Automatic Image Analysis Exercise 3

The Generalised Hough Transformation

(Due 23.06.2017)

Mid-term Exam

- Friday, **09.06.2017, 10:15am, room H 3010**

- In place of an exercise
- Duration: ca. 30 min
- No grade, but pass is necessary to take part at the final exam

- Topics from lecture **and** exercise

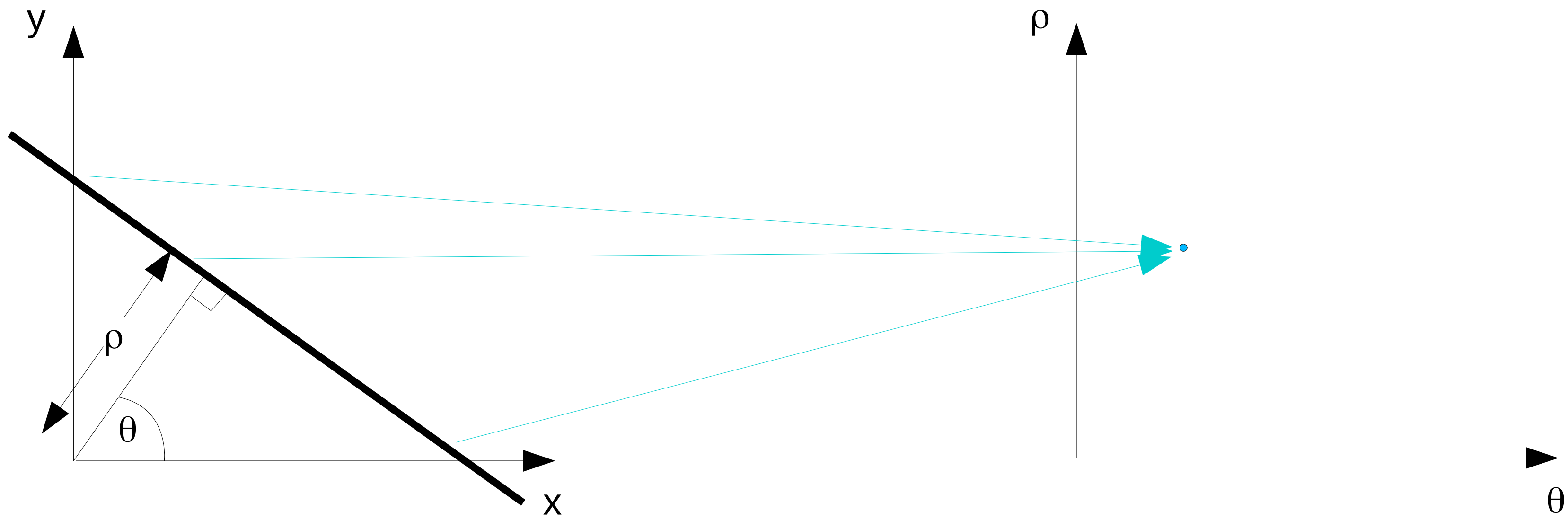
- Questions in English, answers in English or German

- No books, no calculator, no script, no paper, ...

(ρ,θ) Invariance

A line in the image plane is defined by parameters (ρ,θ):

In parameter space over (ρ,θ), the line collapses to a point:



Invariant for all pixels on the line:

$$\boxed{\begin{pmatrix} x \\ y \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} = \rho}$$

Line Detection

0) Compute (ρ,θ) at each line-pixel
1) Increment the parameter space at coordinate (ρ,θ)
2) A line leads to a pronounced maximum

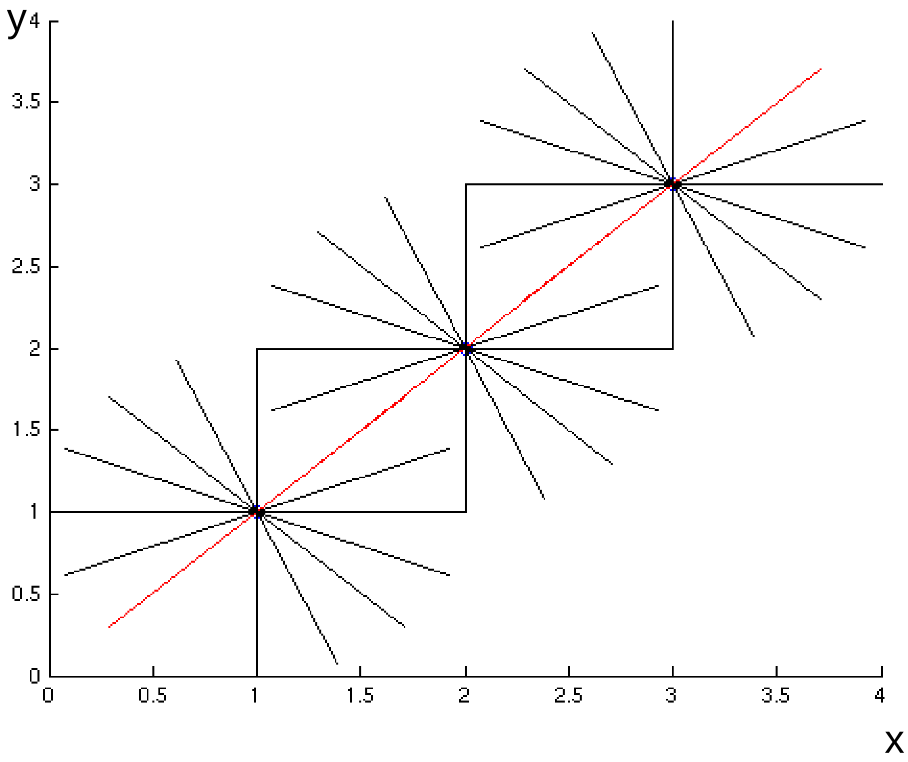
Radon Transformation

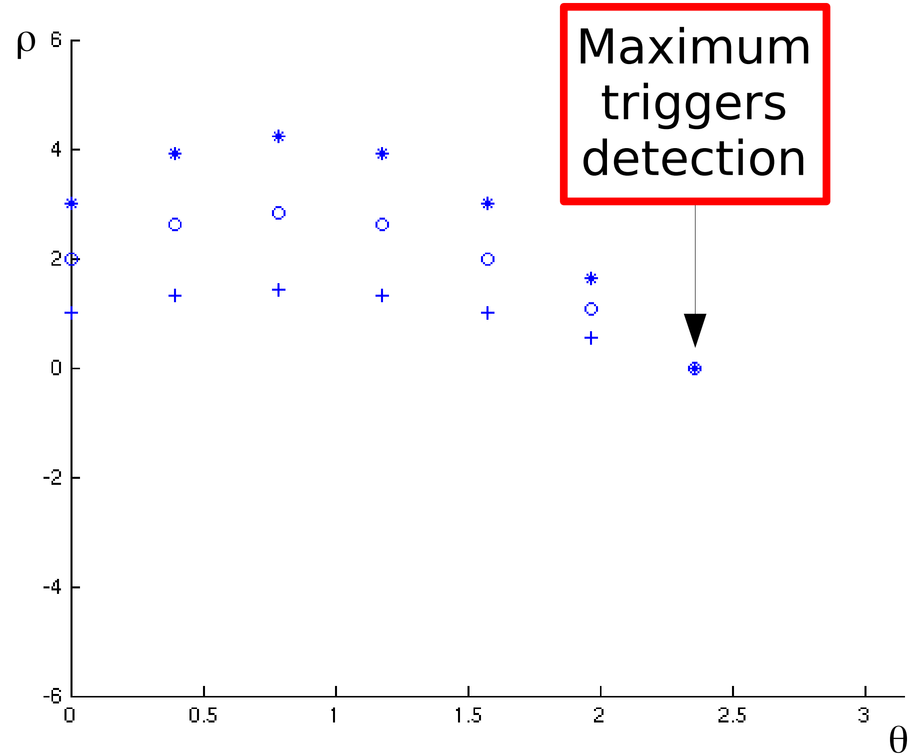
Problem: (ρ,θ) are unknowns

- Infinite number of potential lines intersect at every point

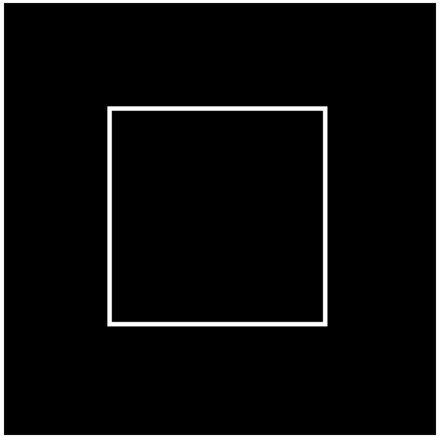
Solution:

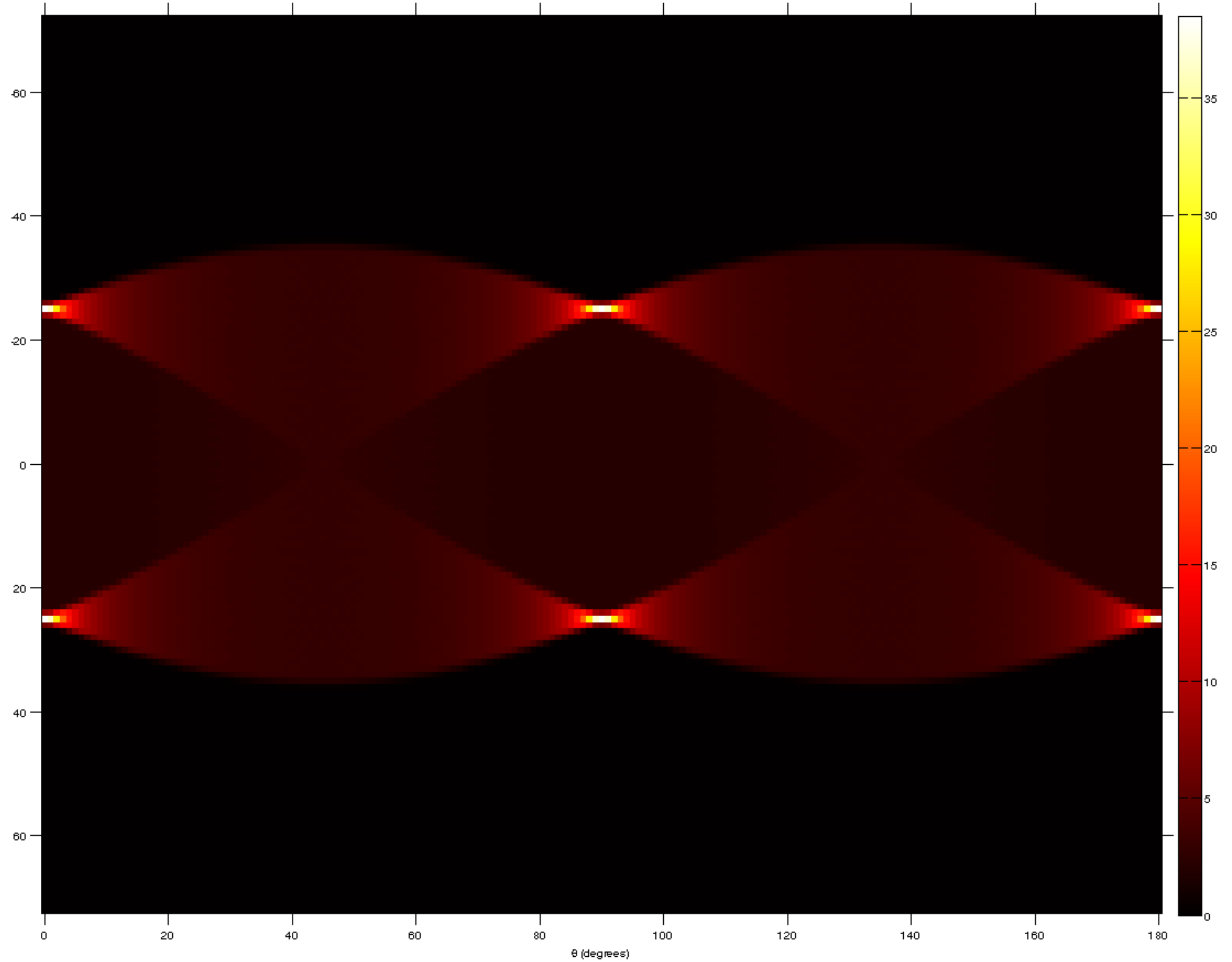
- Each line through a point (x,y) is defined by fixing its direction θ.
- ρ can be computed from x,y and θ.
- For all directions θ: Compute ρ and increase element (ρ,θ) in parameter space



