OUTLINE

- Feature detection:
  - Edge detection
    - Detecting discontinuities: 2D gradient and numerical derivatives
    - Simple edge detection using gradients
    - Marr and Hildreth edge detector (Laplacian of Gaussian)
    - The Canny Edge Detector
  - Corner detection
    - Corner Points: basic idea
    - Harris detector: mathematics
    - Classification via eigenvalues
    - Corner response example

Image features: edges

- What Are Edges? discontinuities in intensity
  - Boundaries of objects
  - Boundaries of Material Properties
  - Boundaries of Lighting

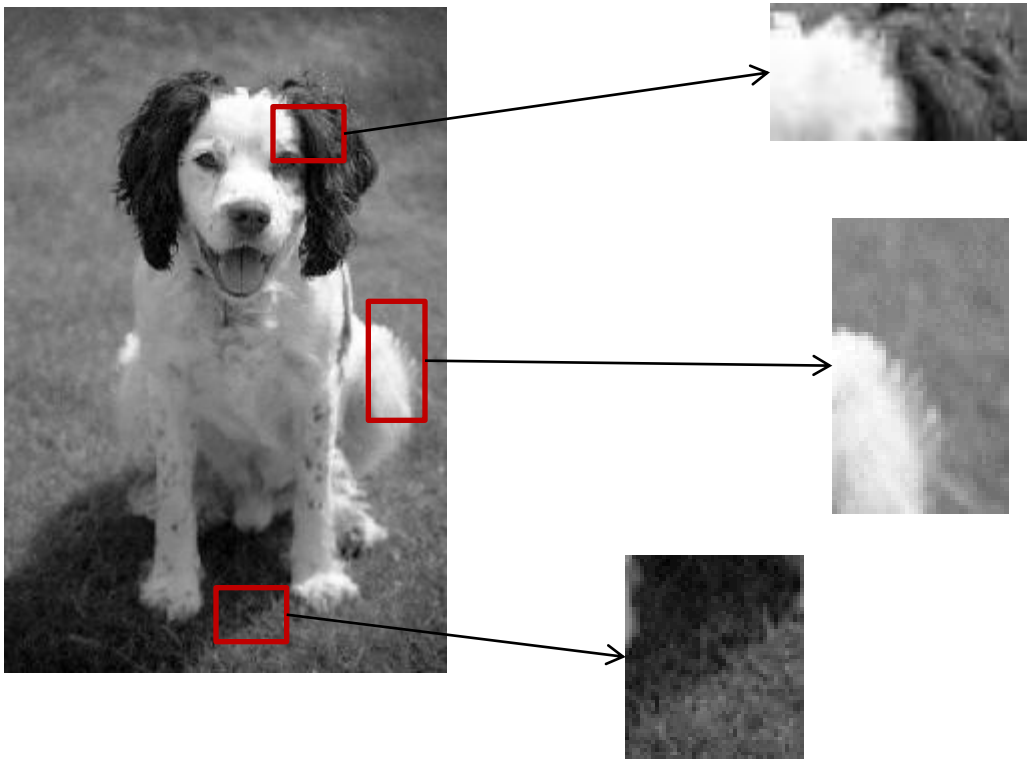
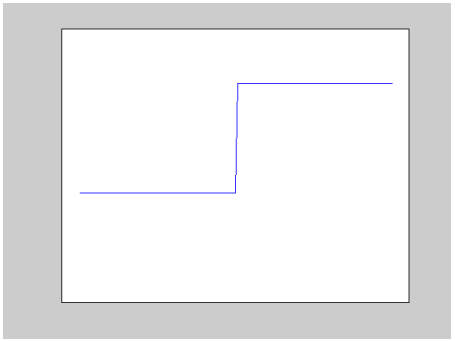


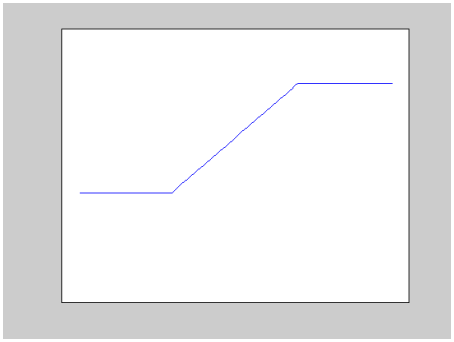
Image features: edges

- Edges can be modeled according to their intensity profiles.

Edge Models:

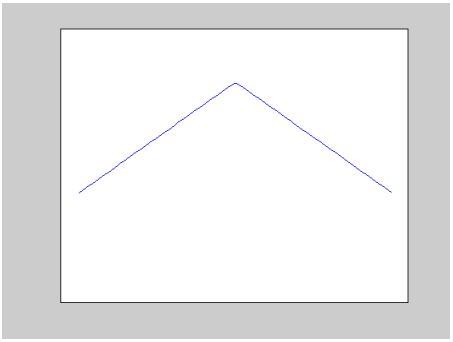
Step edge

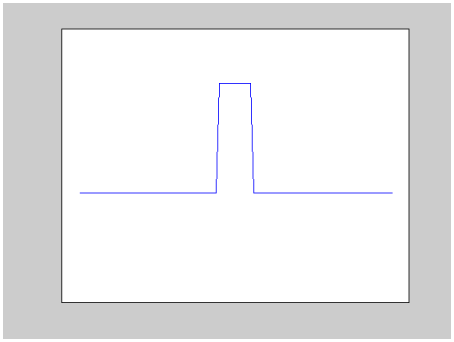




Ramp edge

Roof edge





Spike edge

Detecting discontinuities

- Discontinuities in signal can be detected by computing the derivative of the signal. In particular we compute the image gradient:

$$\nabla I(x,y) = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]^T = [I_x, I_y]^T$$

- Math Example : 2D Gradient

  f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2

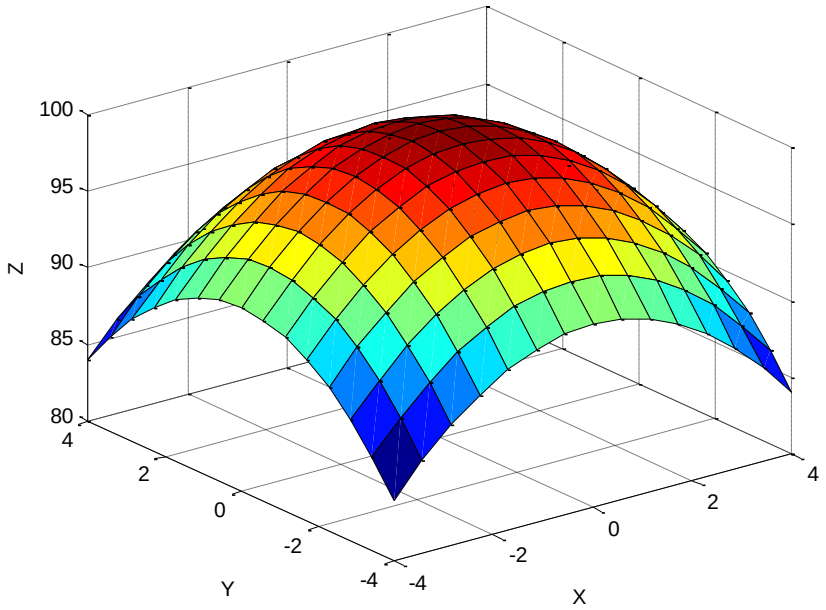
  df(x,y)/dx = - x df(x,y)/dy = - y

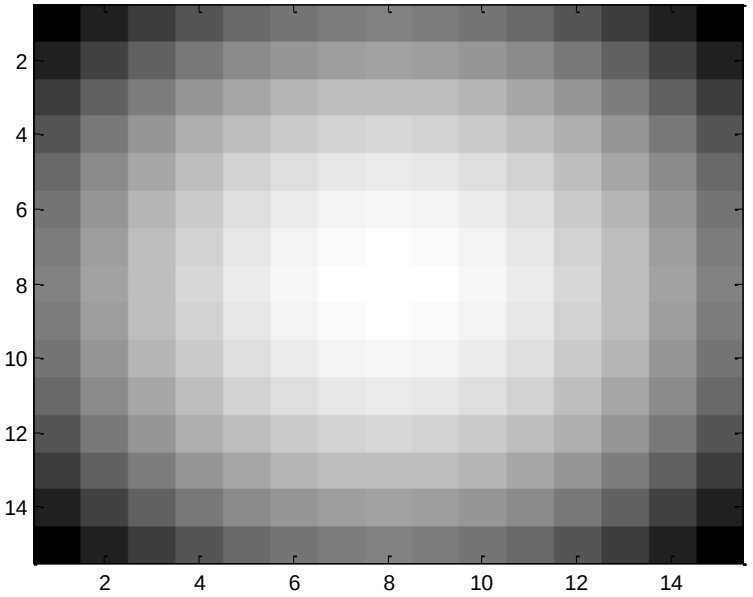
  Gradient = [df(x,y)/dx , df(x,y)/dy] = [- x , - y]

