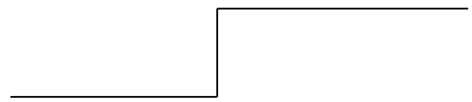
## Announcements

- Since Thursday we've been discussing chapters 7 and 8.
- "matlab can be used off campus by logging into your wam account and bringing up an xwindow and running "tap matlab" to find out the command to run matlab which will bring it up in the xwindow."

## Edge is Where Change Occurs

- Change is measured by derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or $2^{nd}$ derivative is zero.



## Noisy Step Edge

- Gradient is high everywhere.
- Must smooth before taking gradient.

## Implementing1D Edge Detection

1. Filter out noise: convolve with Gaussian
2. Take a derivative: convolve with [-1 0 1]

- *Matlab*
- We can combine 1 and 2.
- *Matlab*

## Implementing1D Edge Detection

3. Find the peak: Two issues:
   - Should be a local maximum.
   - Should be sufficiently high.

*Matlab*

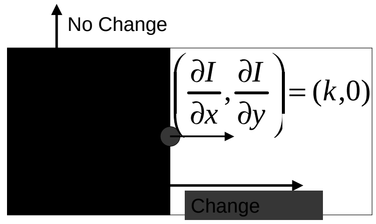
## 2D Edge Detection: Canny

1. Filter out noise
   - Use a 2D Gaussian Filter. $J = I \otimes G$
2. Take a derivative
   - Compute the magnitude of the gradient:

$$\nabla J = (J_x, J_y) = \left( \frac{\partial J}{\partial x}, \frac{\partial J}{\partial y} \right) \text{is the Gradient}$$

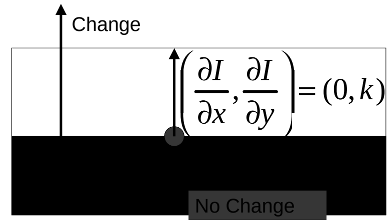
$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

## What is the gradient?




$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (k, 0)$$

## What is the gradient?

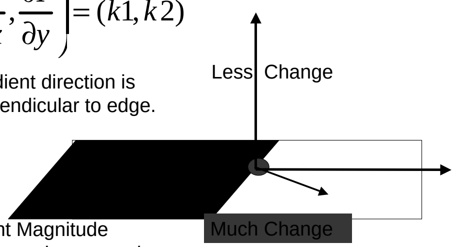



$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (0, k)$$

## What is the gradient?

$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (k1, k2)$$

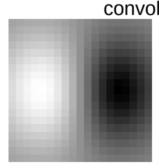
Gradient direction is perpendicular to edge.

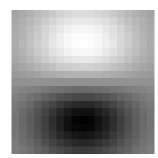


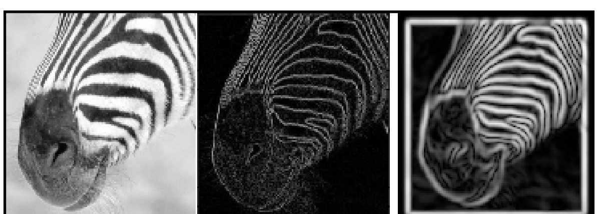

Gradient Magnitude measures edge strength.

## Smoothing and Differentiation

- Need two derivatives, in x and y direction.
- We can use a derivative of Gaussian filter
  - because differentiation is convolution, and convolution is associative



