MANCHESTER 1824

The University of Manchester

Lecture 4: Edge Based Vision

Dr Carole Twining

Tuesday 10th March 2020

15:00pm – 16:00pm

MANCHESTER 1824

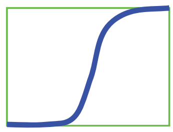
The University of Manchester

Overview:

- Why Edges Matter:
  - Edges in images correspond to physical events: edge of object, change in colour, change of surface orientation
- Edges and Derivatives
  - Convolution and filters (to detect changes)
- Edges and Scale
  - Physical edges persist across scales
- Edge Detection
  - Problem with noise, and accurate edge location
- Edge growing
  - Thresholding with hysteresis
  - Edge relaxation
- Hough Transform
  - Finding lines

Edges and Derivatives

First-Derivative Edge Filters



- What is an edge?
- To detect: look at the slope

Discrete version of ∂_x,
Central difference

Prewitt

-1	0	1
-1	0	1
-1	0	1

Sobel

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

Roberts

1	0
0	-1

?		?
-1	0	1
?		?

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

6 5

Decomposable:
Exterior product

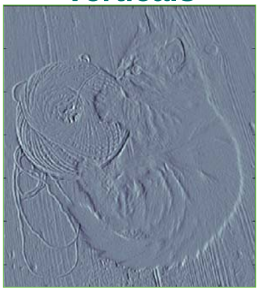
Multiplies and adds

$$(a \otimes b) * \mathcal{I} \equiv a * (b * \mathcal{I})$$

First Derivative Filters : Sobel

Image

$\mathcal{I}$

Verticals

$\mathcal{I}_\mathrm{x} \doteq \mathrm{S}_\mathrm{x} * \mathcal{I}$

Horizontals

$\mathcal{I}_\mathrm{y} \doteq \mathrm{S}_\mathrm{y} * \mathcal{I}$

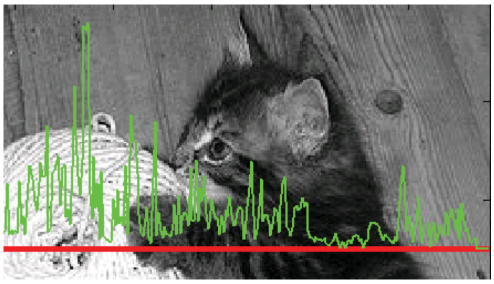
Edge Strength

$\sqrt{\mathcal{I}_x^2 + \mathcal{I}_y^2}$

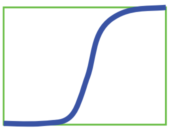
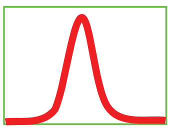
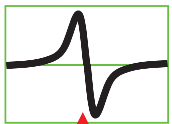
Edge strength: $g = |\vec{\nabla}\mathcal{I}| = \sqrt{\mathcal{I}_x^2 + \mathcal{I}_y^2}$

Ridges of g at edges, but noisy.

Normal to Edge: $\underline{\hat{n}} = \dfrac{\vec{\nabla}\mathcal{I}}{|\vec{\nabla}\mathcal{I}|}$



Second-Derivative Edge Filters

$\mathcal{I}$  $\partial_x \mathcal{I}$  $\partial_x^2 \mathcal{I}$ 

zero crossing

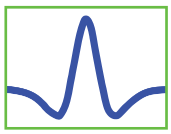
- Laplacian: scalar operator

$$\triangle = \nabla^2 = \partial_\mathrm{x}^2 + \partial_\mathrm{y}^2$$

- Difference of Gaussian, Laplacian of Gaussian: includes gaussian smoother
- False edges: every peak/trough of gradient gives a zero-crossing, not just big peaks
- Doesn't tell us the direction of the edge (scalar operator)
- Tends to create closed loops of edges ('plate of spaghetti' effect)

