Model Fitting: The Hough transform II

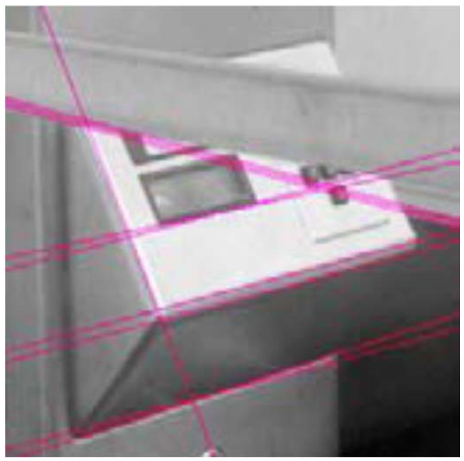
Guido Gerig, CS6640 Image Processing, Utah

Theory: See handwritten notes GG:
HT-notes-GG-II.pdf

Credits: S. Narasimhan, CMU, Spring 2006 15-385,-685, Link
Svetlana Lazebnik (Computer Vision UNC Chapel Hill, 2008), and Ioannis Stamos

Fitting Parametric Models: Beyond Lines

- Choose a parametric model to represent a set of features



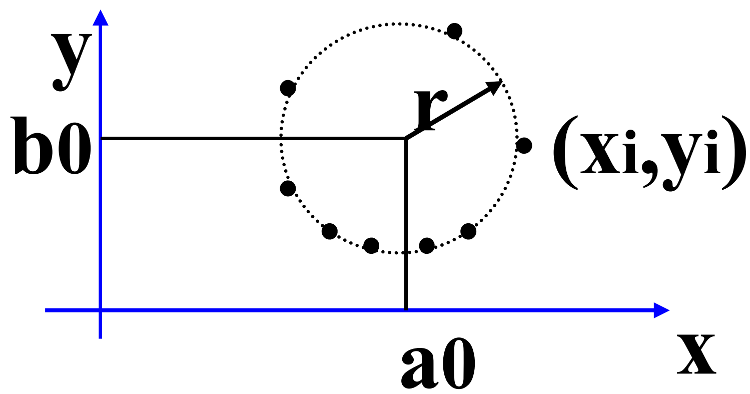
simple model: **lines**

simple model: **circles**

complicated model: **car**

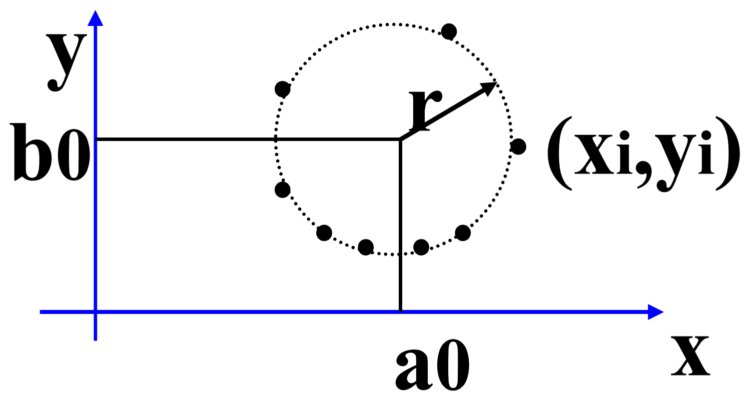
Source: K. Grauman

Finding Circles by Hough Transform



Equation of Circle: $(x_i - a_0)^2 + (y_i - b_0)^2 = r^2$

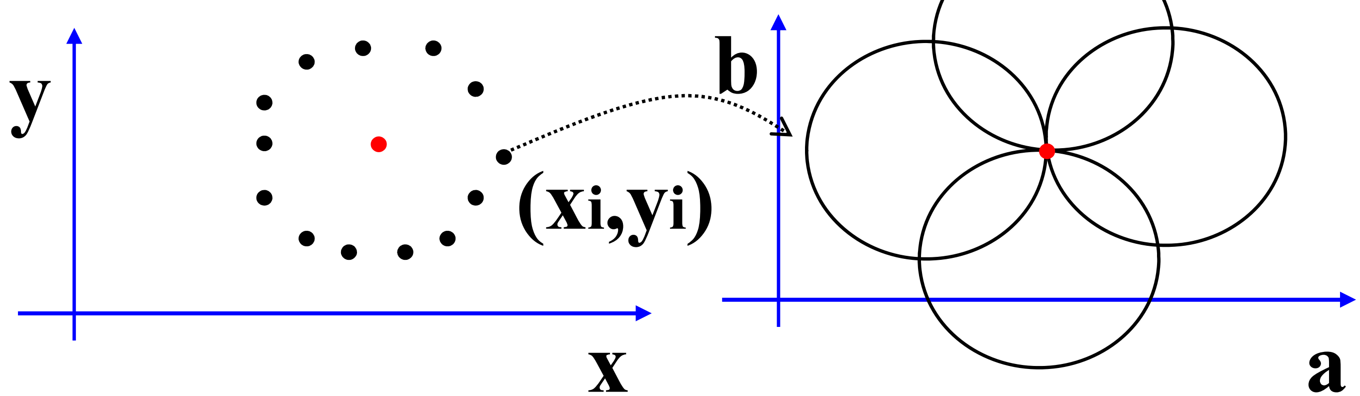
Finding Circles by Hough Transform



Equation of Circle: $(x_i - a_0)^2 + (y_i - b_0)^2 = r^2$

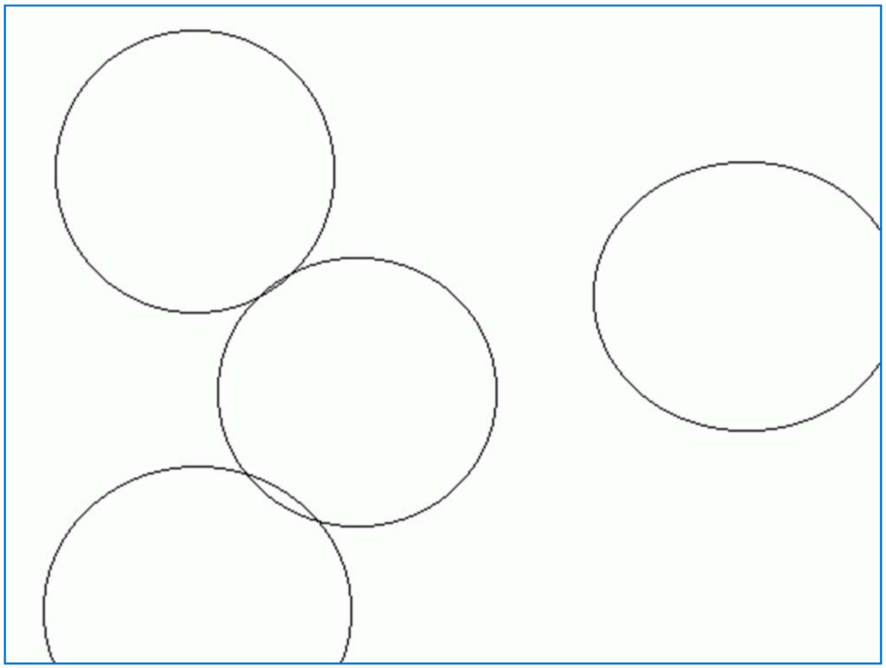
If radius r is known:

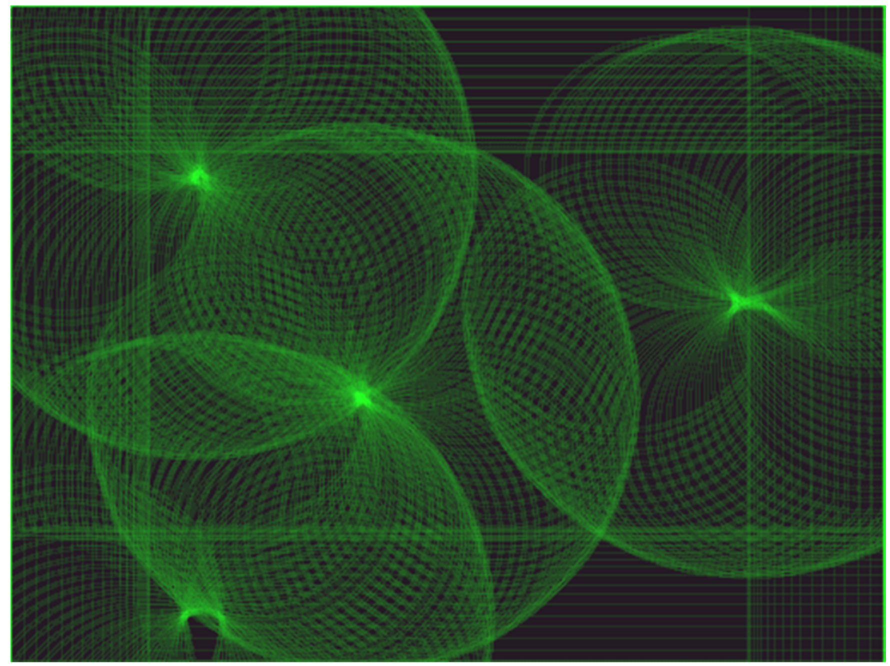
Circles!



Accumulator array A(a,b)

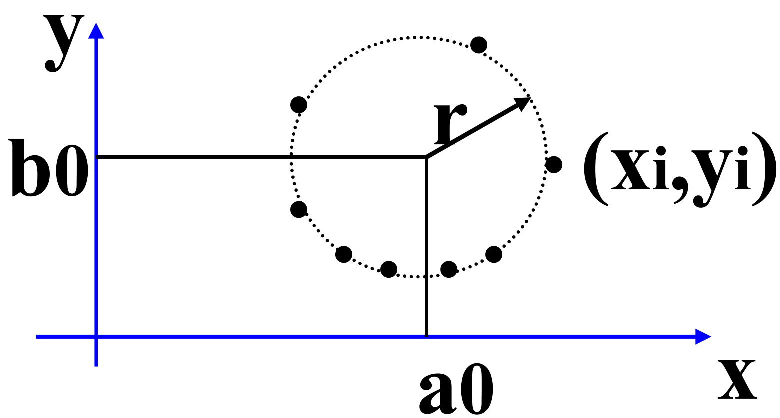
Example: Set of circles





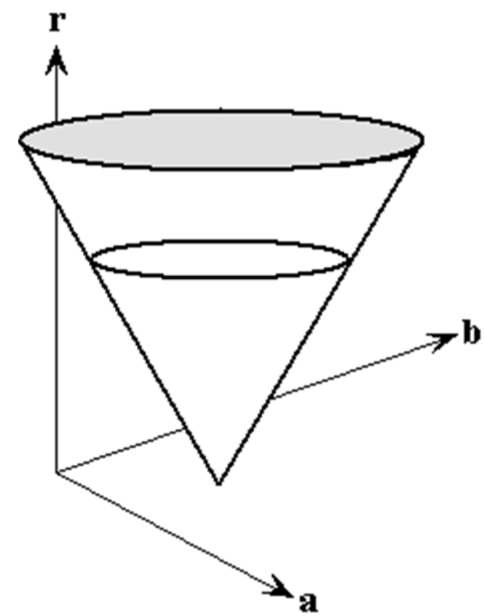
http://www.avishek.net/blog/wp-content/uploads/2011/07/circles_hough.gif

Finding Circles by Hough Transform



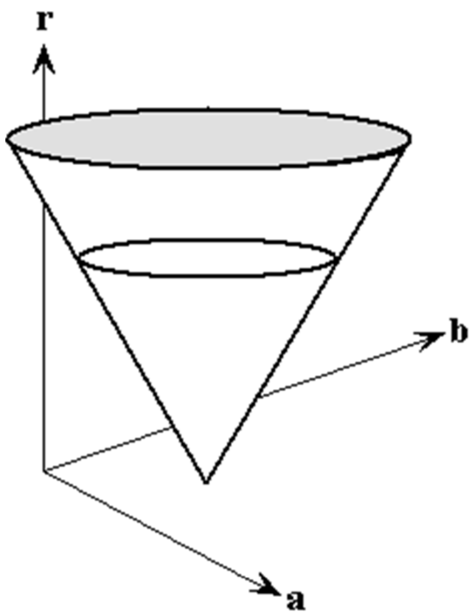
If r is not known
Use accumulator array A(a,b,r)

For each (xi,yi) increment A(a,b,r)
such that $(x_i - a)^2 + (y_i - b)^2 = r^2$

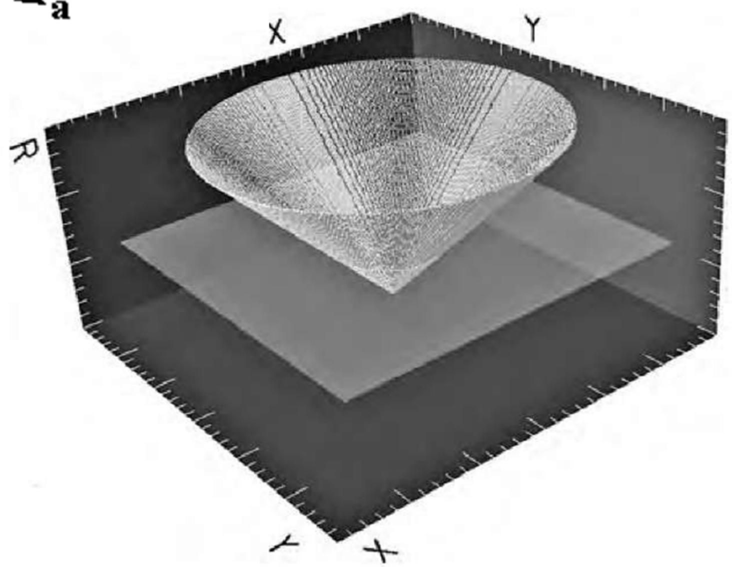


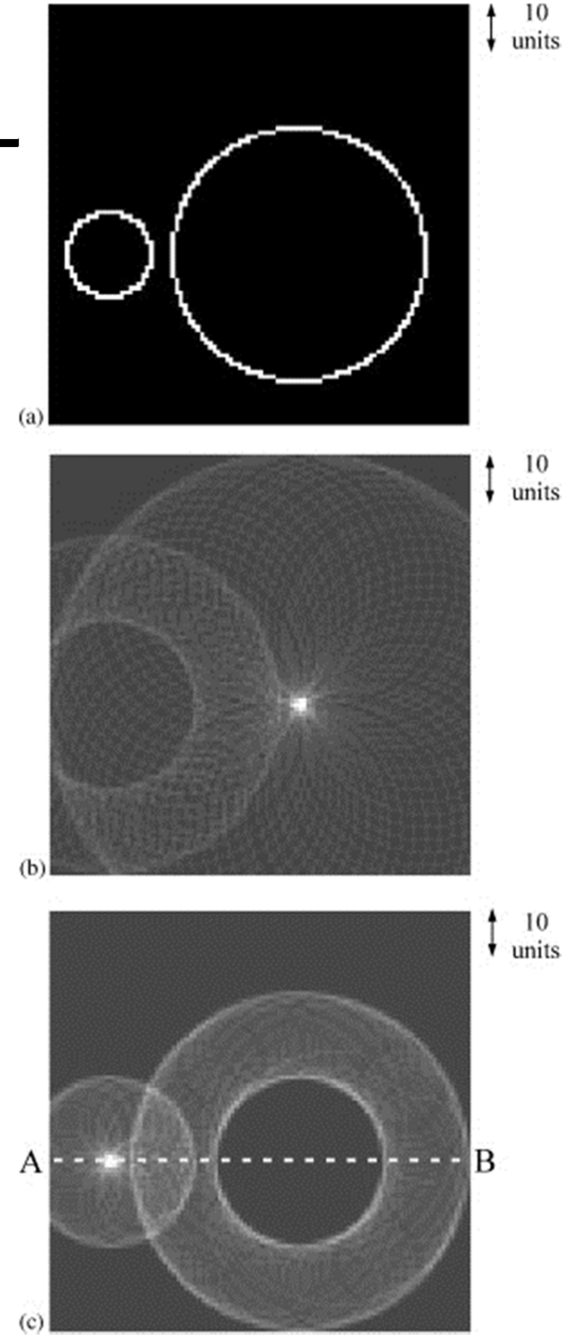
Parameter curves represent right cone.

