Lecture 8 Finding Shapes

COMP3204 Computer Vision

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

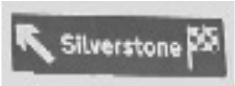
(a) image (b) low threshold (c) high threshold

Conclusion: we need shape!

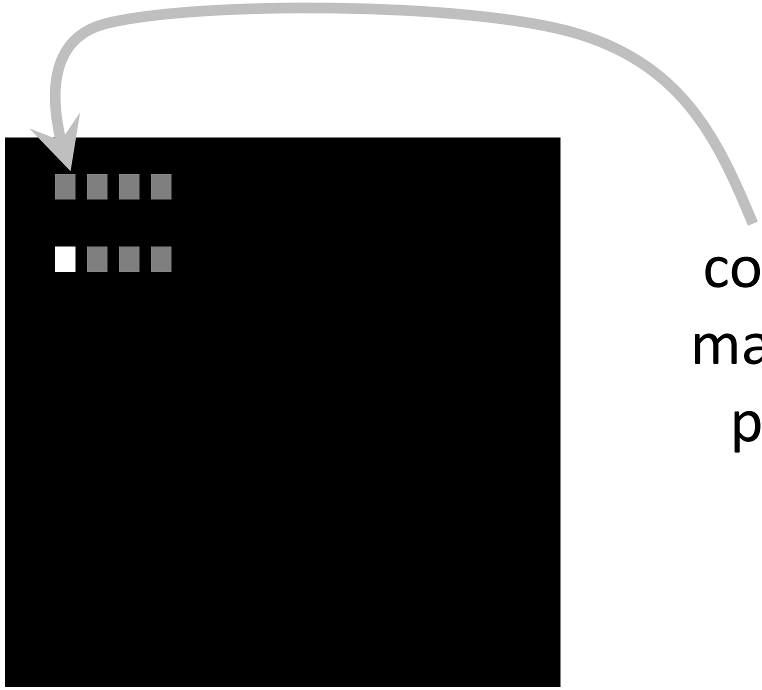
Template Matching -basis

Process of template matching

image



template



count of matching points

accumulator space

Suggestions for improving the process? Use edges!