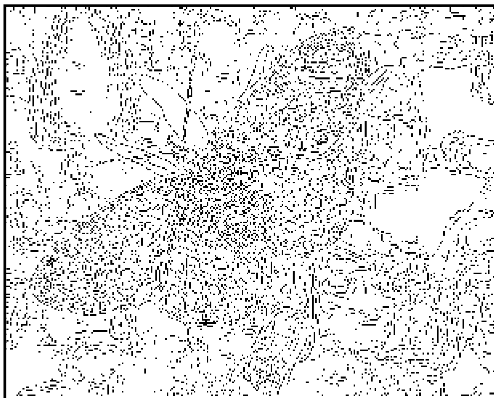




## Scale

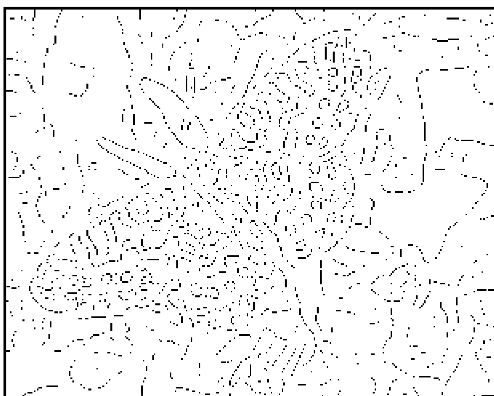
Smoothing
- Eliminates noise edges.
- Makes edges smoother.
- Removes fine detail.
- *Matlab*

(Forsyth & Ponce)



fine scale
high
threshold

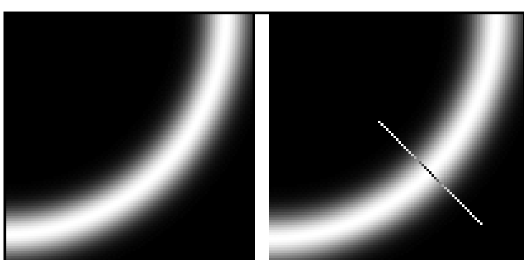
coarse
scale,
high
threshold



coarse
scale
low
threshold

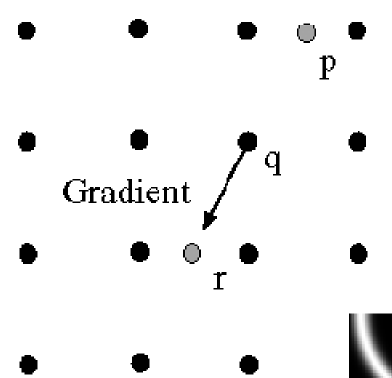
## Finding the Peak

1) The gradient magnitude is large along thick trail; how do we identify the significant points?
2) How do we link the relevant points up into curves?



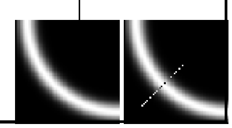
We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?

(Forsyth & Ponce)

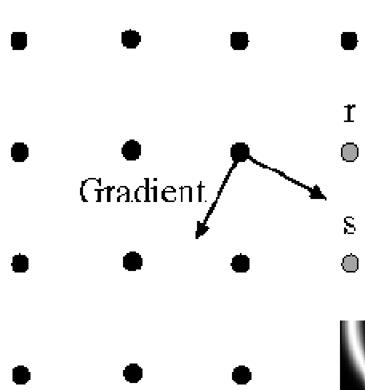



Non-maximum suppression

At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

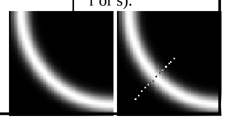


(Forsyth & Ponce)




Predicting the next edge point

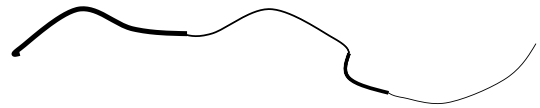
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



(Forsyth & Ponce)

## Hysteresis

- Check that maximum value of gradient value is sufficiently large
  - – drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



*Demo of Edge Detection*

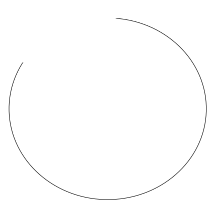
## Why is Canny so Dominant

- Still widely used after 20 years.

1. Theory is nice (but end result same).
2. Details good (magnitude of gradient).
3. Hysteresis an important heuristic.
4. Code was distributed.
5. Perhaps this is about all you can do with linear filtering.