HT for Circles



Each point in image space: Accumulate right cone in (a,b,r) accumulator.

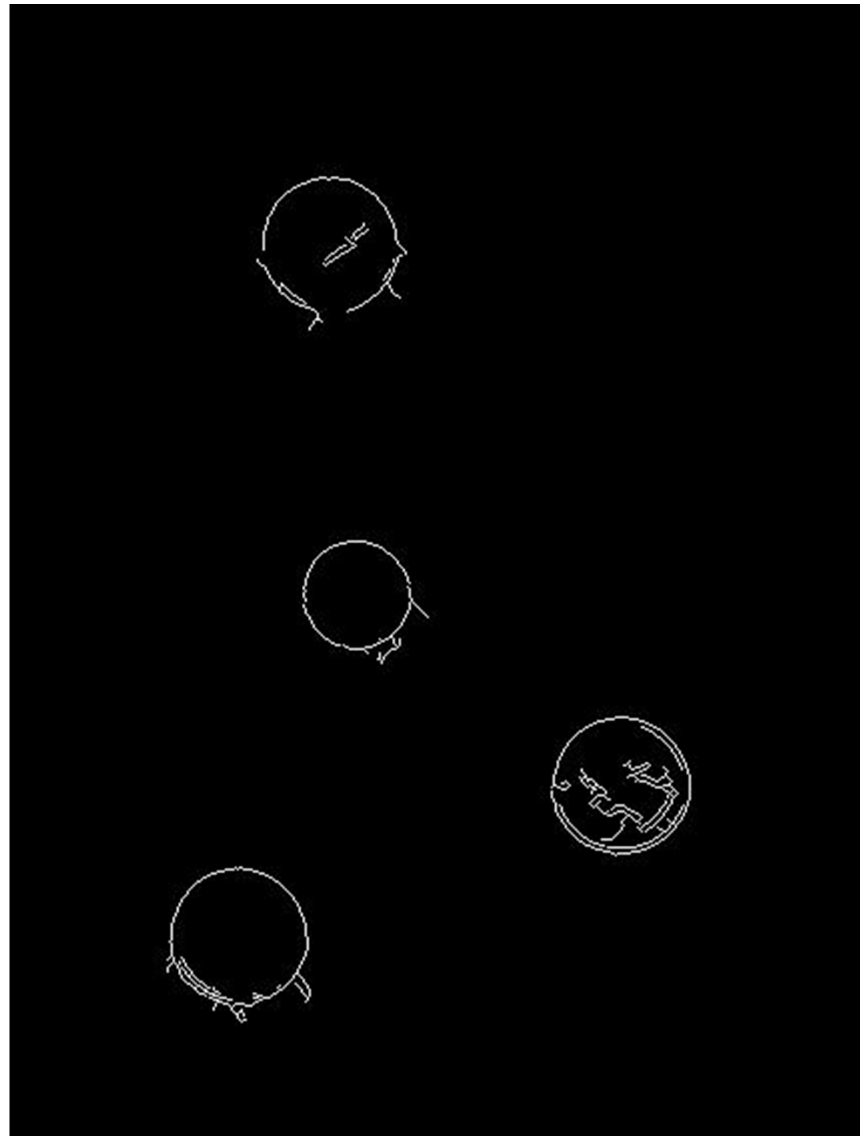




Finding Coins

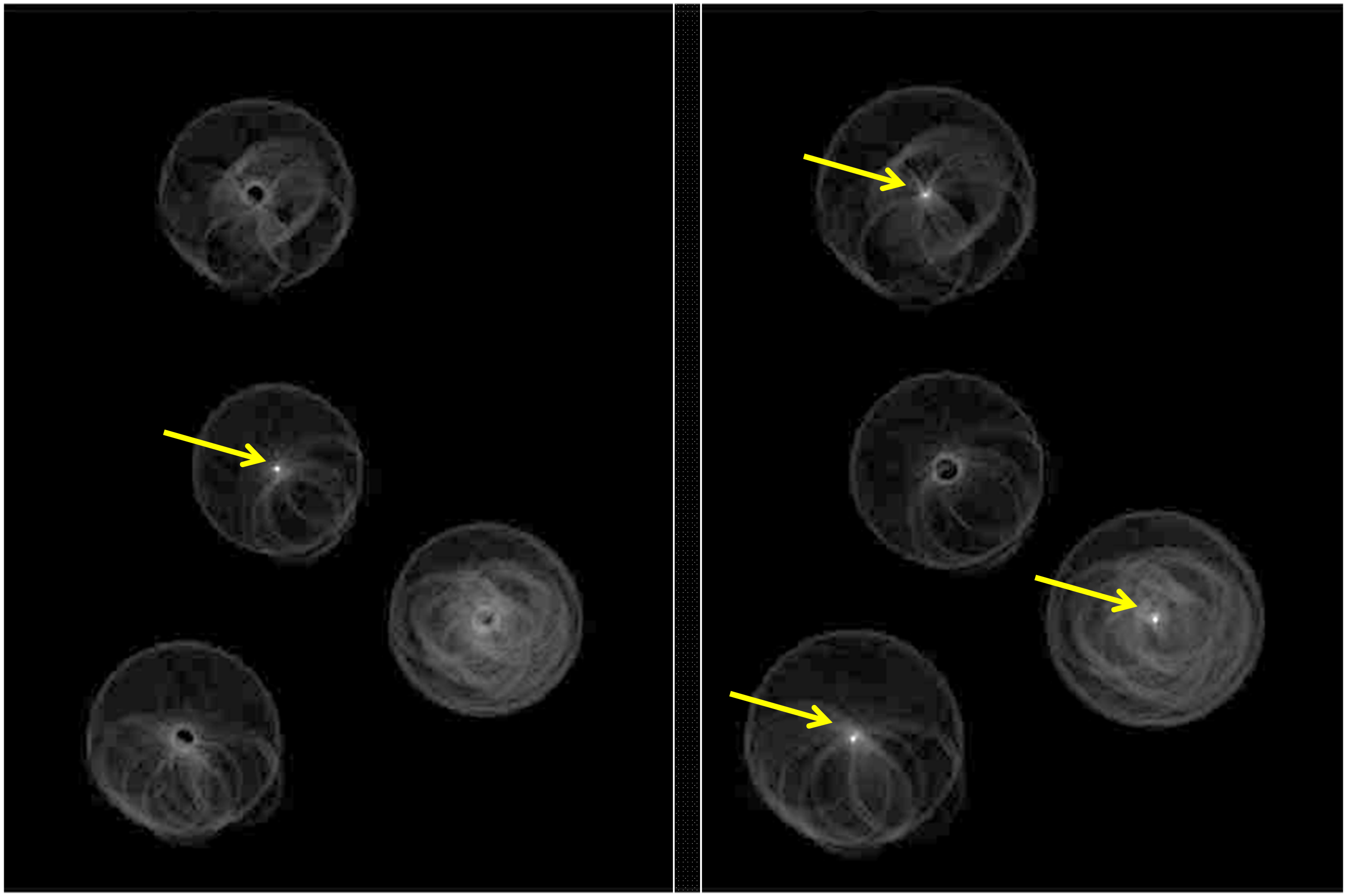
Original

Edges (note noise)

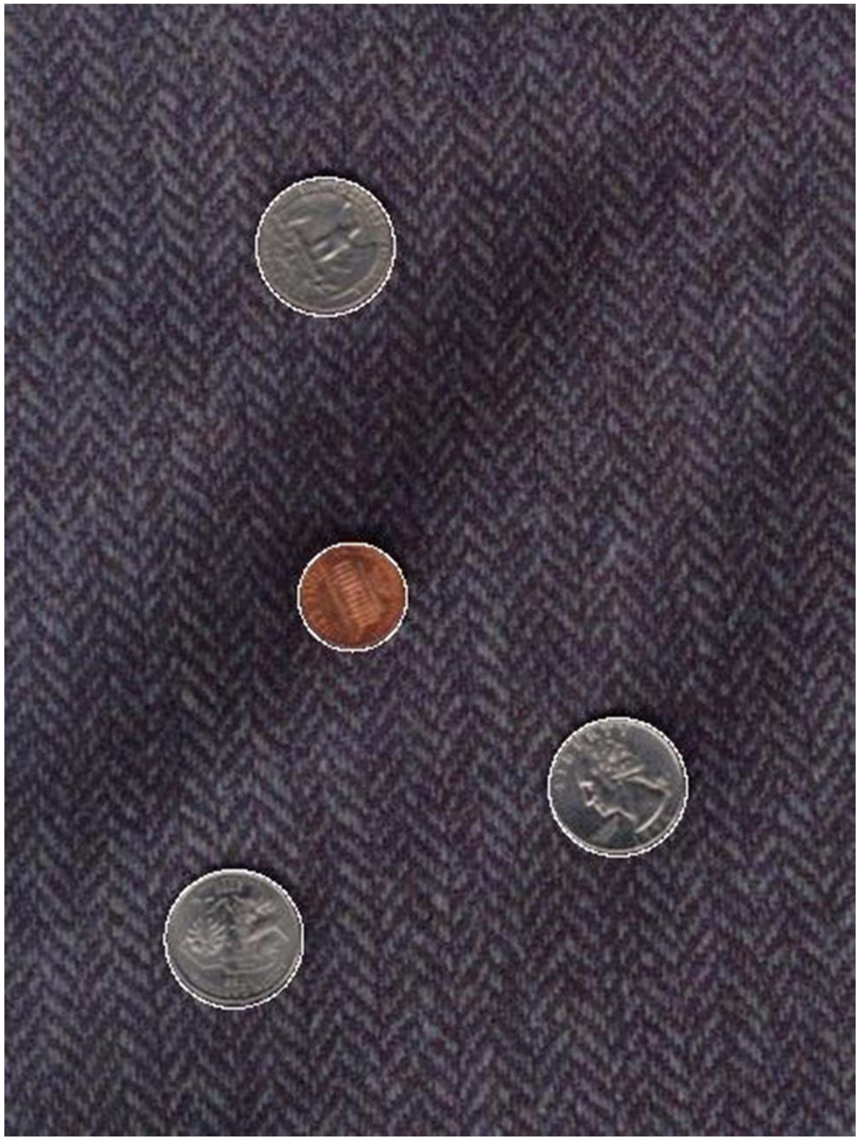


Finding Coins (Continued)

Penn **Quarter**



Finding Coins (Continued)



Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.

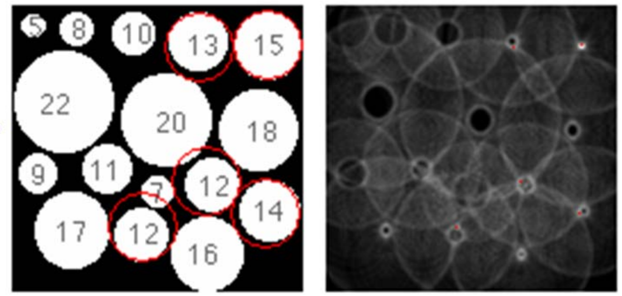
Coin finding sample images from: Vivek Kwatra

Real World Circle Examples

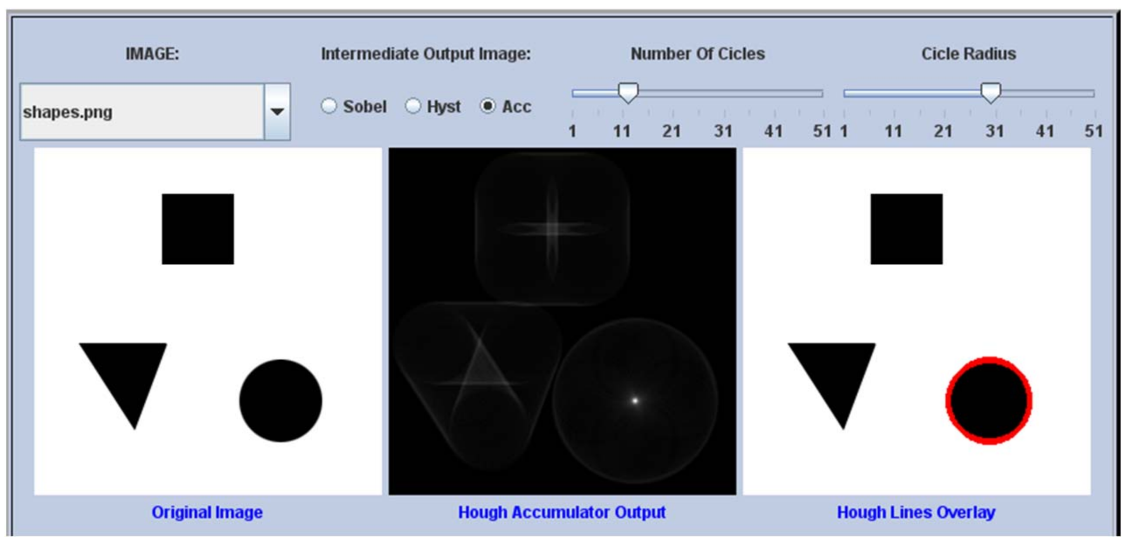
Crosshair indicates results of Hough transform, bounding box found via motion differencing.

