UNIVERSITÄT BONN

Juergen Gall

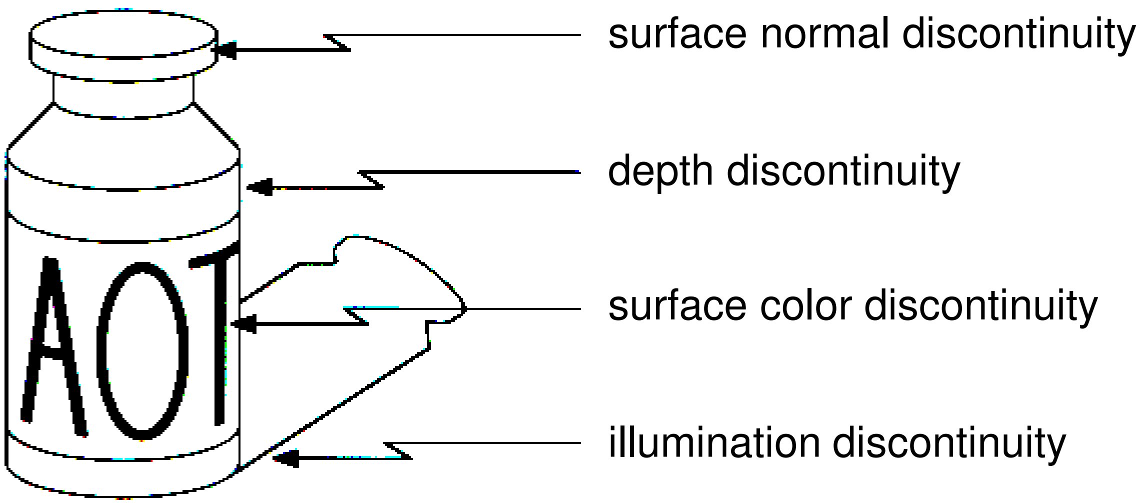
Edges and Corners

MA-INF 2201 - Computer Vision

WS24/25

Origin of edges

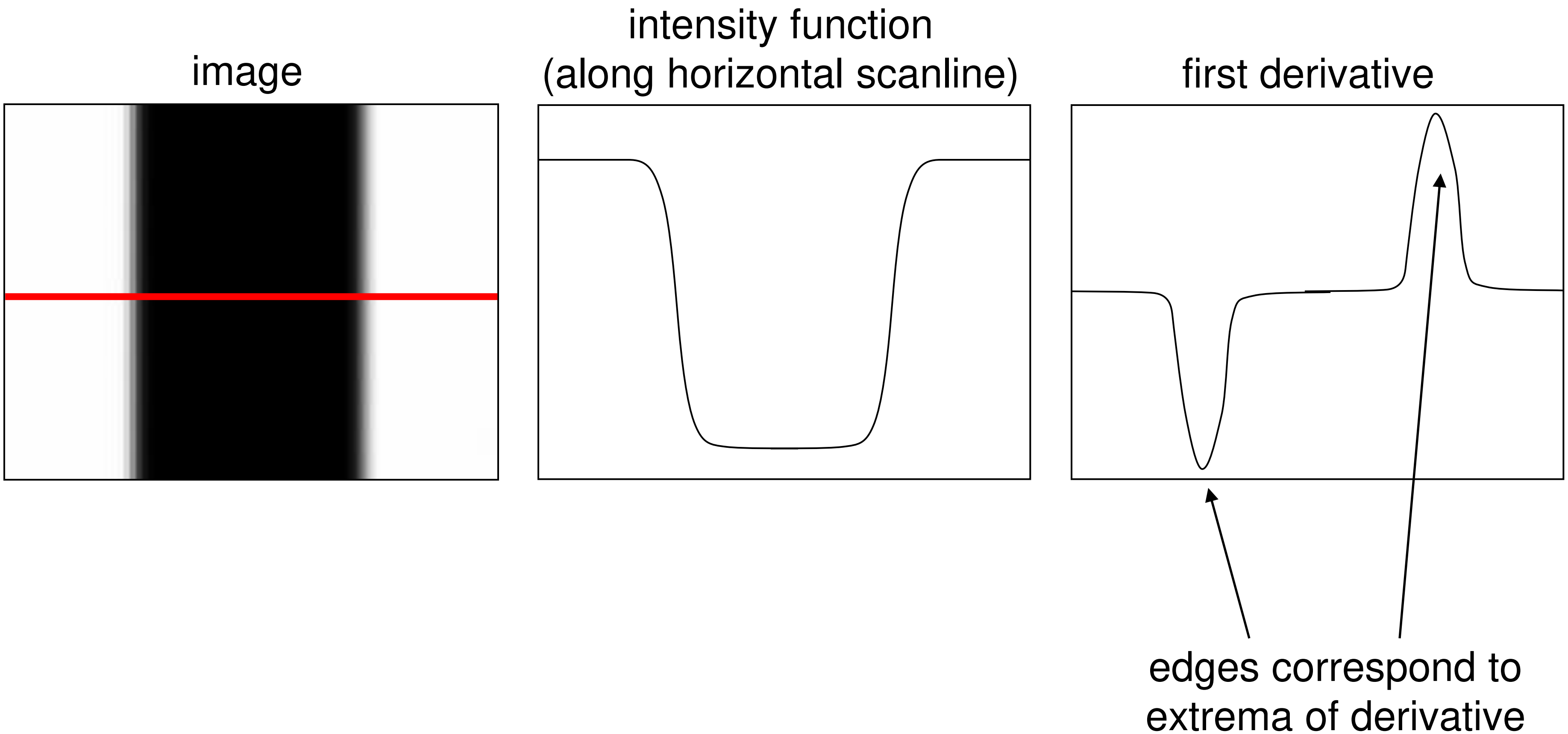
Edges are caused by a variety of factors:



Source: S. Seitz

Characterizing edges

- An edge is a place of rapid change in the image intensity function



Source: S. Lazebnik

Derivatives with convolution

For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

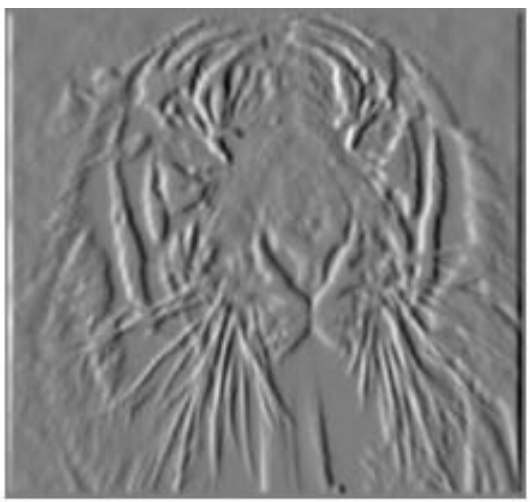
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

To implement above as convolution, what would be the associated filter?

Source: K. Grauman

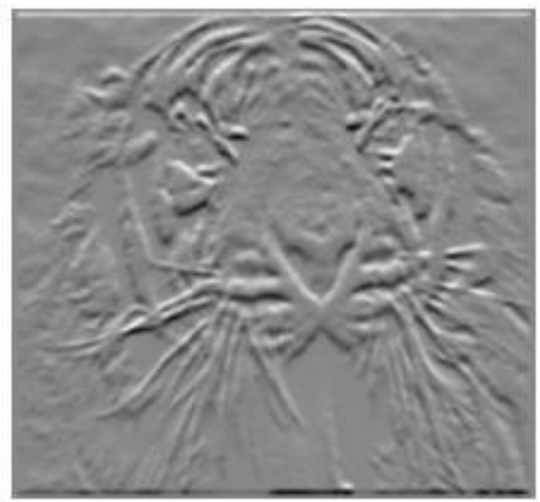
Partial derivatives of an image

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



-1	1

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



-1
1

Source: S. Lazebnik

Finite difference filters

Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline\end{array}$

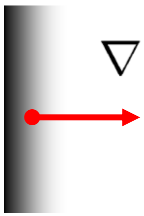
Sobel: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$

Roberts: $M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline\end{array}$; $M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array}$

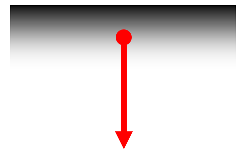
Source: K. Grauman

Image gradient

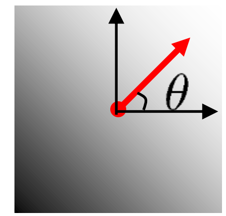
The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



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

- The gradient points in the direction of most rapid increase in intensity
- The gradient direction is given by

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

- The edge strength is given by the gradient magnitude

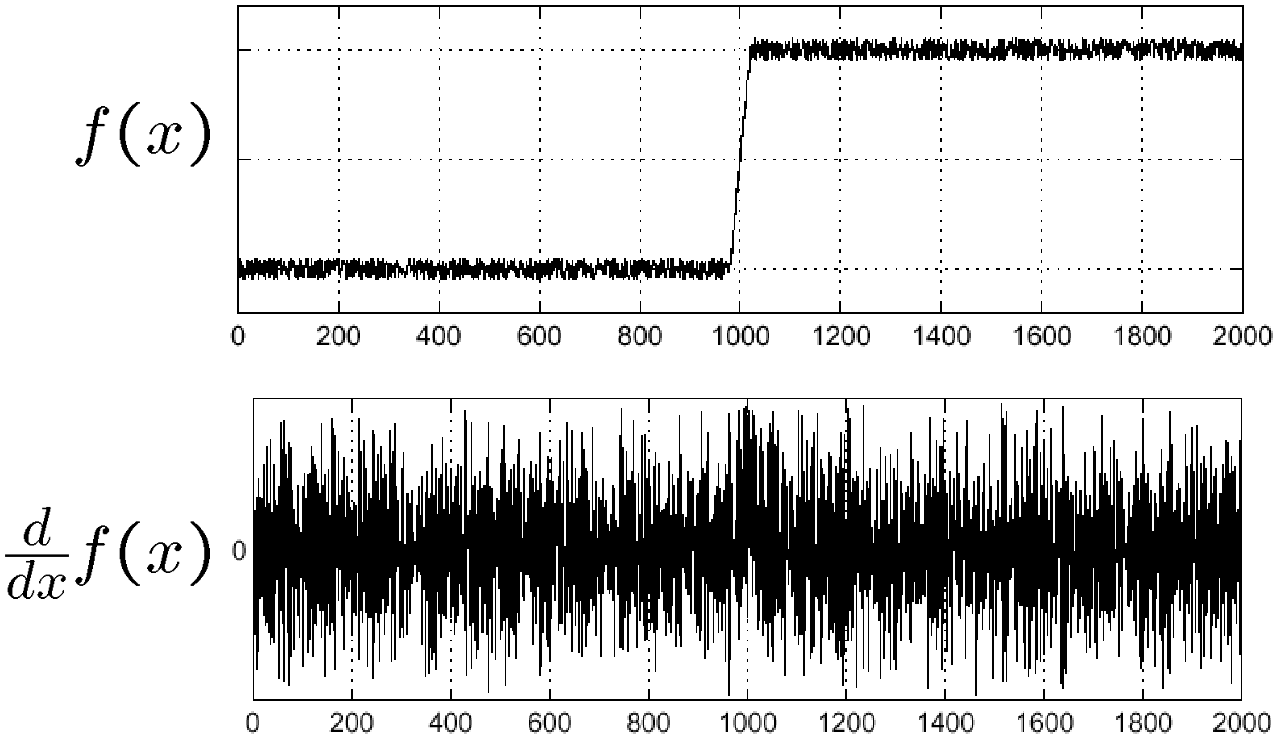
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: S. Seitz

Effects of noise

Consider a single row or column of the image

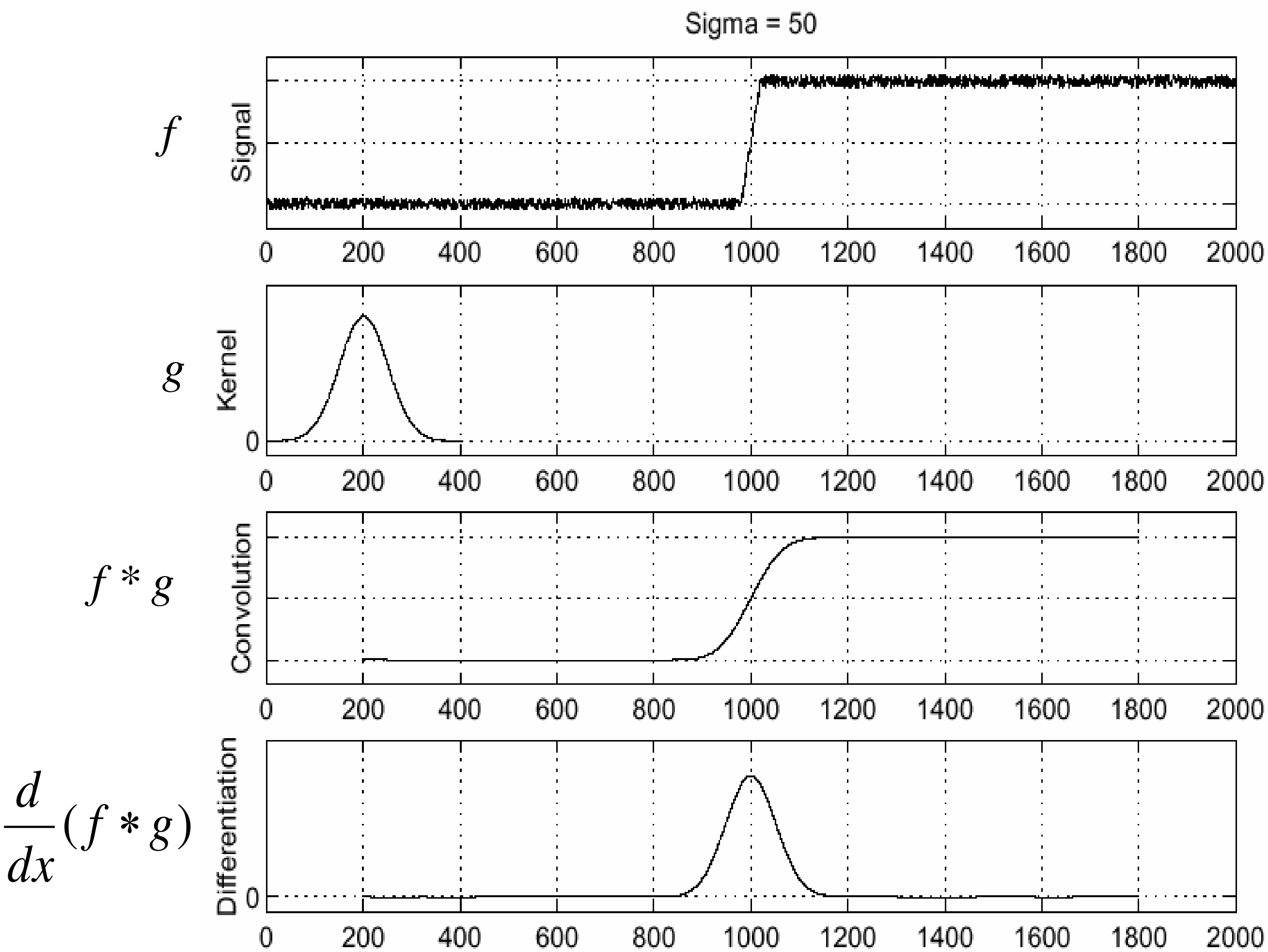
– Plotting intensity as a function of position gives a signal



Where is the edge?

Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

Derivative theorem of convolution

Differentiation is convolution, and convolution is associative:

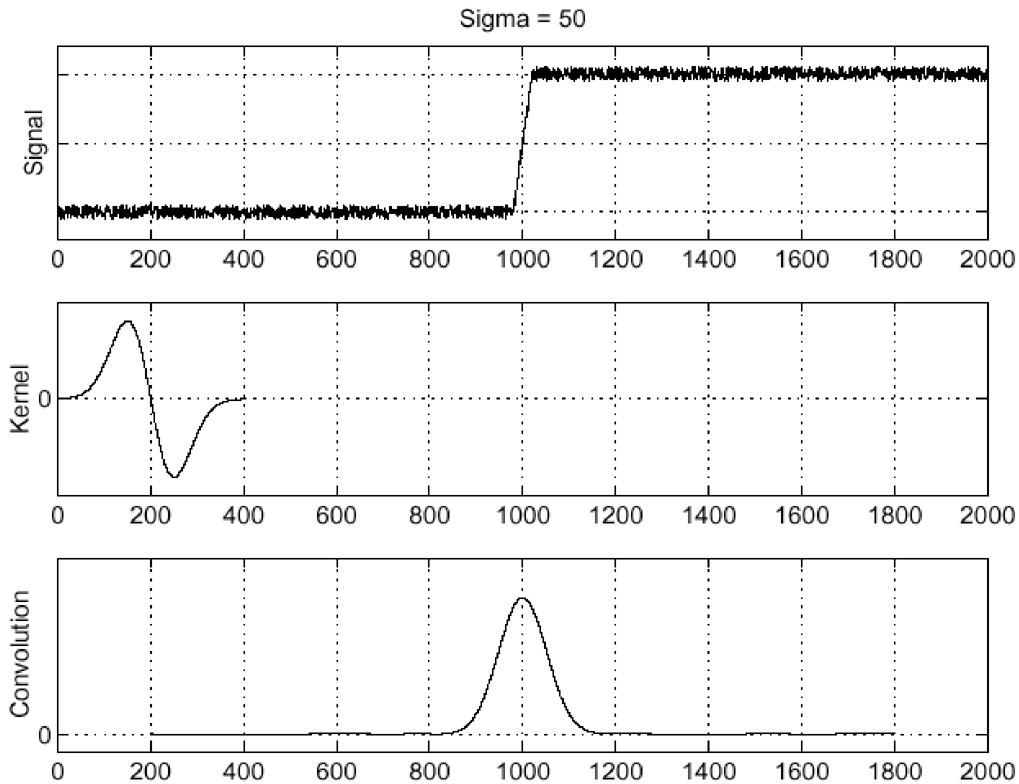
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

This saves us one operation:

f

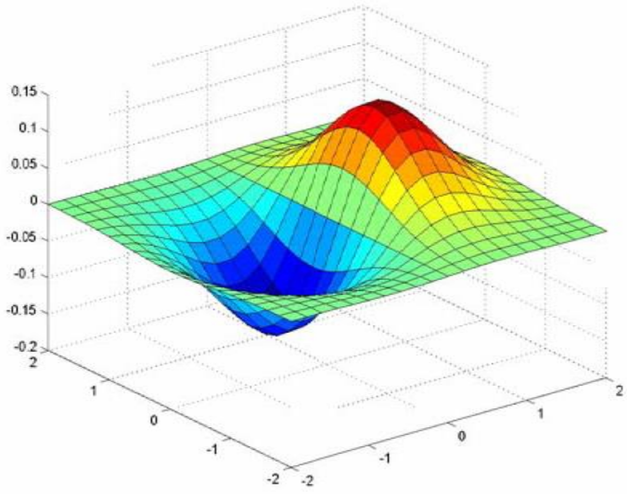
$\frac{d}{dx} g$

$f * \frac{d}{dx} g$

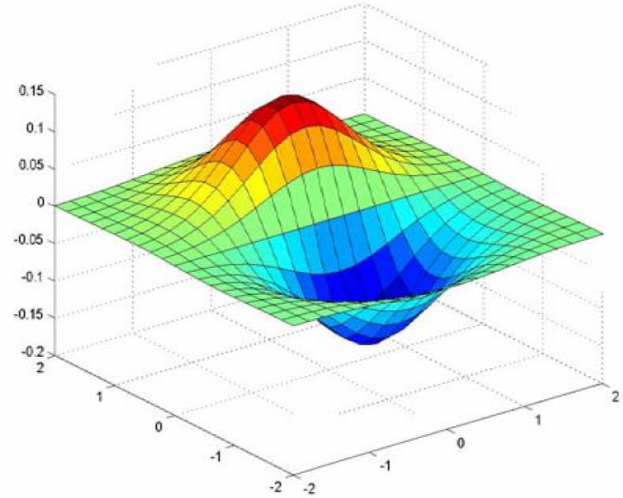


Source: S. Seitz

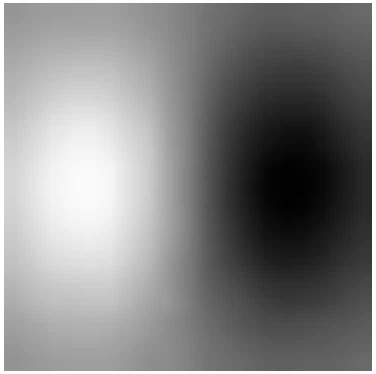
Derivative of Gaussian filter

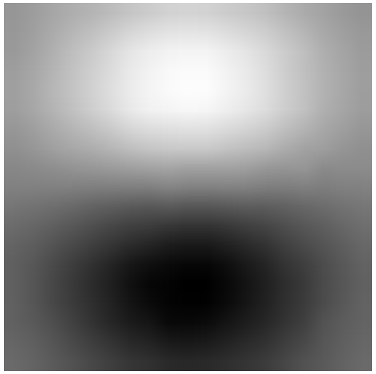


x-direction



y-direction

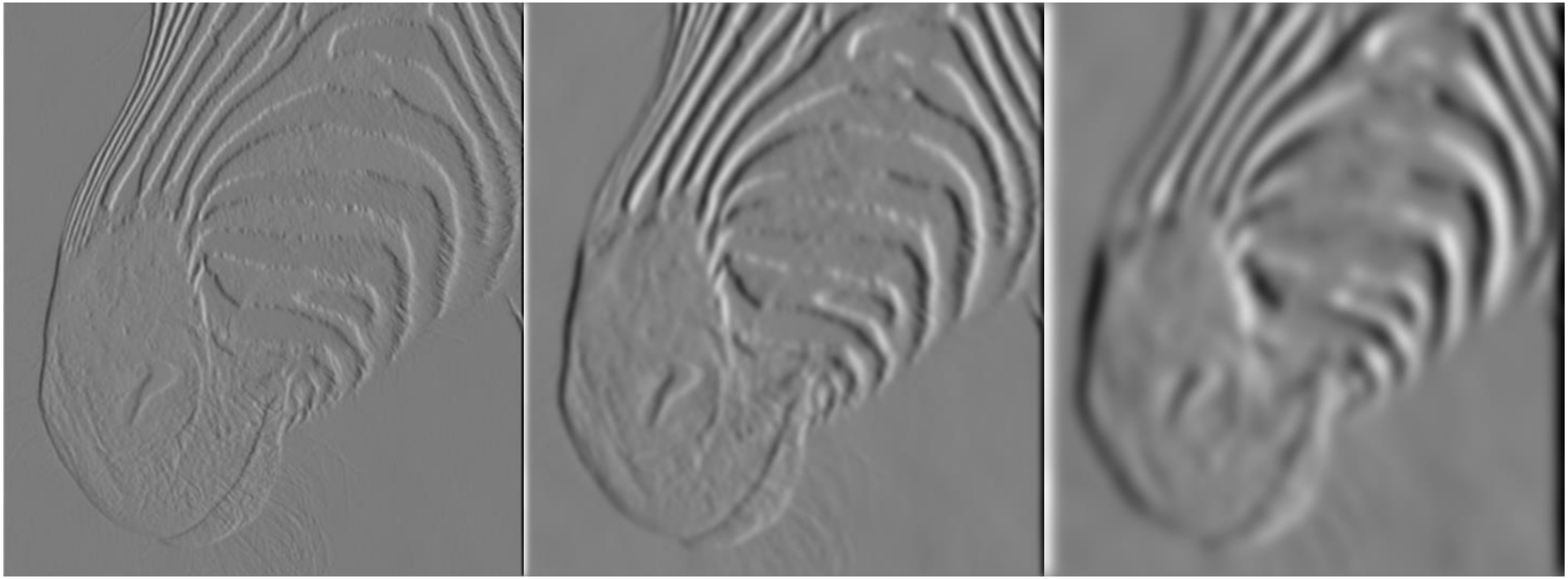




Source: S. Lazebnik

Scale of Gaussian derivative filter

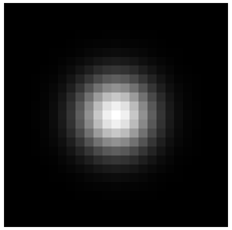
Smoothed derivative removes noise, but blurs edge.
Also finds edges at different “scales”



