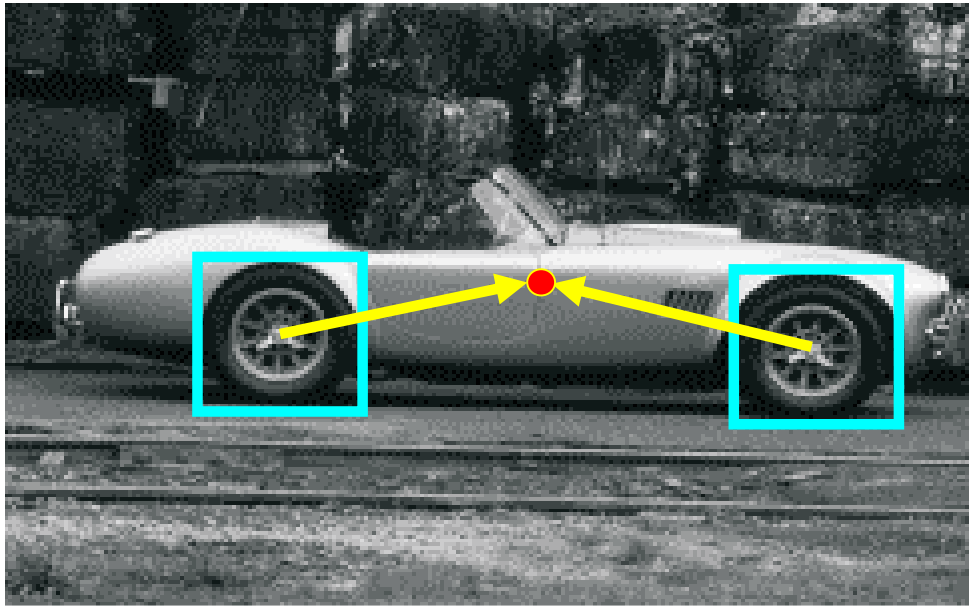
Generalized HOUGH Transform



Friday, 22/03/2024

Antonis Argyros

e-mail: argyros@csd.uoc.gr

Image and parameter space

variables

$$y = mx + b$$

parameters

$$x\cos\theta + y\sin\theta = \rho$$

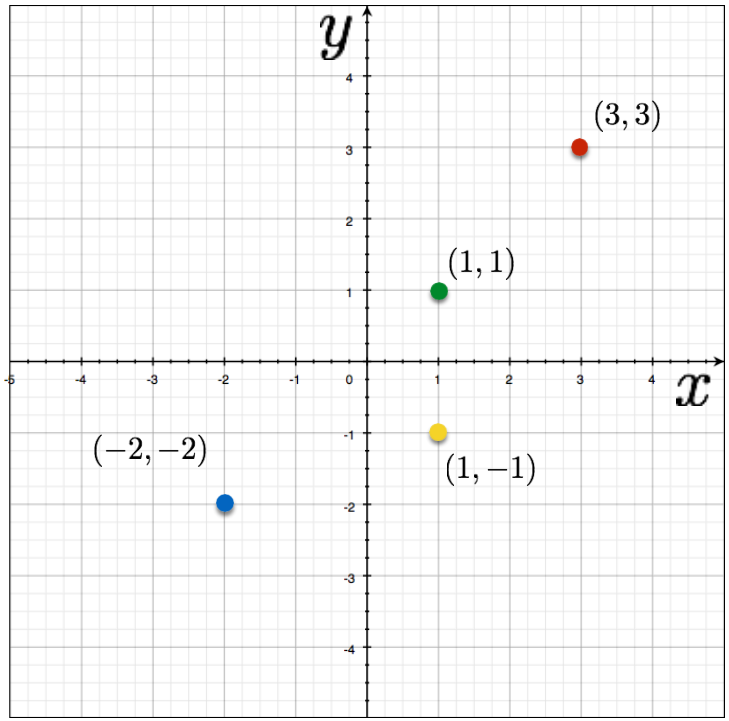
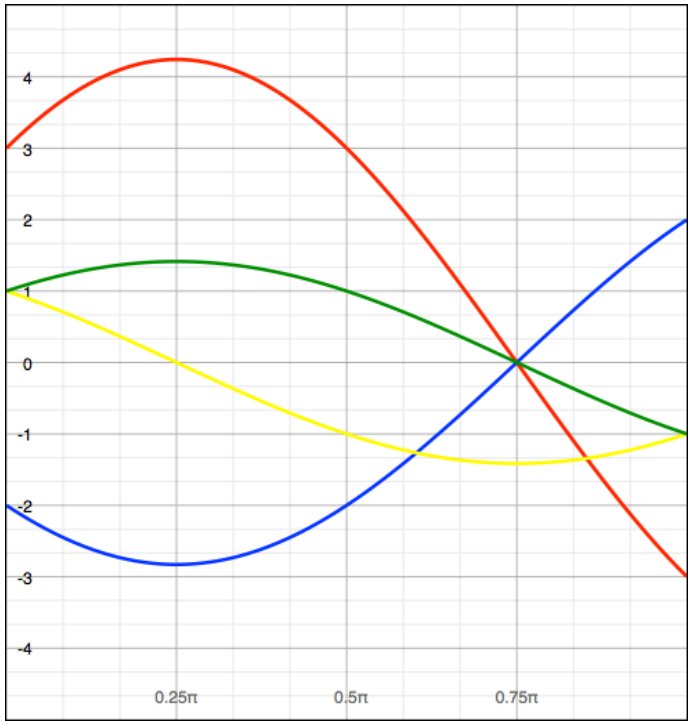


Image space



Parameter space θ

Hough Transform

1. Initialize accumulator H to all zeros

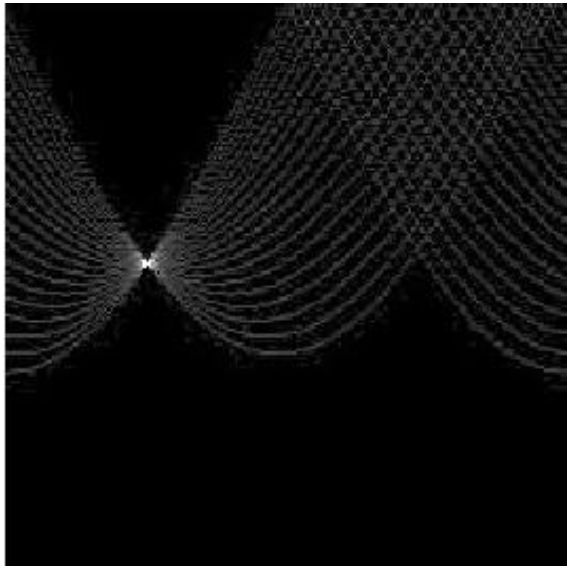
2. For each edge point (x,y) in the image
   For θ = 0 to 180
   $\rho = x \cos \theta + y \sin \theta$
   $H(\theta, \rho) = H(\theta, \rho) + 1$
   end
   end

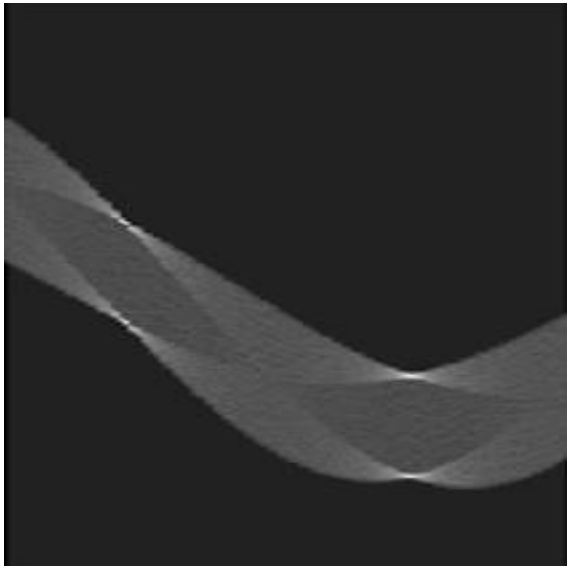
3. Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum

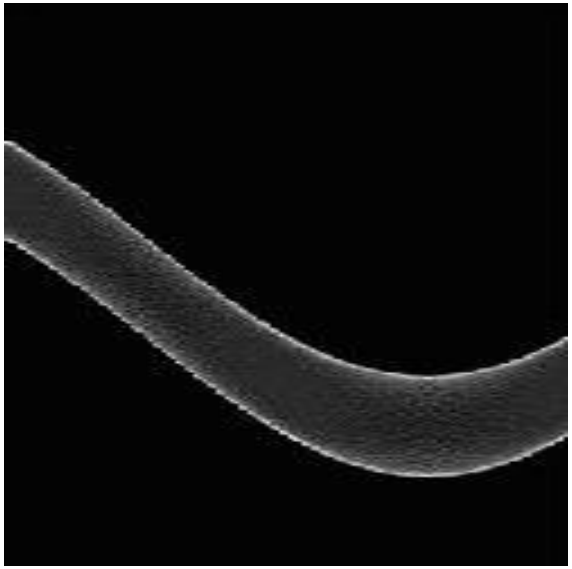
4. The detected line in the image is given by
   $\rho = x \cos \theta + y \sin \theta$

Basic shapes

(in parameter space)



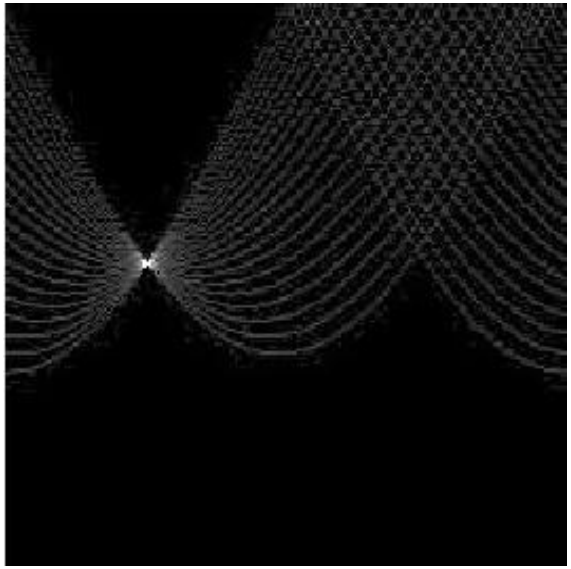




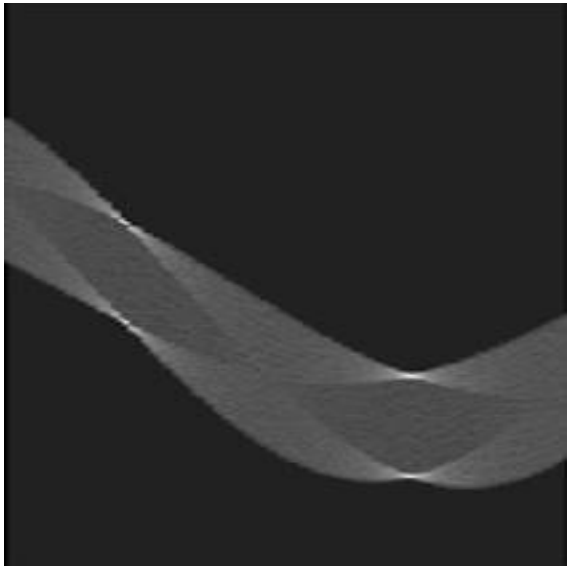
line

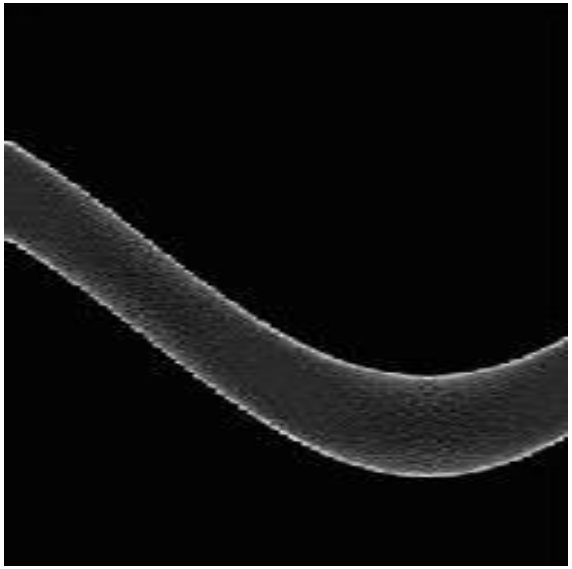
Basic shapes

(in parameter space)



