Lecture 8 Finding Shapes

COMP6223 Computer Vision (MSc)

How can we group points to find shapes?

Book
pp
187-201;
208-215

Department of Electronics and Computer Science

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Content

1. How do we define and detect shapes in images?
2. How can we improve the detection process?

Feature extraction by thresholding

(a) image (b) low threshold (c) high threshold

Conclusion: we need shape!

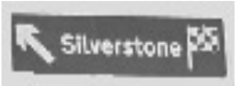
Template Matching - basis

$$\| \mathbf{I}_{\Omega} - \mathbf{T} \|^{2}$$

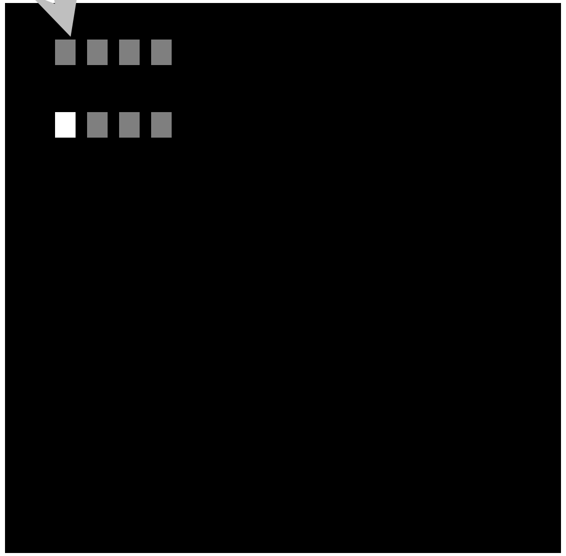
Process of template matching

$\boldsymbol{\Omega}$

Image $\mathbf{I}$



Template $\mathbf{T}$



accumulator space

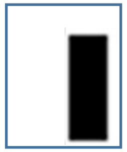
count of matching points

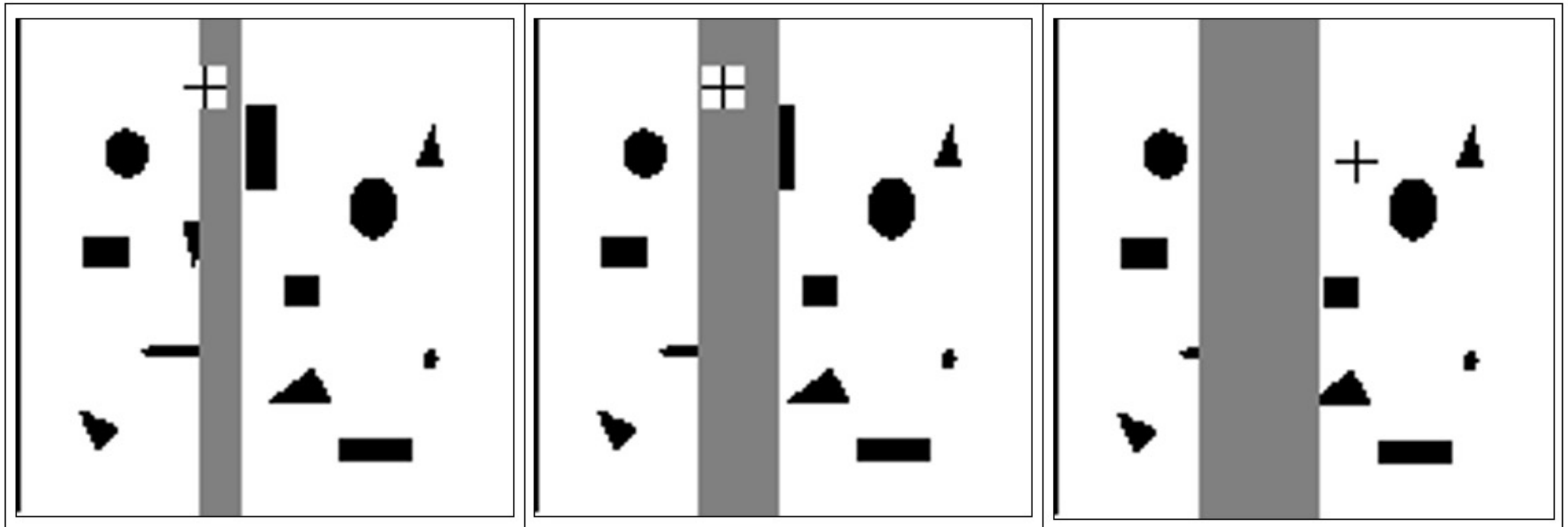
Suggestions for improving the process?