1 pixel | 3 pixels | 7 pixels

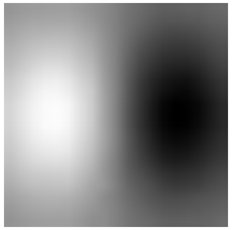
Source: D. Forsyth

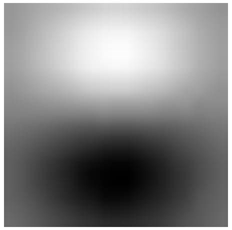
Review: Smoothing vs. derivative filters



Smoothing filters

- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
    - **One:** constant regions are not affected by the filter





Derivative filters

- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
    - **Zero:** no response in constant regions
- High absolute value at points of high contrast

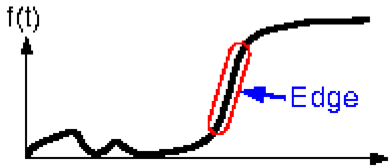
Source: S. Lazebnik

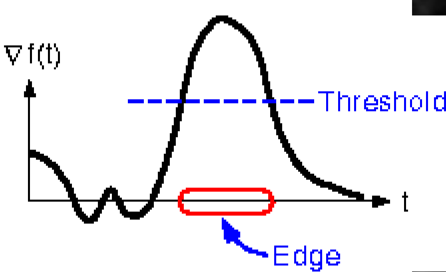
The Canny edge detector

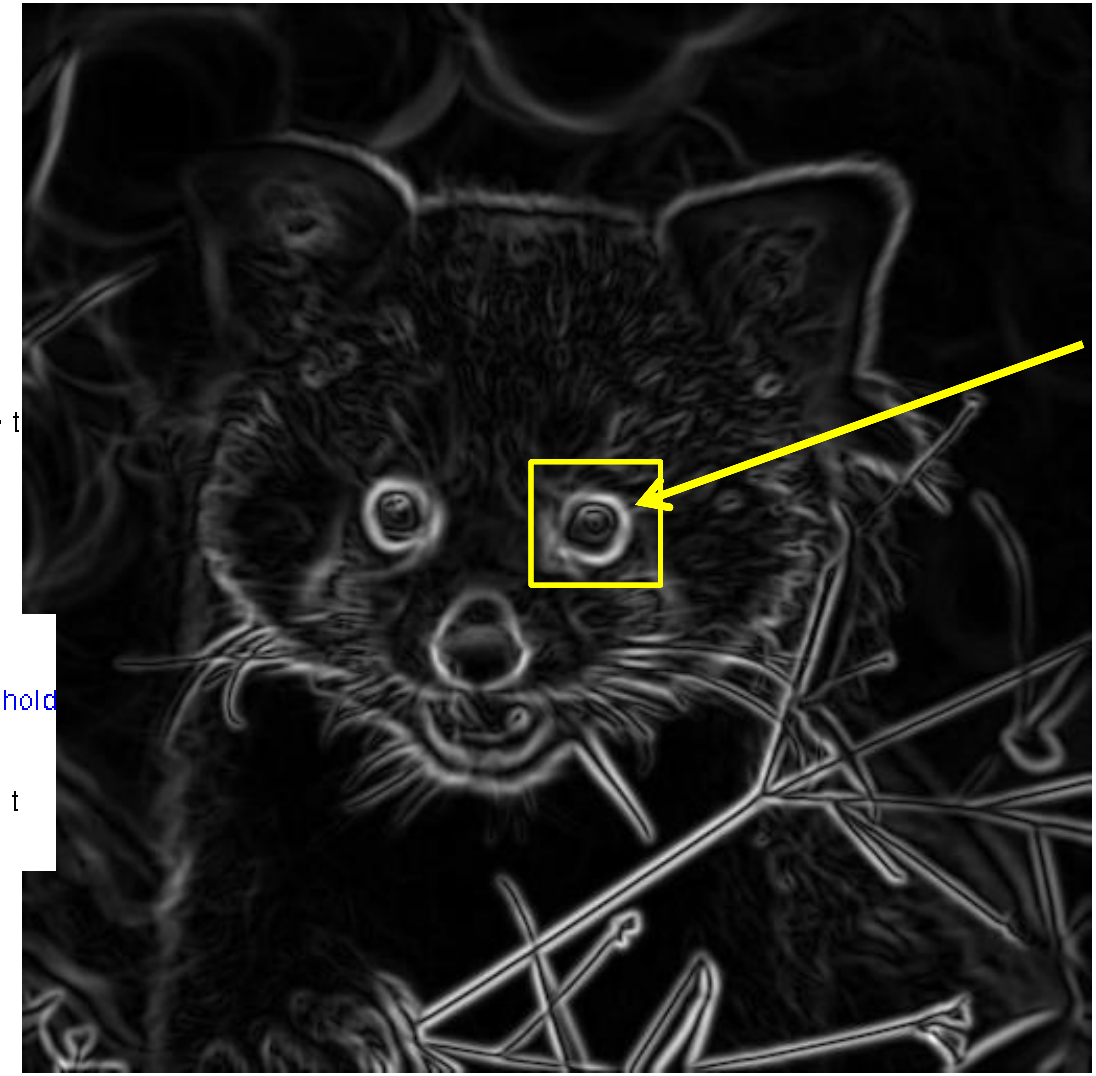
The Canny edge detector

Magnitude of gradient

The Canny edge detector







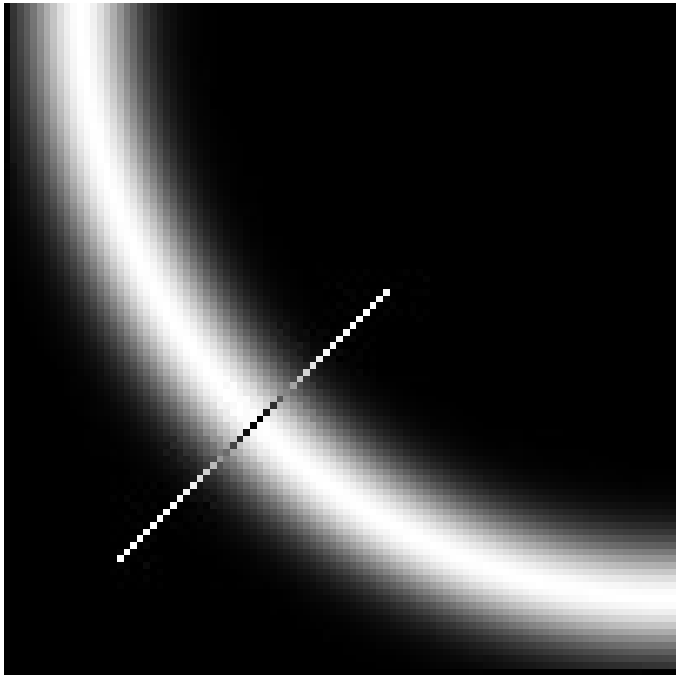
How to turn these thick regions of the gradient into curves?

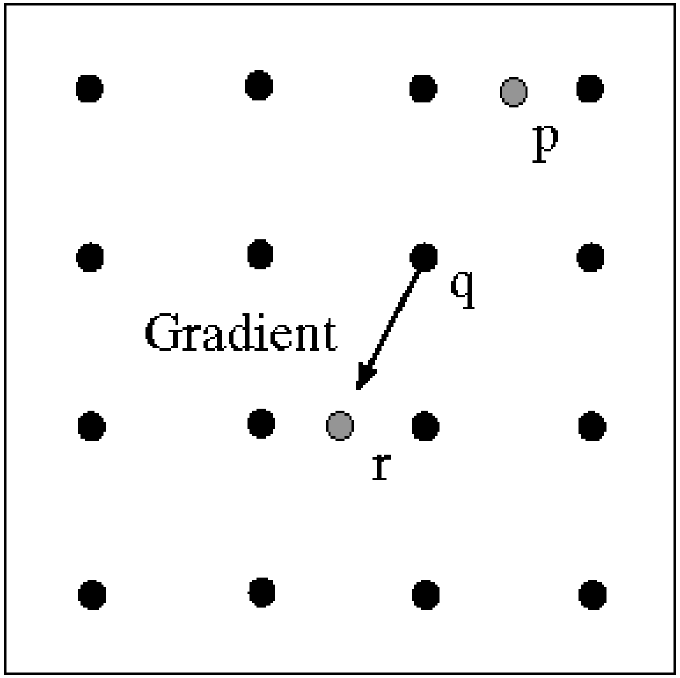
Magnitude of gradient

Non-maximum suppression

Check if pixel is local maximum along gradient direction, select single max across width of the edge

– requires checking interpolated pixels p and r





Source: S. Seitz

The Canny edge detector

Magnitude of gradient

The Canny edge detector

Non-maximum suppression

The Canny edge detector

Threshold

Problem: pixels along this edge didn’t survive the thresholding

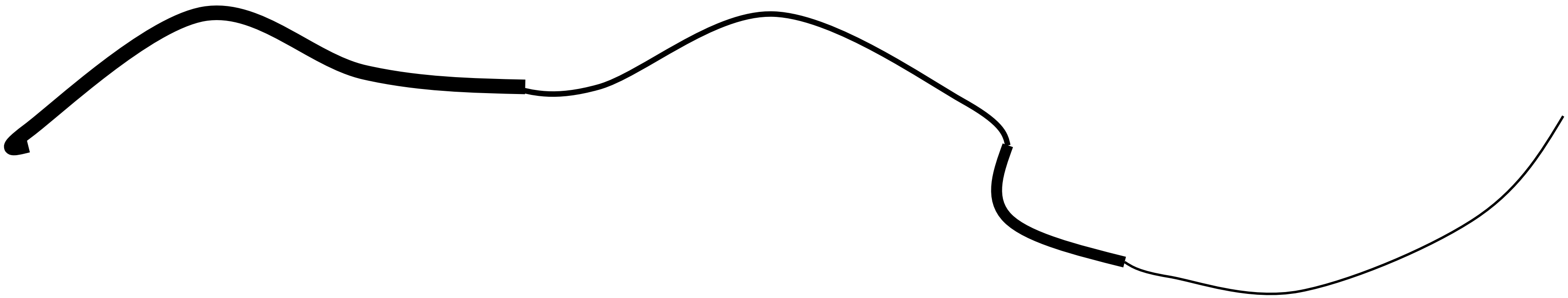
The Canny edge detector

Lower threshold

Problem:
Too much
noise

Hysteresis thresholding

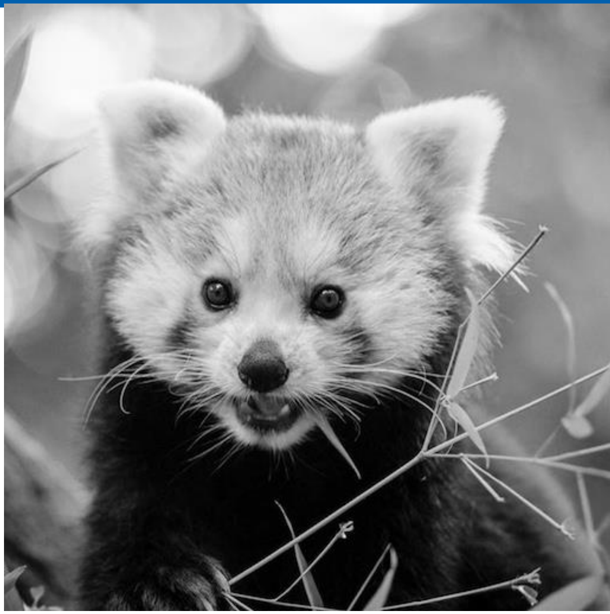
Use a high threshold to start edge curves, and a low threshold to continue them.



$$\begin{array}{rl} \|\nabla f(x,y)\| \geq t_1 & \text{definitely an edge} \\ t_0 \geq \|\nabla f(x,y)\| < t_1 & \text{maybe an edge, depends on context} \\ \|\nabla f(x,y)\| < t_0 & \text{definitely not an edge} \end{array}$$

Source: S. Seitz

Hysteresis thresholding



Low threshold

High threshold

Hysteresis threshold

Canny edge detector

UNIVERSITÄT BONN

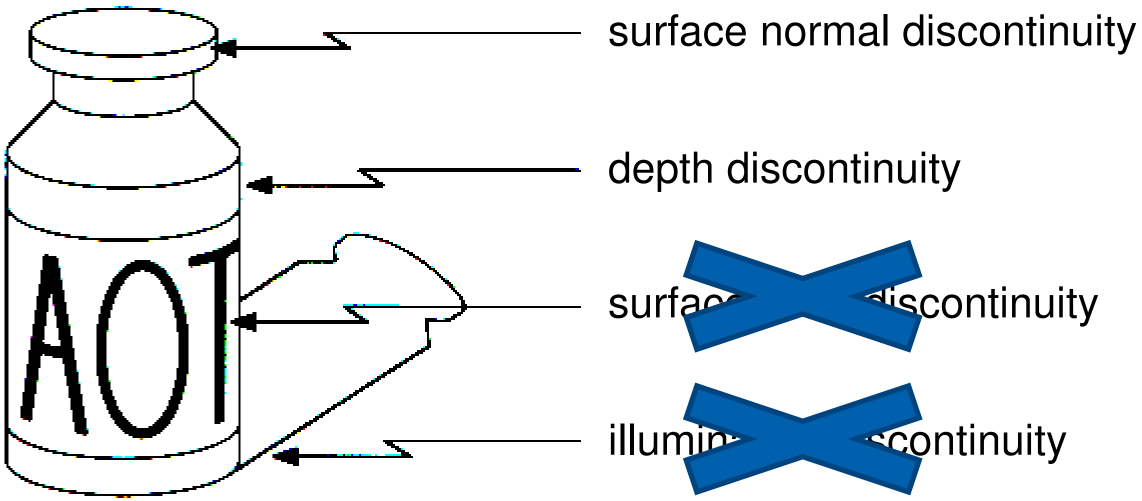
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   – Thin wide “ridges” down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   – Define two thresholds: low and high
   – Use the high threshold to start edge curves and the low threshold to continue them

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: S. Lazebnik

Relevance of edges

Application shape matching



Source: S. Seitz

Not all edges are relevant…

Berkeley segmentation database:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

image | human segmentation | gradient magnitude

Source: S. Lazebnik

Is boundary detection solved?

UNIVERSITÄT BONN

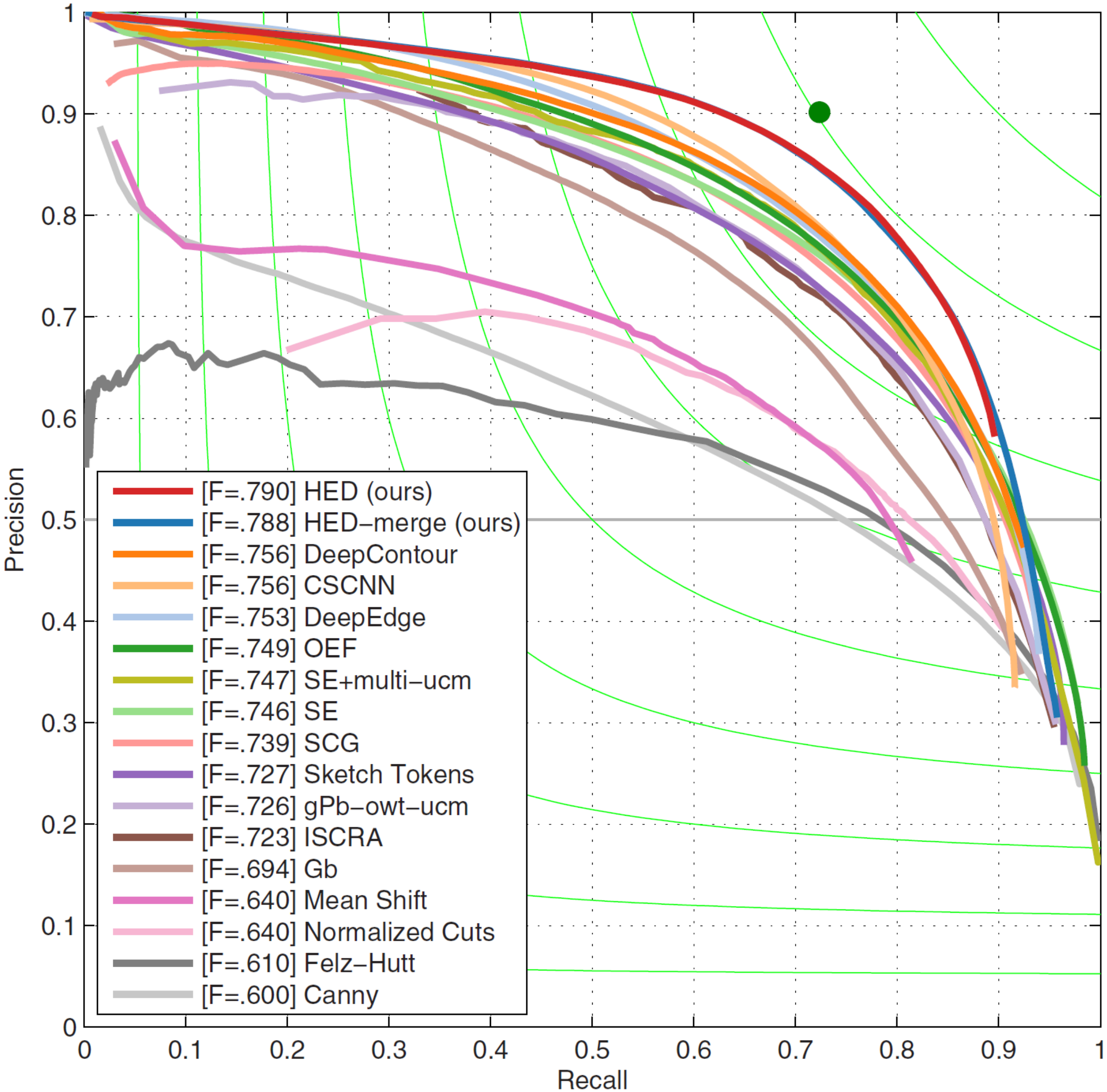
$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

P. Arbelaez et al. **Contour Detection and Hierarchical Image Segmentation.** TPAMI 2011

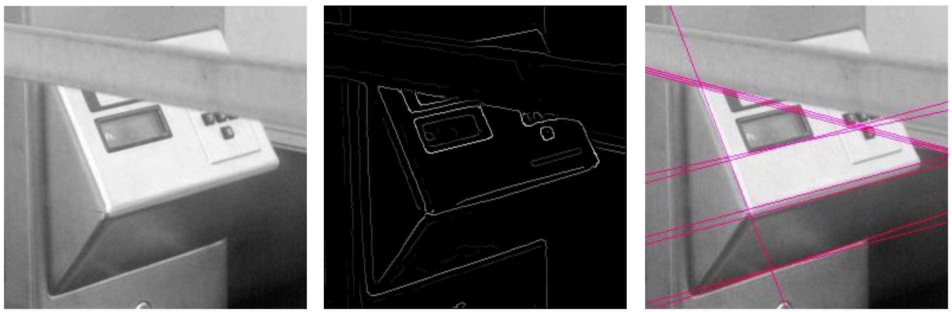
Saining Xie and Zhuowen Tu **Holistically-Nested Edge Detection.** IJCV 2017

Machine Learning for boundary detection



Fitting

Want to associate a model with observed features



[Fig from Marszalek & Schmid, 2007]

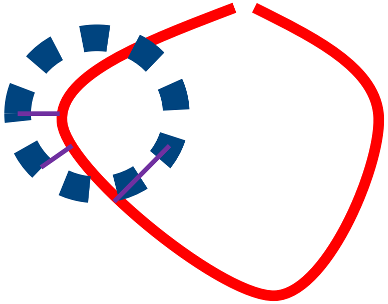
For example, the model could be a line, a circle, or an arbitrary shape.

Source: K. Grauman

Chamfer distance

- Average distance to nearest feature

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$



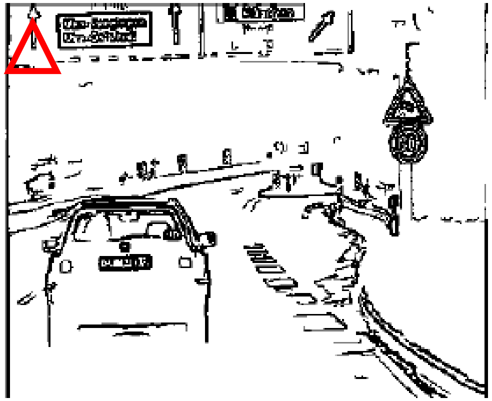
- T: template shape→ a set of points
- I: image to search→ a set of points
- $d_I(t)$: min distance for point t to some point in I

Source: K. Grauman

Chamfer distance

Average distance to nearest feature

$$D_{chamfer}(T,I) \equiv \frac{1}{|T|}\sum_{t\in T} d_I(t)$$



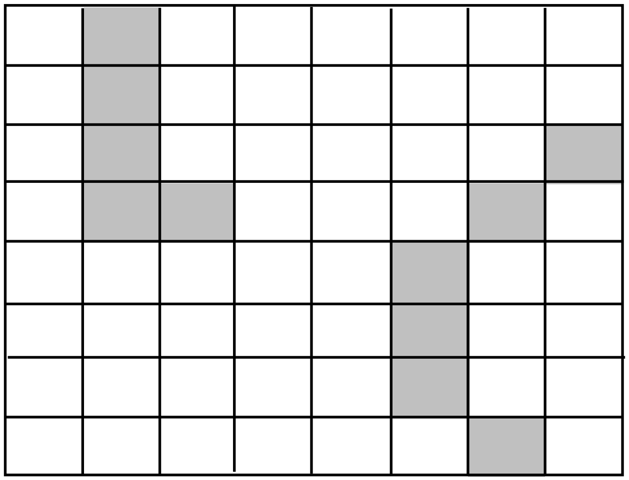
Edge image

Can it be implemented as correlation or convolution?