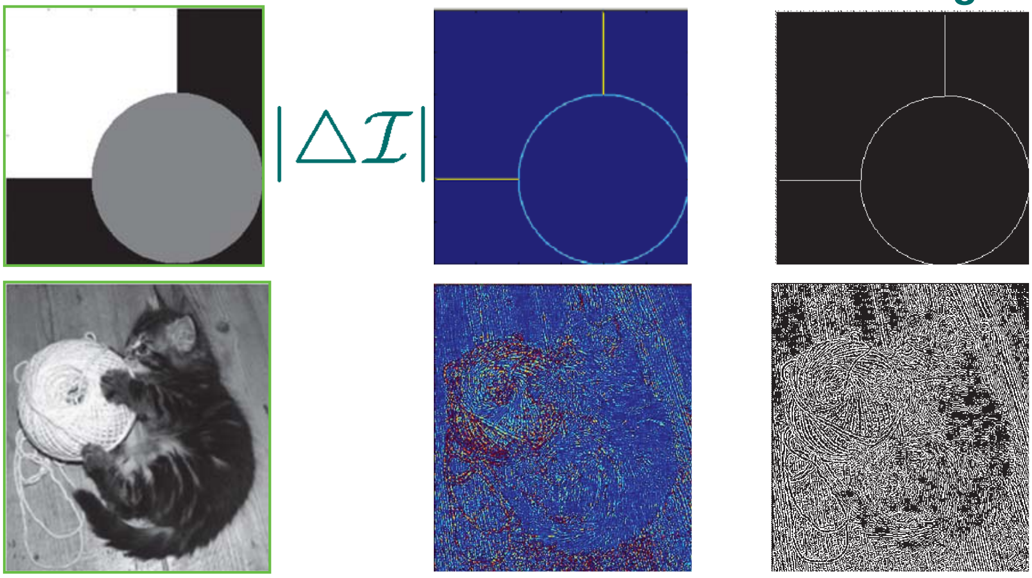
-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian



'mexican hat'

Laplacian Filter

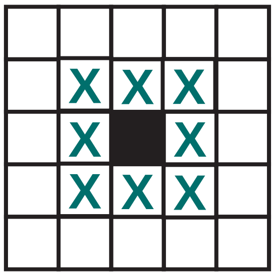


- Need to consider smoothing and noise
- Need to consider scale
- Need to consider edge detection

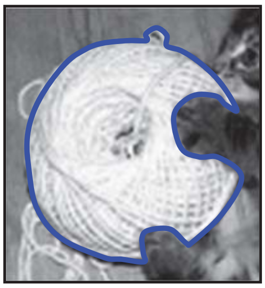
-1	-1	-1
-1	8	-1
-1	-1	-1

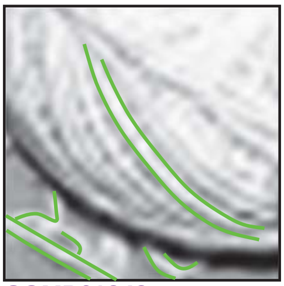
Edges and Scale

Edges and Scale



- Edge filters enhance noise
- What is a ‘real’ edge and what noise?
- Edges exist at many different scales
- What scales matter depends on application
- Sensible approach: use many different scales
  - Edges persist across scales, allows fusion across scales
- Gaussian gives scale & smoothing separable filter

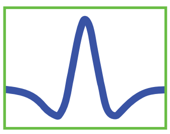




Edges and Scale

Marr-Hildreth:

- Convolve with gaussian $\mathcal{G}$
- Take Laplacian ∇^2 of result:
  - combine into single stage LoG
- Edges at zero-crossings
- Edges move with scale if curved
- No information on direction
- ‘Plate of spaghetti’ problem



‘mexican hat’

Canny:

- Convolve with gaussian $\mathcal{G}$
- Take gradient $\vec{\nabla}$ of result

$$\vec{\nabla}(\mathcal{G} * \mathcal{I}), \quad g = |\vec{\nabla}(\mathcal{G} * \mathcal{I})|$$

- Find gradient direction:

$$\hat{n} = \vec{\nabla}(\mathcal{G} * \mathcal{I})/g$$

- Create gaussian-smoothed derivative tuned to this direction
- Take another derivative in that direction to find local maximum, zero-crossing
- Stable across scales

Marr-Hildreth vs Canny

- Both involve pre-smoothing with gaussian
- Both involve second-derivative BUT:

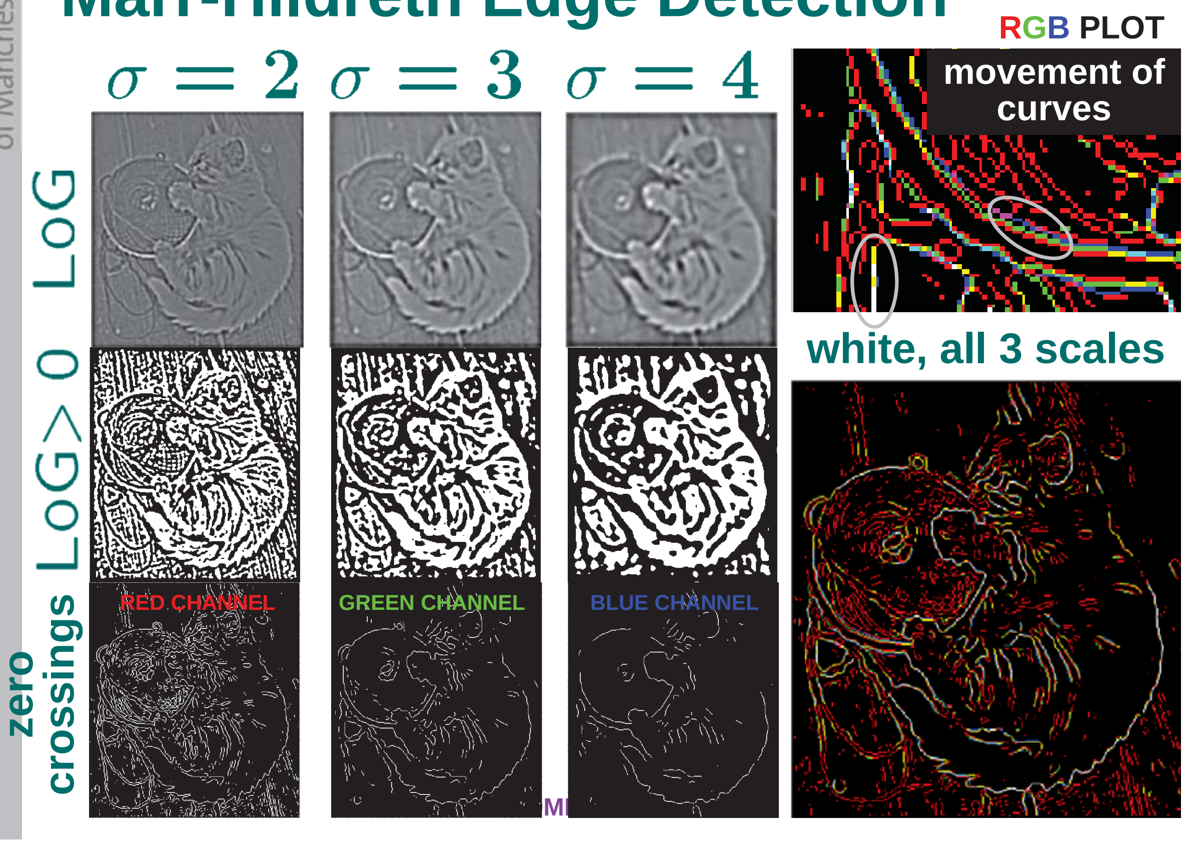
Marr-Hildreth:

- No information on direction
- By adding second-derivative in other direction, increases effect of noise

Canny:

- Create tuned derivative given estimated gradient direction
- Only compute second derivative in gradient direction
- Check that it really is local maximum of edge strength in that direction (see non-maximum suppression)

Marr-Hildreth Edge Detection

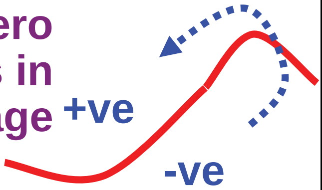


Marr-Hildreth Edge Detection

$\sigma = 10$

- Some edges not well localized
- ‘Plate of spaghetti’ effects

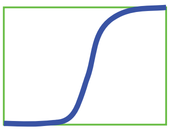
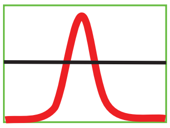
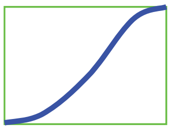
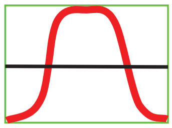
Trace zero crossings in image



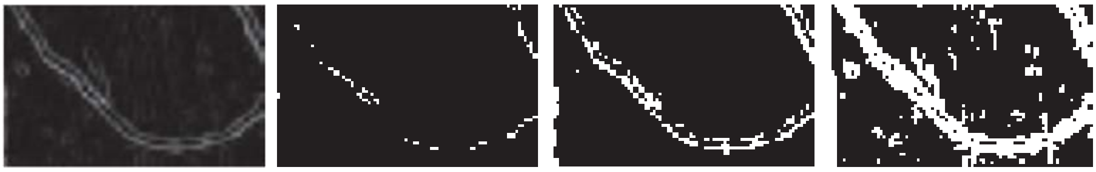
Keeps going until meets edge or closes the loop

Edge Detection

Edge Detection: First Derivatives

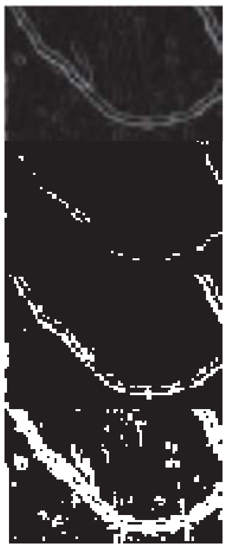
$\mathcal{I}$  $\partial_x \mathcal{I}$  $\mathcal{I}$  $\partial_x \mathcal{I}$ 