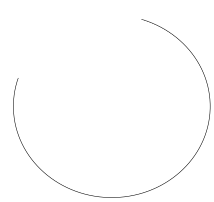
## Corners

- Why are they important?



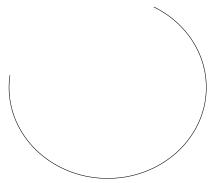
## Corners

- Why are they important?



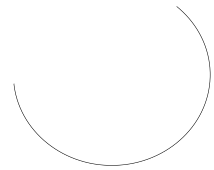
## Corners

- Why are they important?



## Corners

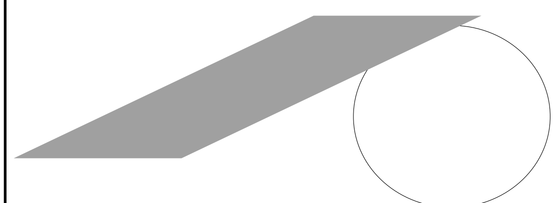
- Why are they important?



# STOP

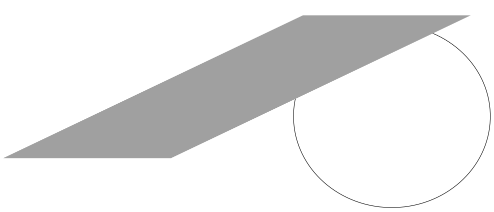
## Corners

- Why are they important?



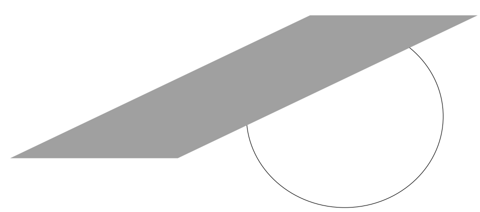
## Corners

- Why are they important?



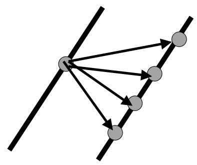
## Corners

- Why are they important?



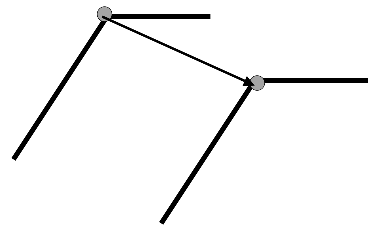
## Corners contain more edges than lines.

- A point on a line is hard to match.

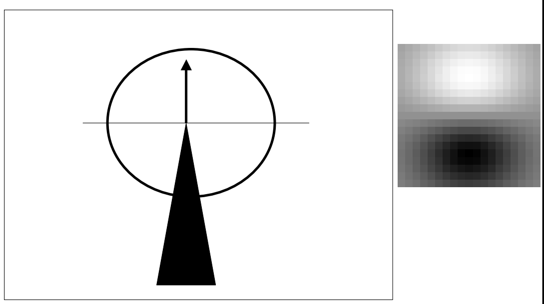


## Corners contain more edges than lines.

- A corner is easier



## Edge Detectors Tend to Fail at Corners



*Matlab*

## Finding Corners

Intuition:

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

## Formula for Finding Corners

We look at matrix:

Sum over a small region, the 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

***WHY THIS?***

First, consider 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 all gradients in neighborhood are:

(k,0) or (0, c) or (0, 0) (or off-diagonals cancel).

What is region like if:

1. $\lambda 1 = 0$?
2. $\lambda 2 = 0$?
3. $\lambda 1 = 0$ and $\lambda 2 = 0$?
4. $\lambda 1 > 0$ and $\lambda 2 > 0$?

## General Case:

From Linear Algebra we haven't talked about it follows that since C is symmetric:

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

So every case is like one on last slide.

## So, to detect corners

- Filter image.
- Compute magnitude of the gradient everywhere.
- We construct C in a window.
- Use Linear Algebra to find $\lambda 1$ and $\lambda 2$.
- If they are both big, we have a corner.