2D Gradient

- Math Example : 2D Gradient

  f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2





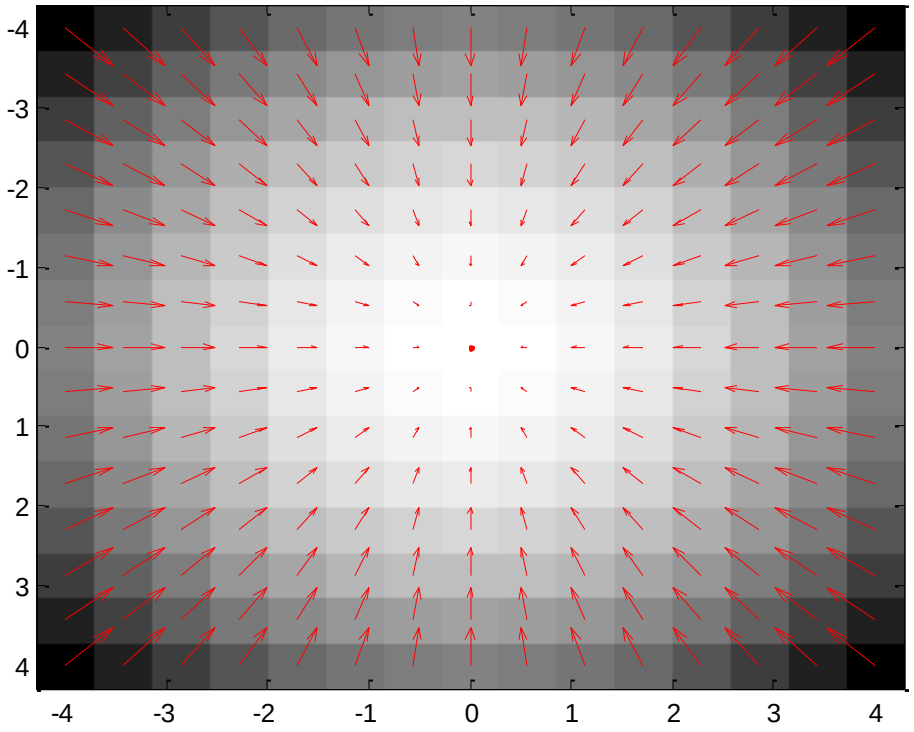
See math_2d_grad.m

2D Gradient

- Math Example : 2D Gradient

  f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2

  Gradient = [df(x,y)/dx , df(x,y)/dy] = [- x , - y]



Plotted as a **vector field**, the gradient vector at each pixel points “uphill”

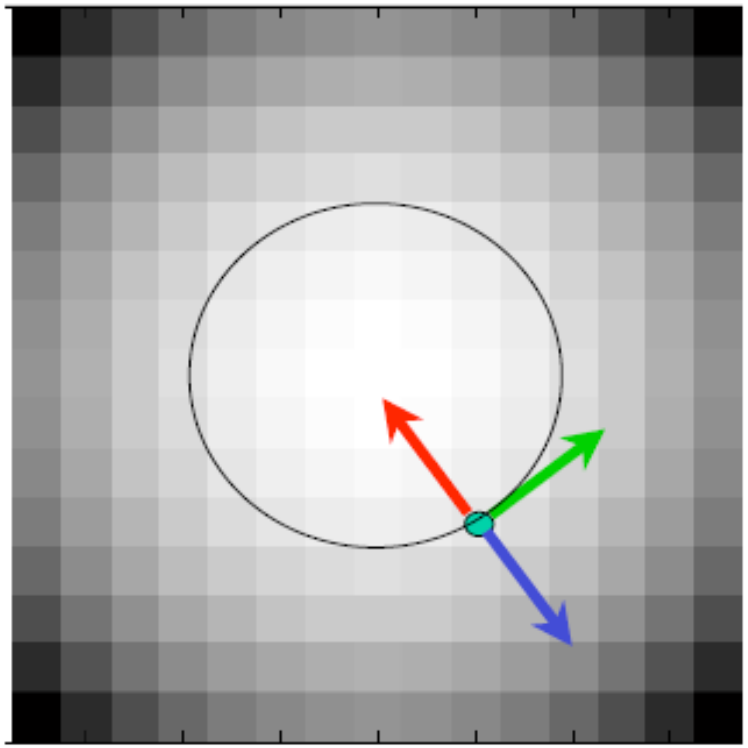
The gradient indicates the direction of steepest ascent.

2D Gradient

- Math Example : 2D Gradient

  f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2

  Gradient = [df(x,y)/dx , df(x,y)/dy] = [- x , - y]



Let $g=[g_x,g_y]$ be the gradient vector at point/pixel (x_0,y_0)

Vector g points uphill
(direction of steepest ascent)

Vector - g points downhill
(direction of steepest descent)

Vector $[g_y, -g_x]$ is perpendicular, and denotes direction of constant elevation. i.e. normal to contour line passing through point (x_0,y_0)

Numerical Derivatives

- The same is true of 2D image gradients.
- The underlying function is numerical (tabulated) rather than algebraic. So numerical derivatives need.

Finite forward difference

$$\frac{f(x+h)-f(x)}{h} = f'(x) + O(h)$$

Finite backward difference

$$\frac{f(x)-f(x-h)}{h} = f'(x) + O(h)$$

Finite central difference

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + O(h^2)$$

More accurate

Numerical Derivatives

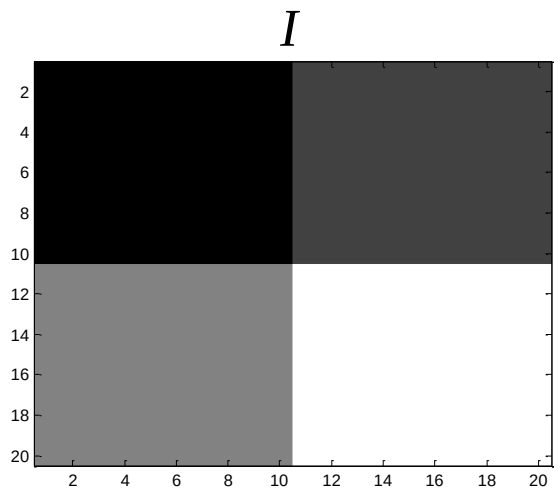
- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters.

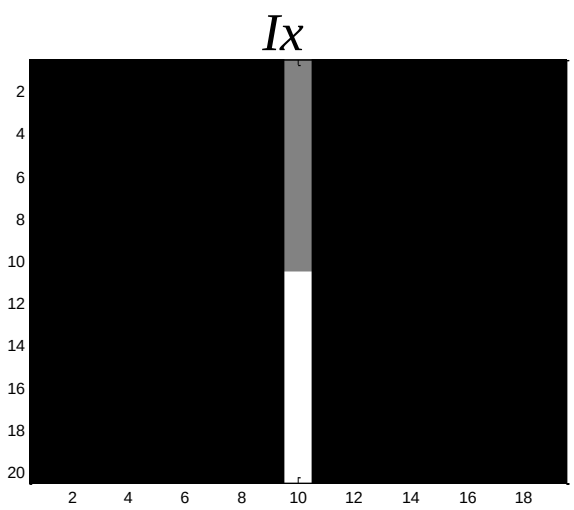
- By considering the finite forward difference and h=1 for images, we have:

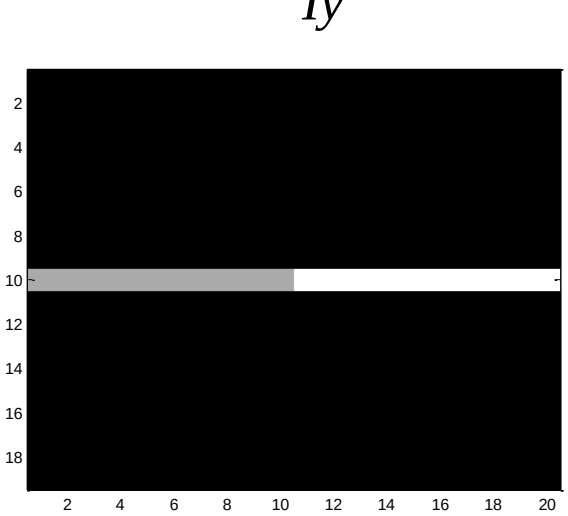
  the partial derivative with respect to rows as img(i+1,j) – img(i,j)),

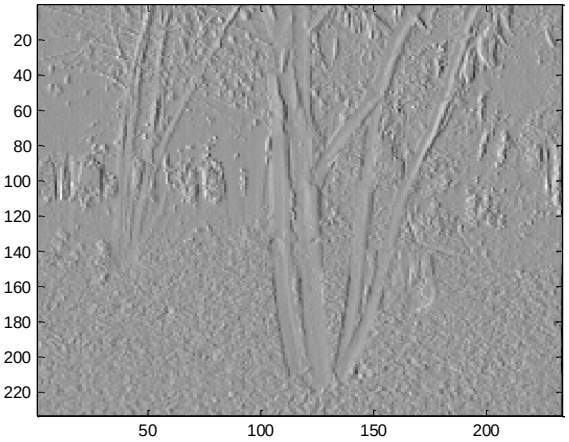
  which can be considered as a convolution with the kernel [1 , -1]

