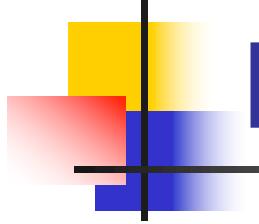


Feature detection and Model Fitting

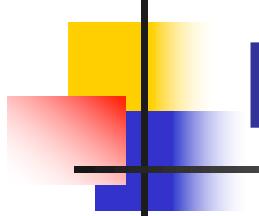
Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.



Key Feature Points



- Intensity (Maximum or Minimum)
 - Edges
 - Corners
-
- Noise
 - Orientation
 - Scale

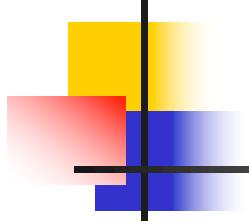


Harris corner detector

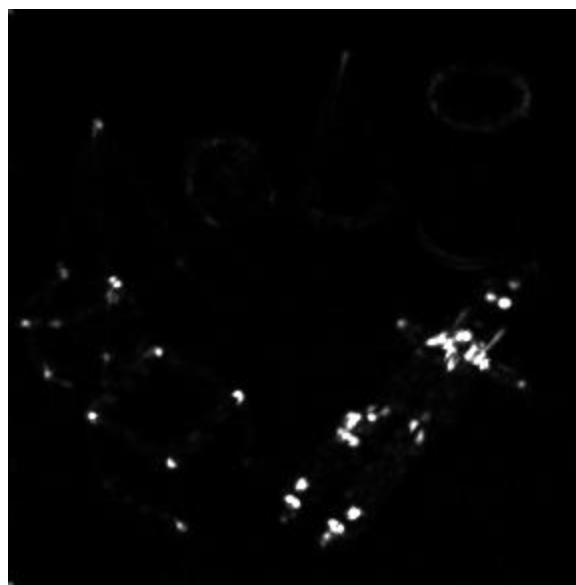
$$H = \begin{bmatrix} \overline{f_x^2} & \overline{f_x f_y} \\ \overline{f_y f_x} & \overline{f_y^2} \end{bmatrix}$$

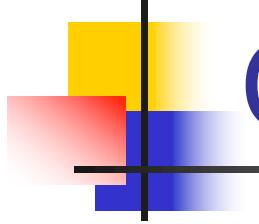
Algorithm:

1. Compute $hm = \det(H)/\text{trace}(H)$.
2. Retain $hm > \text{threshold}$.
3. Select local maxima as key-points.



Examples: Harris corner points





Other key-point extractor

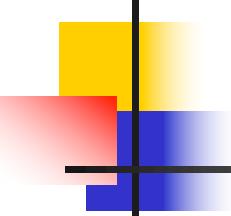
- Difference of Gaussian (DoG)
- Measures from Hessian Matrix

$$g(x, y) = \sigma \cdot f(x, y) * \frac{\partial G_\sigma(x, y)}{\partial \sigma}$$

Local Maxima

$$H = \begin{bmatrix} \bar{f}_{xx} & \bar{f}_{xy} \\ \bar{f}_{yx} & \bar{f}_{yy} \end{bmatrix} \rightarrow \frac{(Tr(H))^2}{Det(H)}$$

Smaller values



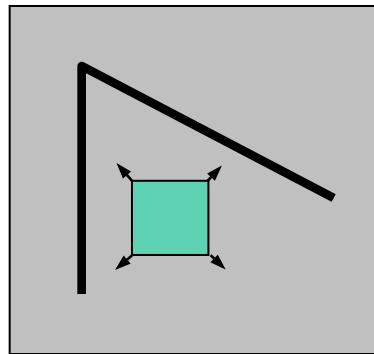
Feature Descriptors

- Scale Invariant Feature Transform (SIFT)
- Speeded Up Robust Feature (SURF)
- Histogram of Gradients (HOG)
- Accumulate statistics of neighboring gradients.
- Final descriptor is a multidimensional feature vector.
- Feature vectors are used in classification / similarity matches.

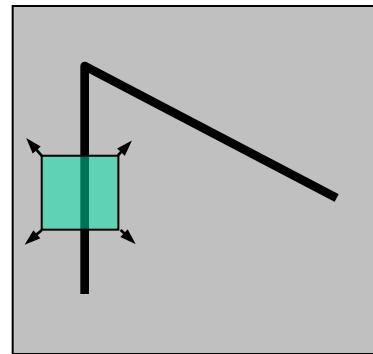
Feature detection (Chapter 4 - Szeleski)

Local measure of feature uniqueness

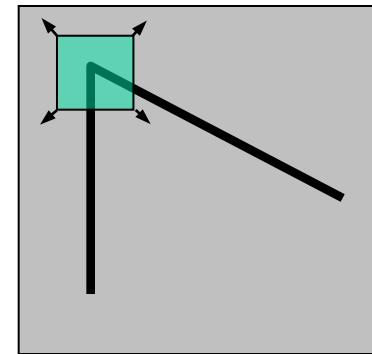
- How does the window change when you shift it?
- Shifting the window in *any direction* may cause a *big change*.



“flat” region:
no change in
all directions



“edge”: no
change along
the edge
direction

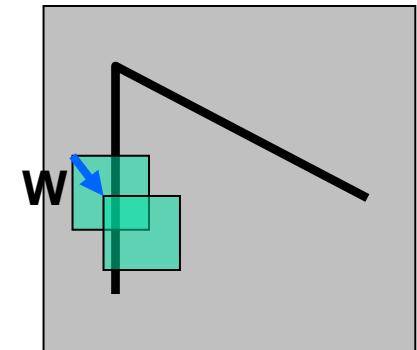


“corner”:
significant
change in all
directions

Feature detection

Consider shifting the window \mathbf{W} by (u, v)

- how do the pixels in \mathbf{W} change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” of $E(u, v)$:



$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

Small motion assumption

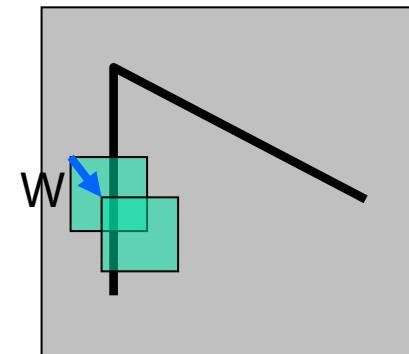
$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

For small u and v

$$\begin{aligned} I(x + u, y + v) &\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

Feature detection



$$\begin{aligned}
 I(x + u, y + v) &\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\
 &\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}
 \end{aligned}$$

$$\begin{aligned}
 E(u, v) &= \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\
 &\approx \sum_{(x,y) \in W} [I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x, y)]^2 \\
 &\approx \sum_{(x,y) \in W} [[I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}]^2
 \end{aligned}$$

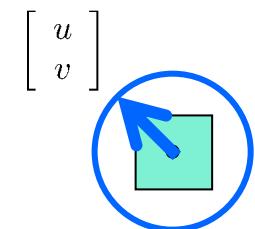
$$E(u, v) = \sum_{(x,y) \in W} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u, v) = \sum_{(x,y) \in W} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Feature detection

$$E(u, v) = \sum_{(x,y) \in W} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

H



For the example above

- You can move the center of the green window to anywhere on the blue unit circle
- Which directions will result in the largest and smallest E values?
- We can find these directions by looking at the eigenvectors of H .

Quick eigenvalue/eigenvector

review

$$Ax = \lambda x$$

The **eigenvectors** of a matrix \mathbf{A} are the vectors \mathbf{x} that satisfy:

$$\det(A - \lambda I) = 0 \quad \det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

The scalar λ is the **eigenvalue** corresponding to \mathbf{x}

- The eigenvalues are found by solving:
- In our case, $\mathbf{A} = \mathbf{H}$ is a 2x2 matrix, so we have

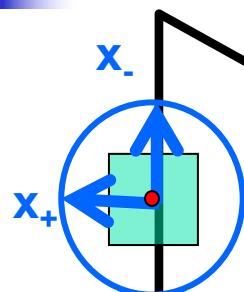
- The solution:
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

Once you know λ , you find \mathbf{x} by solving

$$E(u, v) = \sum_{(x,y) \in W} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

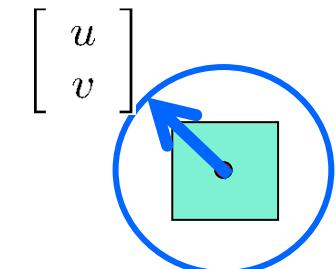
Feature detection



$$\begin{aligned} Hx_+ &= \lambda_+ x_+ \\ Hx_- &= \lambda_- x_- \end{aligned}$$

$\underbrace{\qquad\qquad\qquad}_{H}$

H



Eigenvalues and eigenvectors of H

- Define shifts with the smallest and largest change (E value)
- x_+ = direction of **largest** increase in E .
- λ_+ = amount of increase in direction x_+
- x_- = direction of **smallest** increase in E .
- λ_- = amount of increase in direction x_+

Feature detection

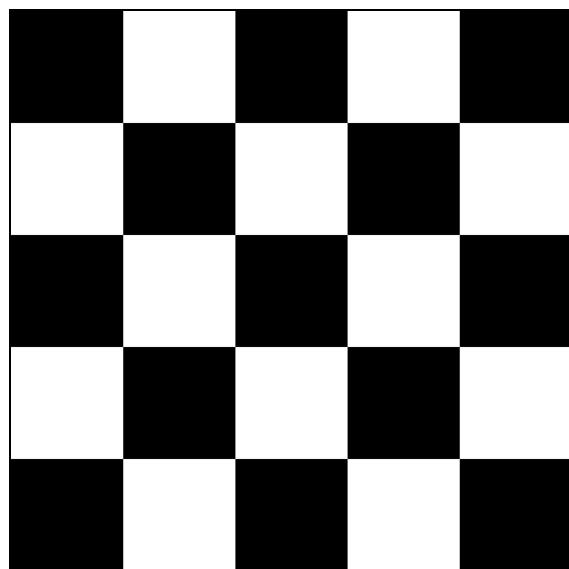
How are λ_+ , \mathbf{x}_+ , λ_- , and \mathbf{x}_- relevant for feature detection?

- What's our feature scoring function?