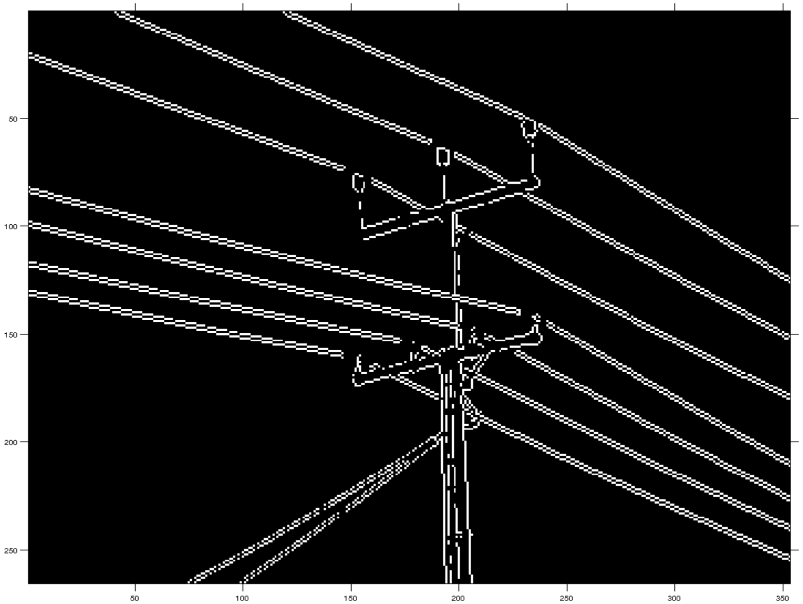


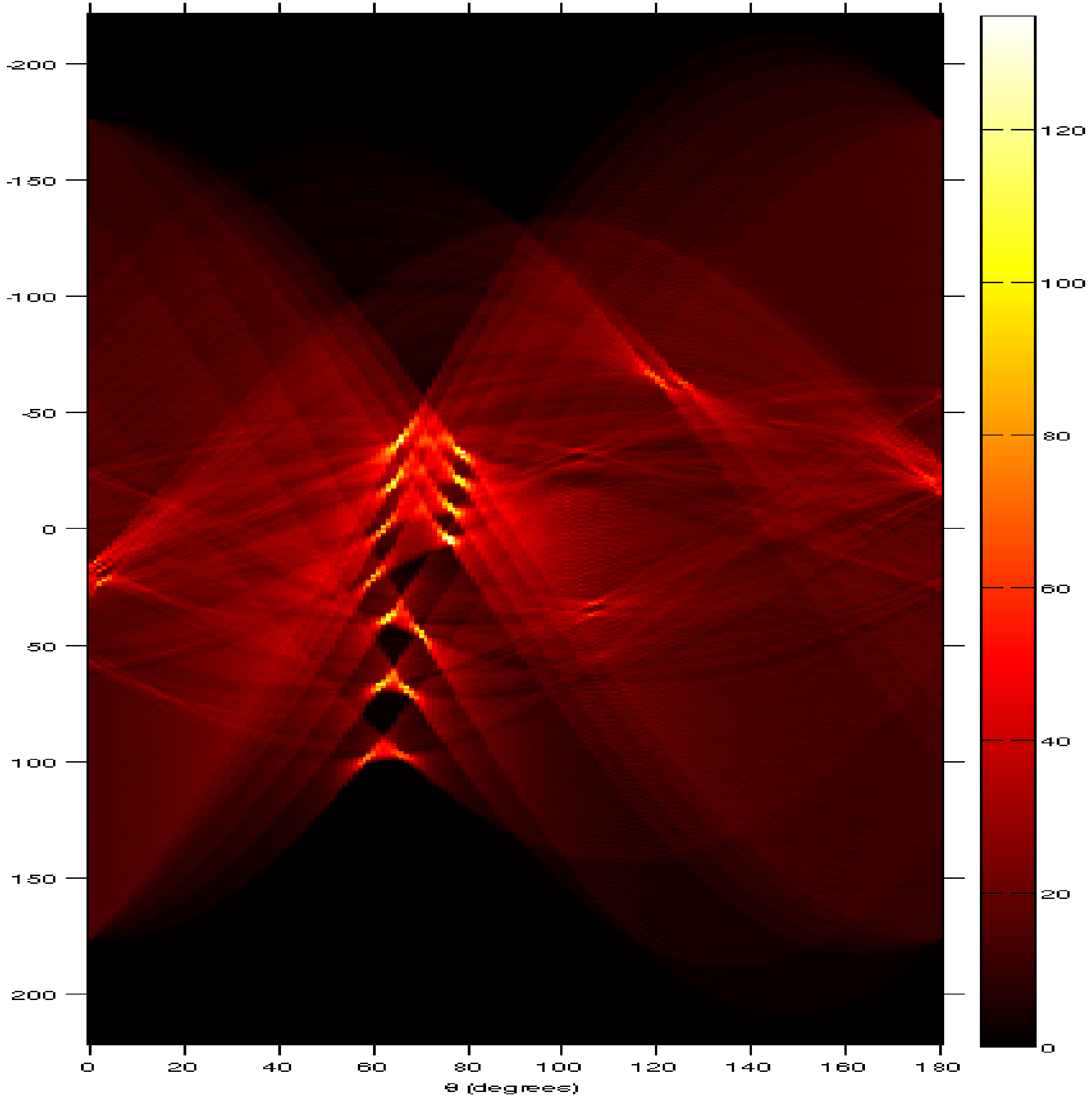
Radon Transformation



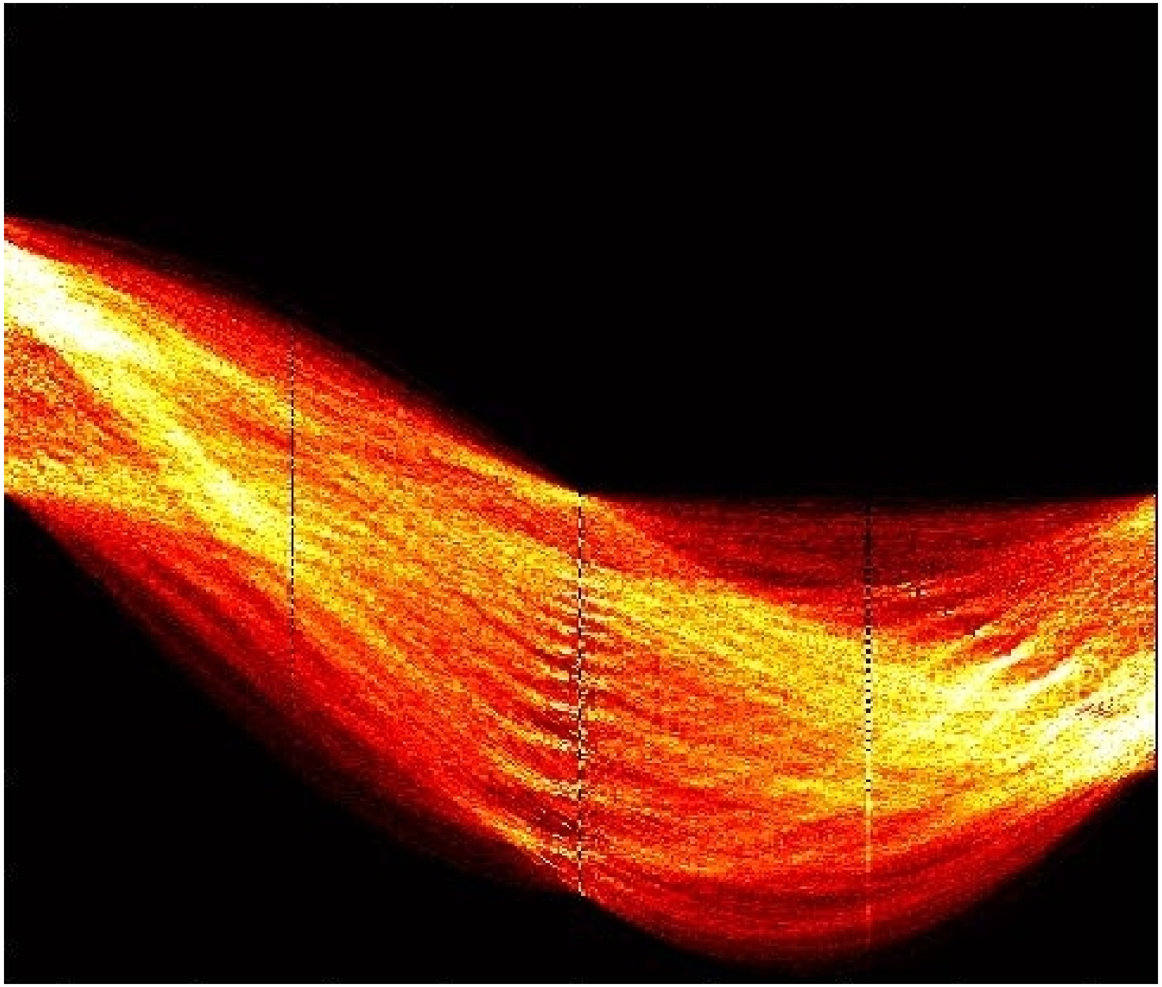


Radon Transformation



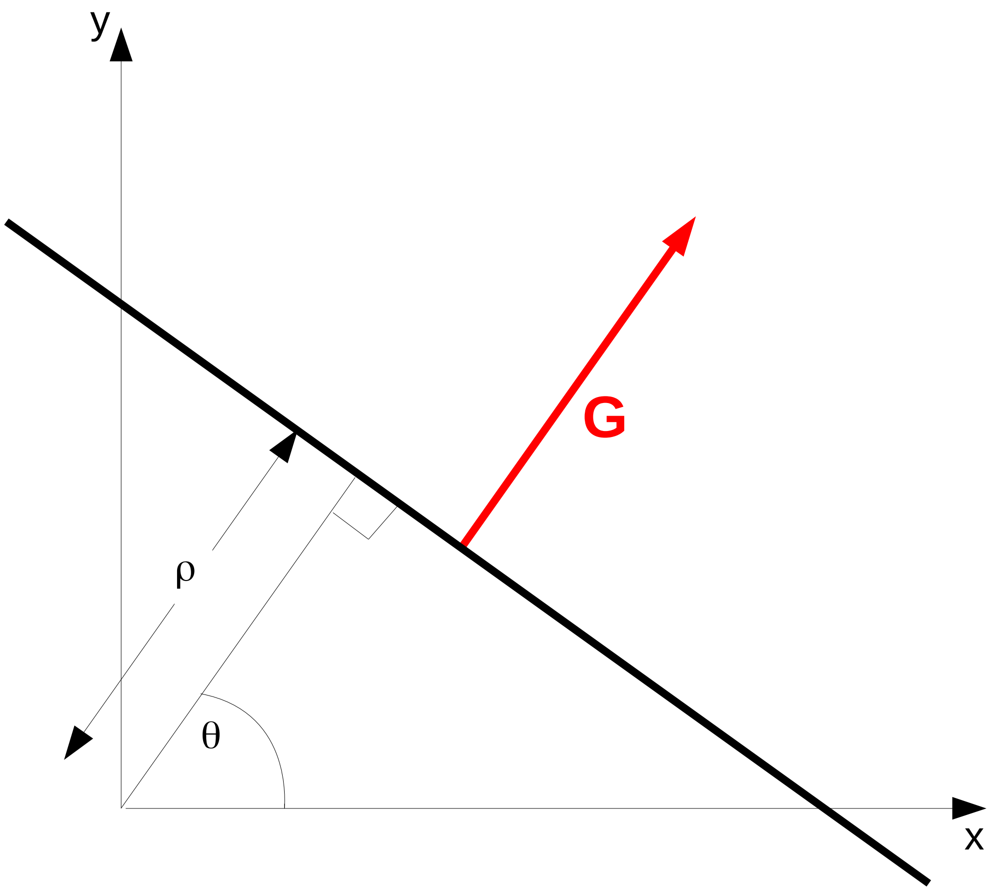


Radon Transformation



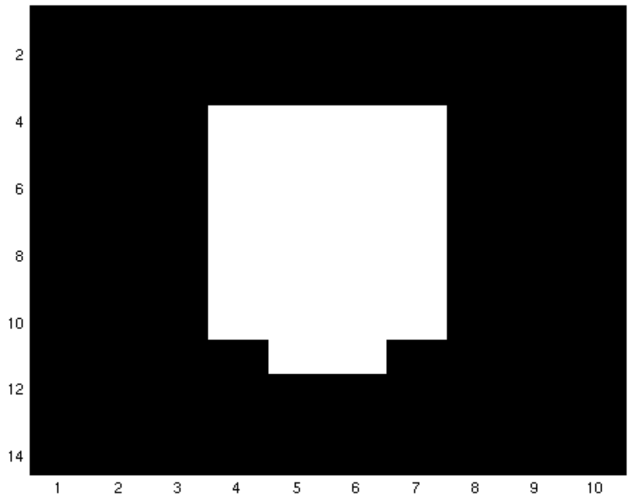
Hough Line Detector

- Alternative (more intelligent) solution: Use gradient information
- The local gradient direction fixes θ, and (ρ,θ) can be calculated directly
- It is no longer necessary to consider every possible line through each pixel!



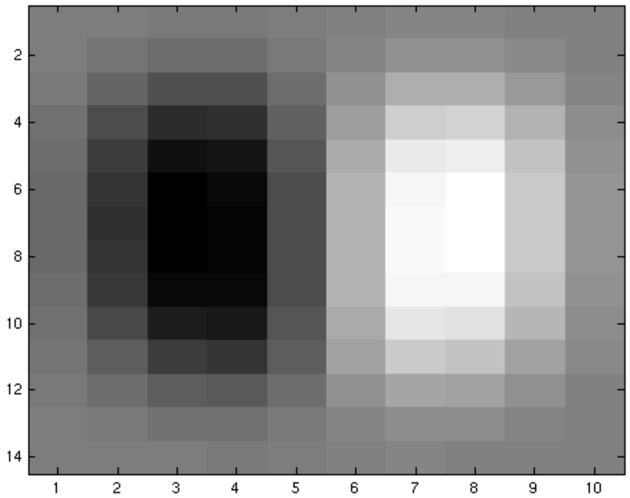
Hough Line Detector

- Alternative (more intelligent) solution: Use gradient information
- The local gradient direction fixes θ, and (ρ,θ) can be calculated directly
- It is no longer necessary to consider every possible line through each pixel!



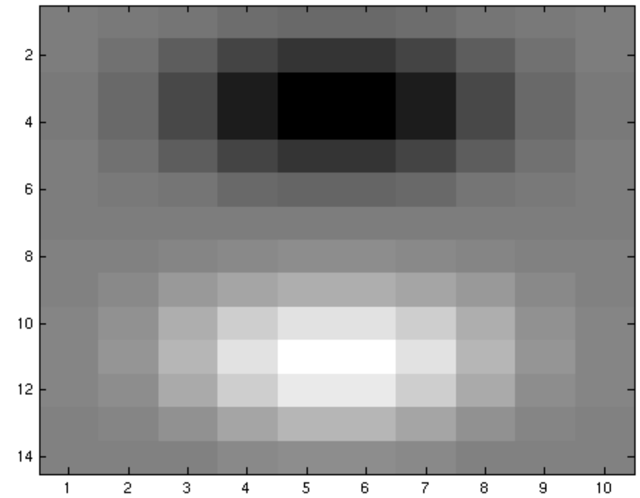
Image

$$B(x,y)$$



Gradient in x direction

$$G_x(x,y)=\frac{\partial}{\partial x}B(x,y)$$

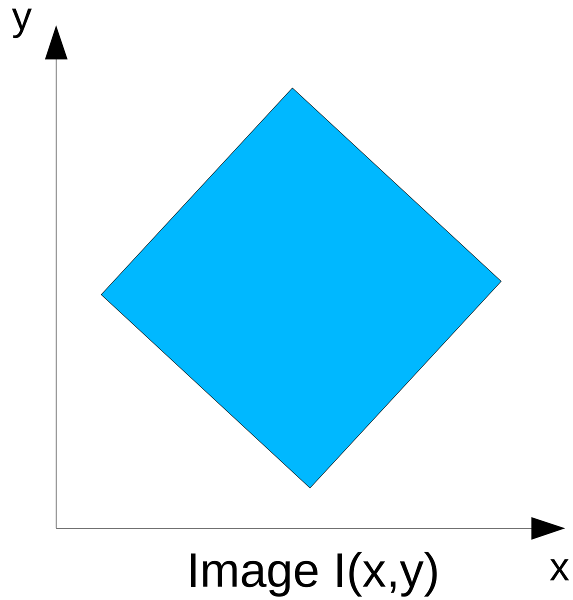


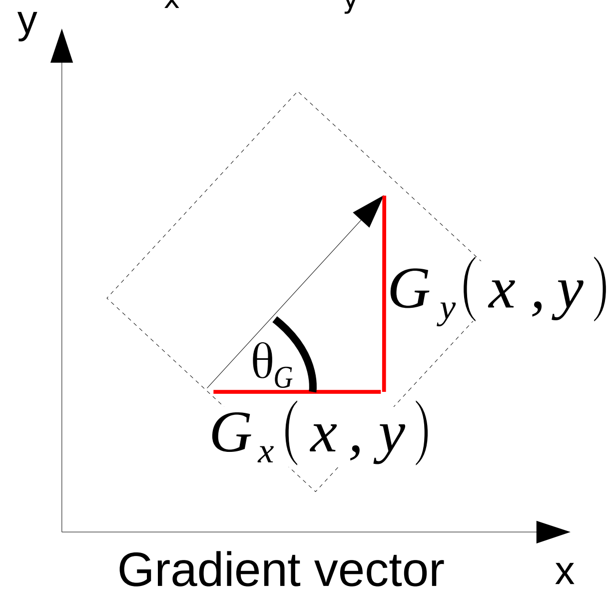
Gradient in y direction

$$G_y(x,y)=\frac{\partial}{\partial y}B(x,y)$$

Hough Line Detector

- Each pixel (x,y) is associated with a gradient vector ($G_x(x,y)$,$G_y(x,y)$).