Source: K. Grauman

Distance transform

Image features (2D)



Distance Transform

1	0	1	2	3	4	3	2
1	0	1	2	3	3	2	1
1	0	1	2	3	2	1	0
1	0	0	1	2	1	0	1
2	1	1	2	1	0	1	2
3	2	2	2	1	0	1	2
4	3	3	2	1	0	1	2
5	4	4	3	2	1	0	1

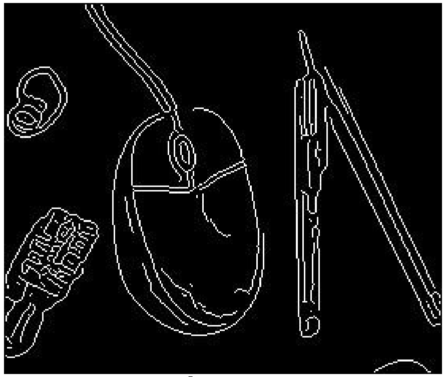
Distance Transform is a function $D(\cdot)$ that for each image pixel p assigns a non-negative number $D(p)$ corresponding to distance from p to the nearest feature in the image I

Features could be edge points, foreground points,…

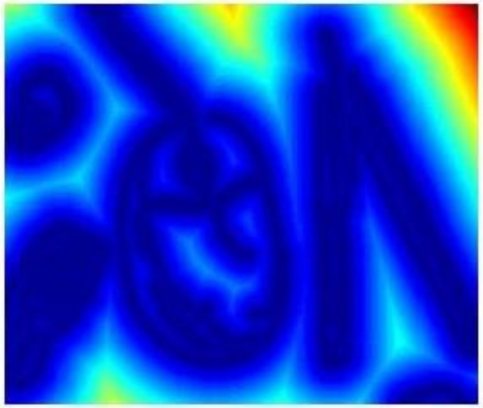
Source: Y. Boykov

Distance transform

original



edges



distance transform

Value at (x,y) tells how far that position is from the nearest edge point (or other binary mage structure)

Source: K. Grauman

Distance transform (1D)

Two pass O(n) algorithm for 1D L_1 norm

1. Initialize: For all j
   $D[j] \leftarrow 1_P[j]$ // 0 if j is in **P**, infinity otherwise

P. Felzenszwalb and D. Huttenlocher. **Distance Transforms of Sampled Functions.** Cornell Computing and Information Science TR2004-1963

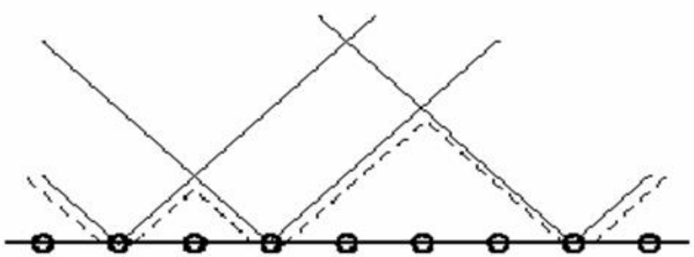
Source: D. Huttenlocher

Distance transform (1D)

UNIVERSITÄT BONN

Two pass O(n) algorithm for 1D L_1 norm

1. Initialize: For all j
   $D[j] \leftarrow 1_P[j]$ // 0 if j is in **P**, infinity otherwise
2. Forward: For j from 1 up to n-1
   $D[j] \leftarrow \min(D[j], D[j-1]+1)$
3. Backward: For j from n-2 down to 0
   $D[j] \leftarrow \min(D[j], D[j+1]+1)$



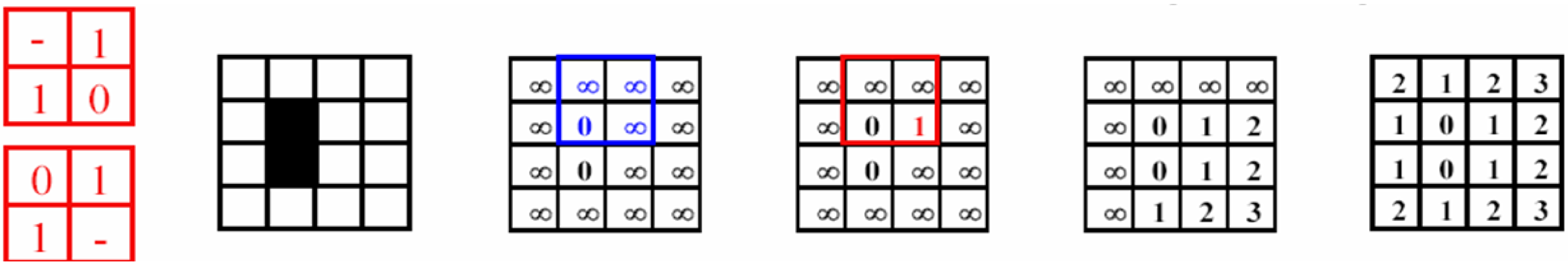
∞	0	∞	0	∞	∞	∞	0	∞
∞	0	1	0	1	2	3	0	1
1	0	1	0	1	2	1	0	1

P. Felzenszwalb and D. Huttenlocher. **Distance Transforms of Sampled Functions.** Cornell Computing and Information Science TR2004-1963

Source: D. Huttenlocher

Distance Transform (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right



Source: D. Huttenlocher

Distances

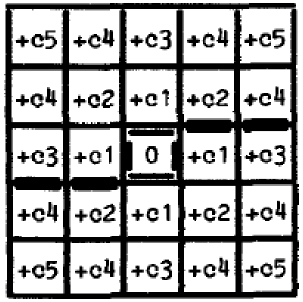
Minkowski distance:

$$d_p(x, y) = \left(\sum_i |x_i - y_i|^p \right)^{1/p}$$

$$d_\infty(x, y) = \lim_{p \to \infty} \left(\sum_i |x_i - y_i|^p \right)^{1/p} = \max_i \left(|x_i - y_i| \right)$$

Special cases: L1-norm, L2-norm (Euclidean)

Approximate Euclidean



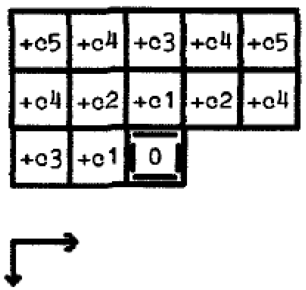
Parallel mask

Forward:

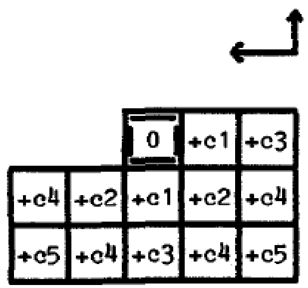
for $i = (\text{size} + 1)/2, \ldots, \text{lines}$ do

for $j = (\text{size} + 1)/2, \ldots, \text{columns}$ do

$$v_{i,j} = \underset{\substack{(k,l)\in \\ \text{forward mask}}}{\text{minimum}} \left(v_{i+k,j+l} + c(k,l)\right)$$



Forward mask



Backward mask

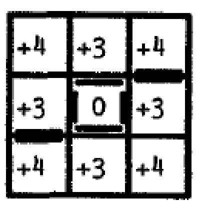
Backward:

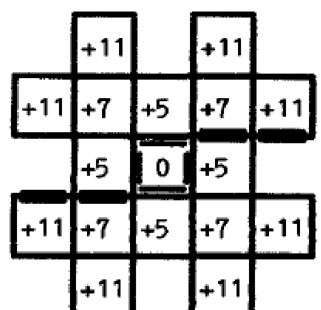
for $i = \text{lines} - (\text{size} - 1)/2, \ldots, 1$ do

for $j = \text{columns} - (\text{size} - 1)/2, \ldots, 1$ do

$$v_{i,j} = \underset{\substack{(k,l)\in \\ \text{backward mask}}}{\text{minimum}} \left(v_{i+k,j+l} + c(k,l)\right)$$

Chamfer 3-4
3 x sqrt(2) ≈ 4
3 x 1 = 3



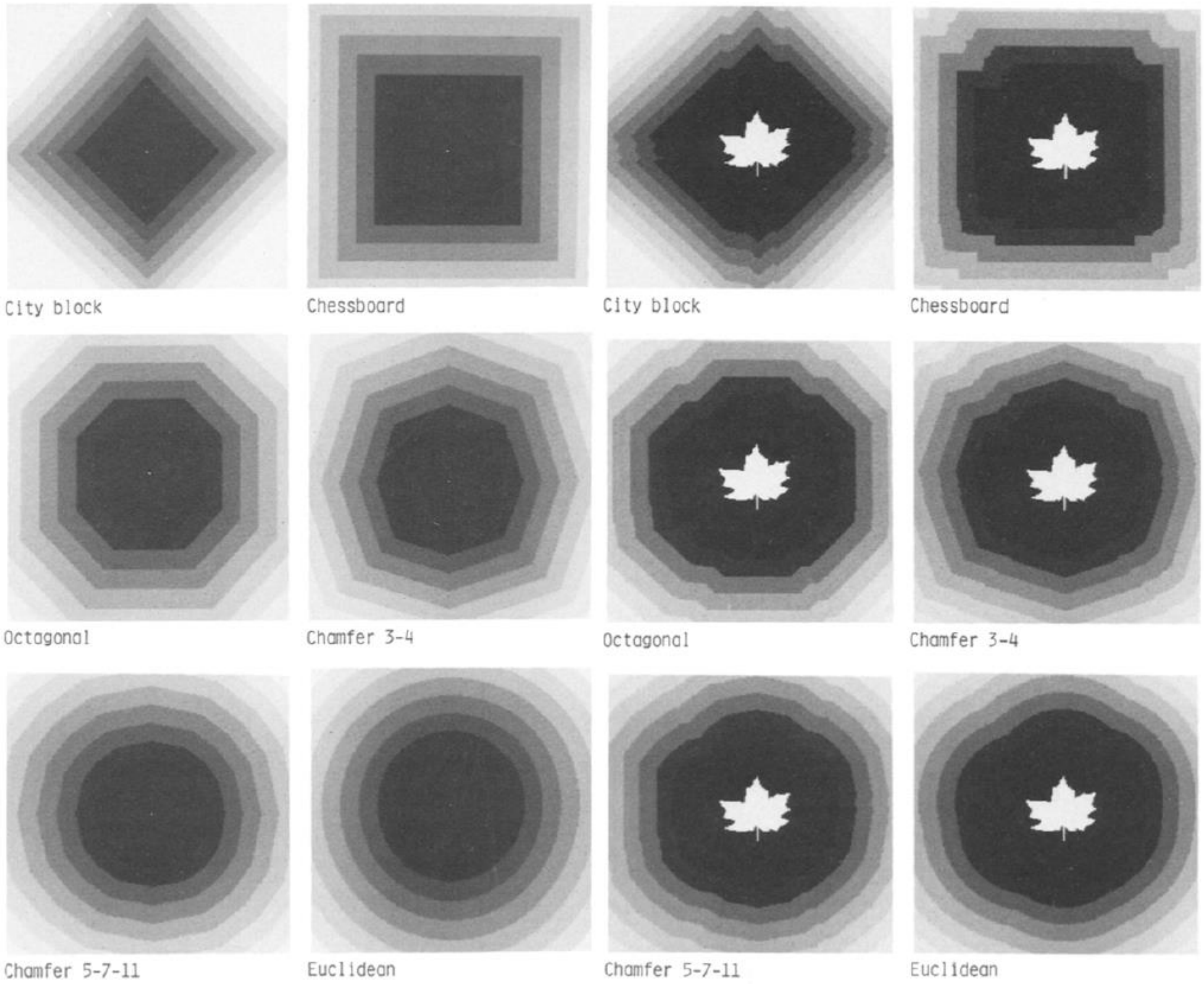


Chamfer 5-7-11
5 x sqrt(5) ≈ 11
5 x sqrt(2) ≈ 7
5 x 1 = 5

G. Borgefors. **Distance transformations in digital images.** Computer Vision, Graphics, and Image Processing, 34:344–371, 1986.

Approximate Euclidean

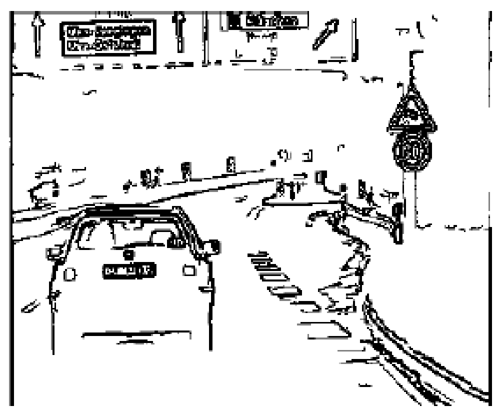
UNIVERSITÄT BONN



Chamfer distance

Average distance to nearest feature

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

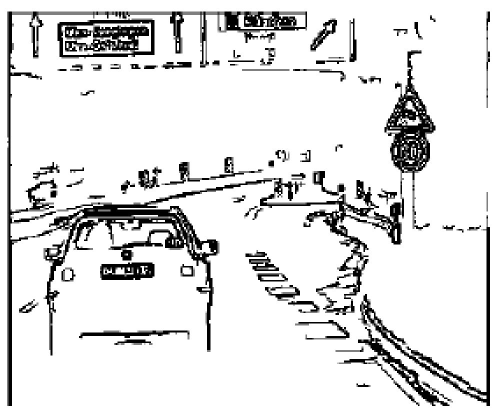


Edge image

Distance transform image

Source: D. Gavrila

Chamfer distance



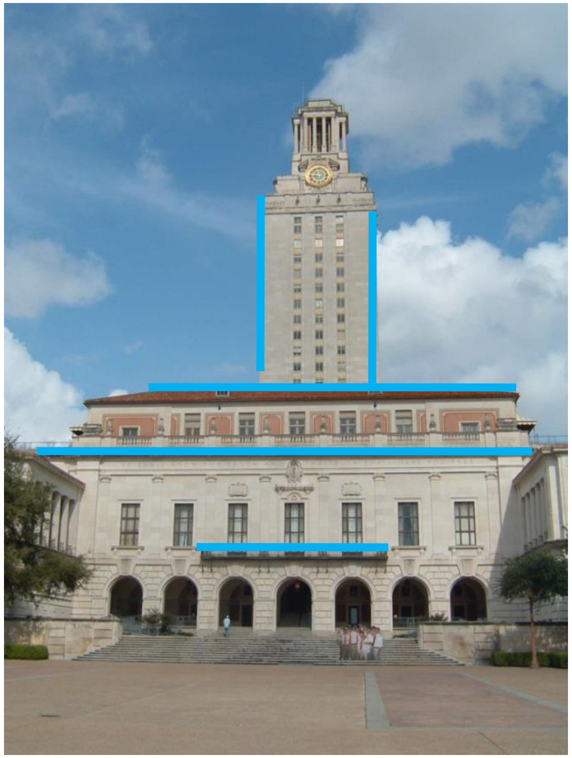
Edge image

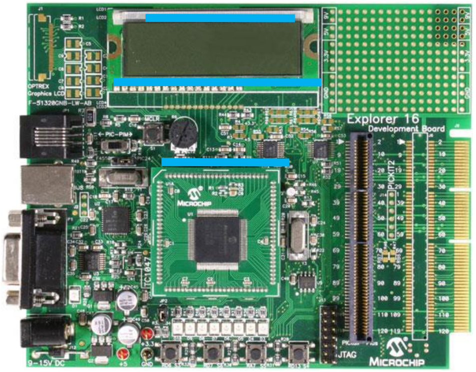
Distance transform image

Source: D. Gavrila

Line fitting

- Many objects characterized by presence of straight lines





Lines can be detected by the Hough Transform

Source: K. Grauman

Hough space

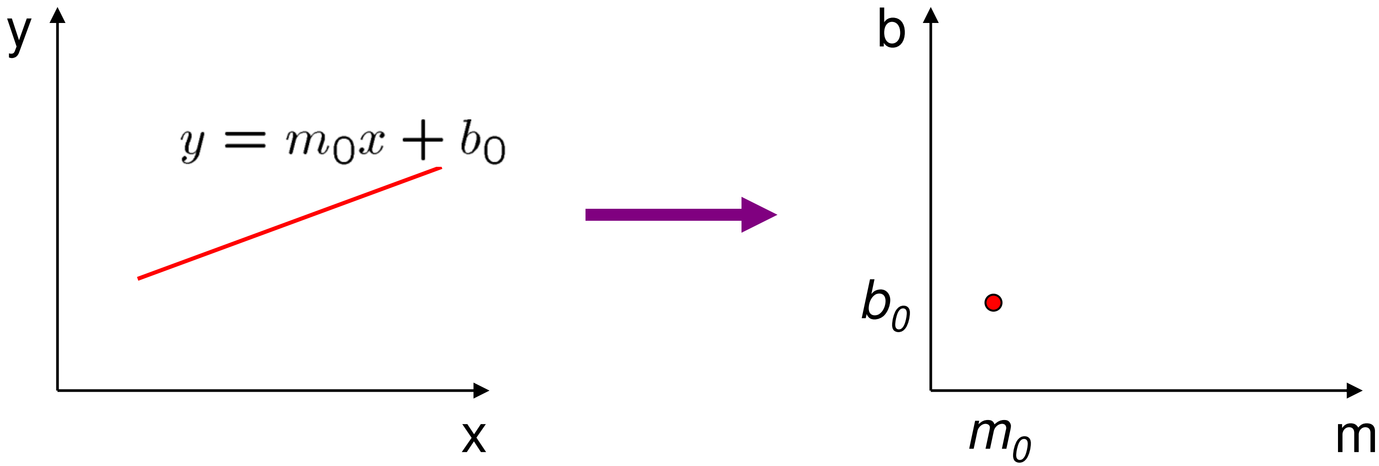


image space

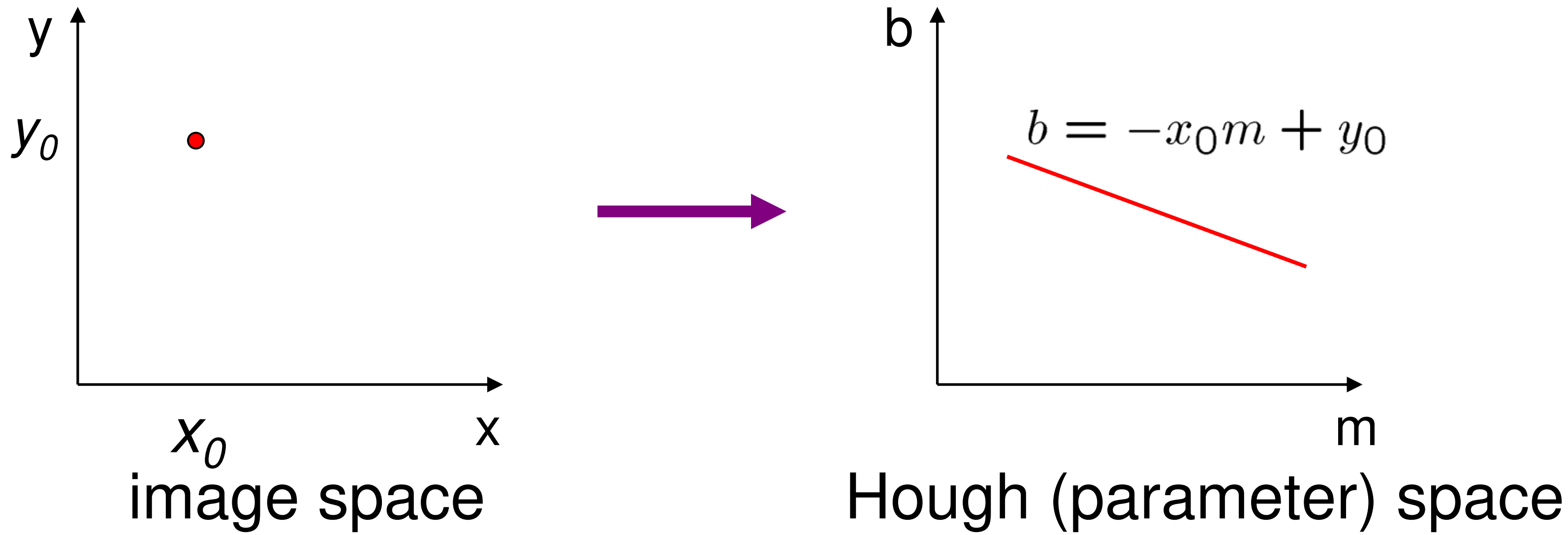
Hough (parameter) space

Connection between image (x,y) and Hough (m,b) spaces:

- A line in the image corresponds to a point in Hough space

Source: S. Seitz

Hough space

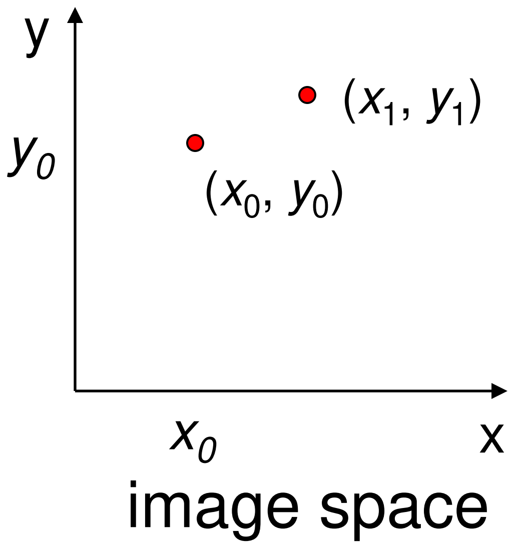


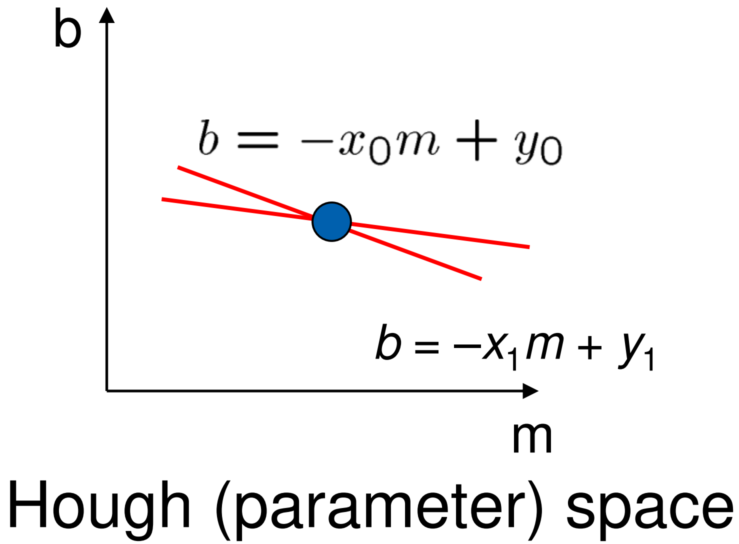
Connection between image (x,y) and Hough (m,b) spaces:

- A line in the image corresponds to a point in Hough space
- What does a point (x_0, y_0) in the image space map to?
- The solutions of $b = -x_0 m + y_0$ (line in Hough space)

Source: S. Seitz

Hough space





- What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1)?
- It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

Source: S. Seitz

Hough algorithm

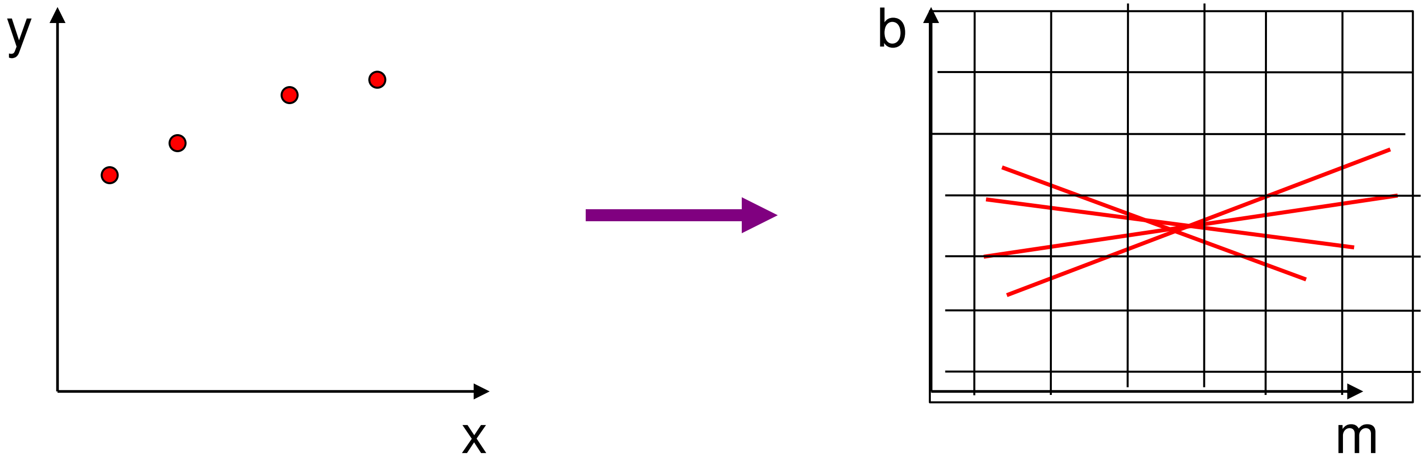


image space Hough (parameter) space

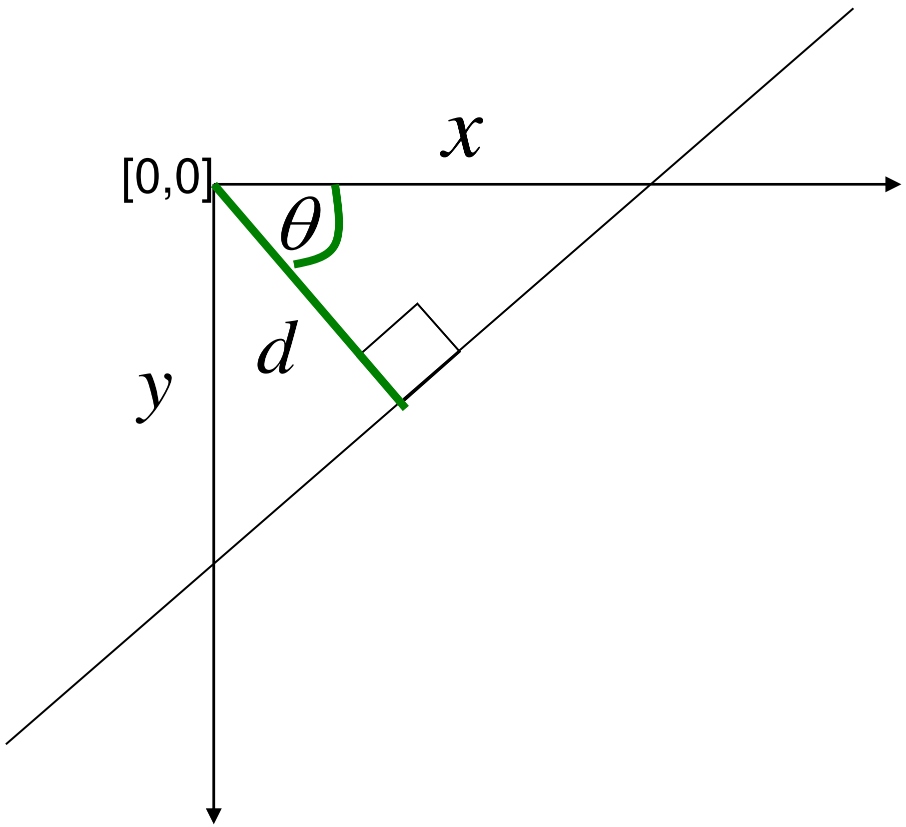
How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?