line

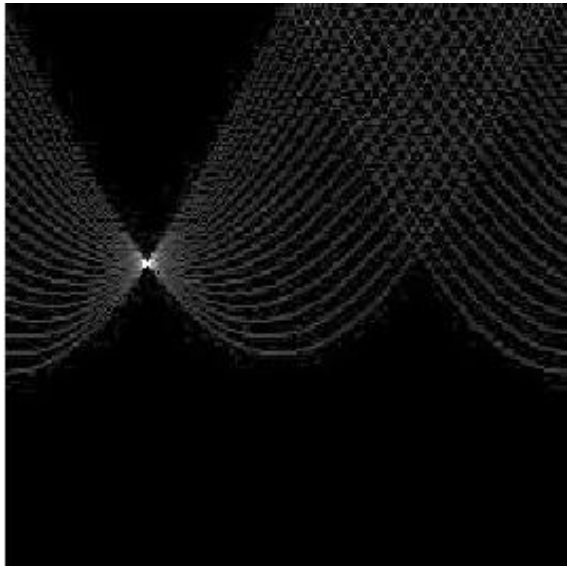




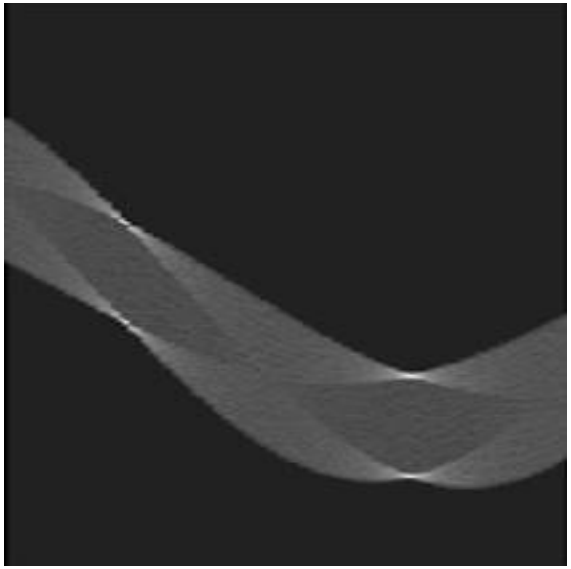
rectangle

Basic shapes

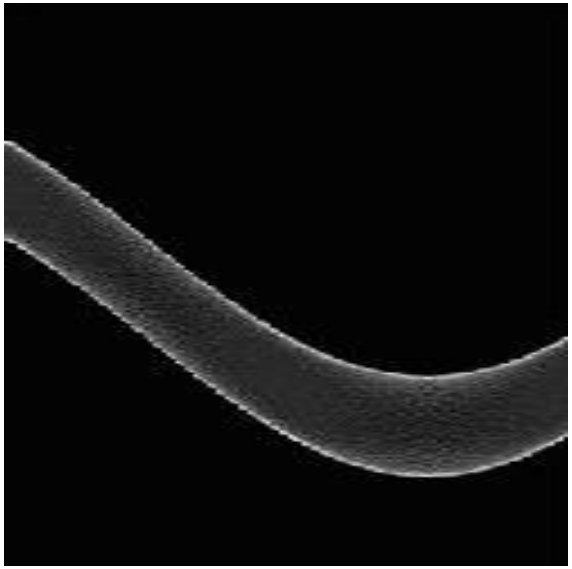
(in parameter space)



line



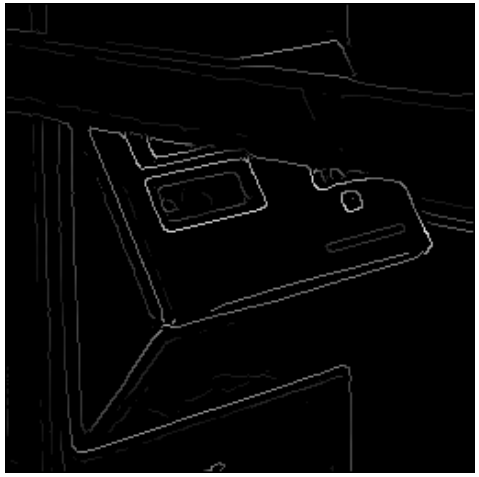
rectangle



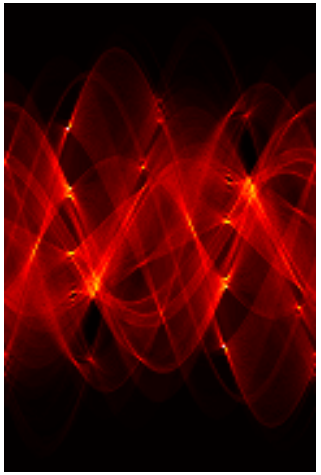
circle

Real-world example

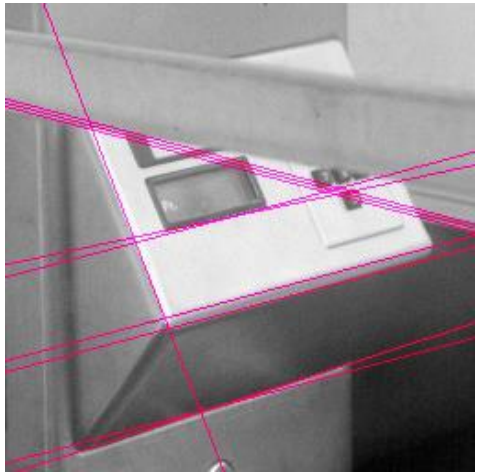
Original



Edges

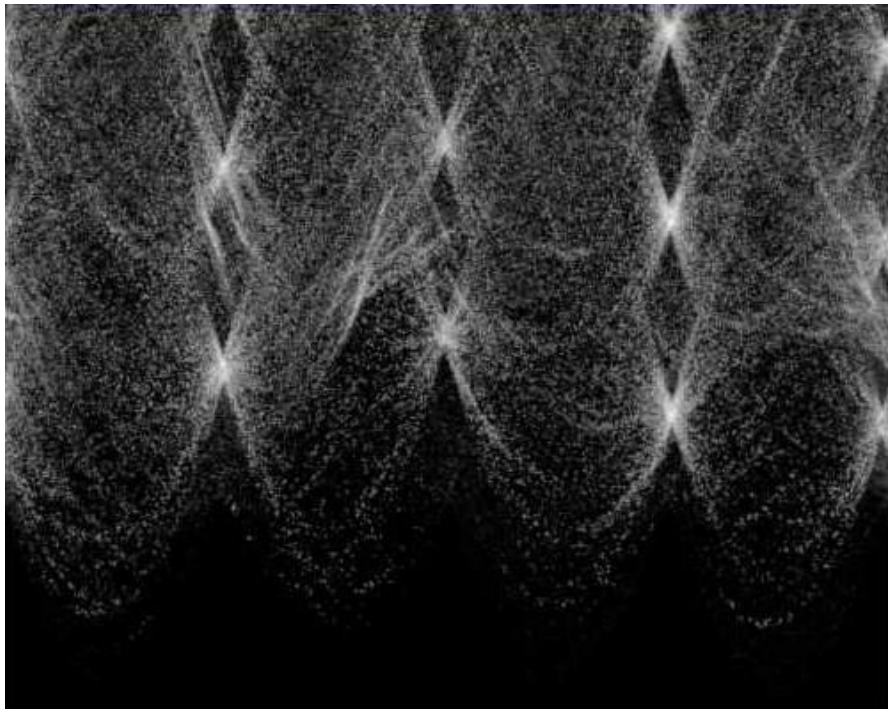


parameter space



Hough Lines

More complex image



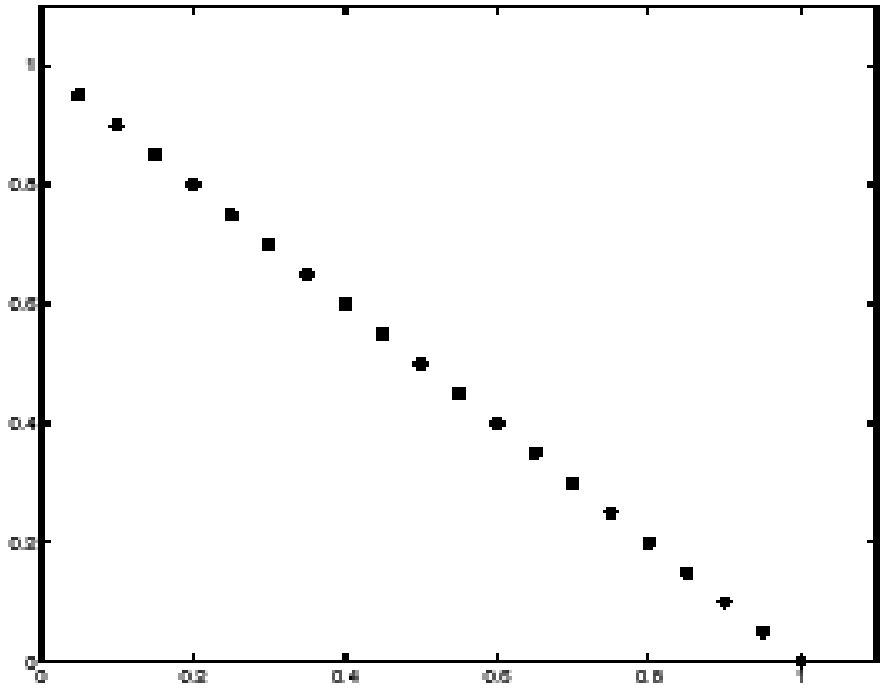
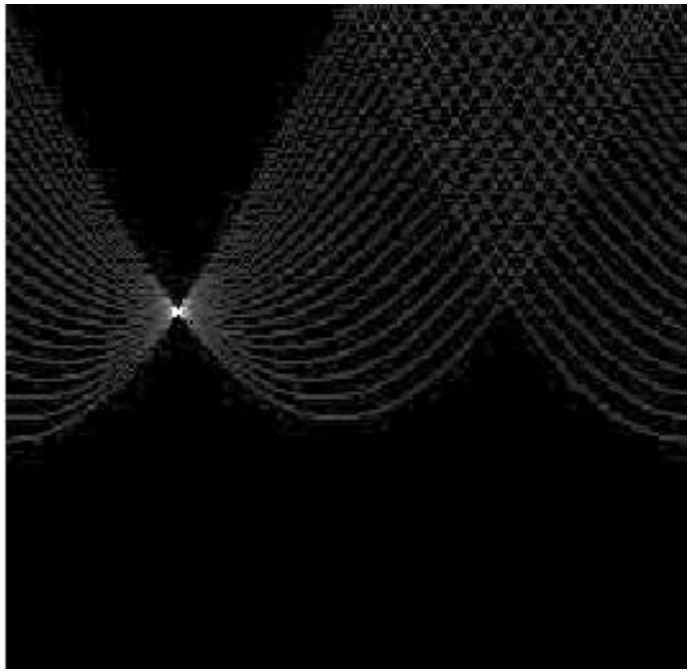


