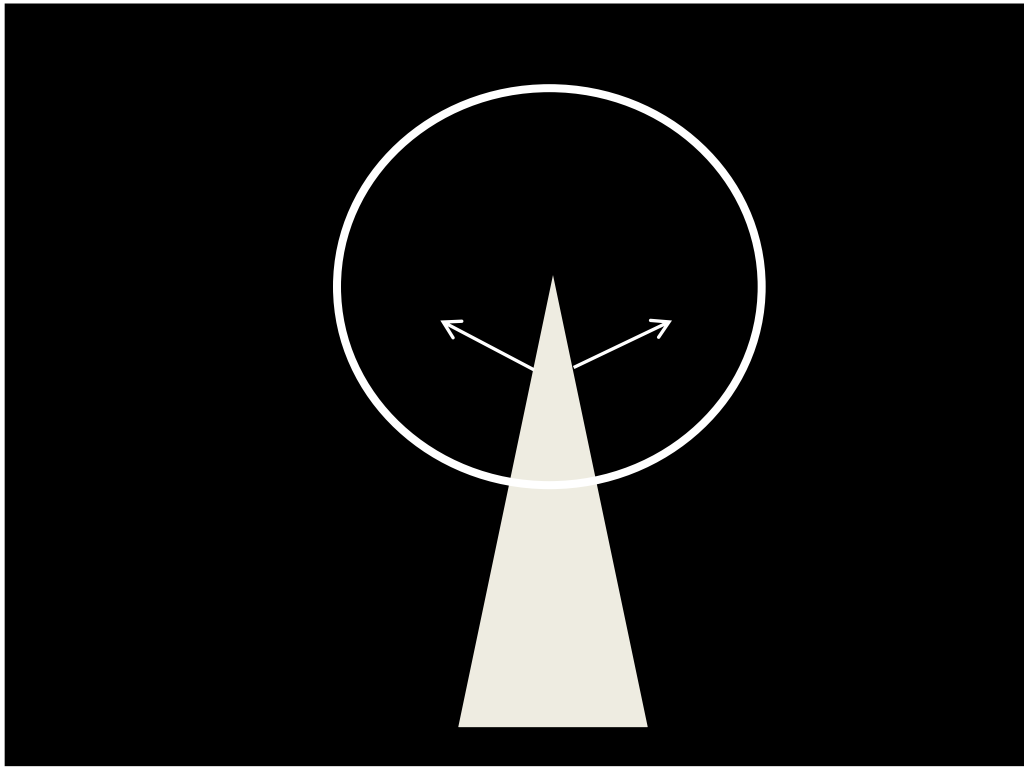
Detection of corner-like features

- Examine a small window over an image



The wiggly arrows indicate graphically a directional response in the detector as it moves in the three areas shown

Detection of corner-like features



Intuition:

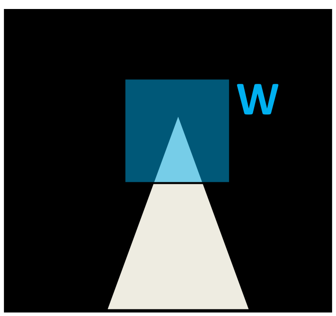
- Right at corner, gradient is ill-defined.
- Near corner, gradient has two different values.

The Harris corner detector

Compute second-moment matrix:

Sum over a small window W around hypothetical corner

Gradient with respect to x, times gradient with respect to y



$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

Slide credit: David Jacobs

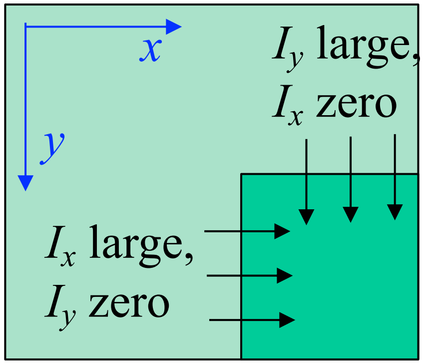
Simple Case

First, consider the case where:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis.

If either λ close to 0, then **not** a corner, so seek locations where both large.



Slide based on: David Jacobs

General Case

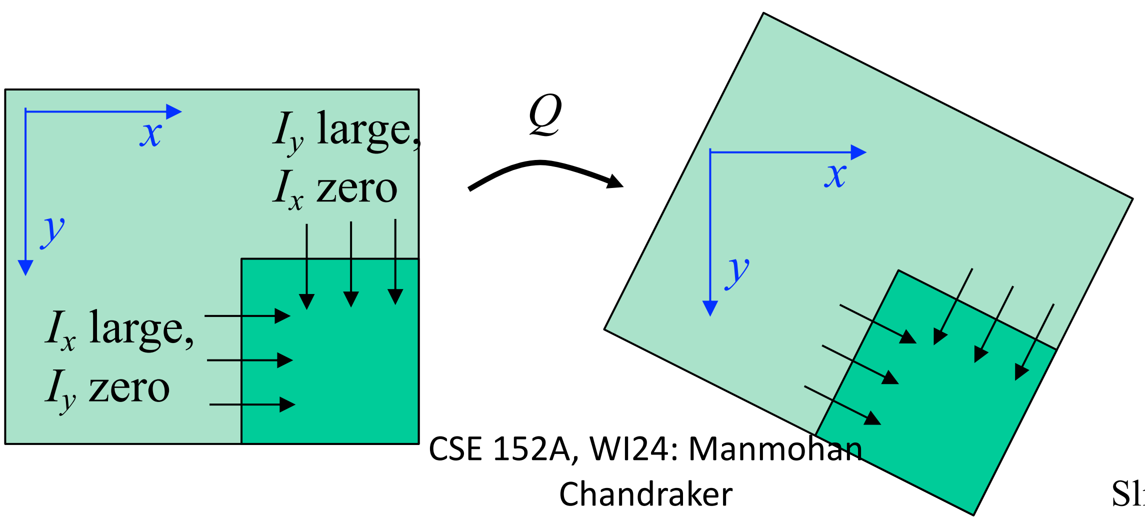
It can be shown that since C is symmetric:

$$C = Q^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} Q$$

Eigenvalues

Rotation

So every case is a rotated version of previous slide.



Slide based on: David Jacobs

Simple Corner Detector: Overall Idea

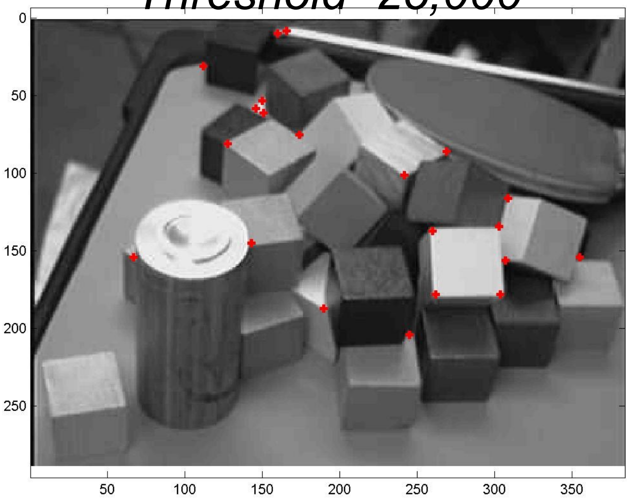
- Smooth image with Gaussian filter to reduce noise
- Compute magnitude of the *x* and *y* gradients at each pixel
- Construct C in a window around each pixel
- If λs are both big, we have a corner.

Simple Corner Detector: Implementation

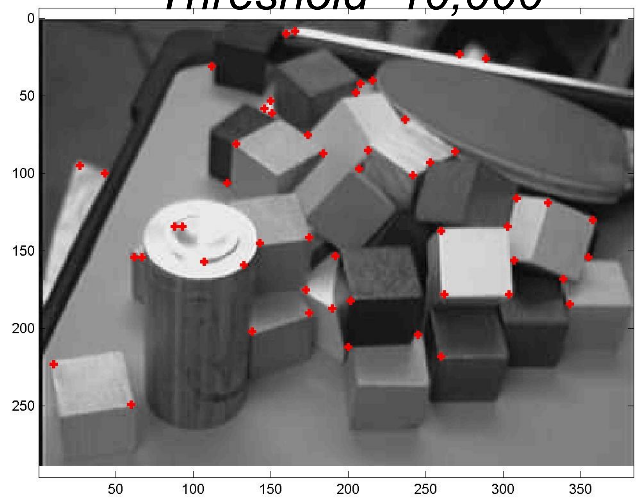
- Run a small window over an image and compute spatial gradient matrix ***C*** at every pixel
- Compute the minor eigenvalue of ***C*** at every pixel to obtain the corner response "image" ***R***
- Apply nonmaximal suppression to the "image" ***R***
  - Divide into grid, choose maximum within each grid cell
  - Resulting image ***R***' has only one corner candidate per grid cell
  - Prevents corners from being too close to each other
- Threshold resulting image ***R'*** using a global threshold T
  - Corners at pixels (x, y) corresponding to $\boldsymbol{R}'(x, y) > T$

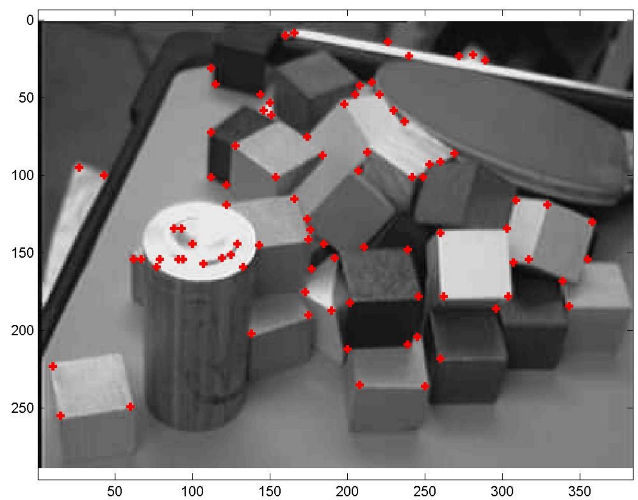
Simple Corner Detector: Outputs

Threshold=25,000



Threshold=10,000





Threshold=5,000