- Let each edge point in image space *vote* for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Source: S. Seitz

Polar representation for lines

Issues with usual (*m,b*) parameter space: can take on infinite values, undefined for vertical lines.



d : perpendicular distance from line to origin

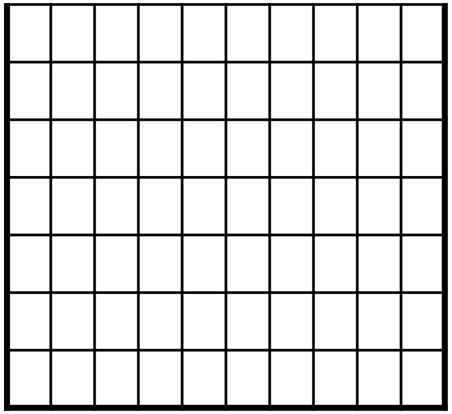
θ : angle the perpendicular makes with the x-axis

$$x\cos\theta - y\sin\theta = d$$

Point in image space → sinusoid segment in Hough space

Source: K. Grauman

Hough transform algorithm



Using the polar parameterization:

$$x\cos\theta - y\sin\theta = d$$

Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

   for θ = 0 to 180 // some quantization

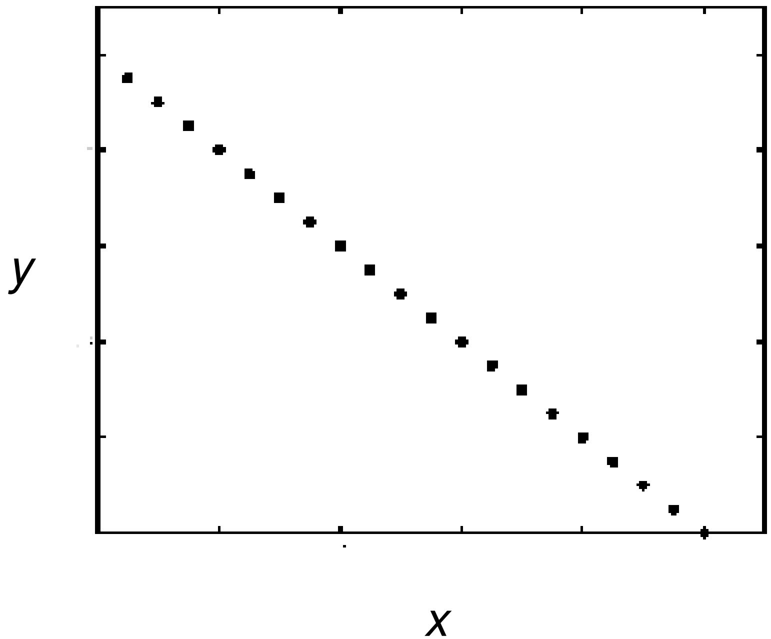
   $$d = x\cos\theta - y\sin\theta$$

   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta - y\sin\theta$

Source: S. Seitz

Lines

UNIVERSITÄT BONN



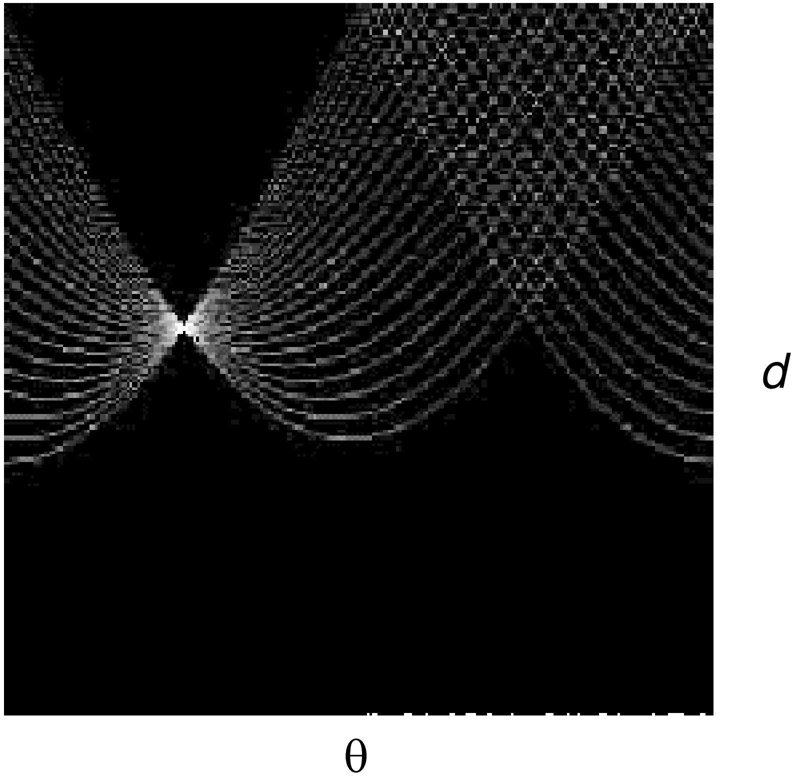


Image space edge coordinates

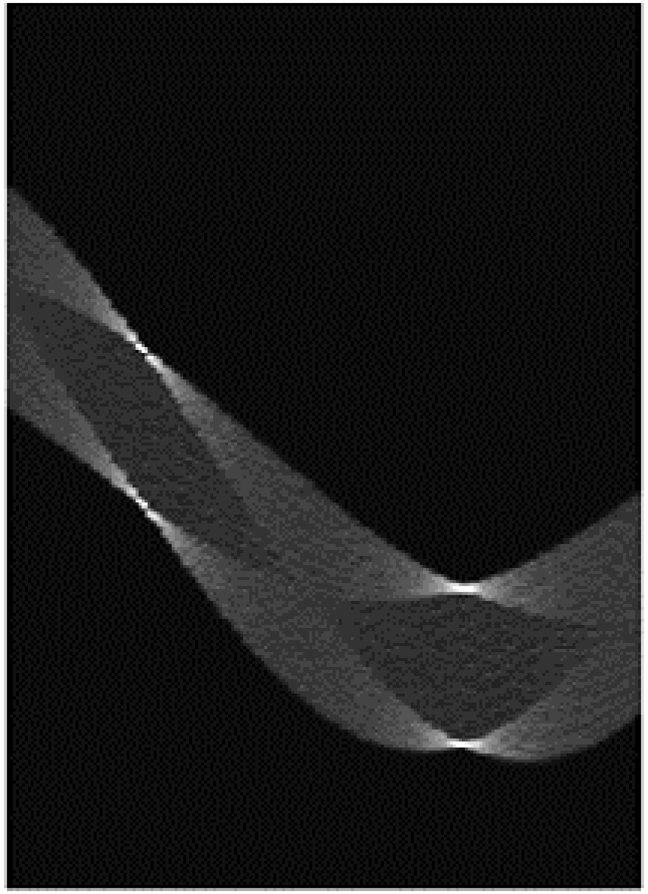
Votes

Bright value = high vote count
Black = no votes

Source: K. Grauman

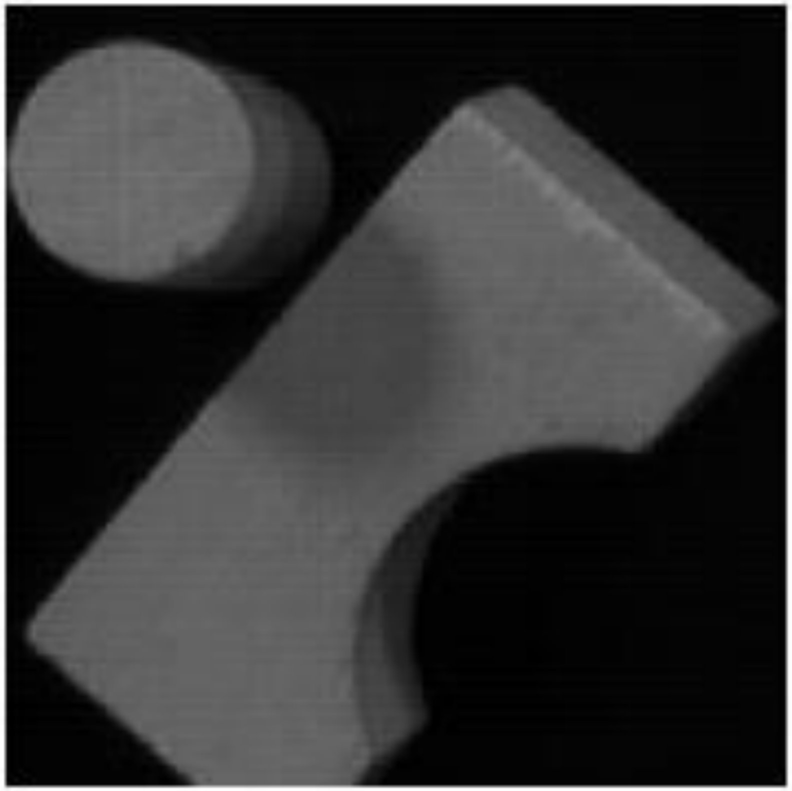
Lines

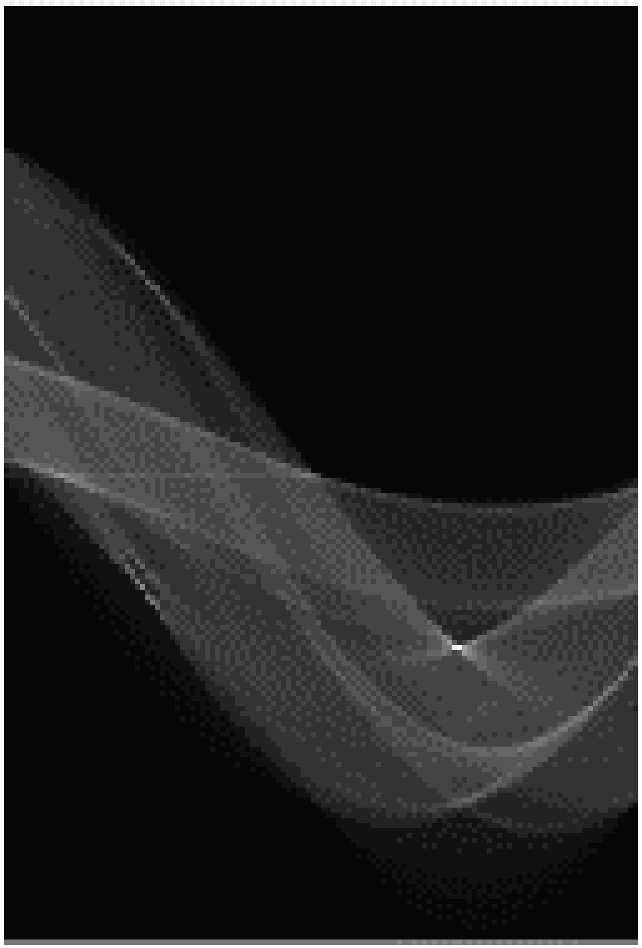
Square :



Source: K. Grauman

Lines





Source: K. Grauman

Hough transform for circles

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

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

- For a fixed radius r, unknown gradient direction

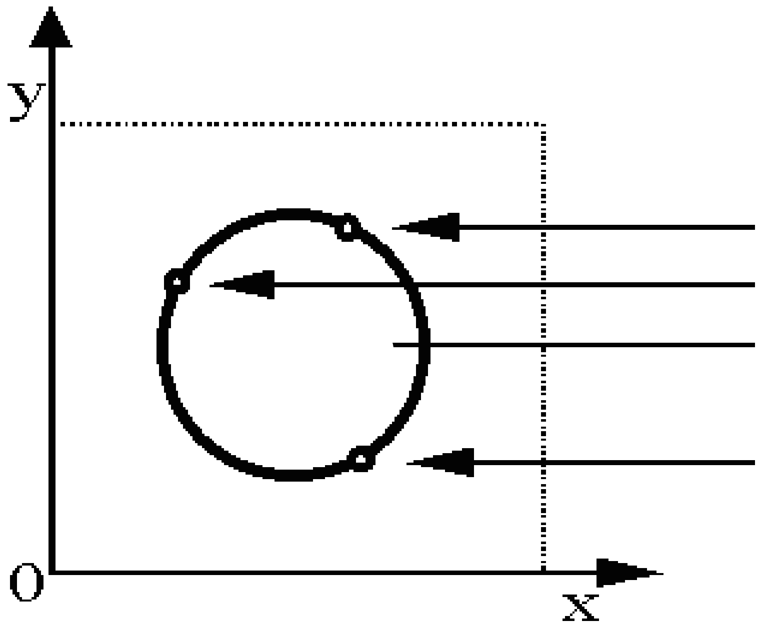
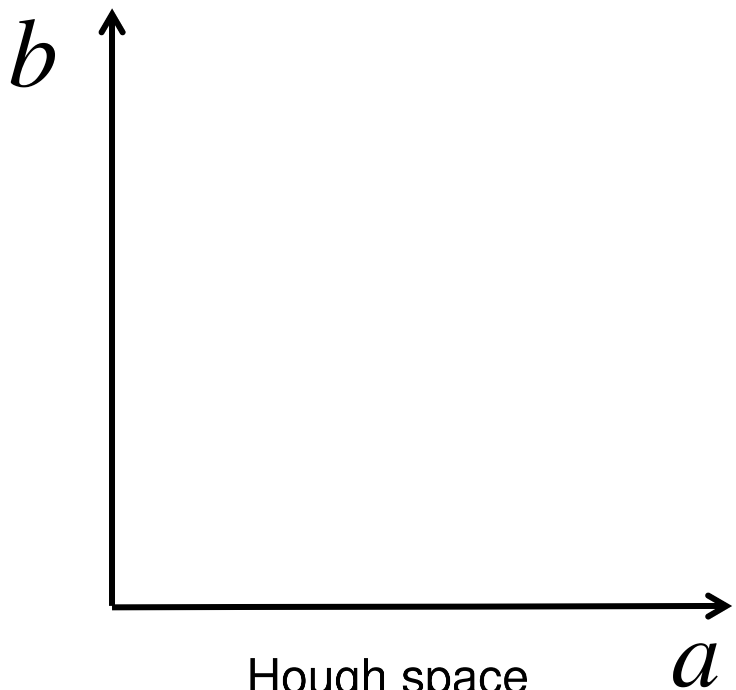


Image space



Hough space

Source: K. Grauman

Hough transform for circles

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

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

- For a fixed radius r, unknown gradient direction

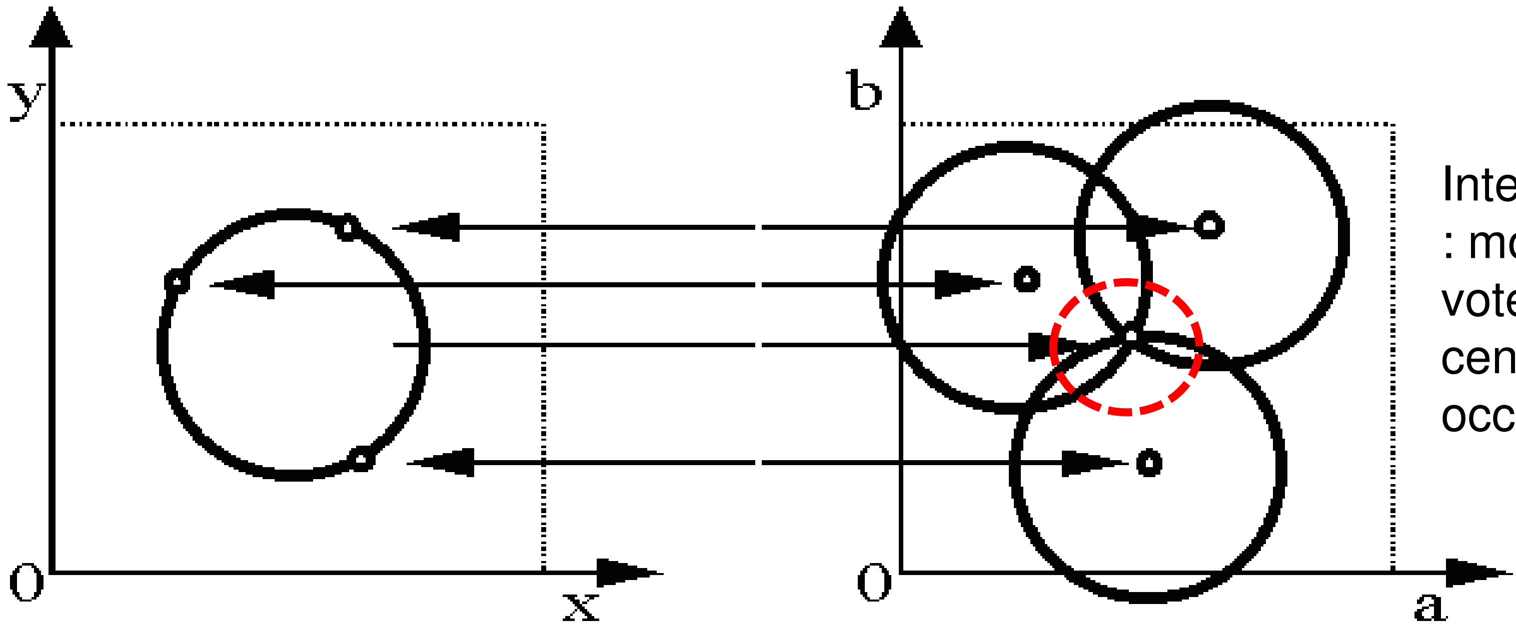


Image space

Hough space

Intersection : most votes for center occur here.