Use edges!

Template matching in occluded images

Template





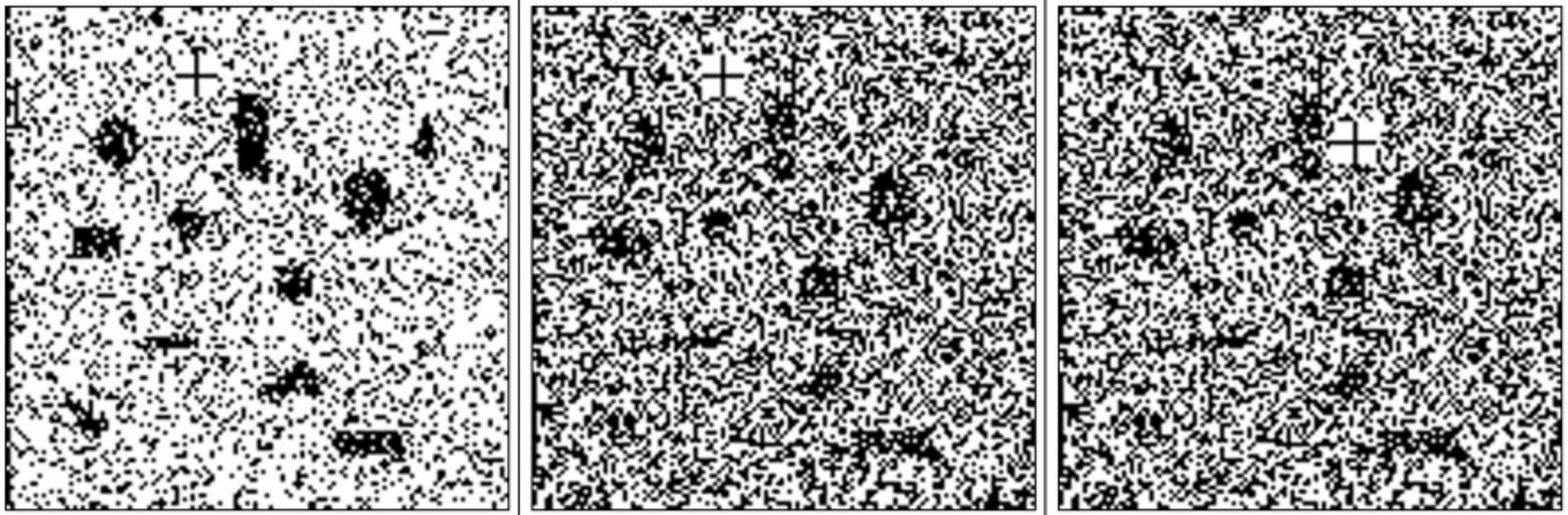
(a) extraction (of the black rectangle) in no occlusion

(b) extraction in some occlusion

(c) extraction in complete occlusion (failed)

Template matching is optimal in occlusion

Template matching in noisy images



(a) extraction (of the black rectangle) in some noise

(b) extraction in a lot of noise

(c) extraction in too much noise (failed)

Template matching is optimal in noise

… but **slow** ….