Template Matching

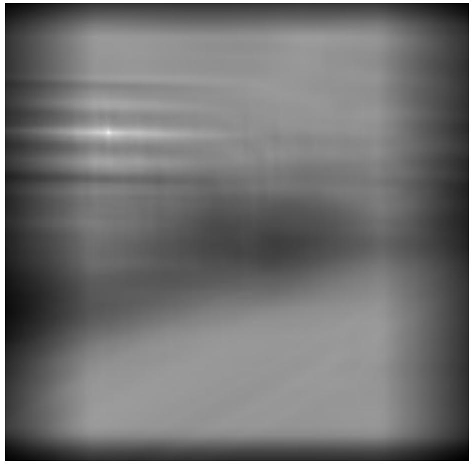
Intuitively simple

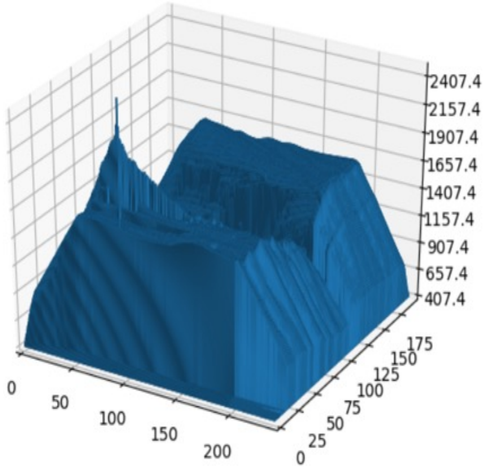
Correlation and convolution

Implementation via Fourier

image