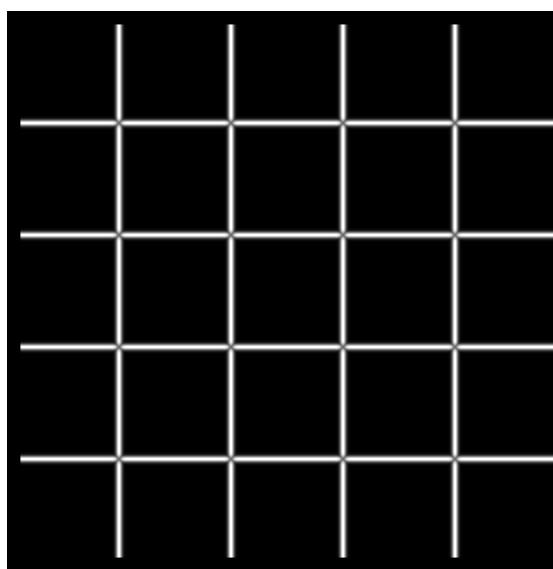
Want $E(u, v)$ to be *large* for small shifts in **all** directions

- the *minimum* of $E(u, v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_-) of H

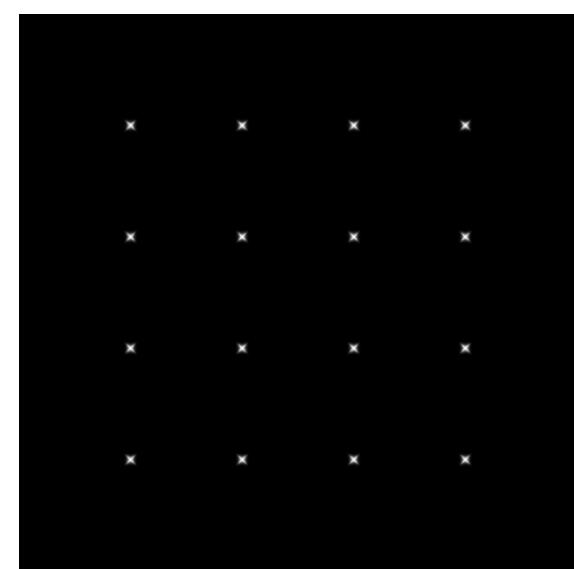
Feature detection



I



λ_+



λ_-



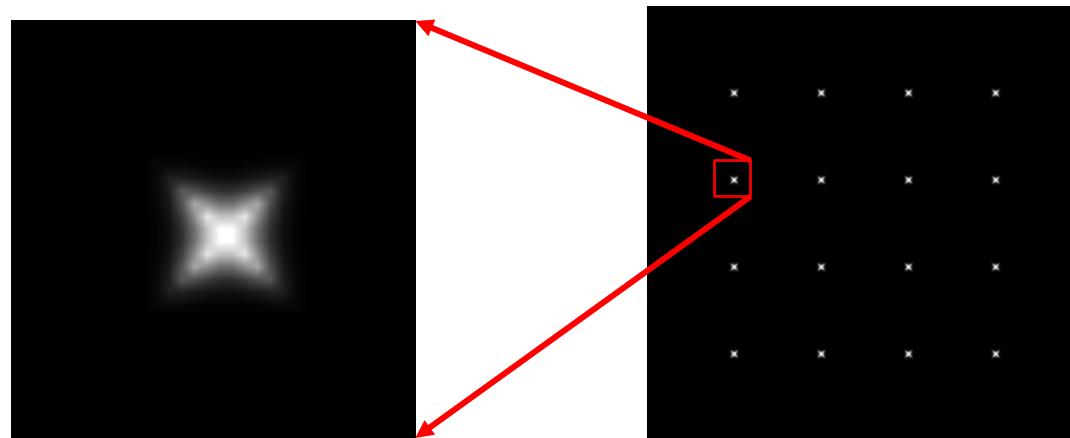
Feature detection summary

Here's what we do

- Compute the gradient at each point in the image
- Create the H matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ($\lambda_>$ threshold)
- Choose those points where $\lambda_>$ is a local maximum as features

Feature detection summary

- Choose those points where λ_- is a local maximum as features



The Harris operator

λ_- is a variant of the “Harris operator” for feature detection.

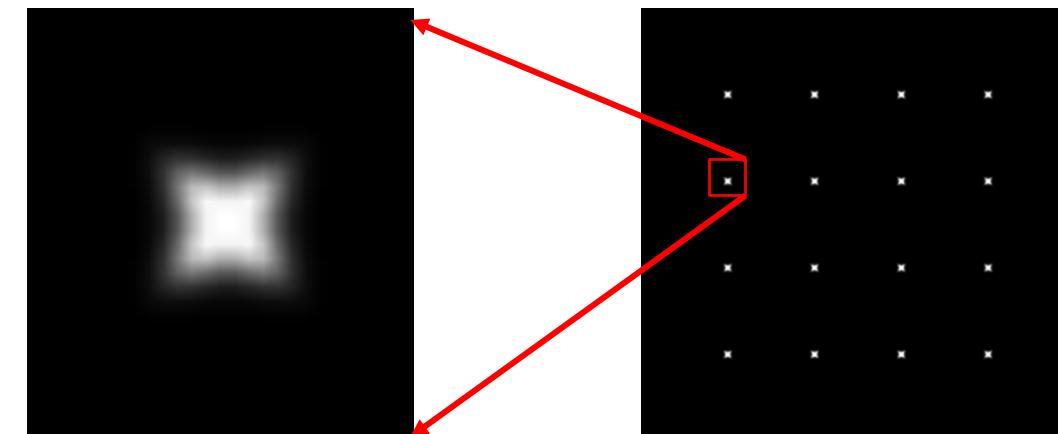
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$
$$= \frac{\text{determinant}(H)}{\text{trace}(H)}$$

- The *trace* is the sum of the diagonals, i.e.,

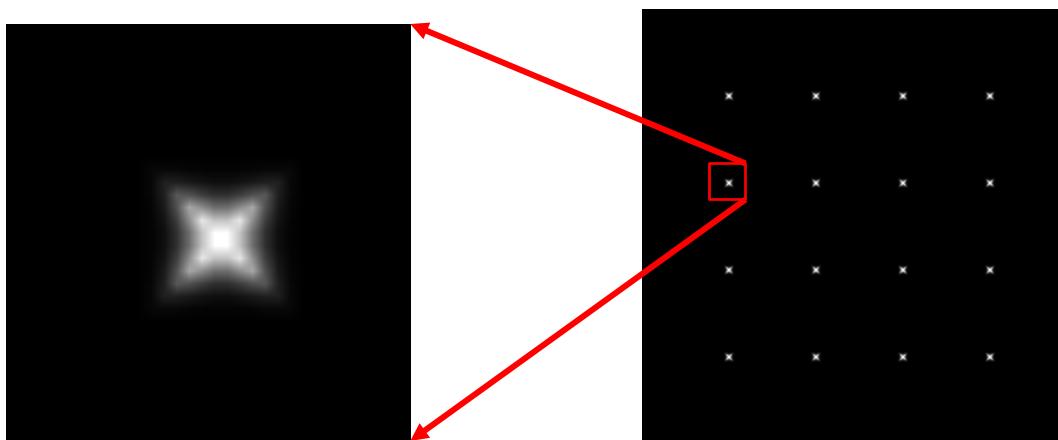
$$\text{trace}(H) = h_{11} + h_{22}$$

- Very similar to λ_- but less expensive (no square root)
- Called the “Harris Corner Detector” or “Harris Operator”
- Lots of other detectors, this is one of the most popular.