Template Matching

$$\| \mathbf{I}_\Omega - \mathbf{T} \|^2 = \Sigma\,(\mathbf{I}_\Omega \mathbf{I}_\Omega - 2\mathbf{I}_\Omega\mathbf{T} + \mathbf{T}\,\mathbf{T})$$

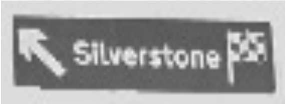
Approximately constant

Fixed

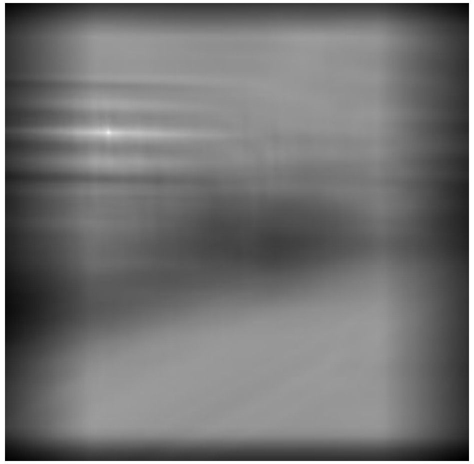
- Correlation and convolution
- Implementation via **Fourier**

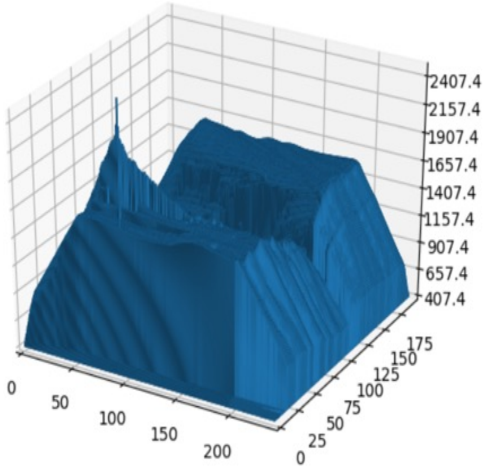
Intuitively simple

image



template





accumulator space

Convolution and Correlation

Convolution:

Application of a template and involves flipping the template:

$$(\mathbf{I} * \mathbf{T})(i,j) = \sum_{(x,y)\in W} \mathbf{I}_{x,y}\mathbf{T}_{i-x,j-y}$$

Utilising Fourier transform F :

$$\mathbf{I} * \mathbf{T} = F^{-1}(F(\mathbf{I}).\times F(\mathbf{T}))$$

Correlation:

Matching of a template:

$$(\mathbf{I} \otimes \mathbf{T})(i,j) = \sum_{(x,y)\in W} \mathbf{I}_{x,y}\mathbf{T}_{x+i,y+j}$$

Utilizing Fourier transform after flipping the template:

$$\mathbf{I} \otimes \mathbf{T} = F^{-1}(F(\mathbf{I}).\times F(\mathbf{T}_{\text{flip}}))$$