Java Demos

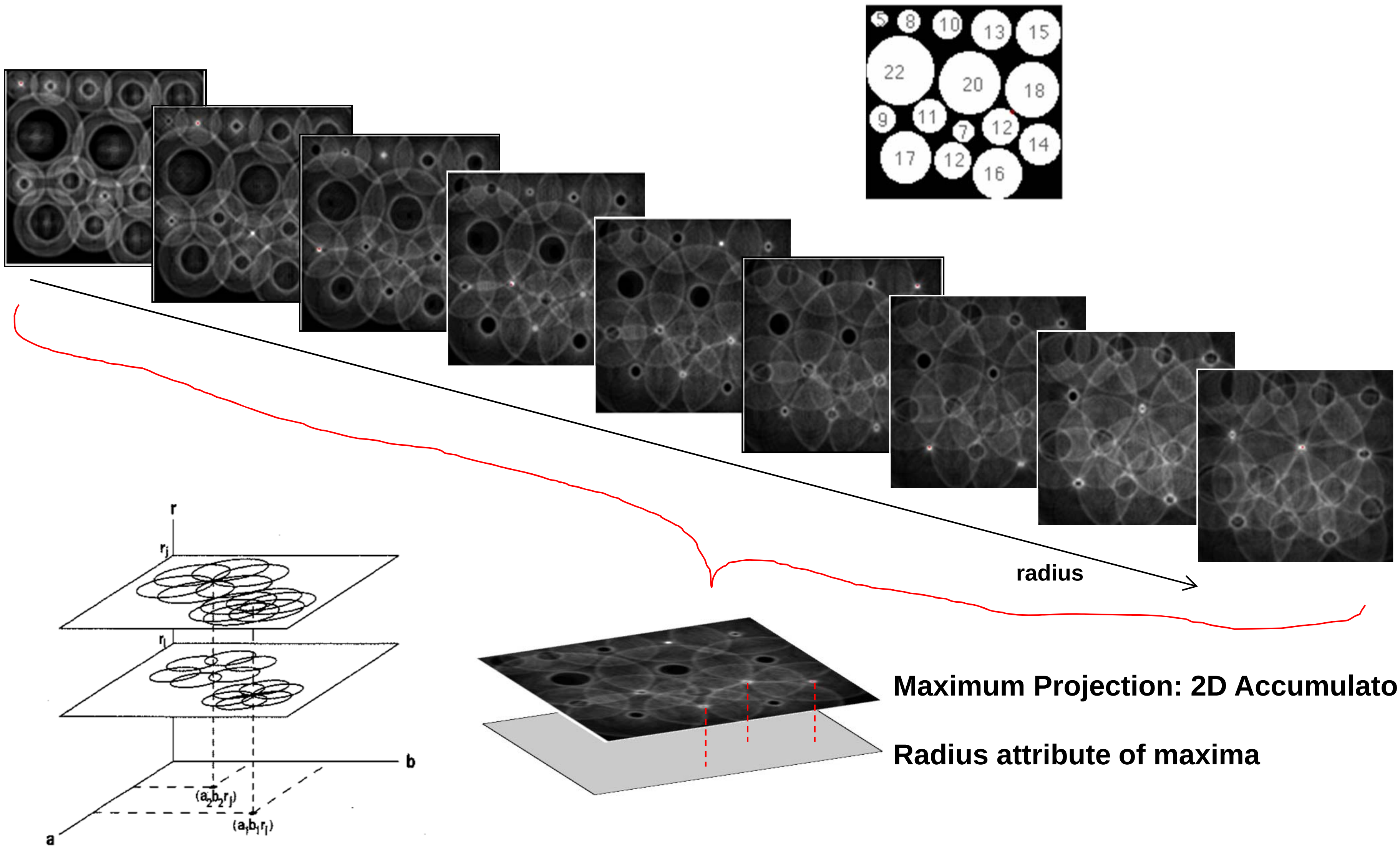
Java Demo Circle Detection I:
http://www.markschulze.net/java/hough/



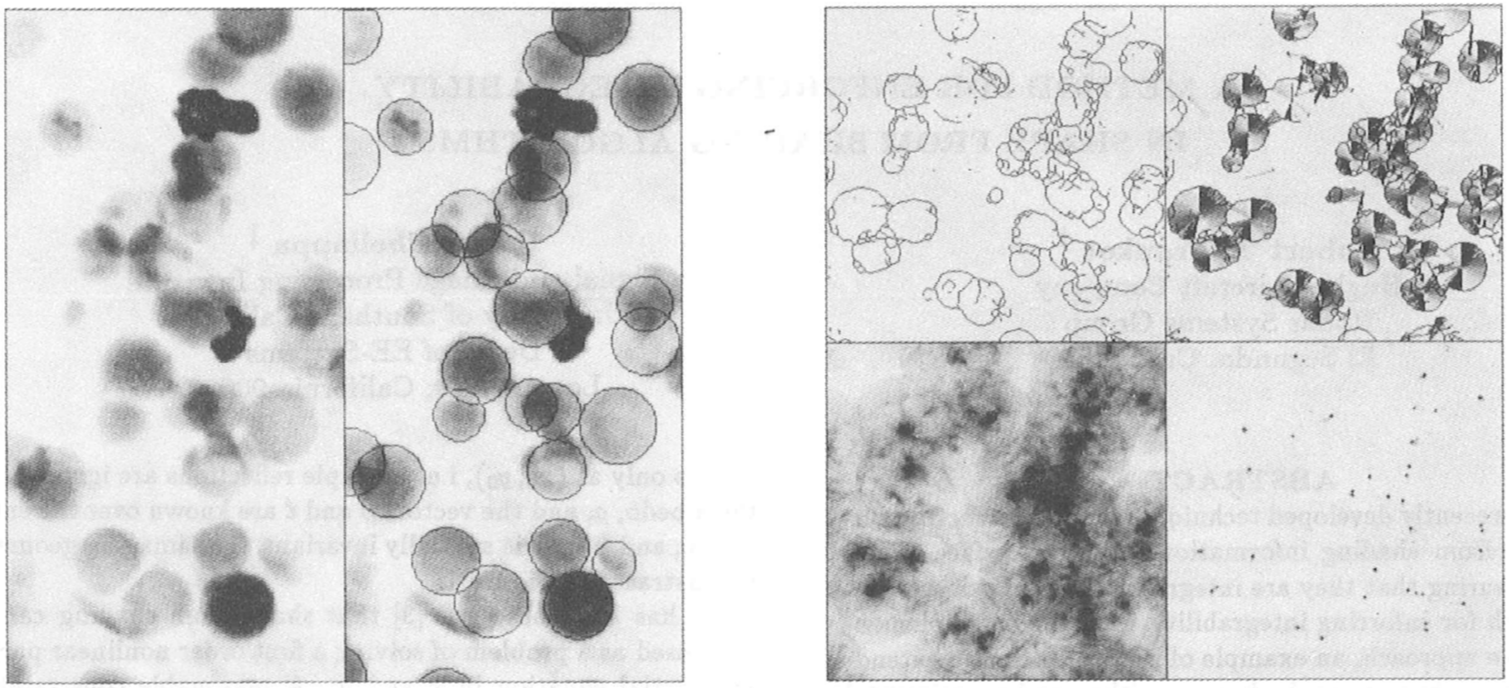
Java Demo Circle Detection II:
http://users.ecs.soton.ac.uk/msn/book/new_demo/houghCircles/



How to avoid 3D accumulator: Maximum Projection



How to avoid 3D accumulator: Maximum Projection



a | b

c | d
e | f

fig. 4: Circular boundary detection

a) (most difficult) subframe of original image

b) overlay of original image and classification result (black circles)

c) edge-features used for matching

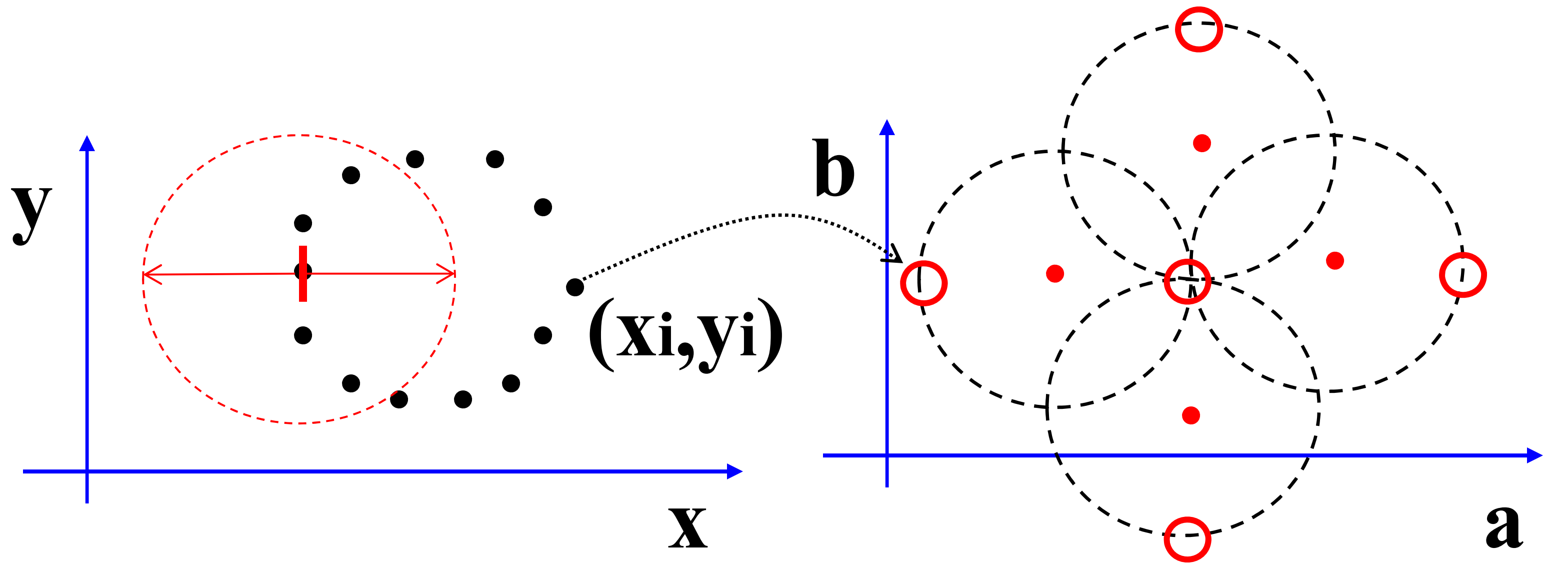
d) pointer vectors directing from surviving counts to contributing boundary points (grey indicates pointer orientation)

e) accumulator plane of evident center coordinates (scale-axis projected)

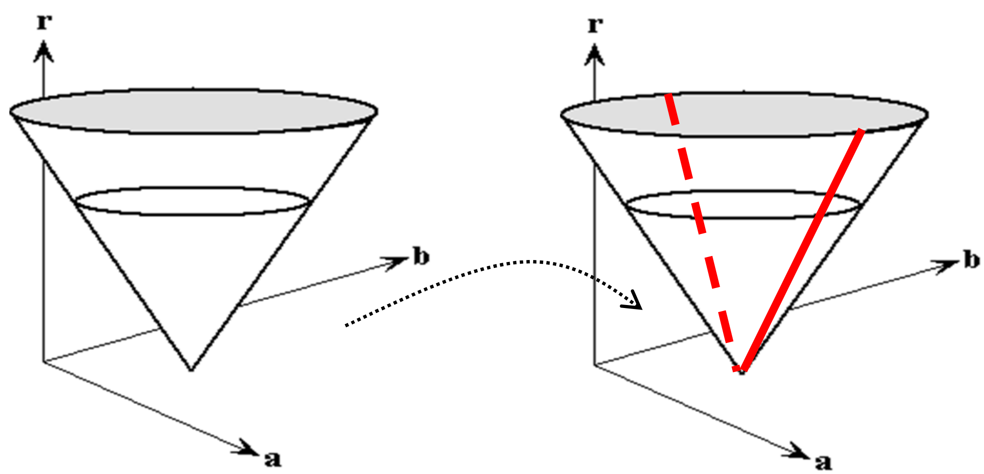
f) surviving accumulator counts after backmapping

See: Gerig et al., ICCV'87

If radius r and edge orientation known



Search for center reduces from circle to two locations only.



Radius not known:

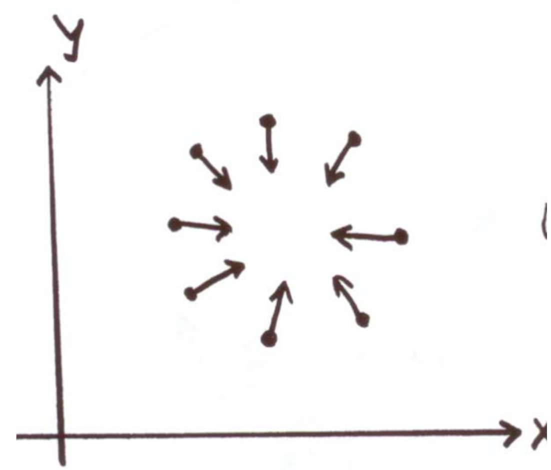
Search for center reduces from accumulation of cone to two lines only.

