Edge and Corner Detection

Basics

- **Edge pixels** are those at which image intensity changes abruptly.
- **Edge segments** or edges are sets of connected edge pixels.

Digital derivatives: first versus second

- Along a ramp (i.e. intensity change with constant slope), first derivative is a non-zero constant.
- Second derivative is zero along the ramp, except at the start and end!
- Second derivative is preferable for image sharpening! Many edges in images are ramp-like, in which case **first** derivative will give **thick** edges (undesirable), whereas **second** derivative gives **thin** edges two pixels wide (desirable).
- Second derivative changes sign midway at a step edge **(zero crossing property).**

Digital derivatives: first versus second

- At **roof edges**, second derivative gives stronger response than first derivative (i.e. has larger magnitude) – except if both parts of the roof were at 45 degrees, where both will have equal response.
- At **isolated points**, second derivative gives stronger response than first derivative
- A rotationally invariant second-derivative operator is used for edge detection: the Laplacian, in fact zero-crossings of a Laplacian.

Laplacian of an image

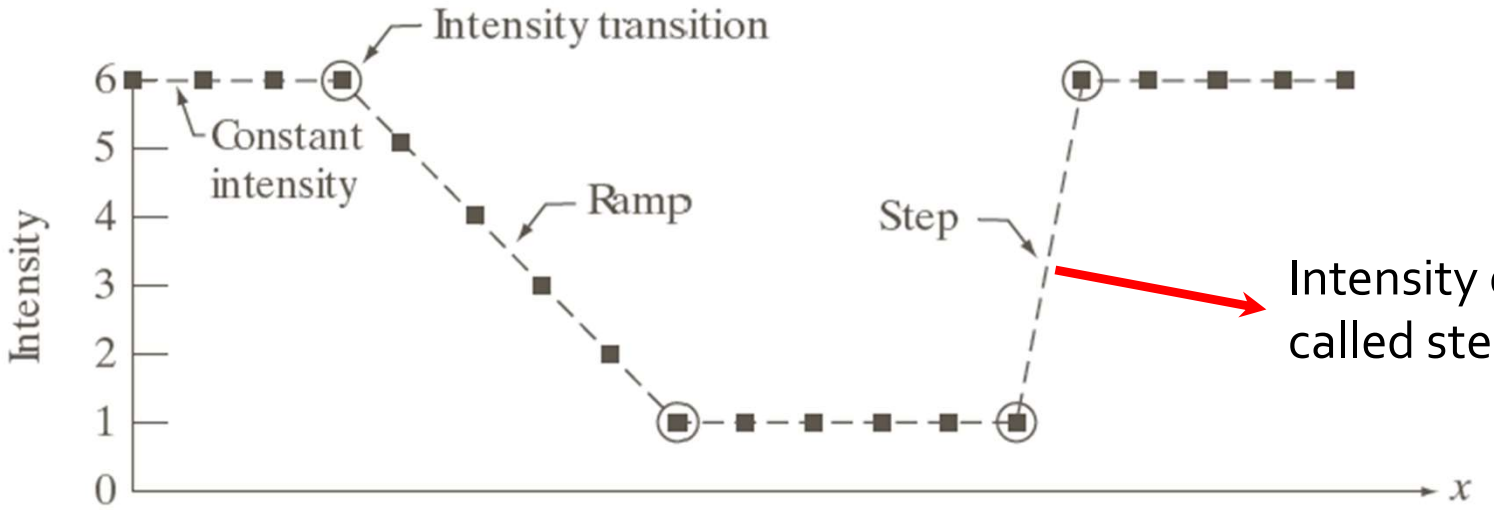
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Rotationally symmetric operator
(in the continuous domain)

$$= f(x+1, y) + f(x-1, y) - 2f(x, y)$$
$$+ f(x, y+1) + f(x, y-1) - 2f(x, y)$$
$$= f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

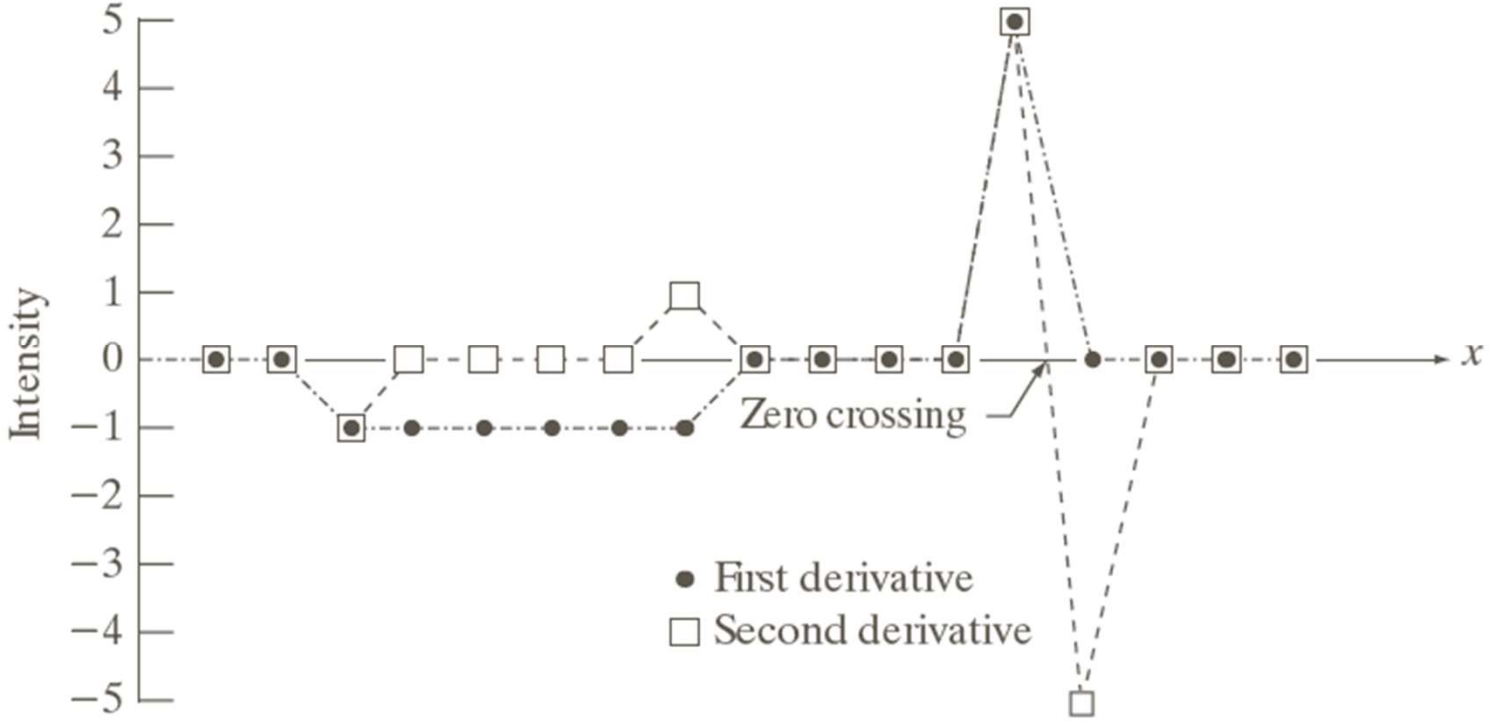
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Laplacian operators: second operator is obtained by adding second derivatives along both the diagonals, to the first operator



Intensity discontinuity, also called step edge

Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0		
2nd derivative	0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0		



From book by Gonzalez and Woods

Edge detection under noise

First and second derivatives are very sensitive to noise!

Left to right in each row: original image, first derivative, second derivative

In any column: increase in noise level

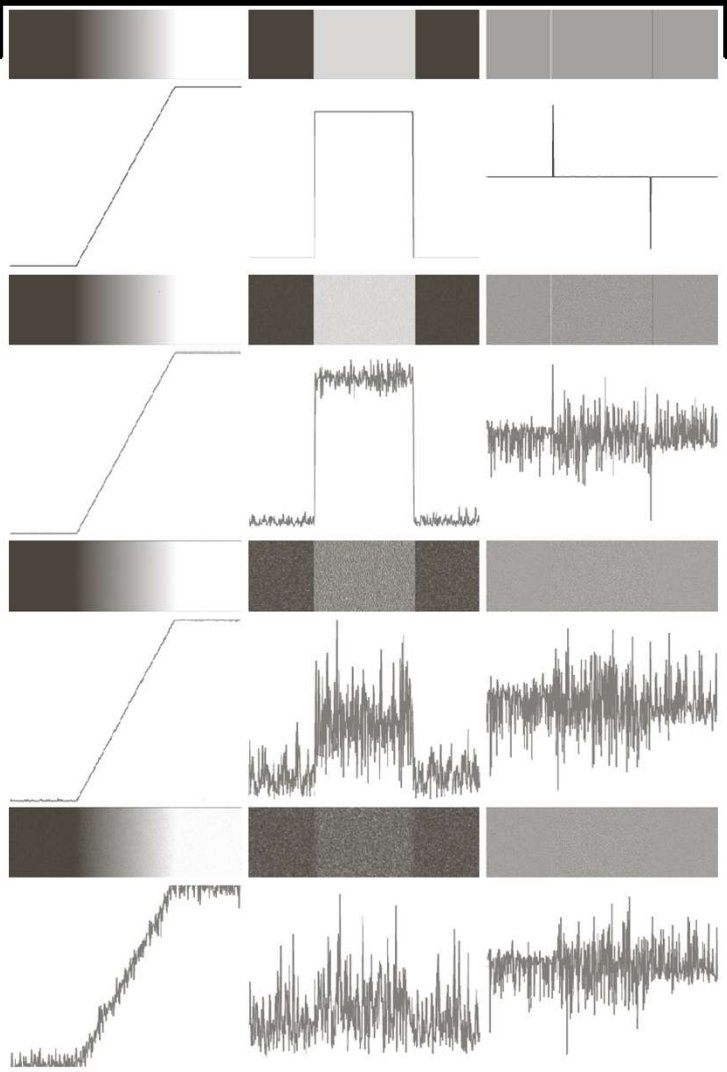


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

Steps in robust edge detection

- Image smoothing to reduce noise: because first and second derivatives are noisy
- Actual detection of edge points (may produce thick or disconnected edges)
- Pruning or linking of the edge points

Edge detection: image gradient

- An image gradient is defined as follows:

$$\nabla f = \begin{pmatrix} \partial f / \partial x \\ \partial f / \partial y \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$

$$\text{Gradient magnitude} = \|\nabla f\|_2 = \sqrt{f_x^2 + f_y^2}$$

$$\text{Gradient direction} = \theta = \tan^{-1}(f_y / f_x)$$

Gradient operator :

$$f_x(x, y) = f(x+1, y) - f(x, y)$$

$$f_y(x, y) = f(x, y+1) - f(x, y)$$

Robust edge operators

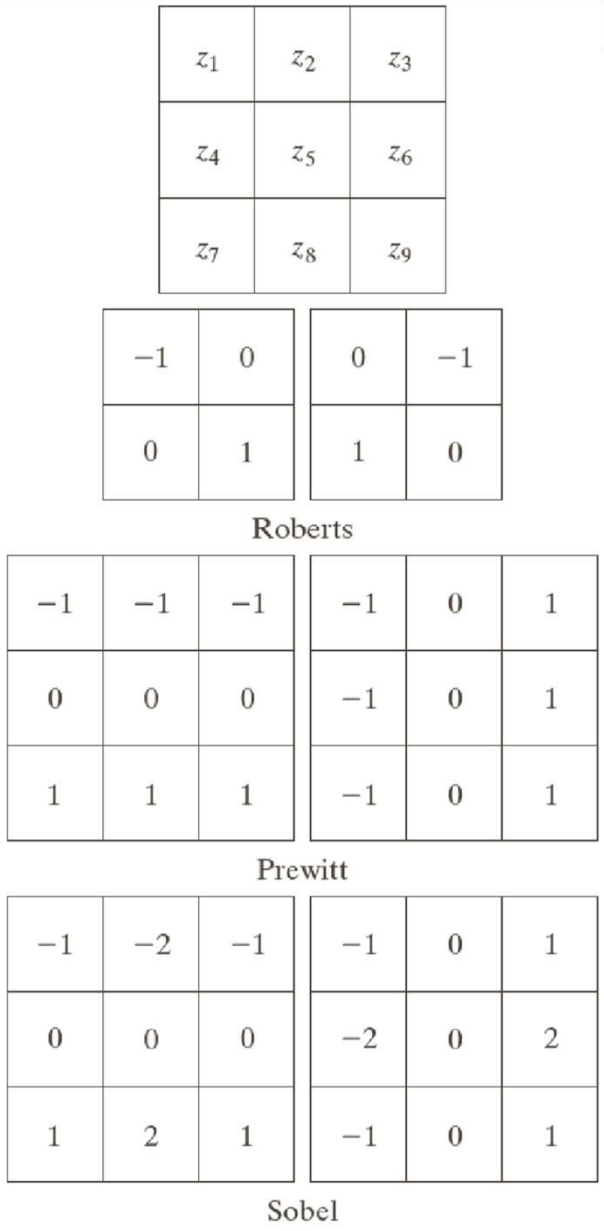
These operators: Prewitt and Sobel perform some smoothing prior to difference computation, and hence are more robust than the simple finite differences from the previous slide.

$$f_y = \frac{\partial f}{\partial y} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$f_x = \frac{\partial f}{\partial x} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

$$f_y = \frac{\partial f}{\partial y} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$f_x = \frac{\partial f}{\partial x} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$



a
b c
d e
f g

FIGURE 10.14 A 3 × 3 region of an image (the z's are intensity values) and various masks used to compute the gradient at the point labeled z_5.