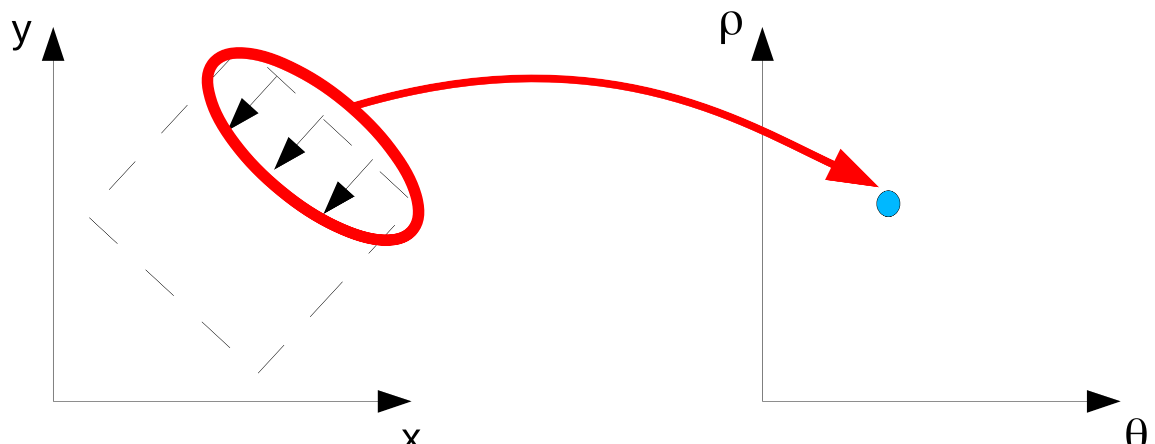
- The angle θ_G(x,y) is easy to obtain, e.g. in C++:

$$\theta_G(x,y)=atan2(G_y(x,y),G_x(x,y))$$

- Once θ_G(x,y) is known, coordinates (ρ,θ) in parameter space are uniquely determined:

$$\begin{pmatrix} x \\ y \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} = \rho$$

Hough Liniendetektor



Line in image I(x,y)

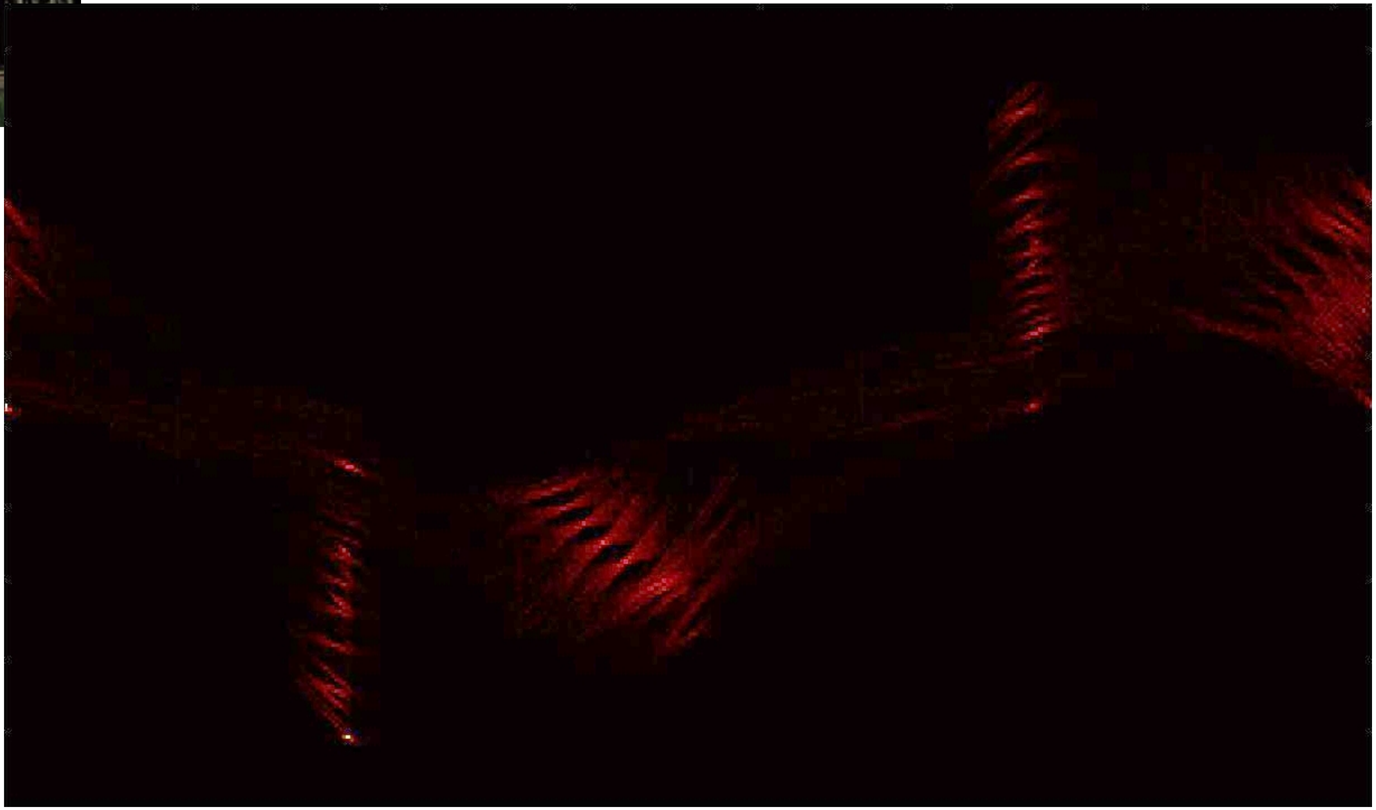
Point in $H_L(\rho,\theta)$

1) Compute gradients in x and y
2) Determine ρ und θ using the local gradient direction
3) Increase $H_L(\rho,\theta)$ by the local gradient magnitude:

$$H_L(\rho,\theta) \leftarrow H_L(\rho,\theta) + \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

A line in I(x,y) produces a pronounced maximum $H_L(\rho,\theta)$.

Hough Liniendetektor



Generalised Hough Transformation

- An image I(x,y) is processed to discover instances of an object
- The object of interest is known from an example image O(x,y)
- Processing is once again concerned with accumulating „votes“ in a parameter space $H_G(x,y,\theta,s)$.

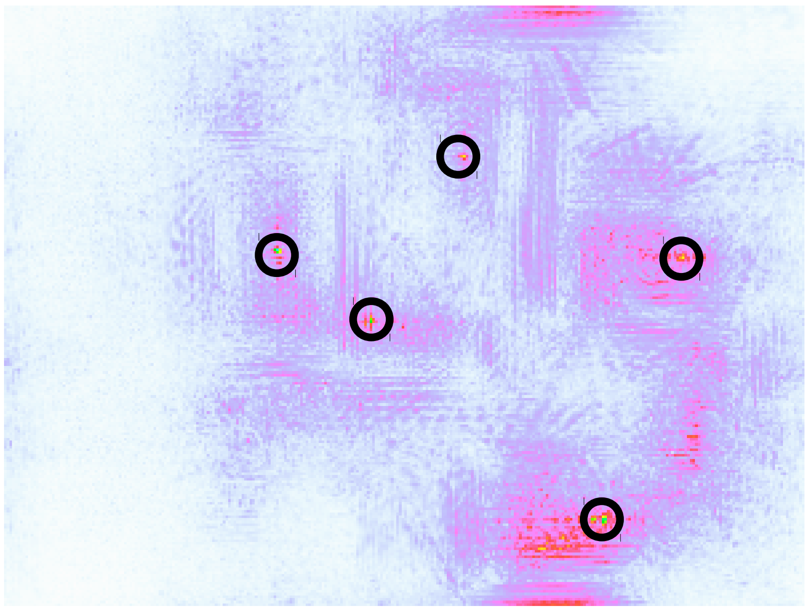
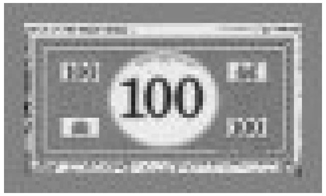


Image I(x,y) → $H_G(x,y,\theta,s)$

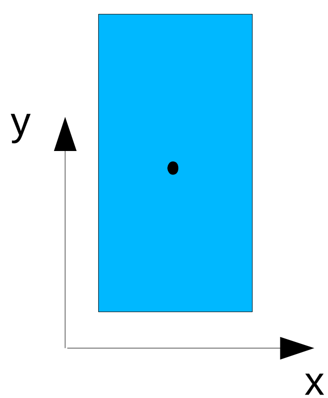


Object O(x,y)

Pronounced maxima in $H_G(x,y,\theta,s)$ indicate that the image contains an object of interest centered at (x,y), scaled by factor s, and rotated by angle θ.

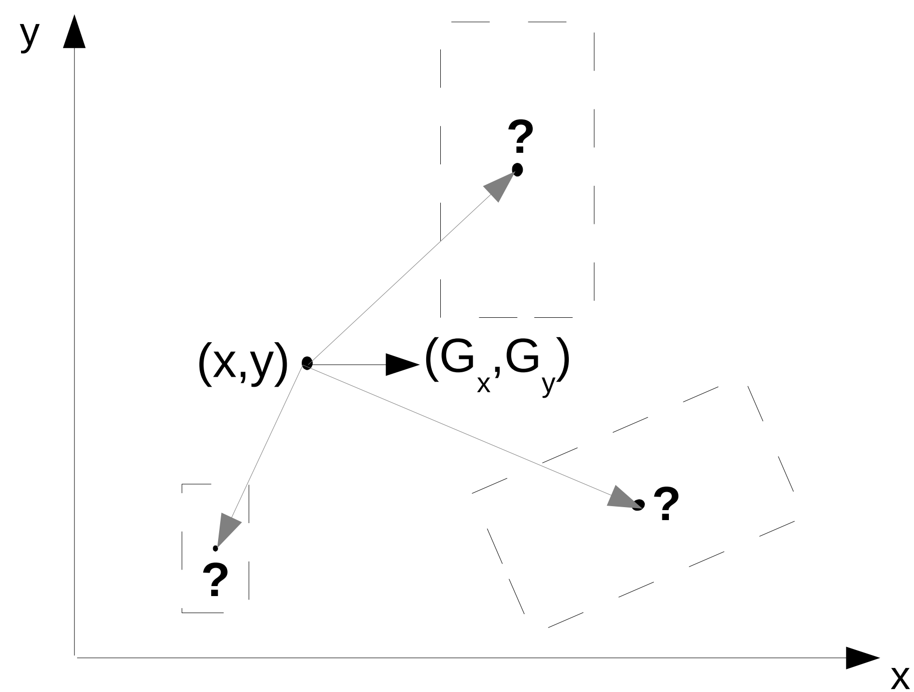
Generalised Hough Transformation

- **Given**: the gradient vector (G_x,G_y) at each image location
- **Required**: possible locations of the object centre
  - How can possible centre locations be infered from local gradient directions?
  - Which locations in $H_G(x,y,\theta,s)$ are to be increased?



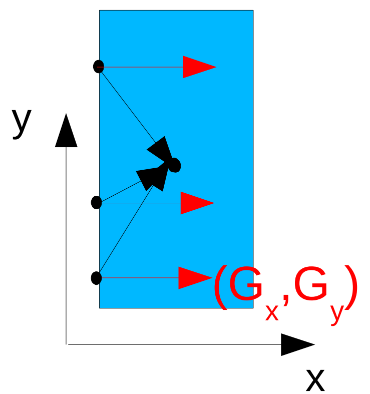
Objekt O(x,y)

Image I(x,y)



Generalised Hough Transformation

- Exploit relation between local gradient and object center
- The local gradient is ambiguous
  - The same gradient occurs at different points on the object
  - A given gradient direction may correspond to several possible object centres
  - In general, several locations in $H_G(x,y,\theta,s)$ must be increased



Objekt O(x,y)

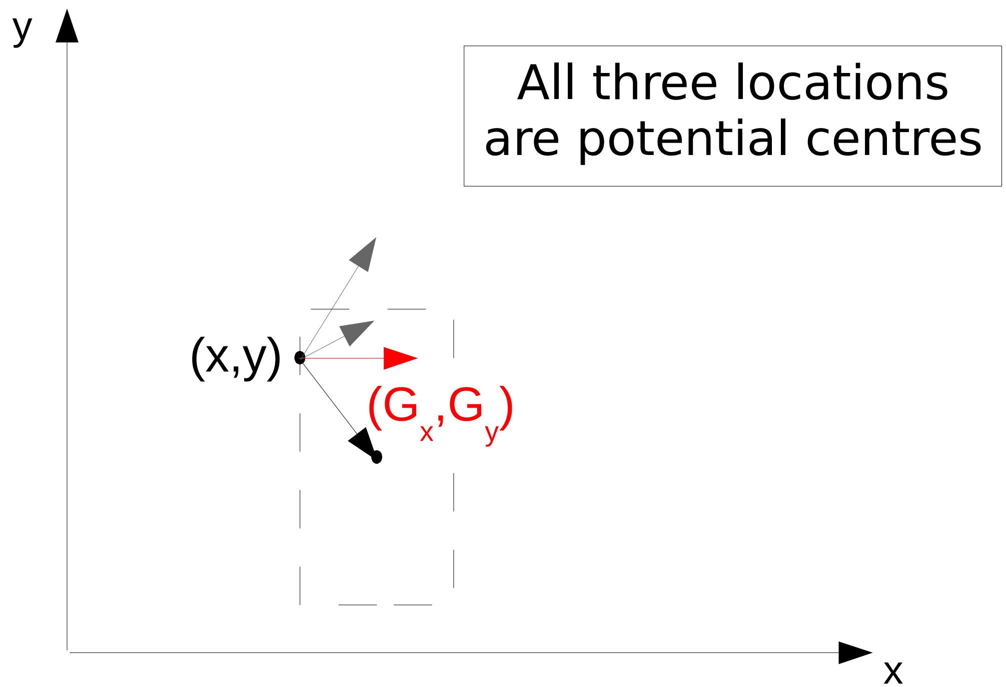
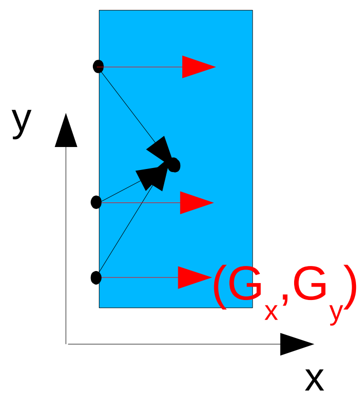


Image I(x,y)

Generalised Hough Transformation

- Exploit relation between local gradient and object center
- The local gradient is ambiguous
  - The same gradient occurs at different points on the object
  - A given gradient direction may correspond to several possible object centres
  - In general, several locations in $H_G(x,y,\theta,s)$ must be increased



Objekt O(x,y)

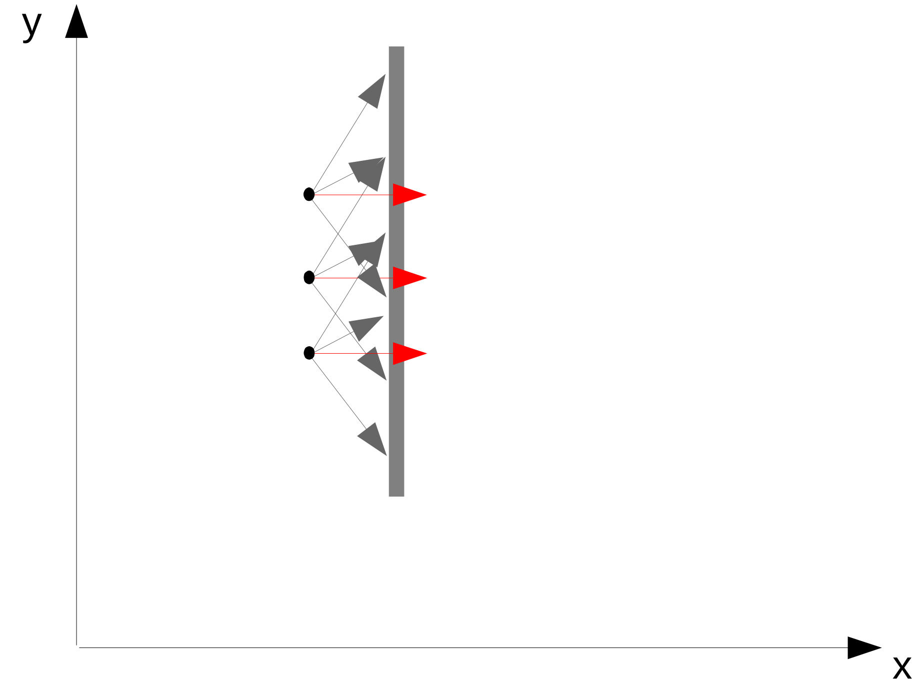
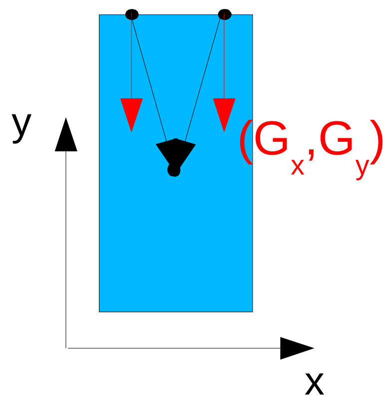