Source: K. Grauman

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, unknown gradient direction

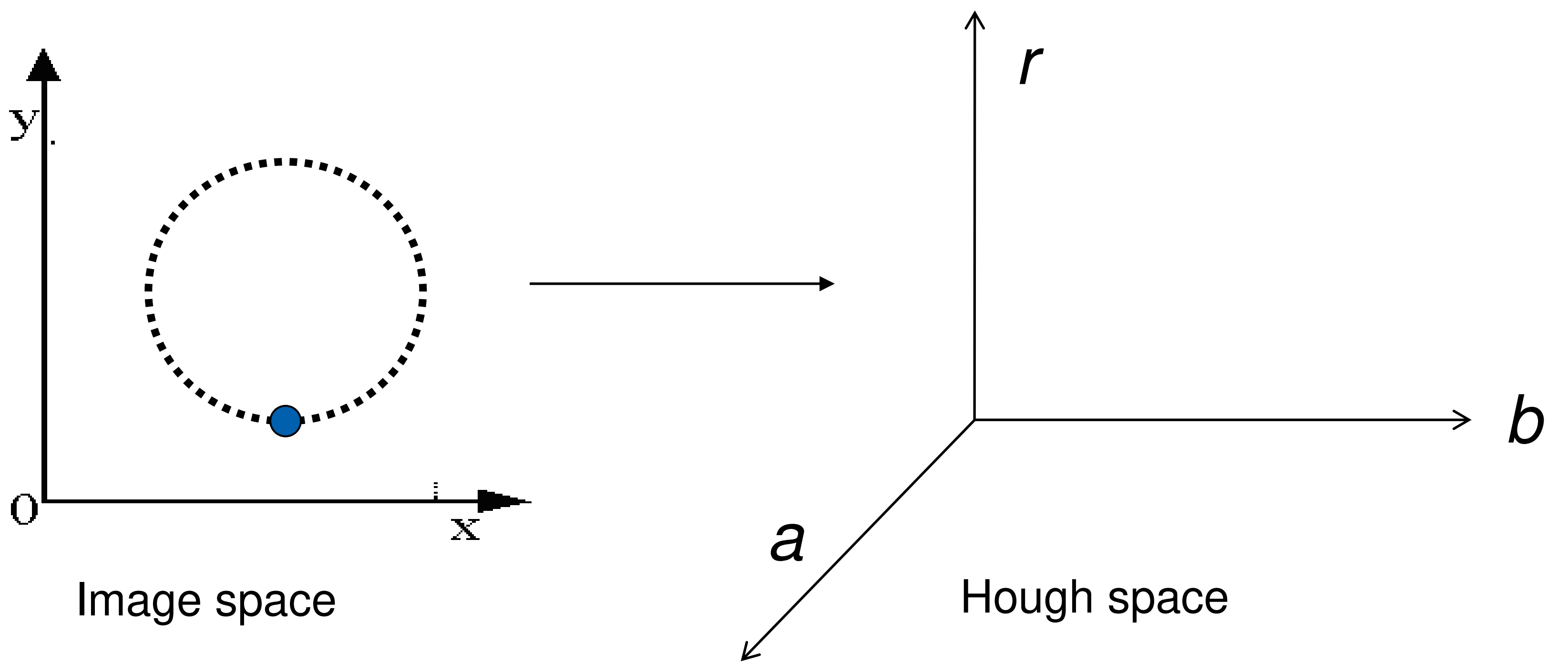


Image space

Hough space

Source: K. Grauman

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, unknown gradient direction

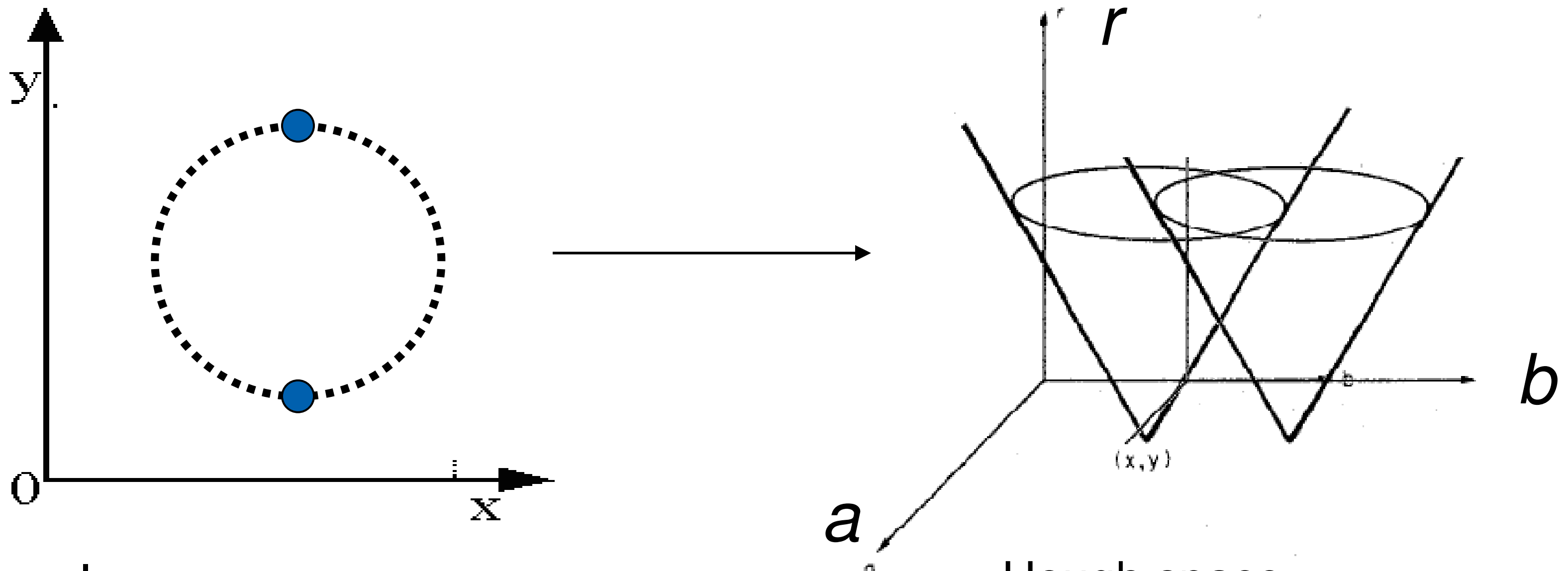


Image space

Hough space

Source: K. Grauman

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, **known** gradient direction

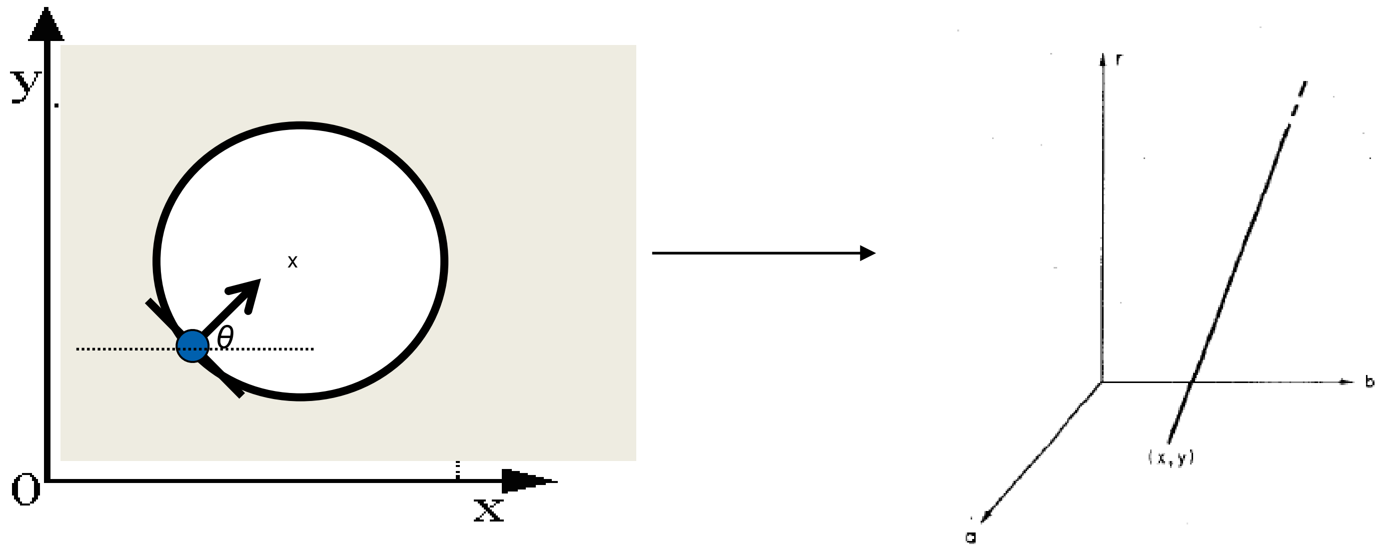


Image space

Hough space

Source: K. Grauman

Hough transform for circles

For every edge pixel (x,y) :

For each possible radius value r:

For each possible gradient direction θ:

// or use estimated gradient

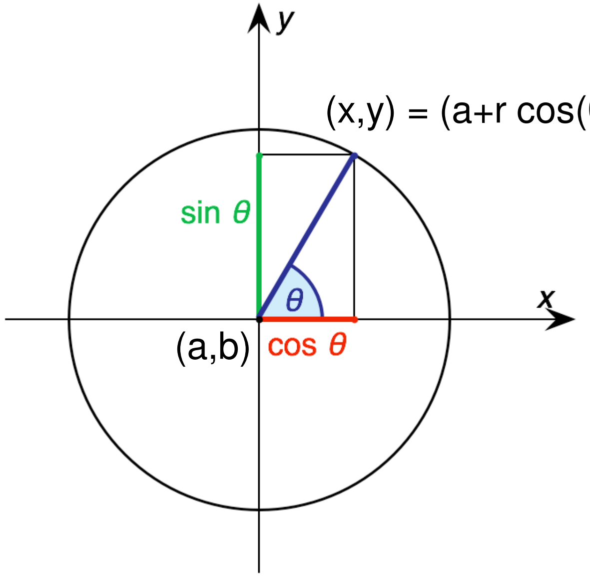
$a = x - r\cos(\theta)$

$b = y - r\sin(\theta)$

H[a,b,r] += 1

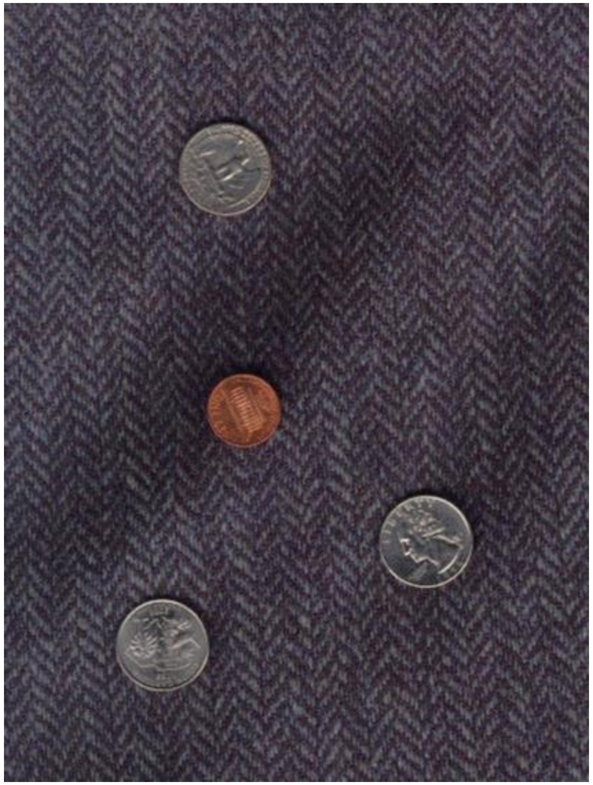
end

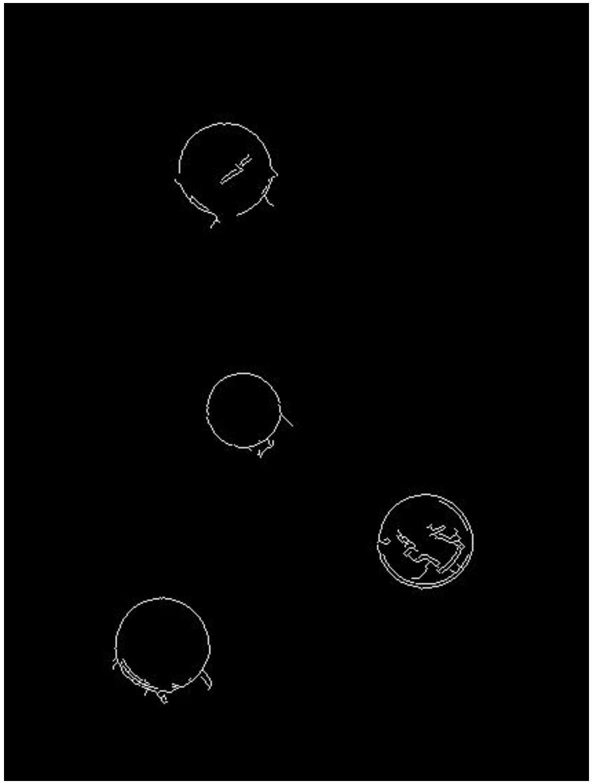
end

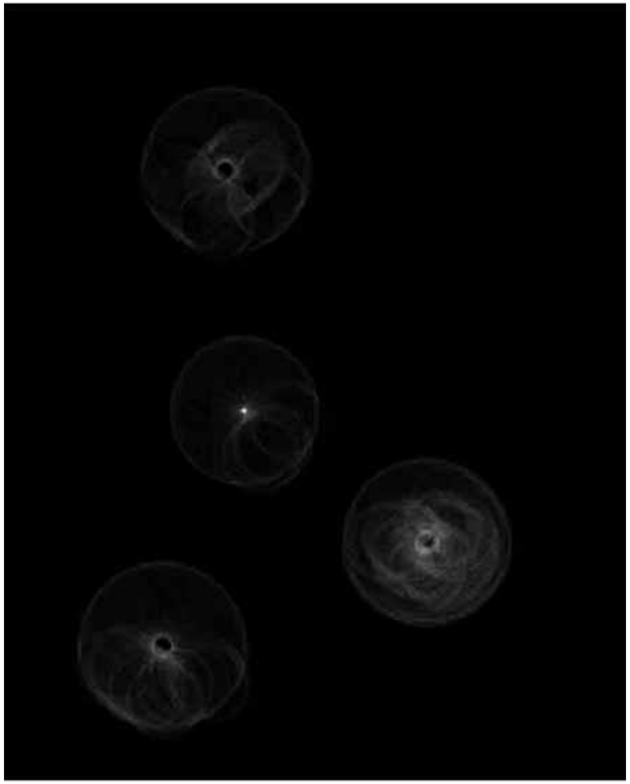


Source: K. Grauman

Circles





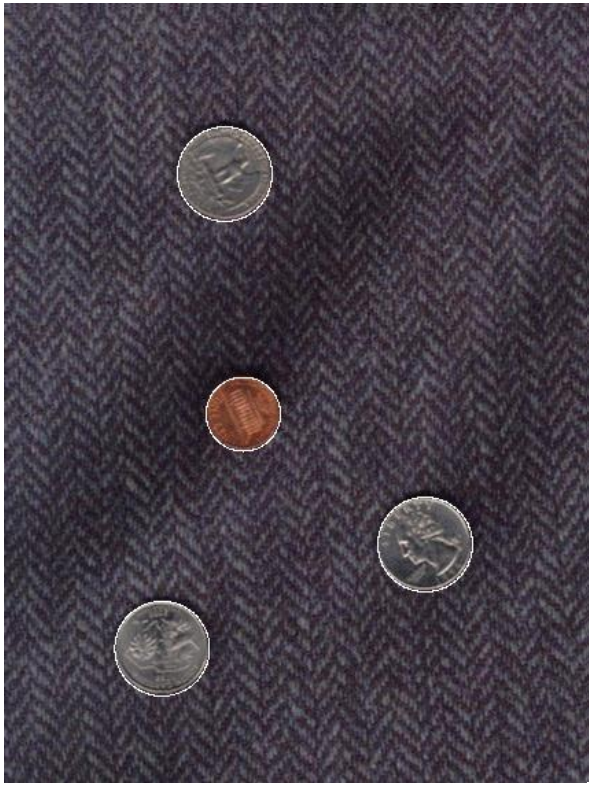


Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

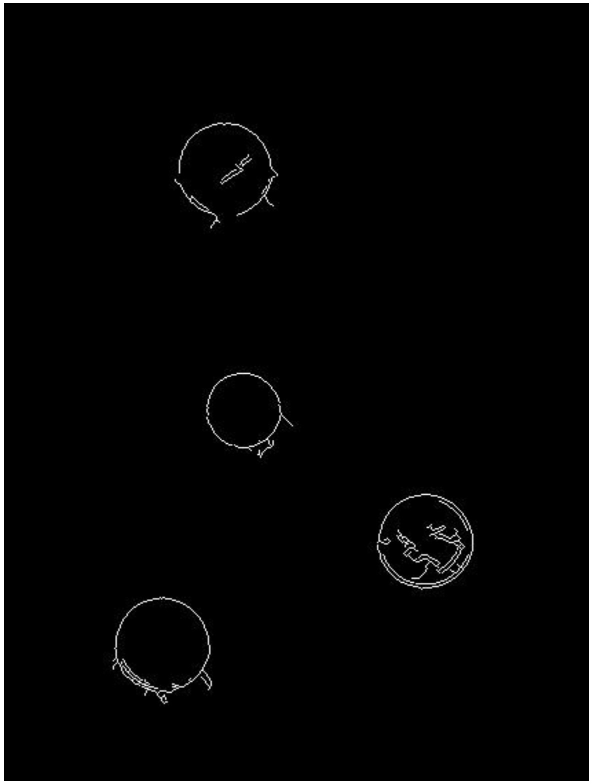
Source: V. Kwatra

Circles

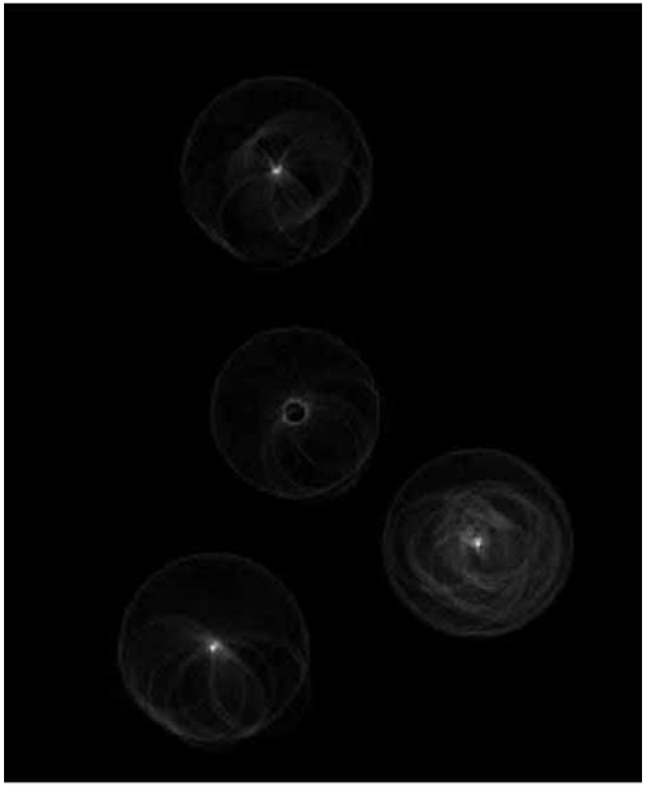
Original



Edges



Votes: Quarter



Combined detections

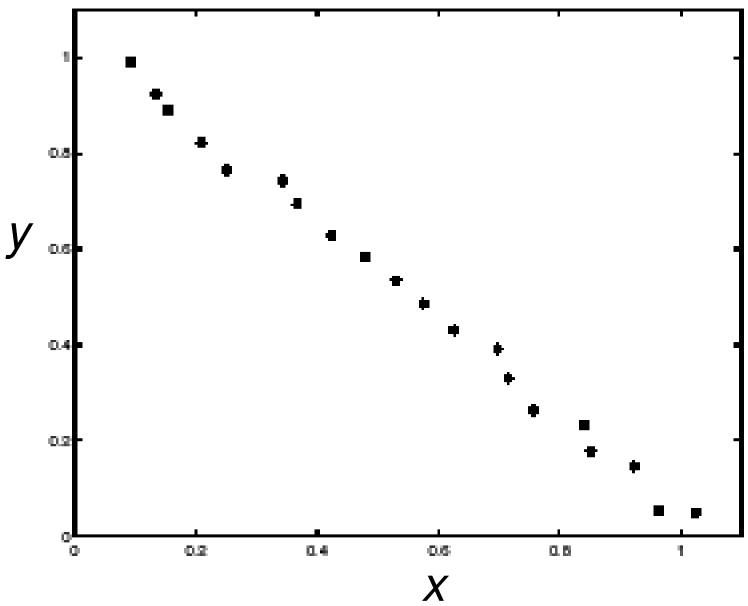
Source: V. Kwatra

Parameters for analytic curves

Analytic Form	Parameters	Equation
Line	d, θ	$x\cos\theta + y\sin\theta = d$
Circle	x_0, y_0, r	$(x-x_o)^2+(y-y_0)^2=r^2$
Parabola	x_0, y_0, ρ, θ	$(y-y_0)^2=4\rho(x-x_o)$
Ellipse	x_0, y_0, a, b, θ	$(x-x_o)^2/a^2+(y-y_0)^2/b^2=1$

Source: S. Tandon

Impact of noise on Hough



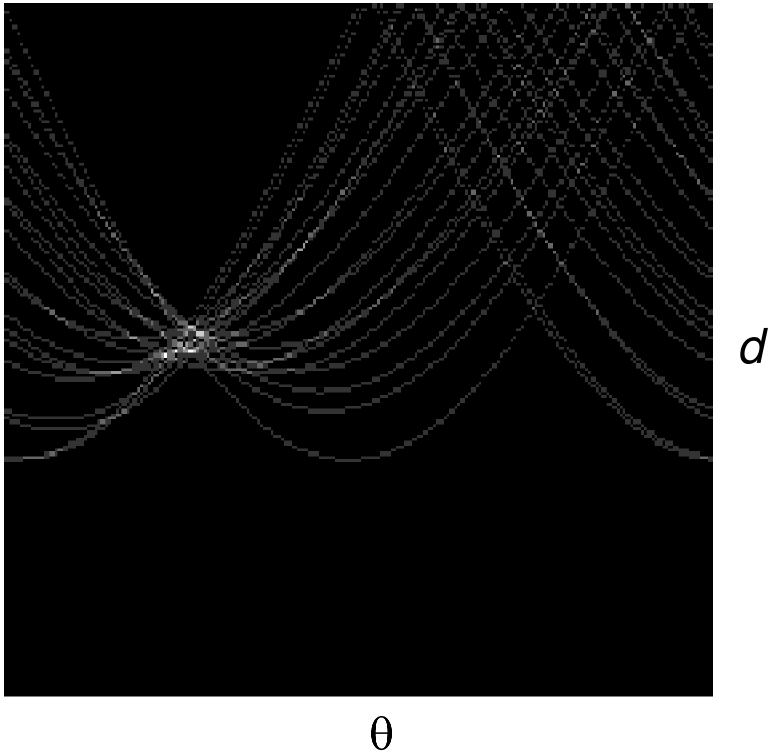


Image space edge coordinates

Votes

What difficulty does this present for an implementation?

Source: K. Grauman

Impact of noise on Hough

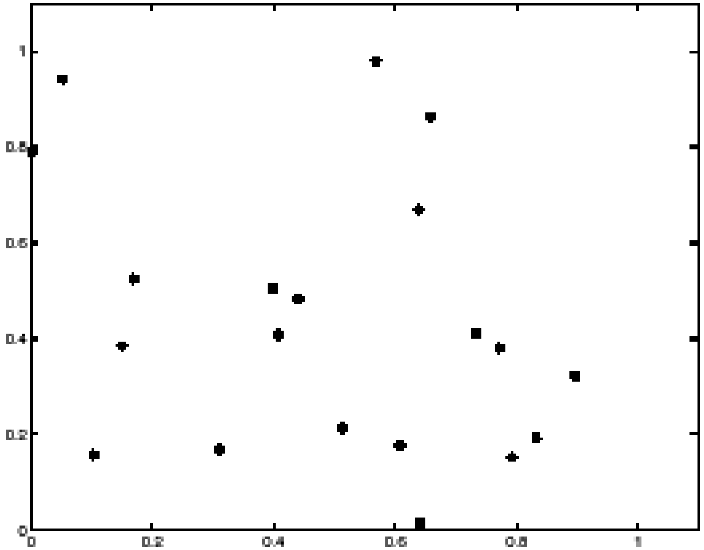
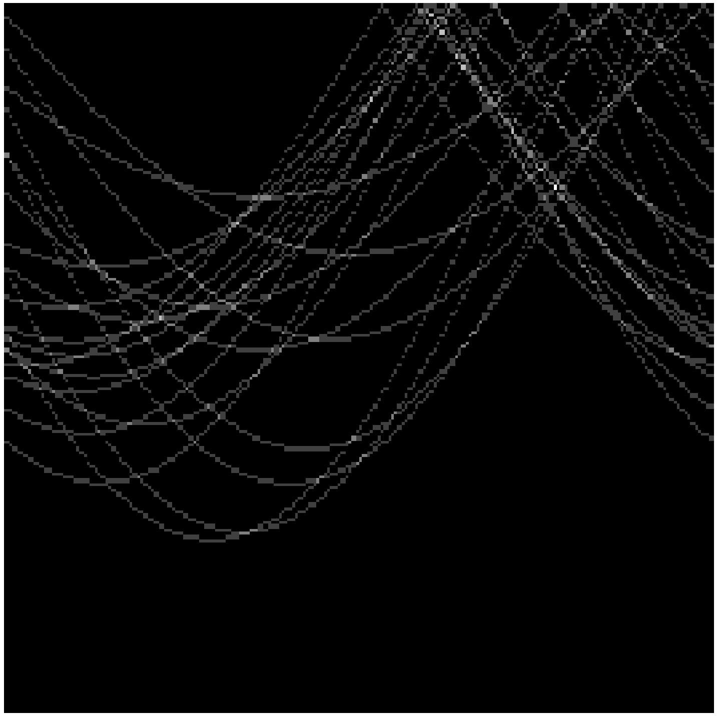


Image space edge coordinates



Votes

Here, everything appears to be "noise", or random edge points, but we still see peaks in the vote space.