Using Gradient Information

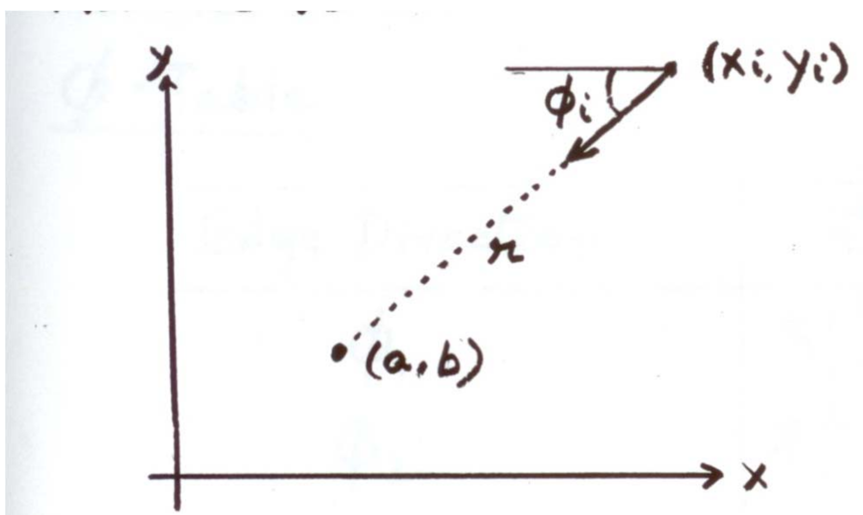
- **Gradient information can save lot of computation:**

Edge Location (x_i, y_i)

Edge Direction ϕ_i



Assume radius is known:

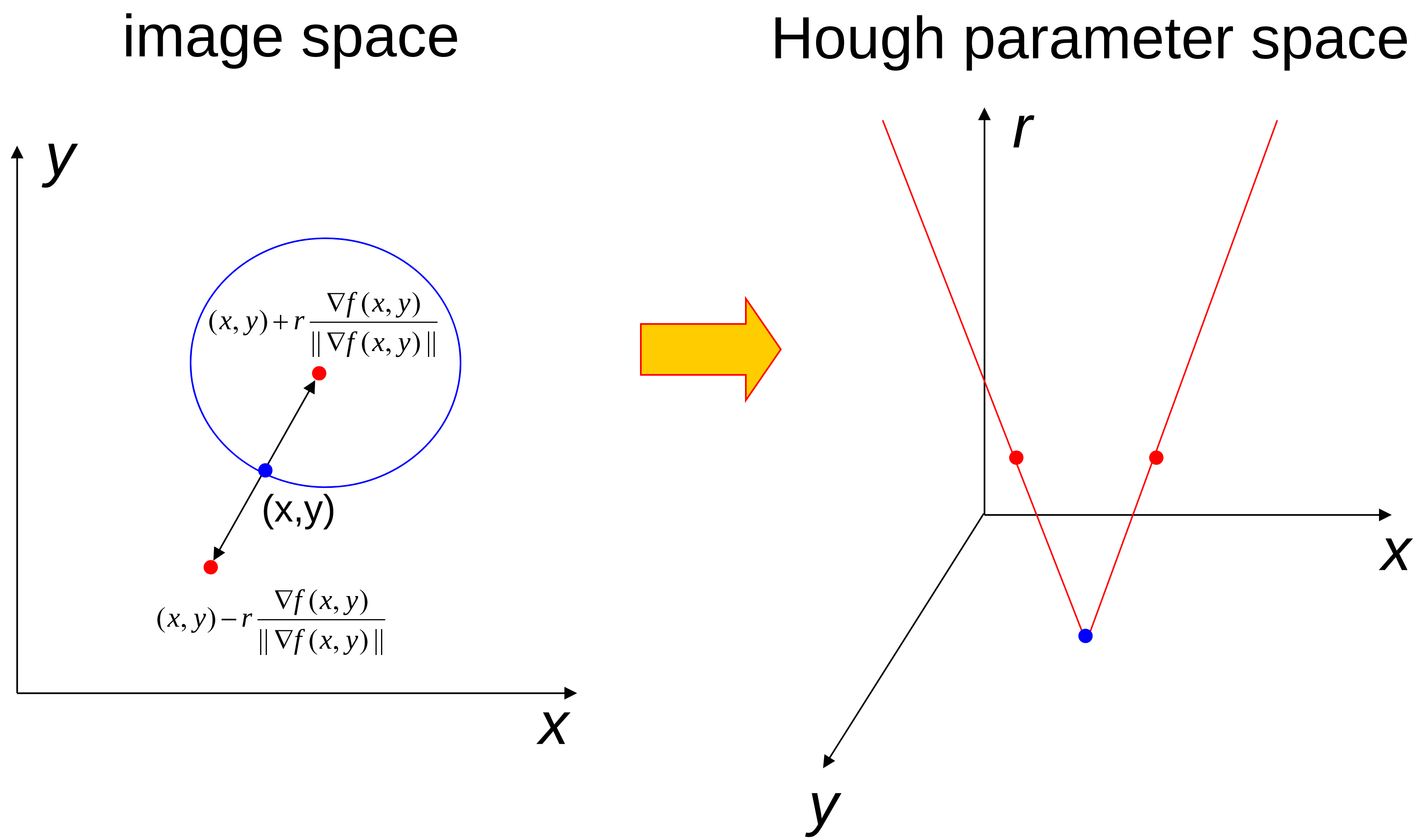


$$a = x - r\cos\phi$$

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

Need to increment only one point in Accumulator!!, Assuming not only orientation by direction is known.

Hough transform for circles with known edge orientation



Fast Tracking using Hough Transform

http://www.lirtex.com/robotics/fast-object-tracking-robot-computer-vision/

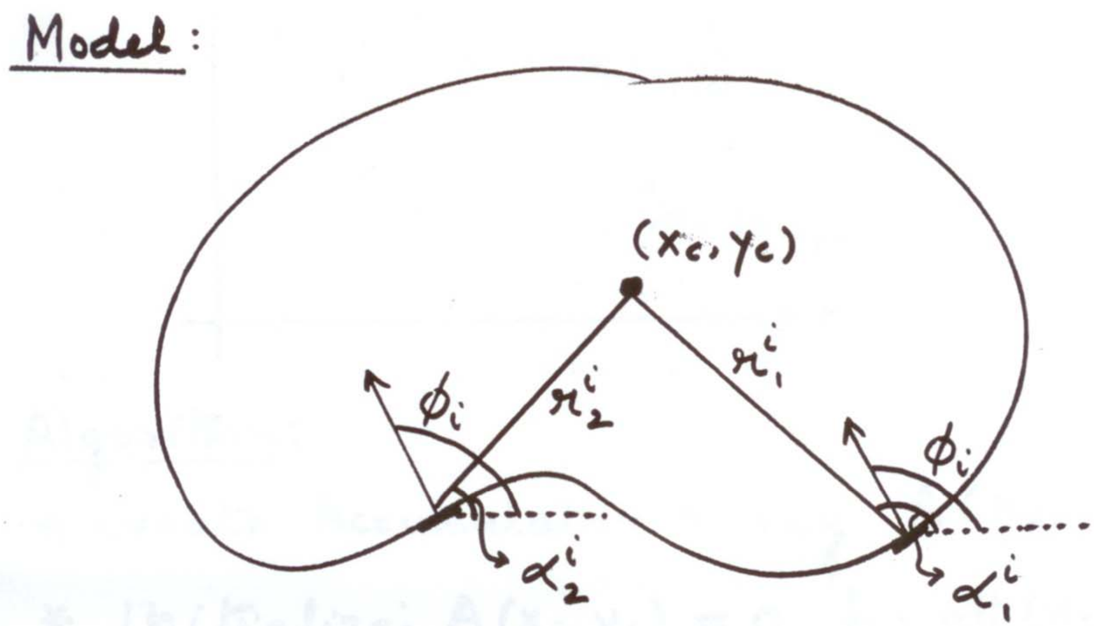
What about general objects?

complicated model: **car**

Source: K. Grauman

Generalized Hough Transform

- **Model Shape NOT described by equation but by sets of vectors from the boundary to the center.**



GENERALIZING THE HOUGH TRANSFORM TO DETECT ARBITRARY SHAPES*

D. H. BALLARD

Computer Science Department, University of Rochester, Rochester, NY 14627, U.S.A.

(*Received* 10 *October* 1979; *in revised form* 9 *September* 1980; *received for publication* 23 *September* 1980)

Generalized Hough Transform

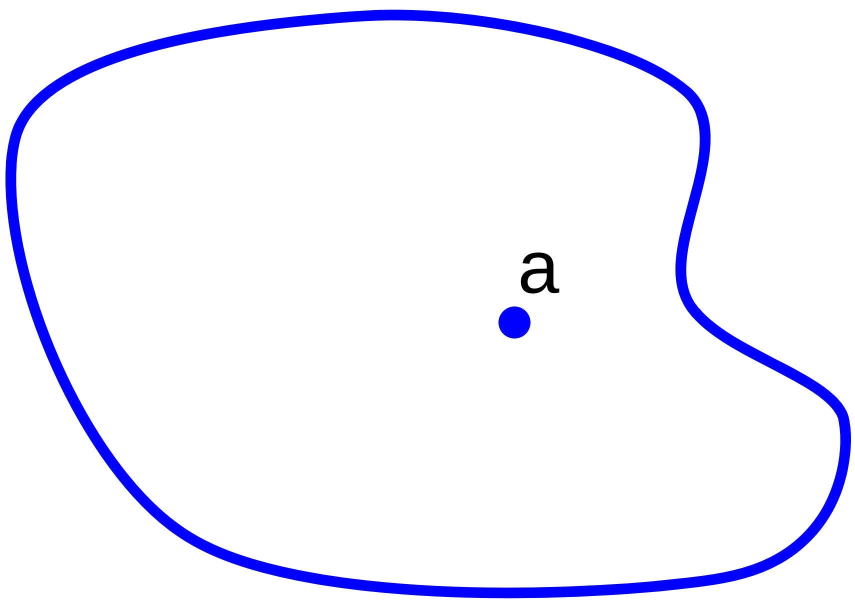
- **Model Shape NOT described by equation but by sets of vectors from the boundary to the center, sorted by edge orientation.**

ϕ - Table

Edge Direction	$\bar{r} = (r, \alpha)$
ϕ_1	$\bar{r}^1_1, \bar{r}^1_2, \bar{r}^1_3$
ϕ_2	$\bar{r}^2_1, \bar{r}^2_2$
$\vdots$	$\vdots$
ϕ_i	$\bar{r}^i_1 \; \vdots \; \bar{r}^i_2$
$\vdots$	$\vdots$
ϕ_n	$\bar{r}^n_1, \bar{r}^n_2$

Generalized Hough transform

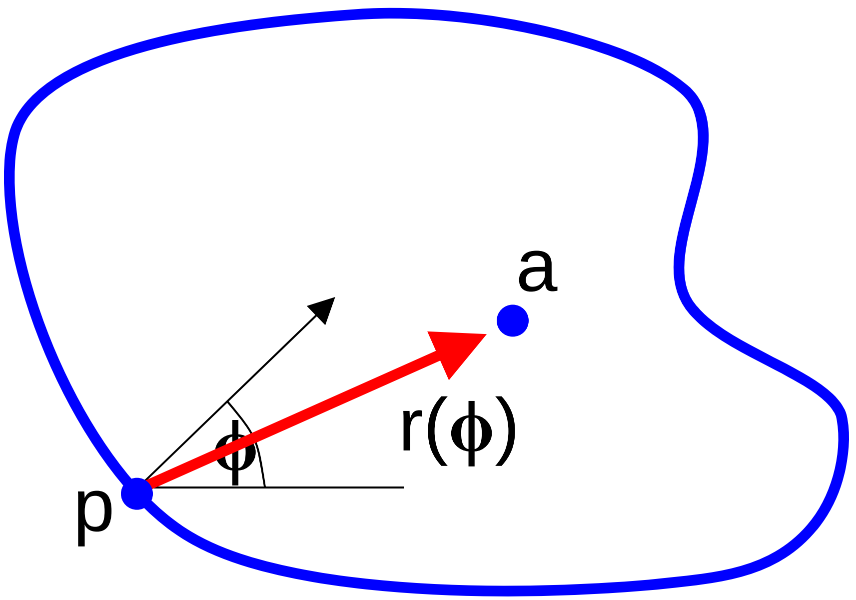
- We want to find a shape defined by its boundary points and a reference point



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

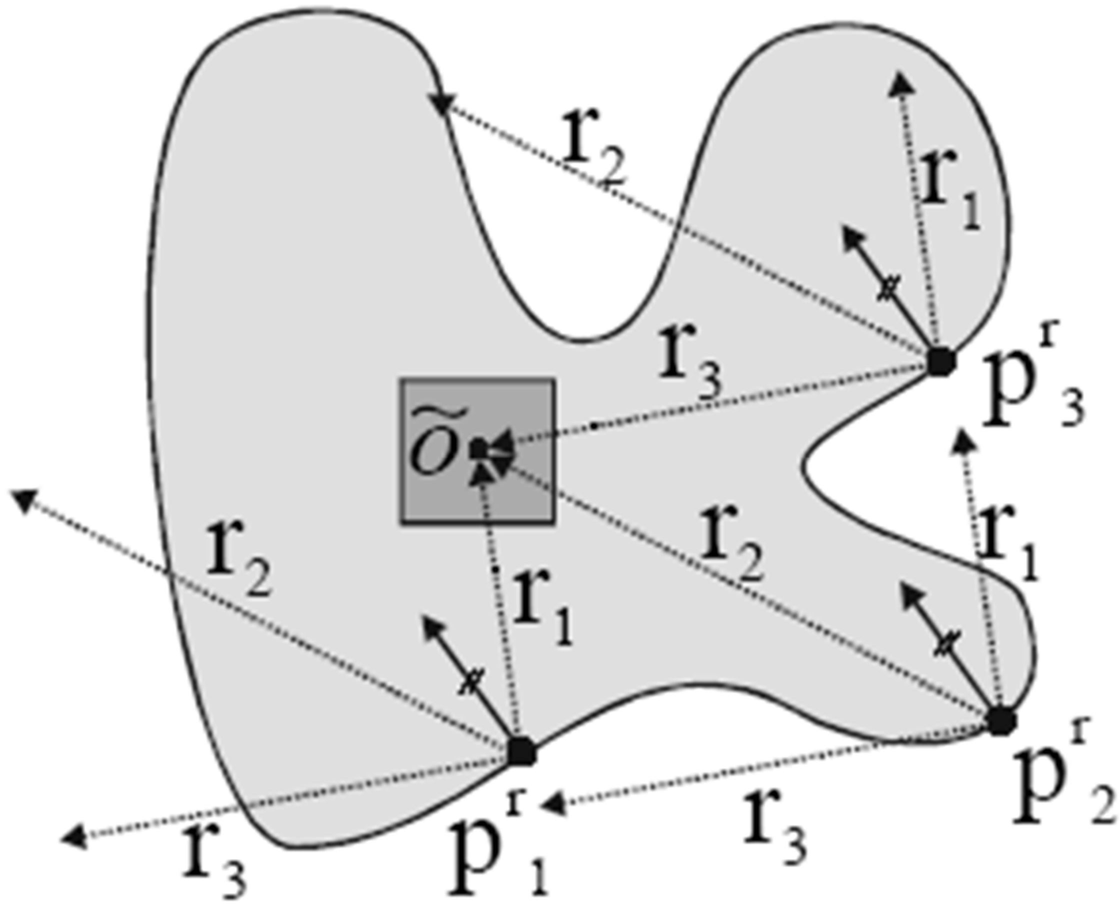
Generalized Hough transform

- We want to find a shape defined by its boundary points and a reference point
- For every boundary point p, we can compute the displacement vector r = a – p as a function of gradient orientation ϕ



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

Generalized Hough Transform



Philipp Robel

Generalized Hough Transform

Find Object Center (x_c, y_c) **given edges** (x_i, y_i, ϕ_i)

Create Accumulator Array $A(x_c, y_c)$

Initialize: $A(x_c, y_c) = 0 \quad \forall (x_c, y_c)$

For each edge point (x_i, y_i, ϕ_i)