Prewitt and Sobel

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

- Apply the differencing mask $[-1\ 0\ 1]^T$ and then apply the smoothing mask [1 1 1] on the result. This is equivalent to applying the first Prewitt mask (i.e. for f_y).
- For the Prewitt operator for f_x, apply [-1 0 1] and then $[1\ 1\ 1]^T$.

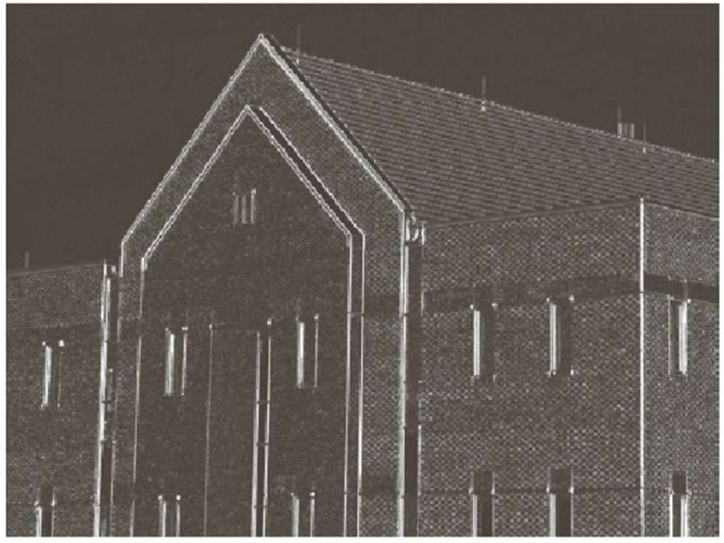
- Apply the differencing mask $[-1\ 0\ 1]^T$ and then apply the smoothing mask [1 2 1] on the result. This is equivalent to applying the first Sobel mask (i.e. for f_y).
- For the Sobel operator for f_x, apply [-1 0 1] and then $[1\ 2\ 1]^T$.

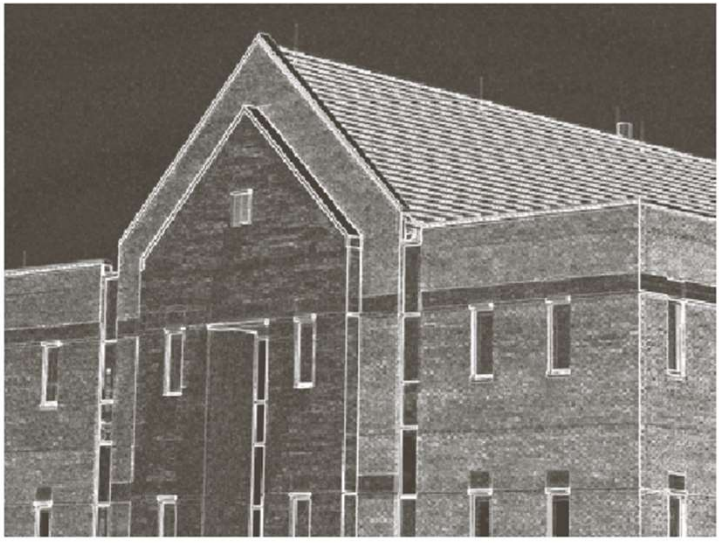
Edges with Sobel Mask

Original image

Derivatives in y direction: f_y

Derivatives in x direction: f_x

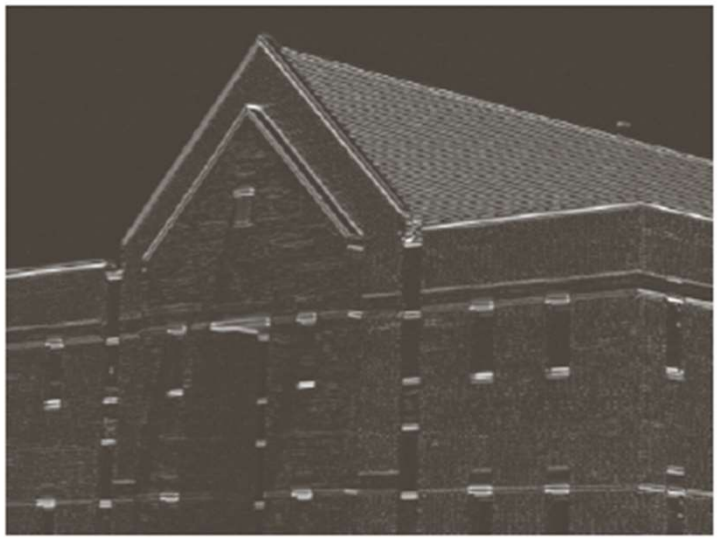




$|f_x| + |f_y|$

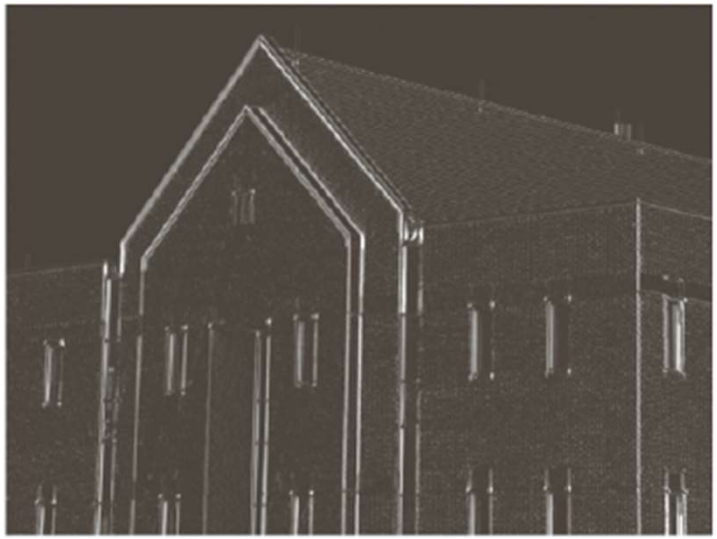
Edges with 5 x 5 averaging followed by Sobel Mask

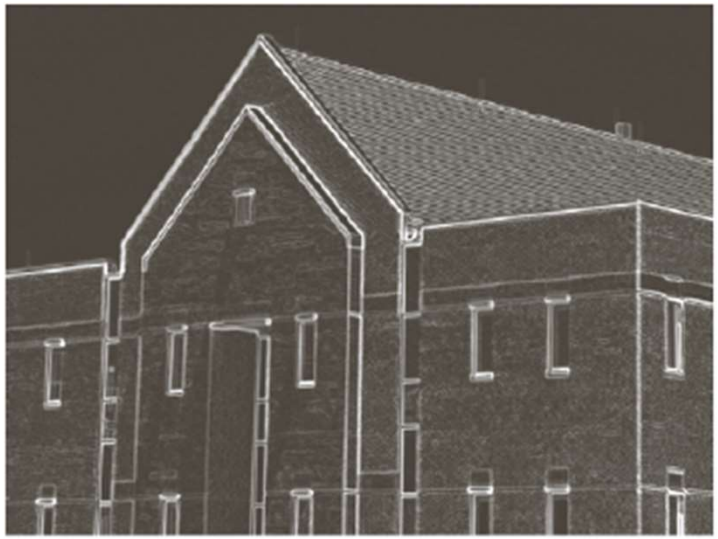
Original image



Derivatives in y direction: f_y

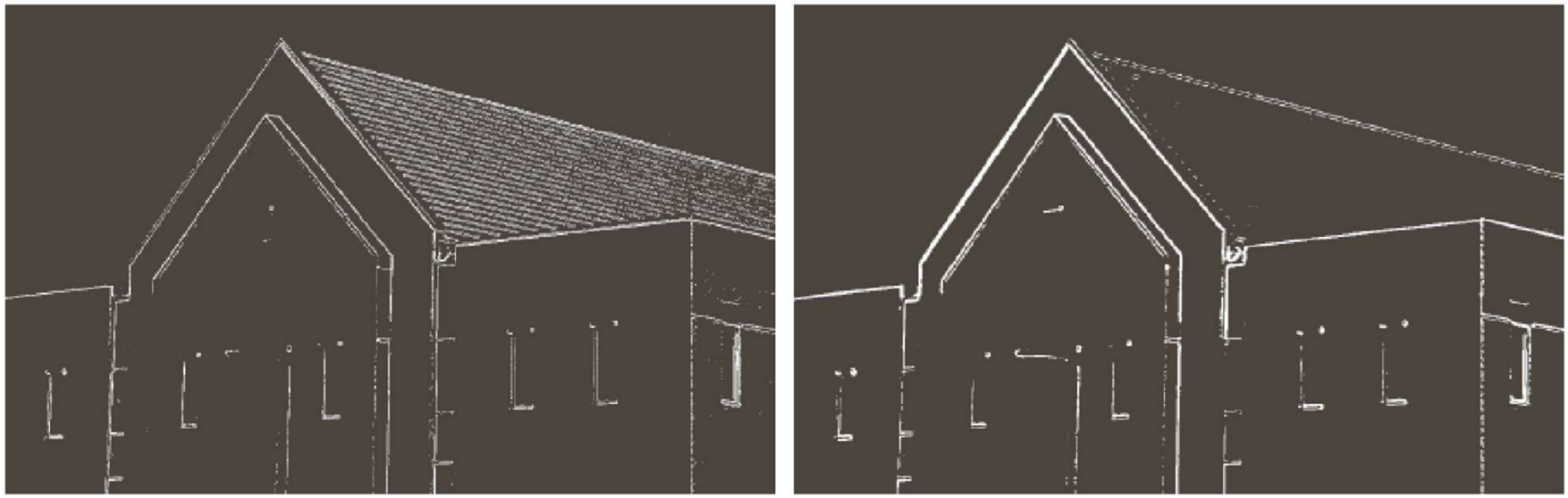
Derivatives in x direction: f_x





$|f_x| + |f_y|$

Notice: finer details removed



a b

FIGURE 10.20 (a) Thresholded version of the image in Fig. 10.16(d), with the threshold selected as 33% of the highest value in the image; this threshold was just high enough to eliminate most of the brick edges in the gradient image. (b) Thresholded version of the image in Fig. 10.18(d), obtained using a threshold equal to 33% of the highest value in that image.

Edge map after Thresholding the earlier map.