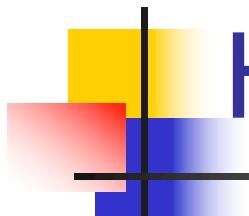
The Harris operator



Harris
operator

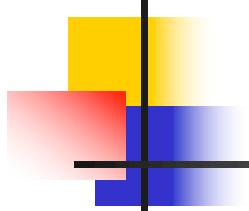


λ_-

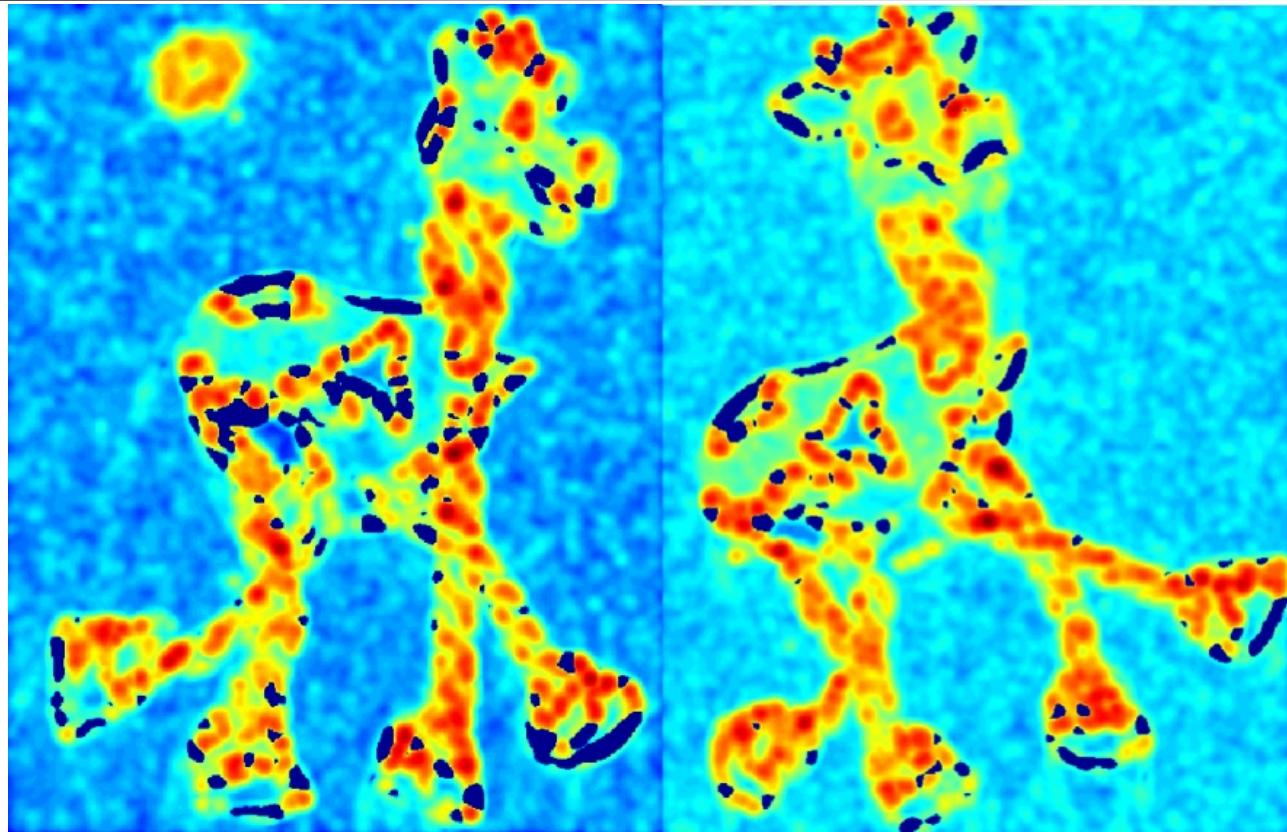


Harris detector example

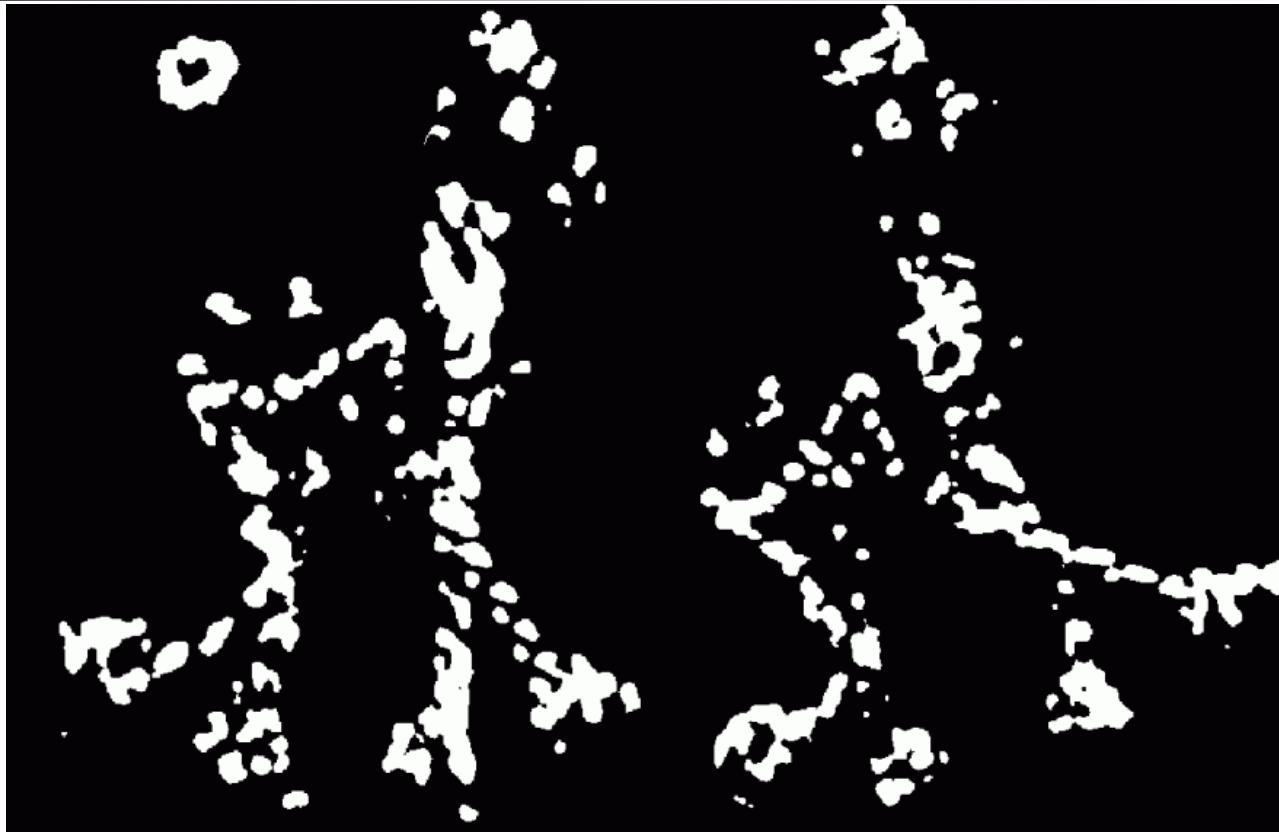


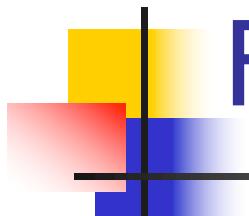


f value (red high, blue low)



Threshold ($f > \text{value}$)





Find local maxima of f

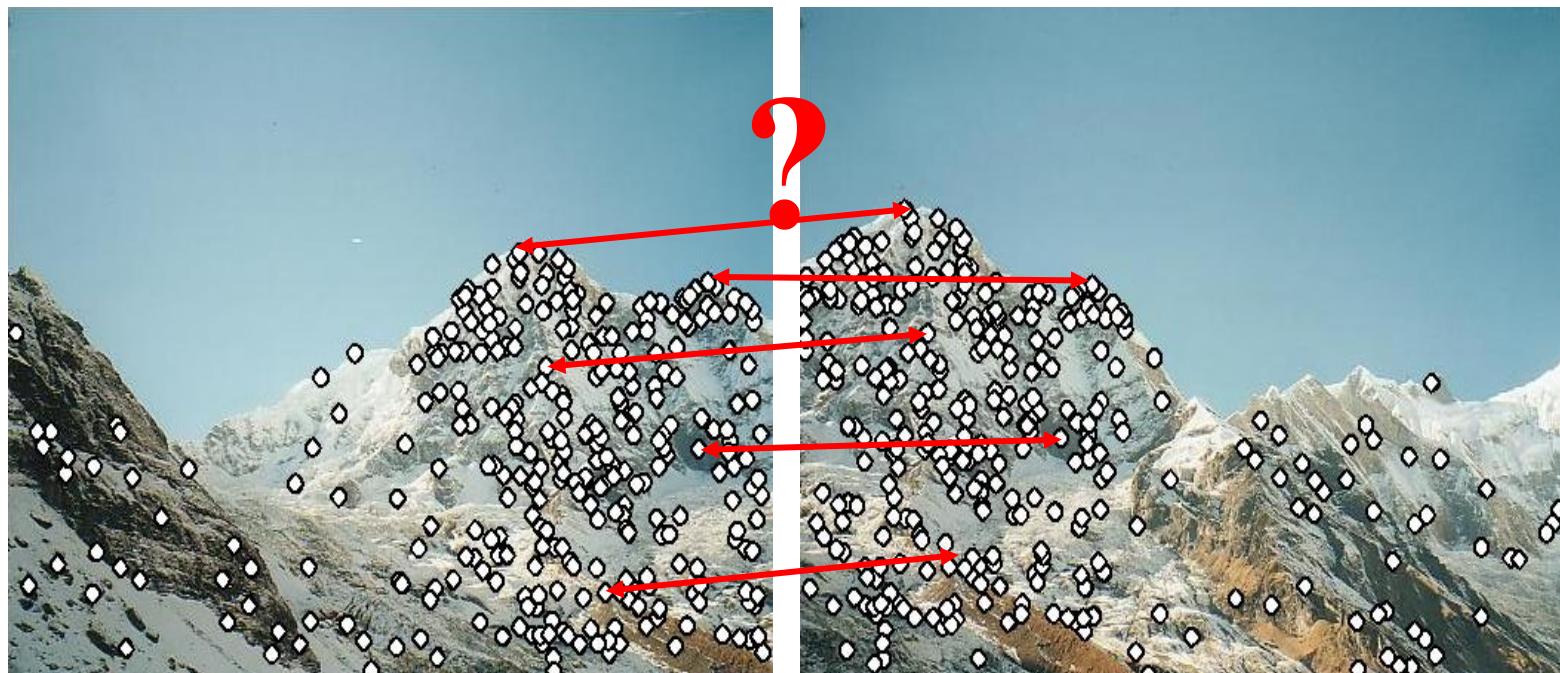


Harris features (in red)

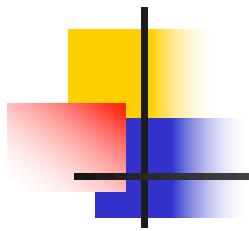


Feature Matching

We know how to detect good points
Next question: **How to match them?**



Describe feature points: Feature descriptor.



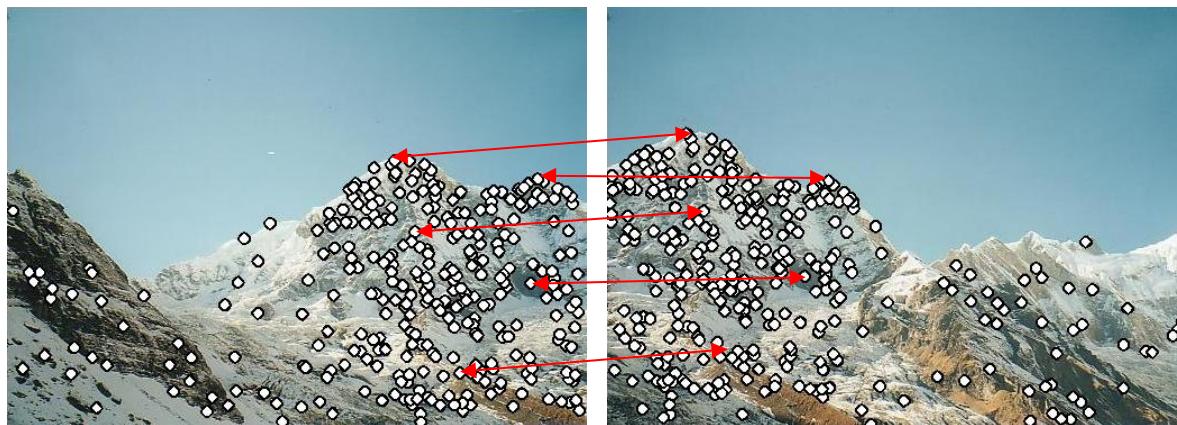
Matching with Features

- Detect feature points in both images



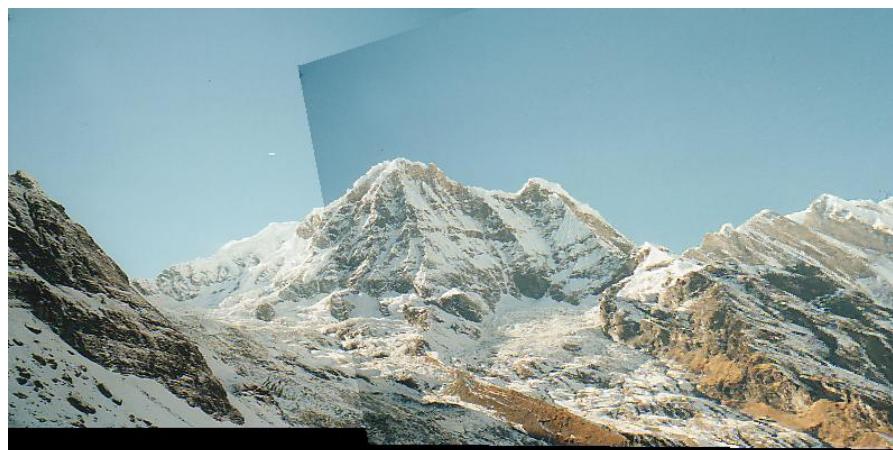
Matching with Features

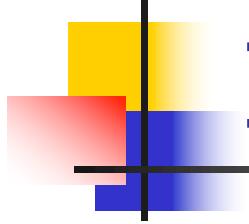
- Detect feature points in both images
- Find corresponding pairs



Matching with Features

- Detect feature points in both images.
- Find corresponding pairs.
- Use these pairs to align images.