Image space



Votes

In practice, measurements are noisy…

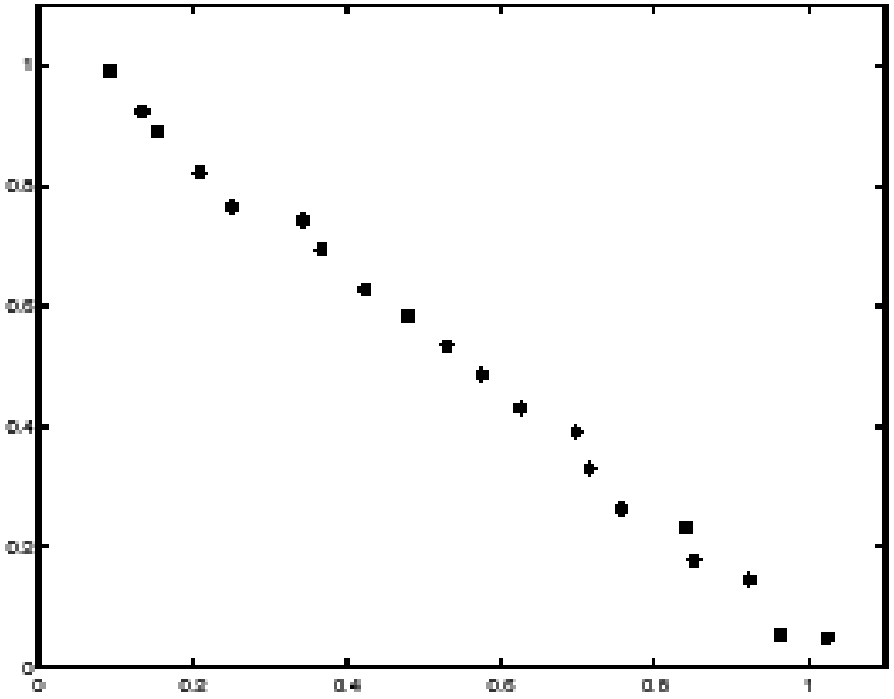
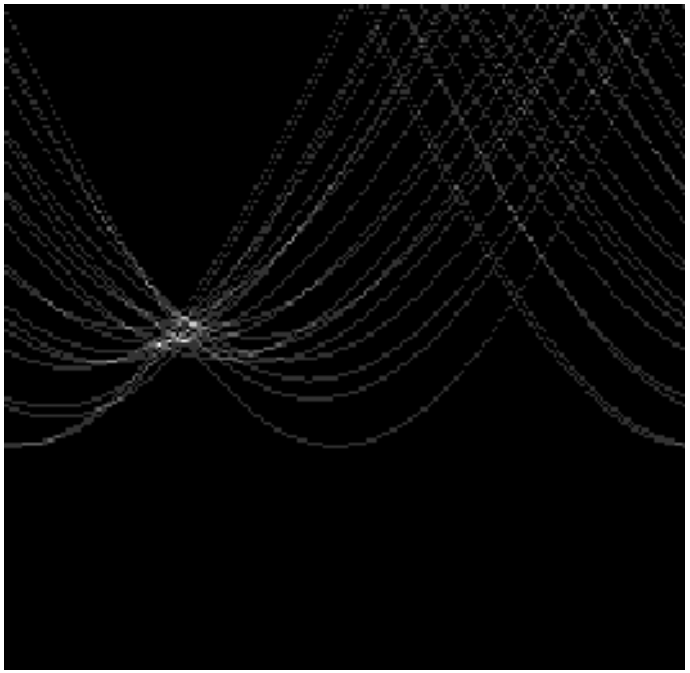


Image space



Votes

Too much noise …

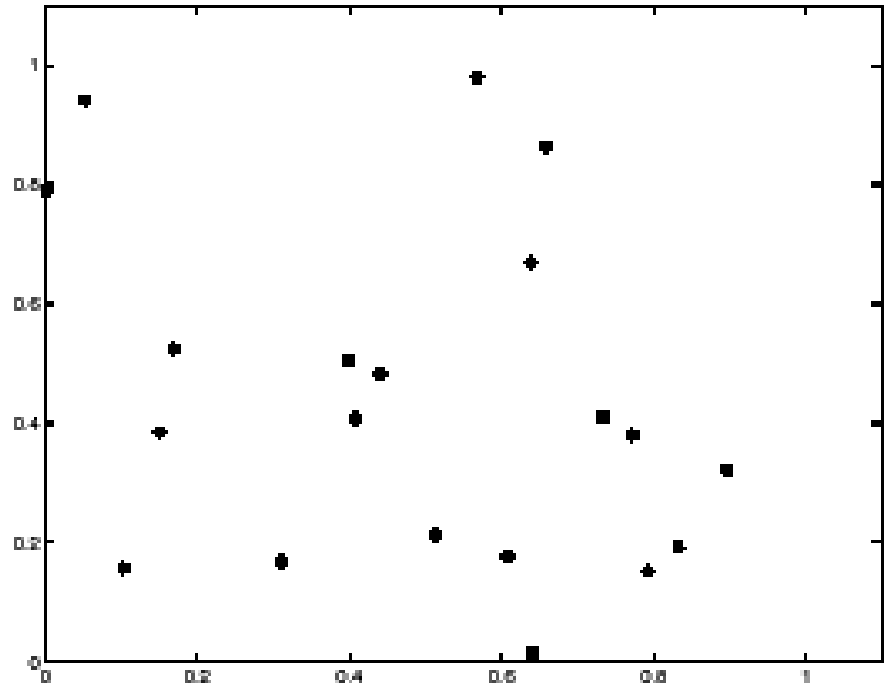
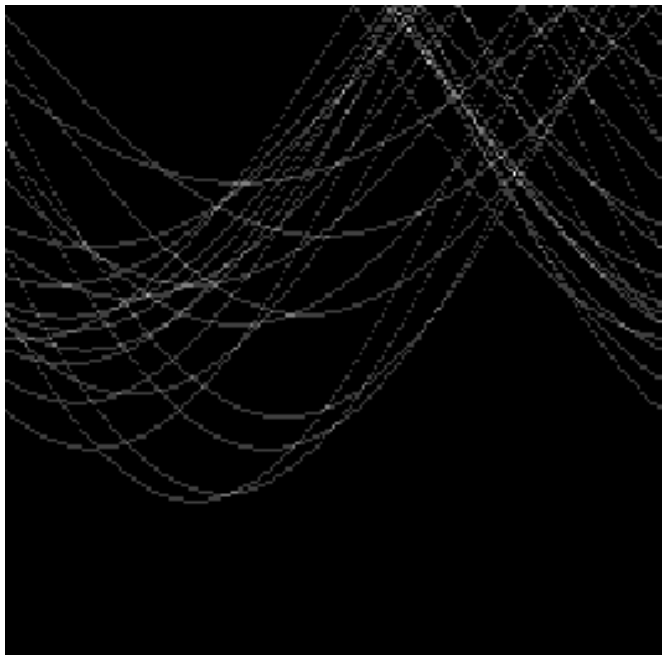


Image space



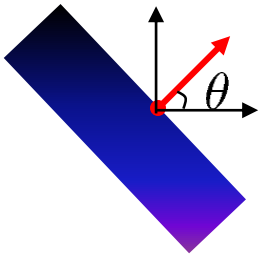
Votes

Dealing with noise

- Choose a good grid / discretization
  - **Too coarse:** large votes obtained when too many different lines correspond to a single bucket
  - **Too fine:** miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Try to get rid of irrelevant features
  - E.g., if applied to **edge points**, take only those with significant gradient magnitude

Hough transform for detecting lines in image edges: Incorporating image gradients

- Recall: when we detect an edge point, we also know its gradient direction
- But this means that the line is uniquely determined!



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Modified Hough transform:

For each edge point (x,y)

θ = π/2 + gradient orientation at (x,y)

ρ = x cos θ + y sin θ

H(θ, ρ) = H(θ, ρ) + 1

end

Hough for detecting circles

Let's assume radius known

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

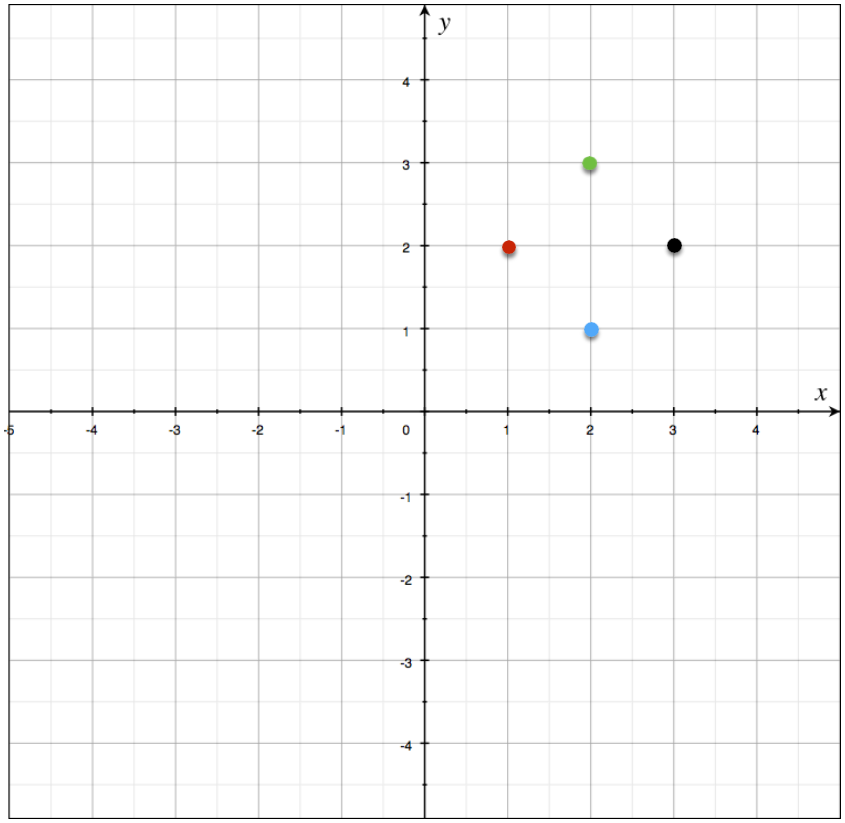
What is the parametric space?

How are the image points expected to vote in the parametric space?

parameters

$$(x-a)^2+(y-b)^2=r^2$$

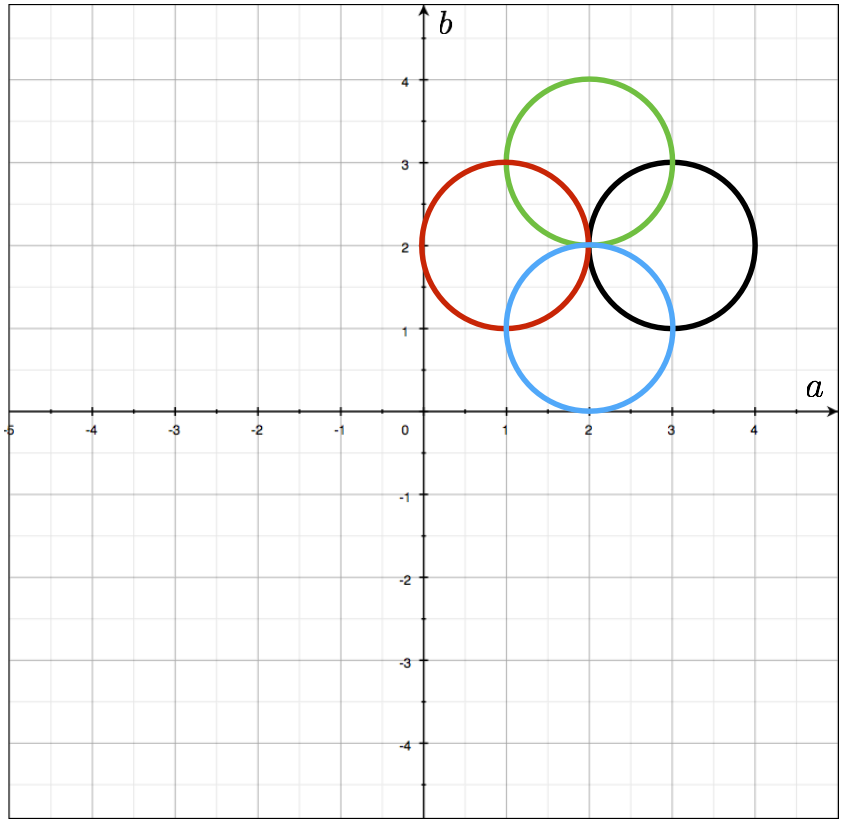
variables



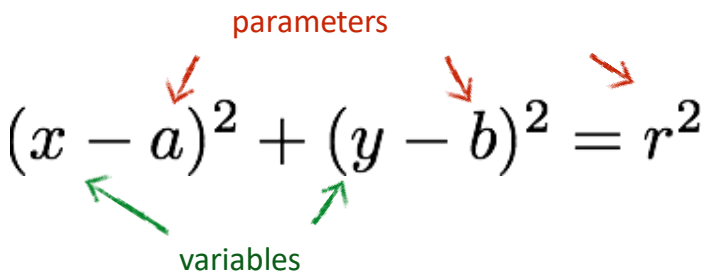
parameters

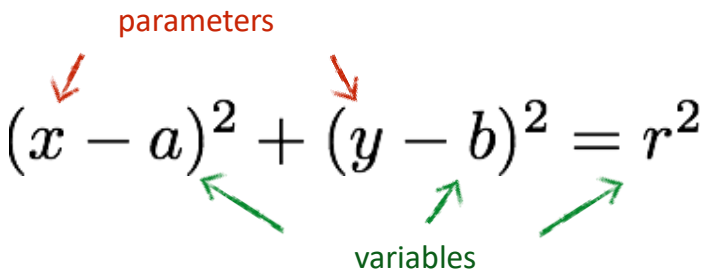
$$(x-a)^2+(y-b)^2=r^2$$

variables



What if radius is unknown?