Marr-Hildreth edge detector

- Principled approach to combined edge detection and image smoothing

- An edge operator should satisfy two criteria:

(1) Notion of **scale** in edge detection (operators of different sizes)

(2) Differential operator to compute first or second derivatives of the intensity

- Uses the **Laplacian of Gaussian** operator on an image satisfies both criteria

Laplacian of Gaussian (LoG or Mexican hat)

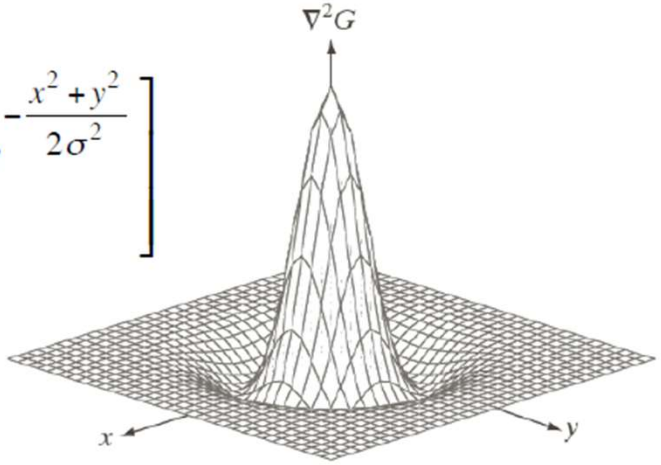
$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

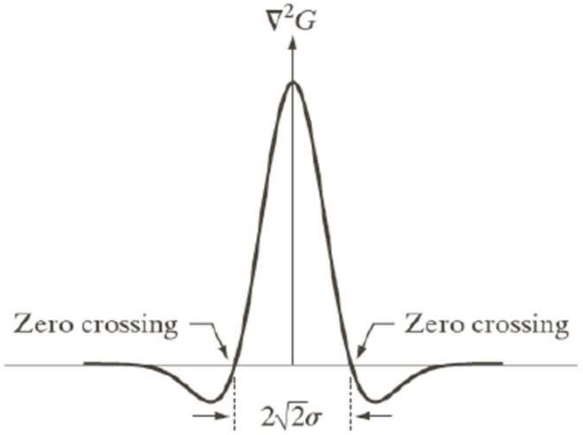
$$\frac{\partial^2 G(x,y)}{\partial x^2} = \frac{\partial^2}{\partial x^2}\left[e^{-\frac{x^2+y^2}{2\sigma^2}}\right] = \frac{\partial}{\partial x}\left[-\frac{x}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

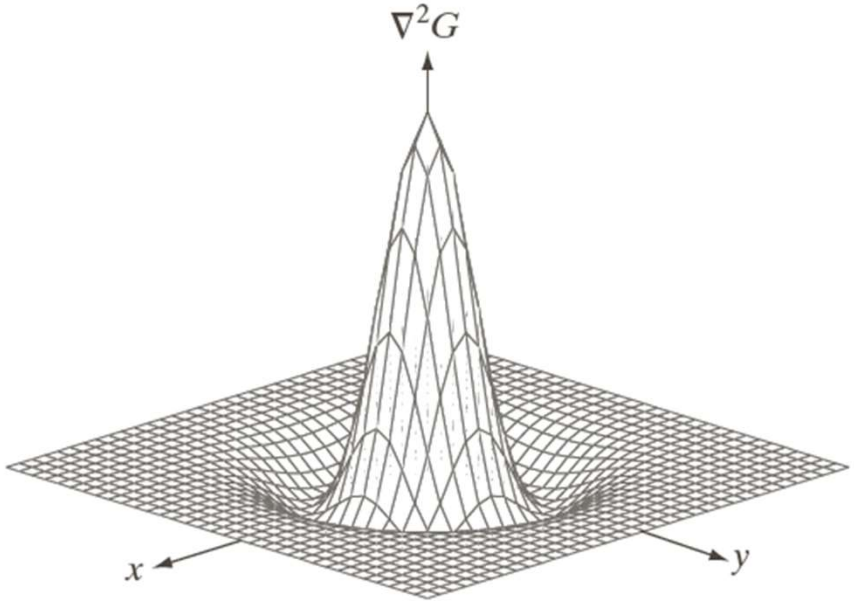
$$= \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

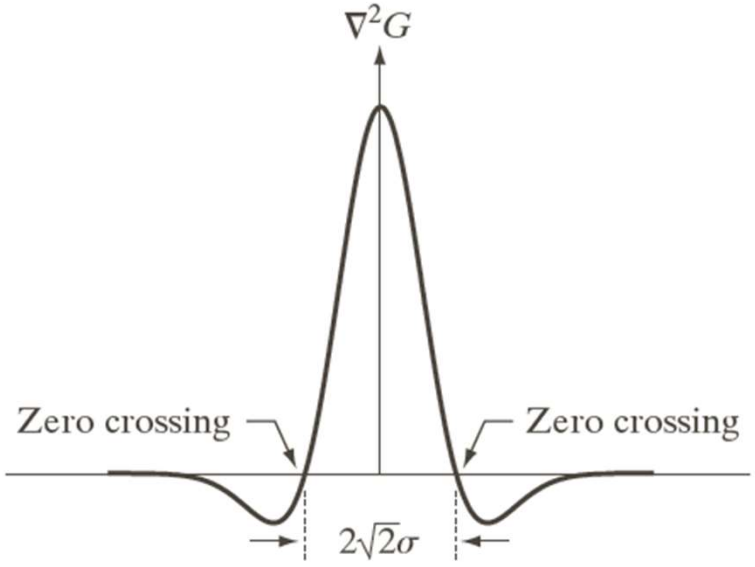
$$\frac{\partial^2 G(x,y)}{\partial y^2} = \left(\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

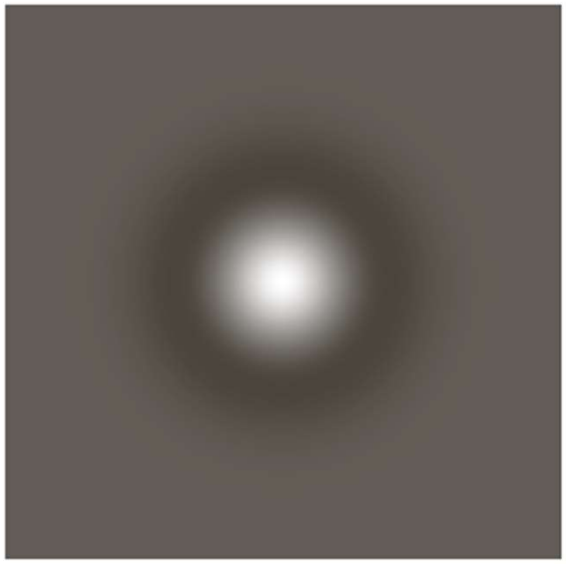
$$\nabla^2 G(x,y) = \left(\frac{x^2}{\sigma^4} + \frac{y^2}{\sigma^4} - \frac{2}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$











0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

a b
c d

FIGURE 10.21
(a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5×5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

Why LoG for edge detection?

- Remember: we are interested in zero-crossings of the second derivative!
- We want the second derivative operator to be rotationally invariant. The Laplacian is the simplest (though not the only) such operator.
- The Gaussian is a low-pass filter, which blurs structures with intensity difference smaller than σ.
- Gaussian is smooth in spatial and frequency domains.

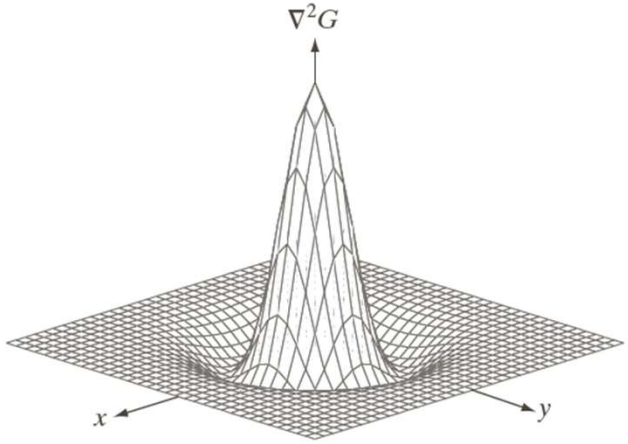
Using LoG for edge detection

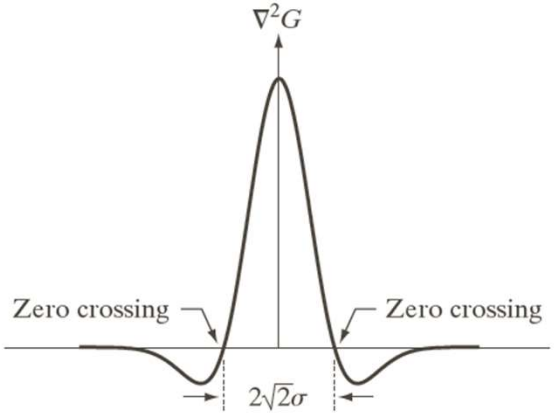
- (1) Convolve the image with the LoG operator, i.e.:

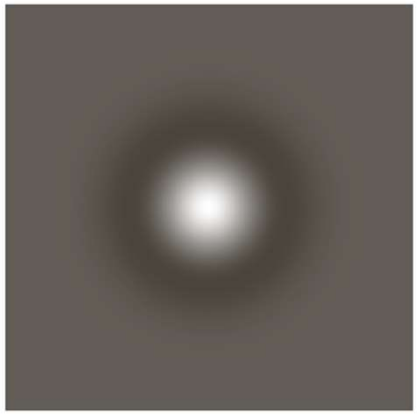
$$g(x,y) = [\nabla^2 G(x,y)] * f(x,y) = \nabla^2 [G(x,y) * f(x,y)]$$

- (2) Find the zero-crossings in the resultant image.

Implementing the convolution







0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

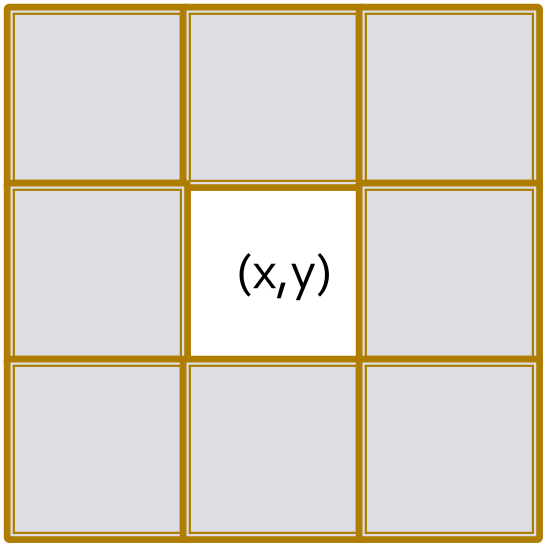
Use an n x n mask, created by sampling the formula for LoG on a discrete Cartesian crid of size n x n.

How to choose n? Depends on σ: n should be >= 6σ, because more than 99% of the volume underneath a 2D Gaussian lies between μ-3σ and μ+3σ.

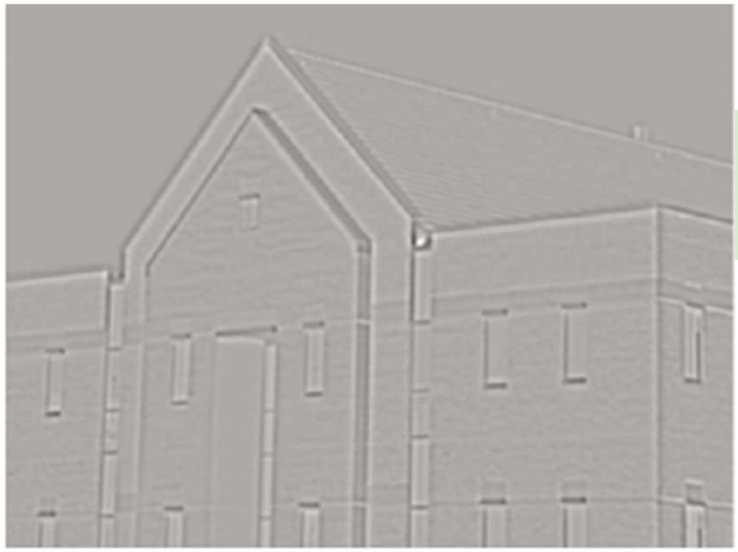
Finding zero-crossings

- Look at a 3 x 3 window around a pixel.

- There exists a zero-crossing at that pixel if the signs of at least one pair of opposing neighbors (top & bottom, left & right, top-left & right-bottom, top-right & left-bottom) differ and the difference in their values must exceed a threshold *p*.



σ = 4,
n = 25



Convolution with LoG

Zero-crossings

Zero-crossings + thresholding

Some properties of the Marr-Hildreth Edge Detector

- Detected edges are 1-pixel in thickness
- Can be used for different sigma values, keeping only those edge points that are common to multiple scales

Canny Edge Detector: using 1st derivatives

- Three aims/objectives:

(a) Low error (no spurious responses due to noise)
(b) Well-localized edge points
(c) Single pixel wide edges

Canny's detector was formulated by analyzing step edges in 1D under zero-mean i.i.d. Gaussian noise, and expressing the preceding three criteria mathematically and deriving an (approximate) operator that obeys all criteria.

Canny's Edge Detector

- In 1D, the approximate operator was the first derivative of the Gaussian: $\frac{d}{dx} e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$

- In 2D, the approximate operator was the first derivative of the Gaussian in the direction of the local image gradient (i.e. perpendicular to the edge).

- But the true gradient is not known, so we first smooth the image with a 2D Gaussian and then use its gradient magnitude and direction.