- **Position of maximum can be difficult to locate:**
  - **second-derivative, zero crossing more precise**
- **Simple threshold:**
  - **thick edges, need to apply thinning**
  - **missed edges, streaking (see thresholding with hysteresis)**



Edge Detection:

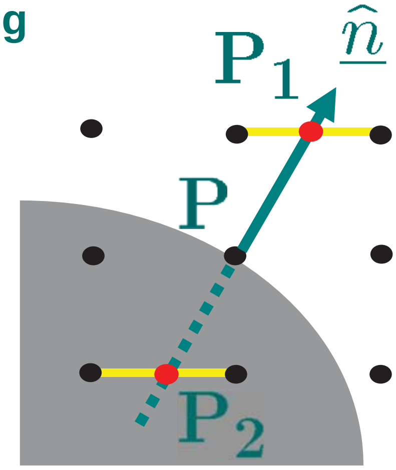
- **Zero-crossing more precisely located than maximum**
  - **Sub-pixel accuracy?**
- **Thresholding in Marr-Hildreth (LoG):**
  - **Threshold at ~zero, but what about noise?**
  - **Doesn't use directional information**
  - **Other second derivative increases noise**
- **'Plate of spaghetti':**
  - **continuity => closed loops or meets boundary**
- **Thresholding & Thinning 1st Derivative**
  - **Incorporates neighbourhood information**
  - **Still doesn't use all available information**
- **If we had the edge direction as well…….**



Non-Maximum Suppression

- Start from edge-strength signal g
- Locate possible edge point P
- Identify gradient direction $\widehat{n}$
- Interpolate g at P_1 and P_2
- P is local maximum provided:

  g(P) > g(P_1) & g(P) > g(P_2)
- Only accepts as edge if proper maximum, rejects if not
- In practise, only allow a set of discrete possible directions



Object & pixel positions

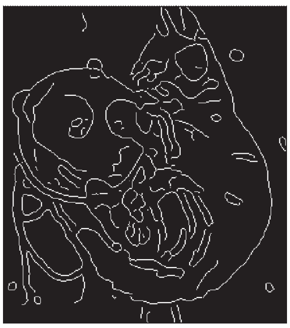
Canny Edge Detector

$\sigma = 1$

$\sigma = 1.5$

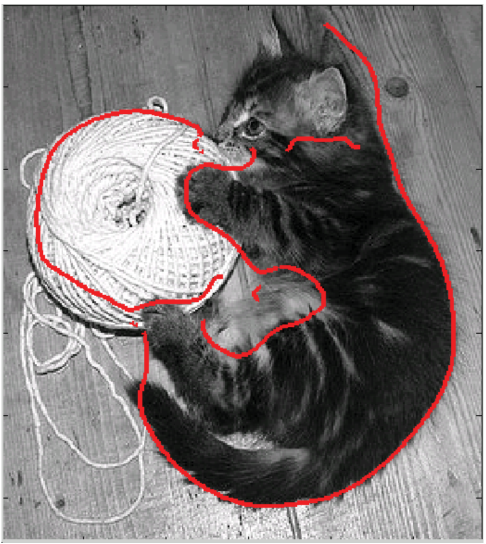
white, all 3 scales

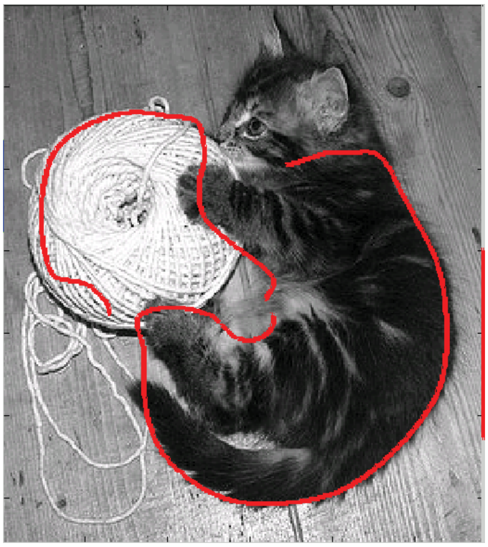
$\sigma = 2$



Canny Edge Detector:

$\sigma = 10$ $\sigma = 20$





Marr-Hildreth vs Canny at σ=10

From Edge Pixels to Edges

- Have candidate edge pixels
- Have information on edge direction and strength
- Want connected edges:

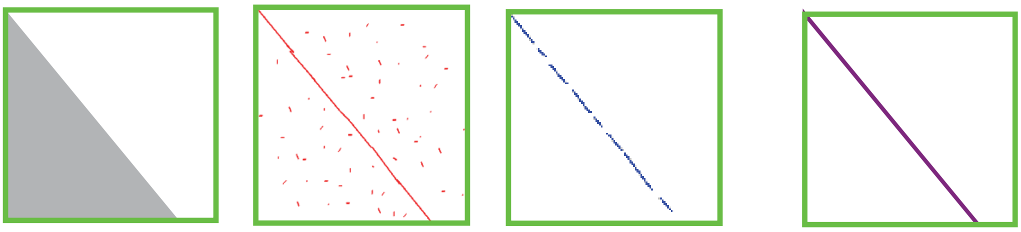
Edge growing

- Going from individual edge pixels, to entire, connected edges – curves that are boundaries of objects

Edge Growing

Edge Thresholding with Hysteresis

- Edge strength image, two thresholds T_H & T_L
- Only edges have points $g > T_H$
- Edges have all points $g > T_L$
- Start at point $g > T_H$, and trace connected points with $g > T_L$



$\mathcal{I}$ $\quad g > T_L \quad g > T_H \quad Result$

Edge Relaxation

- Use context to resolve ambiguity (as in segmentation)

$g(i)$: Edge strength at pixel i

$\underline{e}(i)$: Edge direction at pixel i

Normalise edge strengths $g(i) \Rightarrow P(\underline{e}, i) \leq 1$

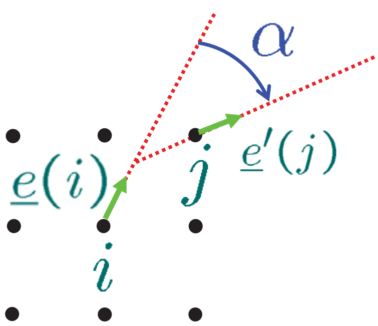
- **Compatibility**

Pixels i and j,

edge directions $\underline{e}$ and $\underline{e}'$:

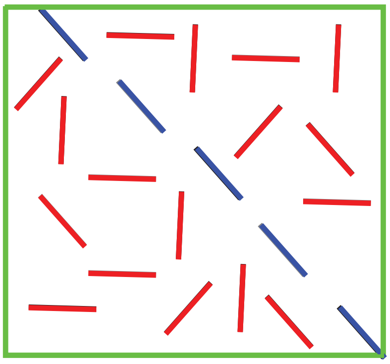
$c_{i,j}(\underline{e}, \underline{e}') = 0$ not neighbours

$c_{i,j}(\underline{e}, \underline{e}') = |\cos(\alpha)|$

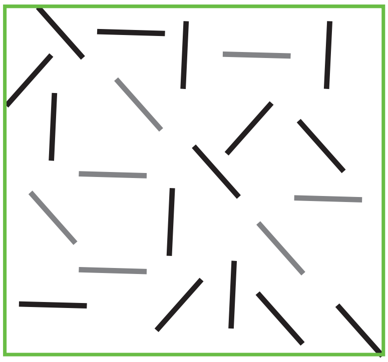


- As before, update probabilities based on support

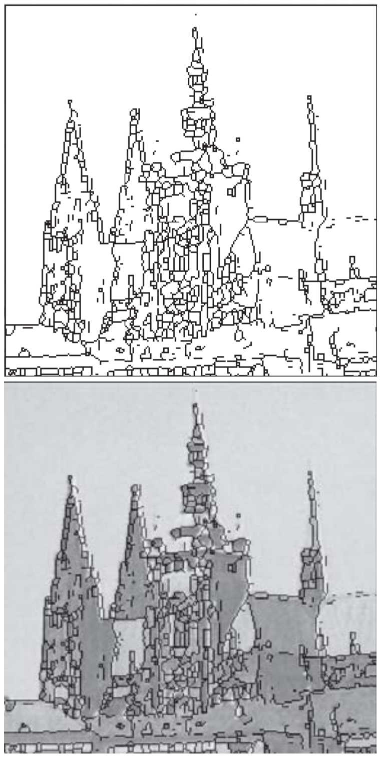
Edge Relaxation



weak and strong edges

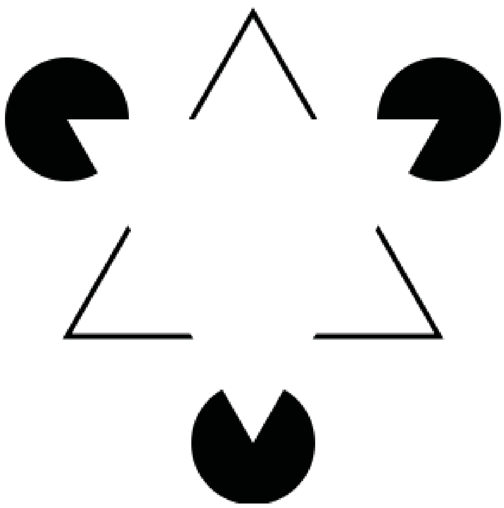


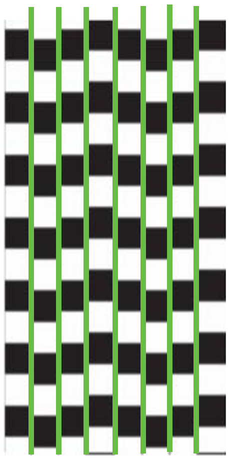
- Supporting each other
- Many refinements and alternatives in the literature, but all applying same basic ideas