If radius is not known: 3D Hough Space!

Use Accumulator array $A(a,b,r)$

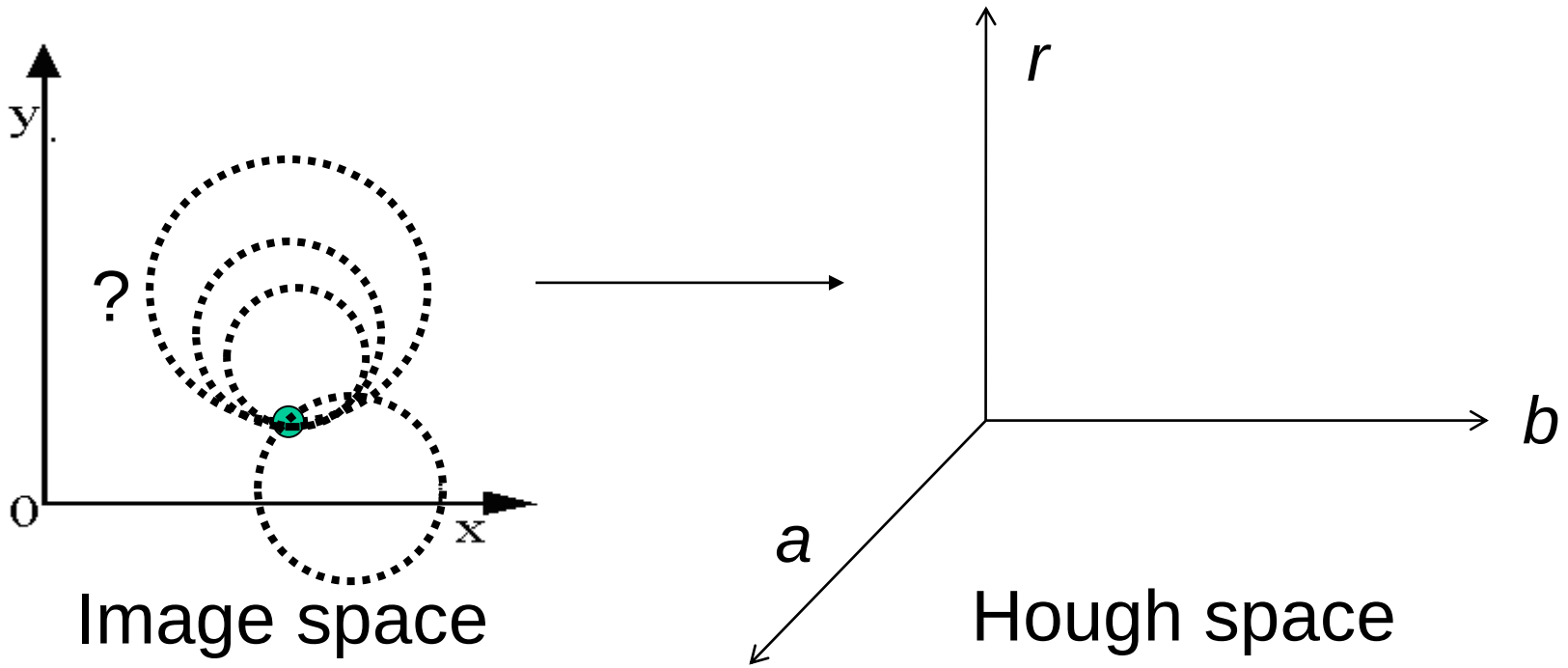
Surface shape in Hough space?

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

For an unknown radius r



Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

For an unknown radius r

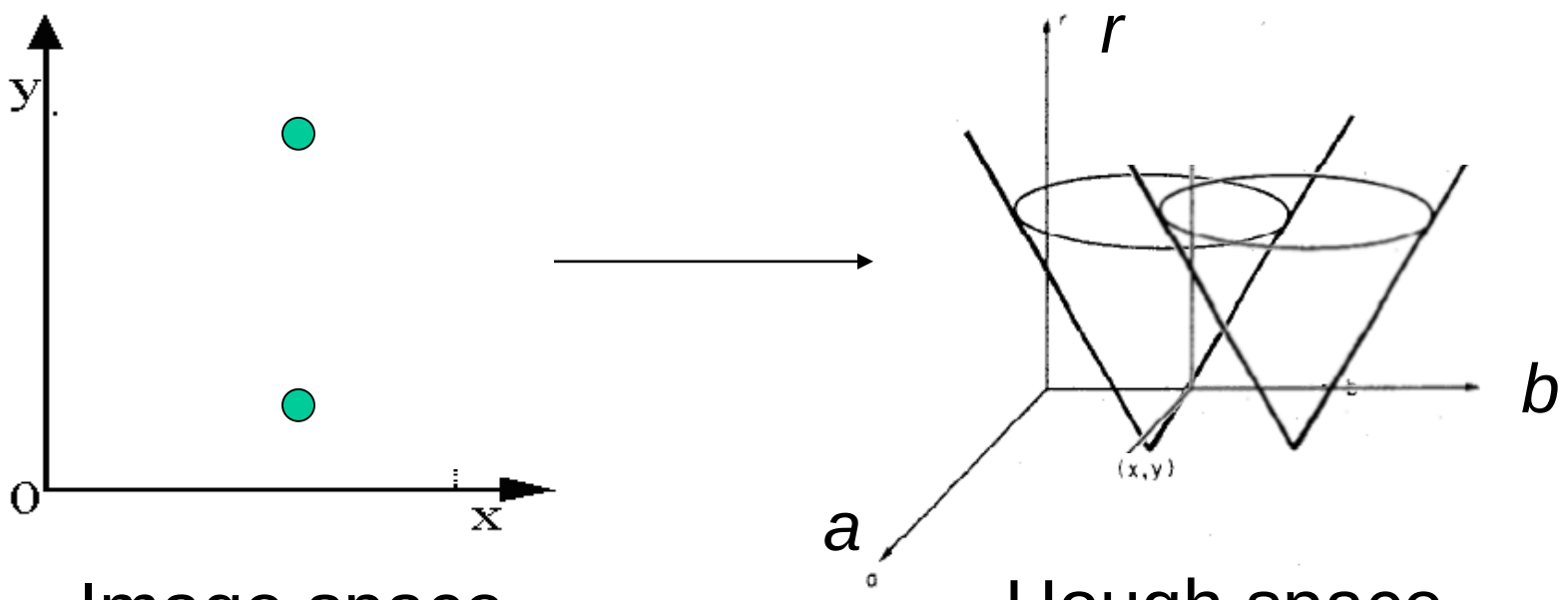
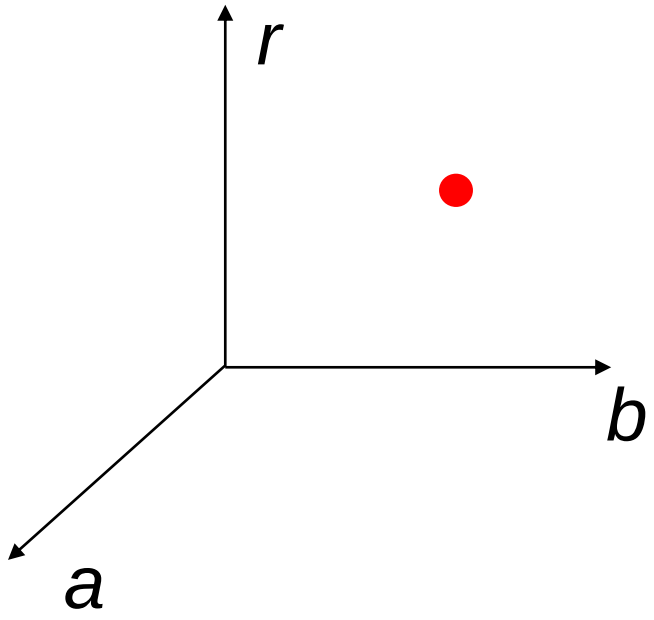


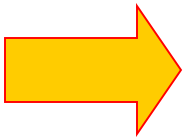
Image space

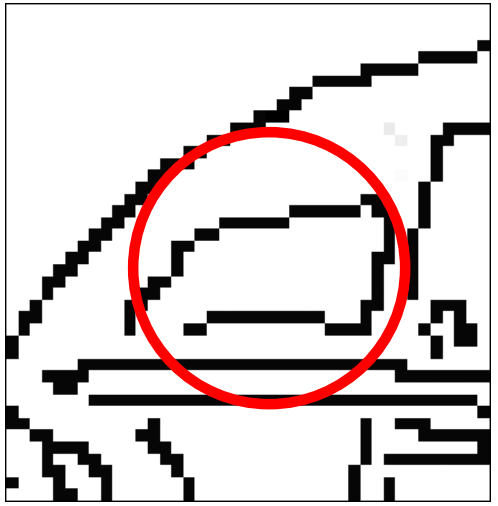
Hough space

How about this…

- Conceptually equivalent procedure: for each (a,b,r), draw the corresponding circle in the image and compute its “support”







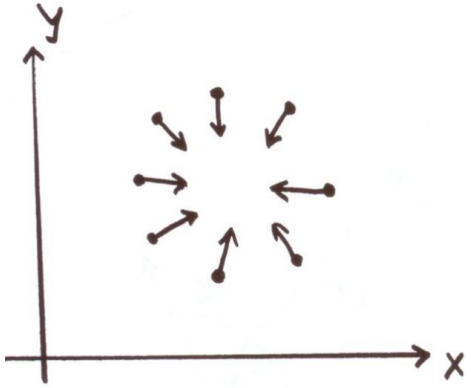
Is this more, or less efficient than voting with features?

Using Gradient Information

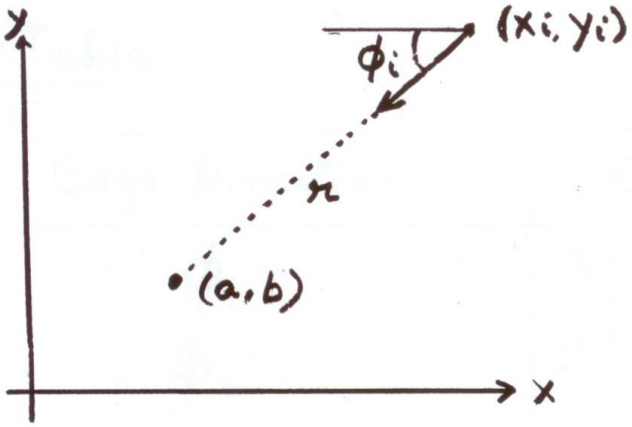
Gradient information can save lot of computation:

Edge Location (x_i, y_i)

Edge Direction ϕ_i



Assume radius is known:



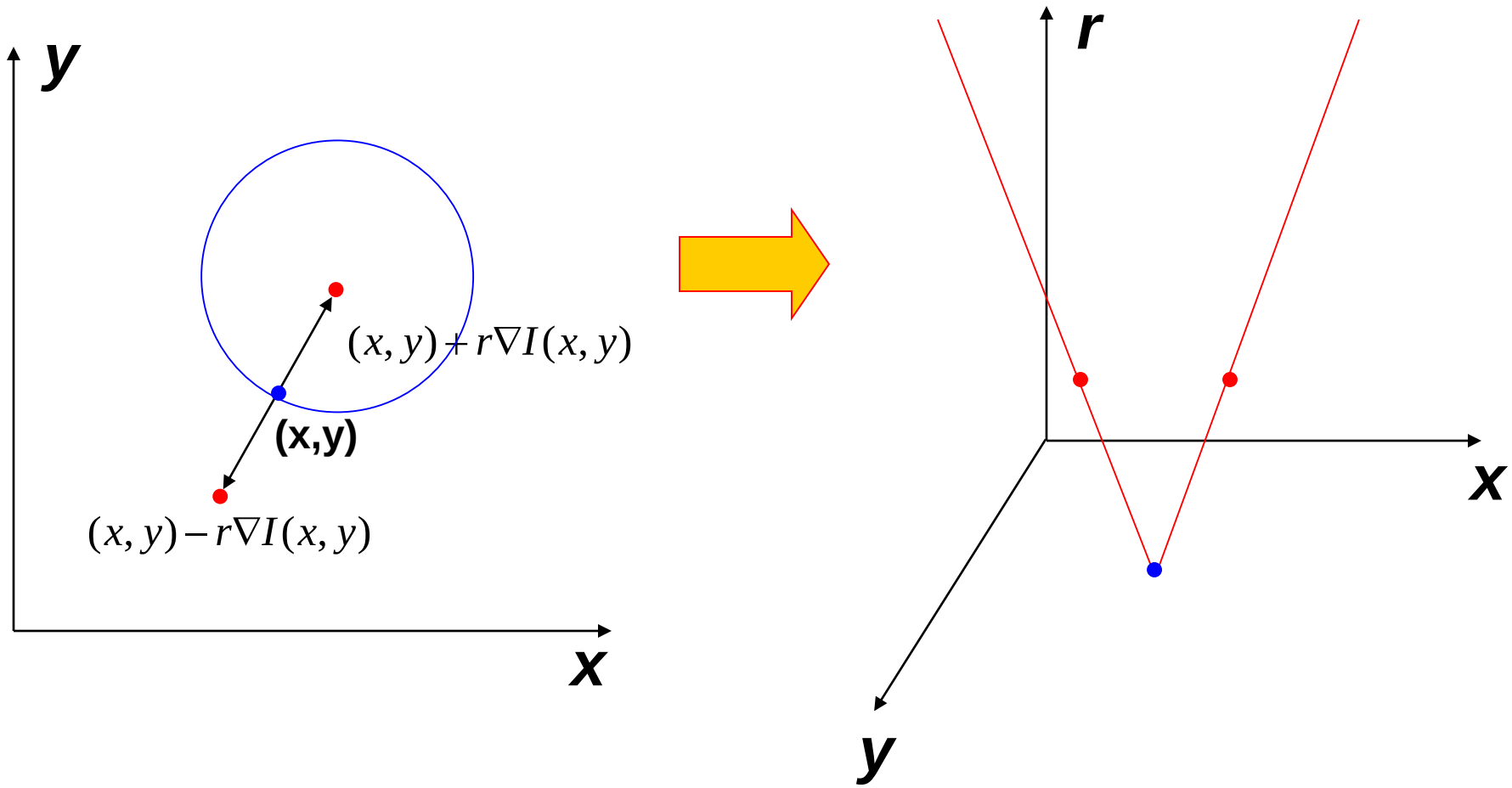
$$a = x - r\cos\phi$$

$$b = y - r\sin\phi$$

Hough transform for circles **given gradient**

image space

Hough parameter space



Hough transform for circles (given gradient information)

For every edge pixel (x,y) :

  For each possible radius value r:

    For each possible gradient direction θ:

      // or use estimated gradient at (x,y)

        $a = x - r\cos(\theta)$ // column

        $b = y + r\sin(\theta)$ // row

        H[a,b,r] += 1

  end

end

- Check out online demo :
  http://www.markschulze.net/java/hough/

Hough transform for ellipses

Equation of an ellipse:

$$\frac{(x_i \cos\vartheta + y_i \sin\vartheta - a)^2}{d_1^2} + \frac{(-x_i \sin\vartheta + y_i \cos\vartheta - b)^2}{d_2^2} = 1$$

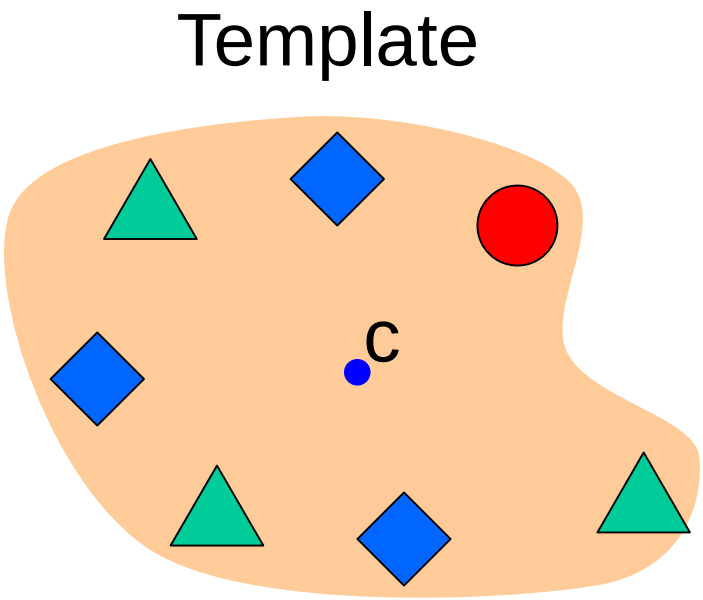
where (a,b) is the center of the ellipse, d_1 and d_2 are the lengths of the major and minor axes of the ellipse and ϑ is the angle between the major axis and the horizontal axis.

An ellipse is then uniquely identified by the location of its center, its size and its orientation, i.e. by the parameter tuple (a,b,d_1,d_2,ϑ)

For arbitrary ellipse detection we have a 5D Hough space. The accumulator array is $A(a,b,d_1,d_2,\vartheta)$.

Generalized Hough transform

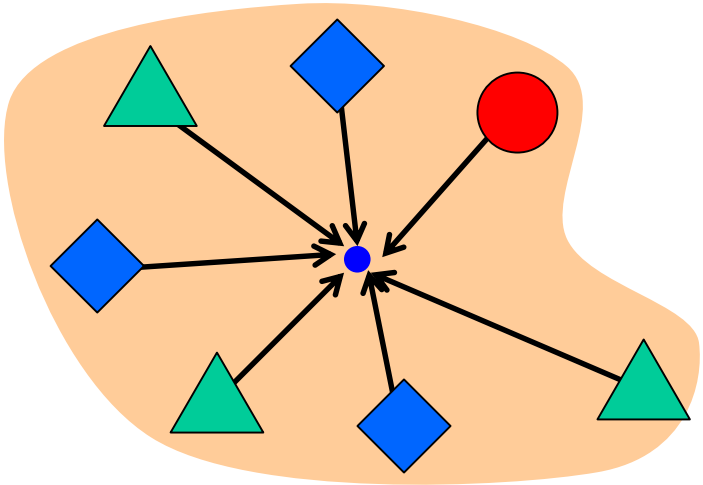
- We want to find a template defined by its reference point (center) and several distinct types of landmark points in stable spatial configuration



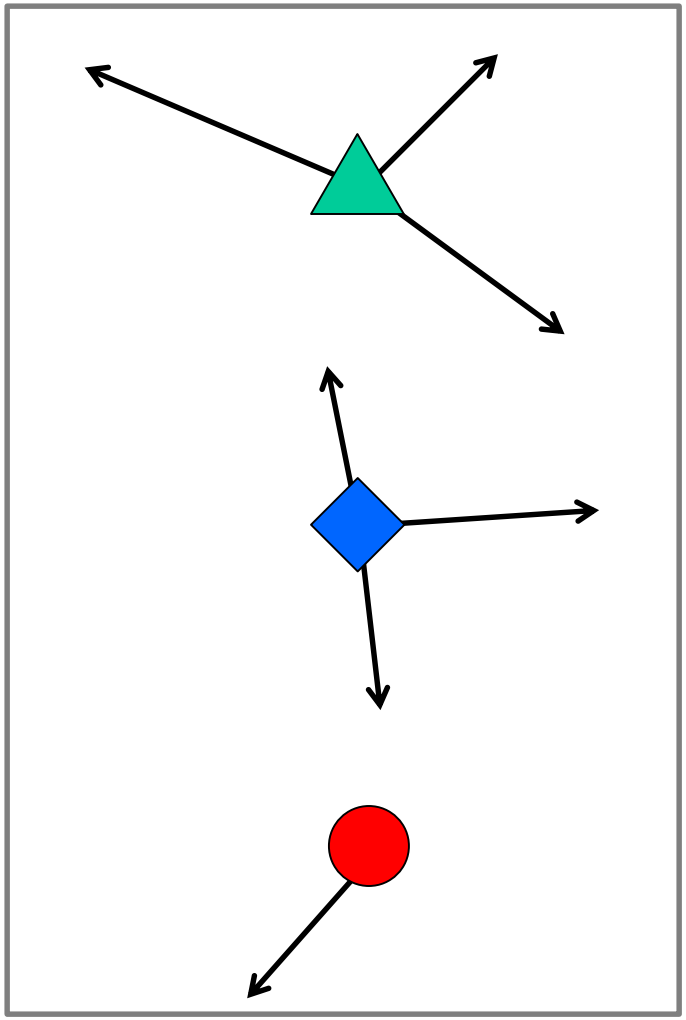
Generalized Hough transform

- Template representation: for each type of landmark point, store all possible displacement vectors towards the center

Template



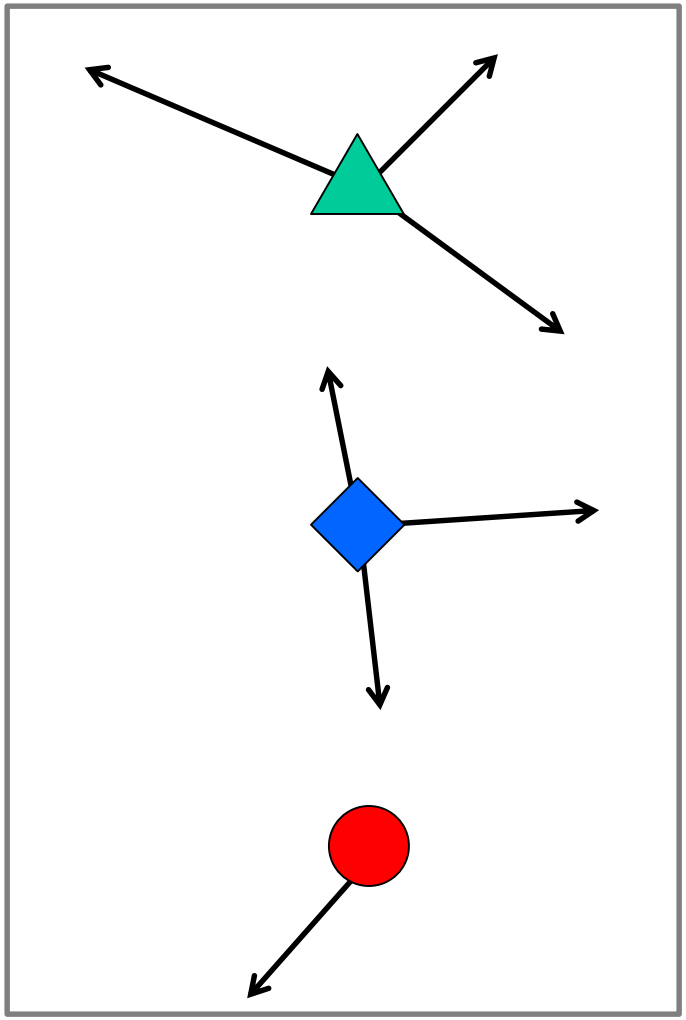
Model



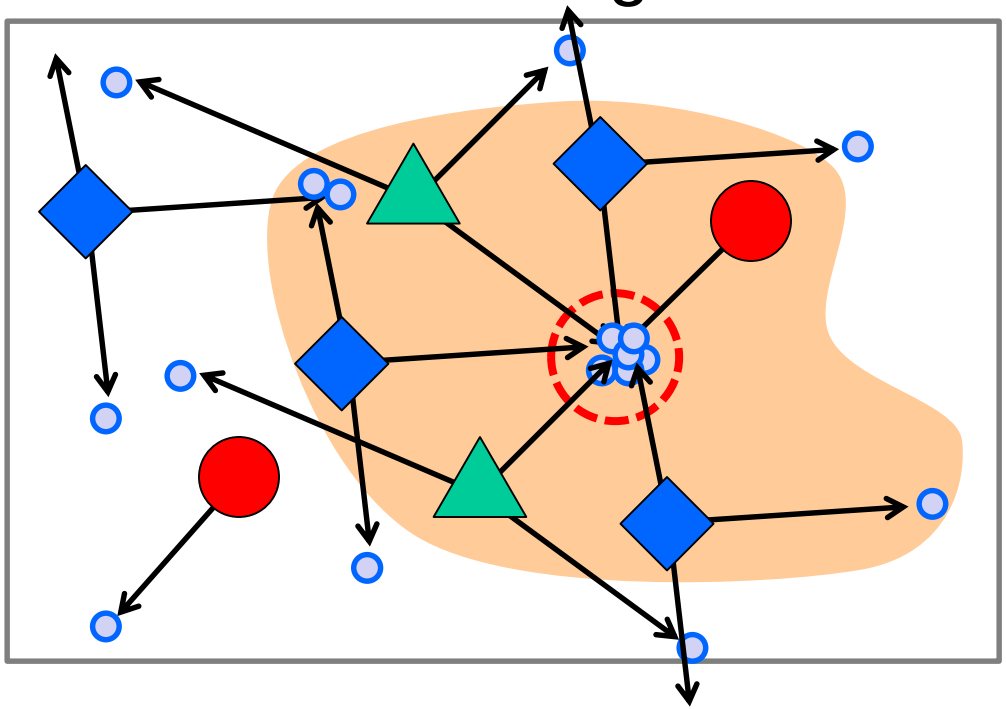
Generalized Hough transform

- Detecting the template:
  - For each feature in a new image, look up that feature type in the model and vote for the possible center locations associated with that type in the model