Image I(x,y)

Generalised Hough Transformation

- Exploit relation between local gradient and object center
- The local gradient is ambiguous
  - The same gradient occurs at different points on the object
  - A given gradient direction may correspond to several possible object centres
  - In general, several locations in $H_G(x,y,\theta,s)$ must be increased



Objekt O(x,y)

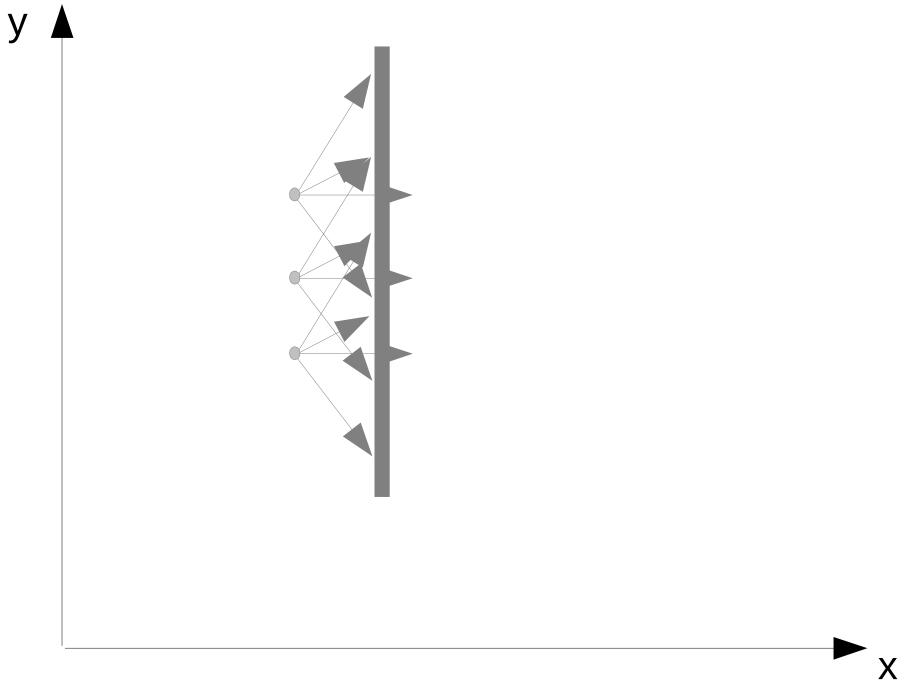
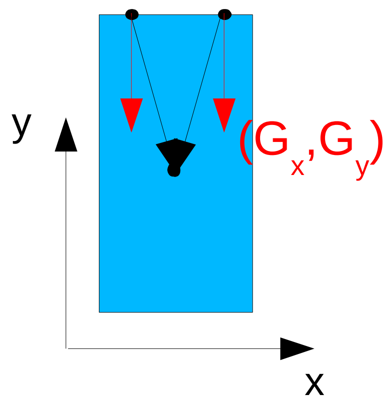


Image I(x,y)

Generalised Hough Transformation

- Exploit relation between local gradient and object center
- The local gradient is ambiguous
  - The same gradient occurs at different points on the object
  - A given gradient direction may correspond to several possible object centres
  - In general, several locations in $H_G(x,y,\theta,s)$ must be increased



Objekt O(x,y)

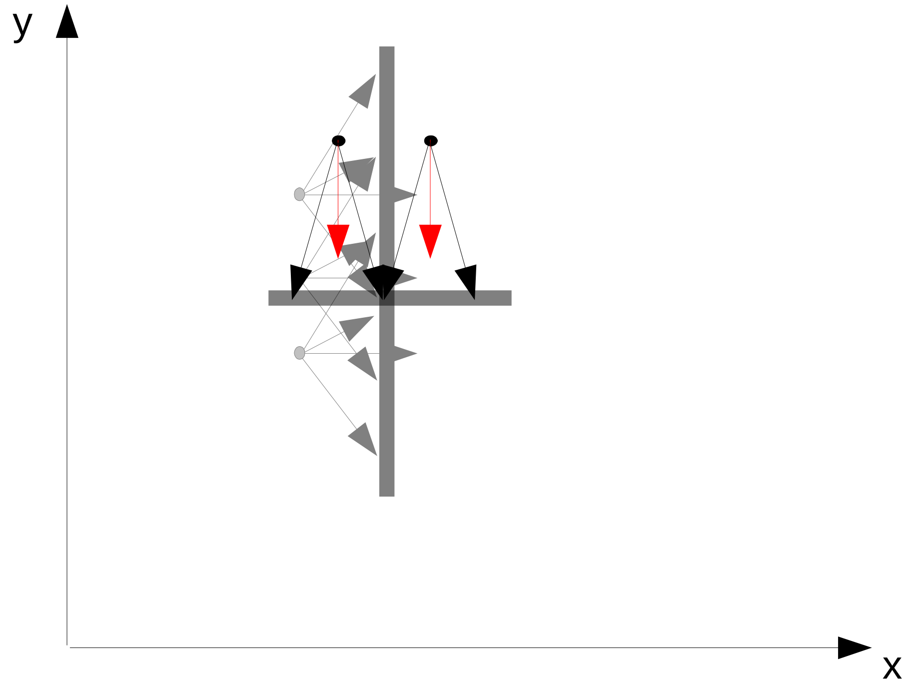
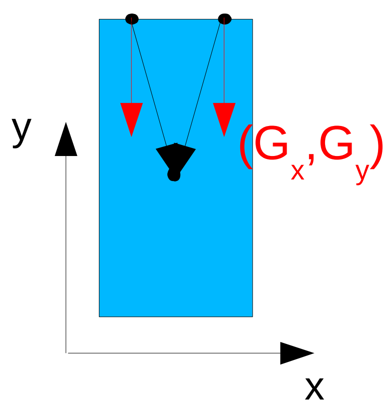


Image I(x,y)

Generalised Hough Transformation

- Exploit relation between local gradient and object center
- The local gradient is ambiguous
  - The same gradient occurs at different points on the object
  - A given gradient direction may correspond to several possible object centres
  - In general, several locations in $H_G(x,y,\theta,s)$ must be increased



Objekt O(x,y)

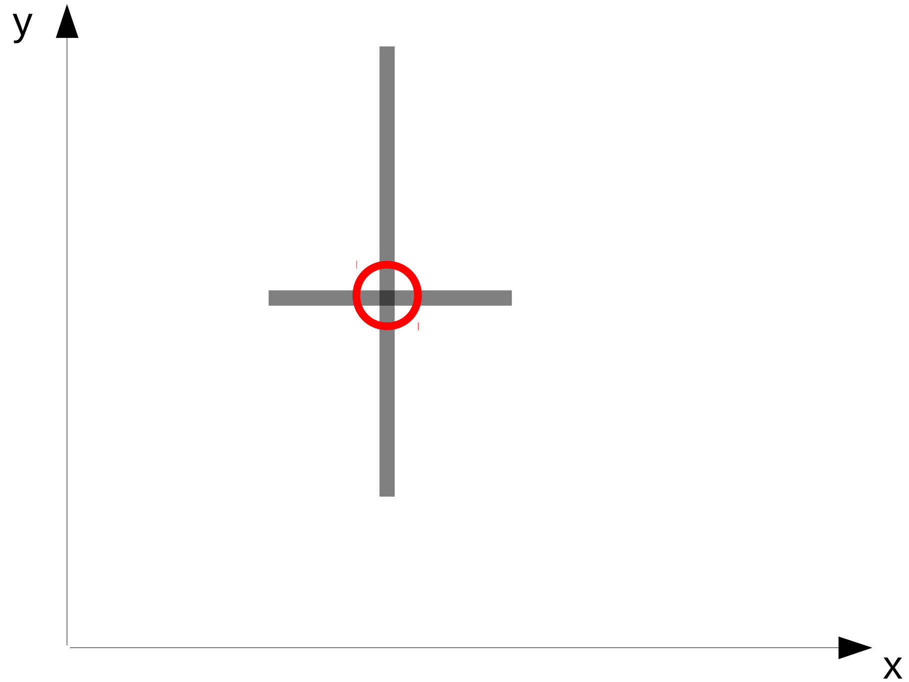
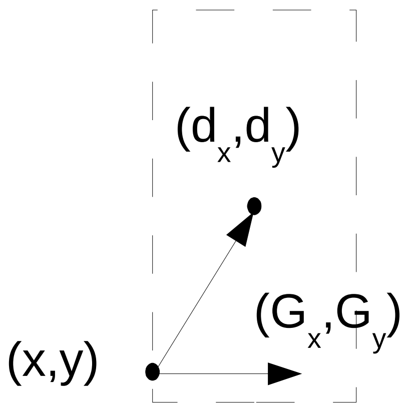


Image I(x,y)

Generating the R-Table

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



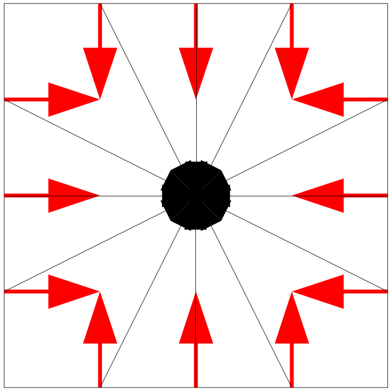
$\theta=\frac{0\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{1,1}, d_y^{(1,1)}$	$\cdots$	d_x^{1,N_1}, d_y^{1,N_1}
$\theta=\frac{1\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{2,1}, d_y^{(2,1)}$	$\cdots$	d_x^{2,N_2}, d_y^{2,N_2}
$\theta=\frac{2\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{3,1}, d_y^{(3,1)}$	$\cdots$	d_x^{3,N_3}, d_y^{3,N_3}
$\vdots$	$\rightarrow$		$\vdots$	
$\theta=\frac{(A-1)\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{A,1}, d_y^{(A,1)}$	$\cdots$	d_x^{A,N_A}, d_y^{A,N_A}

Row T of the R-Table

- Applies when local gradient direction is close to $2\pi(T-1)/A$
- The displacement to the centre is $(d_x^{T,1}, d_y^{T,1})$ or $(d_x^{T,2}, d_y^{T,2})$ or … or $(d_x^{T,NT}, d_y^{T,NT})$

Generating the R-Table

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



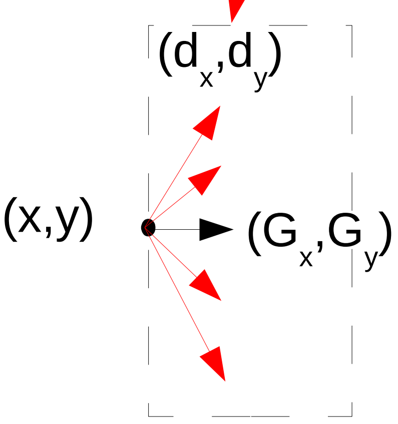
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[90, 120)	
[120, 150)	
[150, 180)	

Applying the R-Table

Considering an image location (x,y) with associated gradient direction θ:

- θ is discretised by dividing the interval $[0,2\pi]$ into *A* sub-intervals
- The sub-interval assigned to θ identifies a row of the R-Table
- The row contains displacements $(d_x^{T,1..NT}, d_y^{T,1..NT})$ to possible object centres
- H_G can be increased by the local gradient magnitude at points

  $(x+d_x^{T,1}, y+d_y^{T,1}) \ldots (x+d_x^{T,NT}, y+d_y^{T,NT})$



$\theta=\frac{0\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{1,1}, d_y^{(1,1)}$	$\cdots$	d_x^{1,N_1}, d_y^{1,N_1}
$\theta=\frac{1\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{2,1}, d_y^{(2,1)}$	$\cdots$	d_x^{2,N_2}, d_y^{2,N_2}
$\theta=\frac{2\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{3,1}, d_y^{(3,1)}$	$\cdots$	d_x^{3,N_3}, d_y^{3,N_3}
$\vdots$	$\rightarrow$		$\vdots$	
$\theta=\frac{(A-1)\cdot 2\pi}{A}$	$\rightarrow$	$d_x^{A,1}, d_y^{(A,1)}$	$\cdots$	d_x^{A,N_A}, d_y^{A,N_A}

Applying the R-Table

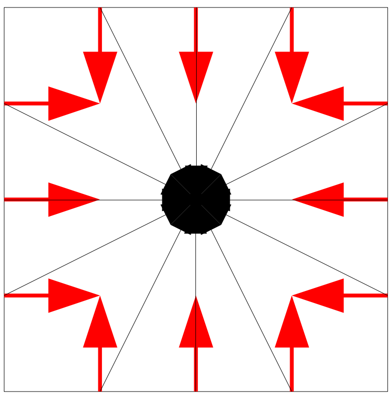
Considering an image location (x,y) with associated gradient direction θ:

- θ is discretised by dividing the interval $[0,2\pi]$ into *A* sub-intervals
- The sub-interval assigned to θ identifies a row of the R-Table
- The row contains displacements $(d_x^{T,1..NT}, d_y^{T,1..NT})$ to possible object centres
- H_G can be increased by the local gradient magnitude at points $(x+d_x^{T,1}, y+d_y^{T,1}) \ldots (x+d_x^{T,NT}, y+d_y^{T,NT})$

These steps compute $H_G(x,y,\theta=0,s=1)$.
Scaled or rotated versions of the object are not detected.