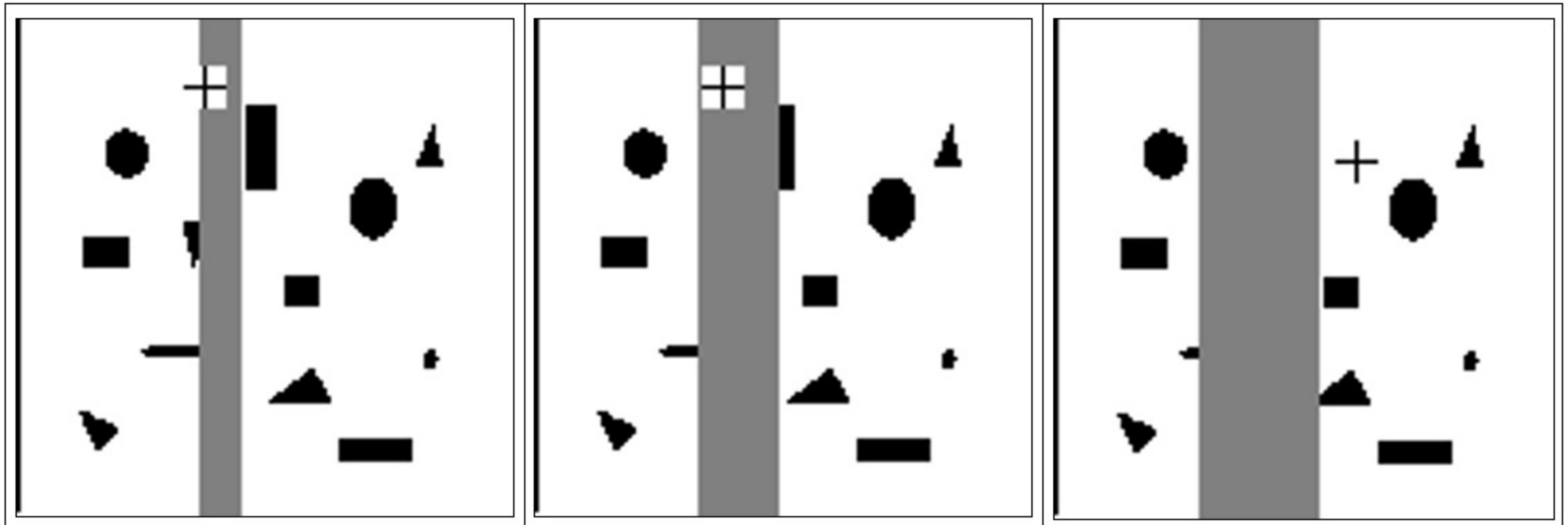
template





accumulator space

Template matching in occluded images



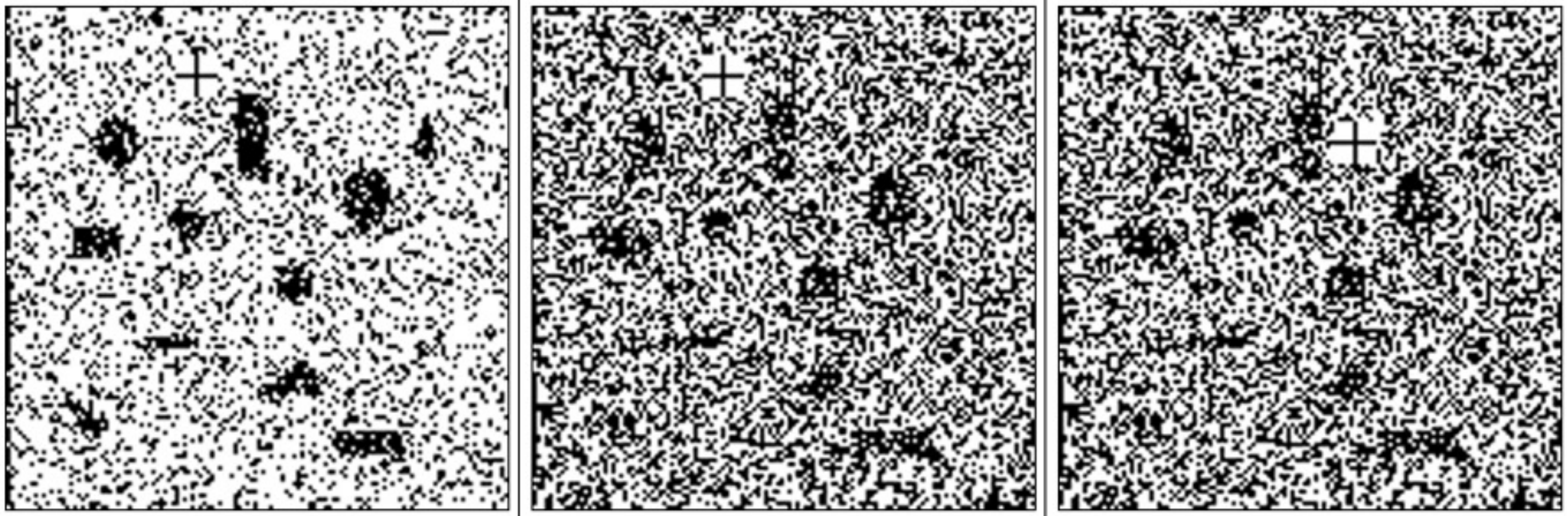
(a) extraction (of the black rectangle) in no occlusion