Model

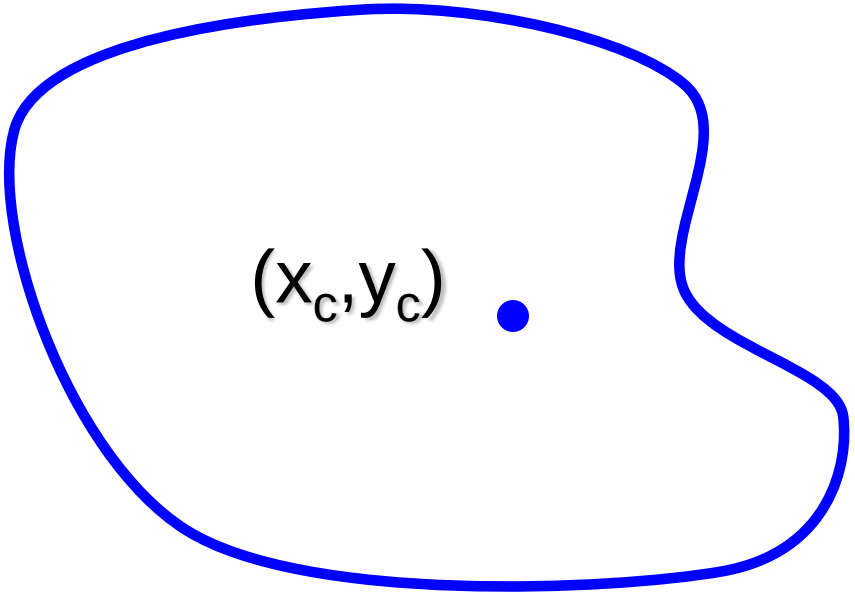


Test image



Generalized Hough transform

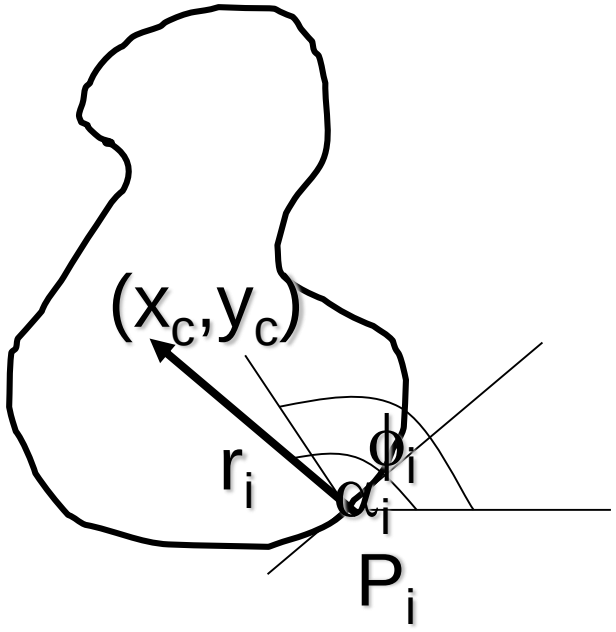
- We want to find a shape defined by its boundary points and a reference point



D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recognition 13(2), 1981, pp. 111-122.

Generalized Hough transform

The H.T. can be used even with curves that have no simple analytic form.
First, build a representation:



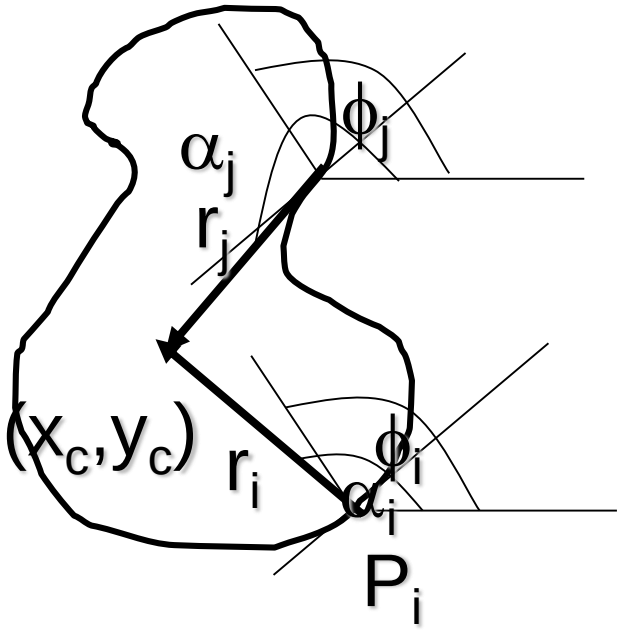
1. Pick a reference point (x_c,y_c)
2. For i = 1,…,n :
   1. Draw segment to P_i on the boundary.
   2. Measure its length r_i, and its orientation α_i.
   3. Write the coordinates of (x_c,y_c) as a function of r_i and α_i
   4. Record the gradient orientation ϕ_i at P_i.
3. Build a table with the data, indexed by ϕ_i .

$$x_c = x_i + r_i\cos(\alpha_i)$$
$$y_c = y_i + r_i\sin(\alpha_i)$$

Generalized Hough transform

Suppose, there were m ***different*** gradient orientations:

(m <= n)



$$x_c = x_i + r_i\cos(\alpha_i)$$

$$y_c = y_i + r_i\sin(\alpha_i)$$

ϕ_1	$(r^1_1,\alpha^1_1),(r^1_2,\alpha^1_2),\ldots,(r^1_{n1},\alpha^1_{n1})$
ϕ_2	$(r^2_1,\alpha^2_1),(r^2_2,\alpha^1_2),\ldots,(r^2_{n2},\alpha^1_{n2})$
.	.
.	
ϕ_m	$(r^m_1,\alpha^m_1),(r^m_2,\alpha^m_2),,(r^m_{nm},\alpha^m_{nm})$

H.T. table

Generalized Hough transform (A)

Finds a translated version of the curve:

1. Form an A accumulator array of possible reference points (x_c,y_c).
2. For each edge (x_i, y_i) in the image:
    1. Compute $\phi(x_i, y_i)$
    2. For each (r,α) corresponding to $\phi(x_i, y_i)$ do:
        1. $x_c = x_i + r(\phi) \cos[\alpha(\phi)]$
        2. $y_c = y_i + r(\phi) \sin[\alpha(\phi)]$
        3. $A(x_c,y_c)$++
3. Find maxima of A.

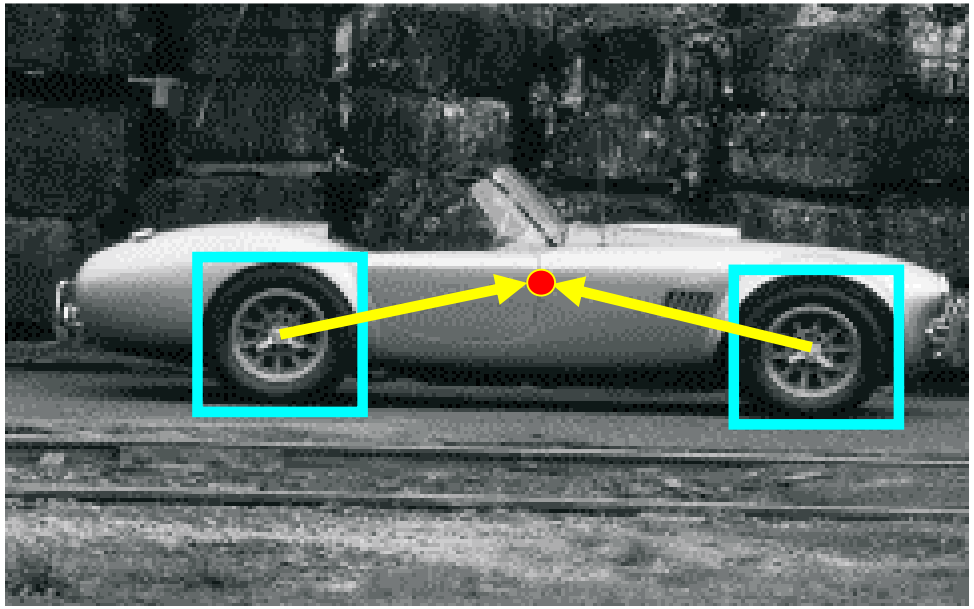
Generalized Hough transform (B)

Finds a **rotated, scaled**, and translated version of the curve:

1. Form an A accumulator array of possible reference points (x_c,y_c), scaling factor S and Rotation angle θ.
2. For each edge (x,y) in the image:
    1. Compute $\phi(x,y)$
    2. For each (r,α) corresponding to $\phi(x,y)$ do:
        1. For each S and for each θ:
            1. $x_c = x_i + r(\phi)\, S \cos[\alpha(\phi) + \theta]$
            2. $y_c = y_i + r(\phi)\, S \sin[\alpha(\phi) + \theta]$
            3. $A(x_c,y_c,S,\theta)$ ++
    3. Find maxima of A.