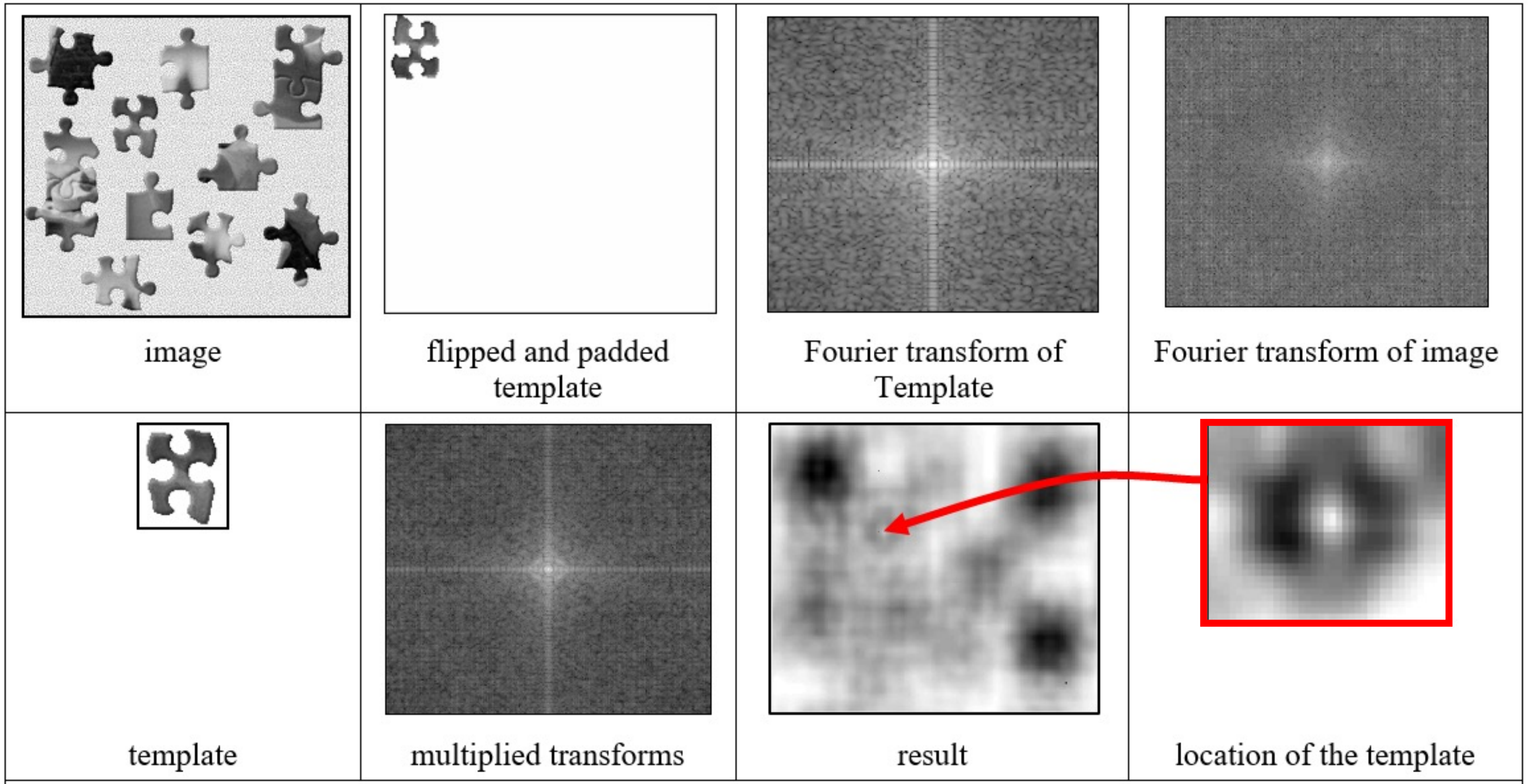
Flipped template

Encore, Baron Fourier!

$$(\mathbf{I} \otimes \mathbf{T})(i,j) = \sum_{(x,y)\in W} \mathbf{I}_{x,y}\mathbf{T}_{x+i,y+j}$$

$$F^{-1}(F(\mathbf{I}).\times F(\mathbf{T}_{\text{flip}}))$$

Template matching is slow, so use FFT



Template Matching via Fourier Transform

No sliding of templates here

Cost is 2×FFT plus multiplication

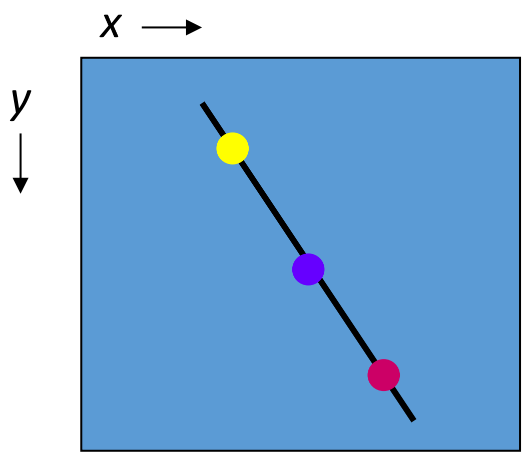
Applying template matching

Hough Transform

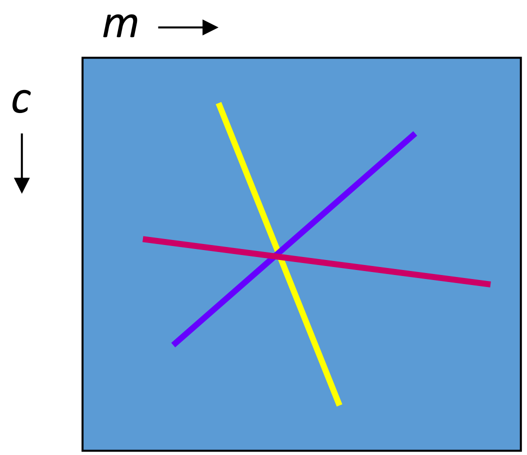
- Performance same as template matching, but faster
- A line is points *x, y* gradient *m* intercept *c* $y = m \times x + c$
- and is points *m, c* gradient *-x* intercept *y* $c = -x \times m + y$