For each entry $\overline{r}_k^i$ **in table, compute:**

$$x_c = x_i + r_k^i \cos \alpha_k^i$$

$$y_c = y_i + r_k^i \sin \alpha_k^i$$

Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$

Find Local Maxima in $A(x_c, y_c)$

Generalized Hough transform

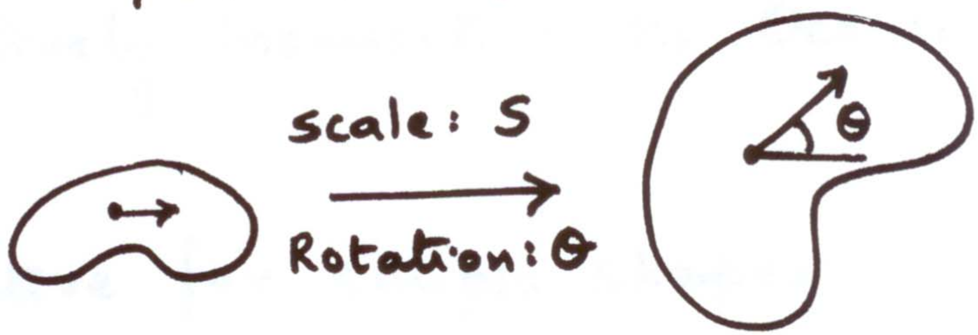
- For model shape: construct a table storing displacement vectors r as function of gradient direction
- Detection: For each edge point *p* with gradient orientation ϕ :
  - Retrieve all *r* indexed with ϕ
  - For each $r(\phi)$, put a vote in the Hough space at $p + r(\phi)$
- Peak in this Hough space is reference point with most supporting edges
- ***For orientation and scaling****: "Transform" table by updating edge orientation index and vectors, then repeat procedure as above.*

Modified from: K. Grauman

Scale & Rotation:

Use Accumulator Array:

$$A[x_c, y_c, S, \theta]$$



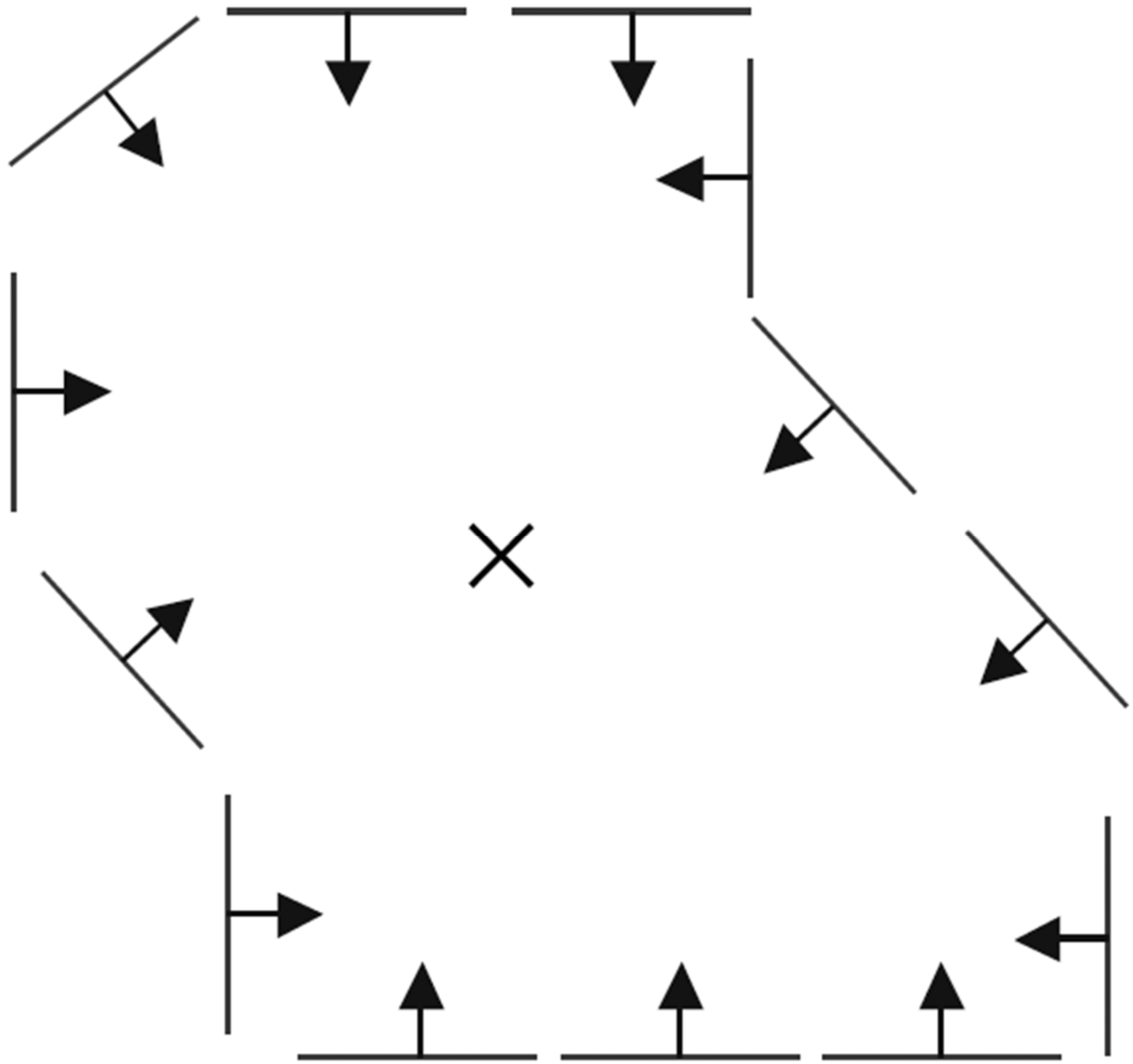
Use:

$$x_c = x_i + r^i_k S \cos(\alpha^i_k + \theta)$$

$$y_c = y_i + r^i_k S \sin(\alpha^i_k + \theta)$$

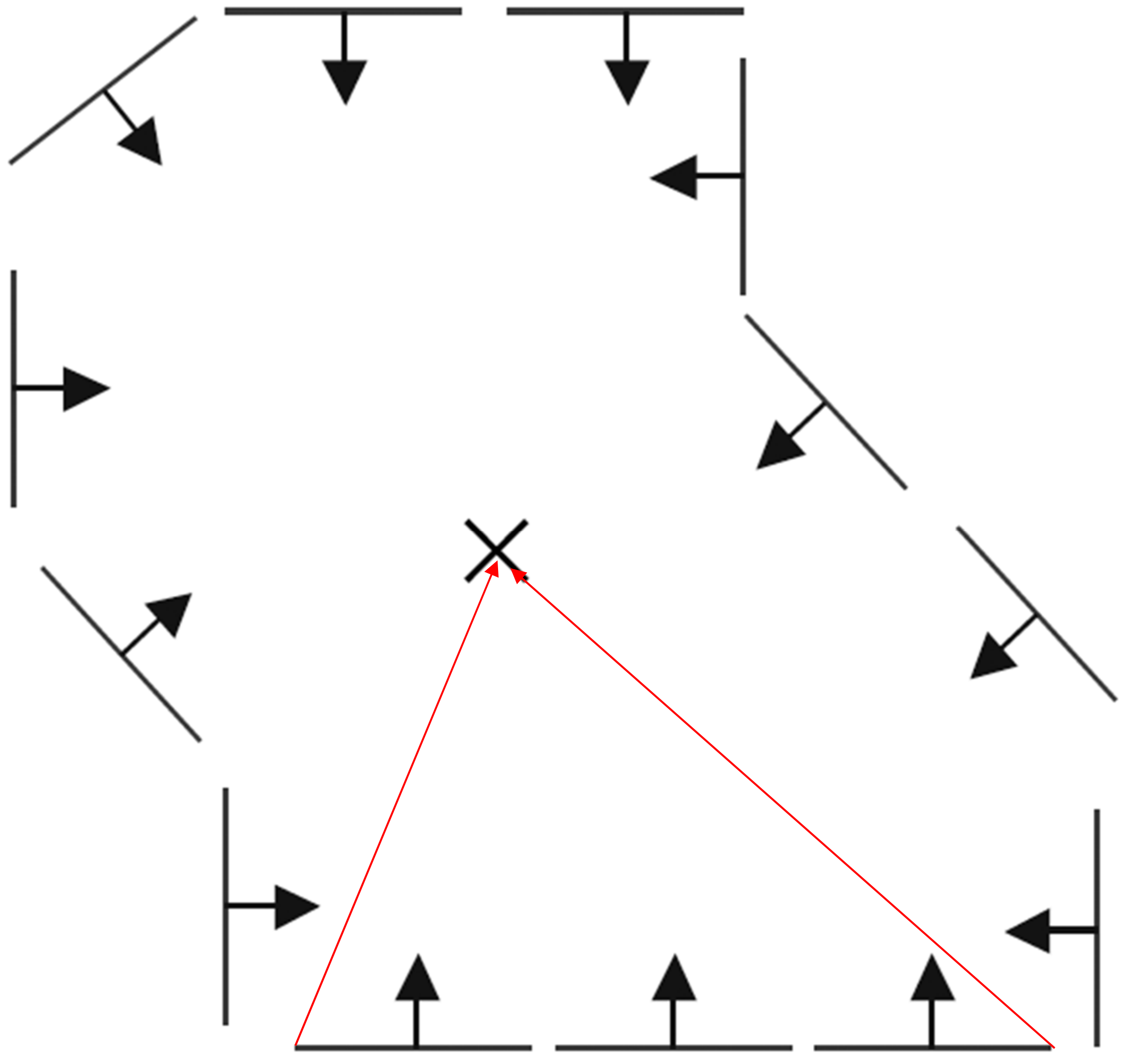
$$A(x_c, y_c, S, \theta) = A(x_c, y_c, S, \theta) + 1.$$