Invariance

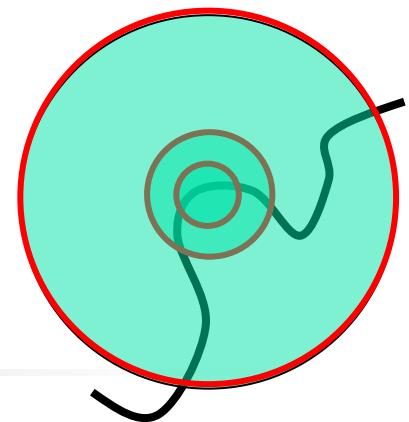
Suppose you **rotate** the image by some angle

- Will you still pick up the same features?

What if you change the brightness?

Scale?

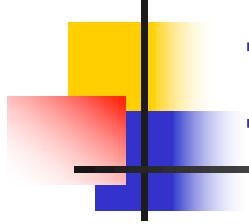
Scale invariant detection



Suppose you're looking for corners

Key idea: find scale that gives local maximum of f

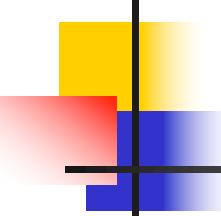
- f is a local maximum in both position and scale
- Common definition of f : Laplacian
(or difference between two Gaussian filtered images with different s.d.'s.).



Invariance

Suppose we are comparing two images I_1 and I_2

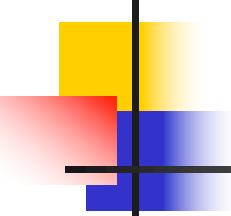
- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?



Invariance

We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
 - Translation, 2D rotation, scale
- They can usually also handle
 - Limited 3D rotations (SIFT works up to about 60 degrees)
 - Limited affine transformations (some are fully affine invariant)
 - Limited illumination/contrast changes



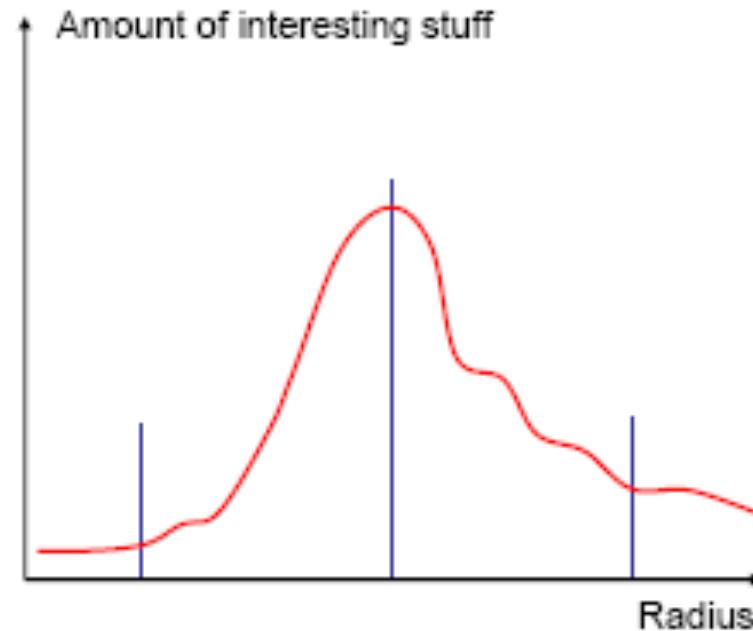
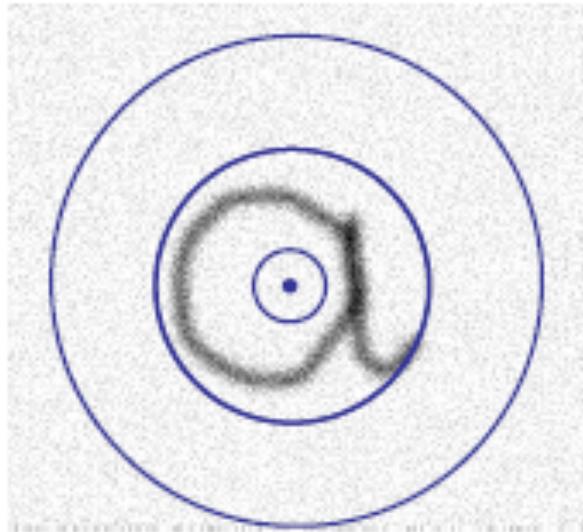
How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
 - Harris is invariant to translation and rotation
 - Scale is trickier
 - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
 - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)
2. Design an invariant feature *descriptor*
 - A descriptor captures the information in a region around the detected feature point
 - The simplest descriptor: a square window of pixels
 - What’s this invariant to?
 - Let’s look at some better approaches...

Finding Keypoints – Scale, Location

- How do we choose scale?



Scale Invariant Detection

■ Functions for determining scale

Kernels:

$$f = \text{Kernel} * \text{Image}$$

$$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$$

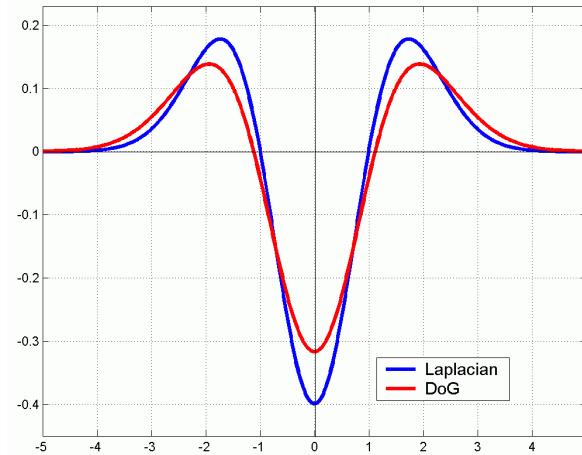
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

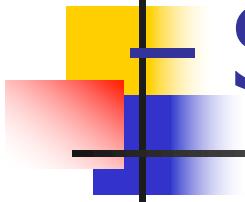
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



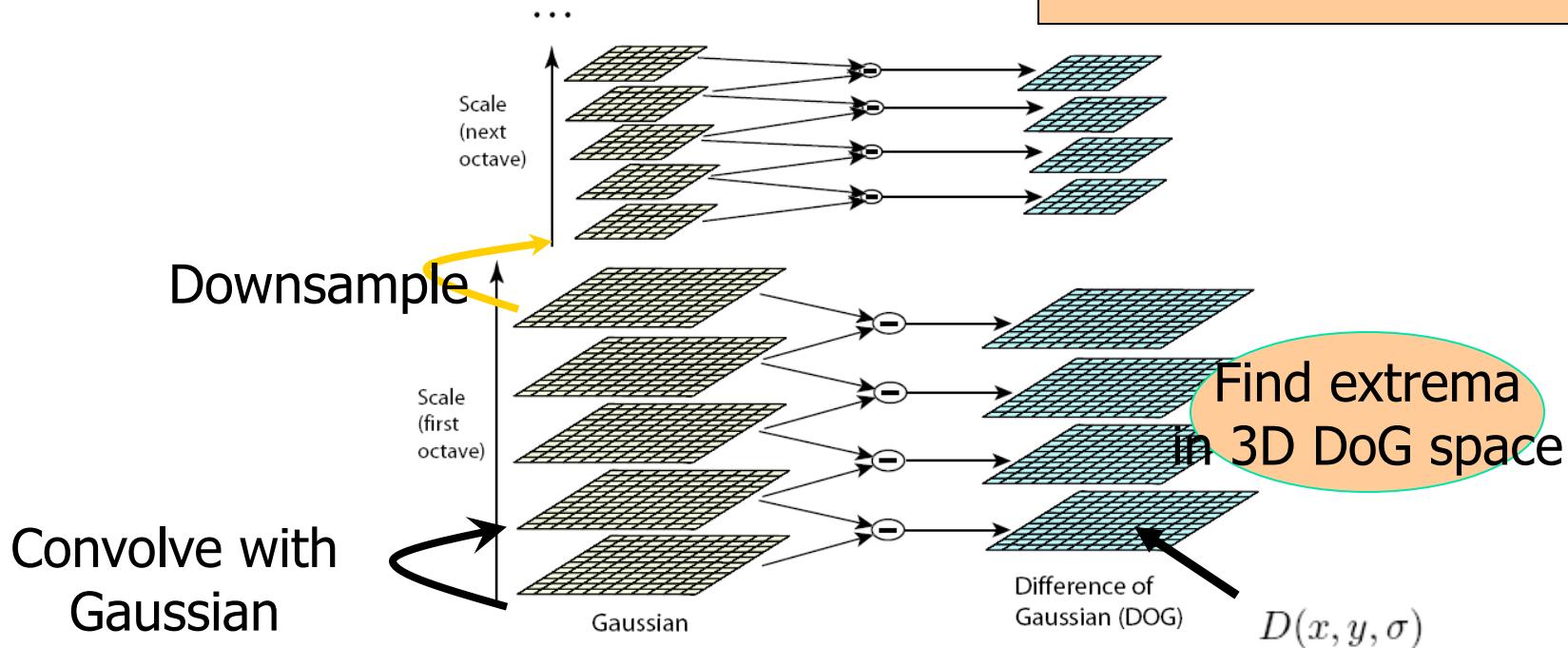
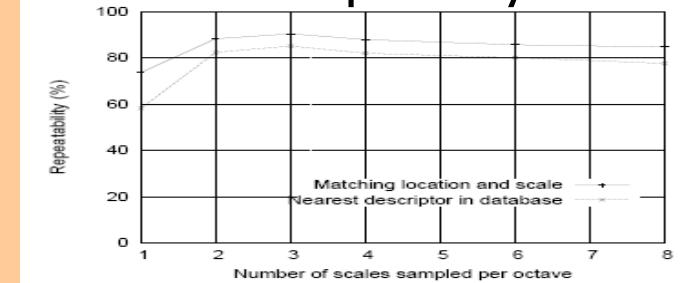
Note: both kernels are invariant to *scale* and *rotation*