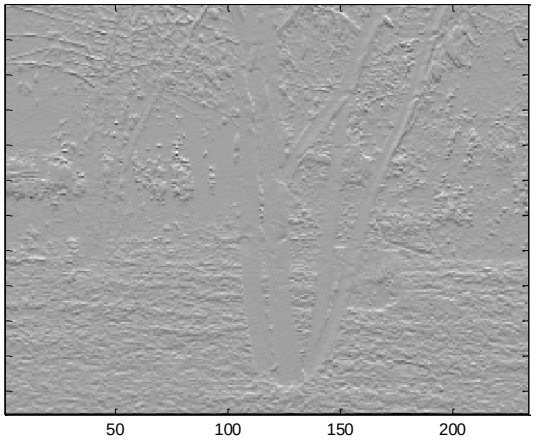
Detecting discontinuities







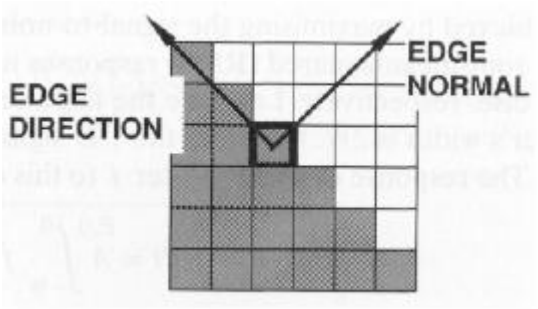




See der.m

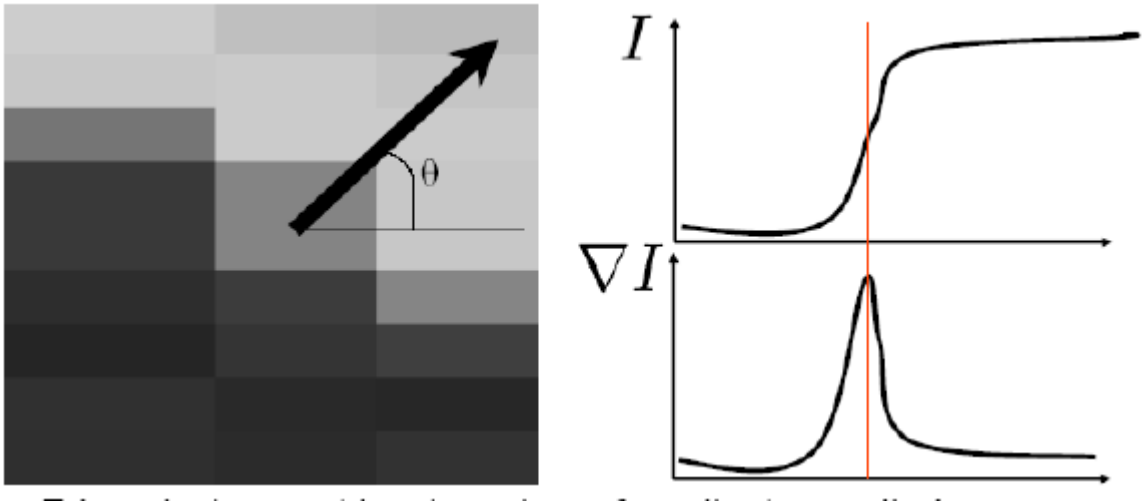
Edge Terminology

- Edge descriptors
  - Edge normal: unit vector in the direction of maximum intensity change.
  - Edge direction: unit vector along edge (perpendicular to edge normal).
  - Edge position/center: the image position at which the edge is located.
  - Edge strength or magnitude: local image contrast along the normal.



- All of this information can be computed from the gradient vector field

Summary of Gradients



Edge pixels are at local maxima of gradient magnitude
Gradient direction is always perpendicular to edge direction

Gradient Vector: $\nabla I = \left[\frac{\partial I}{\partial x} , \frac{\partial I}{\partial y} \right]^T$

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

Orientation $\theta = atan2(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x})$

Simple Edge Detection Using Gradients

- A simple edge detector using gradient magnitude:

1. Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters
2. Compute gradient magnitude at each pixel
3. If magnitude at a pixel exceeds a threshold, report a possible edge point.

Simple Edge Detection Using Gradients

- A primary problem in edge detection is *image noise*.
  - This is because edge detectors are constructed to respond strongly to sharp changes.
  - The noise values at each pixel are typically uncorrelated, meaning they can be very different.

- A solution is to smooth the image, then to differentiate it.
  - *Smoothing an image and then differentiating, it is the same as convolving it with the derivative of a smoothing kernel.*
  - Differentiation is linear and shift invariant, and convolution is associative.

Simple Edge Detection Using Gradients

- Issue: the noise
  - smooth before differentiation;
  - two convolutions: to smooth, then to differentiate;
  - actually, we can use the derivative of the kernel.

For instance, *Prewitt's operator*:

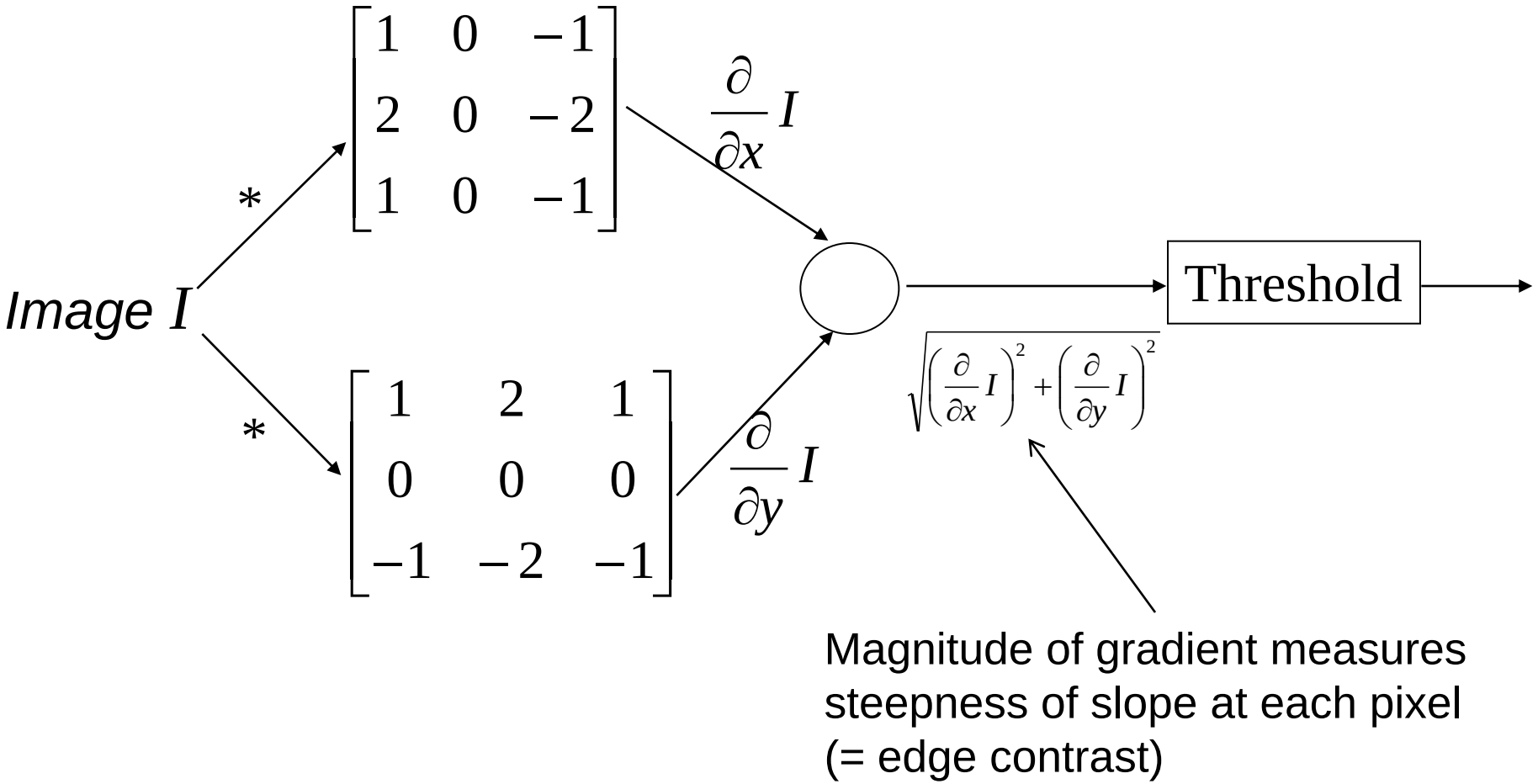
$$\longrightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & -1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Smooth Differentiate

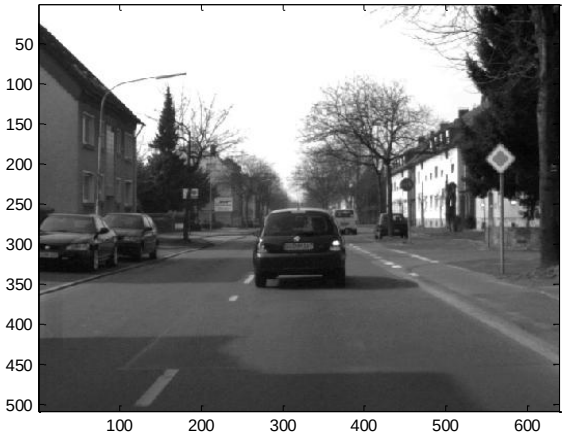
$$\longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Simple Edge Detection Using Gradients

- *Sobel edge detector*:

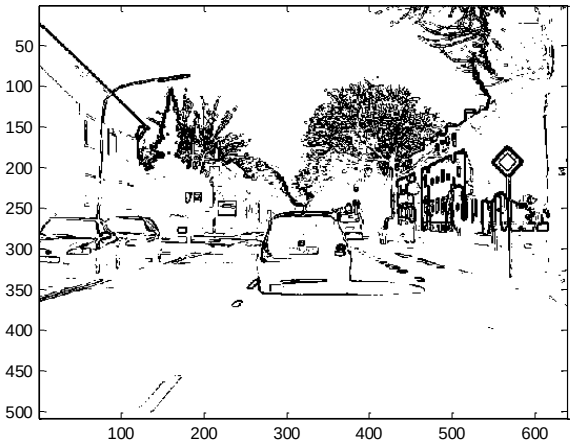


Simple Edge Detection Using Gradients



image

Magnitude map

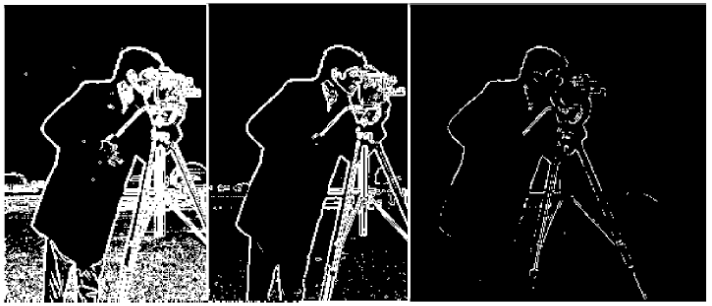


Edge map

See sobel.m

Issues to Address

- How should we choose the threshold?