Example



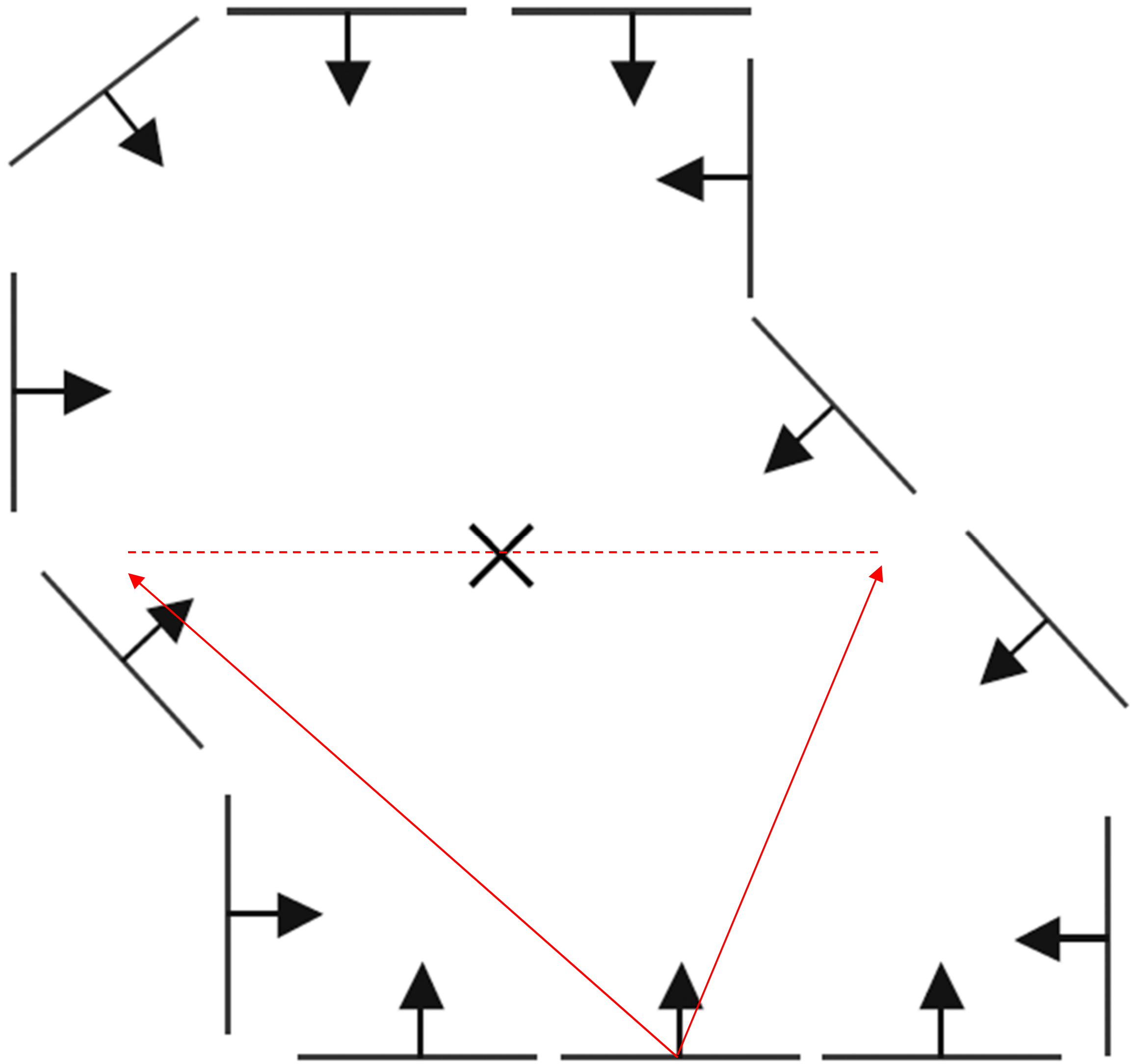
model shape

Example



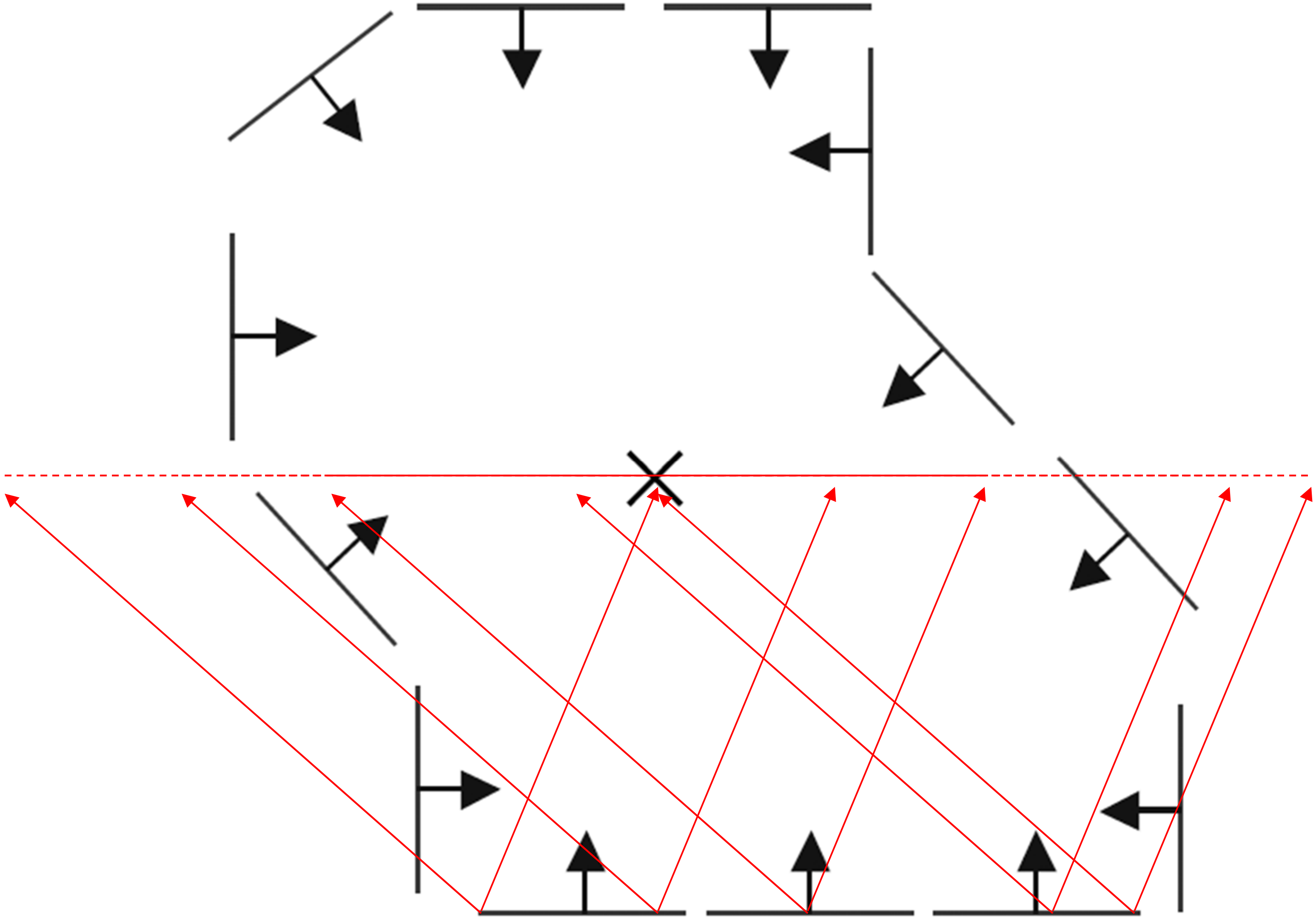
displacement vectors for model points

Example



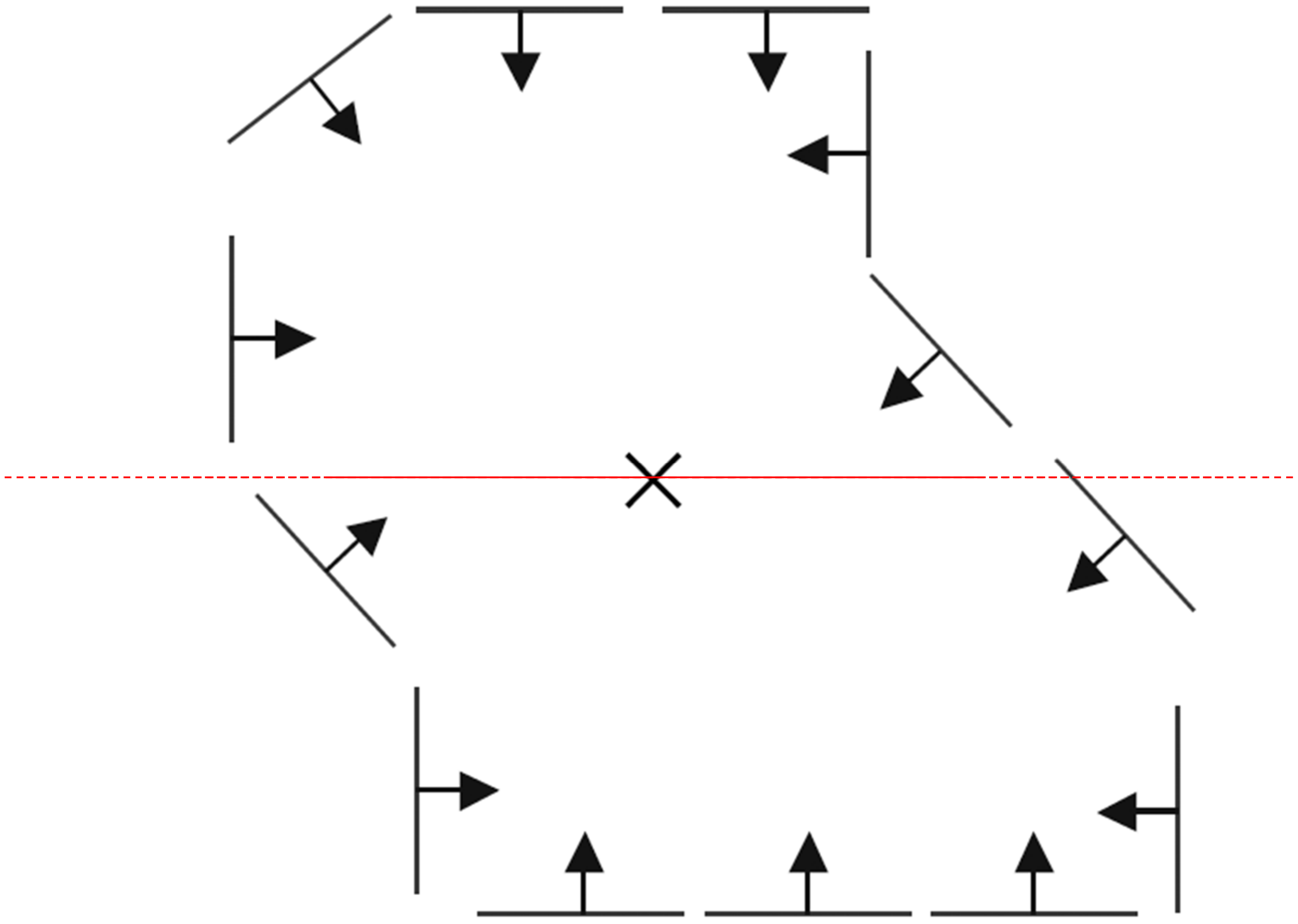
range of voting locations for test point

Example



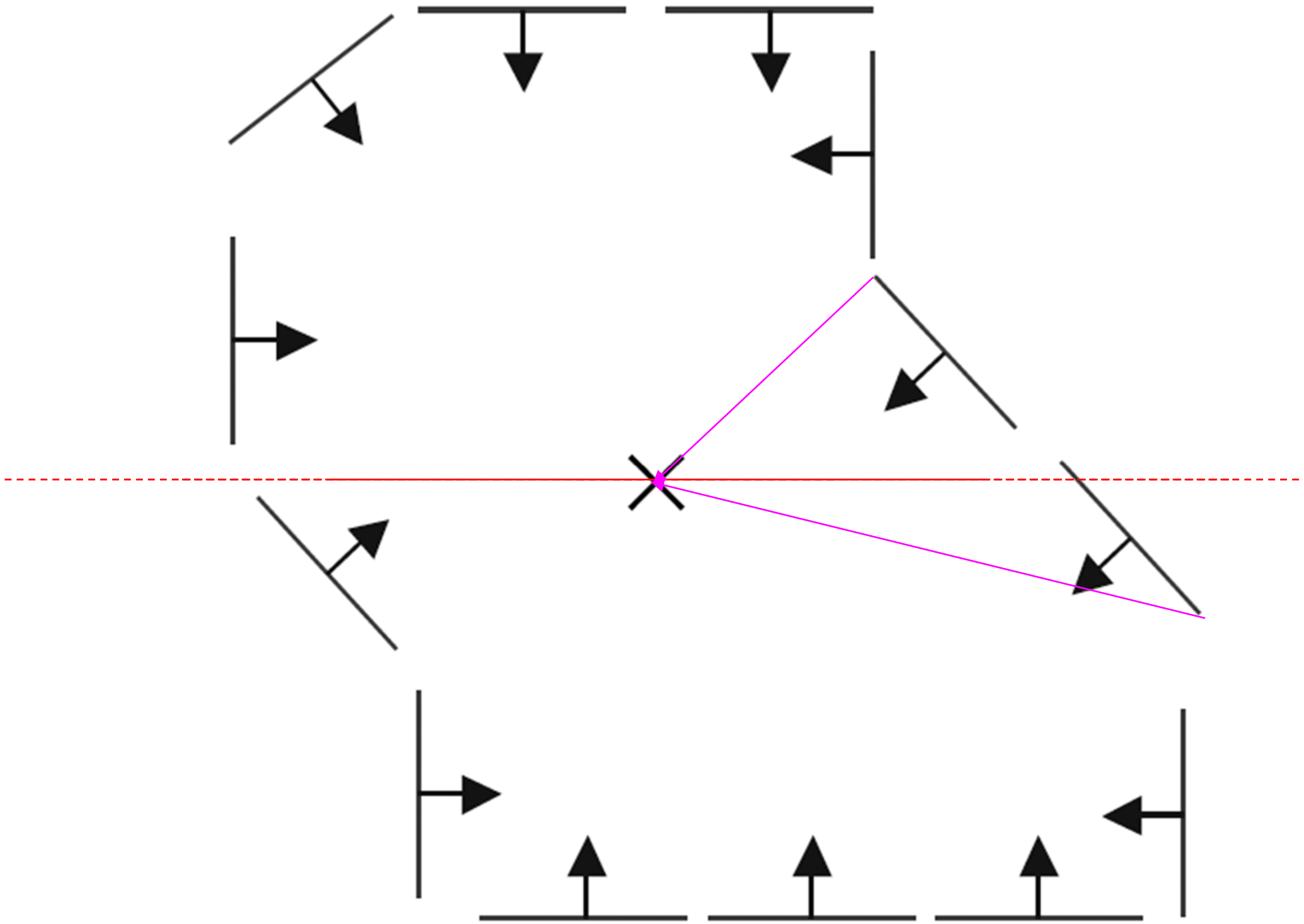
range of voting locations for test point

Example



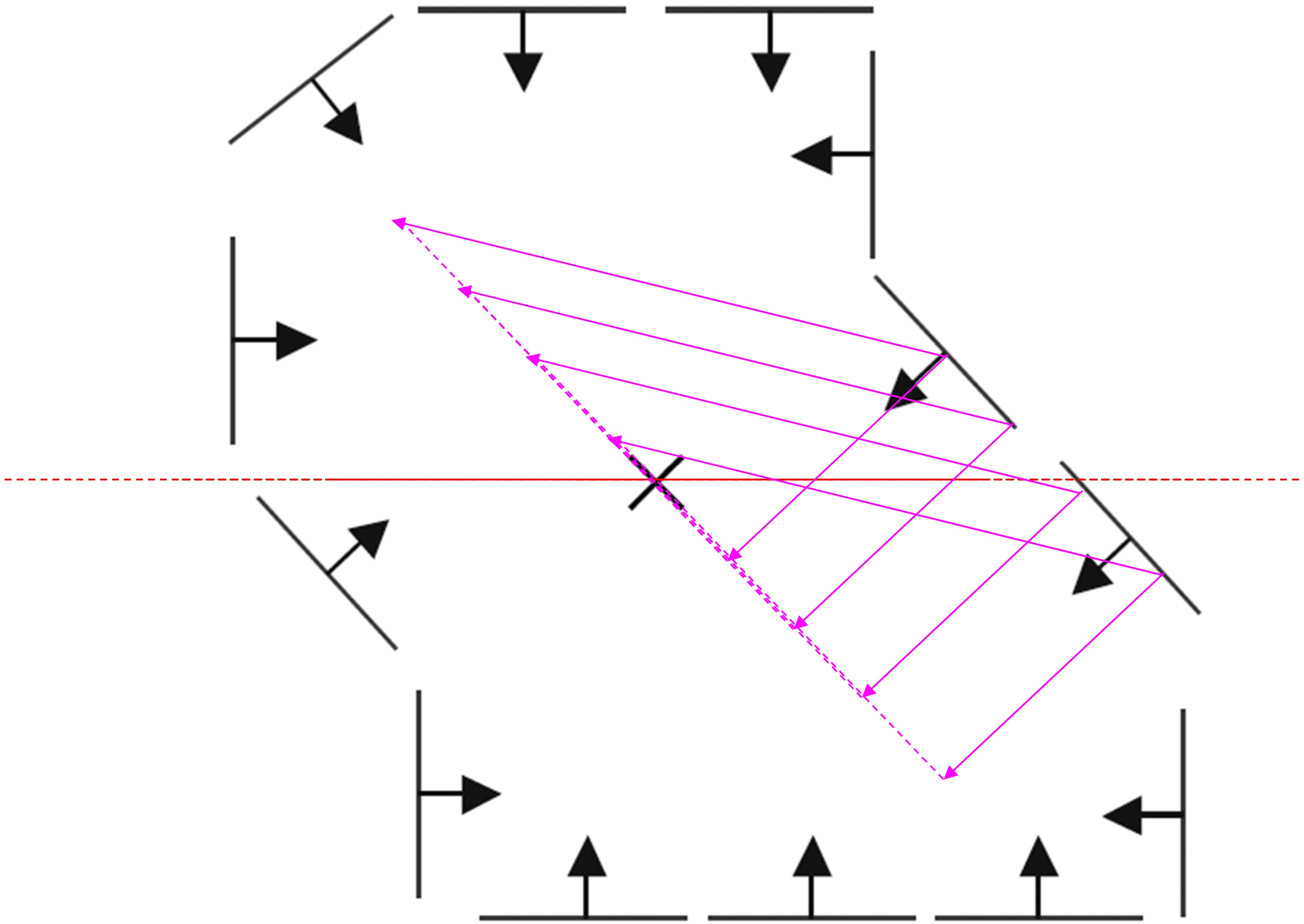
votes for points with $\theta = \uparrow$