Finding Keypoints

Scale, Location



of scales/octave
=> empirically



Relationship between LoG and DoG operator

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

$$\sigma \nabla^2 G = \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

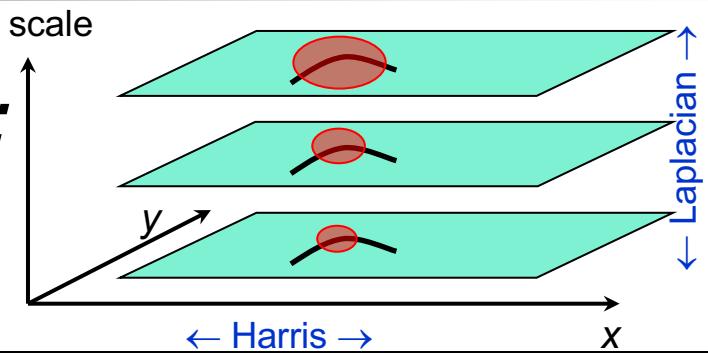
$$G(x, y, k\sigma) - G(x, y, \sigma) = (k - 1)\sigma^2 \nabla^2 G$$

Scale Invariant Detectors

Harris-Laplacian¹

Find local maximum of:

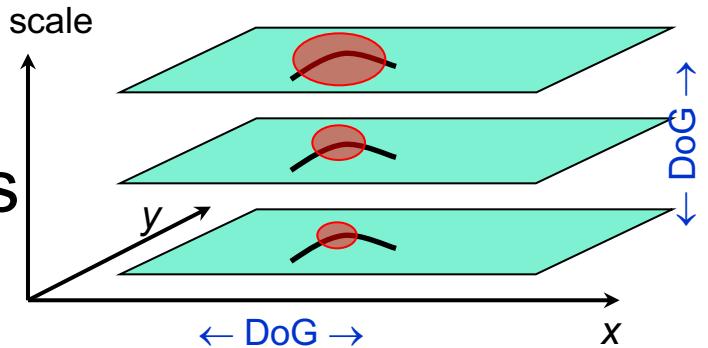
- Harris corner
- Laplacian in scale



SIFT (Lowe)²

Find local maximum of:

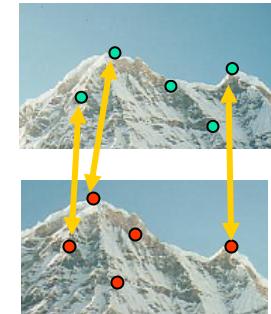
- Difference of Gaussians
in space and scale



¹ K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001

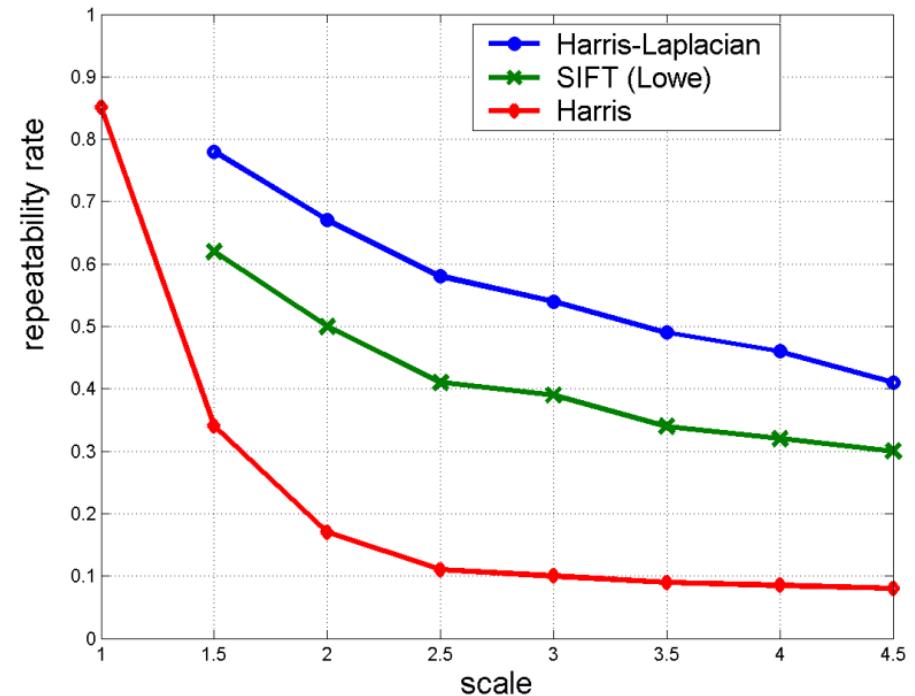
² D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 2004

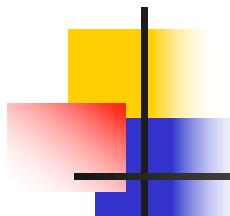
Scale Invariant Detectors



- Experimental evaluation of detectors w.r.t. scale change

Repeatability rate:
$$\frac{\# \text{ correspondences}}{\# \text{ possible correspondences}}$$



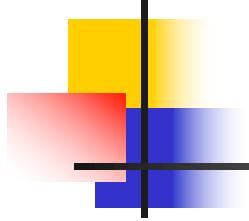


Scale Invariant Detection: Summary

- **Given:** two images of the same scene with a large *scale difference* between them.
- **Goal:** find *the same* interest points *independently* in each image.
- **Solution:** search for *maxima* of suitable functions in *scale* and in *space* (over the image).

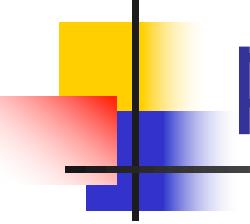
Methods:

1. **Harris-Laplacian** [Mikolajczyk, Schmid]: maximize Laplacian over scale, Harris' measure of corner response over the image.
2. **SIFT** [Lowe]: maximize Difference of Gaussians over scale and space.



Keypoint localization

- There are still a lot of points, some of them are not good enough.
- The locations of keypoints may be not accurate.
- Eliminating edge points.



Eliminating edge points

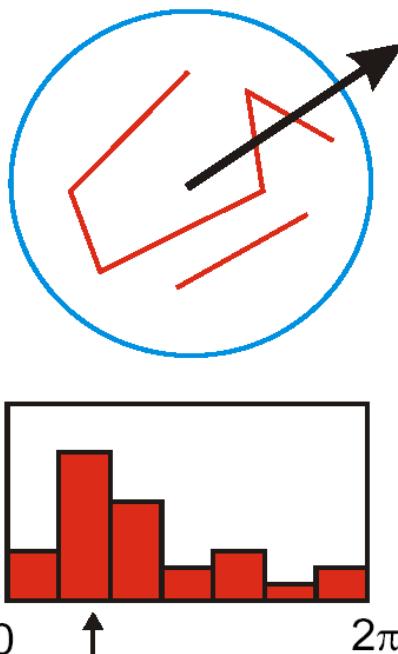
$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

- Edge point: large principal curvature across the edge but a small one in the perpendicular direction.
- The eigenvalues of \mathbf{H} are proportional to the principal curvatures, so two eigenvalues shouldn't differ too much.

$$\frac{\text{Tr}(H)^2}{\text{Det}(H)} < \frac{(r+1)^2}{r} \text{ for high value of } r \text{ (say 10)}$$

Orientation assignment

- Create histogram of local gradient directions at selected scale.
- Assign canonical orientation at peak of smoothed histogram.
- Each key specifies stable 2D coordinates (x , y , scale, orientation).



If 2 major orientations, use both.

Keypoint localization with orientation

233x189



(a)

729
keypoints
after
gradient
threshold



(c)

initial
keypoints

832

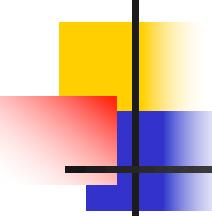


(b)

536
keypoints
after
ratio
threshold



(d)

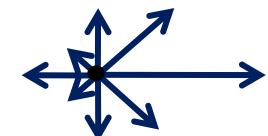
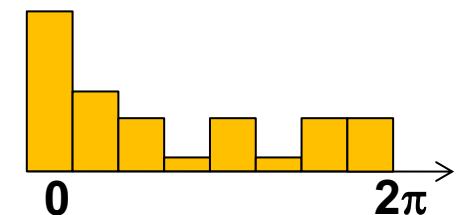
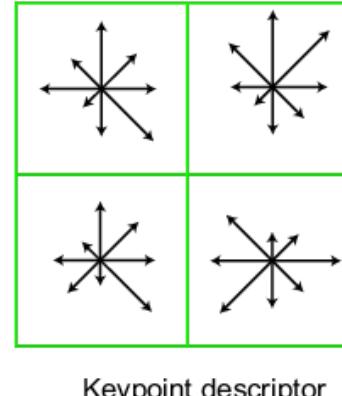
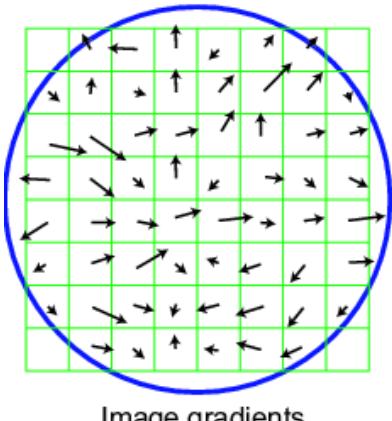


Keypoint Descriptors

- At this point, each keypoint has
 - location
 - scale
 - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
 - highly distinctive
 - invariant as possible to variations such as changes in viewpoint and illumination

Scale Invariant Feature Transform (Lowe'99, ICCV)

- Take 16x16 square window (orientation corrected) around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