Generating the R-Table

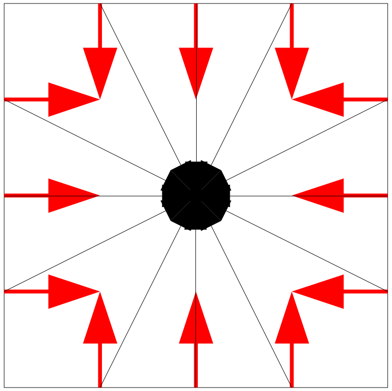
The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



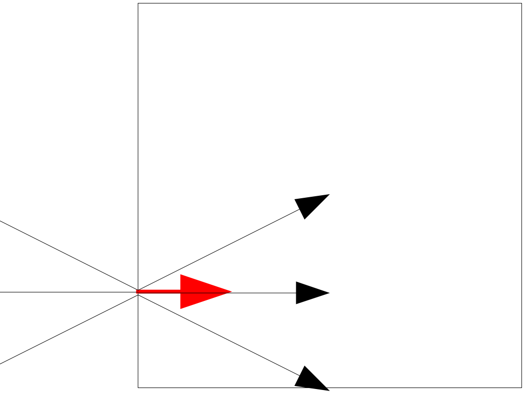
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	

Generating the R-Table

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



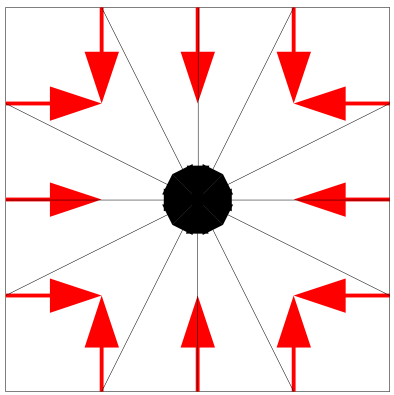
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	

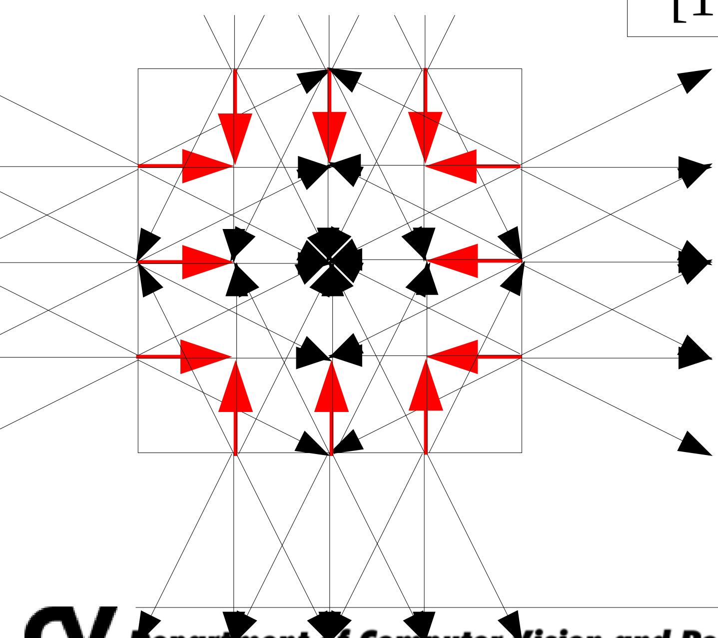


Generating the R-Table

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre

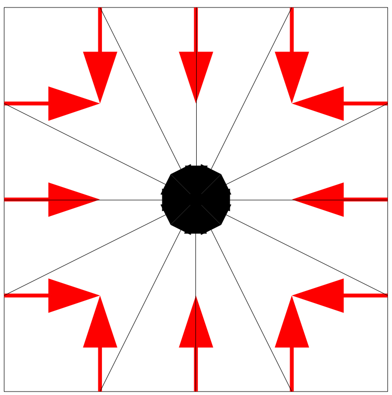
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	



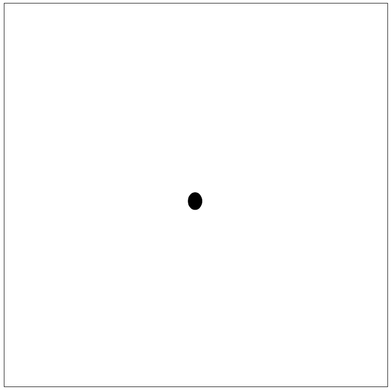


Generating the R-Table

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre

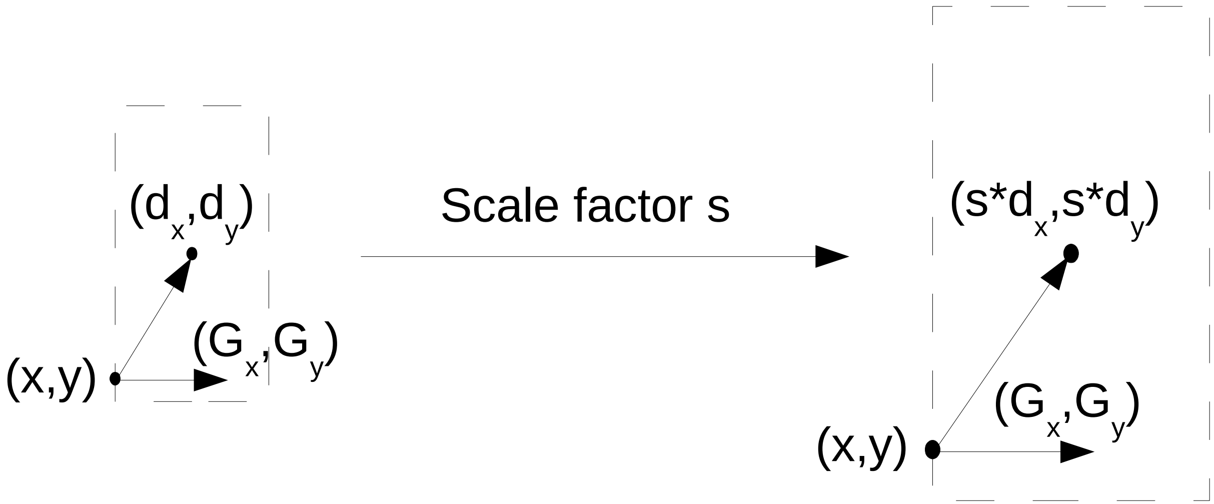


θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	



Changing the R-Table: Scale

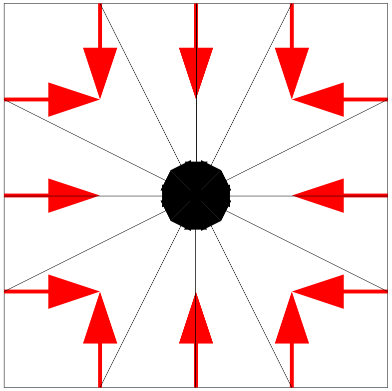
- To detect objects of different scales, the R-Table is modified to describe a scaled version of the original object
- The updates to H_G are repeated at different scales to compute $H_G(x,y,\theta=0,s)$



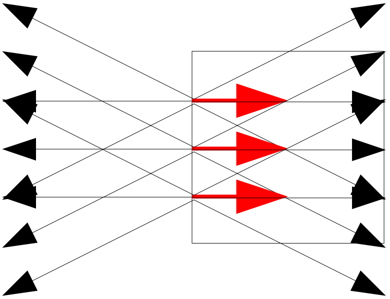
The displacements stored in the R-Table are multiplied by the desired scale factor before H_G is updated for the current scale

Changing the R-Table: Scale

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre

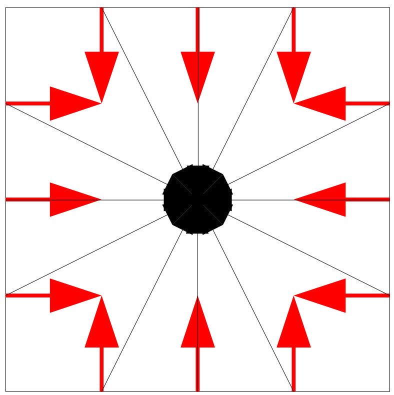


θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	

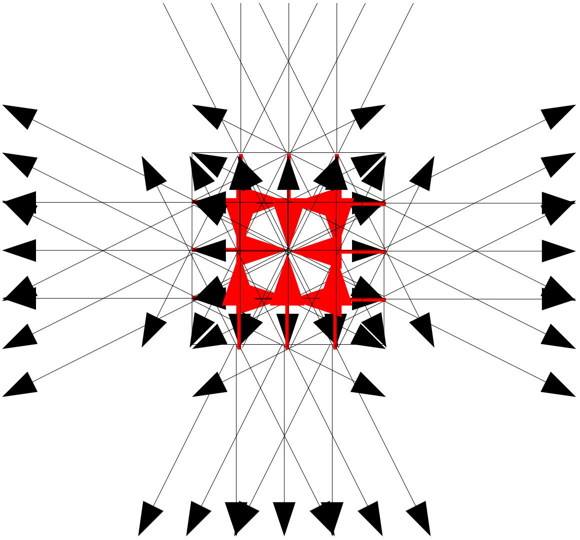


Changing the R-Table: Scale

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre

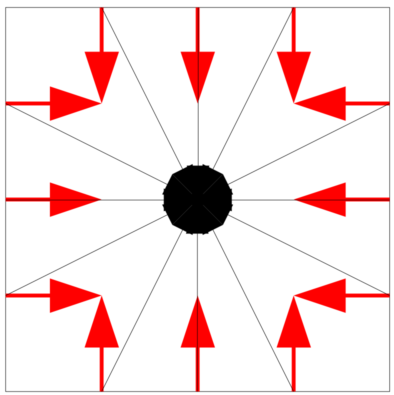


θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	

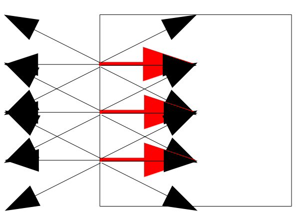


Changing the R-Table: Scale

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



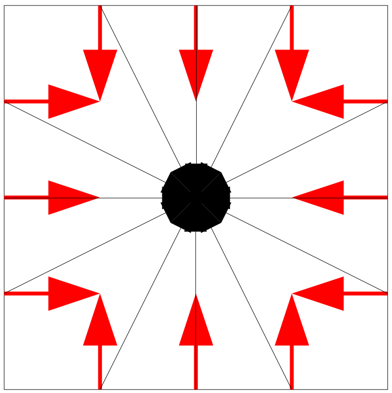
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	



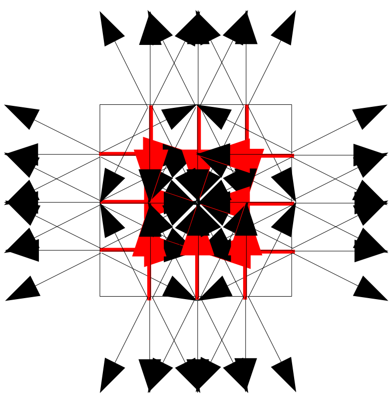
θ	
[0, 30)	$(1,0.5),(1,0),(1,-0.5),(-1,0.5),(-1,0),(-1,-0.5)$
[30, 60)	
[60, 90)	
[90, 120)	$(0.5,1),(0,1),(-0.5,1),(0.5,-1),(0,-1),(-0.5,-1)$
[120, 150)	
[150, 180)	

Changing the R-Table: Scale

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



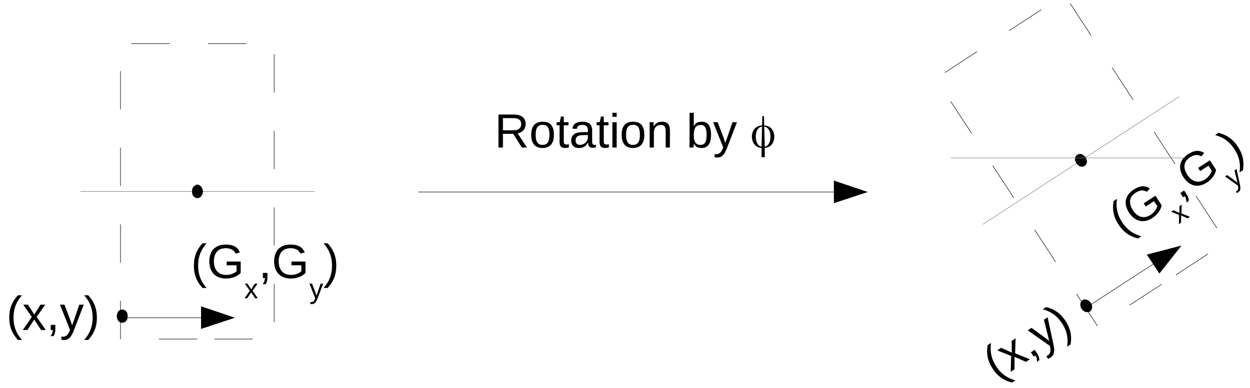
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	



θ	
[0, 30)	$(1,0.5),(1,0),(1,-0.5),(-1,0.5),(-1,0),(-1,-0.5)$
[30, 60)	
[60, 90)	
[90, 120)	$(0.5,1),(0,1),(-0.5,1),(0.5,-1),(0,-1),(-0.5,-1)$
[120, 150)	
[150, 180)	

Changing the R-Table: Orientation

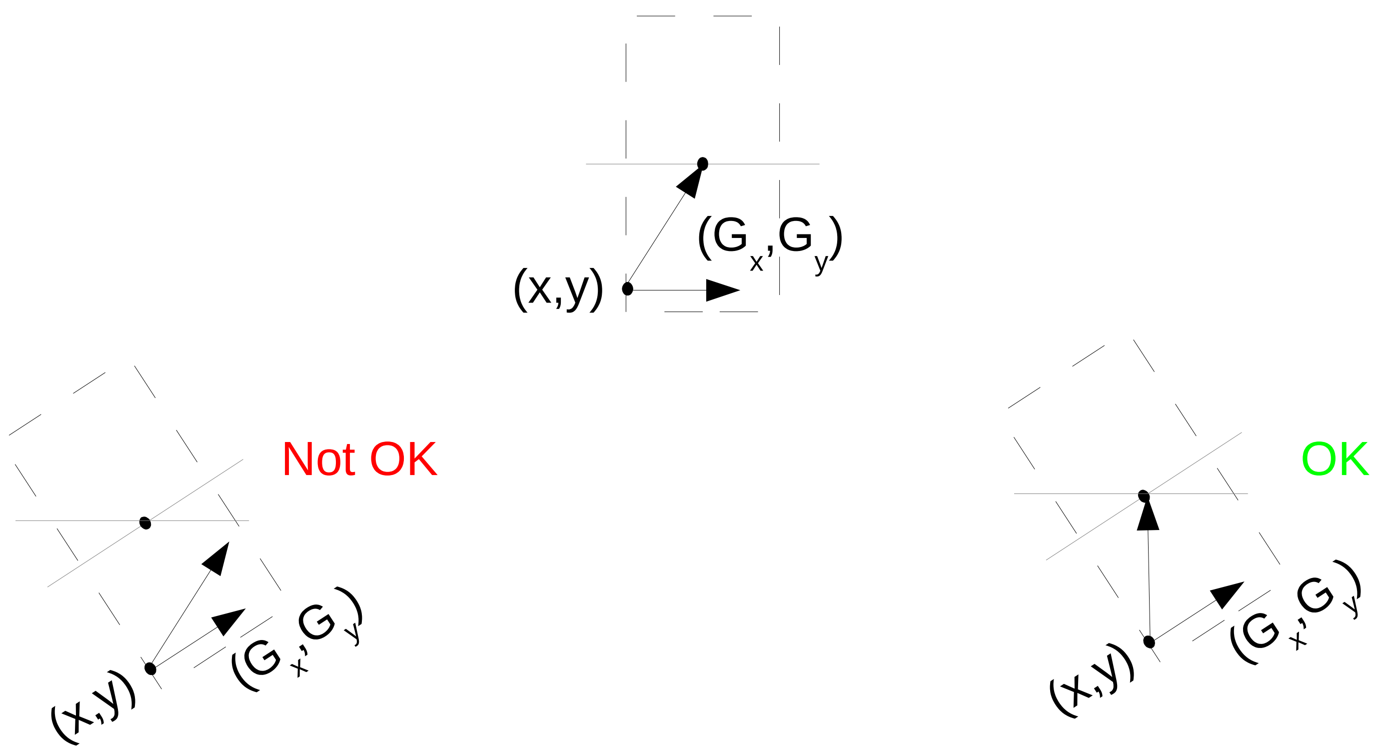
- The R-Table is to be modified to describe a rotated version of the original object
- The rotation angle is assumed to be a multiple of $\phi = (2\pi)/A$
  - ➔ Angular resolution in H_G is limited by the number of rows in the R-Table



- Gradients that used to be in row 1 are now associated with row 2
- The former last row becomes the new first row
  - ➔ Rotation causes a circular shift of the rows of the R-Table

Changing the R-Table: Orientation

- The direction to the object center stays constant with respect to the local gradient

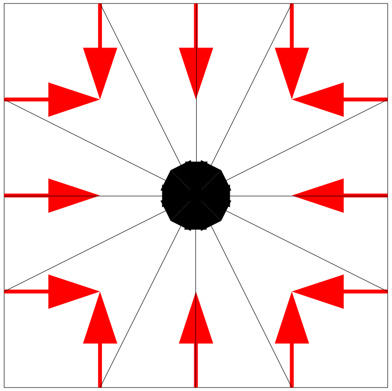


- The displacement vectors also rotate by angle ϕ:

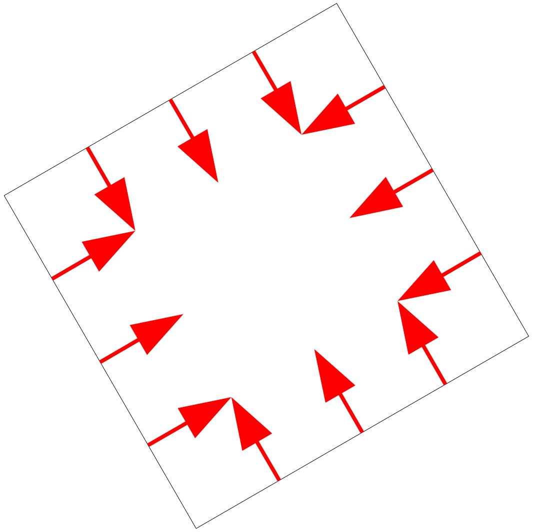
$$\begin{pmatrix} d_x^{a,b} \\ d_y^{a,b} \end{pmatrix} \leftarrow \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} d_x^{a,b} \\ d_y^{a,b} \end{pmatrix}$$

Changing the R-Table: Orientation

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre

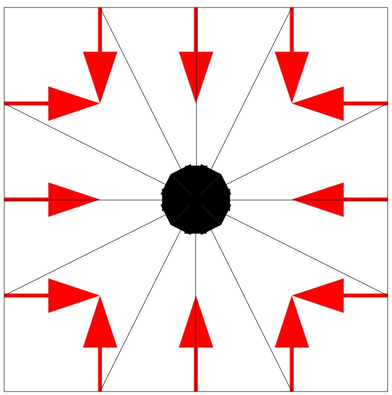


θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	



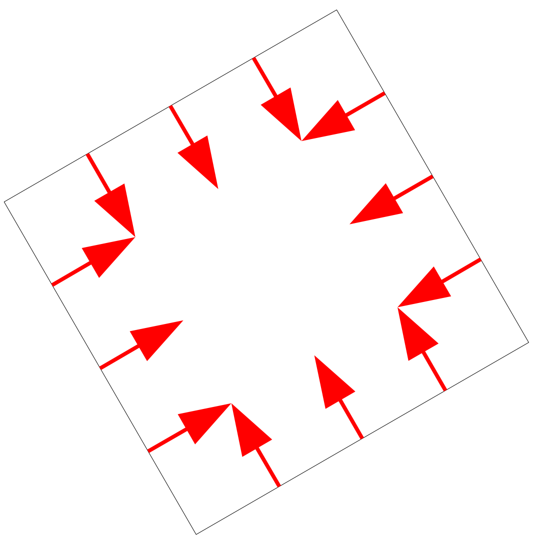
Changing the R-Table: Orientation

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



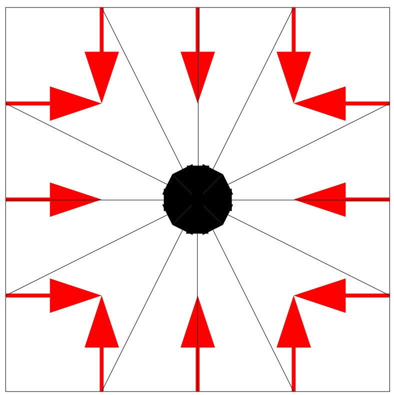
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	

$$\cdot\begin{pmatrix}\cos(30) & -\sin(30)\\ \sin(30) & \cos(30)\end{pmatrix}$$

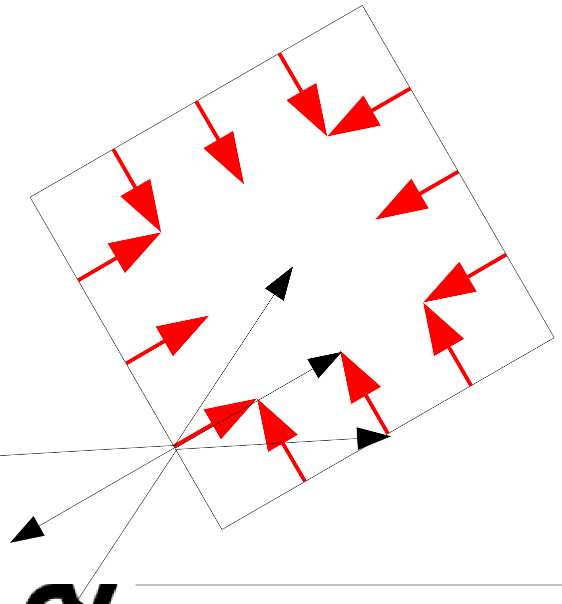


Changing the R-Table: Orientation

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



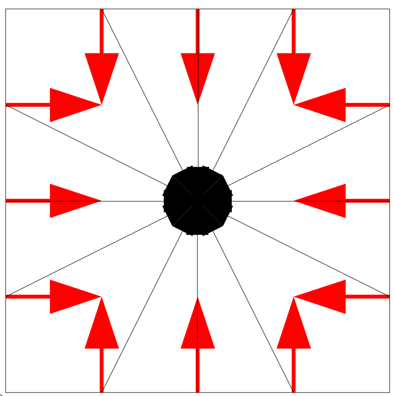
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	



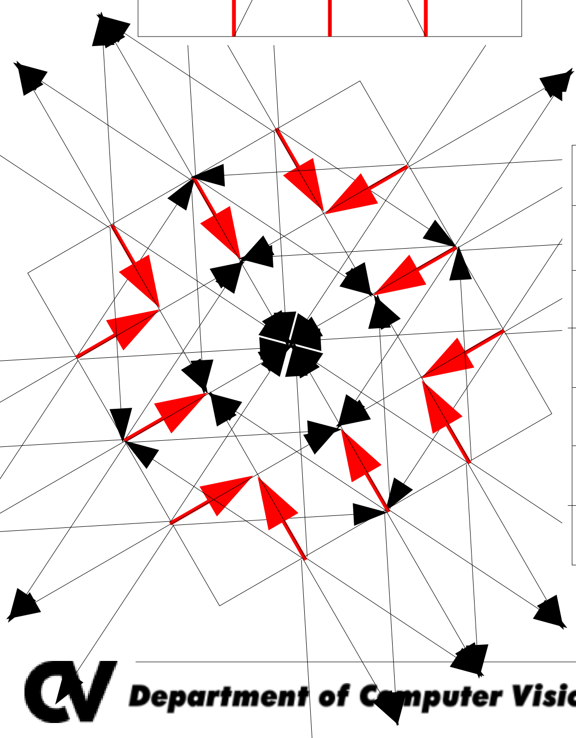
θ	
[0, 30)	
[30, 60)	$(1.2,1.8),(1.7,1),(2.2,0.1),(-2.2,-0.1),(-1.7,-1),(-1.2,-1.8)$
[60, 90)	
[90, 120)	
[120, 150)	$(-0.1,2.2),(-1,1.7),(-1.8,1.2),(1.8,-1.2),(1,-1.7),(0.1,-2.2)$
[150, 180)	

Changing the R-Table: Orientation

The R-Table stores, for each pixel on the edge of an object O, the local gradient direction and the displacement from the pixel to the object centre



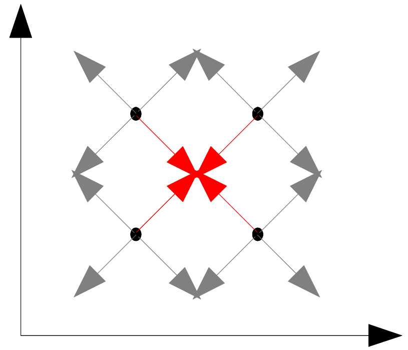
θ	
[0, 30)	$(2,1),(2,0),(2,-1),(-2,1),(-2,0),(-2,-1)$
[30, 60)	
[60, 90)	
[90, 120)	$(1,2),(0,2),(-1,2),(1,-2),(0,-2),(-1,-2)$
[120, 150)	
[150, 180)	



θ	
[0, 30)	
[30, 60)	$(1.2,1.8),(1.7,1),(2.2,0.1),(-2.2,-0.1),(-1.7,-1),(-1.2,-1.8)$
[60, 90)	
[90, 120)	
[120, 150)	$(-0.1,2.2),(-1,1.7),(-1.8,1.2),(1.8,-1.2),(1,-1.7),(0.1,-2.2)$
[150, 180)	

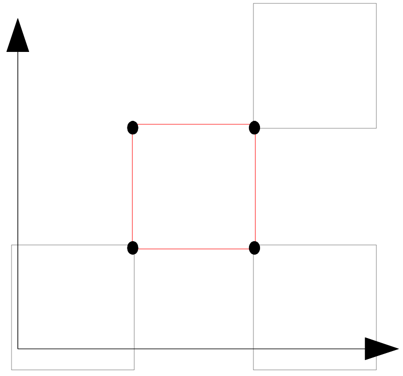
Alternative Formulation using Correlation

- Traditional: **R-Table**



- The R-Table approach loops over pixels:
  - ➔ Consider each local gradient direction
  - ➔ Use displacement vectors from table to accumulate „votes“ in H_G

- Alternative: **Direct Correlation**

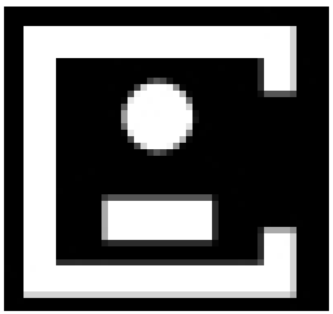


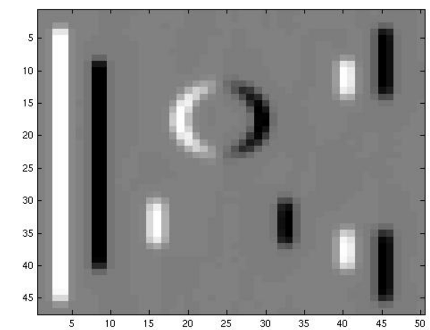
- Explicitly search for the object in the image
- Loop over parameters (s,θ)
  - ➔ Scale s of the object
  - ➔ Orientation θ of the object
  - ➔ Explicitly match the image with this version of the object
- The two formulations are (almost) identical

Preprocessing the Example Object

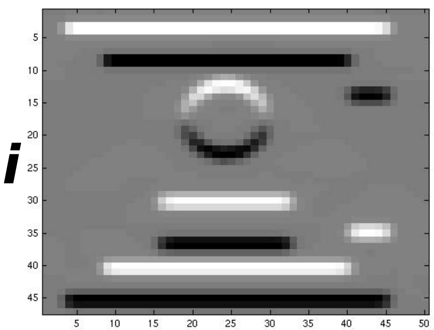
The Object of interest O(x,y) is processed to obtain two representations

1. Extraction of the complex gradients $O^I(x,y)$



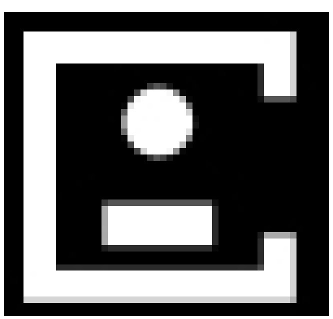


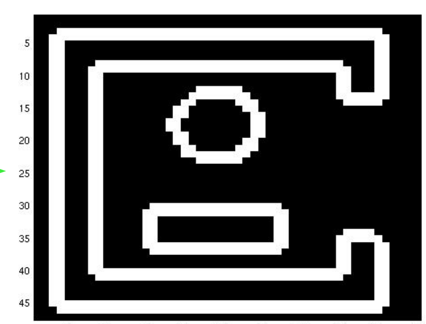
+ ***i***



$$O^I = \frac{\partial}{\partial x} O(x,y) + i \cdot \frac{\partial}{\partial y} O(x,y)$$

2. Obtain the binary edge image $O^B(x,y)$





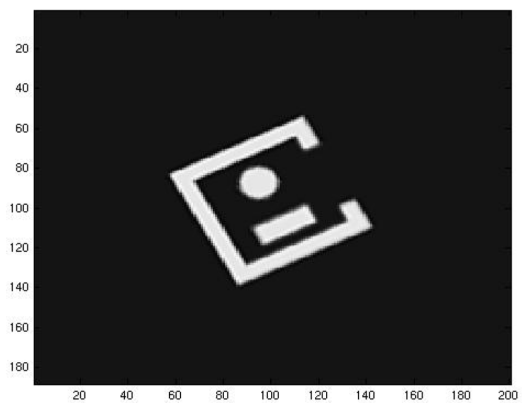
- Binary image indicating pixels on edges
  - ➔ Large local gradient magnitude
- Obtained using threshold T^B:

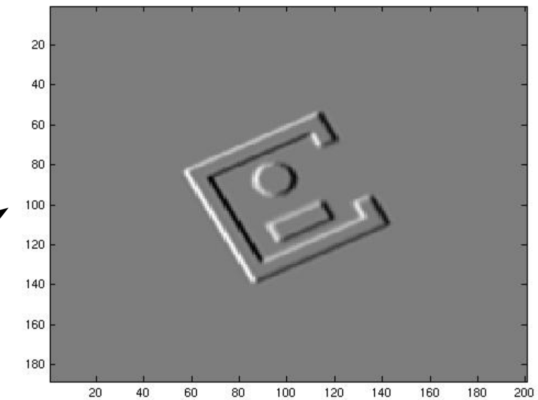
$$O^B(x,y) = |O^I(x,y)| > \left(T^B \cdot max_{x,y}\{|O^I(x,y)|\}\right)$$

Preprocessing the Image

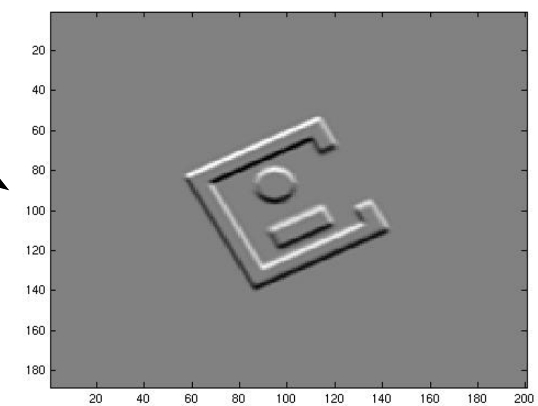
The image I(x,y) containing objects of interest is processed similarly
to give complex gradients $I^{I}(x,y)$

Image I(x,y)





$$\frac{\partial}{\partial x} I(x,y)$$

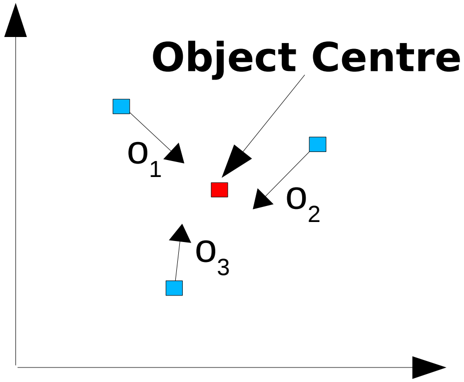


$$\frac{\partial}{\partial y} I(x,y)$$

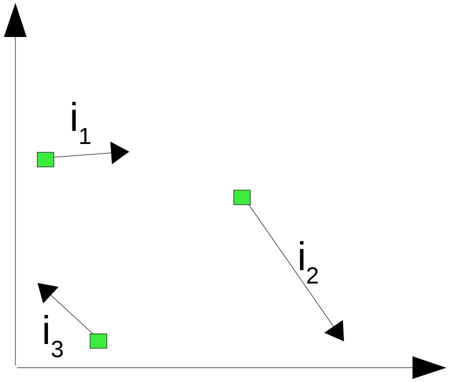
$$\boxed{I^{I} = \frac{\partial}{\partial x} I(x,y) + i \cdot \frac{\partial}{\partial y} I(x,y)}$$