Source: K. Grauman

Voting: practical tips

UNIVERSITÄT BONN

Minimize irrelevant tokens first (take edge points with significant gradient magnitude)

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

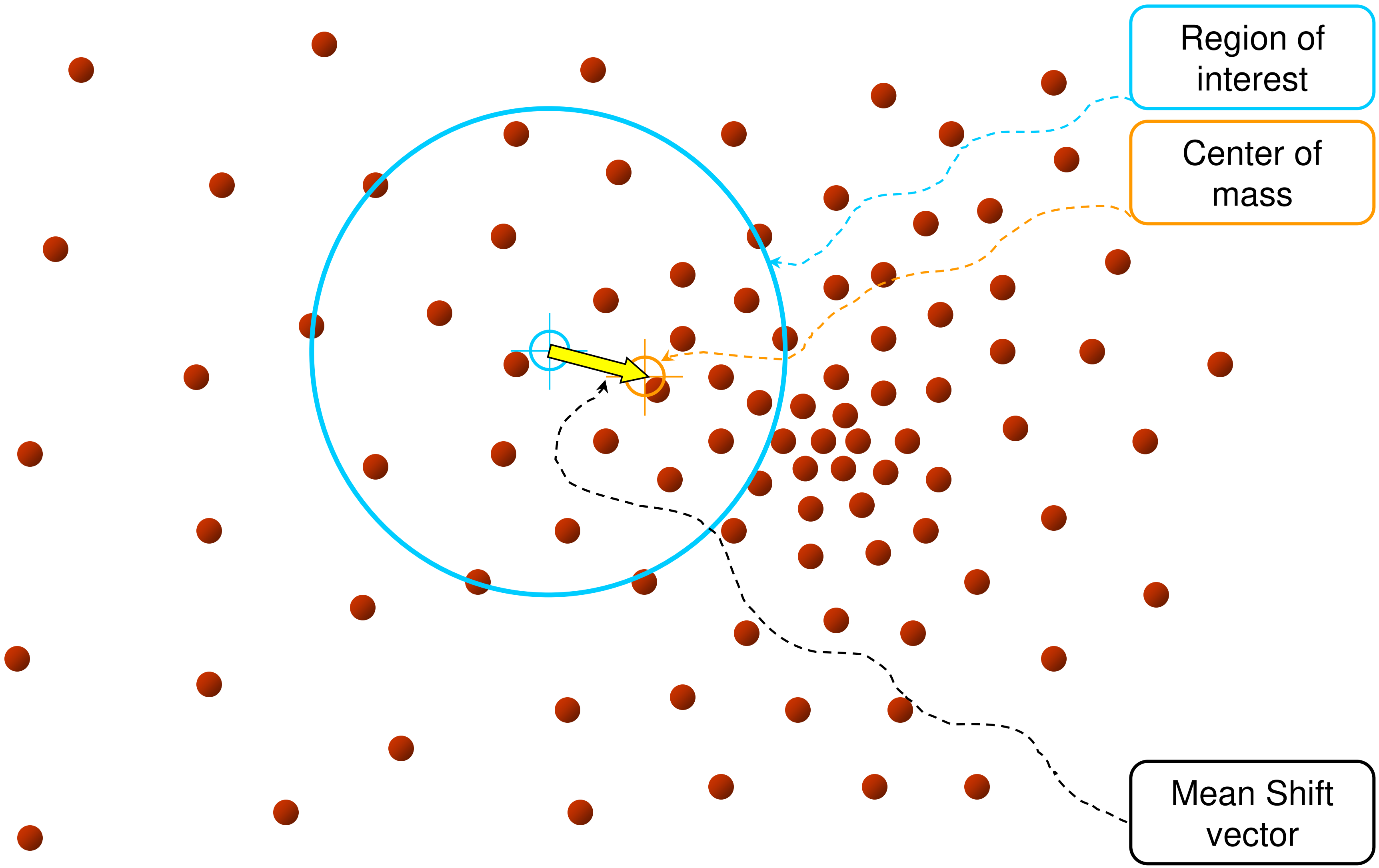
Vote for neighbors also (smoothing in accumulator array) or use mean shift

Utilize direction of edge to reduce free parameters by 1

To read back which points voted for “winning” peaks, keep tags on the votes.

Source: K. Grauman

Mean Shift

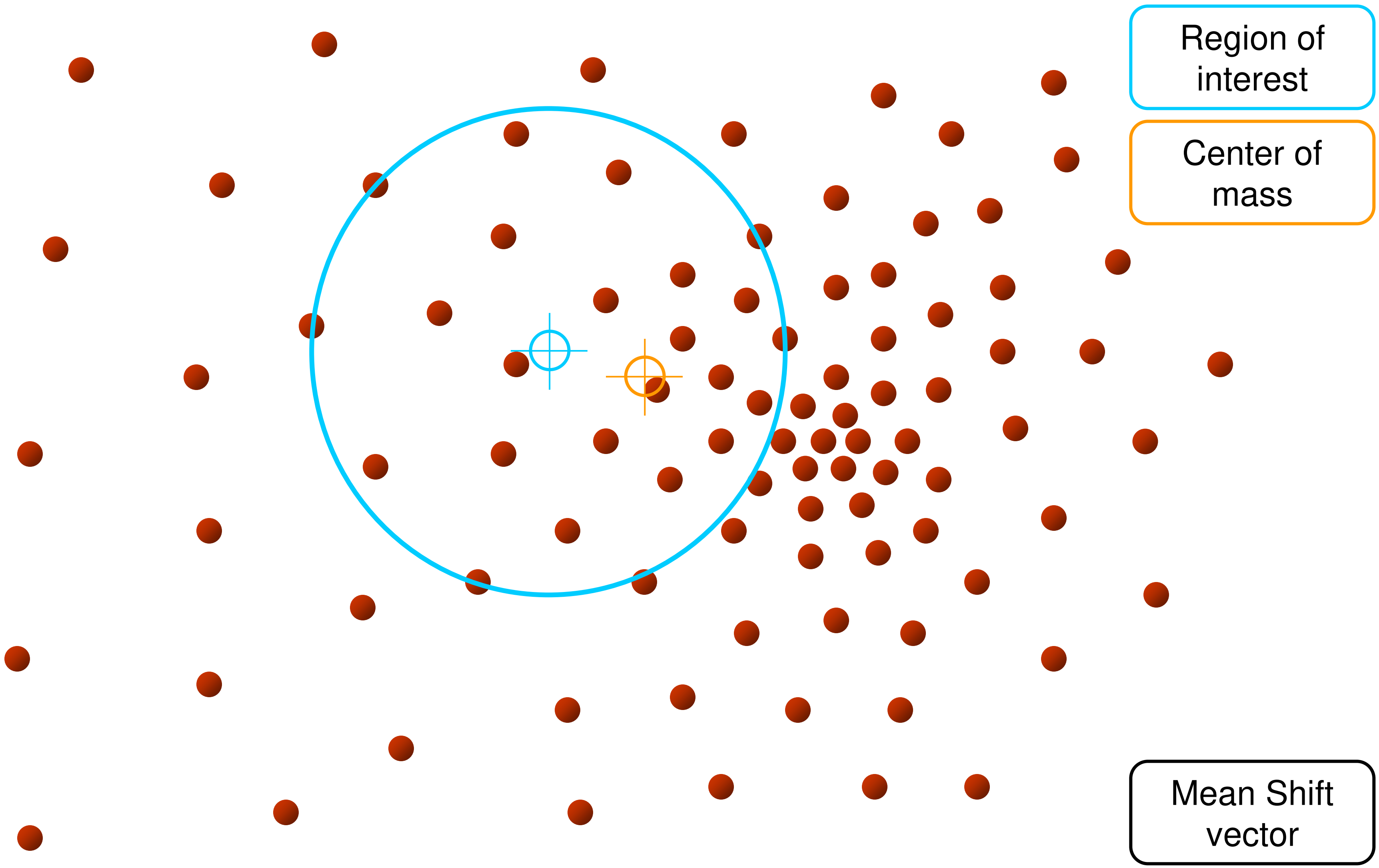


<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Mean Shift

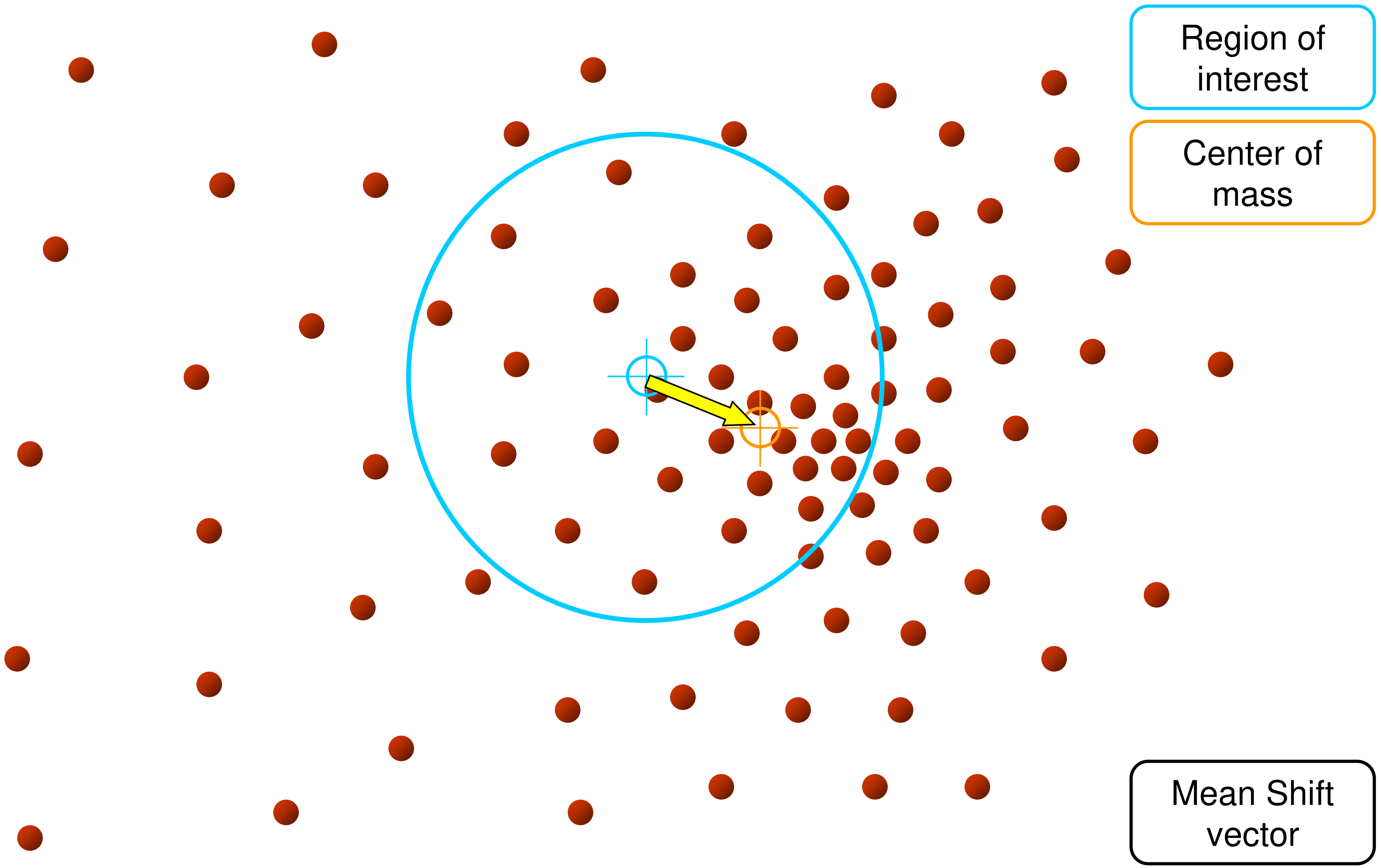


<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Mean Shift

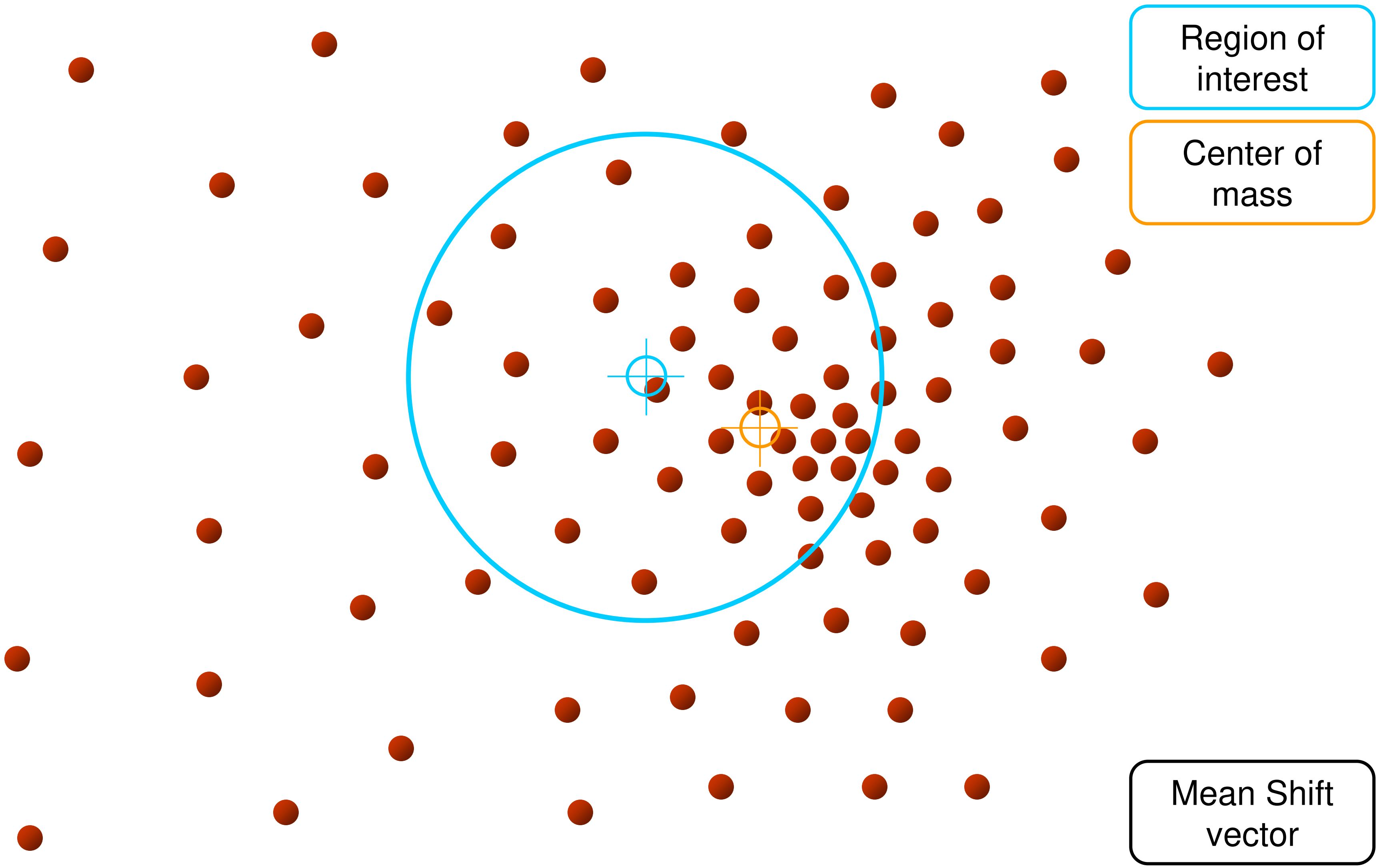


<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

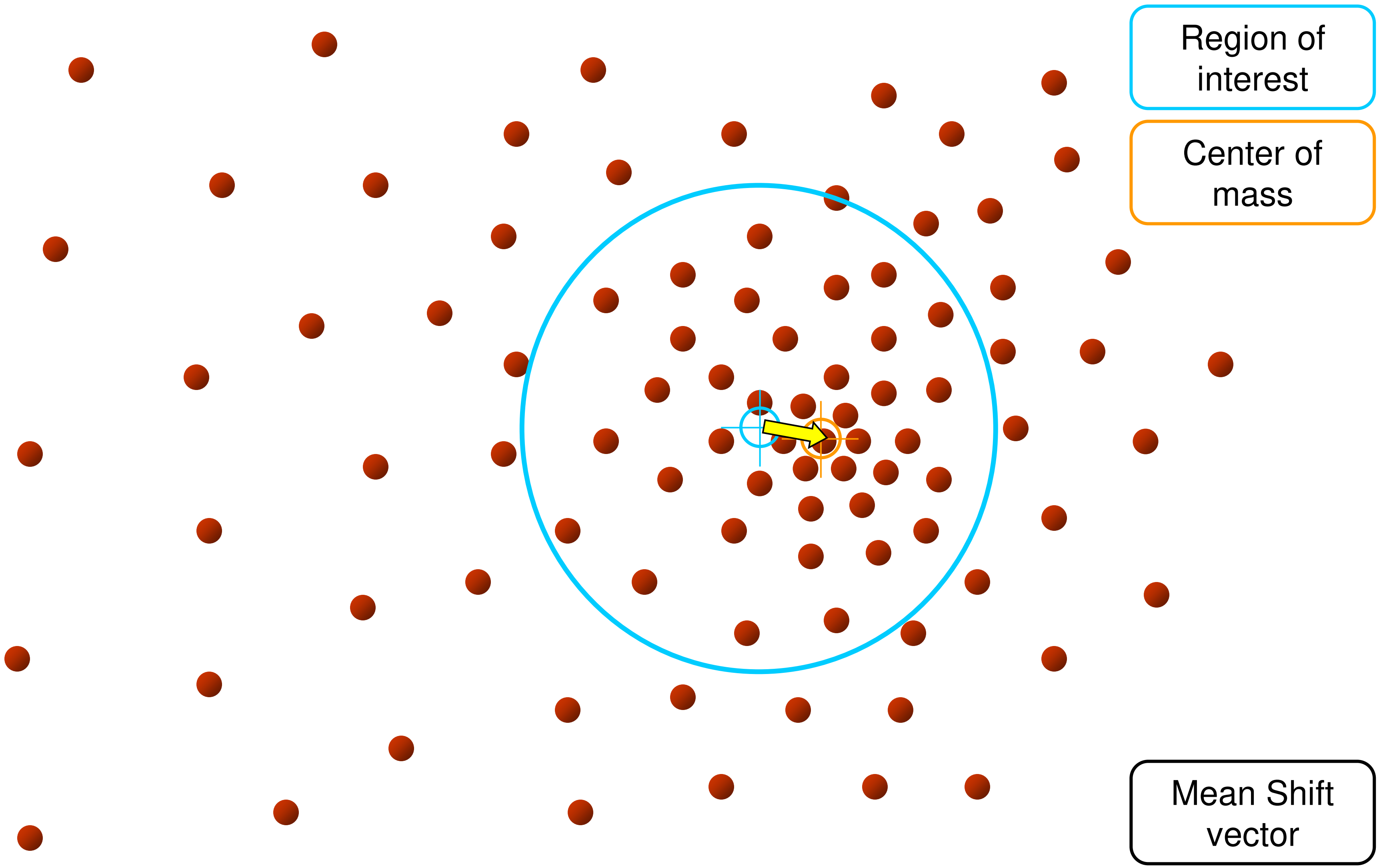
Mean Shift



<u>Objective</u> : Find the densest region
Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Mean Shift