Example



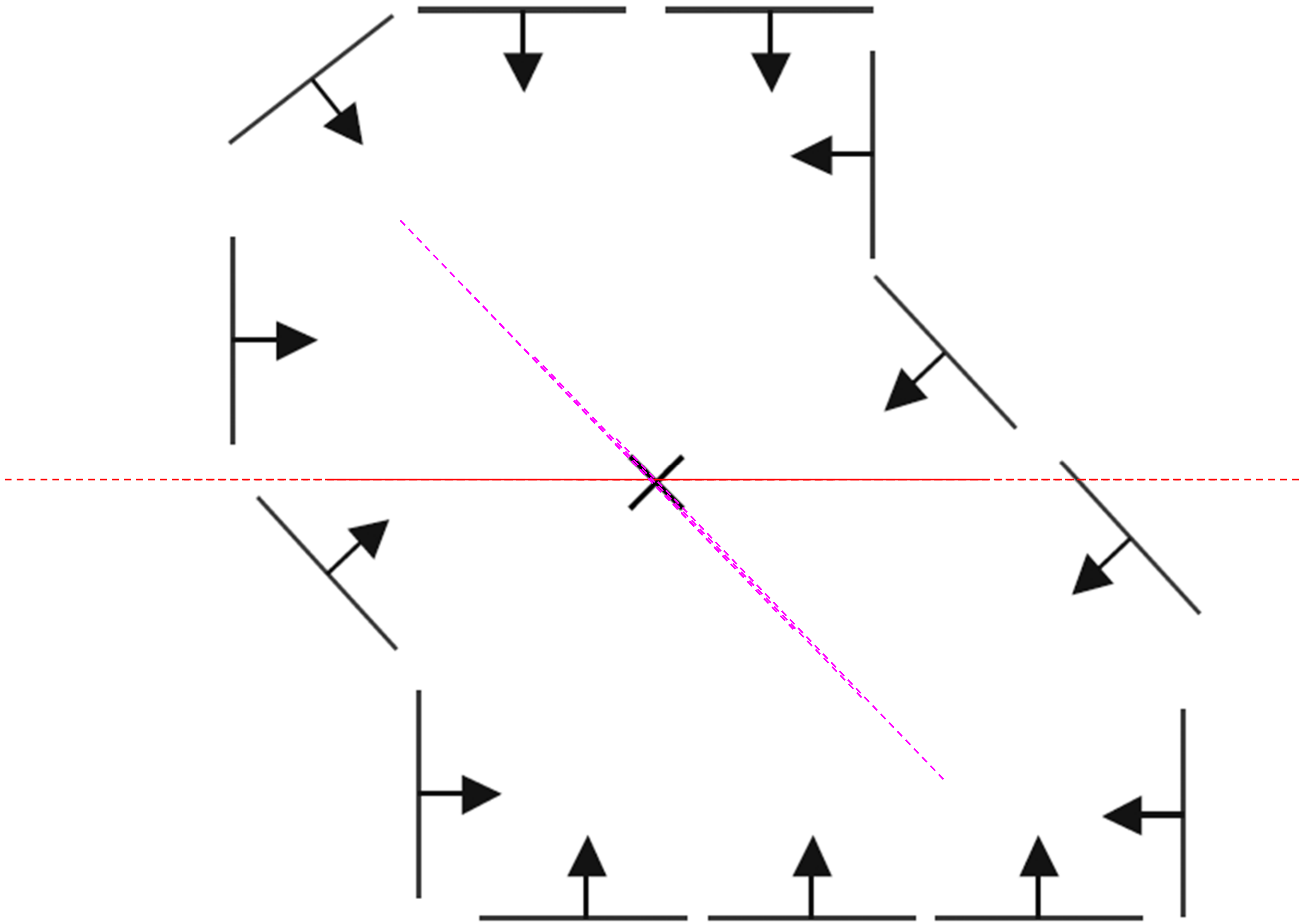
displacement vectors for model points

Example



range of voting locations for test point

Example



votes for points with $\theta = \swarrow$

Application: MRI motion correction

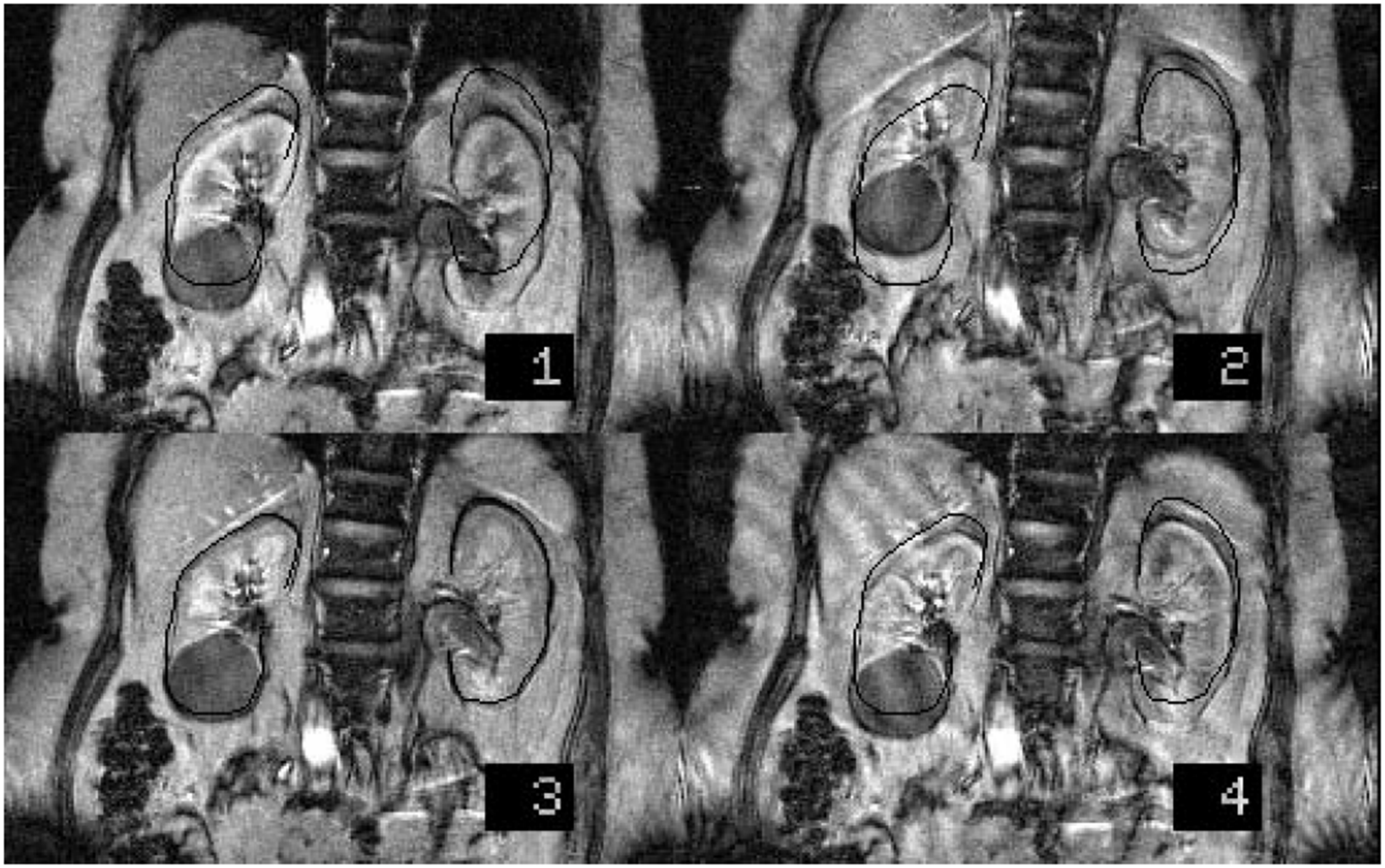
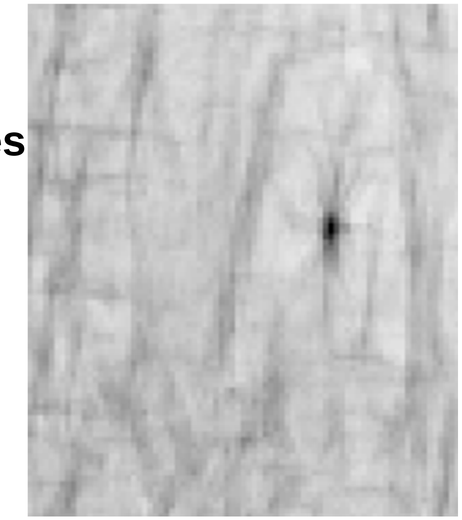


Figure 1: Original sequence of MRI scans (4 out of 64, gradient echo, TE 16.5ms, TR 30ms, flip angle 40 deg, FOV 400mm, slice thickness 10mm)

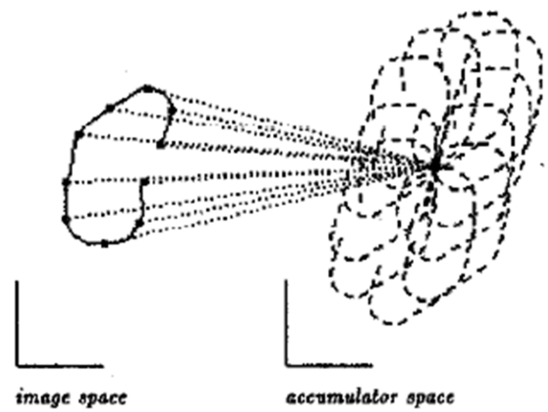
Feature Extraction and Object Detection via GHT

Object detection on feature images rather than original MRI



Personalized organ boundary template

Template matching to find motion parameters (implemented via Generalized Hough Transform)



Application: MRI motion correction

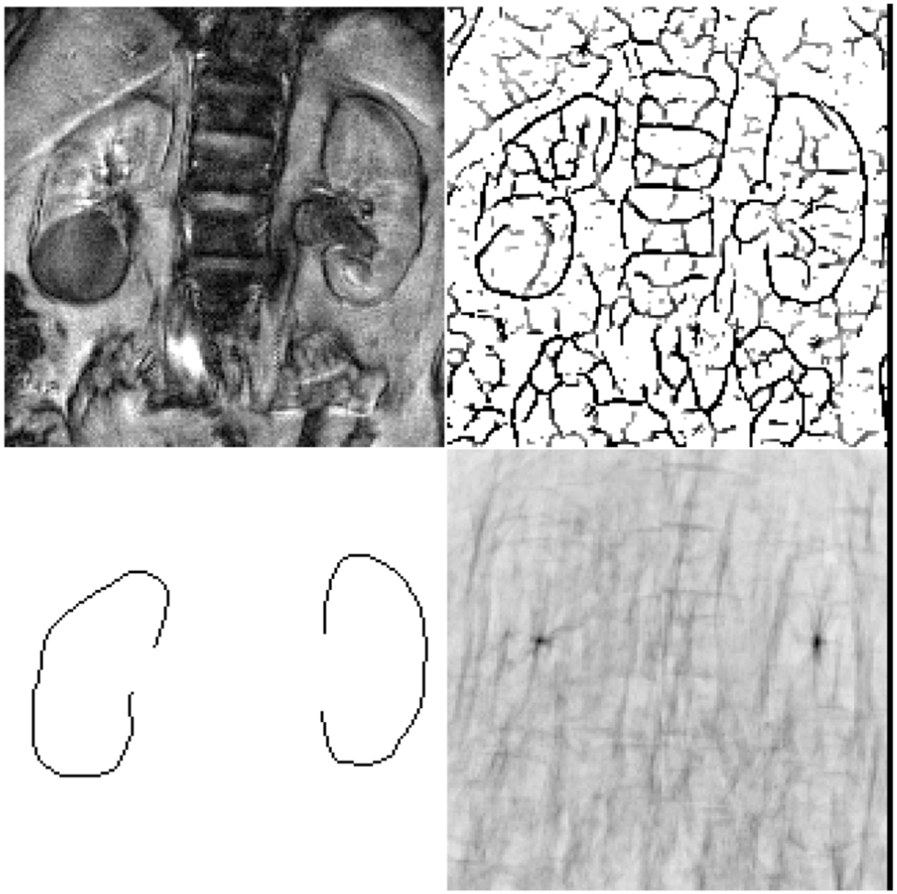
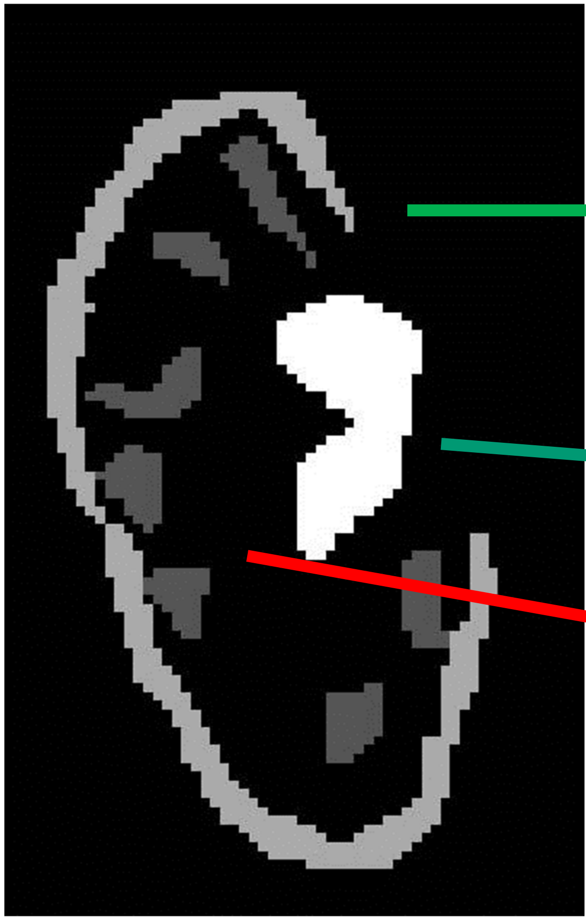
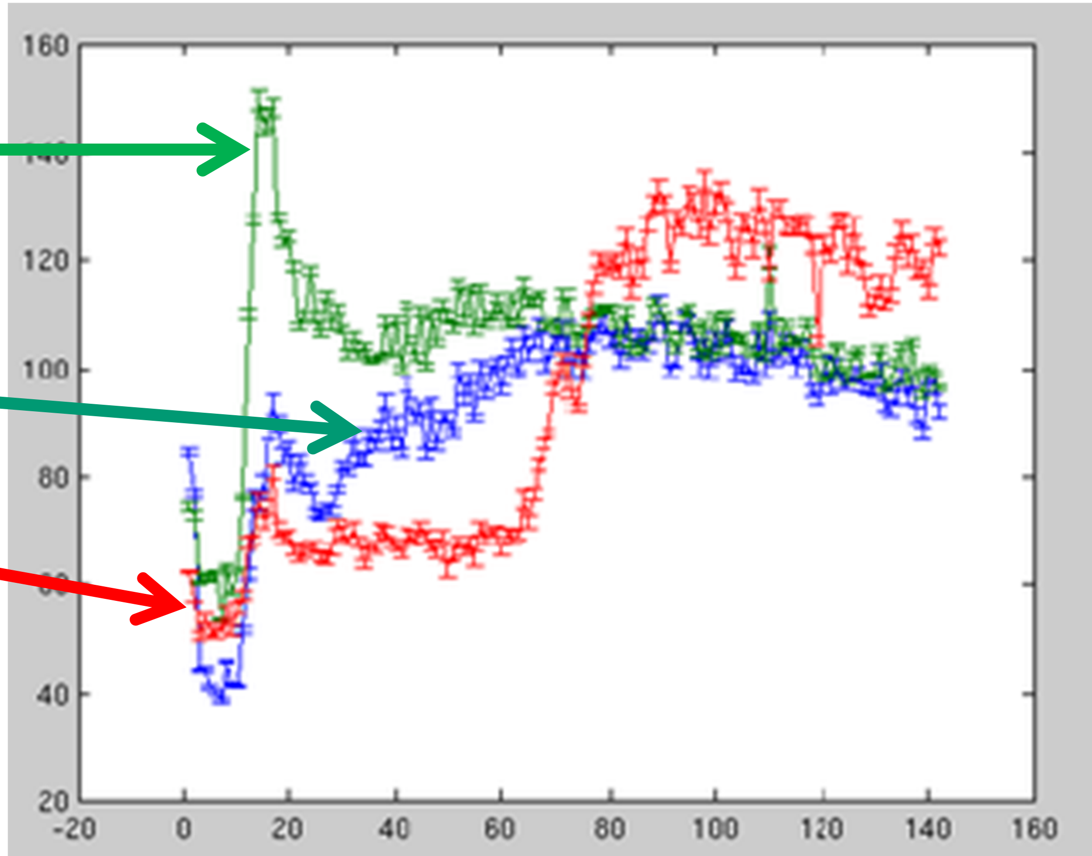