- Trade-off: smoothing vs localization (*smoothed derivative removes noise, but blurs edge*)

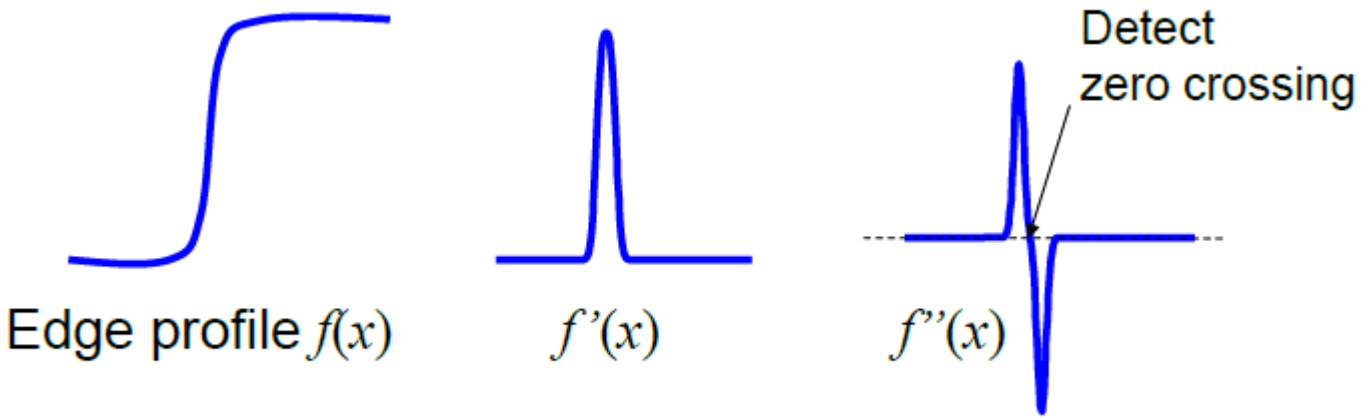
- Edge thinning and linking (*smoothing and thresholding gives us a binary images with "thick" edges*)

Marr and Hildreth edge detector

- Detect edges by considering second derivative

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

- Zero-crossings mark edge location



Marr and Hildreth edge detector

- (this approach, while historically important, is no longer popular)
  - Smooth by Gaussian

$$S = G_\sigma * I \qquad G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

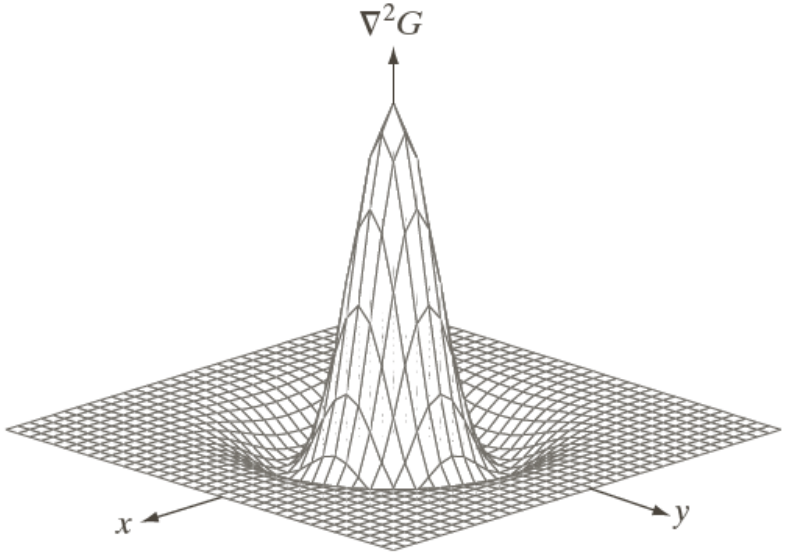
  - Use Laplacian of Gaussian (LoG) to find edges

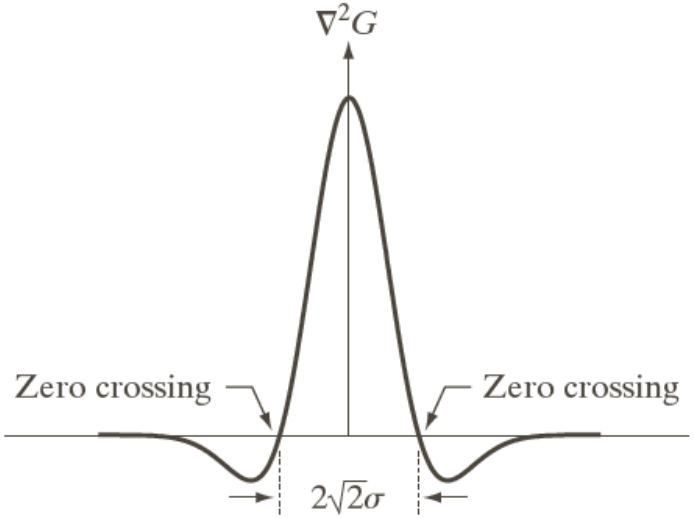
$$\Delta^2 S = \frac{\partial^2}{\partial x^2} S + \frac{\partial^2}{\partial y^2} S$$

$$\Delta^2 S = \Delta^2 \left(G_\sigma * I \right) = \Delta^2 G_\sigma * I$$

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

Marr and Hildreth edge detector

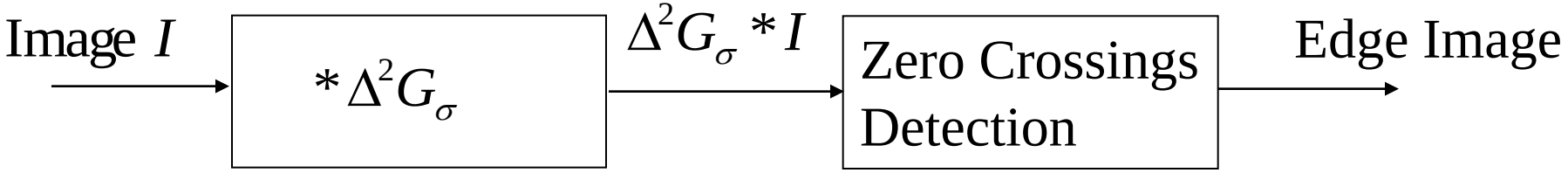




0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

See laplacian.m

Marr and Hildreth edge detector



- There are four cases of zero crossing: {+,-}, {+,0,-}, {-,+}, and {-,0,+}.
- In order to avoid weak zero crossing due to noise, a threshold is applied to the slope: e.g. for {a,-b} is |a+b|.

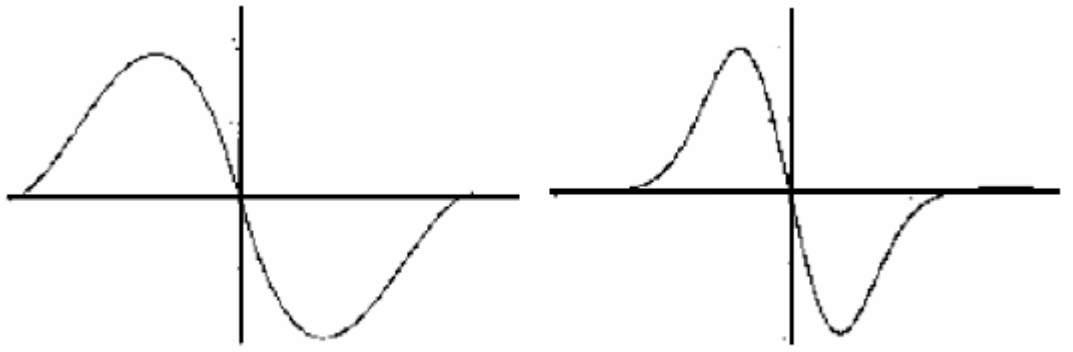
The Canny Edge Detector

- Optimal for step edges corrupted by white noise.

- The Objective
  1. *Low error rate*
     The edges detected must be as close as possible to the true edge
  2. *Edge points should be well localized*
     The edges located must be as close as possible to the true edges
  3. *Single edge point response*
     The number of local maxima around the true edge should be minimum

The Canny Edge Detector

- Canny found a linear, continuous filter that maximized the three given criteria.
- There is no closed-form solution for the optimal filter.
- However, it looks very similar to the derivative of a Gaussian.



Canny

Derivative of Gaussian

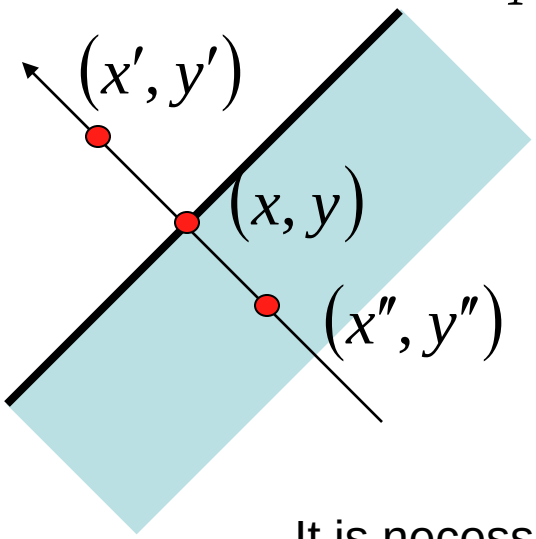
The Canny Edge Detector

- Thus, we filter the image with the derivative of Gaussian along the x axis and the y axis, then we perform the computation of the image gradient.
- The choice of the *standard deviation of Gaussian* is often called the *scale of the smoothing*
- Assume we have a narrow bar on a constant background:
  - smoothing on a scale smaller than the width of the bar will mean that the filter responds on each side of the bar, and we will be able to resolve the rising and falling edges of the bar.
  - If the filter width is much greater, the bar will be smoothed into the background, and the bar will generate little or no response