Canny's Edge Detector

- Gradient magnitude and direction:

$$g(x,y) = G(x,y) * f(x,y)$$

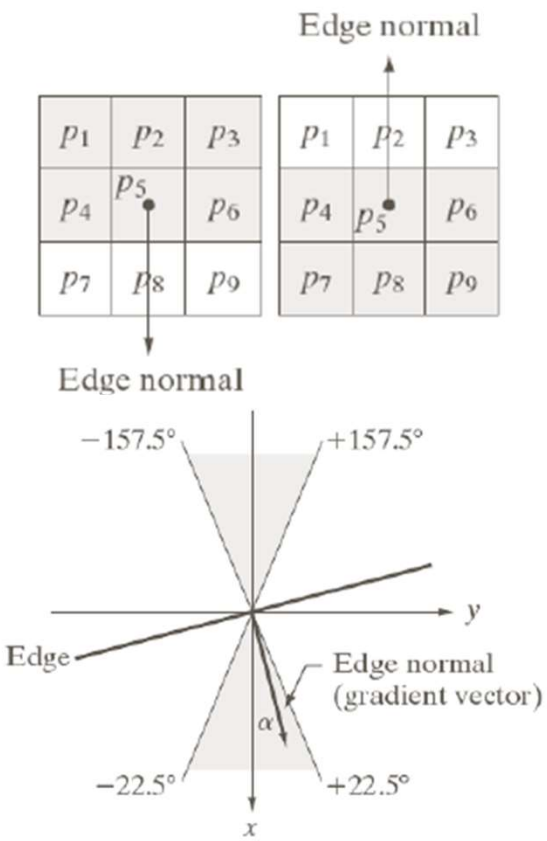
$$M(x,y) = |\nabla g(x,y)| = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$$

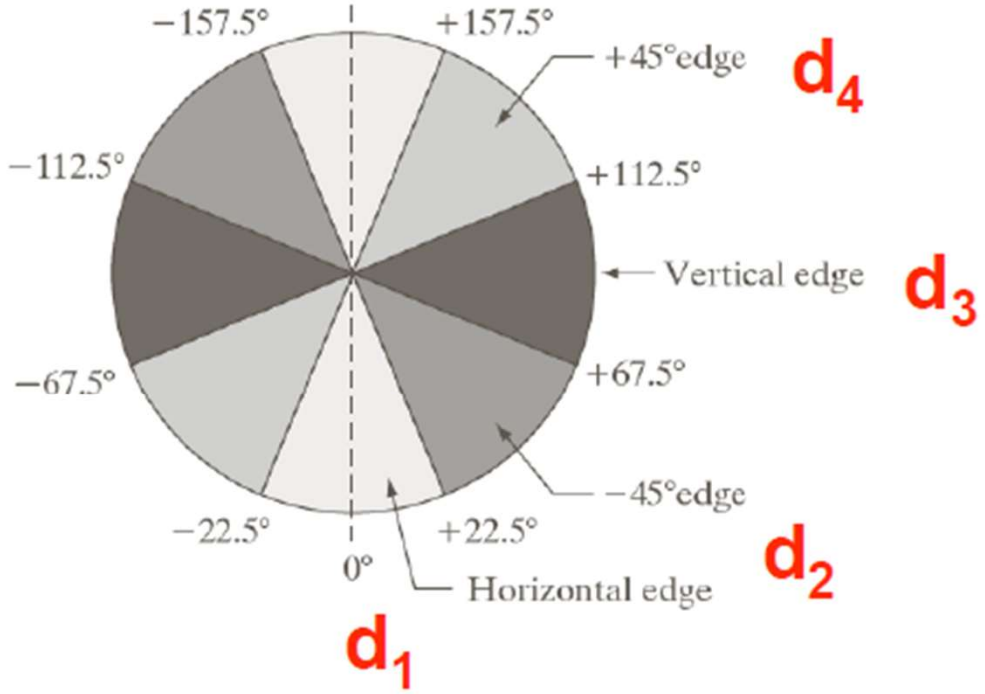
$$\theta(x,y) = \tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right)$$

- M(x,y) contains wide ridges along the true edges of the image.
- Though the maximum values are along the edge, those values don't immediately fall off to 0 on either side of the edge.
- Recall ramp edges.

Canny edge detector

- This requires us to perform a **thinning** operation – called as **non-maximal suppression** (i.e. we set to zero those values of M(x,y) which are likely to NOT be local maxima).





Canny edge detector

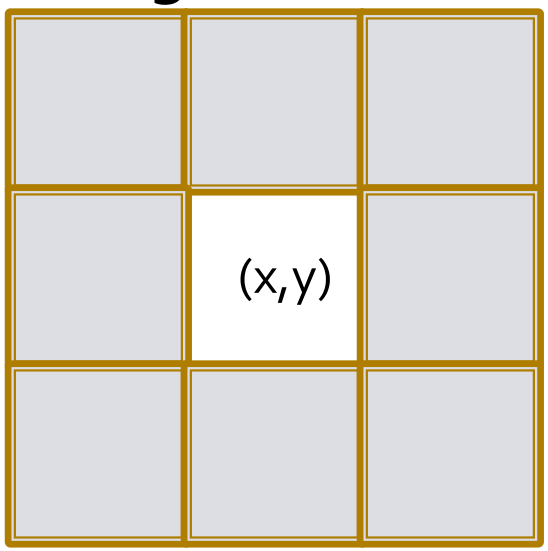
- Consider the four quantized directions shown on the previous slide. Pick the direction d_k, which is closest to $\Theta(x,y)$.
- If M(x,y) is less than either of its two neighbors along the direction d_k, then set it to 0 (i.e. suppress the non-maximum), otherwise we leave it as is.
- After this processing, M(x,y) is the non-maximal suppressed image.

Canny Edge Detector

- The (thinned) edge map so far may still contain some false positives. This is taken care of (to some extent) by **hysteresis thresholding**.
- This uses a low-threshold TL and a high threshold TH.
- We keep only those pixels for which

-M(x,y) > TH, OR

-M(x,y) > TL AND (x,y) is a neighbor of at least one pixel with M(x,y) > TH.



The 8 neighbors of pixel (x,y)

Summary: Canny Edge detector

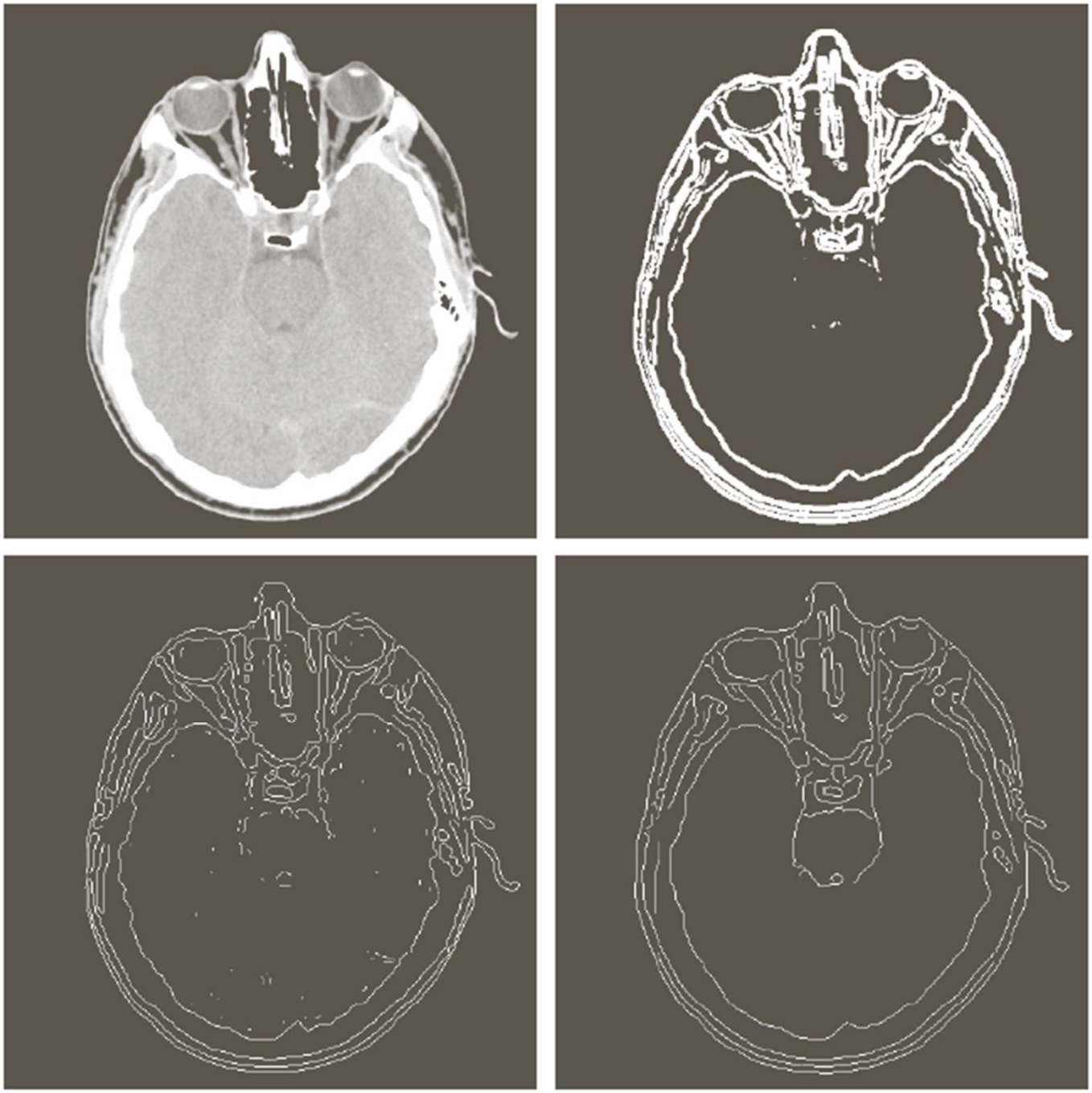
(1) Convolve the image with a Gaussian Filter. Compute the gradient magnitudes and orientations.

(2) Perform non-maximal suppression.

(3) Perform hysteresis thresholding.

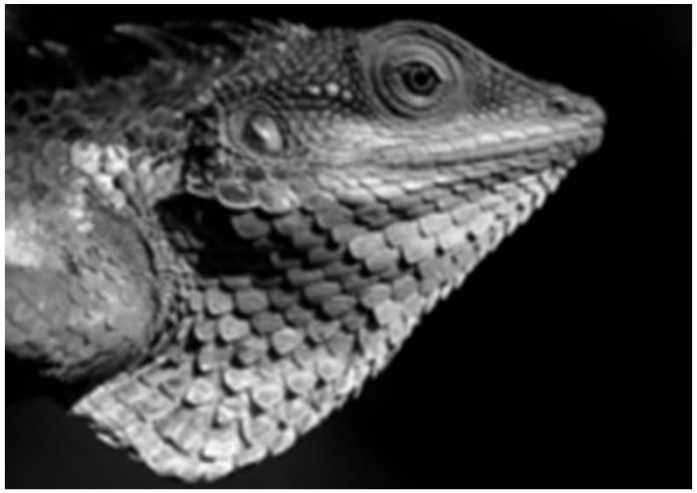
a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

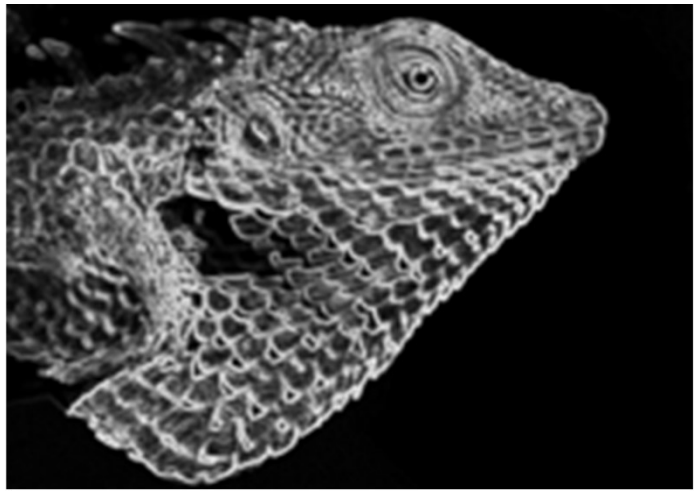


a b
c d

FIGURE 10.26
(a) Original head CT image of size 512 × 512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

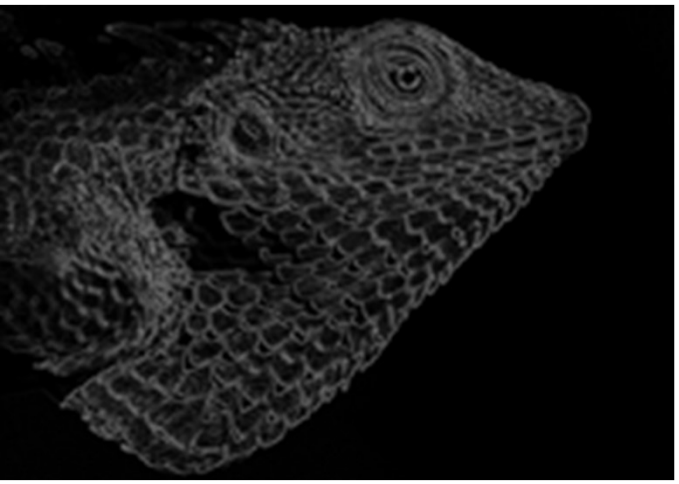


Gaussian filtered (sigma = 1.4, 5 x 5 mask)