Hough Transform

- Performance same as template matching, but faster
- A line is points *x*, *y* gradient *m* intercept *c* $y = m \times x + c$
- and is points *m*, *c* gradient -*x* intercept *y* $c = -x \times m + y$



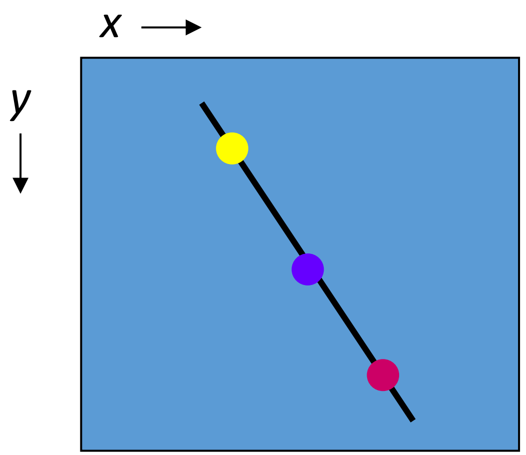
image



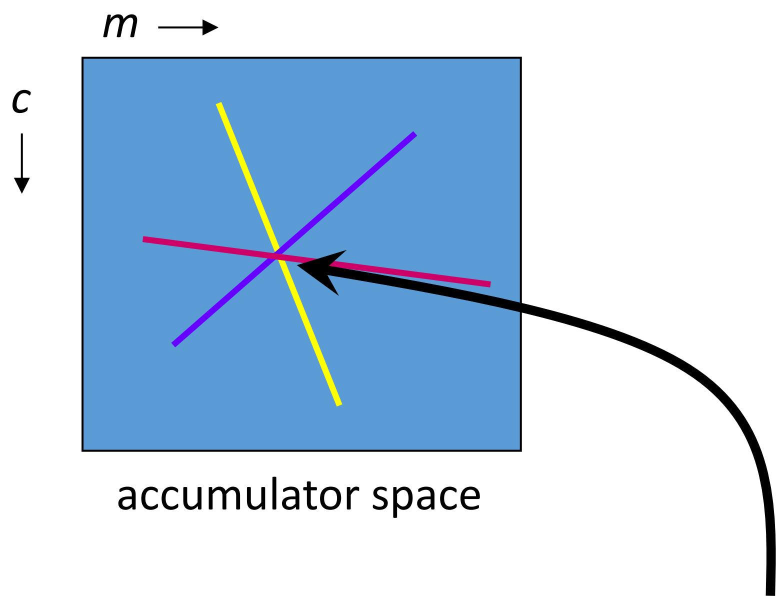
accumulator space

Hough Transform

- Performance same as template matching, but faster
- A line is points *x*, *y* gradient *m* intercept *c* $y = m \times x + c$
- and is points *m*, *c* gradient -*x* intercept *y* $c = -x \times m + y$



image



accumulator space

The coordinates of the peak are the parameters of the line

In maths it's the principle of duality

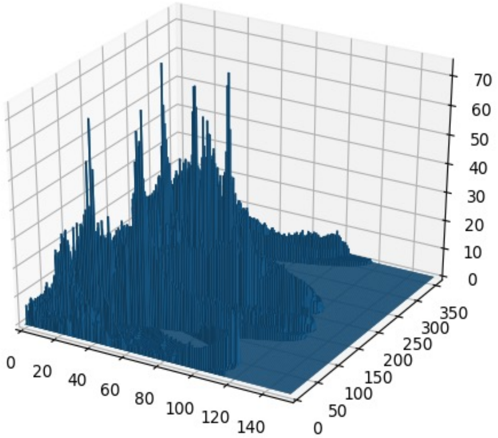
Pseudocode for HT

```
accum=0
for all x,y                              !look at all points
    if edge(y,x)>threshold               !check significance
        for m = -10 to +10               !if so, go thru m
            c = -x*m+y                   !calculate c
            accum(m,c) PLUS 1            !vote in accumulator
m,c = argmax(accum)                      !peak gives parameters
```