The Canny Edge Detector: thinning

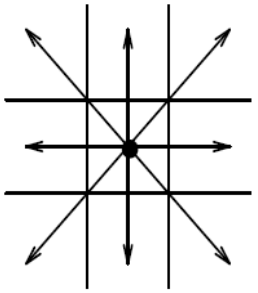
- Suppress the pixels in the gradient magnitude map which are not local maximum (nonmaxima suppression).

$$NMS(x,y)=\begin{cases}|\nabla I|(x,y) & \text{if } |\nabla I|(x,y)>|\nabla I|(x',y') \\ & \&\,|\nabla I|(x,y)>|\nabla I|(x'',y'') \\ 0 & \text{otherwise}\end{cases}$$



(x',y') and (x'',y'') are the neighbors of (x,y) in $|\nabla I|$ along the direction θ normal to an edge

It is necessary to quantize the gradient direction into a fixed number of direction



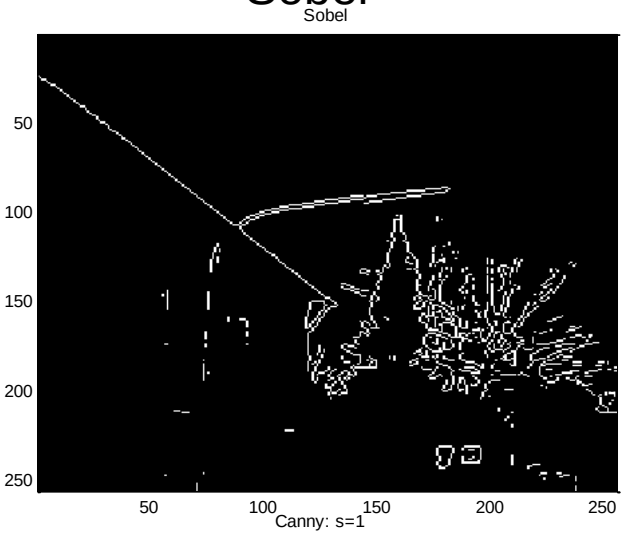
The Canny Edge Detector: Hysteresis Thresholding

- Keep both a high threshold H and a low threshold L.
- Any edges with strength < L are discarded.
- Any edge with strength > H are kept.
- An edge P with strength between L and H is kept only if there is a path of edges with strength > L connecting P to an edge of strength > H.
- In practice, this thresholding is combined with edge linking to get connected contours.

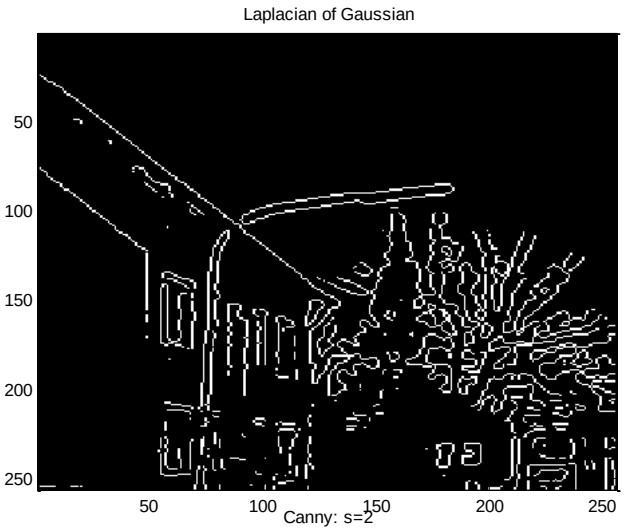
edges

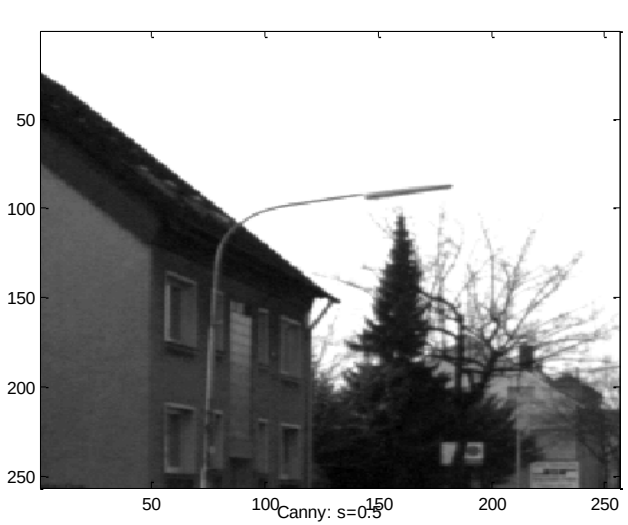
See ex_edge.m

Sobel

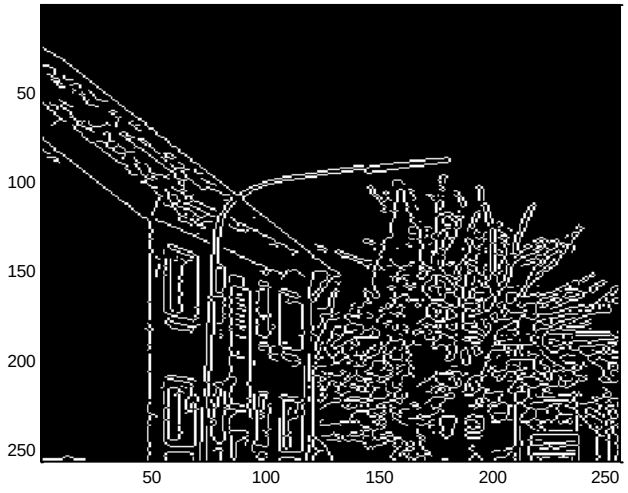


Laplacian of Gaussian

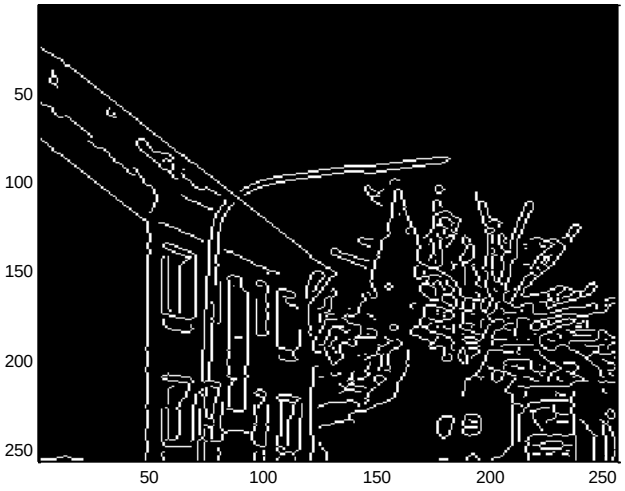




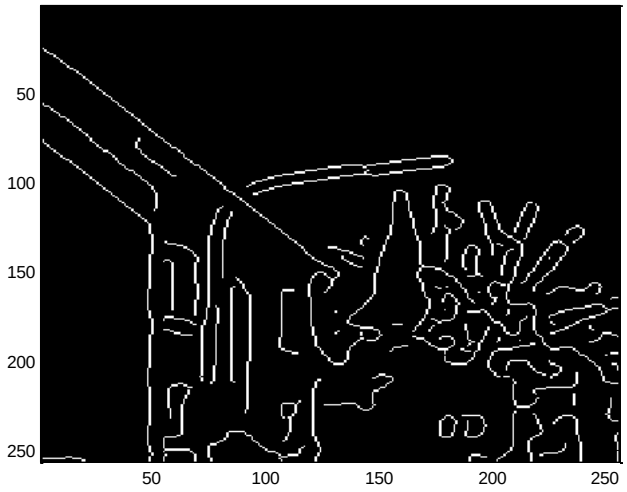
Canny: s=0.5



Canny: s=1

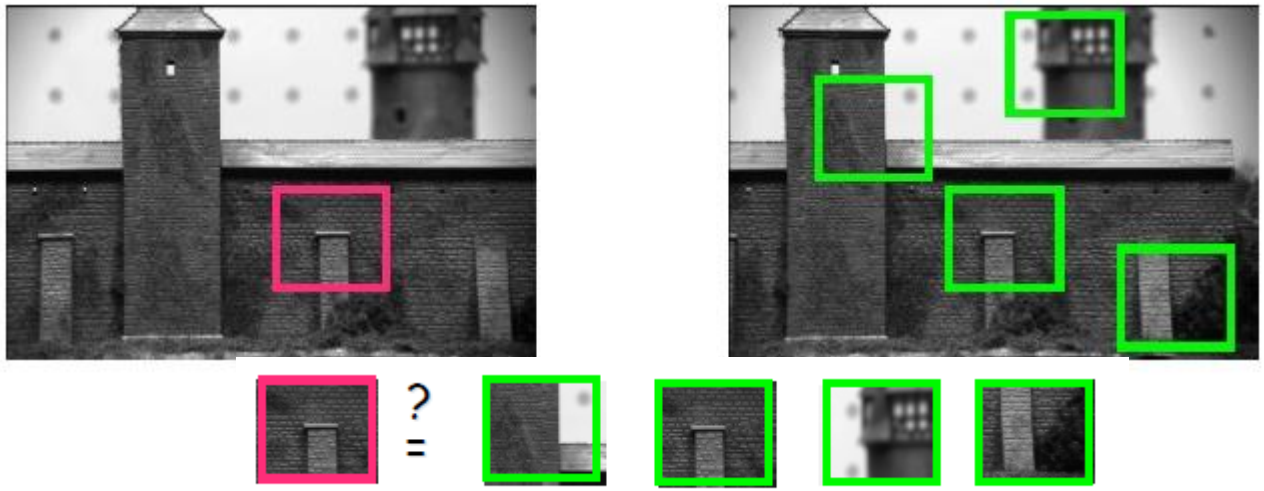


Canny: s=2



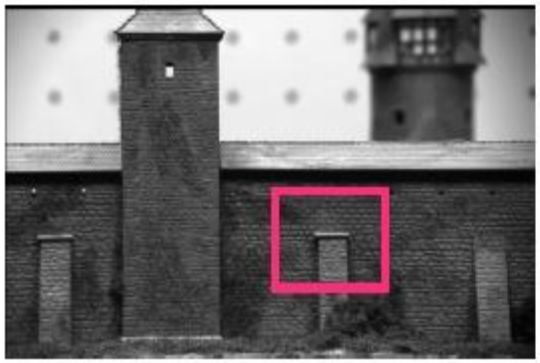
Corner detection

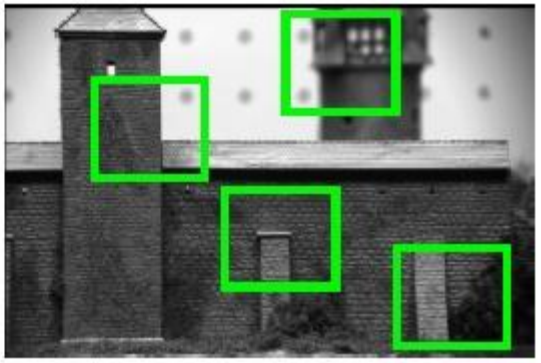
- Motivation: vision tasks such as stereo and motion estimation require finding corresponding features across two or more views.
- Patch Matching: elements to be matched are image patches of fixed size