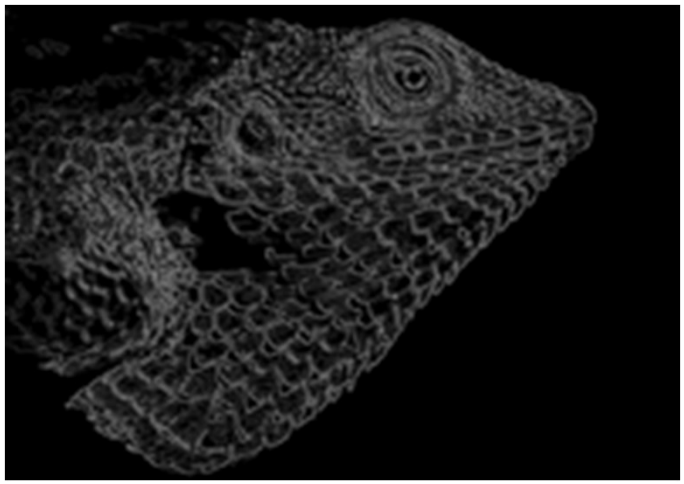


Intensity gradients superimposed

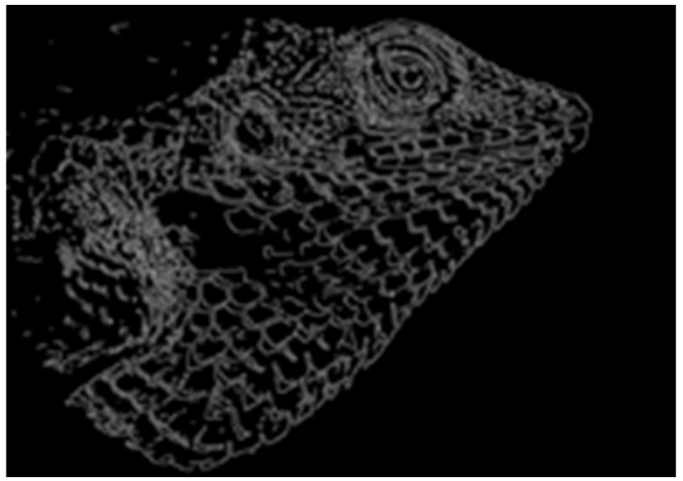
https://en.wikipedia.org/wiki/Canny_edge_detector



Non-maximum suppression



Double thresholding (TL = 0.1, TH = 0.3)

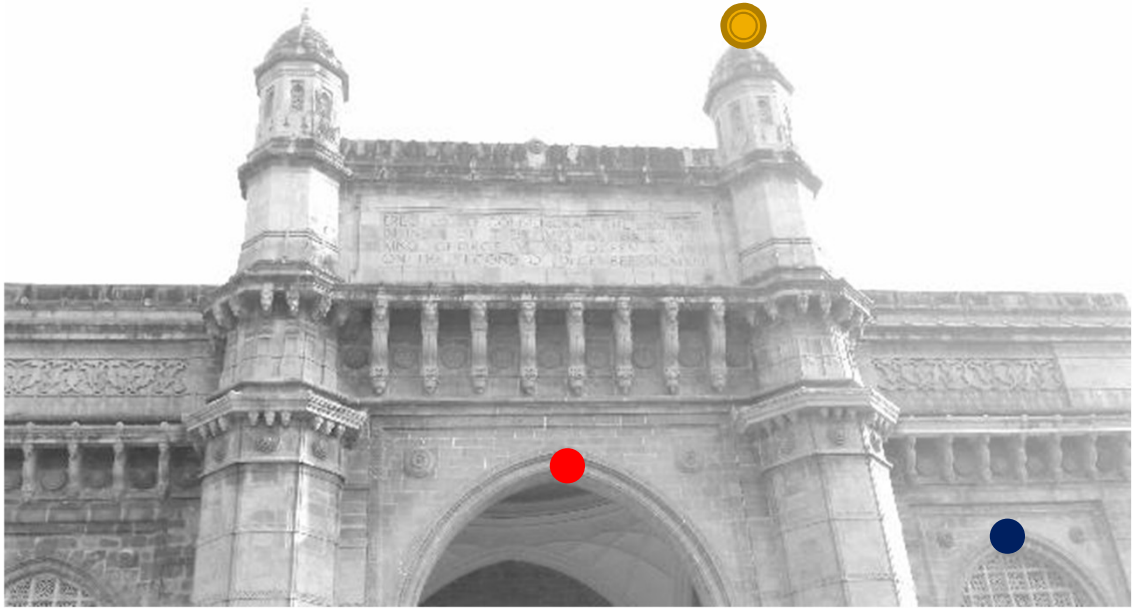


After edge-linking (hysteresis)

Corner Detection

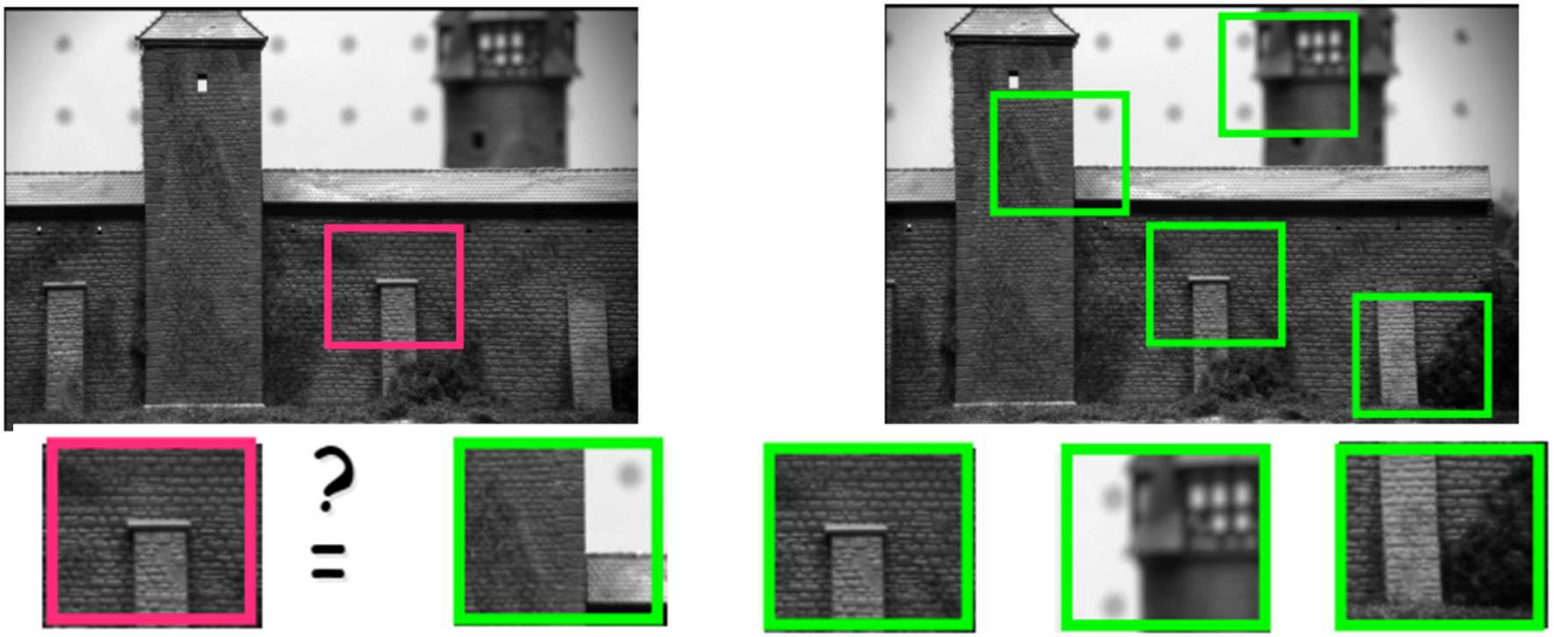
Importance of corners

- Corners are considered salient feature points.
- Useful in many tasks in computer vision and image processing
- Example: corresponding control points for image alignment

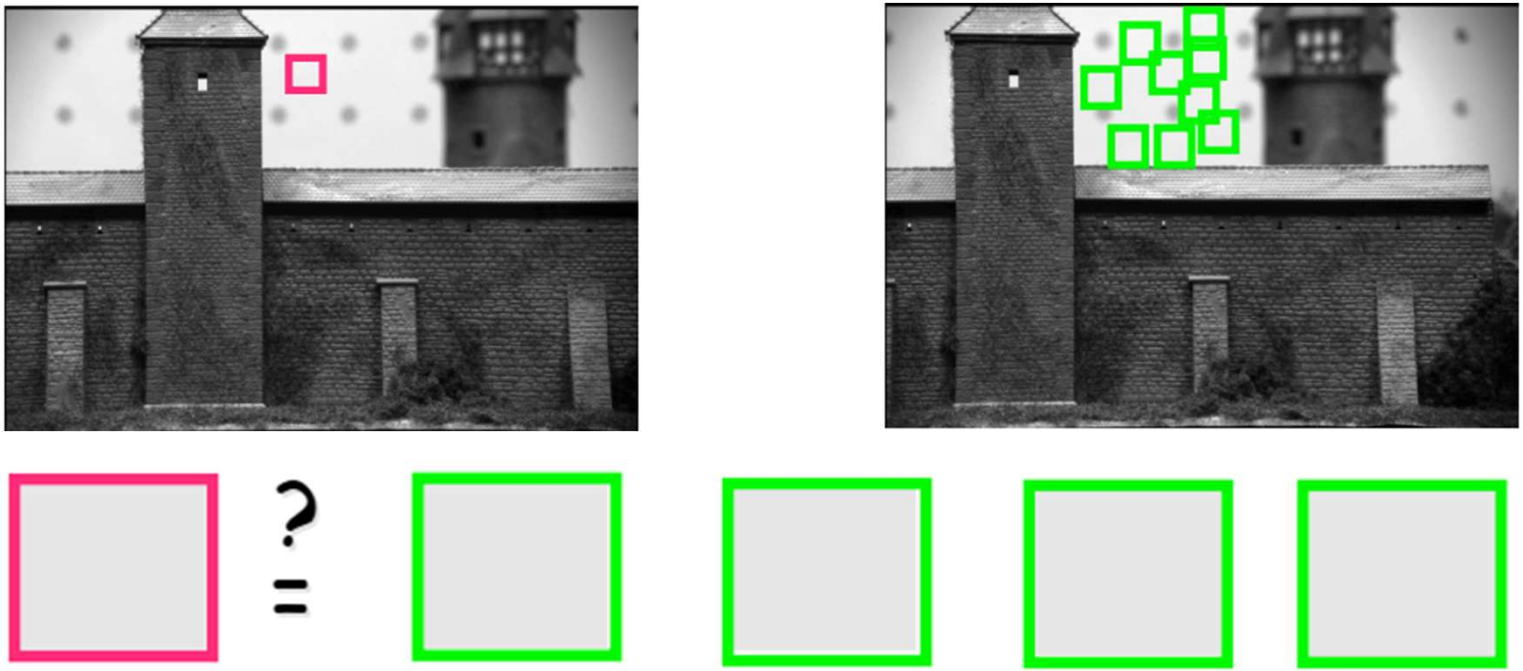


Importance of corners

- In some applications, we need to find matching patches.
- But some patch matches are easier to find than others.
  - Each patch is assumed to be centered at a certain point (say (x,y))
  - The patch encodes information about the local geometry of the image intensities about that point.
- See next slide for examples.