(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….

Convolution and correlation

Beware centring with transforms

Convolution is about application of a template

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

or by multiplying the transforms $\mathbf{I} * \mathbf{T} = F^{-1}(F(\mathbf{I}) .\times F(\mathbf{T}))$

Convolution and correlation

Beware centring with transforms

Convolution is about application of a template

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

or by multiplying the transforms $\mathbf{I} * \mathbf{T} = F^{-1}(F(\mathbf{I}) .\times F(\mathbf{T}))$

Correlation is about matching of a template $\mathbf{I} \otimes \mathbf{T} = \sum_{(\mathrm{x,y})\in \mathrm{W}} \mathbf{I}_{x,y} \mathbf{T}_{x+i,y+j}$

Convolution and correlation

Beware centring with transforms

Convolution is about application of a template

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

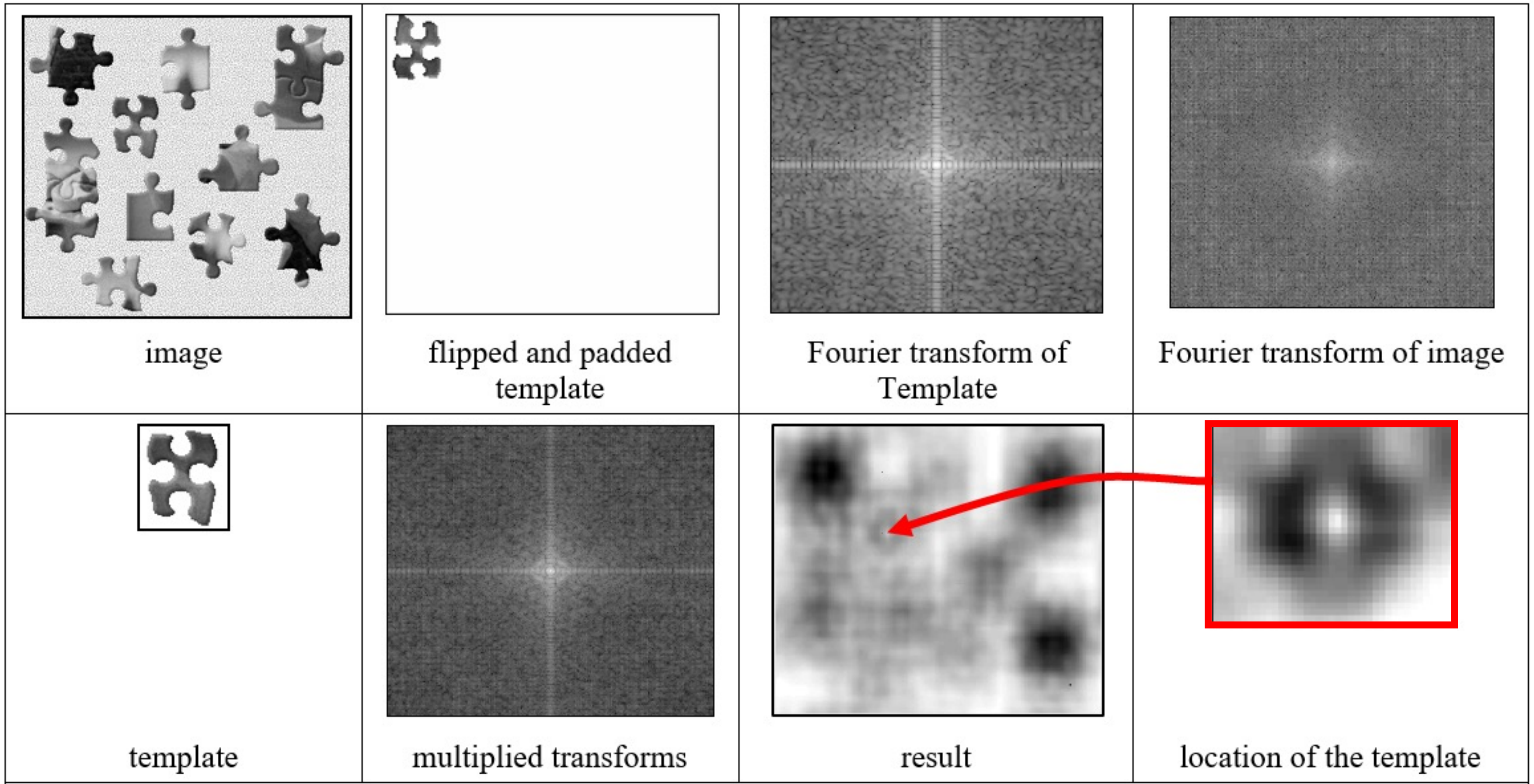
or by multiplying the transforms $\mathbf{I} * \mathbf{T} = F^{-1}(F(\mathbf{I}).\times F(\mathbf{T}))$