- Task: find the best (most similar) patch in a second image

Corner detection

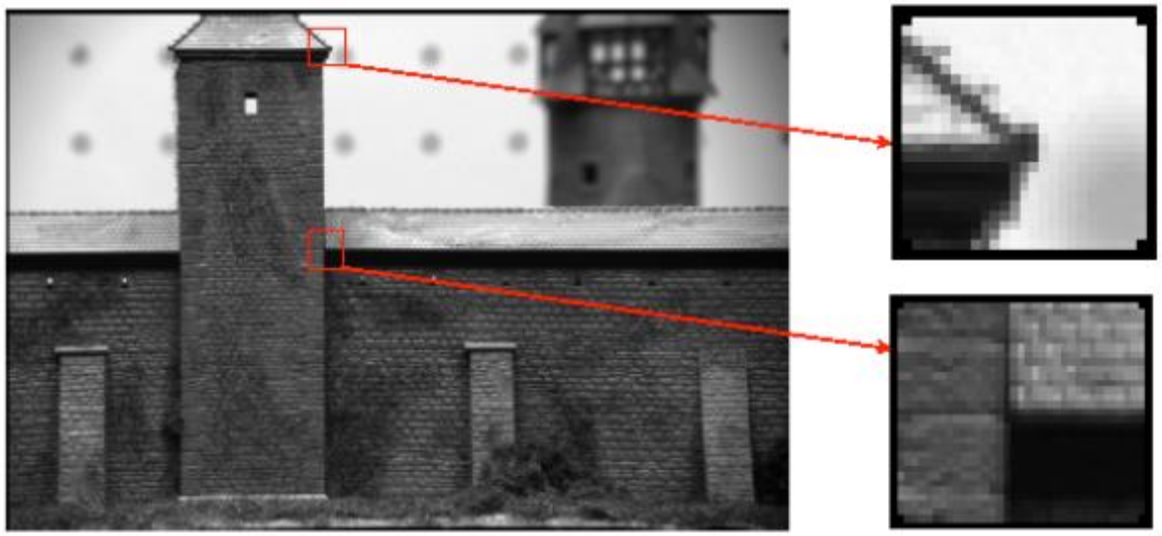




This would be a good patch for matching, since it is very distinctive (there is only one patch in the second frame that looks similar).

This would be a bad patch for matching, since it is not very distinctive (there are many similar patches in the second frame)

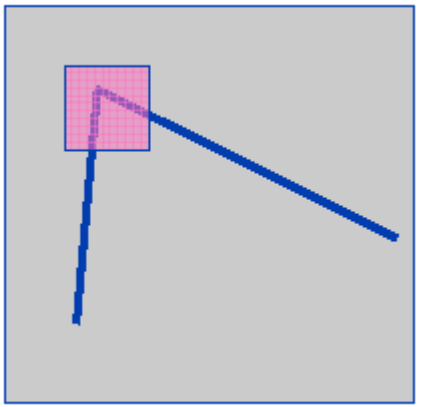
What are Corners?



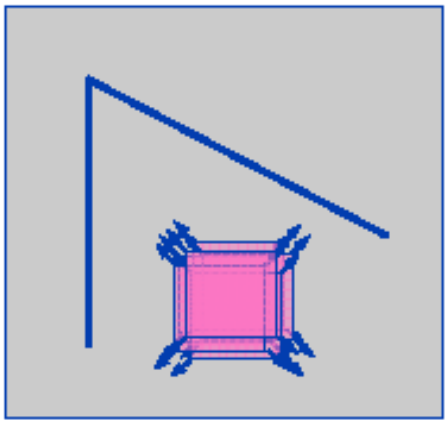
- Intuitively, junctions of contours
- Generally, more stable feature over changes of viewpoints
- Intuitively, large variations in the neighborhood of the point in all directions
- *They are good features to match*

Corner Points: Basic Idea

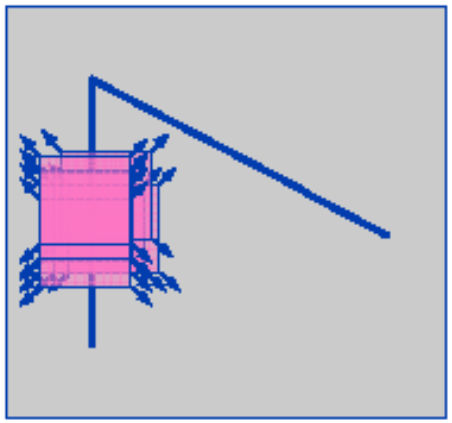
- We should easily recognize the point by looking at intensity values within a small window
- Shifting the window in *any direction* should yield a *large change* in appearance.



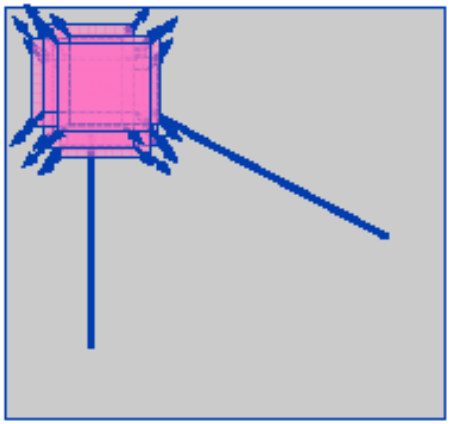
Harris Corner Detector: Basic Idea



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction



"corner":
significant change
in all directions

Harris corner detector gives a mathematical approach for determining which case holds

Harris Detector: Mathematics

- Change of intensity for the shift (*u*,*v*):

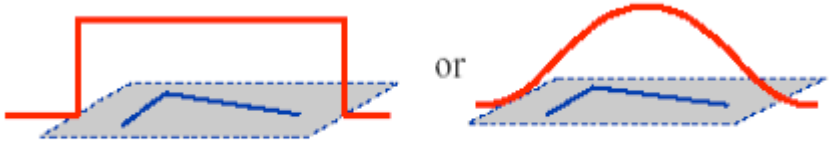
Shifted intensity

Intensity

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

For nearly constant patches, this will be near 0.
For very distinctive patches, this will be larger.
Hence, we want patches where E(u,v) is LARGE.

Window function



1 in window, 0 outside Gaussian

Taylor Series for 2D Functions

$$f(x+u,y+v) = f(x,y) + uf_x(x,y) + vf_y(x,y) +$$

First partial derivatives

$$\frac{1}{2!}\left[u^2 f_{xx}(x,y) + uvf_{xy}x,y + v^2 f_{yy}(x,y)\right] +$$

Second partial derivatives

$$\frac{1}{3!}\left[u^3 f_{xxx}(x,y) + u^2 v f_{xxy}(x,y) + uv^2 f_{xyy}(x,y) + v^3 f_{yyy}(x,y)\right]$$

Third partial derivatives

$+ \ldots$ (Higher order terms)

First order approx

$$f(x+u,y+v) \approx f(x,y) + uf_x(x,y) + vf_y(x,y)$$

Harris Corner Derivation

$$\sum [I(x+u, y+v) - I(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x + vI_y - I(x,y)]^2 \qquad \text{First order approx}$$

$$= \sum u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2$$

$$= \sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \qquad \text{Rewrite as matrix equation}$$

$$= \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

This matrix characterizes the structure of the image (of the grey levels)

Harris Corner Derivation

$$E(u,v) \cong [u,v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2x2 matrix computed from image derivatives

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Windowing function - computing a weighted sum (simplest case, w=1)