Good patch for matching



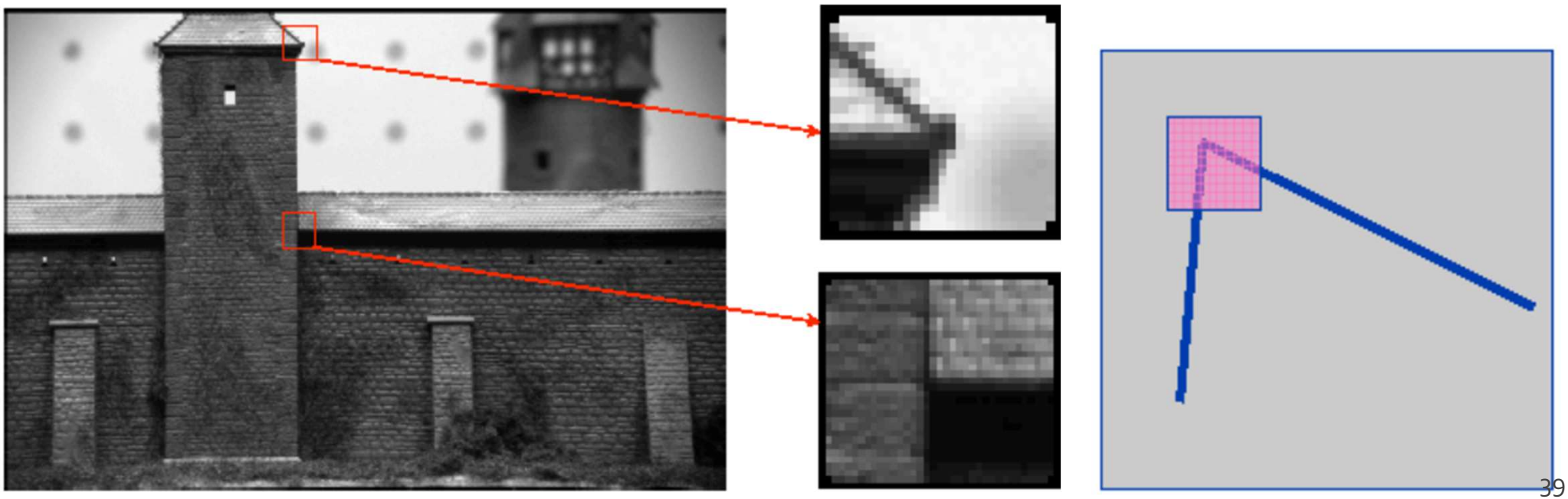
Bad patch for matching

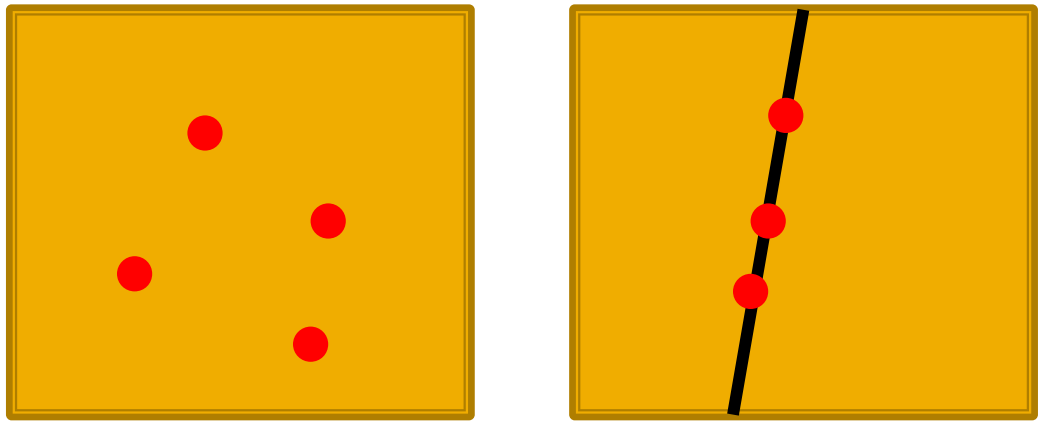
http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf

Importance of corners

- Good patches should contain corners – large intensity variations in all directions.
- Shifting the window in any direction yields large appearance change.

http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf





Ambiguity in point matching in flat intensity regions or along edges

Principle behind Harris corner detector

- Consider two patches – one each centered at (x,y) and (x+u,y+v) in two images (**patch** domain: Ω, typically rectangular, like say 7 x7 or 5 x 5) with u,v being small.

- The image have same intensity at physically corresponding locations.

- The sum of squared difference (SSD) between their intensities is given as:

$$SSD = \sum_{(x,y)\in\Omega} (I(x,y) - I(x+u, y+v))^2$$

By first order Taylor series,

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y)$$

$$SSD = \sum_{(x,y)\in\Omega} (I(x,y) - I(x+u, y+v))^2$$

By first order Taylor series,

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y)$$

$$\therefore SSD = \sum_{(x,y)\in\Omega} (uI_x(x,y) + vI_y(x,y))^2$$

$$= \begin{pmatrix} u & v \end{pmatrix} \begin{pmatrix} \sum_{x,y)\in\Omega} I_x^2(x,y) & \sum_{x,y)\in\Omega} I_x(x,y)I_y(x,y) \\ \sum_{x,y)\in\Omega} I_x(x,y)I_y(x,y) & \sum_{x,y)\in\Omega} I_y^2(x,y) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

$$= \begin{pmatrix} u & v \end{pmatrix} A \begin{pmatrix} u \\ v \end{pmatrix}$$

Structure tensor matrix (size 2 x 2)

Principle behind Harris corner detector

- The local **A** matrix on the previous slide is called the structure tensor.
- **A** carries information about local image geometry.

$$A = \lambda_1 u_1 u_1^t + \lambda_2 u_2 u_2^t; \text{eigenvals}: \lambda_1, \lambda_2; \text{eigenvecs}: u_1, u_2$$

- It is always positive semi-definite, i.e. its two eigenvalues are always non-negative.
- In locally flat regions, **A** will be close to a zero matrix and hence both its eigenvalues will be close to 0.

Proof that the structure tensor matrix is positive semi-definite: Method 1

$$A = \begin{pmatrix} \sum_{(x,y)\in\Omega} I_x^2(x,y) & \sum_{(x,y)\in\Omega} I_x(x,y)I_y(x,y) \\ \sum_{(x,y)\in\Omega} I_x(x,y)I_y(x,y) & \sum_{(x,y)\in\Omega} I_y^2(x,y) \end{pmatrix}$$

$$= \begin{pmatrix} I_x(x_1,y_1) & I_x(x_2,y_2) & . & . & I_x(x_N,y_N) \\ I_y(x_1,y_1) & I_y(x_2,y_2) & . & . & I_y(x_N,y_N) \end{pmatrix} \begin{pmatrix} I_x(x_1,y_1) & I_y(x_1,y_1) \\ I_x(x_2,y_2) & I_y(x_2,y_2) \\ . & . \\ . & . \\ I_x(x_N,y_N) & I_y(x_N,y_N) \end{pmatrix}$$

$$= ZZ^T$$

Any matrix that can be written in the form A = ZZ^T is always positive semi-definite (quoting a result from linear algebra)

Proof that the structure tensor matrix is positive semi-definite: Method 2

$$A = \begin{pmatrix} \sum_{(x,y)\in\Omega} I_x^2(x,y) & \sum_{(x,y)\in\Omega} I_x(x,y)I_y(x,y) \\ \sum_{(x,y)\in\Omega} I_x(x,y)I_y(x,y) & \sum_{(x,y)\in\Omega} I_y^2(x,y) \end{pmatrix}$$

$$trace\,(A) = \lambda_1 + \lambda_2 = \sum_{(x,y)\in\Omega} I_x^2(x,y) + \sum_{(x,y)\in\Omega} I_y^2(x,y)$$

$trace\,(A)$ is clearly non-negative as I_x^2 and I_y^2 are both non-negative

$$|A| = \sum_{(x,y)\in\Omega} I_x^2(x,y) \sum_{(x,y)\in\Omega} I_y^2(x,y) - \left(\sum_{(x,y)\in\Omega} I_x(x,y)I_y(x,y) \right)^2$$

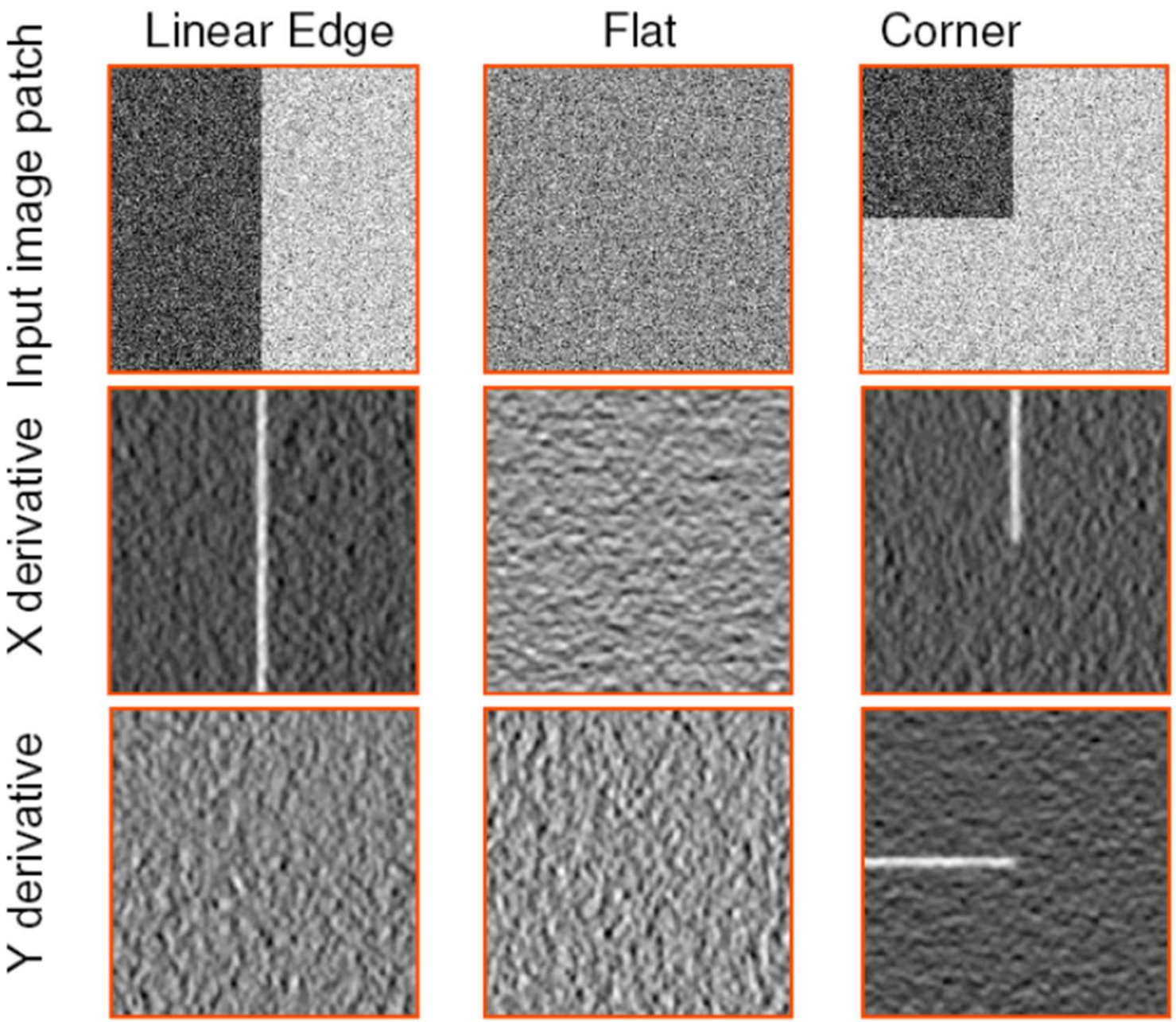
$|A| \geq 0$ by Cauchy-Schwarz inequality (the first term is $\|a\|^2\|b\|^2$, and the second term is $(a \bullet b)^2$ for N element vectors a and b, respectively containing the I_x and I_y values)

Principle behind Harris corner detector

- At a point lying on an edge, only one of the eigenvalues is large (corresponding to the eigenvector that points across the edge) and the other is close to o (corresponding to the eigenvector that points along the edge)

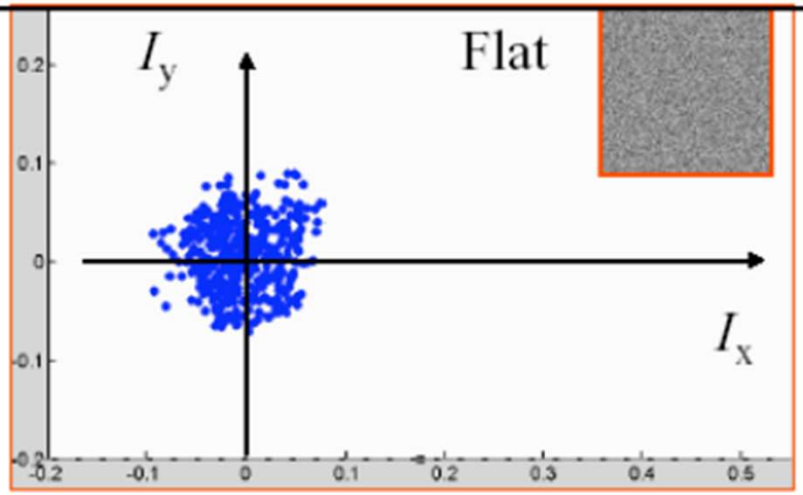
- At a corner point, both eigenvalues will be large.