Correlation is about matching of a template $\mathbf{I} \otimes \mathbf{T} = \sum_{(\mathrm{x,y})\in \mathrm{W}} \mathbf{I}_{x,y}\mathbf{T}_{x+i,y+j}$

so we need to flip the Fourier template $\mathbf{I} \otimes \mathbf{T} = F^{-1}(F(\mathbf{I}).\times F(-\mathbf{T}))$

Encore, Baron Fourier!

Template matching is slow, so use FFT



Template Matching via Fourier Transform

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

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

No sliding of templates here;

Cost is 2×FFT plus multiplication

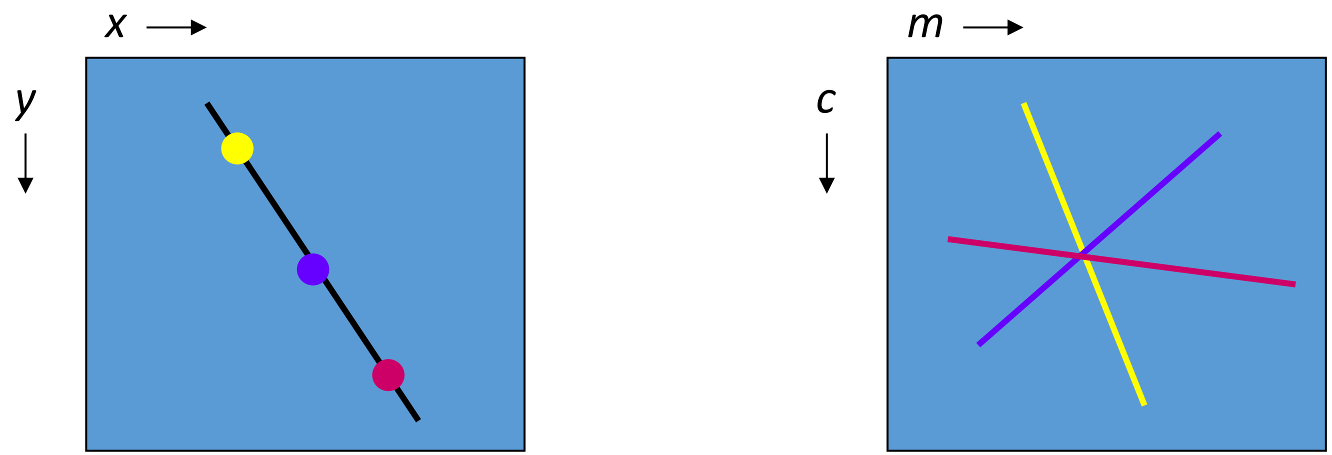
Applying template matching

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* $c = -x \times m + y$