Application in recognition

- Index displacements by “visual codeword”



training image

visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Application in recognition

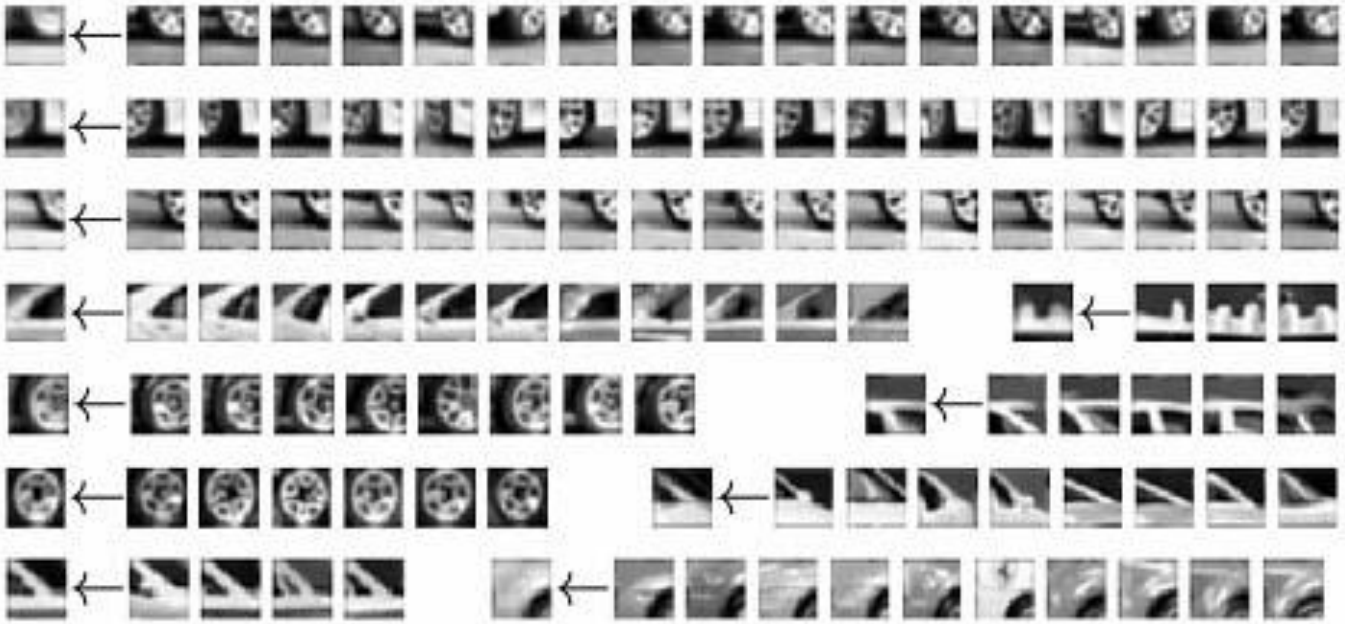
- Index displacements by “visual codeword”

test image

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

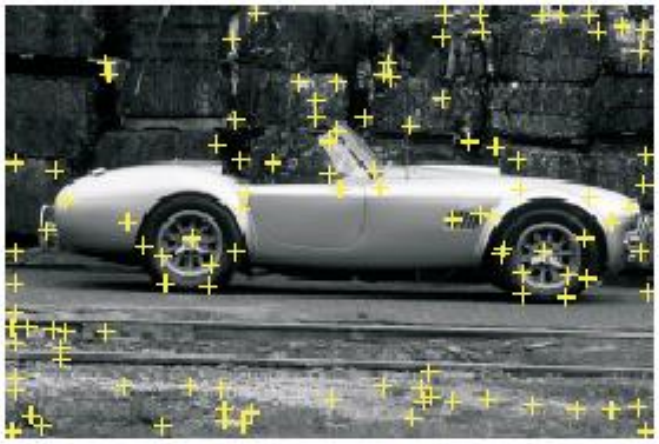
Implicit shape models: Training

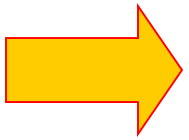
1. Build *codebook* of patches around extracted interest points using clustering (more on this later in the course)

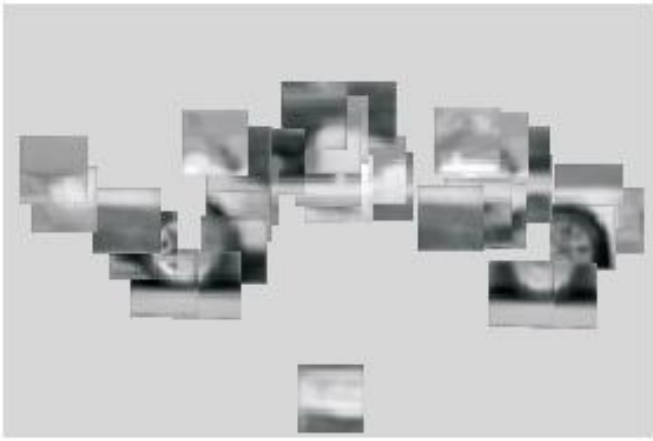


Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry

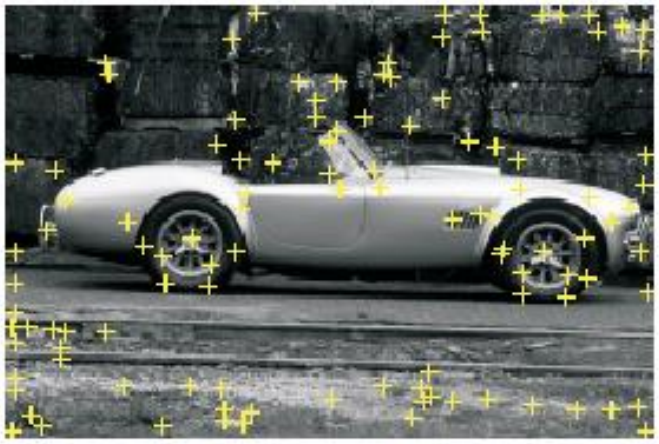


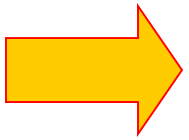


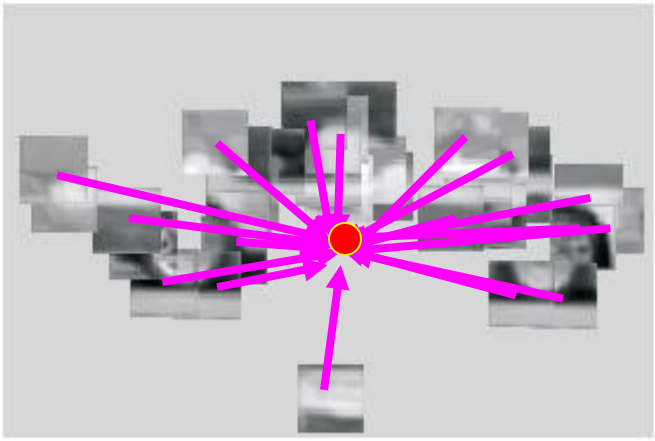


Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry
3. For each codebook entry, store all positions it was found, relative to object center

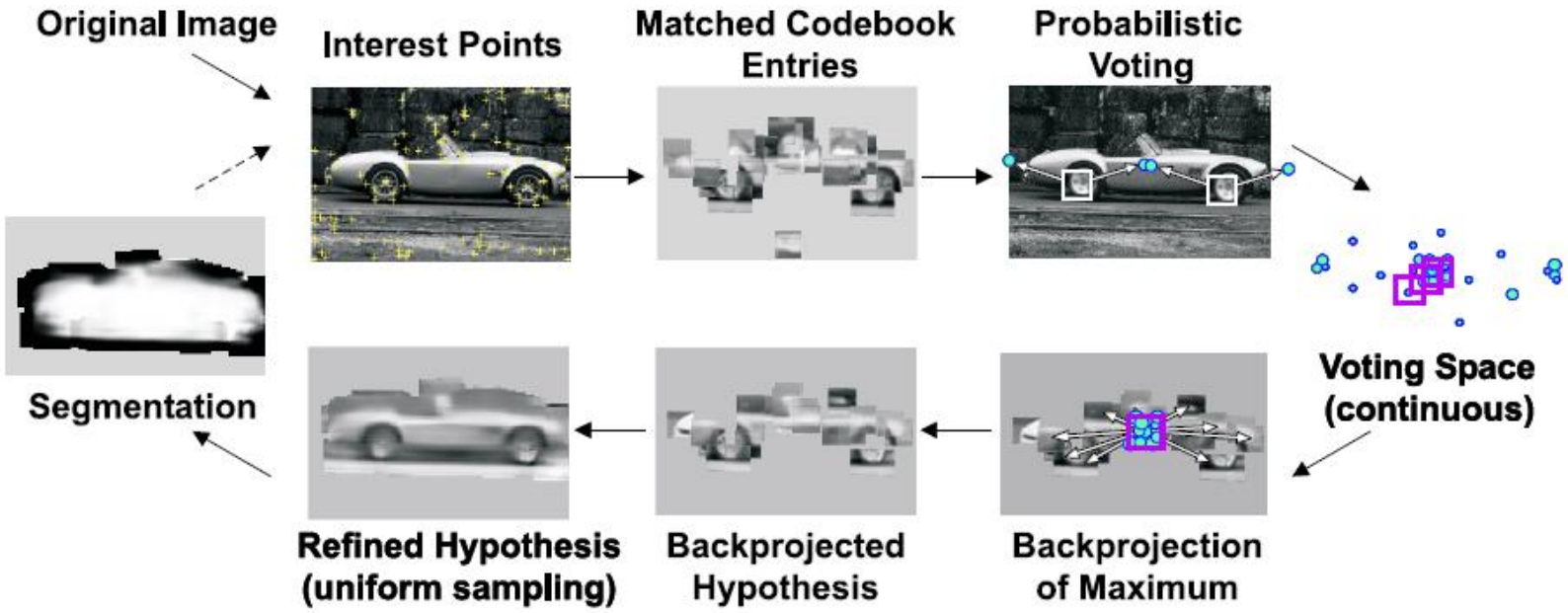






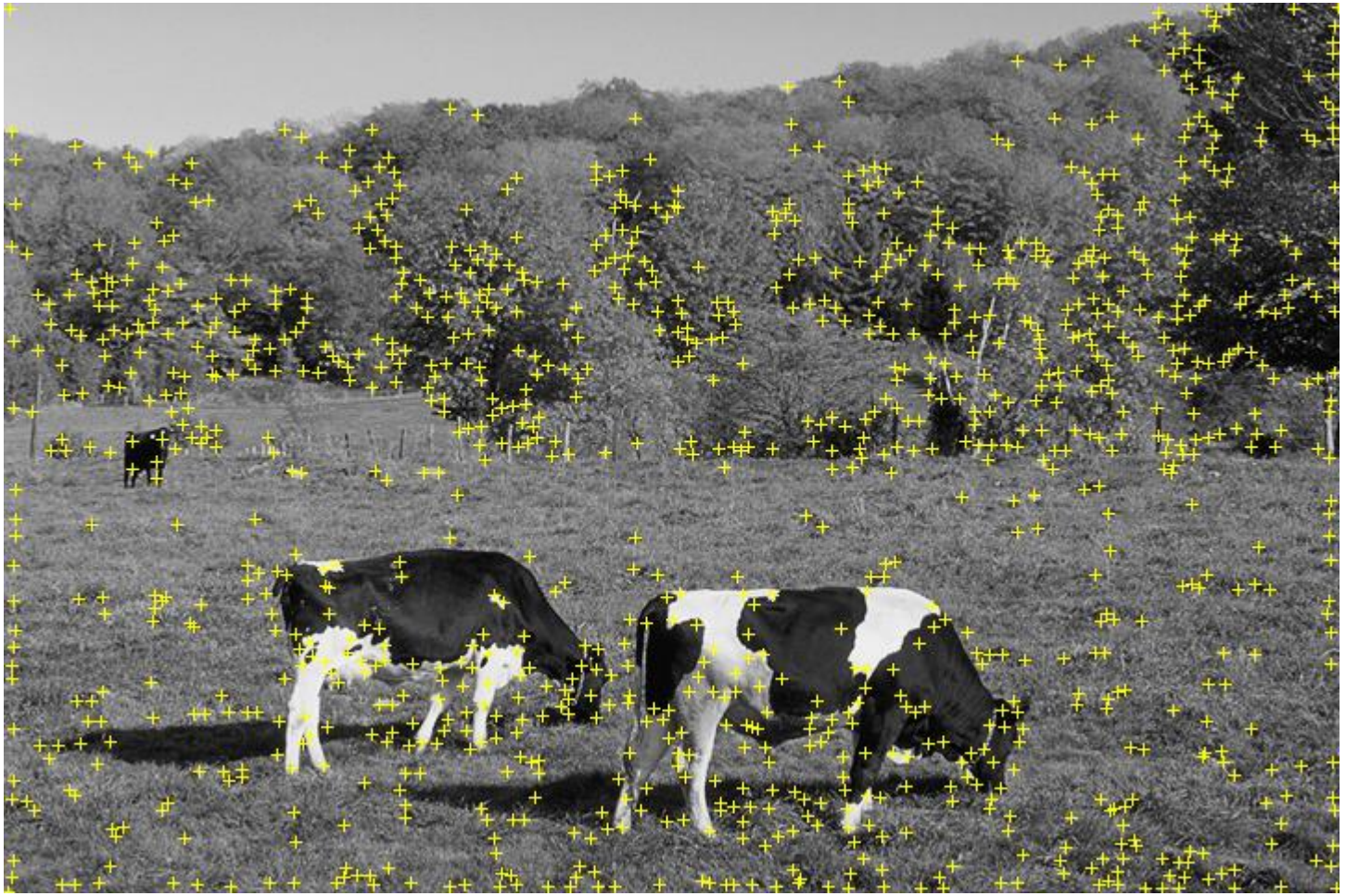
Implicit shape models: Testing

1. Given test image, extract patches, match to codebook entry
2. Cast votes for possible positions of object center
3. Search for maxima in voting space
4. Extract weighted segmentation mask based on stored masks for the codebook occurrences