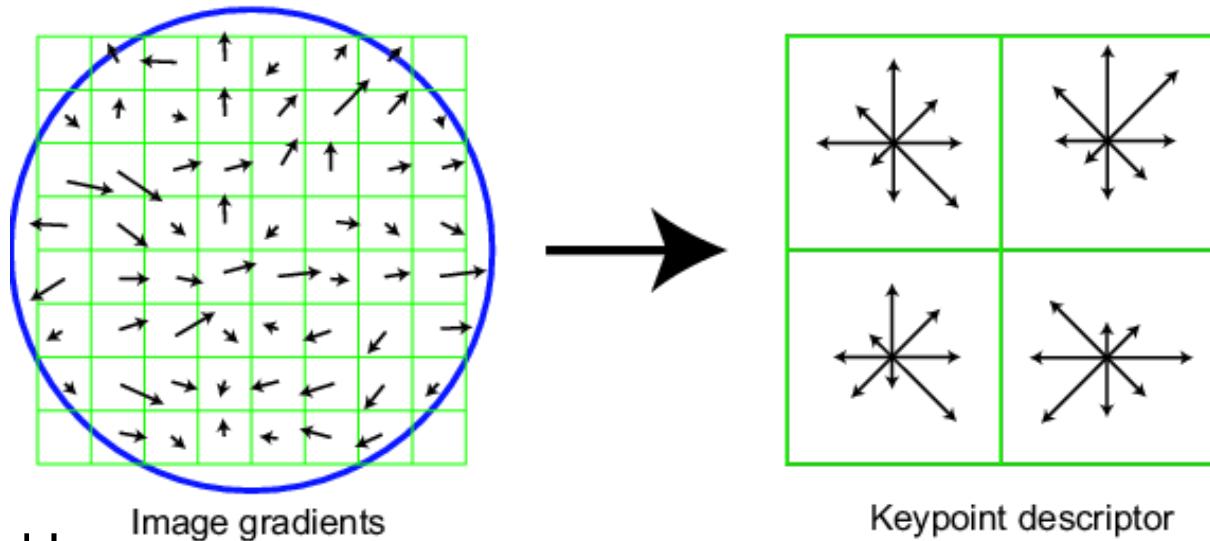


Adapted from slide by David Lowe

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below).
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor.



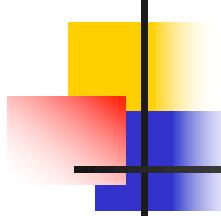
Adapted from slide by David Lowe

Properties of SIFT



Extraordinarily robust matching technique

- Can handle changes in viewpoint
 - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
 - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
 - [http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known implementations of SIFT](http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT)

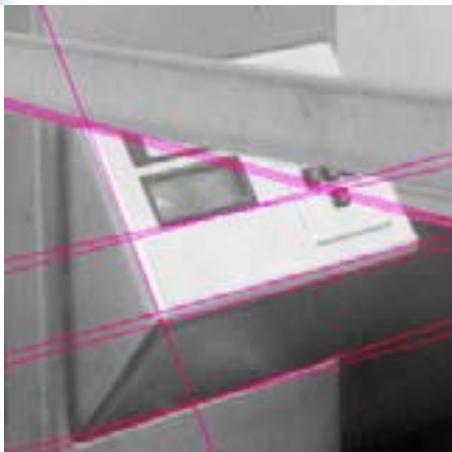


Speeded-Up Robust Features (SURF): Another descriptor

- Speeded-Up Robust Features (SURF)
 - (Bay et al. ECCV, 2006)
 - Box-type convolution filters and use of integral images to speed up the computation.
 - Use of Hessian operator for key point detection → Local maxima of $\det(H)$.
 - Accumulate orientation corrected Haar wavelet responses.

Model Fitting

- Choose a parametric model to represent a set of features



simple model: lines



simple model: circles



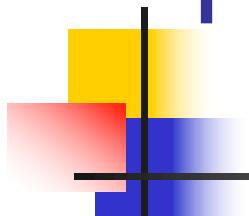
complicated model: car

Model Fitting: Issues

Case study:
Line detection



- **Noise** in the measured feature locations
- **Extraneous data:** clutter (outliers), multiple lines
- **Missing data:** occlusions



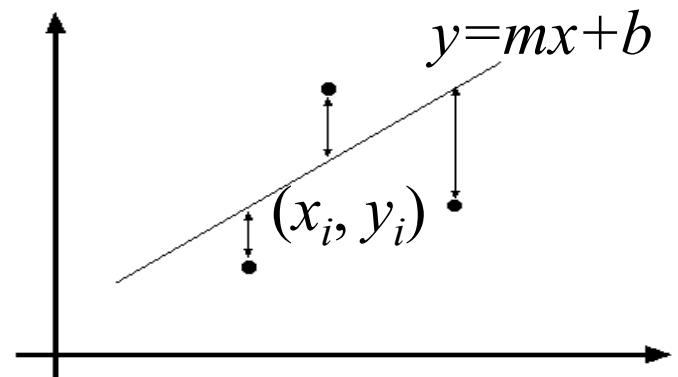
Fitting: Issues

- If we know which points belong to the line, how do we find the “optimal” line parameters?
 - Least squares
- What if there are outliers?
 - Robust fitting, RANSAC
- What if there are many lines?
 - Voting methods: RANSAC, Hough transform
- What if we’re not even sure it’s a line?
 - Model selection

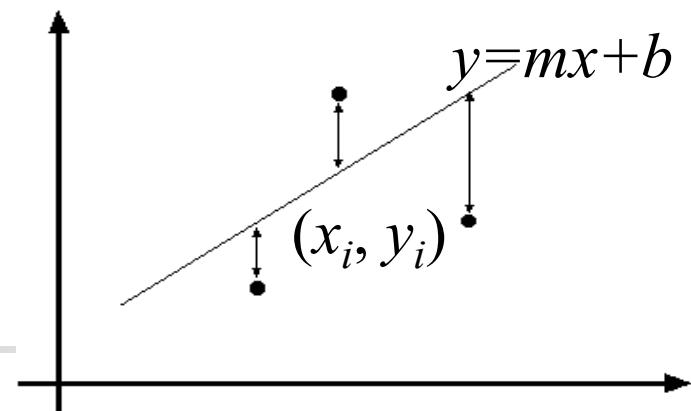
Least squares line fitting

$$E = \sum_{i=1}^n (y_i - mx_i - b)^2$$

- Data: $(x_1, y_1), \dots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize



Least squares line fitting



- Data: $(x_1, y_1), \dots, (x_n, y_n)$
- Line equation: $y_i = mx_i + b$
- Find (m, b) to minimize

$$E = \sum_{i=1}^n (y_i - mx_i - b)^2$$

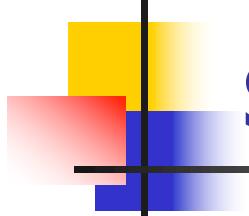
$$E = \sum_{i=1}^n \left(y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right)^2 = \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right\|^2 = \|Y - XB\|^2$$

$$= (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0$$

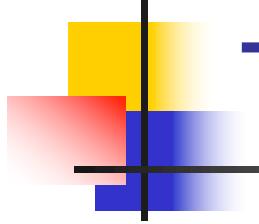
$$X^T XB = X^T Y$$

Normal equations: least squares solution to
 $XB = Y$



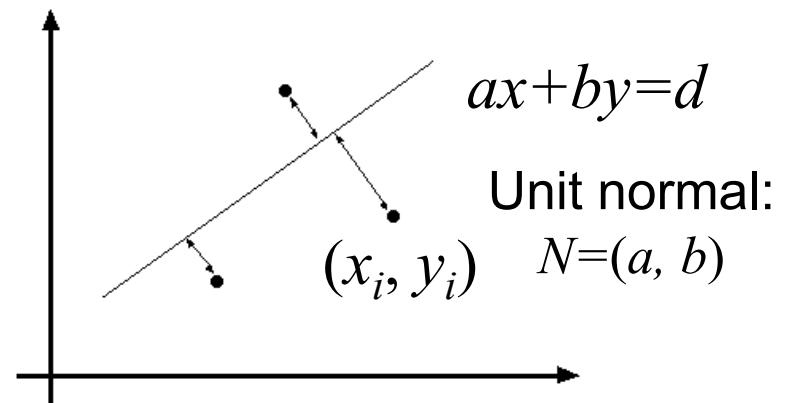
Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines



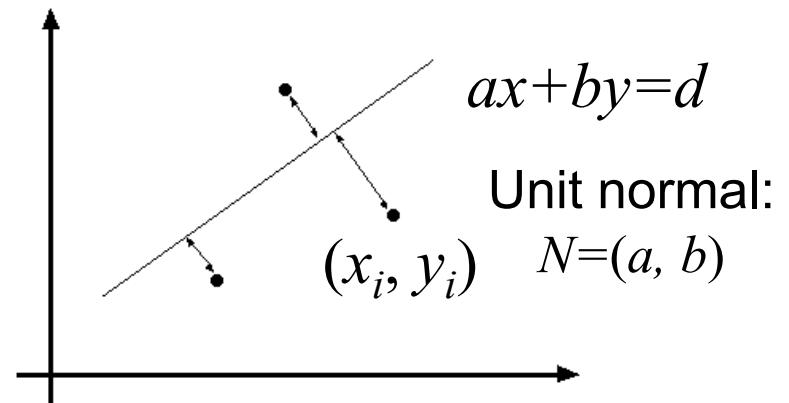
Total least squares

- Distance between point (x_i, y_i) and line $ax+by=d$
 $(a^2+b^2=1): |ax_i + by_i - d|$



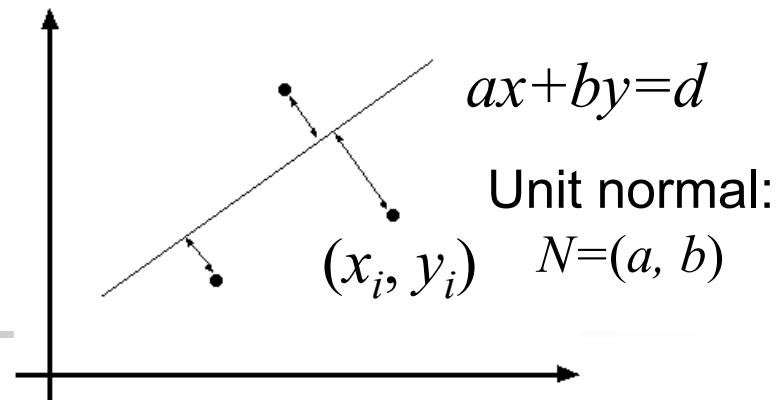
Total least squares

- Distance between point (x_i, y_i) and line $ax+by=d$
 $(a^2+b^2=1): |ax_i + by_i - d|$



$$E = \sum_{i=1}^n (ax_i + by_i - d)^2$$

Total least squares



$$E = \sum_{i=1}^n (ax_i + by_i - d)^2$$

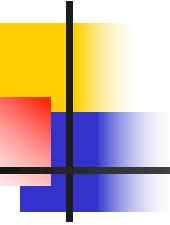
$$\frac{\partial E}{\partial d} = \sum_{i=1}^n -2(ax_i + by_i - d) = 0$$

$$d = \frac{a}{n} \sum_{i=1}^n x_i + \frac{b}{n} \sum_{i=1}^n y_i = a\bar{x} + b\bar{y}$$

$$E = \sum_{i=1}^n (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)$$

Solution to $(U^T U)N = 0$, subject to $\|N\|^2 = 1$: eigenvector of $U^T U$ associated with the smallest eigenvalue (least squares solution to *homogeneous linear system* $UN = 0$)