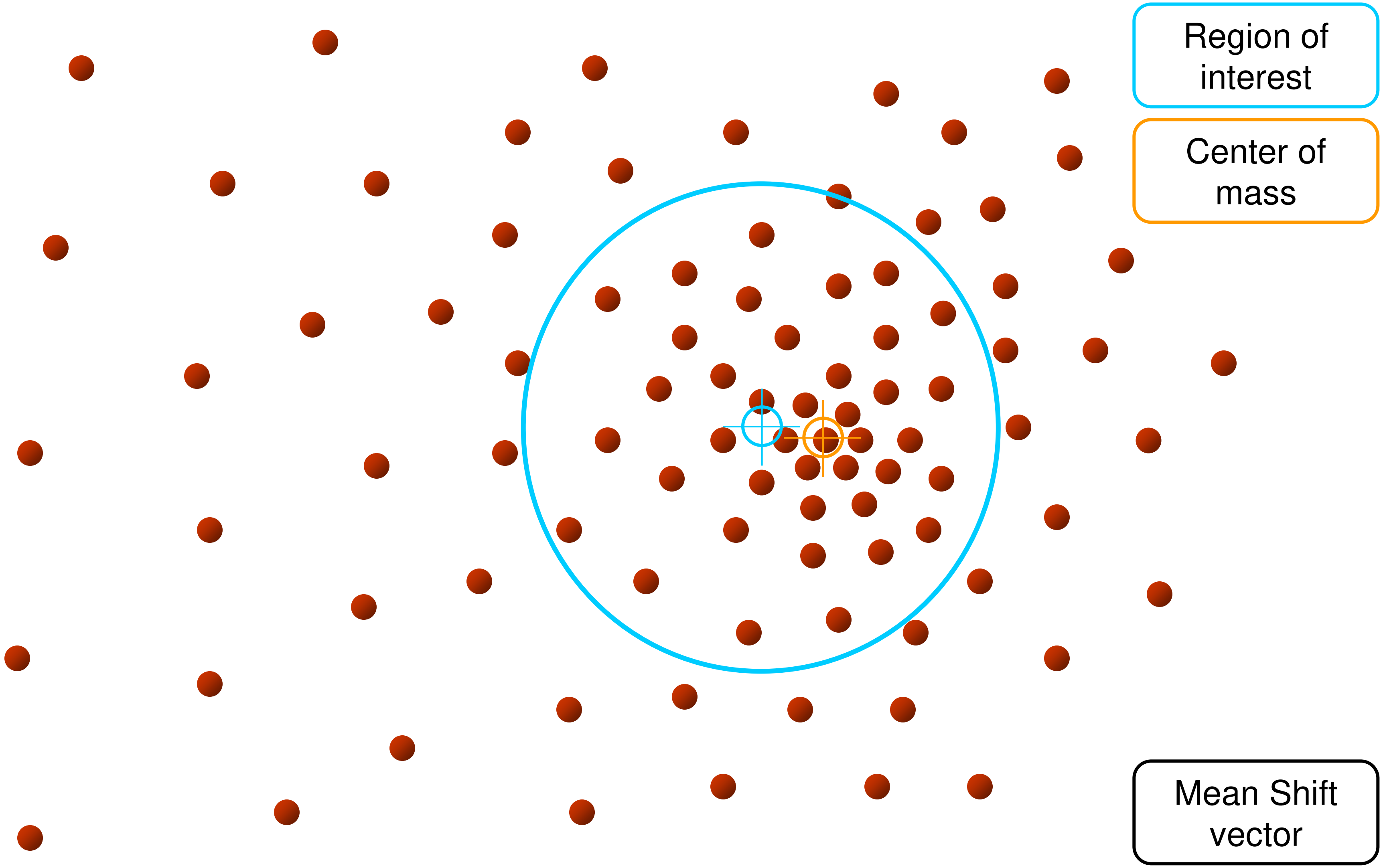


<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Mean Shift

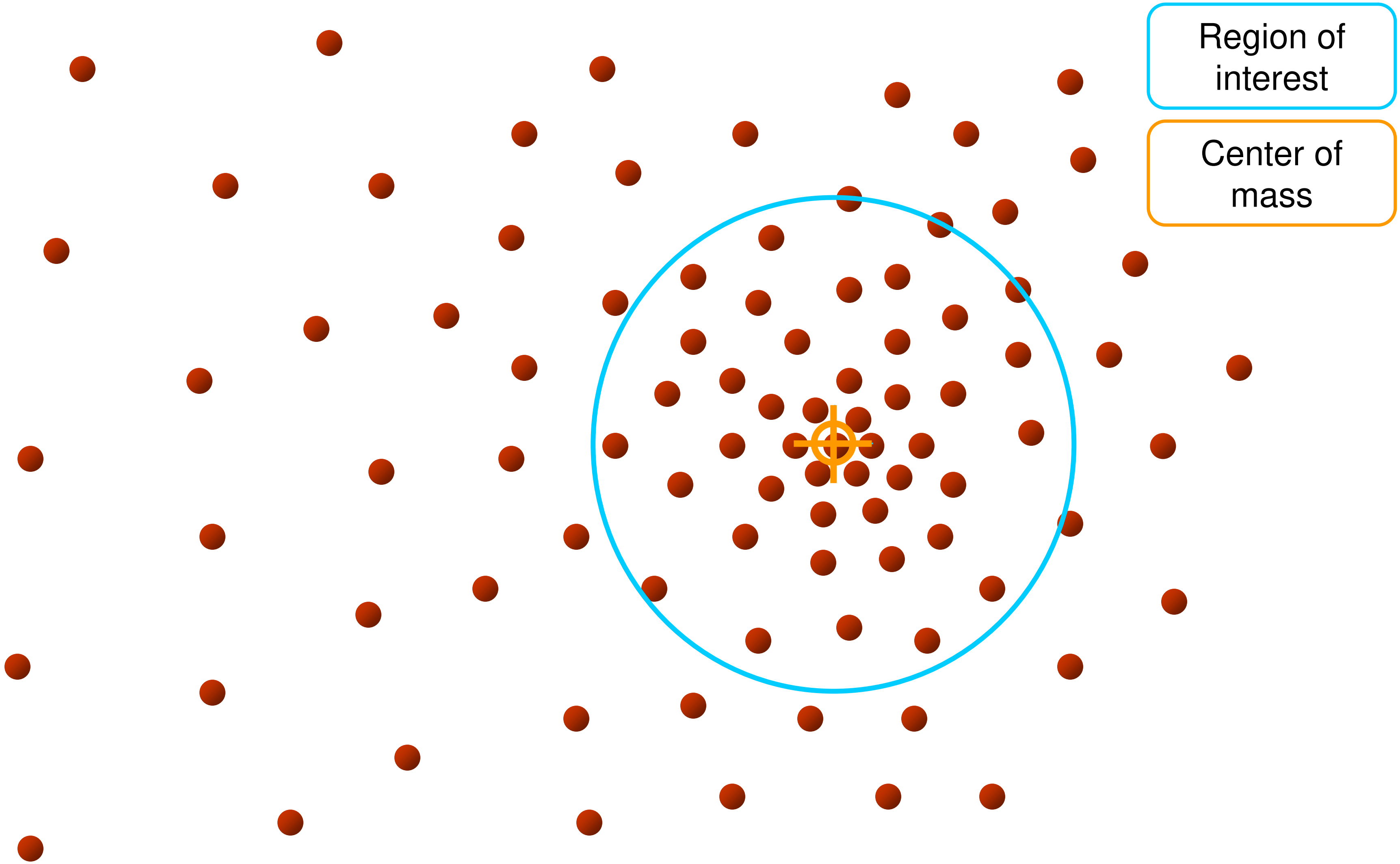


<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Mean Shift



<u>Objective</u> : Find the densest region

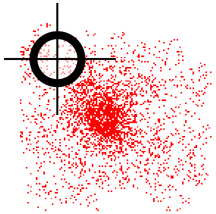
Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Kernel density estimation

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$



Data

In practice one uses the forms:

$$K(\mathbf{x}) = c\prod_{i=1}^{d} k(x_i) \quad \text{or} \quad K(\mathbf{x}) = ck\left(\|\mathbf{x}\|\right)$$

Same function on each dimension

Function of vector length only

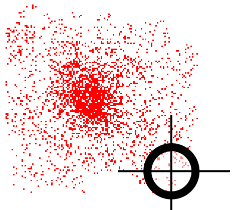
Source: Y. Ukrainitz and B. Sarel

Kernels

UNIVERSITÄT BONN

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

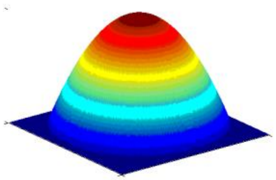
A function of some finite number of data points $x_1 \ldots x_n$



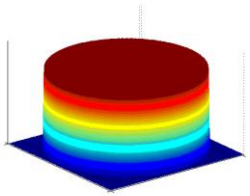
Data

<u>Examples:</u>

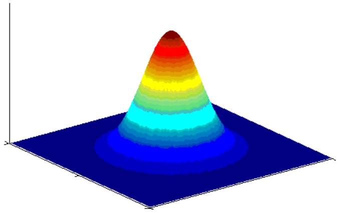
- Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1 - \|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Uniform Kernel $K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Normal Kernel $K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$



Source: Y. Ukrainitz and B. Sarel

Gradient ascent

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x} - \mathbf{x}_i)$$

Using the Kernel form:

$$K(\mathbf{x} - \mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)$$

Size of window

Source: Y. Ukrainitz and B. Sarel

Gradient ascent

UNIVERSITÄT BONN

$$\nabla P(x) = \frac{c}{N} \sum_i \nabla k \left(\|x - x_i\|^2 \right)$$

$$= \frac{c}{N} \sum_i k' \left(\|x - x_i\|^2 \right) 2(x - x_i)$$

$$= \frac{2c}{N} \sum_i g \left(\|x - x_i\|^2 \right) (x_i - x) \qquad \text{Use: } g(x) = -k'(x)$$

$$= \frac{2c}{N} \left(\left\{ \sum_i x_i g \left(\|x - x_i\|^2 \right) \right\} - \left\{ \sum_i g \left(\|x - x_i\|^2 \right) \right\} x \right)$$

$$= \frac{2c}{N} \sum_i g \left(\|x - x_i\|^2 \right) \left(\frac{\sum_i x_i g \left(\|x - x_i\|^2 \right)}{\sum_i g \left(\|x - x_i\|^2 \right)} - x \right)$$

Gradient ascent

$$\nabla P(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla K(\mathbf{x} - \mathbf{x}_i)$$

Using the Kernel form:

$$K(\mathbf{x} - \mathbf{x}_i) = ck\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)$$

Size of window

We get :

$$\nabla P(\mathbf{x}) = \frac{2c}{N} \sum_i g\left(\|x - x_i\|^2 \right) \left(\frac{\sum_i x_i g\left(\|x - x_i\|^2 \right)}{\sum_i g\left(\|x - x_i\|^2 \right)} - x \right)$$

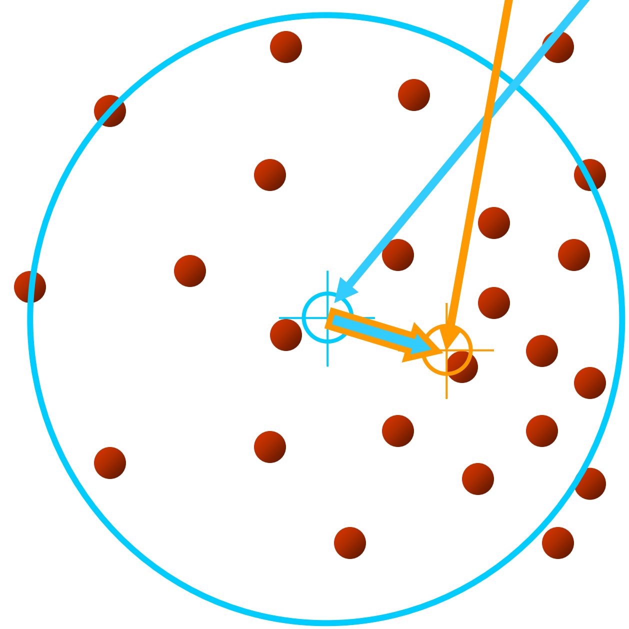
$$\mathrm{g}(\mathbf{x}) = -k'(\mathbf{x})$$

Source: Y. Ukrainitz and B. Sarel

Mean shift

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{2c}{N}\sum_i g\left(\|x - x_i\|^2\right)\left(\frac{\sum_i x_i g\left(\|x - x_i\|^2\right)}{\sum_i g\left(\|x - x_i\|^2\right)} - x\right)$$

Yet another Kernel density estimation !



Simple Mean Shift procedure:

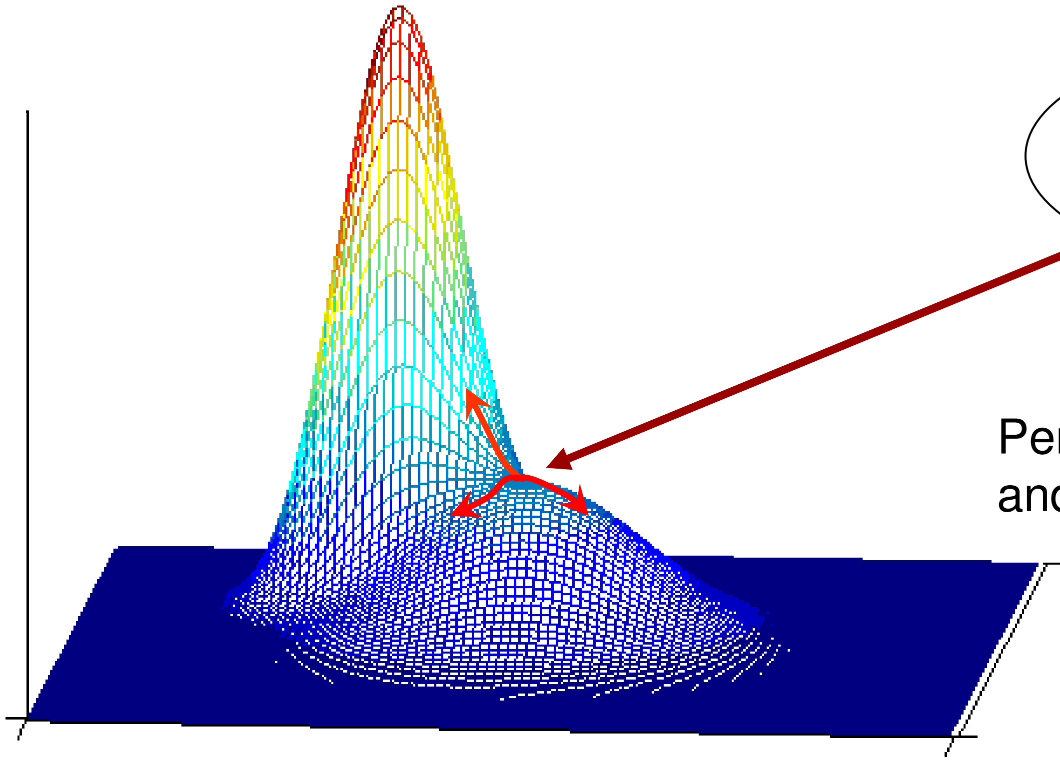
- Compute mean shift vector

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)} - \mathbf{x}\right]$$

- Translate the Kernel window by **m(x)**

Source: Y. Ukrainitz and B. Sarel

Mode detection



What happens if we reach a saddle point ?

Perturb the mode position and check if we return back

<u>Updated Mean Shift Procedure:</u>

- Find all modes using the Simple Mean Shift Procedure
- Prune modes by perturbing them (find saddle points and plateaus)
- Prune nearby – take highest mode in the window

Source: Y. Ukrainitz and B. Sarel

Hough transform: pros and cons

Pros

All points are processed independently, so can cope with occlusion

Some robustness to noise: noise points unlikely to contribute consistently to any single bin

Can detect multiple instances of a model in a single pass

Cons

Complexity of search time increases exponentially with the number of model parameters

Non-target shapes can produce spurious peaks in parameter space

Quantization: hard to pick a good grid size

Source: K. Grauman

Generalized Hough transform

What if want to detect arbitrary shapes defined by boundary points and a reference point?

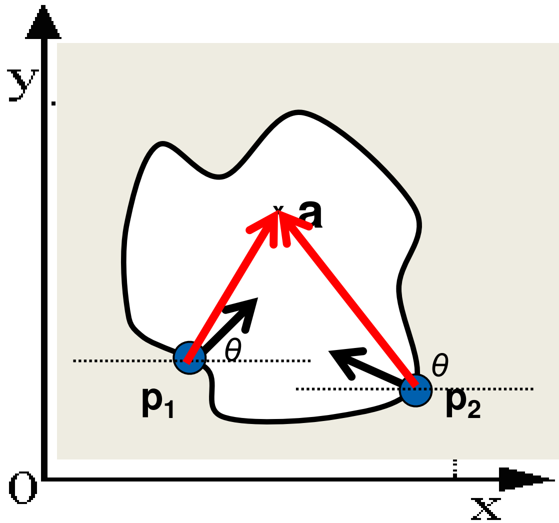


Image space

At each boundary point, compute displacement vector: $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$.

For a given model shape: store these vectors in a table indexed by gradient orientation θ.

[D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

Source: K. Grauman

Generalized Hough transform

UNIVERSITÄT BONN

To *detect* the model shape in a new image:

For each edge point

– Index into table with its gradient orientation θ

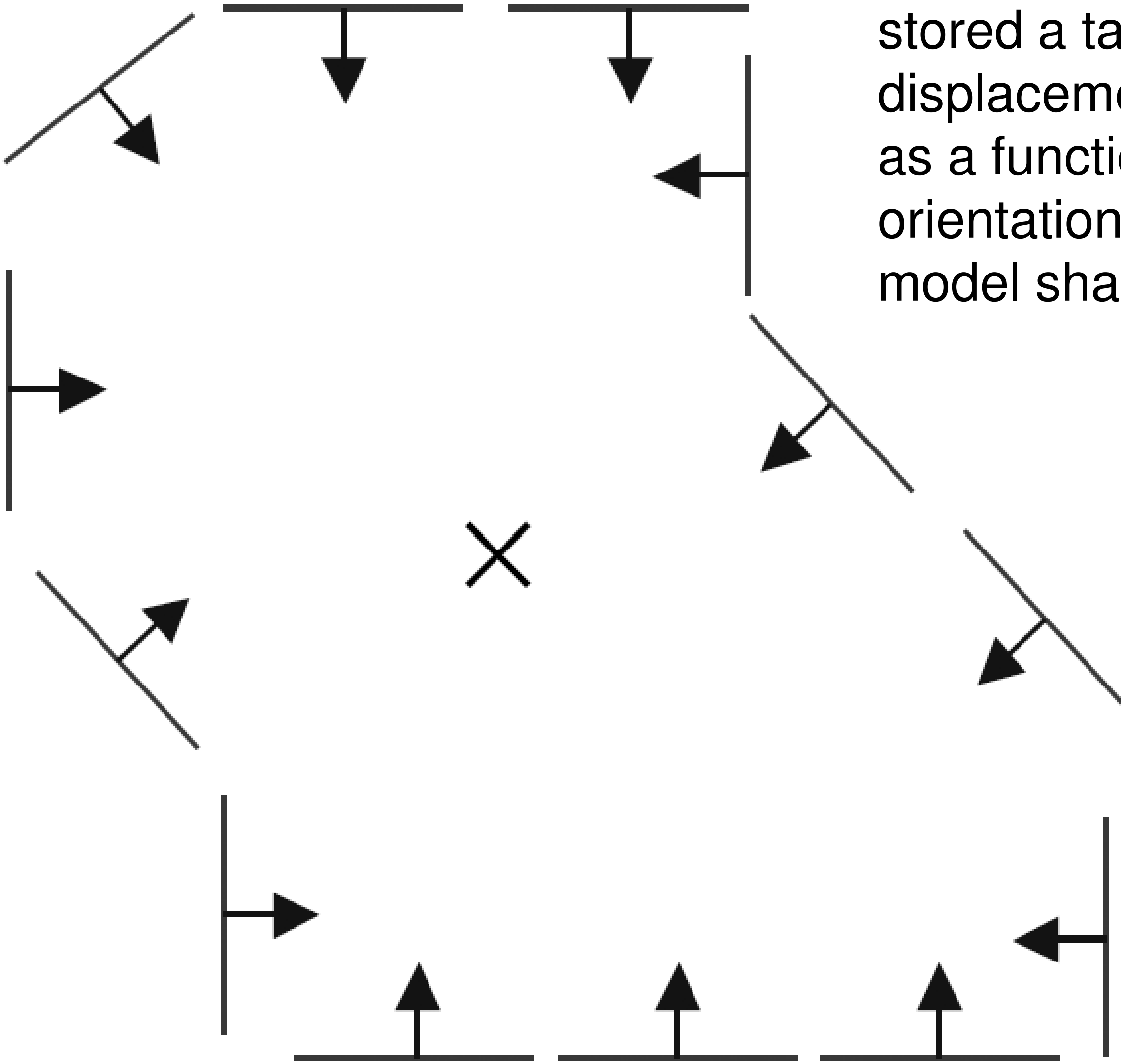
– Use retrieved **r** vectors to vote for position of reference point

Peak in this Hough space is reference point with most supporting edges

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

Source: K. Grauman

Example

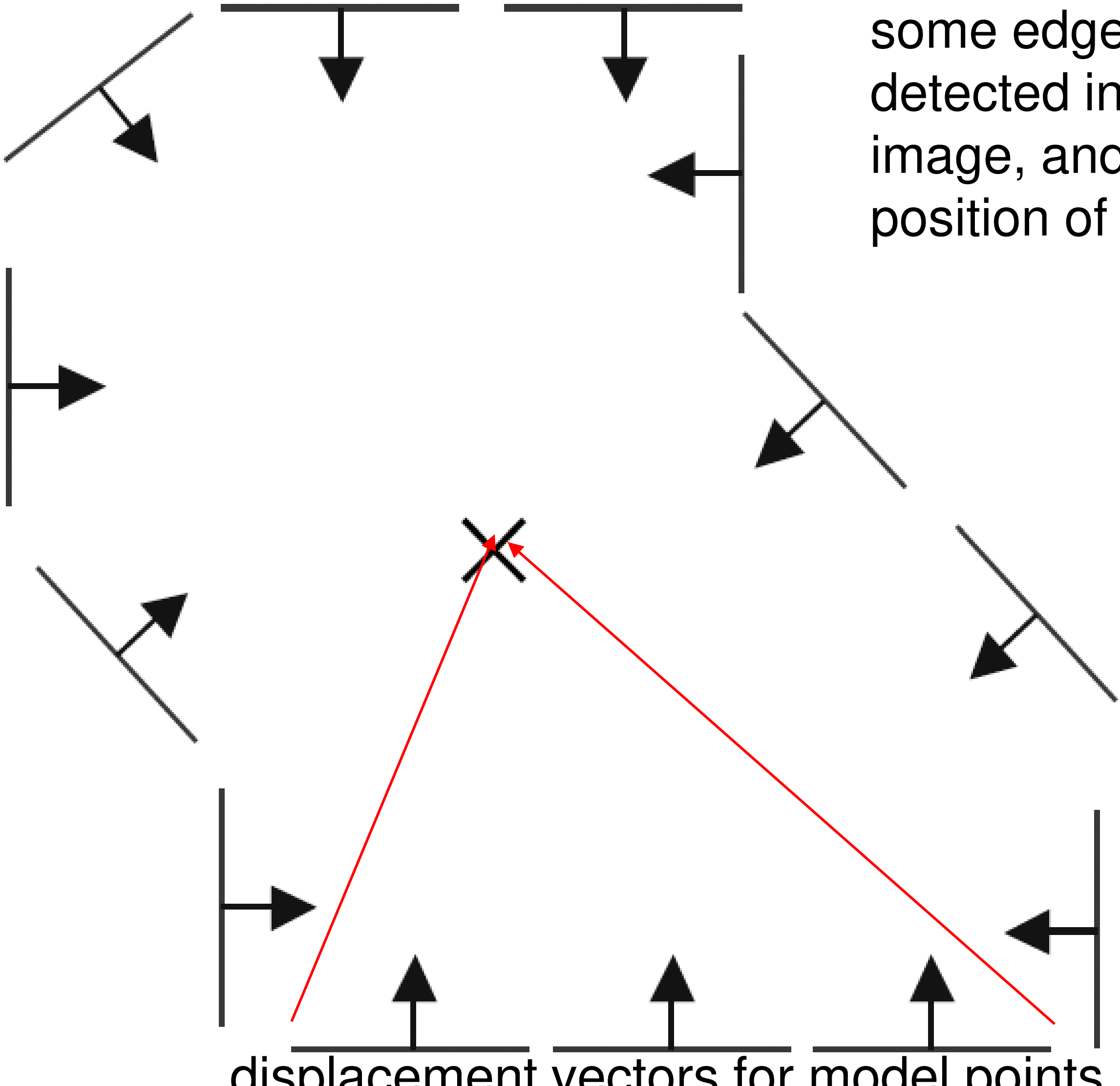


model shape

Say we've already stored a table of displacement vectors as a function of edge orientation for this model shape.