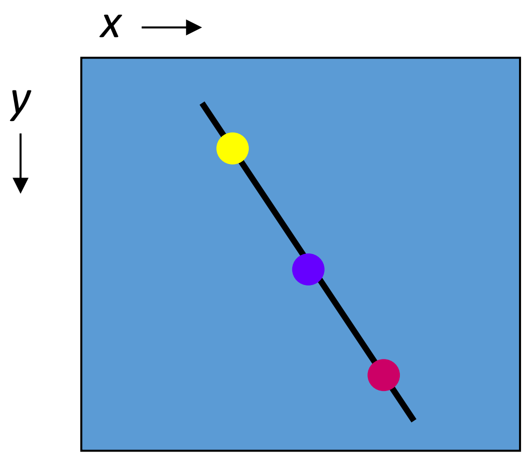


image

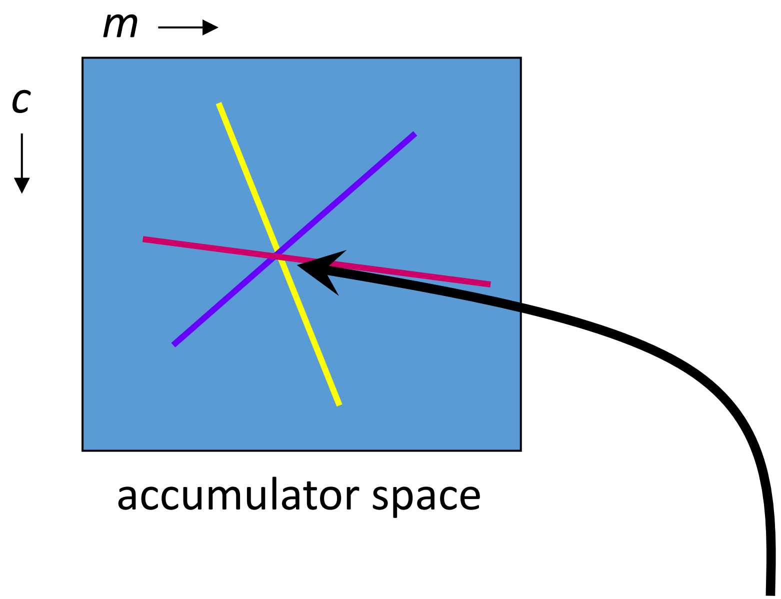
accumulator space

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

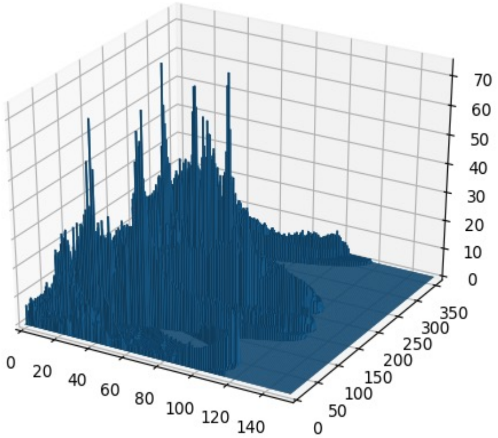
Pseudocode for HT

```
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
- Change the parameterisation
- Use foot of normal $\rho = x\cos\theta + y\sin\theta$
- Gives polar HT for lines

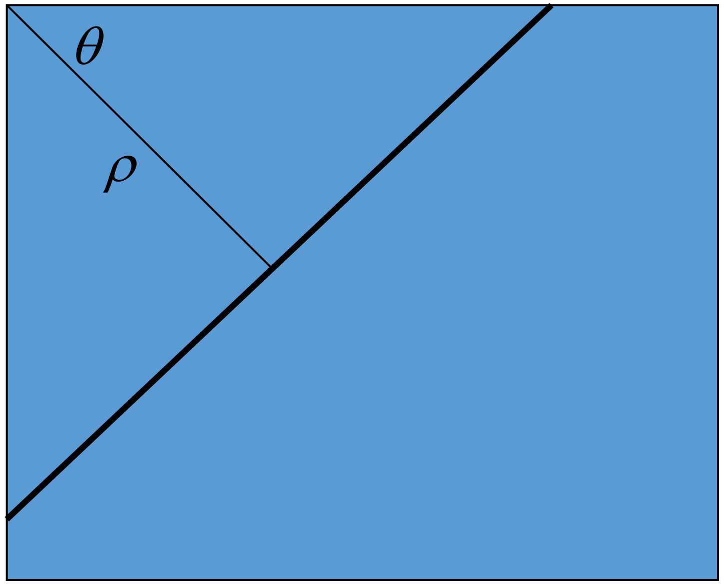
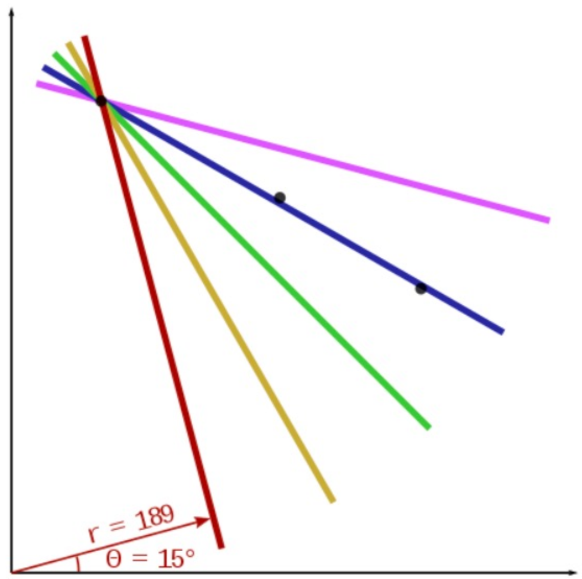
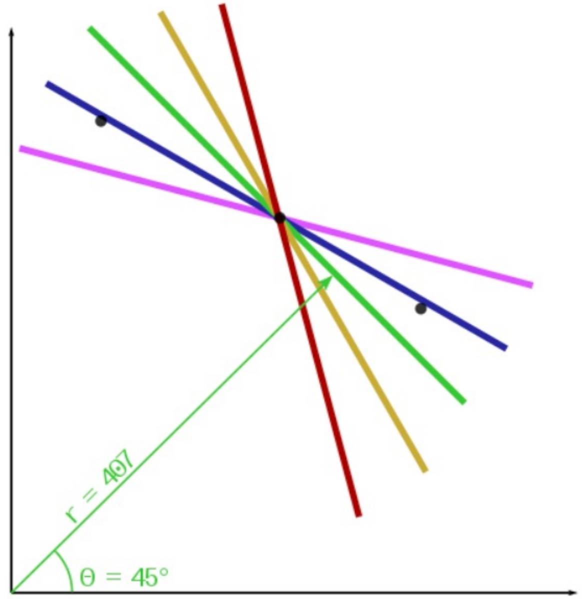


Image containing line

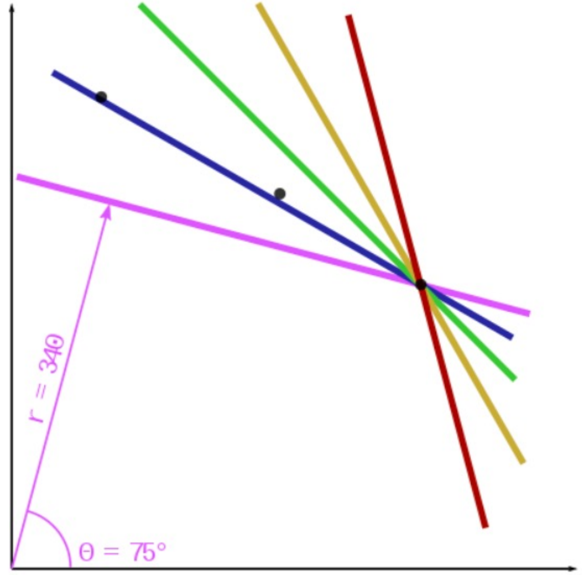
Polar Hough transform for lines



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4