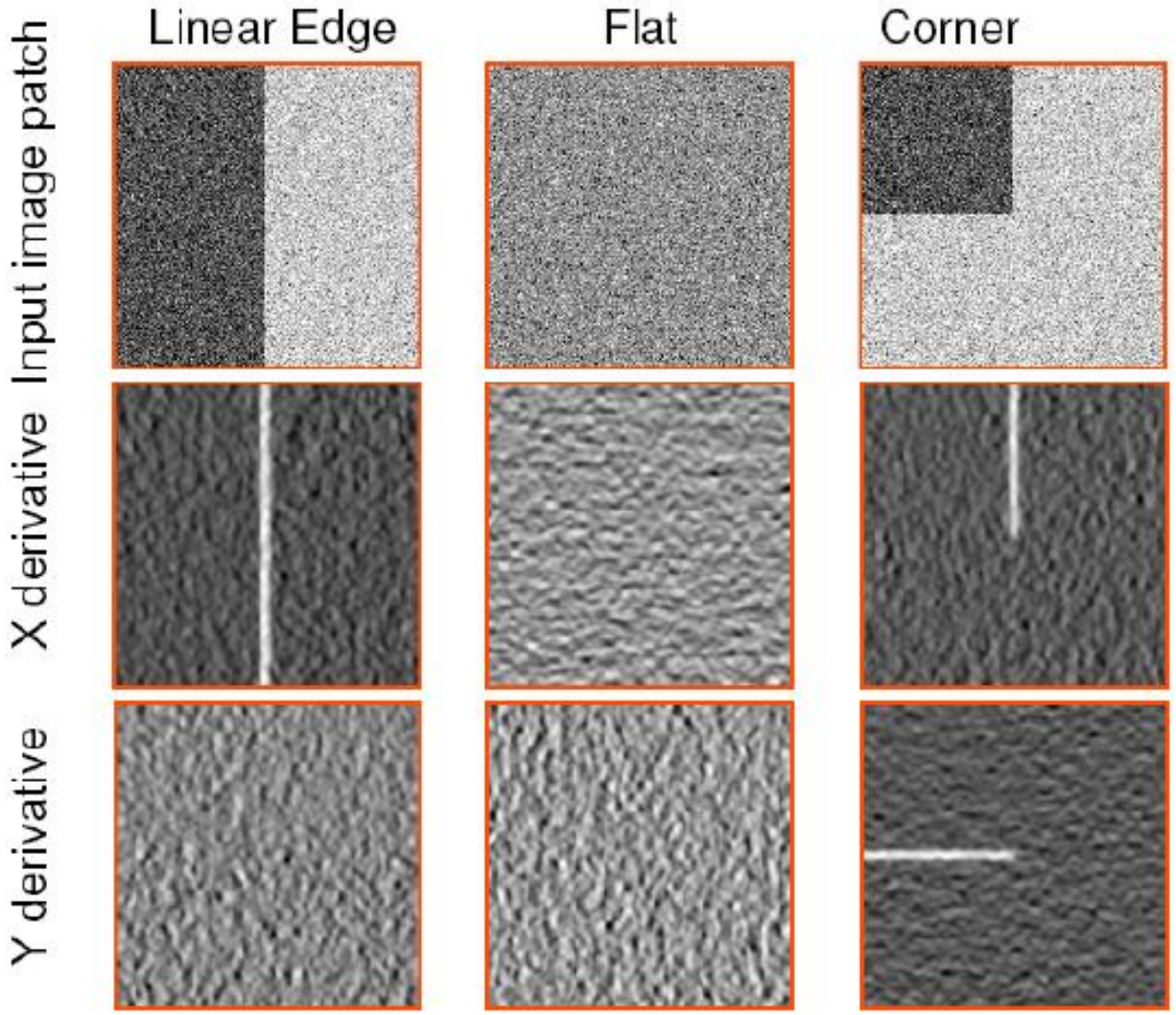
Note: these are just products of components of the gradient, Ix, Iy

Harris Corner Derivation

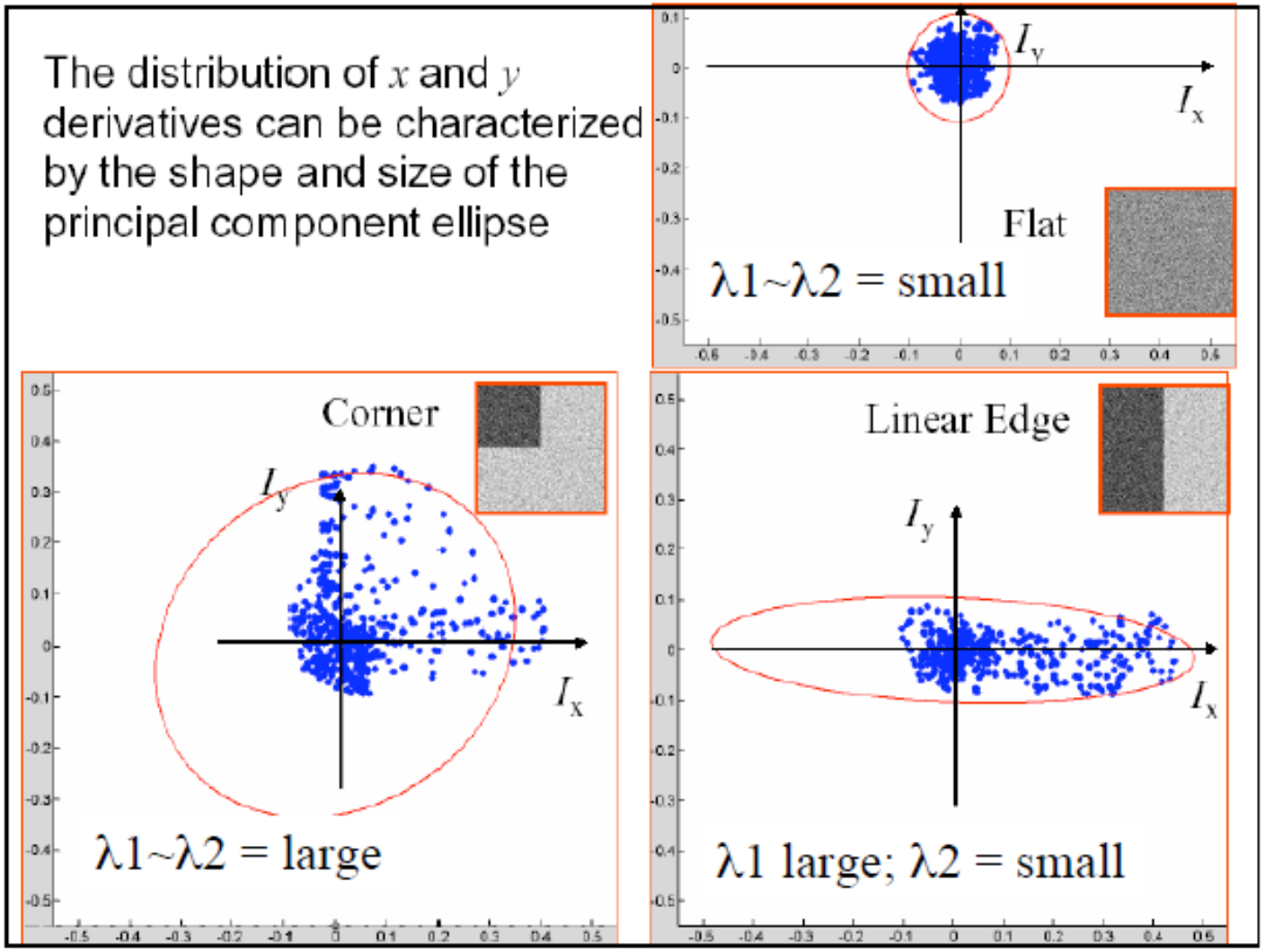
- *“flat” region*: both eigenvalues of this matrix will be small, because all terms will be small.
- *“edge” region*: we expect to see one large eigenvalue associated with gradients at the edge and one small eigenvalue because few gradients will run in other directions.
- *“corner” region*: both eigenvalues will be large

- The behaviour of this matrix is most easily understood by plotting the ellipses of the set of vectors as points.

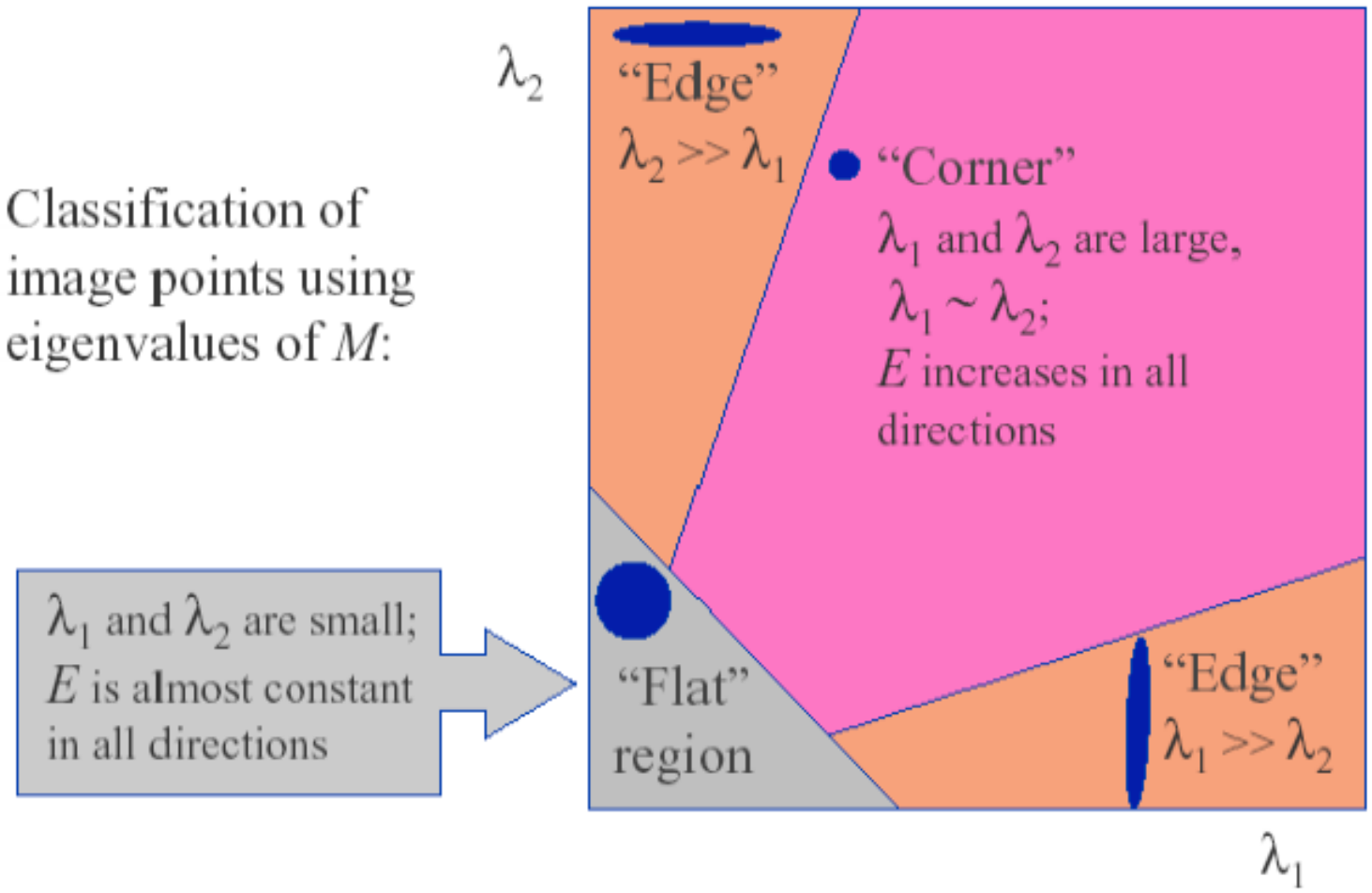
Example: Cases and 2D Derivatives



Example: Cases and 2D Derivatives



Classification via Eigenvalues



The eigenvectors encode edge directions, the eigenvalues edge strength.