Hough Transform

Aside: Lines in human vision





See lines where we have only minimal information

Actually straight, but we don't see them as that!

Hough Transform (1)

- Have some set of points, parts of edges etc
- Want to put them together into continuous lines
- Strategy:
  - Transform to parameter space
  - Let points vote for lines that could pass through them
  - Look for clusters
- Finding the right parameter space
- Can be extended if you can find such a space for shape of interest

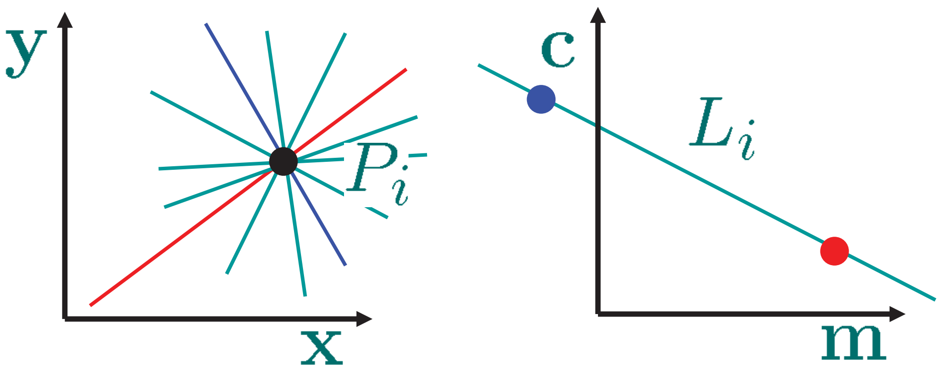
Hough Transform (2)

Set of points $\{P_i = (x_i, y_i)\}$ in image plane.

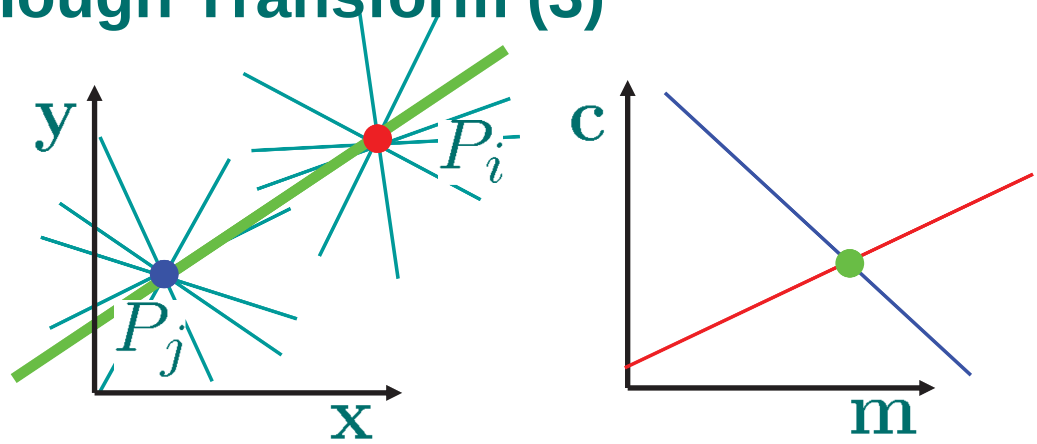
Any and all straight lines thro' P_i:

$\boxed{y_i = mx_i + c} \Rightarrow \boxed{c = -x_i m + y_i}$

L_i: line in (c, m) plane, intercept y_i, gradient $-x_i$

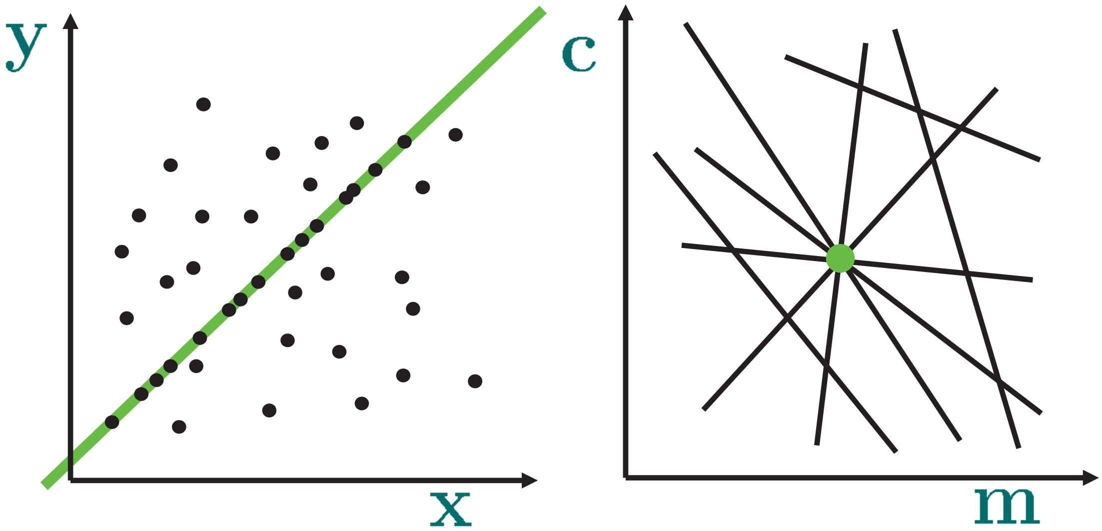


Hough Transform (3)



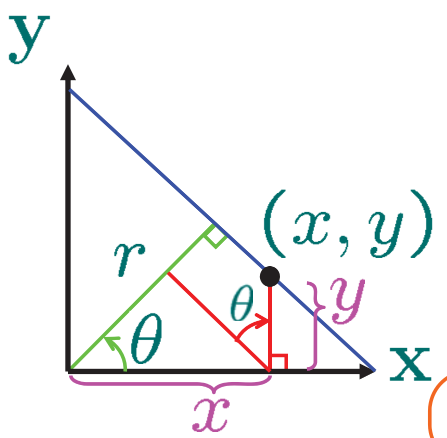
- Repeat for all points $\{P_i = (x_i, y_i)\}$ in image plane
- Look for points in (c,m) plane where lots of lines cross
- Lines which pass thro' lots of points in image plane

Hough Transform (4)



- Verticals, m is infinite! Need better parameter space

Hough Transform (5)



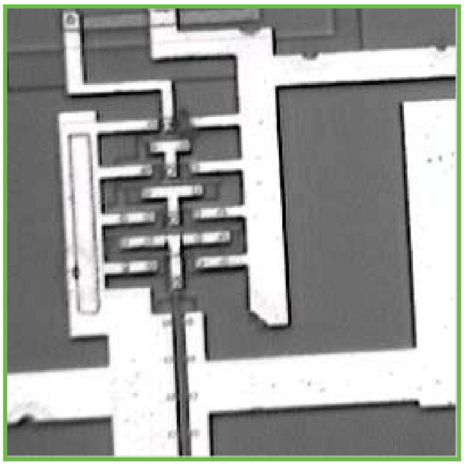
$$y = mx + c$$

$$(c, m) \Rightarrow (r, \theta)$$

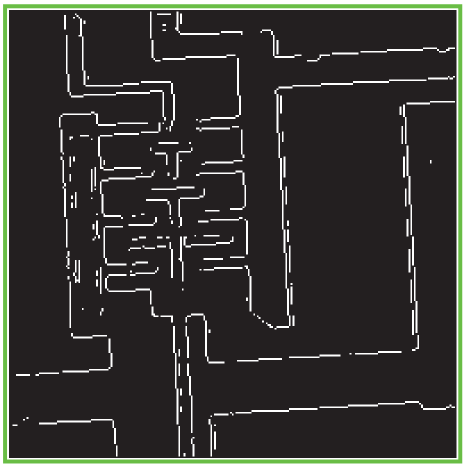
$$\boxed{r = x\cos\theta + y\sin\theta}$$

- Single point $P_i = (x_i, y_i)$
- All possible θ : allowed values of r, sinusoid curve
- Extend to other than lines, generalised Hough transform