Figure 3: Readjustement of image to cover the model-curve (warp and bicubic fit)

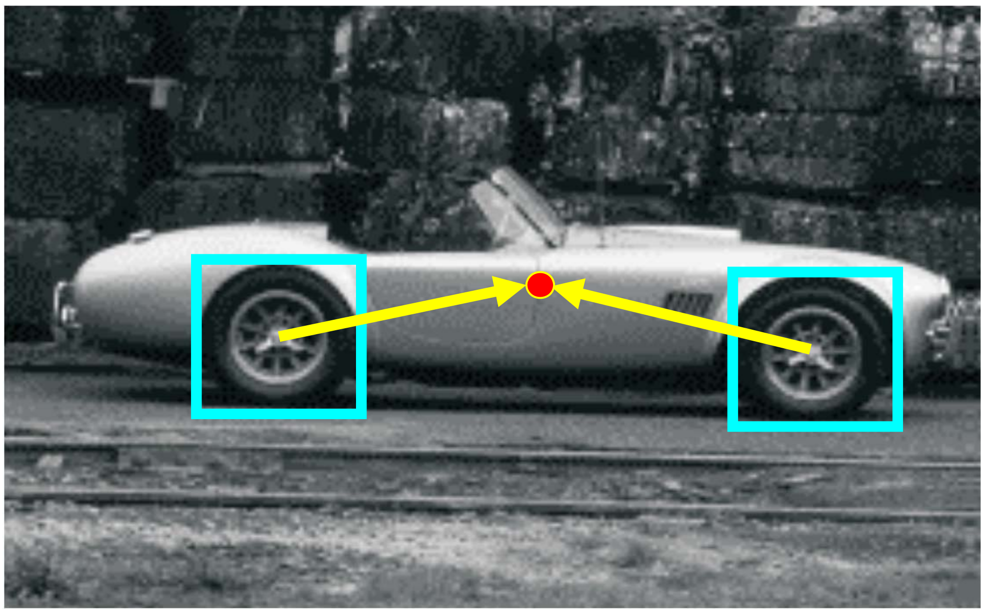
Result: Temporal Functions of Glomerular Filtration



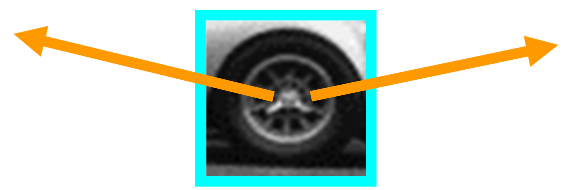


Application in recognition

- Instead of indexing displacements by gradient orientation, index by “visual codeword”



training image

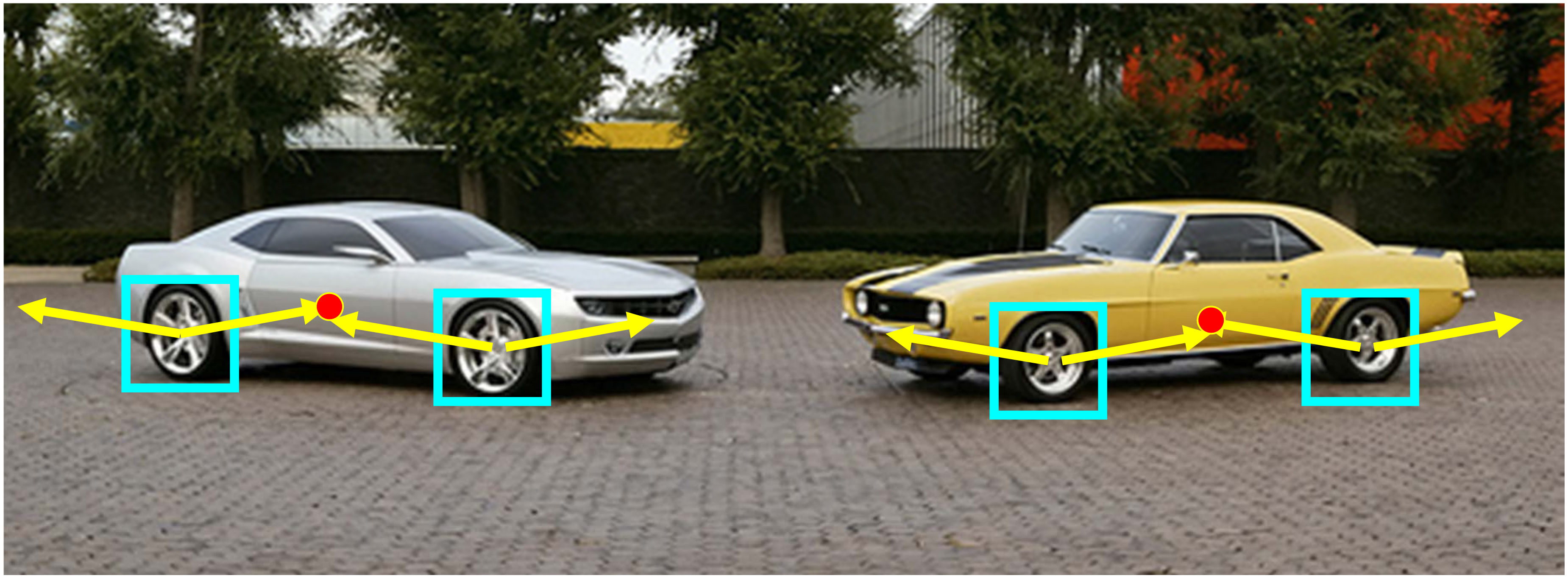


visual codeword with displacement vectors

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

- Instead of indexing displacements by gradient orientation, index by "visual codeword"

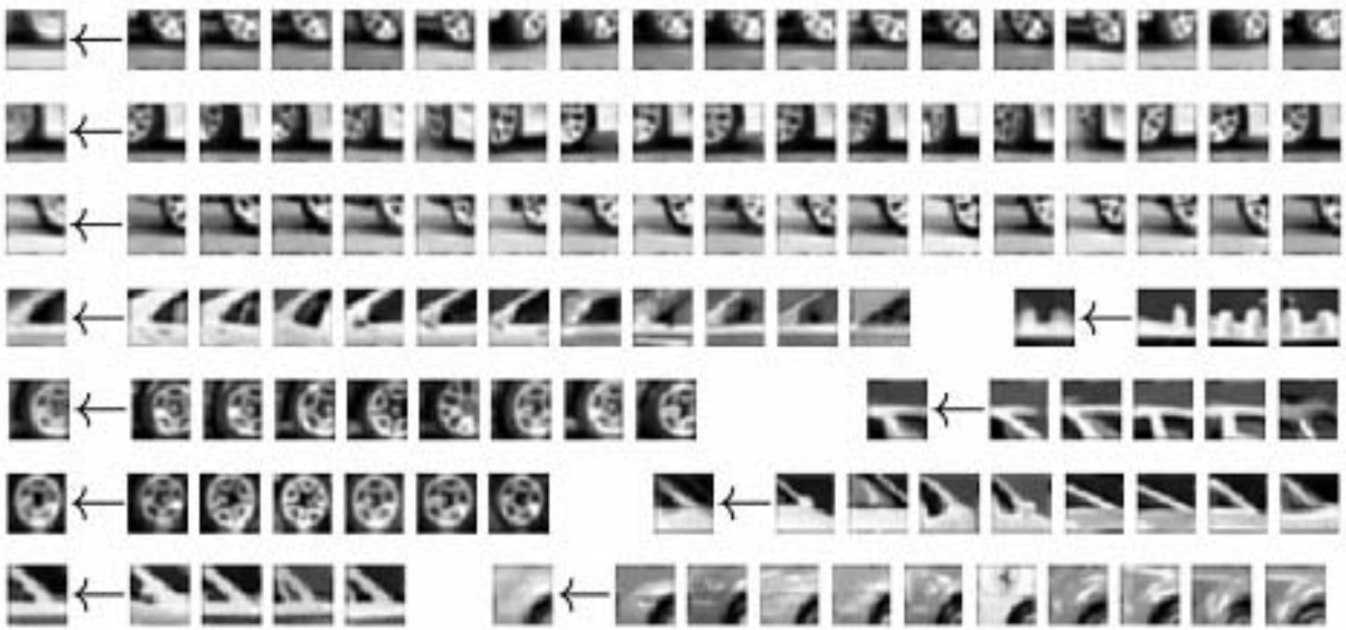


test image

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

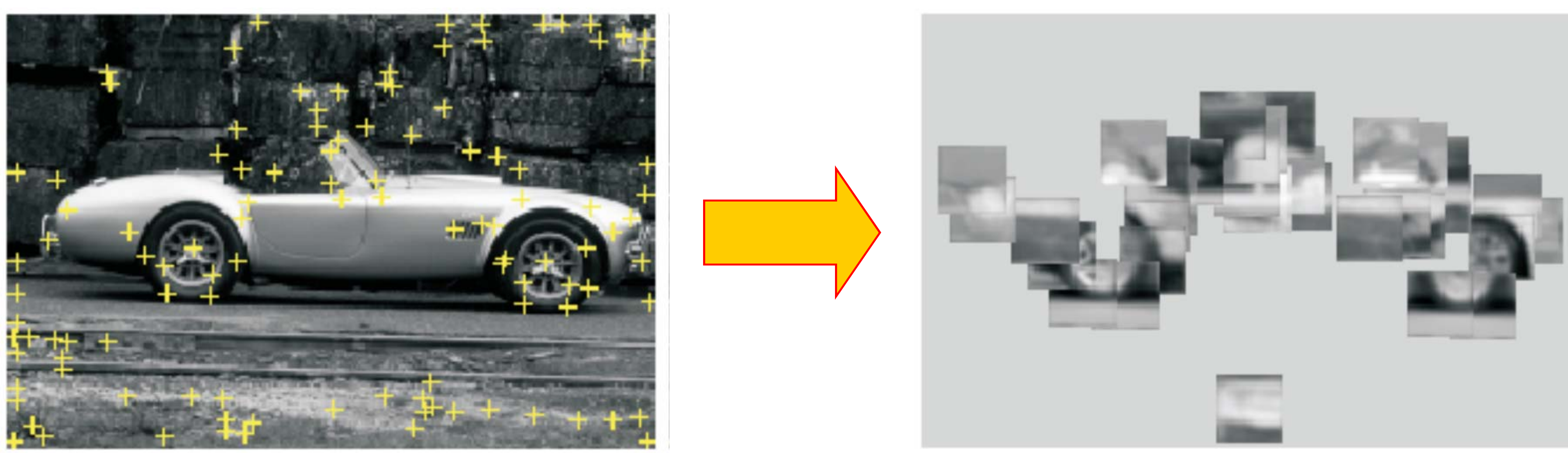
Implicit shape models: Training

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



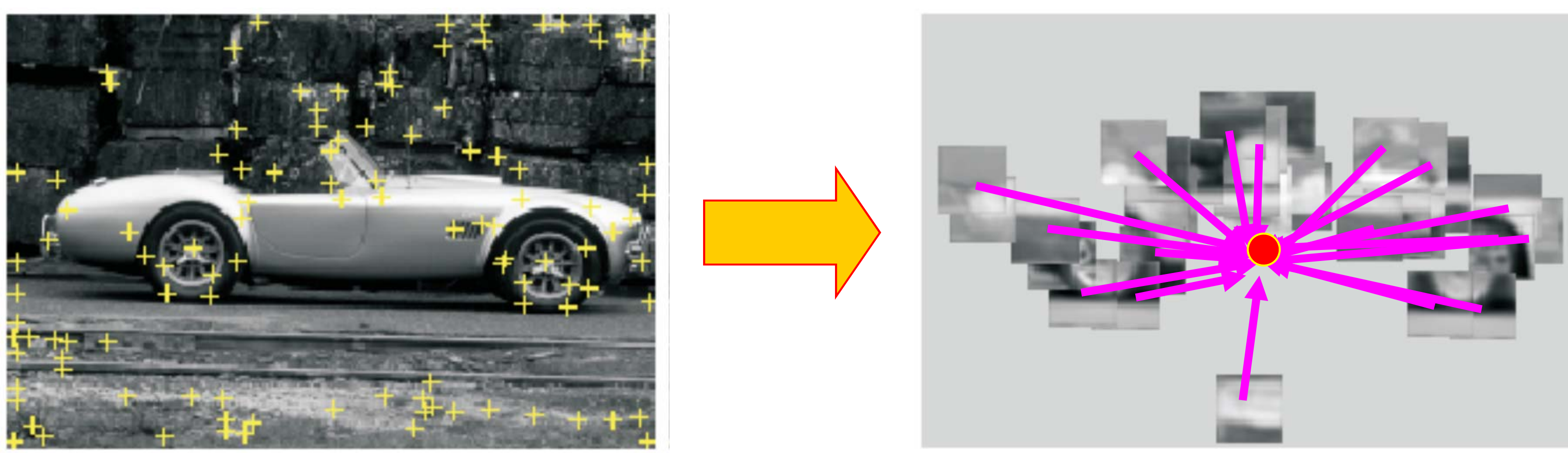
Implicit shape models: Training

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