Source: S. Lazebnik

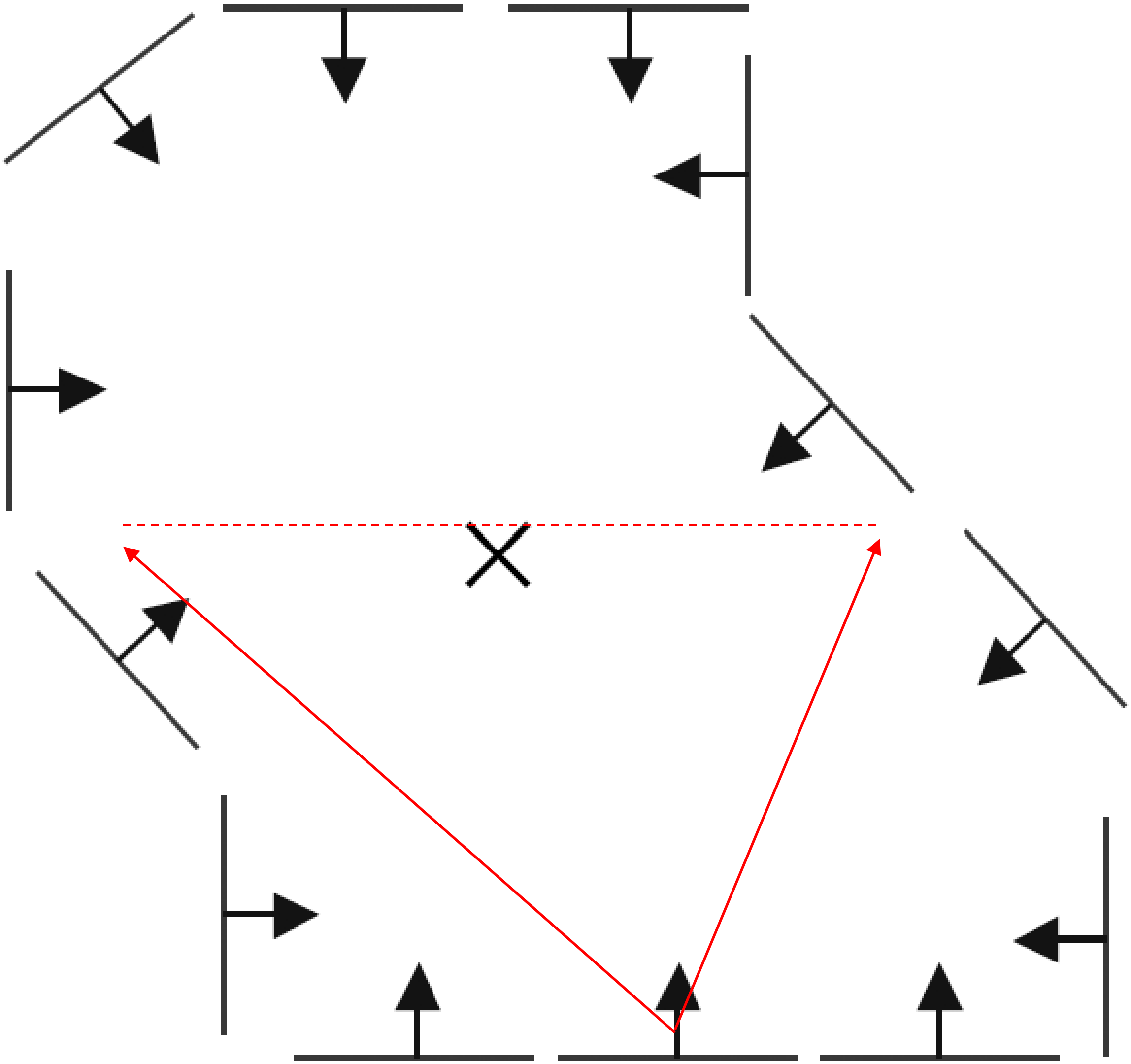
Example



Now we want to look at some edge points detected in a *new* image, and vote on the position of that shape.

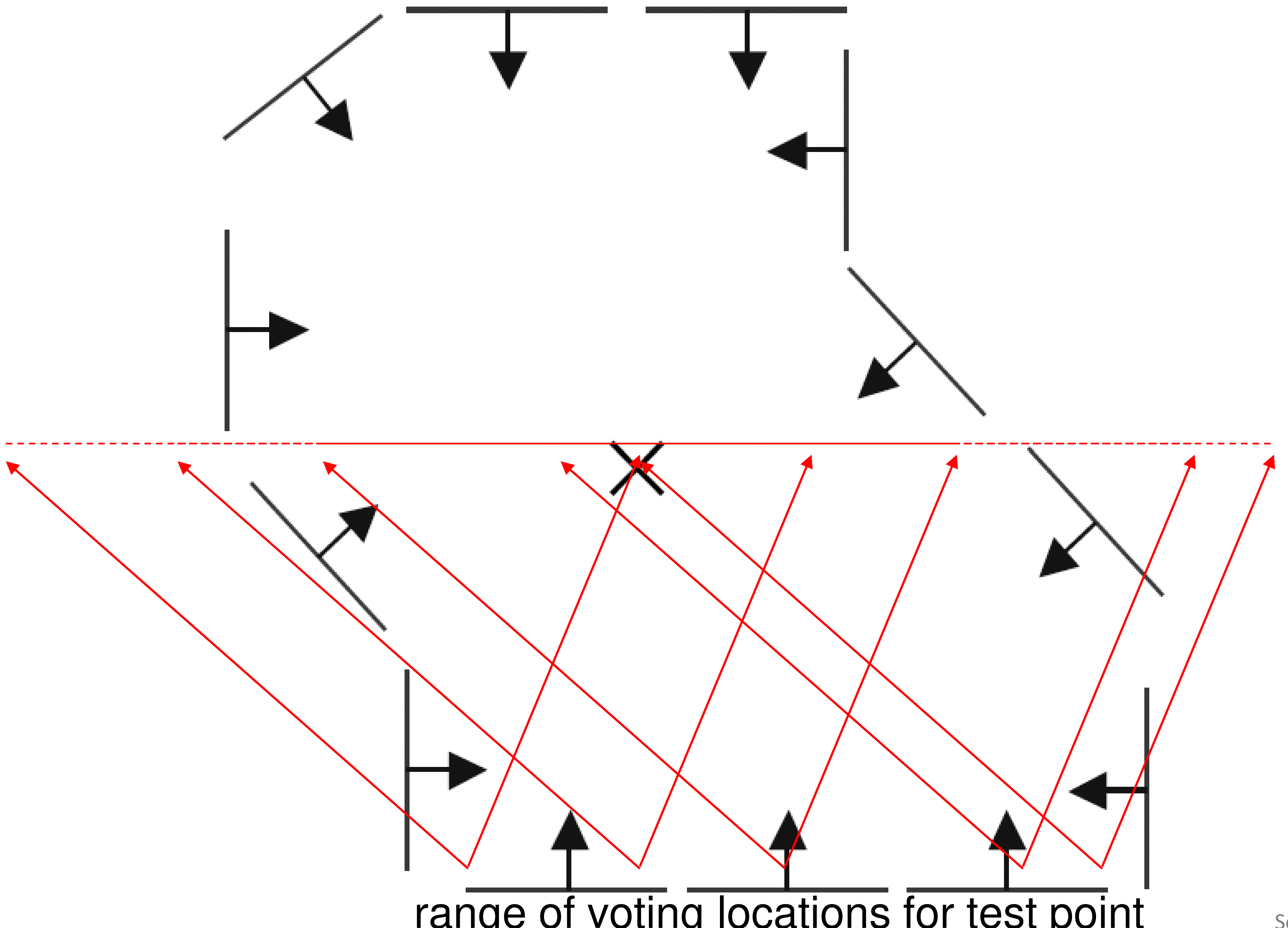
Source: S. Lazebnik

Example



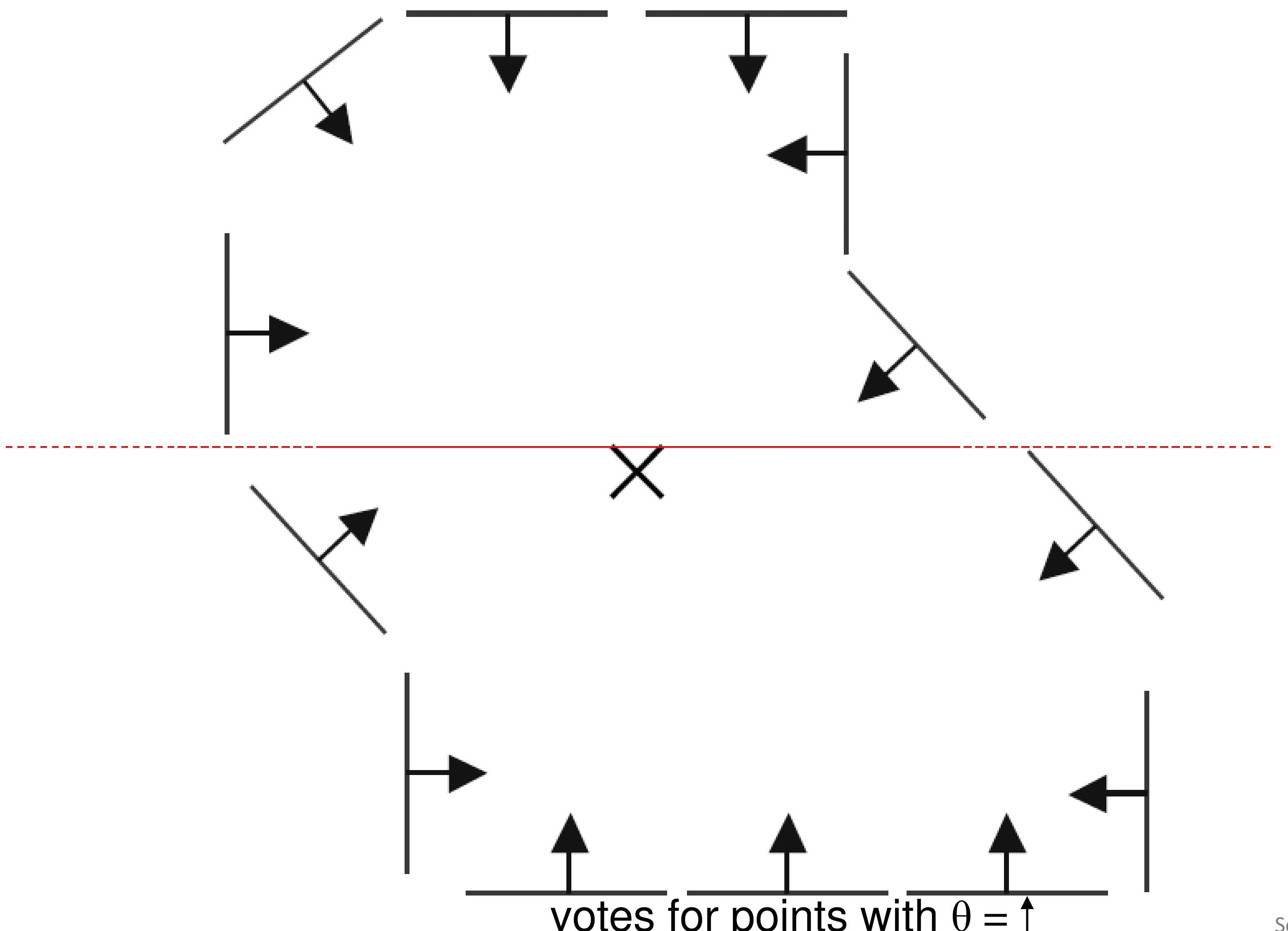
Source: S. Lazebnik

Example



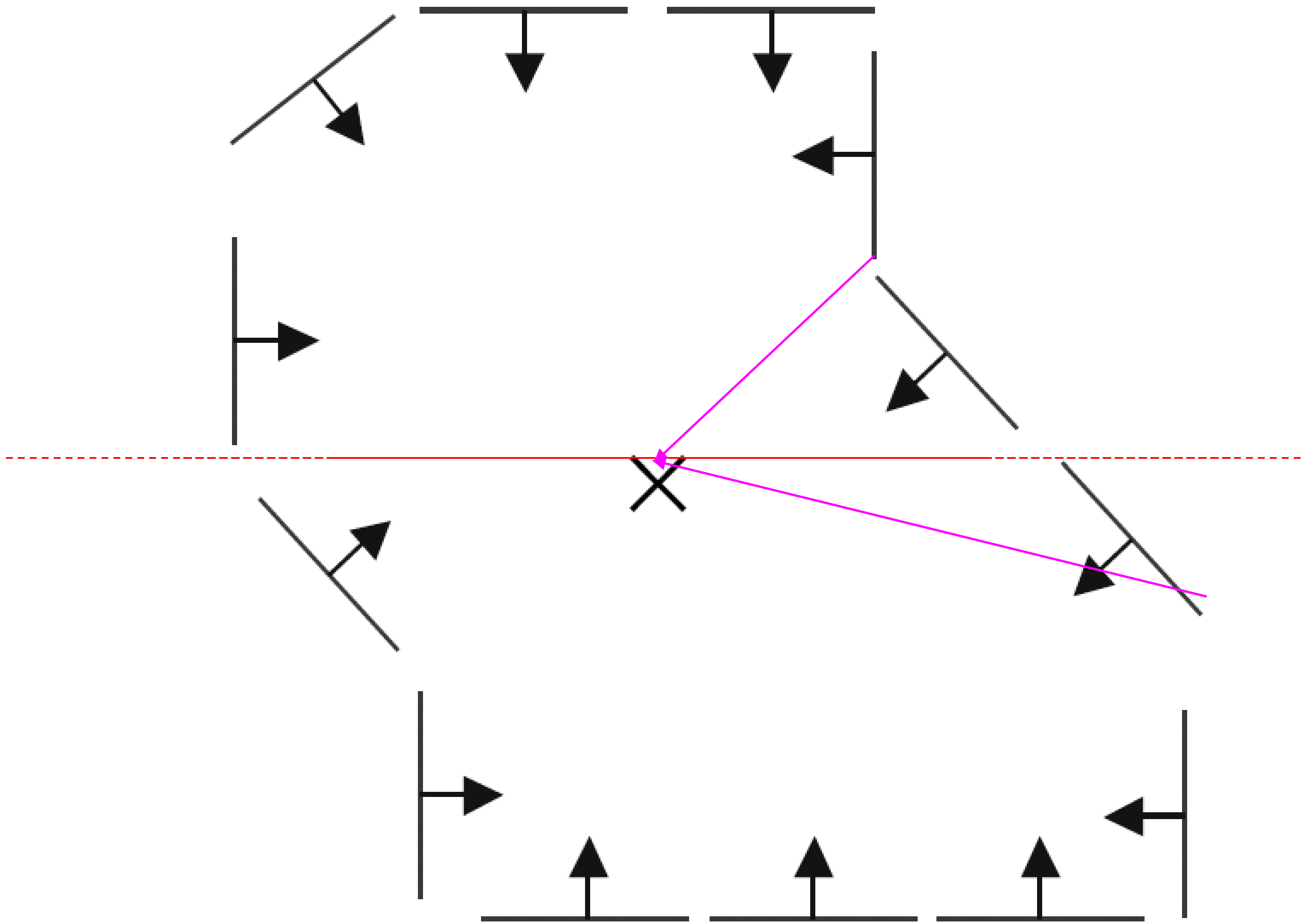
Source: S. Lazebnik

Example



Source: S. Lazebnik

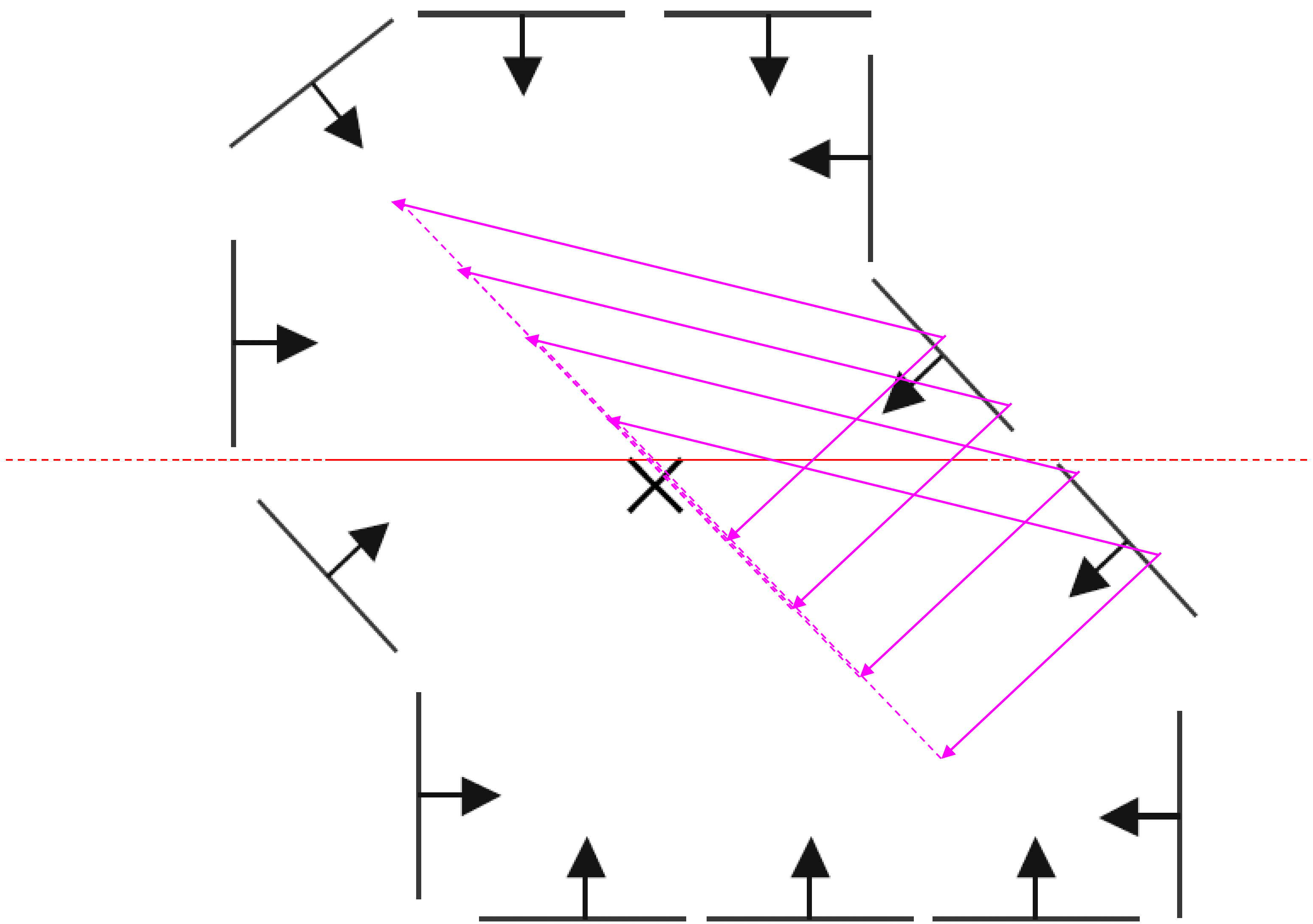
Example



displacement vectors for model points

Source: S. Lazebnik

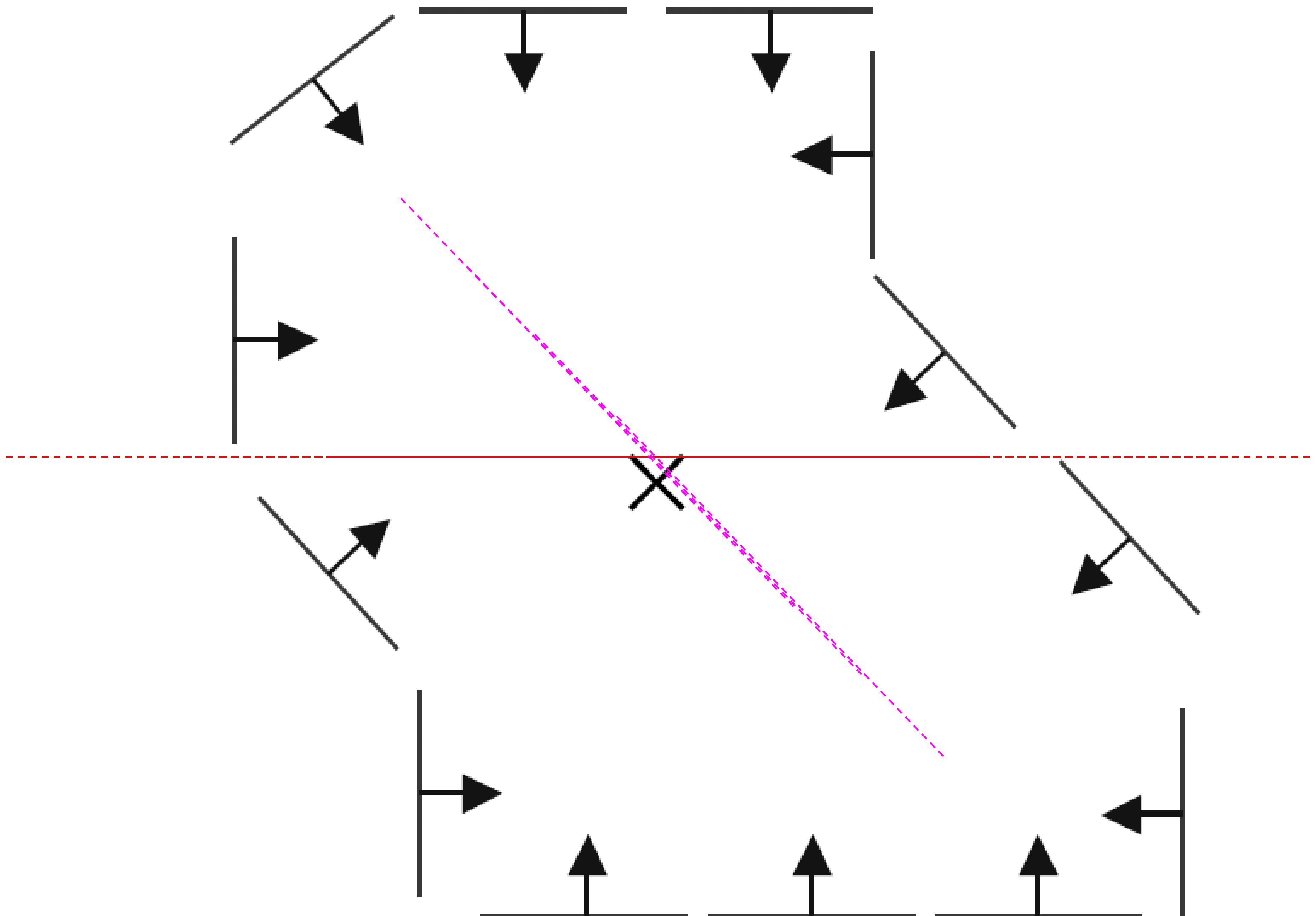
Example



range of voting locations for test point

Source: S. Lazebnik

Example



votes for points with $\theta = \swarrow$

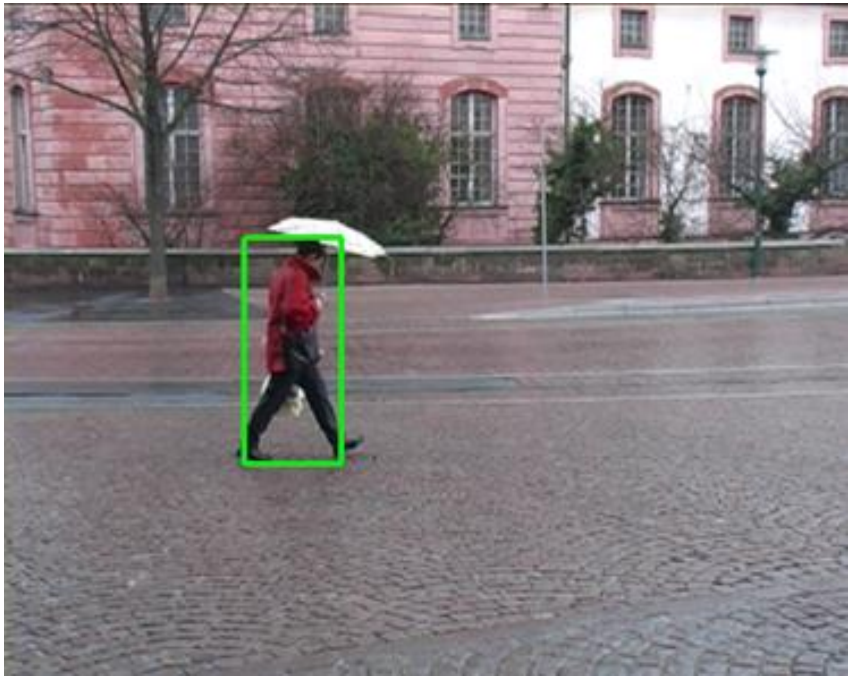
Source: S. Lazebnik

Hough voting for object detection

- Parts of an object provide useful spatial information
- Machine Learning: learn how to vote

Gall et al. **Hough Forests for Object Detection, Tracking, and Action Recognition.** IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 11, 2188-2202, 2011.

Hough voting for object detection



Gall et al. **Hough Forests for Object Detection, Tracking, and Action Recognition.** IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 11, 2188-2202, 2011.