Example: Integrated Circuit

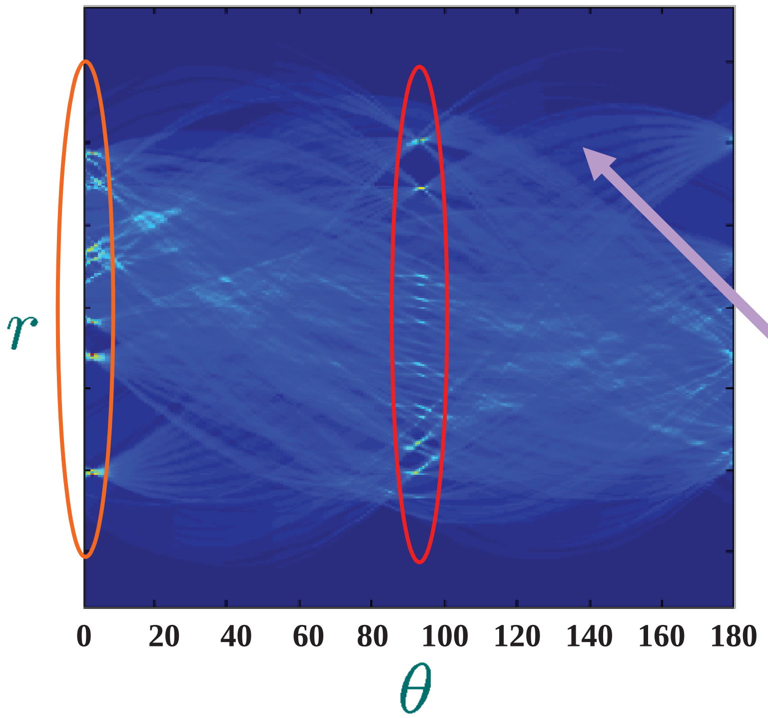


Image



Edge Pixels

Example: Integrated Circuit



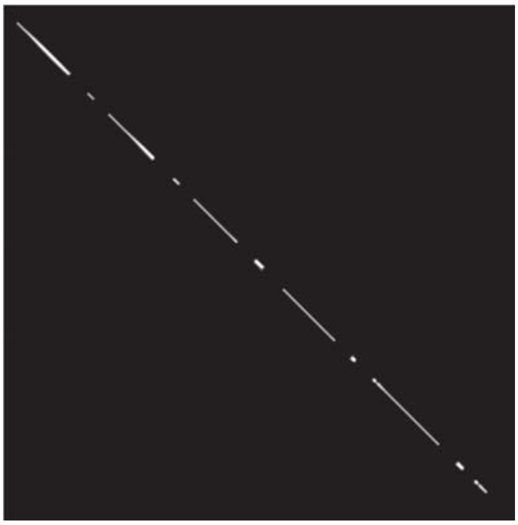
Each edge pixel
= 1 sinusoid

Each peak
= 1 line in image

Set of peaks at
approx 90^0

Another at
approx 0^0

Example: Finding Lines under Noise

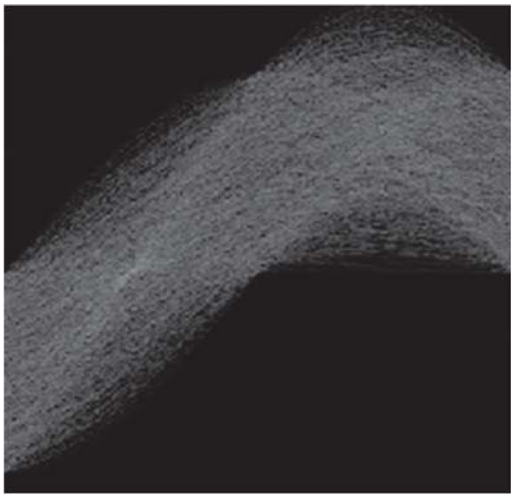


Broken Line

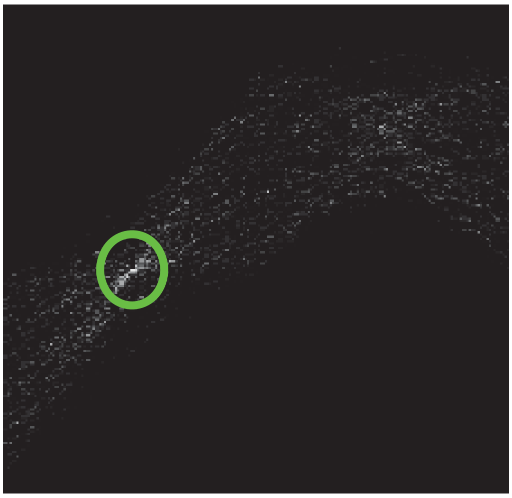
Hidden under noise

Edge Strength Image

Example: Finding Lines under Noise



Hough Space



…thresholded

Summary:

- **Edges and Derivatives**
  - Convolution and filters (first & second derivatives, gaussians)
- **Edges and Scale**
  - Physical edges persist across scales
- **Edge Detection**
  - Problems with noise, and accurate edge location
  - Non-maximum suppression
- **Edge Growing**
  - Thresholding with hysteresis
  - Edge relaxation
- **Hough Transform**
  - Finding lines/circles etc even when occluded