Correlation

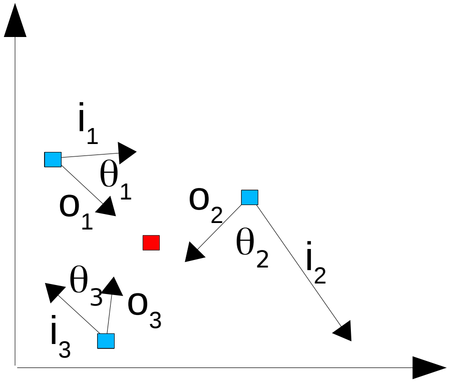
$$H_G(x,y,\theta=0,s=1)=\left|\Re\left\{\left(O^I(x,y)O^B(x,y)\right)\odot I^I(x,y)\right\}\right|$$



$O^I(x,y)O^B(x,y)$



$I^I(x,y)$



Correlation counts all „votes“ in each potential centre

- The correlation is high where gradient directions in O and I match

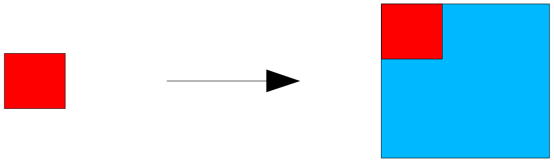
$$K=\left|\sum_{o_j}|o_j|\cdot|i_j|\cos(\theta_j)\right|$$

- The cosine term means that votes can also become negative
- The absolute value ensures that detection is possible irrespective of whether the object is brighter or darker than the background

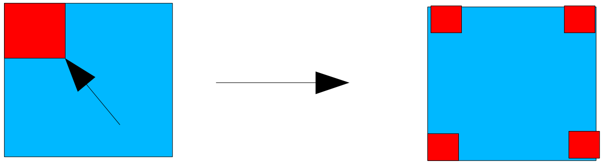
Correlation in the Fequency Domain

1) Filter (object mask) and gradient image
must have the same size before transformation

→ Copy the object mask into a larger matrix



2) Centre the filter (the centre is shifted to coordinates (0,0))



3) Transfer filter and image to the frequency domain

4) The spectrum of the result is the product of the image spectrum and the **complex conjugate** of the filter spectrum

Scale and Orientation

The correlation is repeated with scaled and rotated object masks to fill H_G

1) Changes in Scale

- Binary object mask and complex object gradients are resized

2) Rotation

- Binary object mask and complex object gradients are rotated

- This rotation does not affect the directions of complex gradients!
  - Complex gradients are rotated by applying a phase shift
  - After rotating by θ, complex gradients are multiplied with $\exp(-i\,\theta)$

3) Normalisation

- To retain scale invariance, the magnitudes of complex gradients are normalised:

$$O^I(x,y) \leftarrow \frac{O^I(x,y)}{\sum_x \sum_y |O^I(x,y)|}$$

Exercise 3

1) Theory part
2) Implementing the transformation
3) Detecting multiple objects in an image

Exercise 3 – Theory

The GHT provides the possibility to search for instances of the template object with different scale *s* and/or rotation *θ* by using a <u>4D voting space *H(x,y,s,θ)*</u>.

The table on the right shows the R-table, which was generated by analysing the template object. As reference point the top-left corner of the object was chosen. The angles are defined relative to the x-axis and measured counter-clockwise.

Orientation	Displacement
[0, 45)	(0,10); (0,20)
[45, 90)	
[90,135)	
[135,180)	(-20,10); (-10,20)
[180,225)	
[225,270)	
[270,315)	(-10,0); (-20,0)
[315,360)	

Task 1:
Which R-table below corresponds to scale (*s=1*) with a counter-clockwise rotation of *θ = 90°?*

i)

Orientation	Displacement
[0, 45)	(-10,0); (-20,0)
[45, 90)	
[90,135)	
[135,180)	(-10,20);(-20,-10)
[180,225)	
[225,270)	
[270,315)	(0,-10); (0,-20)
[315,360)	

ii)

Orientation	Displacement
[0, 45)	(0,-10); (0,-20)
[45, 90)	
[90,135)	(-10,0); (-20,0)
[135,180)	
[180,225)	
[225,270)	(-10,-20);(-20,-10)
[270,315)	
[315,360)	

iii)

Orientation	Displacement
[0, 45)	
[45, 90)	(-10,0); (-20,0)
[90,135)	
[135,180)	
[180,225)	(-10,-20);(-20,-10)
[225,270)	
[270,315)	
[315,360)	(0,-10); (0,-20)

iv)

Orientation	Displacement
[0, 45)	
[45, 90)	(-20,10); (-10,20
[90,135)	
[135,180)	
[180,225)	(-10,0); (-20,0)
[225,270)	
[270,315)	(0,10); (0,20)
[315,360)	

Exercise 3 – Theory

The GHT provides the possibility to search for instances of the template object with different scale *s* and/or rotation *θ* by using a 4D voting space *H(x,y,s,θ)*.

The table on the right shows the R-table, which was generated by analysing the template object. As reference point the top-left corner of the object was chosen. The angles are defined relative to the x-axis and measured counter-clockwise.

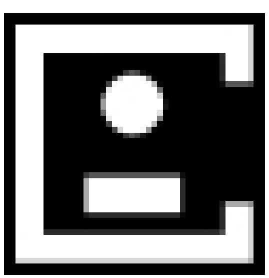
Orientation	Displacement
[0, 45)	(0,10); (0,20)
[45, 90)	
[90,135)	
[135,180)	(-20,10); (-10,20)
[180,225)	
[225,270)	
[270,315)	(-10,0); (-20,0)
[315,360)	

Task 2:
Which simple geometric object does this R-table most likely represent?

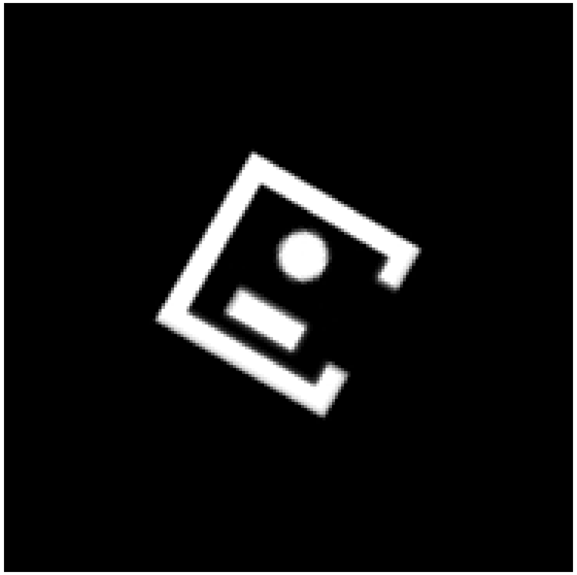
i) A triangle ii) A circle iii) A square iv) A parallelogram

Given Functions

- `main (Usage: aia3 <path to template image> [<path to test image>])`
  - ➔ Test routine for the generalised Hough transformation (main function)
  - ➔ Two different modes depending on number of input parameters:
    - ➔ 1st : Only template image is specified (one parameter)
      - Rotated and scaled version of template image is generated
      - Hough transformation is used to find template within this image
    - ➔ 2nd : Template as well as test image is specified (two parameters)
      - Hough transformation is used to find template within given image



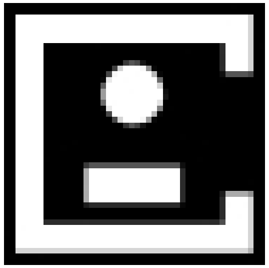
Object of Interest



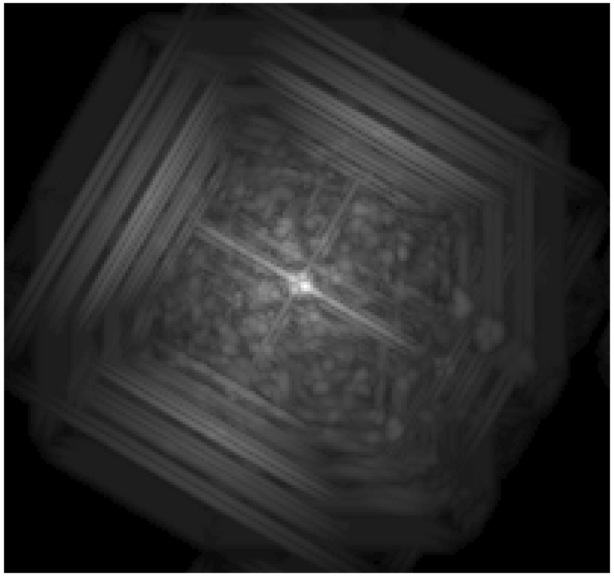
Image

Given Functions

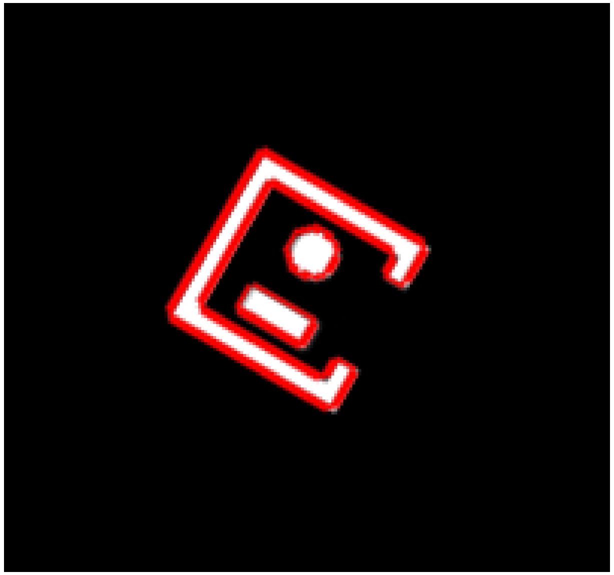
- **main (Usage: aia3 <path to template image> [<path to test image>])**
  - Test routine for the generalised Hough transformation (main function)
  - Two different modes depending on number of input parameters:
    - 1st : Only template image is specified (one parameter)
      - Rotated and scaled version of template image is generated
      - Hough transformation is used to find template within this image
    - 2nd : Template as well as test image is specified (two parameters)
      - Hough transformation is used to find template within given image



Object of Interest



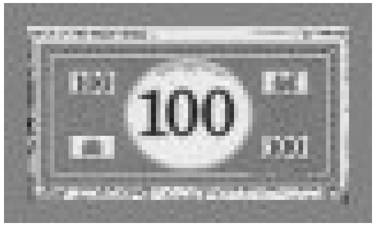
Hough Space



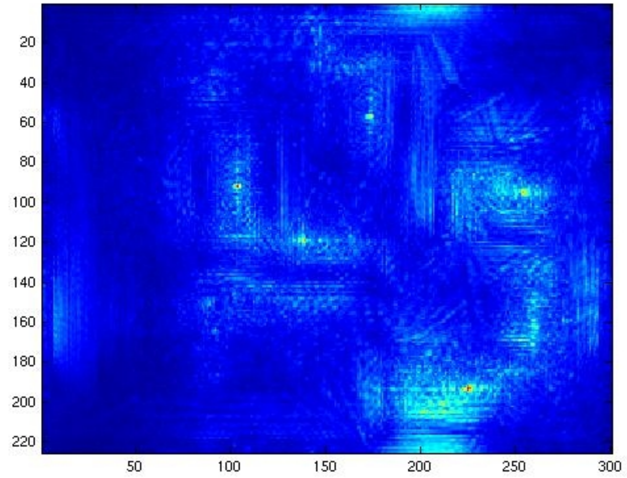
Detection

Given Functions

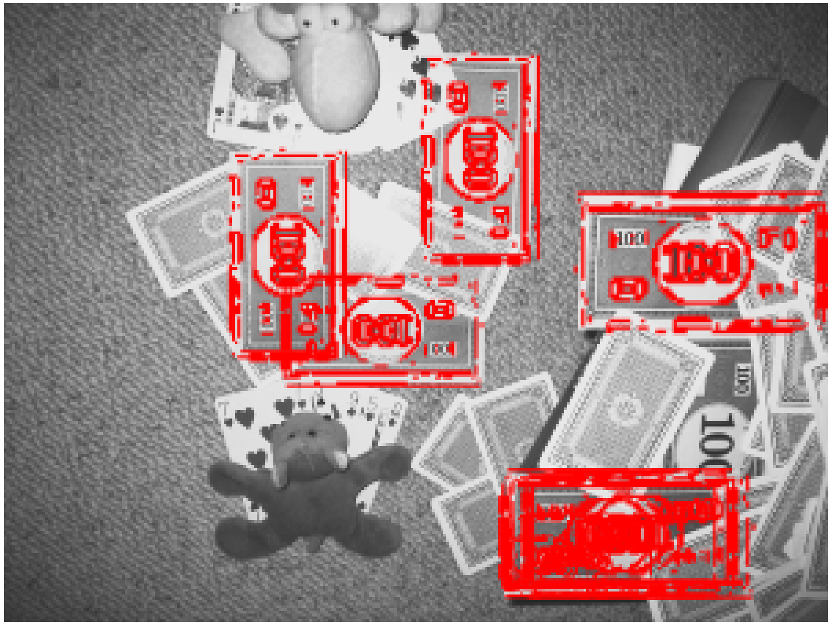
- **main (Usage: aia3 <path to template image> [<path to test image>])**
  - Test routine for the generalised Hough transformation (main function)
  - Two different modes depending on number of input parameters:
    - 1st : Only template image is specified (one parameter)
      - Rotated and scaled version of template image is generated
      - Hough transformation is used to find template within this image
    - 2nd : Template as well as test image is specified (two parameters)
      - Hough transformation is used to find template within given image



Object of Interest



Hough Space



Image

Given Functions

- `main (Usage: aia3 <path to template image> [<path to test image>])`
  - Test routine for the generalised Hough transformation (main function)
  - Two different modes depending on number of input parameters:
    - 1st : Only template image is specified (one parameter)
      - Rotated and scaled version of template image is generated
      - Hough transformation is used to find template within this image
    - 2nd : Template as well as test image is specified (two parameters)
      - Hough transformation is used to find template within given image

If an image contains more than a single object, a suitable thresholding procedure is needed to single out individual detections

1) Global Threshold

- Assume that all objects have received a vote in H_G that exceeds T^O * max(H_G)
- T^O is a fixed threshold

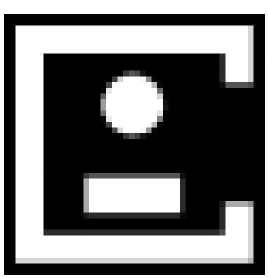
2) Identify Local Maxima

- Where an entire region of H_G received a high vote (>threshold), one should only consider local maxima of H_G as detection events.
- After applying the threshold, local maxima are considered the set of detected objects

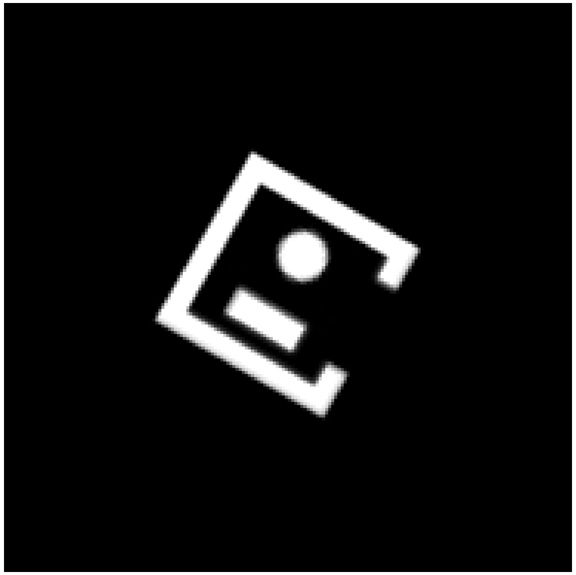
Given Functions

- **`Mat makeTestImage(Mat& temp, double angle, double scale, double* scaleRange)`**
  - `temp`: Grayscale image
  - `angle`: Rotation angle
  - `scale`: Scaling factor
  - `scaleRange`: Scale range