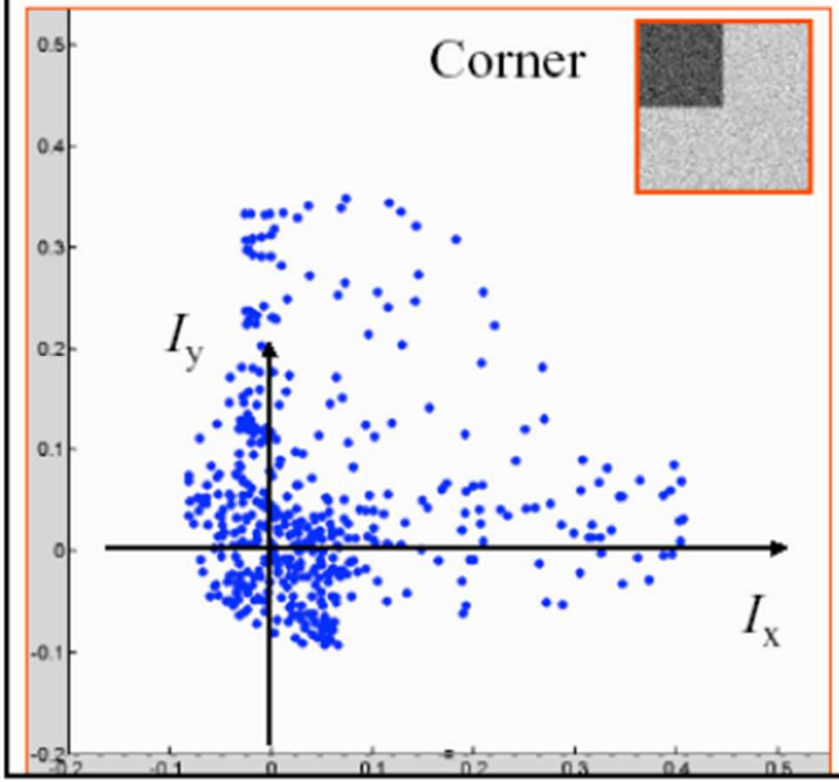
- We would like the SSD to be large for **all** non-zero shifts (u,v) so that we can allow for maximum discriminability and easier point matching.

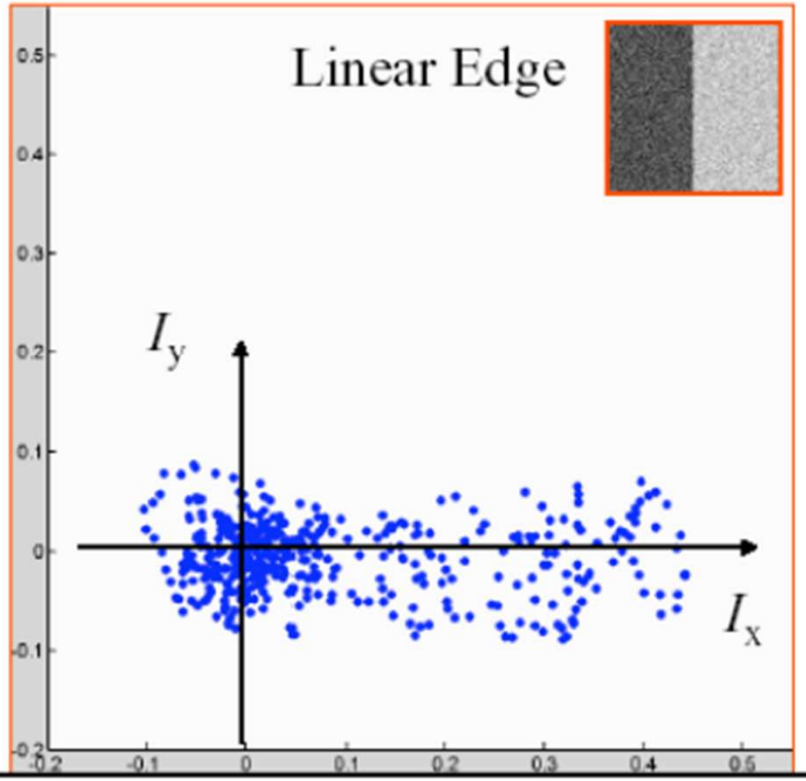


http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf

The distribution of the x and y derivatives is very different for all three types of patches

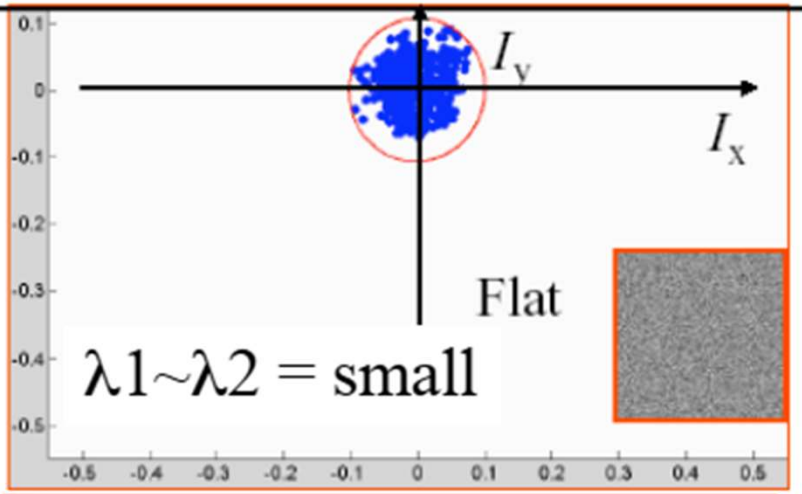


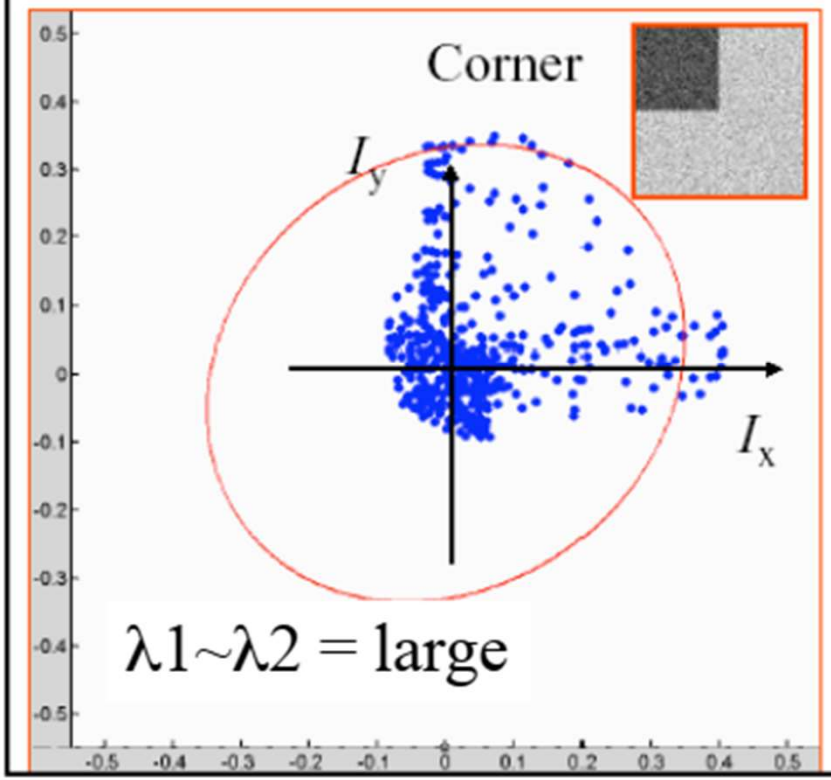


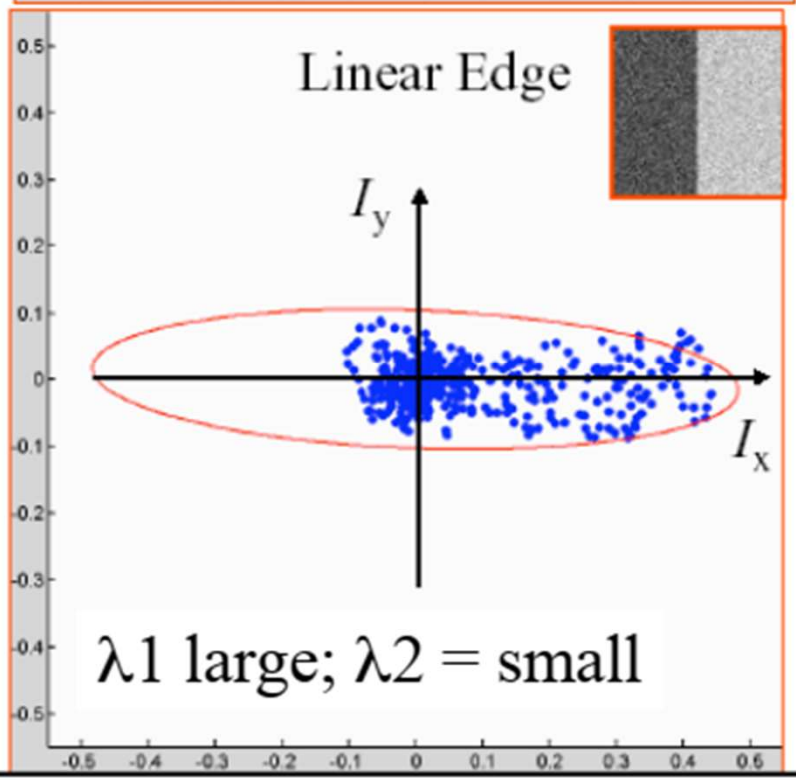


http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf

The distribution of x and y derivatives can be characterized by the shape and size of the principal component ellipse







http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf

"Corner"-ness measure

- We would like both eigenvalues of this matrix to be large.

- Corner response or "corner"ness measure: $R_H = \det(A) - k\,(\text{trace}(A))^2$, k between 0.04 to 0.06
- Does not require explicit eigenvalue/eigenvector computation:

$$|A| = \lambda_1\lambda_2 = A_{11}A_{22} - A_{12}A_{21};$$

$$trace(A) = \lambda_1 + \lambda_2 = A_{11} + A_{22}$$

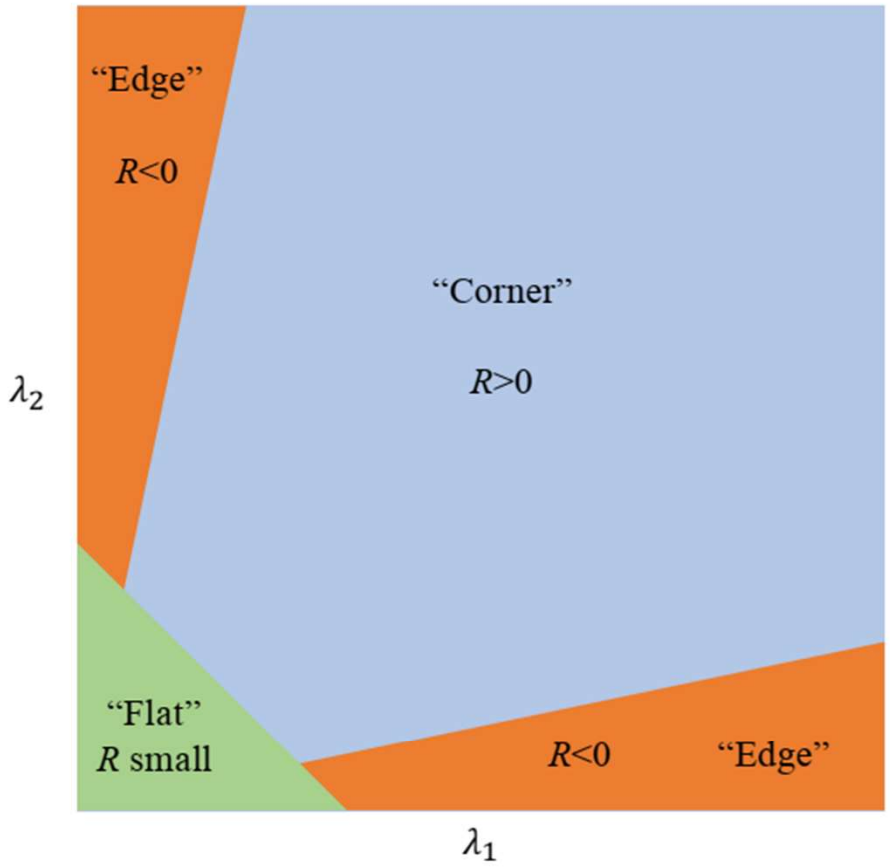
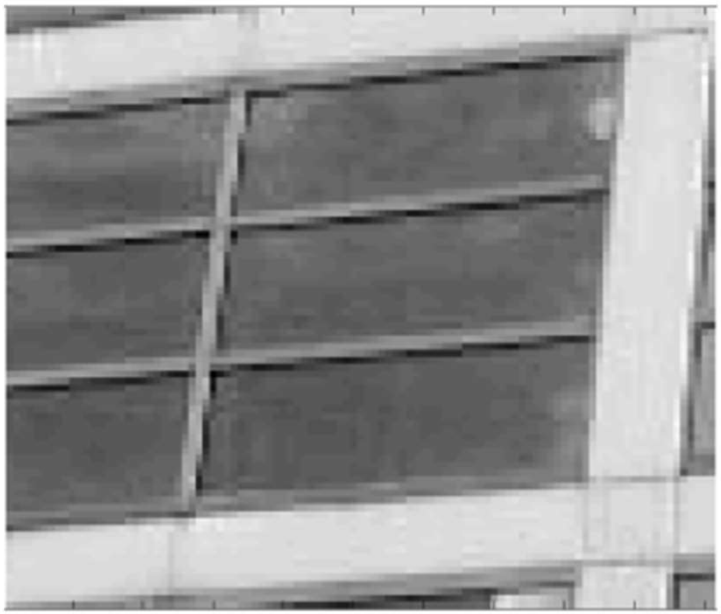
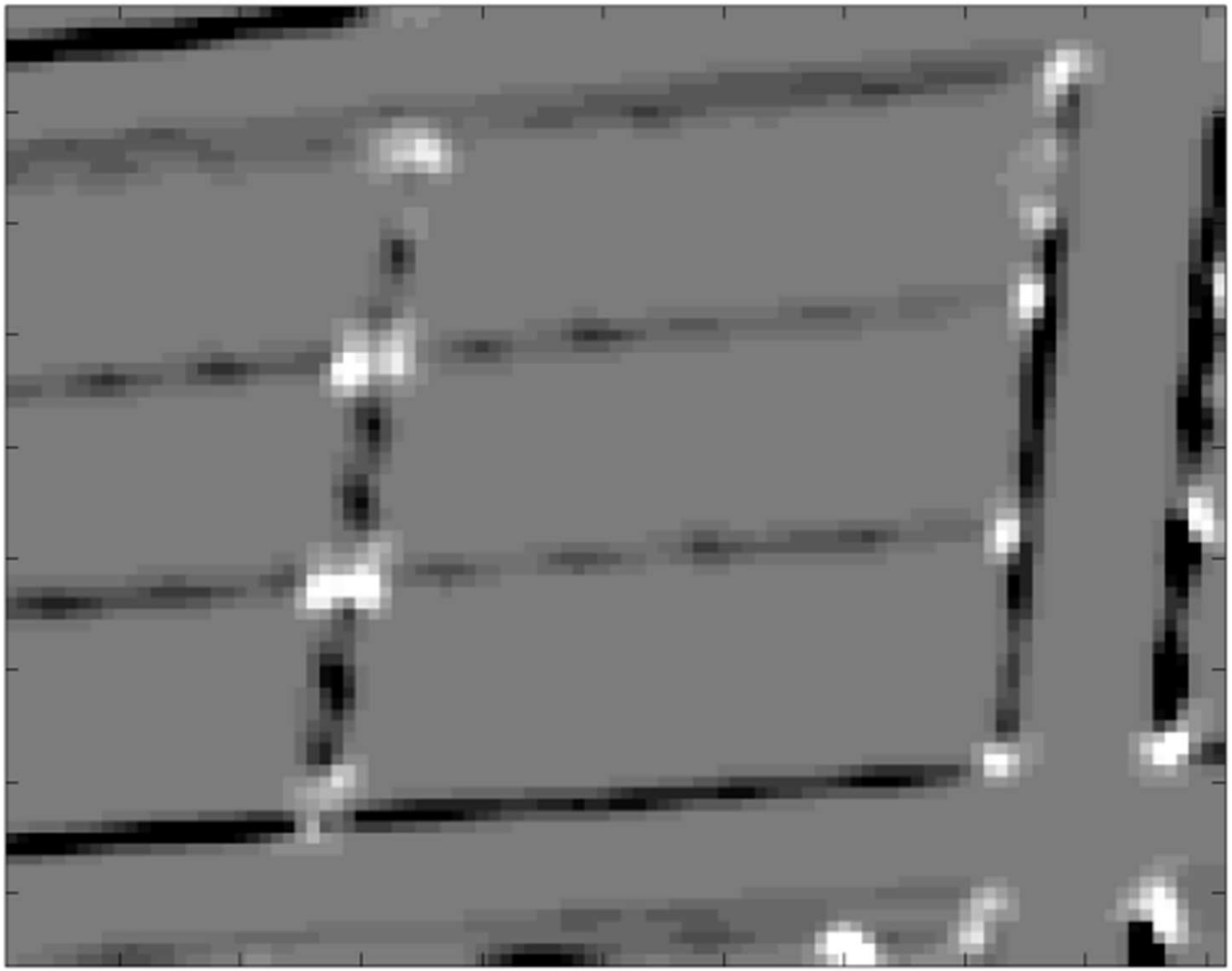


Figure 5: Harris measure. If both eigenvalues are large, then R is large and positive, providing a clue to detect a corner. If both eigenvalues are small, then R is also small and positive, which means that the point is probably part of a homogeneous region. Finally, if one of the eigenvalues is much larger than the other, R becomes negative, and the point belongs to an edge.

http://www.ipol.im/pub/art/2018/229/article_lr.pdf





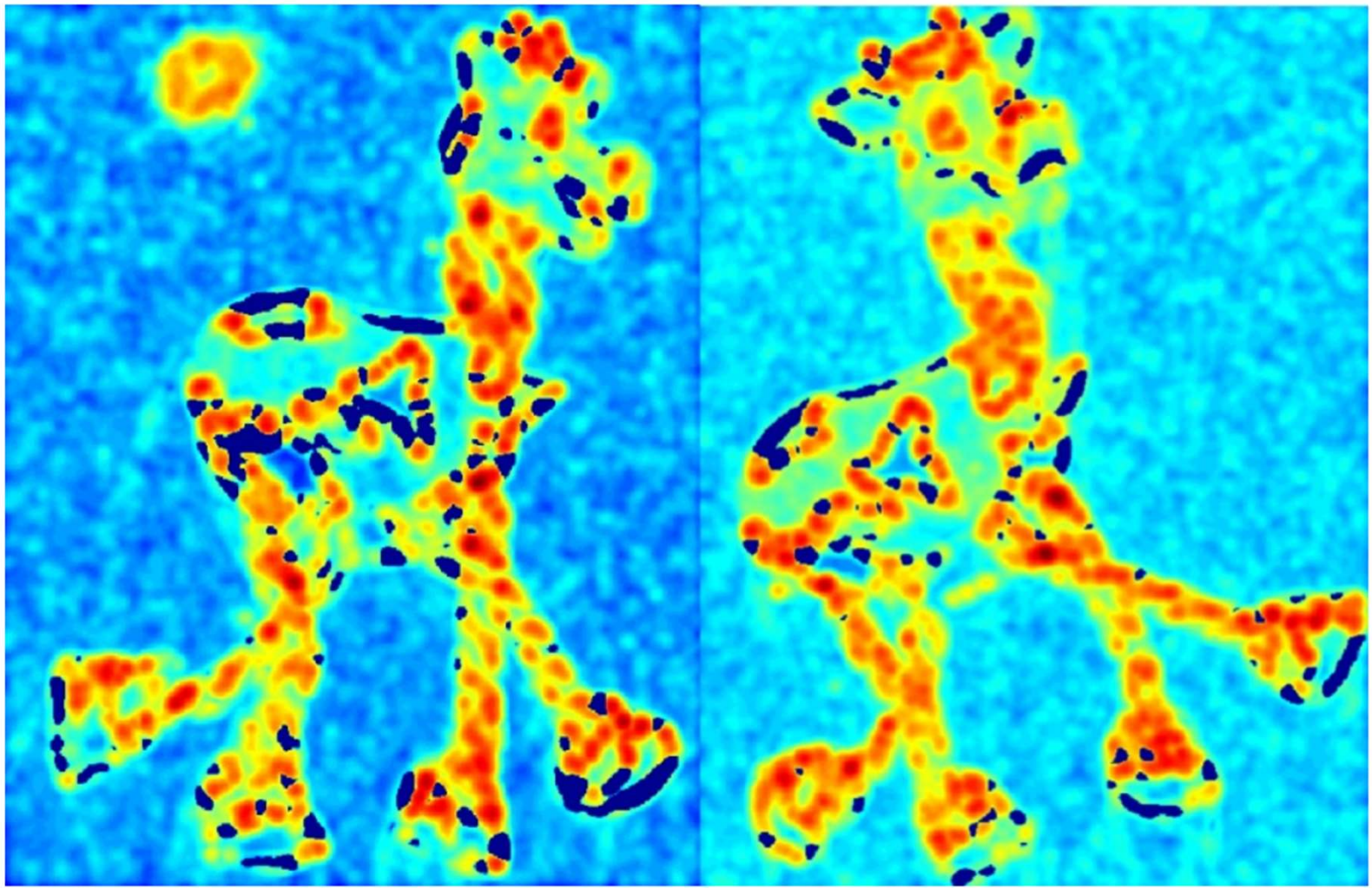
Harris R score.

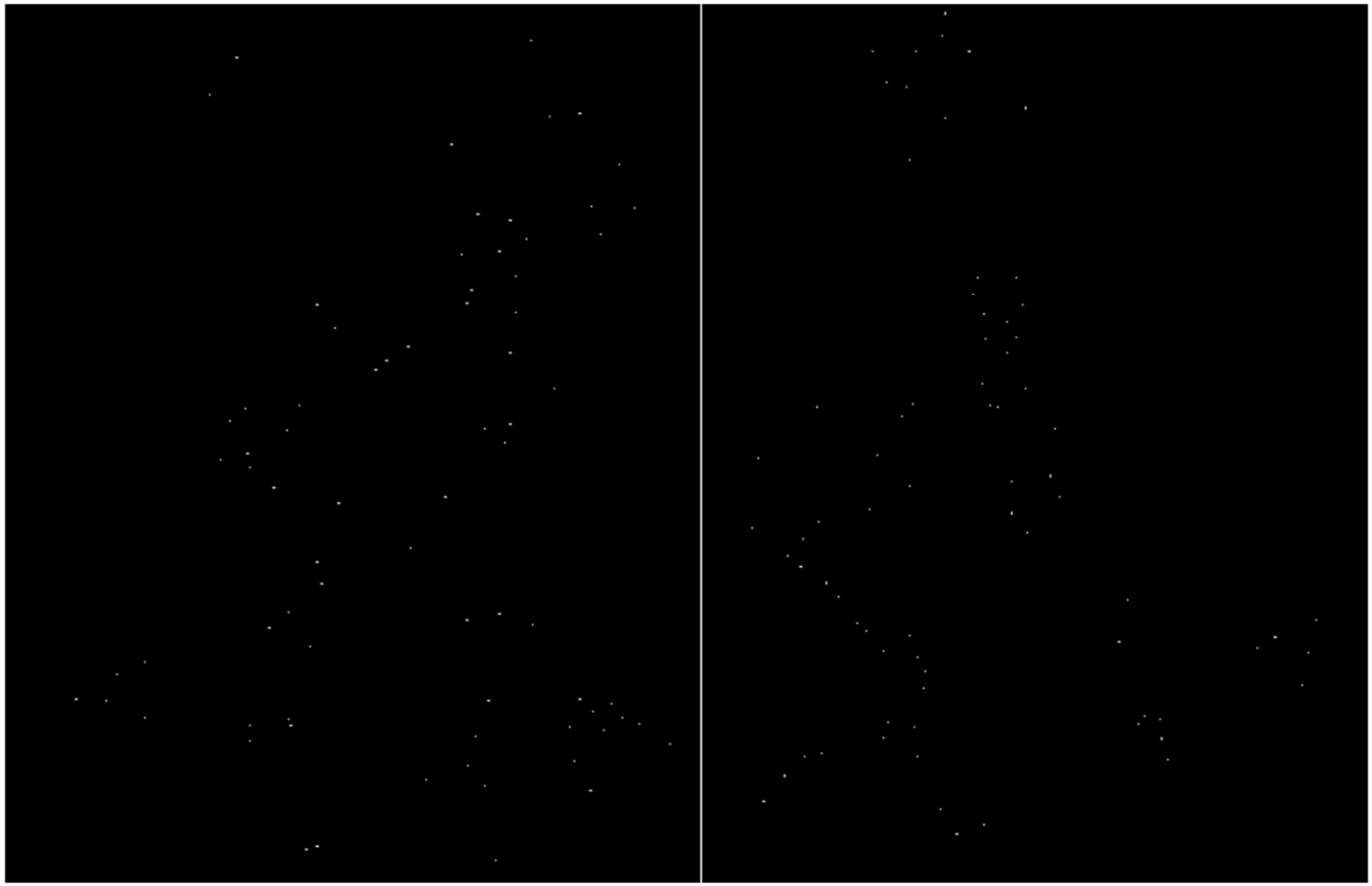
Ix, Iy computed using Sobel operator

Windowing function w = Gaussian, sigma=1

Computing corners

- Smooth image I to compute various derivatives
- Choose a window size (say 7 x 7) and compute the structure tensor at each pixel, given this window size
- Compute the corner-ness measure for each pixel
- Perform thresholding or non-maximal suppression to create a corners map.





Invariance

- The detected corners are invariant under affine intensity change, i.e. image J replaced by aJ + b for scalars a,b.

- The corners are also invariant to rotation and translation.

Other measures of corner-ness

- Based on just the minimum eigenvalue – declared a corner if the minimum eigenvalue exceeds a threshold

- Harmonic mean of eigenvalues

$$R_{HM} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{det(M)}{trace(M)}$$