Applying the Hough transform for lines

image

detected lines



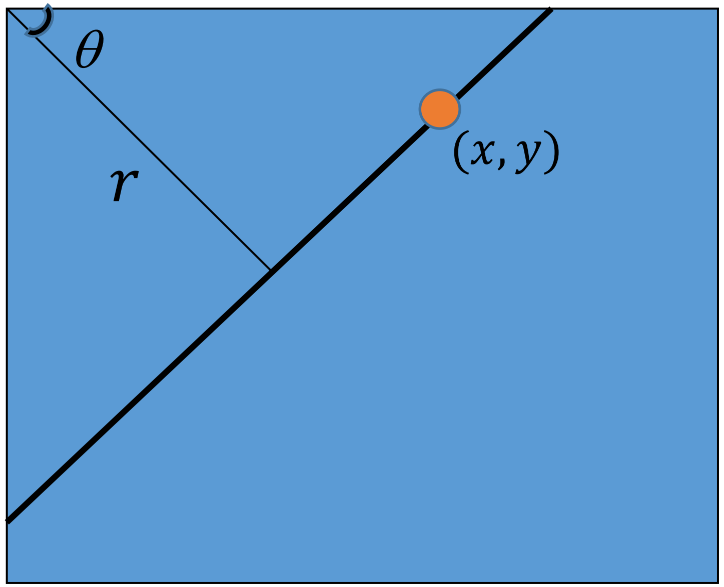
accumulator space

OK, it works. Can anyone see a problem?