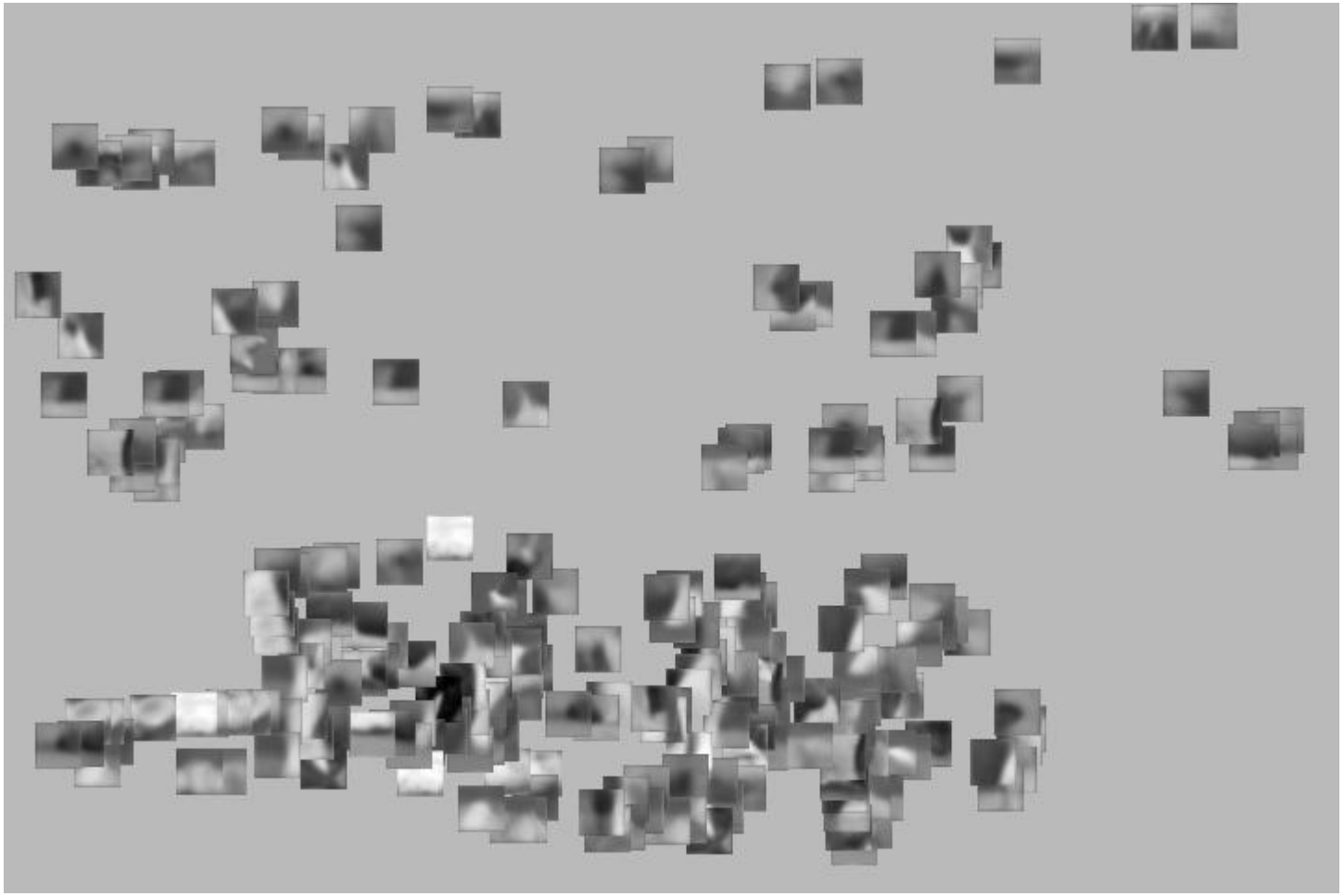


Example: Results on Cows

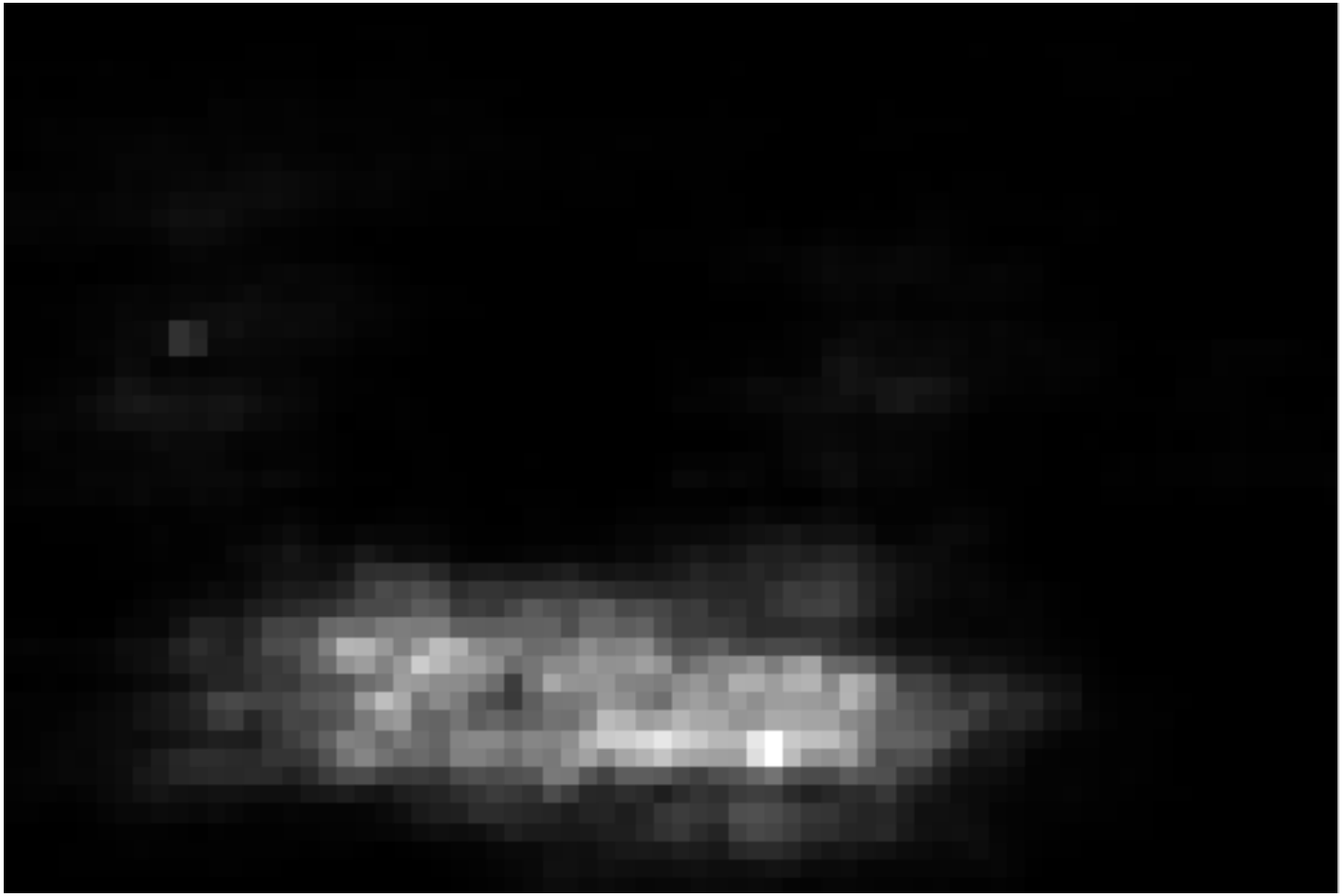
Example: Results on Cows



Example: Results on Cows



Example: Results on Cows

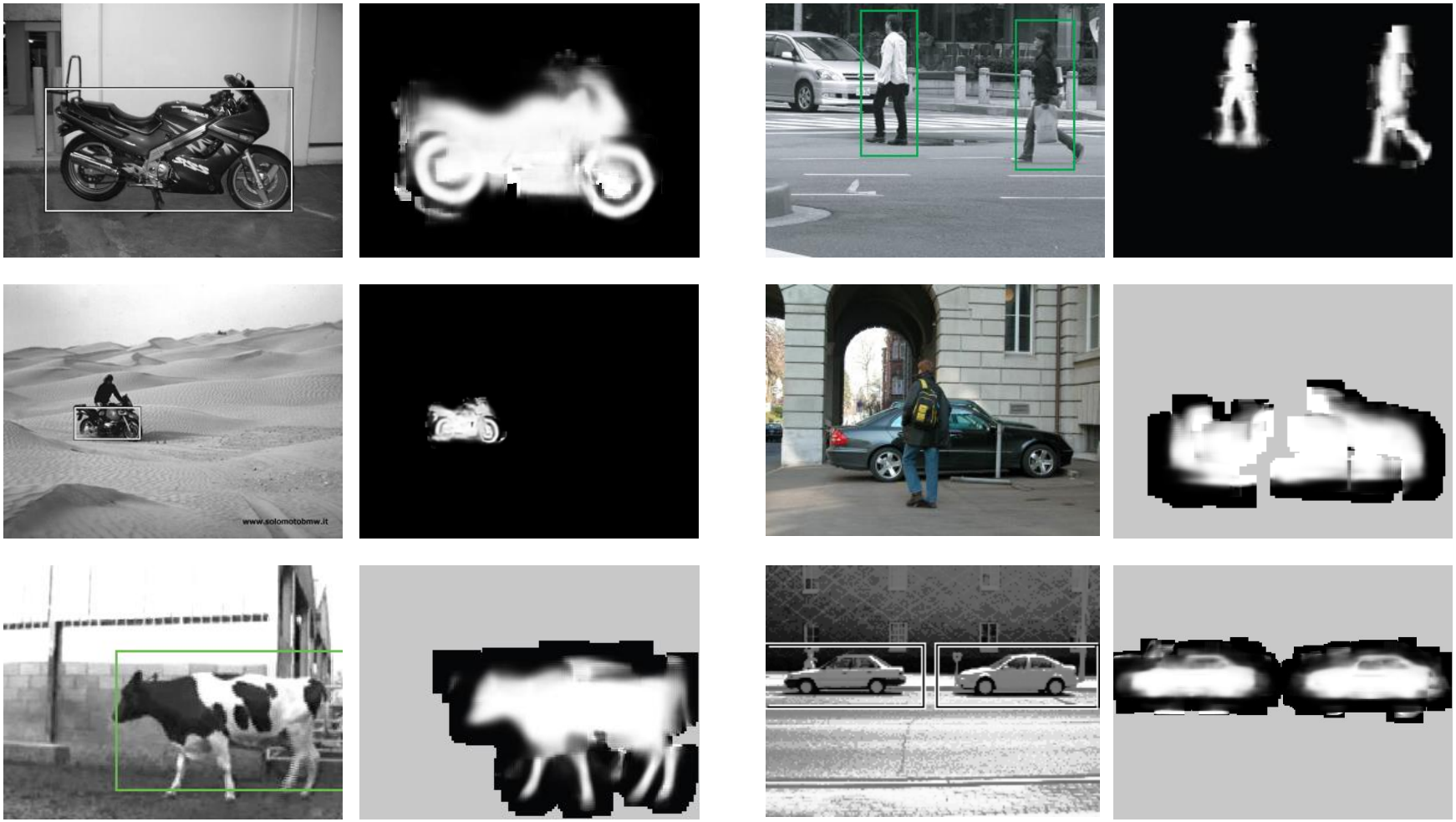


Example: Results on Cows

Example: Results on Cows

Example: Results on Cows

Examples



B. Leibe, A. Leonardis, and B. Schiele, Robust Object Detection with Interleaved Categorization and Segmentation, IJCV 77 (1-3), pp. 259-289, 2008.

Hough transform: Pros and cons

- Pros
  - Can deal with non-locality and occlusion
  - Can detect multiple instances of a model
  - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Cons
  - Complexity of search time increases exponentially with the number of model parameters
  - Non-target shapes can produce spurious peaks in parameter space
  - It’s hard to pick a good grid size