  ➔ Generates a rotated and scaled version of the input image



Object of Interest



Image

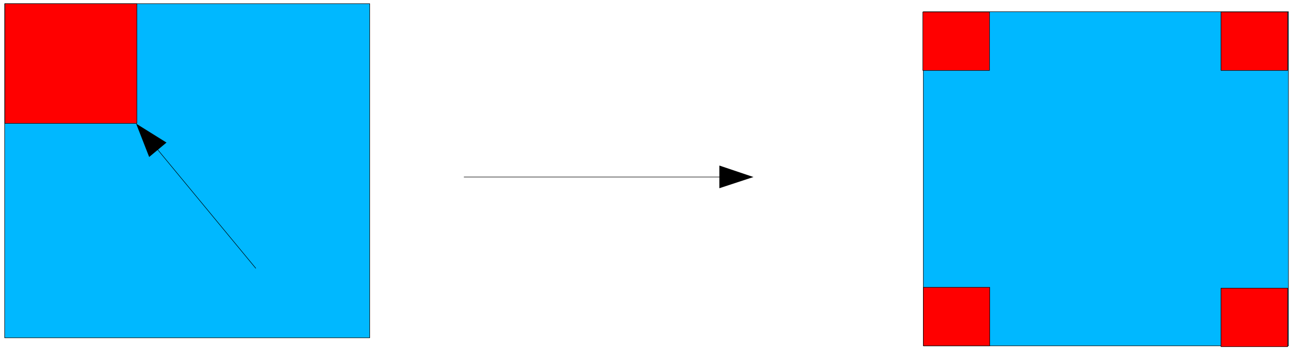
Given Functions

- `Mat rotateAndScale(Mat& temp, double angle, double scale)`
  - `temp`: Grayscale image
  - `angle`: Rotation angle in radian
  - `scale`: Scale factor
  - `return`: Transformed image

  ➔ Scales and rotates given image

- `void circShift(Mat& in, Mat& out, int dx, int dy)`
  - `in`: Image
  - `out`: Shifted image
  - `dx,dy` Shift in x- and y-direction

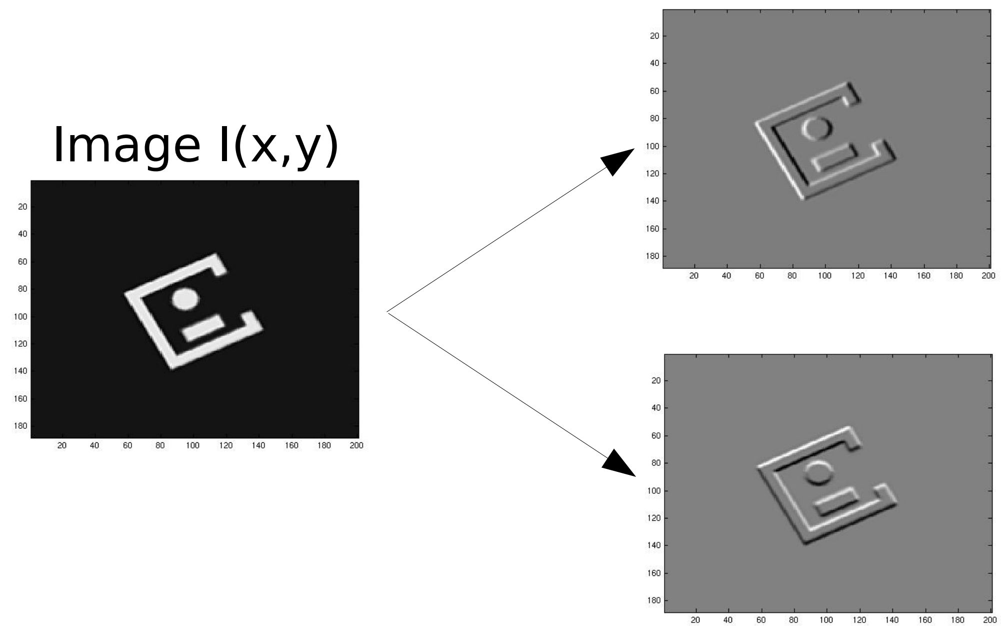
  ➔ Performs circular shift



Given Functions

- `Mat calcDirectionalGrad(Mat& testImage, double sigma)`
  - `image`: Grayscale image
  - `sigma`: Standard deviation of filter (not necessarily integer)
  - `return`: Complex gradient image (two channel image)

➔ Computes the complex gradients in an image
➔ 1st `channel`: gradients in x-direction
➔ 2nd `channel`: gradients in y-direction



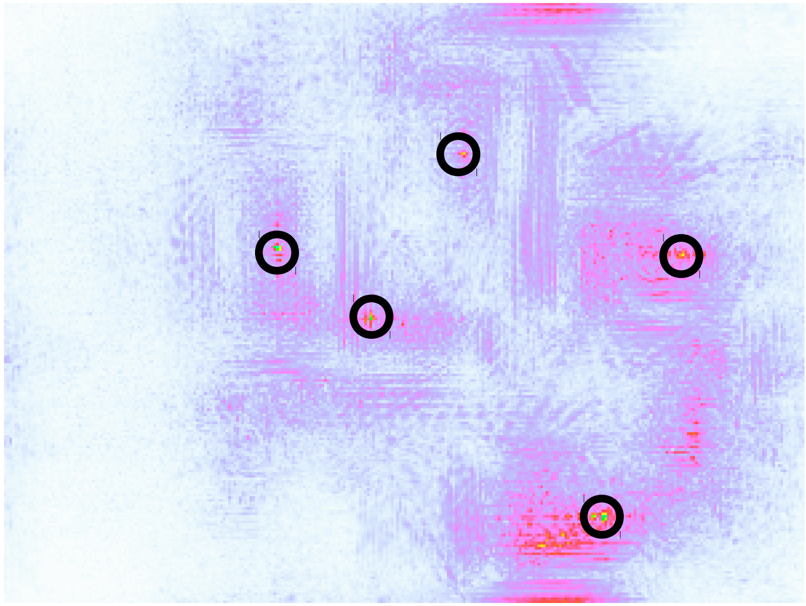
$$\frac{\partial}{\partial x} I(x,y)$$

$$\frac{\partial}{\partial y} I(x,y)$$

Given Functions

- `void findHoughMaxima(vector< vector<Mat> >& houghSpace, double objThresh, vector<Scalar>& objList)`
  - `houghSpace`: The four dimensional Hough space H_G
  - `objThresh`: Global threshold for object detection.
    Threshold is relative to the maximum of H_G
  - `objList`: List of hough space coordinates (scale, angle, x, y)

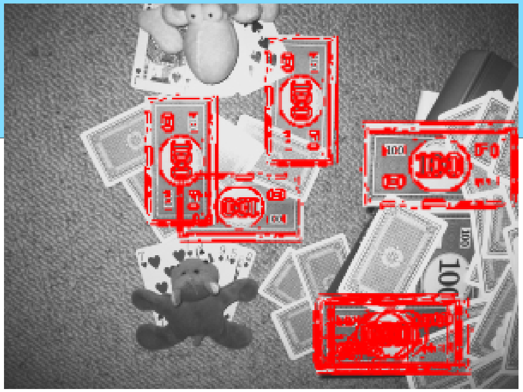
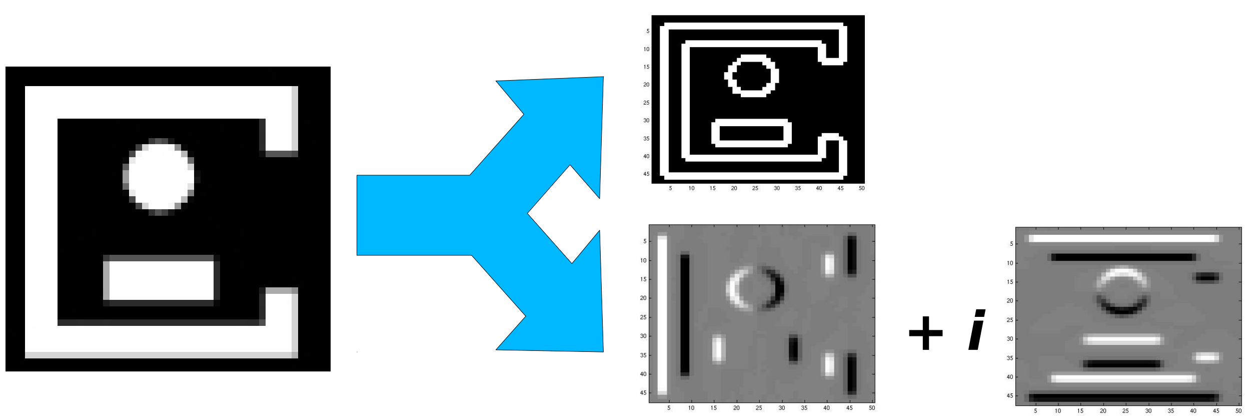
→ Applies a global threshold and identifies remaining local maxima in H_G



Given Functions

- ```
  void plotHoughDetectionResult(Mat& testImage, vector<Mat>& templ,
  vector<Scalar>& objList, double scaleSteps, double* scaleRange,
  double angleSteps, double* angleRange)
  ```
  - `testImage`: Grayscale image
  - `templ`: Object mask (see `makeObjectTemplate(..)`)
  - `objList`: Indices (into the Hough space H_G) of the detected objects. Each entry specifies a set of coordinates: (scale, angle, x, y)
  - `scaleSteps`: Resolution of the Hough space H_G in the scale dimension
  - `scaleRange`: Minimum and maximum scales (e.g. `[0.5, 2.0]`)
  - `angleSteps`: Resolution of Hough space H_G in the orientation dimension
  - `angleRange`: Minimum and maximum angle(typically `[0, 2*pi]`)

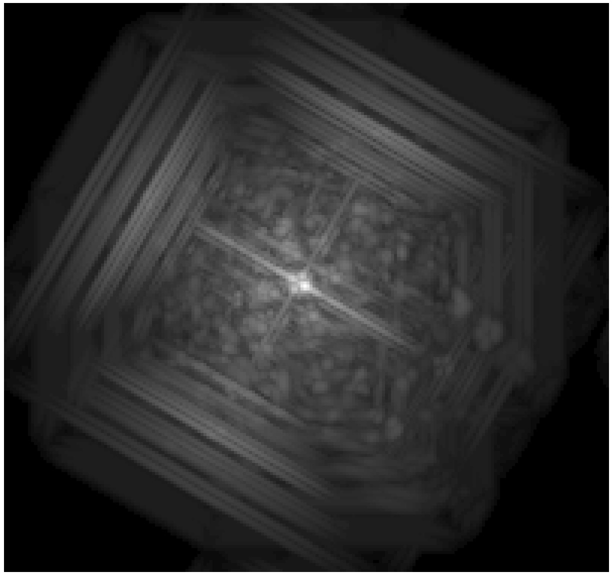
→ Visualises the results of object detection.

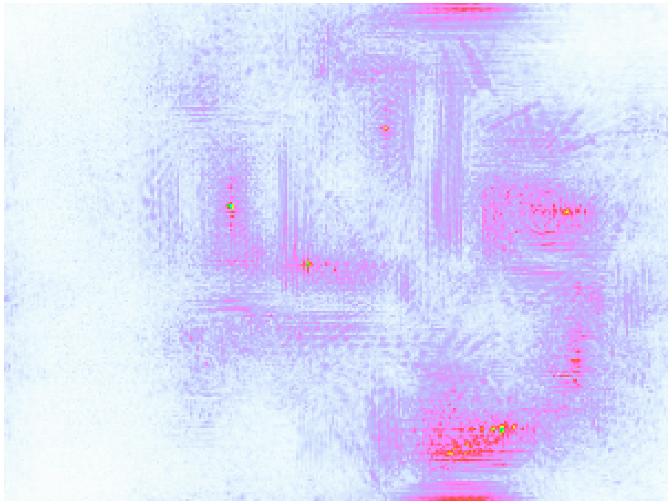

  ```

To Do

- ```
  vector<Mat> makeObjectTemplate(Mat& templateImage,
                                 double sigma, double templateThresh)
  ```
  - `templateImage`: Grayscale image with the object of interest
  - `sigma`: Std. deviation for the computation of image gradients
  - `templateThresh`: Threshold used to obtain the binary edge image (relative to the maximum gradient magnitude)
  - `return`: 2-element Mat-vector:
    `[0]` contains the binary edge mask
    `[1]` contains the corresponding complex gradients

➔ Creates a template of the object of interest, consisting of a binary edge map and a complex gradient image.


  ```

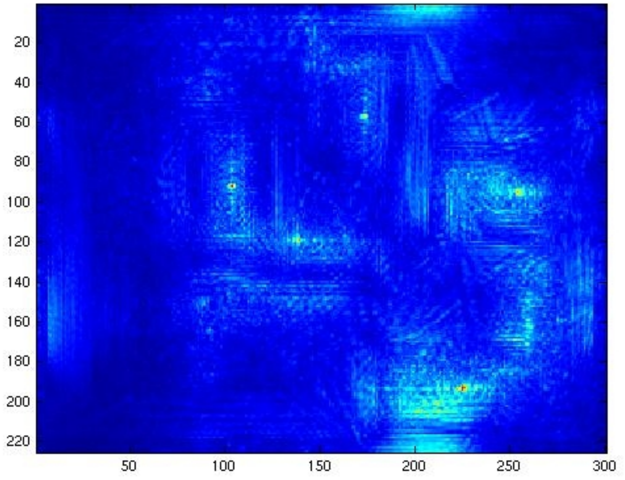
To Do

- **`void plotHough(vector< vector<Mat> >& houghSpace)`**
  - `houghSpace`: Computed 4D hough space

➔ Creates a single image from the 4D hough space
➔ Displays and saves it as image file







To Do

- `void makeFFTObjectMask(vector<Mat>& templ, double scale, double angle, Mat& fftMask)`
  - `templ`: Template for the object of interest (edge mask and gradients)
  - `scale`: New scale factor for the object
  - `angle`: New orientation for the object
  - `fftMask`: Spectrum of the mask that is correlated with a complex gradient image to find the object of interest, scaled by factor `scale` and rotated by `angle` radians.

- ➔ Computes the spectrum of $O^I O^B$ for correlation with the complex gradient image.
  - ➔ Note: $O^I O^B$ means component wise multiplication
- ➔ Scales and rotates the object template (updating the phase of gradients included), followed by normalisation, centering and Fourier transformation

To Do

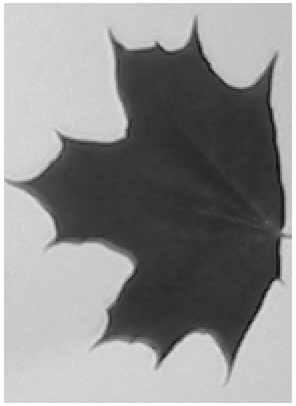
- `vector< vector<Mat> > generalHough(Mat& gradImage, vector<Mat>& templ, double scaleSteps, double* scaleRange, double angleSteps, double* angleRange)`
  - `gradIamge`: Complex gradient image (with objects to be detected)
  - `templ`: Template of an object of interest
  - `scaleSteps`: Resolution of the Hough space in the scale dimension
  - `scaleRange`: Minimal and maximal scale factor (e.g. `[0.5,2.0]`)
  - `angleSteps`: Resolution of the Hough space in the orientation dimension
  - `angleRange`: Smallest and largest orientation (typically `[0,2*pi)`)
  - `return`: The computed Hough space H_G.
    Four “dimensions”: scale, orientation, spatial dimensions

➔ Computes the voting space of the generalised Hough transform

➔ **Be careful to note the sequence of dimensions!**

➔ Useful functions: cv::mulSpectrum(), cv::dft()

Optional



- Apply GHT to other detection / recogntion tasks
→ eg. leaf detection & recognition task from last exercise

How does performance change? On what does it depend?