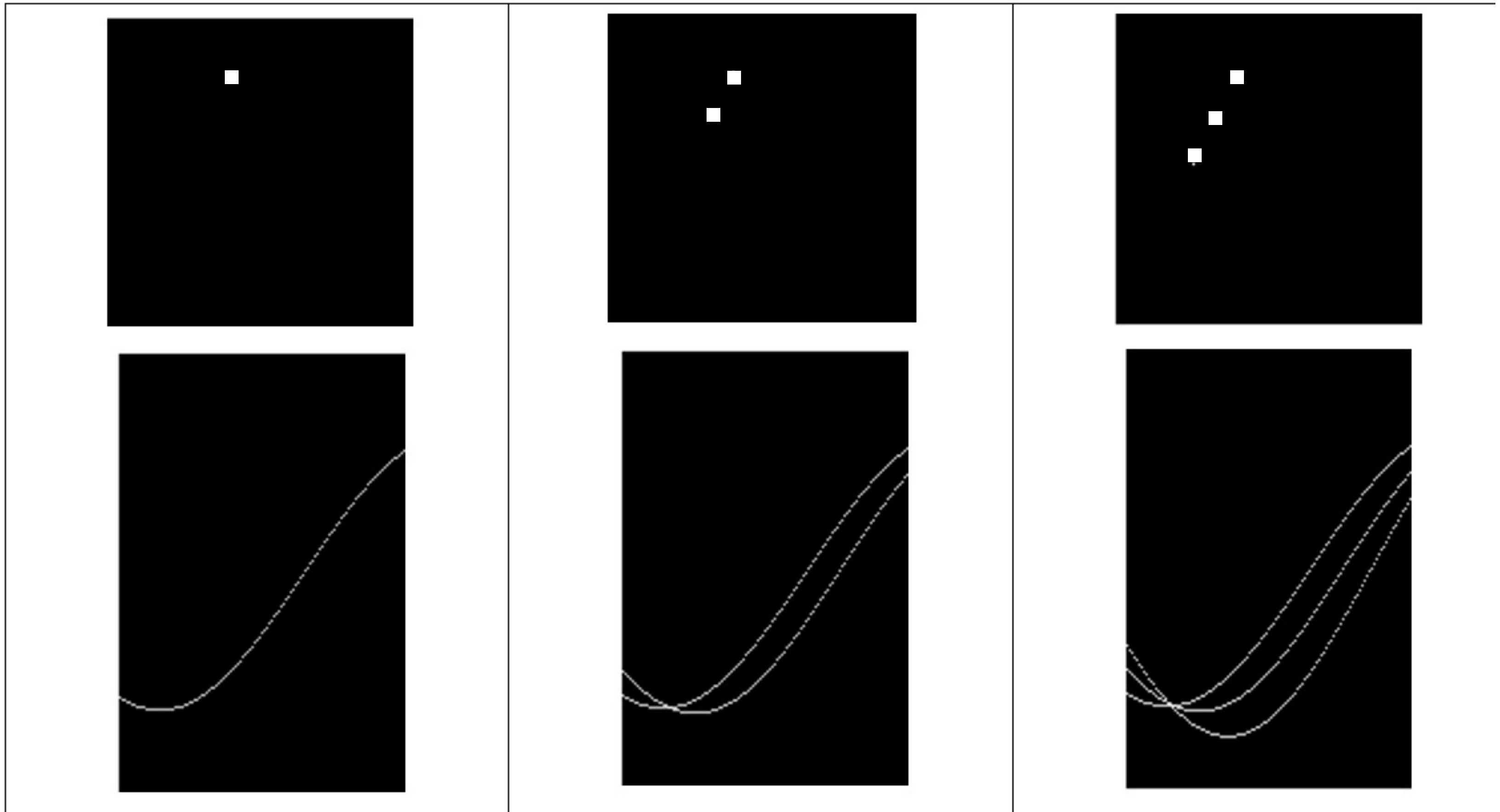
Hough Transform for Lines … problems

- *m, c* tend to **infinity - problem**

- Change the parameterisation
- Use foot of normal: $r = x\cos\theta + y\sin\theta$
- Gives polar Hough transform for lines

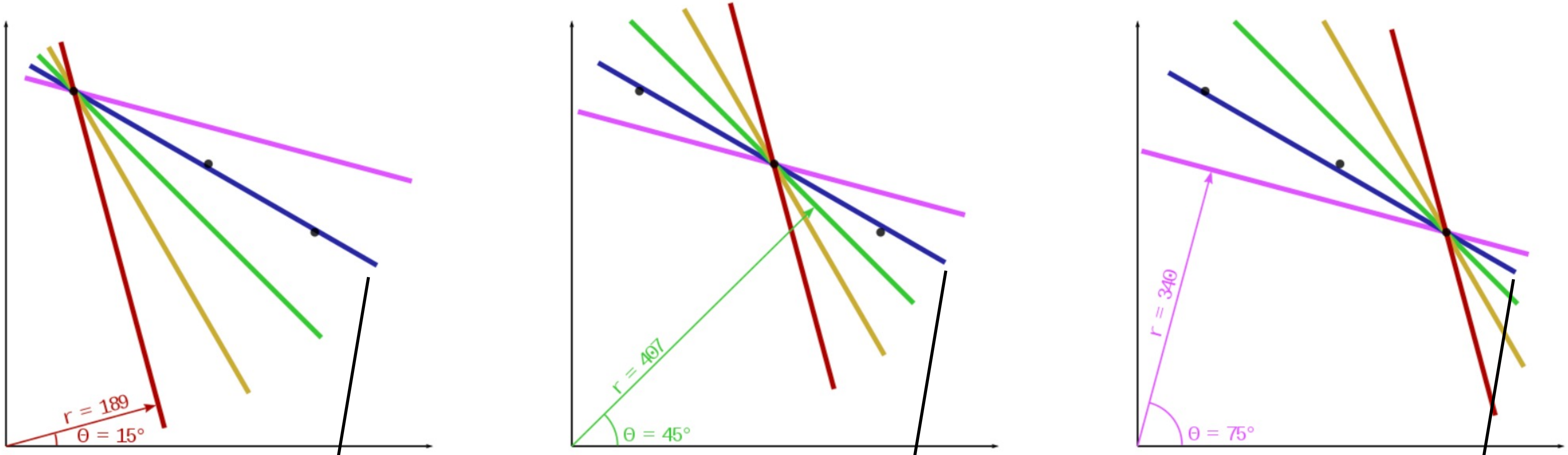


Images and the accumulator space of the polar Hough transform



(a) for one point (b) for two points (c) for three points

Polar Hough transform for lines



θ	r	θ	r	θ	r
15	189.0	15	318.5	15	419.0
30	282.0	30	376.8	30	443.6
45	355.7	45	407.3	45	438.4
60	407.3	60	409.8	60	402.9
75	429.4	75	385.3	75	340.1