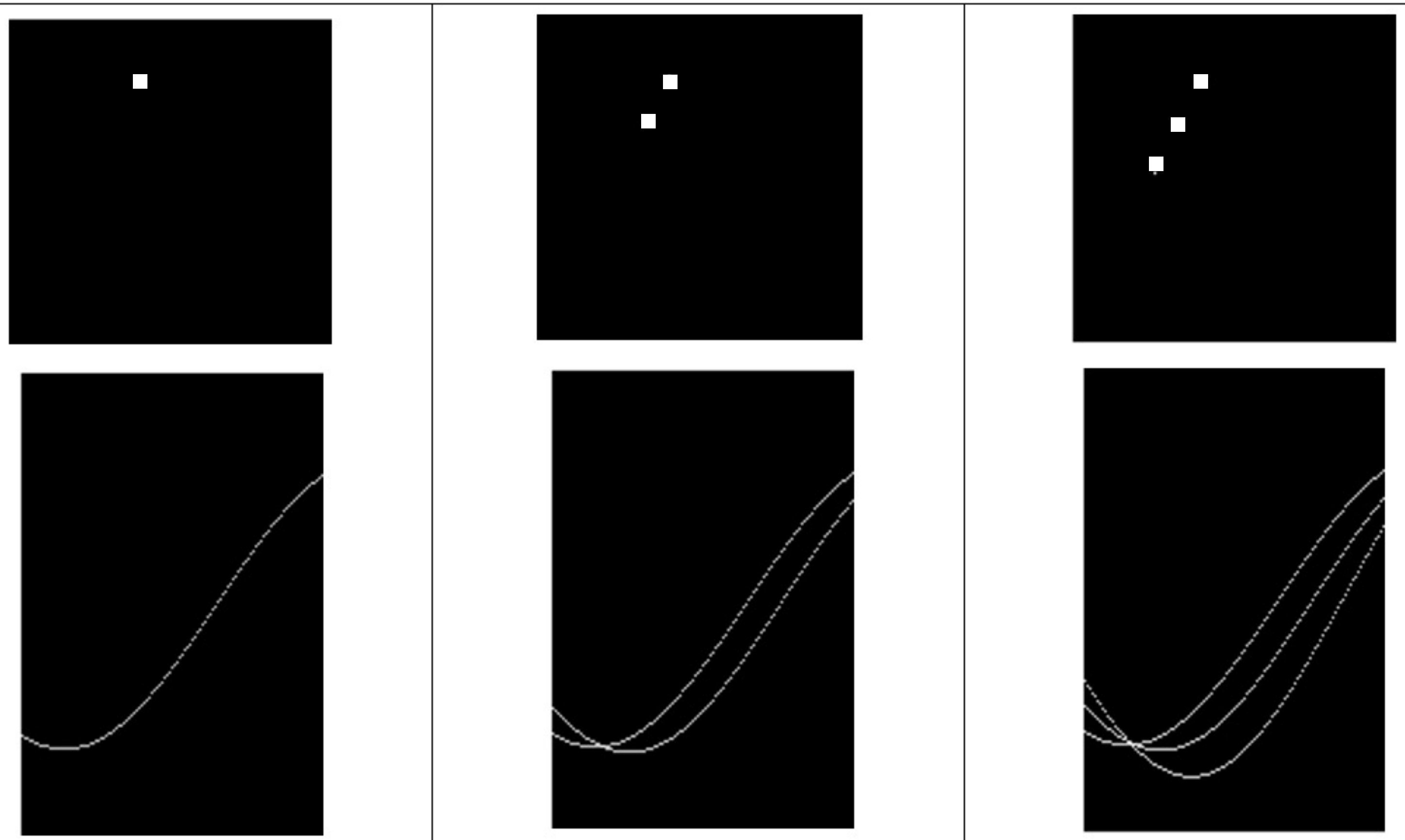
θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3



θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

[Credit Wikipedia]

Images and the accumulator space of the polar Hough transform



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

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

- Grey level image
- Spatial gradient operators ↓
- ..... Edge points .....
- Thinning and linking ↓
- <u>Lines and boundaries</u>
- Feature extraction ↓
- Shapes
- ↓
- Organisation
- Real world knowledge ↓
- Recognition

Tony!
(CBE)

UNIVERSITY OF Southampton
School of Electronics
and Computer Science