Corner Response Measure

Measure of corner response

$$R = \det M - k(traceM)^2$$

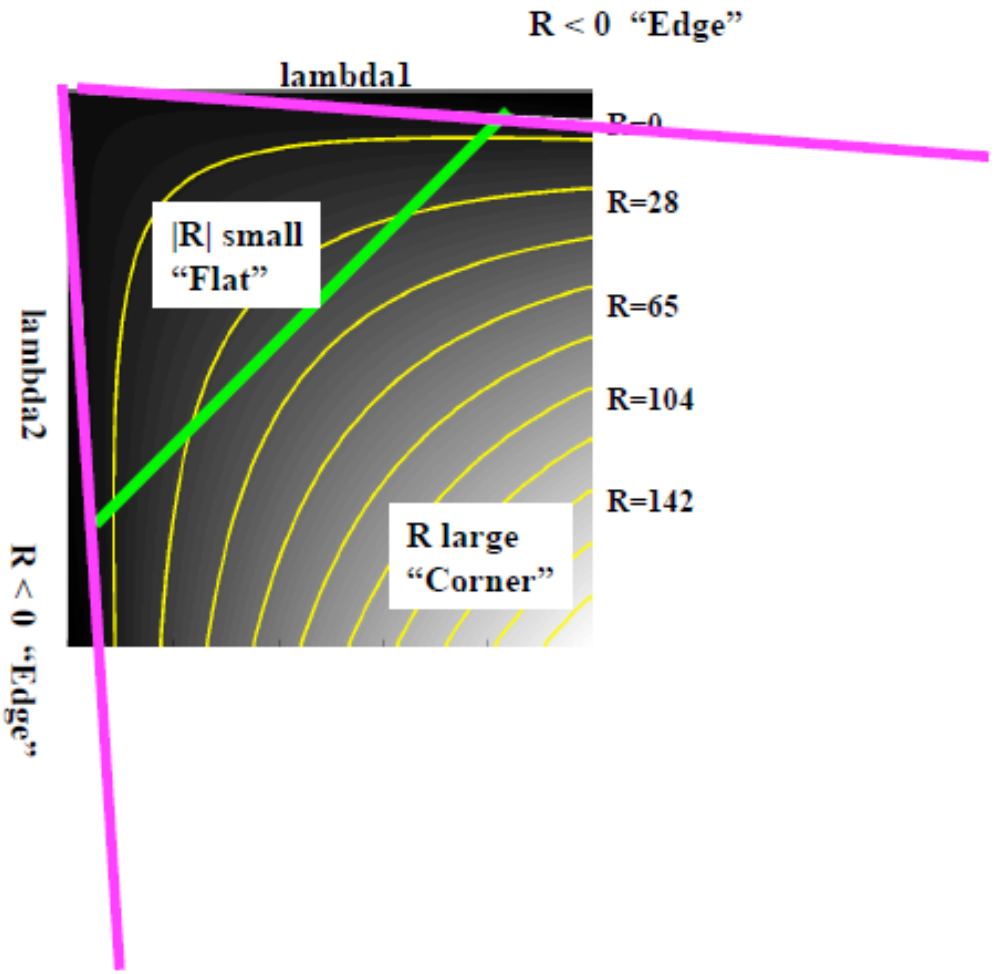
$$\det M = \lambda_1 \lambda_2$$

$$traceM = \lambda_1 + \lambda_2$$

K is empirically determined constant : k = 0.04 -0.06

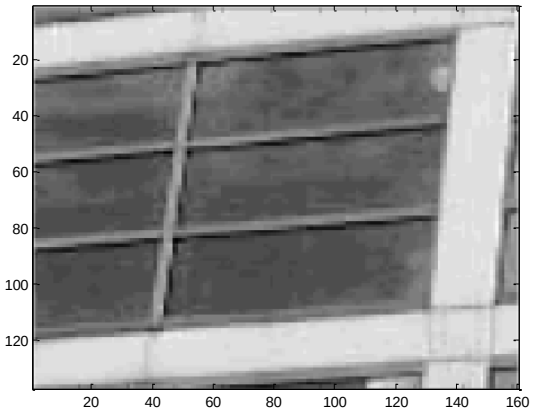
Corner Response Map

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region

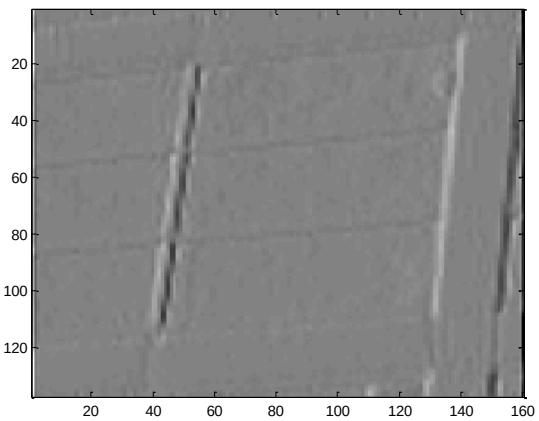


Corner Response Example

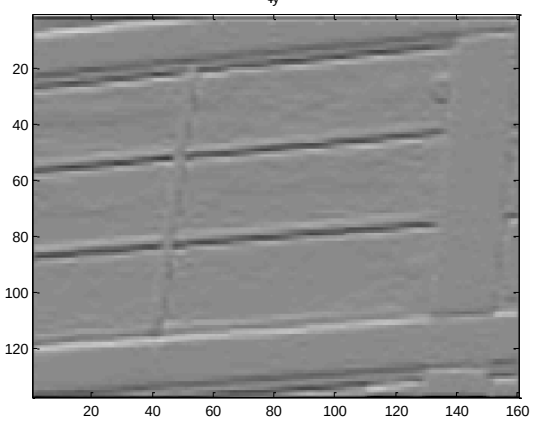
image



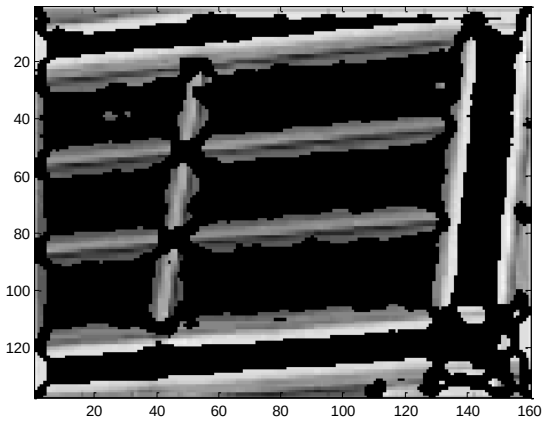
Ix



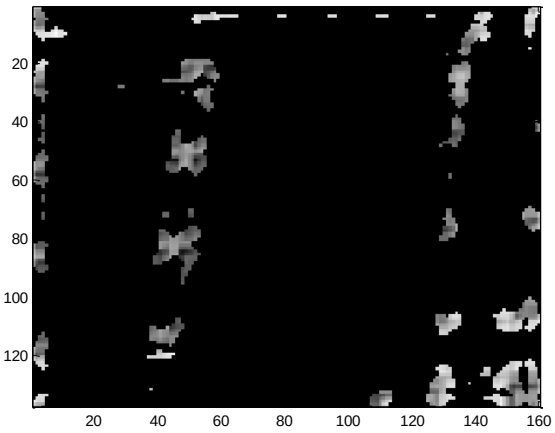
Iy



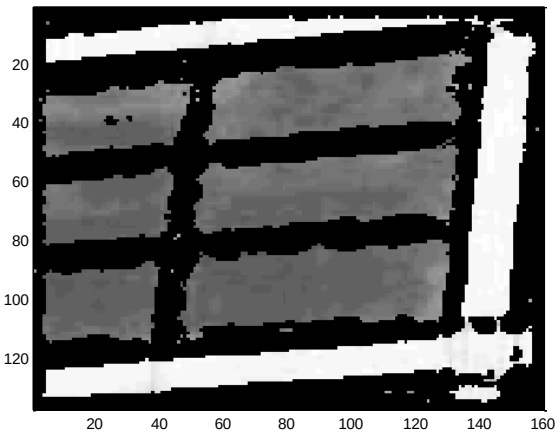
Edge regions



Corner regions



Fat regions



See harris_corner.m

HOMEWORK

- Reading Assignments:
  - Forsyth and Ponce book: sections 7.5, 8.1, 8.2, 8.3
  - Mubarak Shah, “Fundamentals of Computer Vision”: sections 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.10.

- Lab : Wednesday, October 12, 8.30-10.00 (B14)