[Credit Wikipedia]

Applying the Hough transform

Main points so far

1 – target shape defined by template

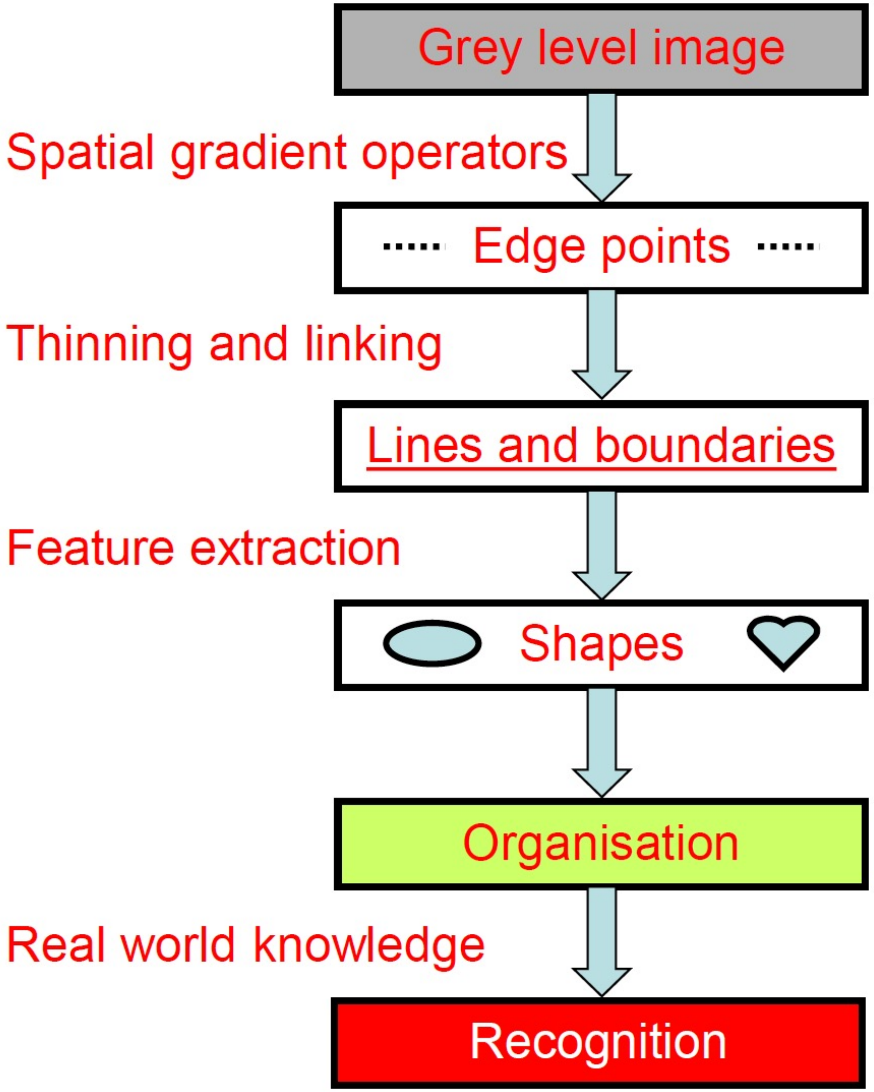
2 – and detected by template convolution

3 – optimal in occlusion and noise

4 – Hough transform gives same result, but faster

But shapes can be more complex than lines and
not defined by an equation. That's next…

A Framework for Computer Vision



Tony!
(CBE)

UNIVERSITY OF Southampton
School of Electronics and Computer Science