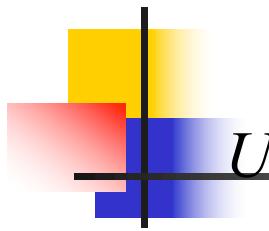
Total least squares

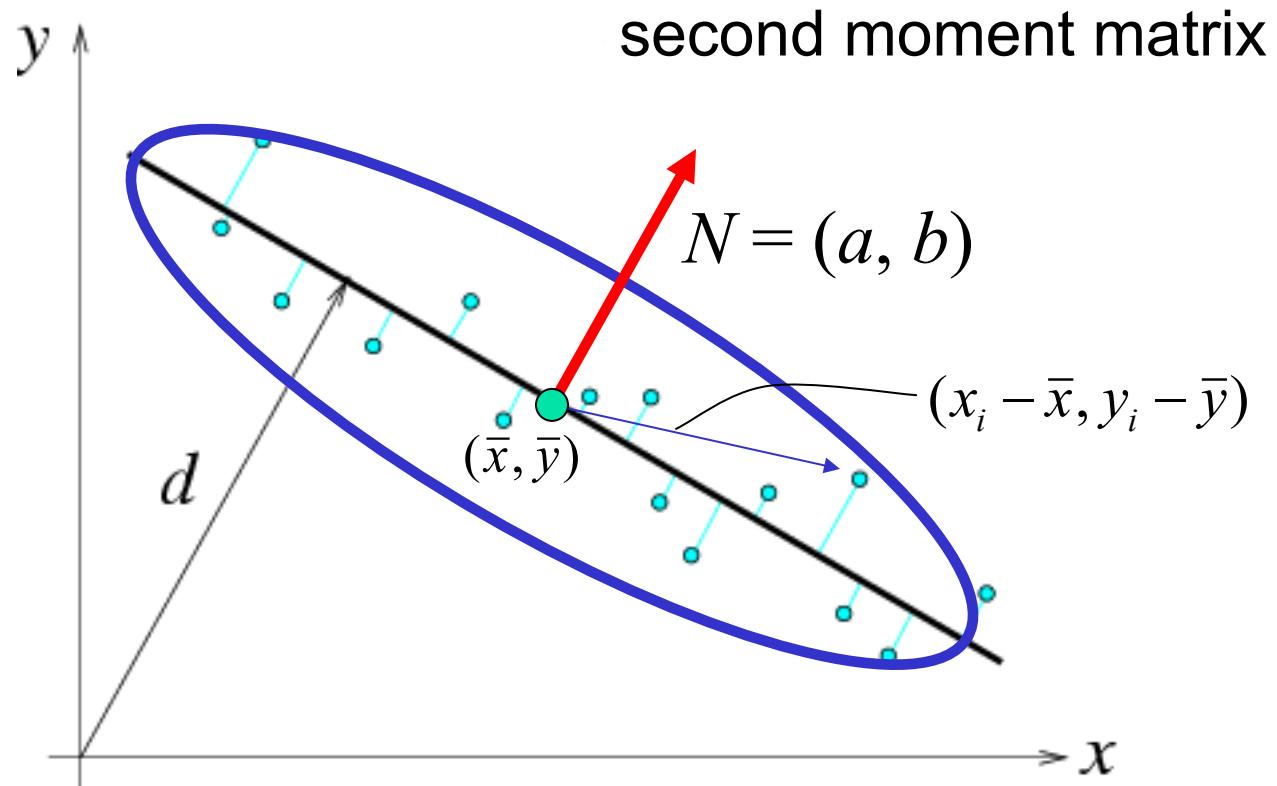


$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \quad U^T U = \begin{bmatrix} \sum_{i=1}^n (x_i - \bar{x})^2 & \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^n (y_i - \bar{y})^2 \end{bmatrix}$$

second moment matrix

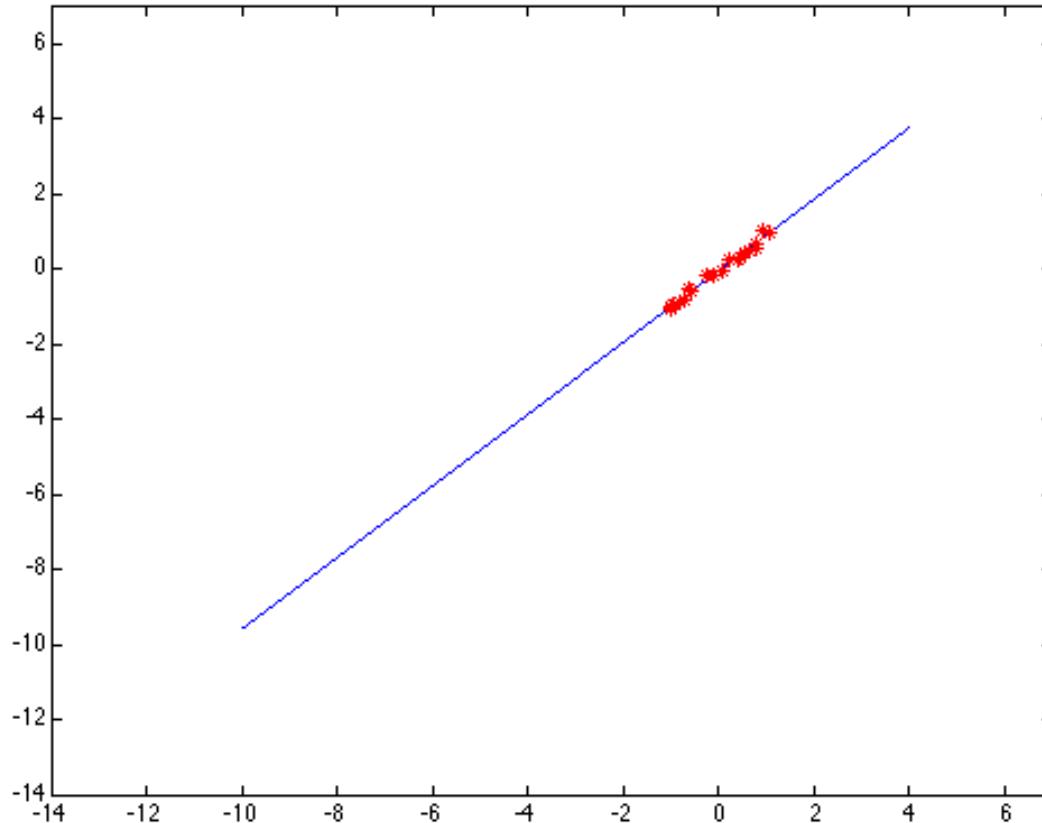
Total least squares


$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \quad U^T U = \begin{bmatrix} \sum_{i=1}^n (x_i - \bar{x})^2 & \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^n (y_i - \bar{y})^2 \end{bmatrix}$$



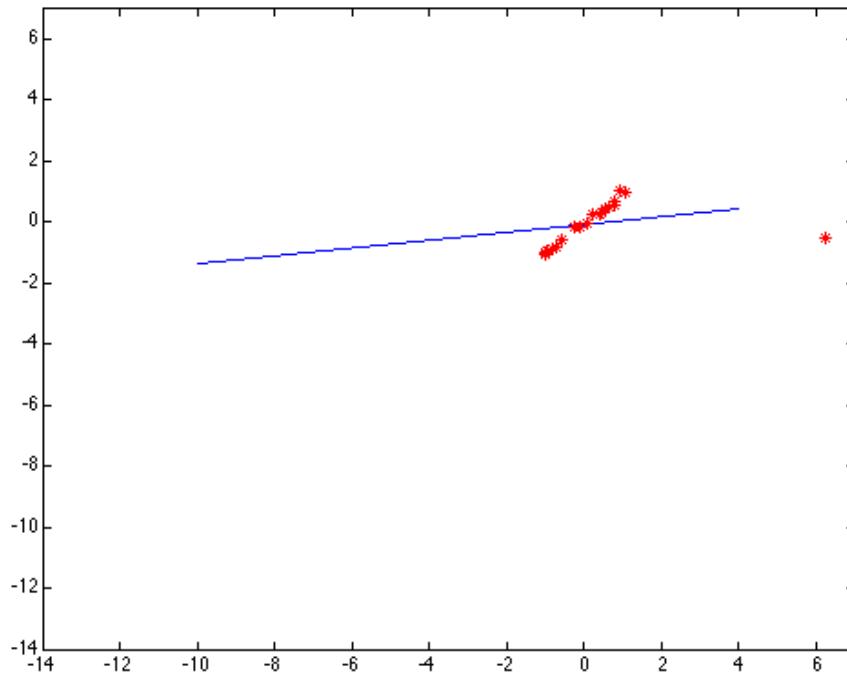
Least squares: Robustness to noise

- Least squares fit to the red points:



Least squares: Robustness to noise

- Least squares fit with an outlier:



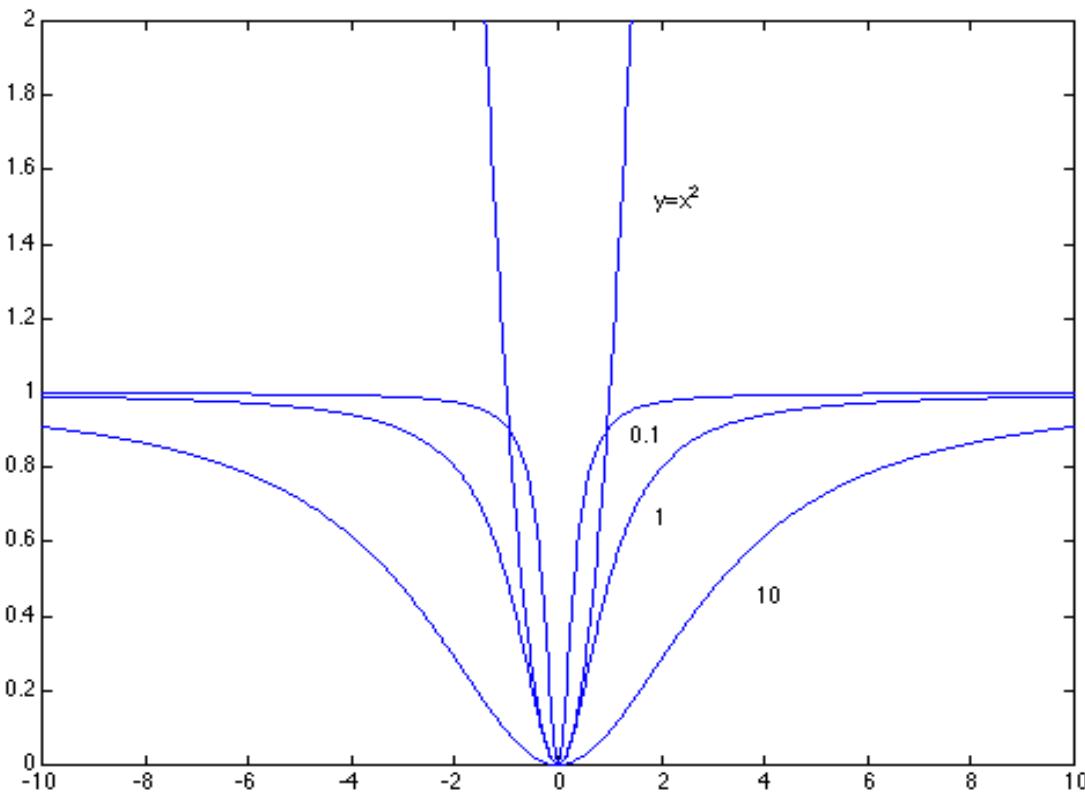
Problem: squared error heavily penalizes outliers

Robust estimators

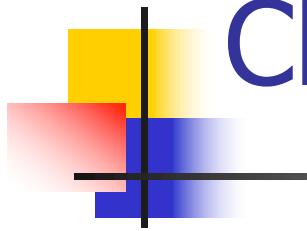
$$\rho(u; \sigma) = \frac{u^2}{\sigma^2 + u^2}$$

- General approach: minimize $\sum_i \rho(r_i(x_i, \theta); \sigma)$

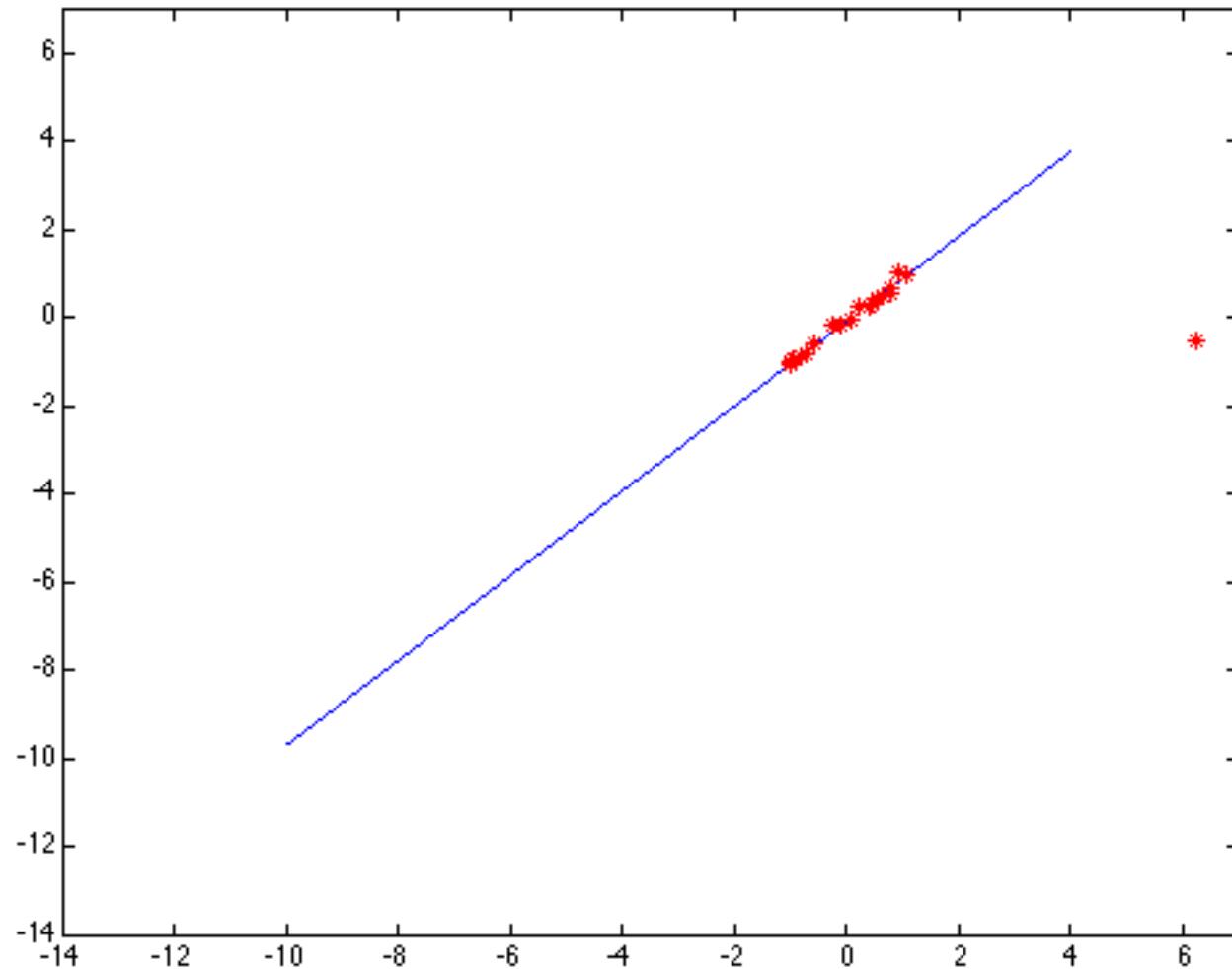
$r_i(x_i, \theta)$ – residual of i th point w.r.t. model parameters θ
 ρ – robust function with scale parameter σ



The robust function ρ behaves like squared distance for small values of the residual u but saturates for larger values of u

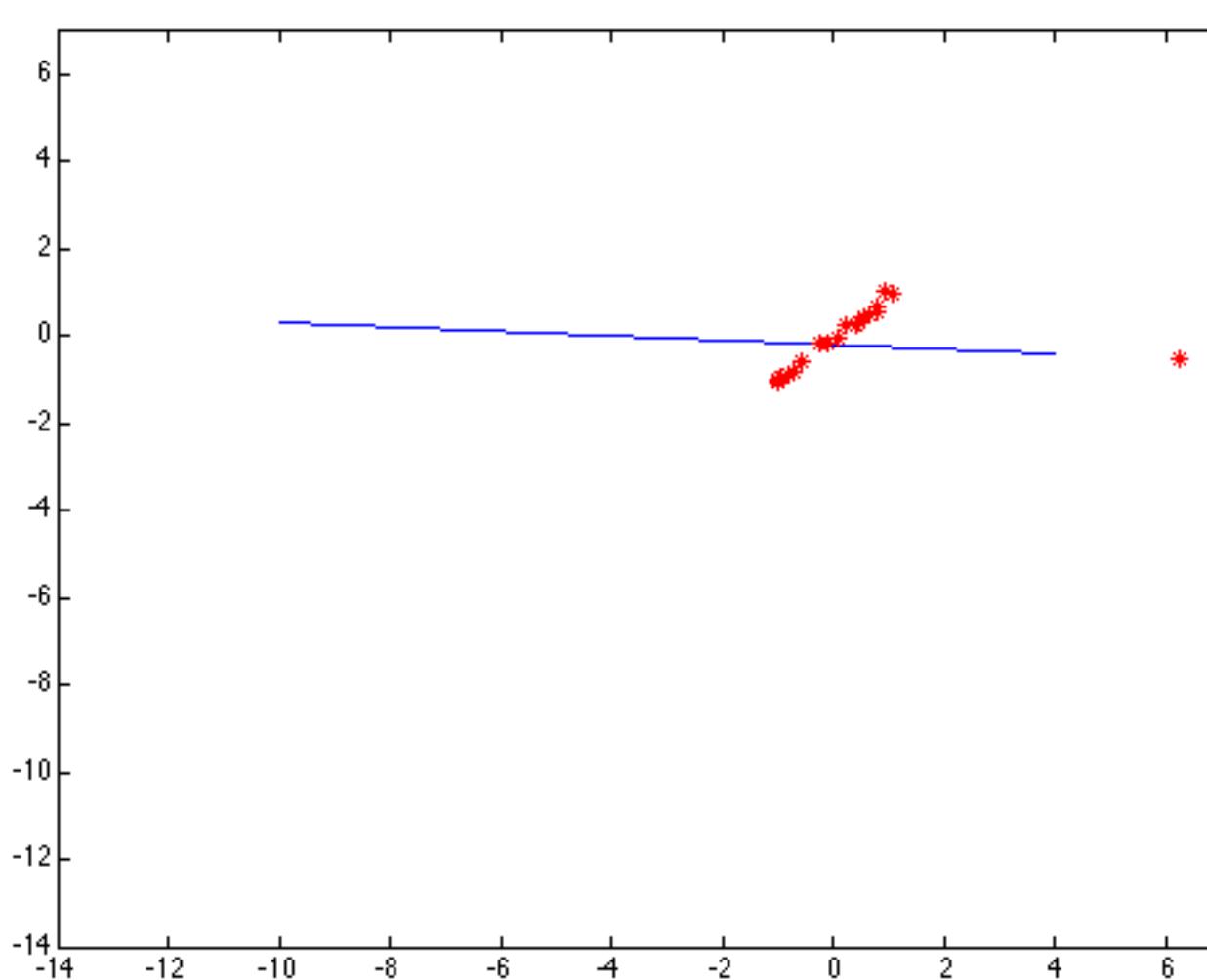


Choosing the scale: Just right



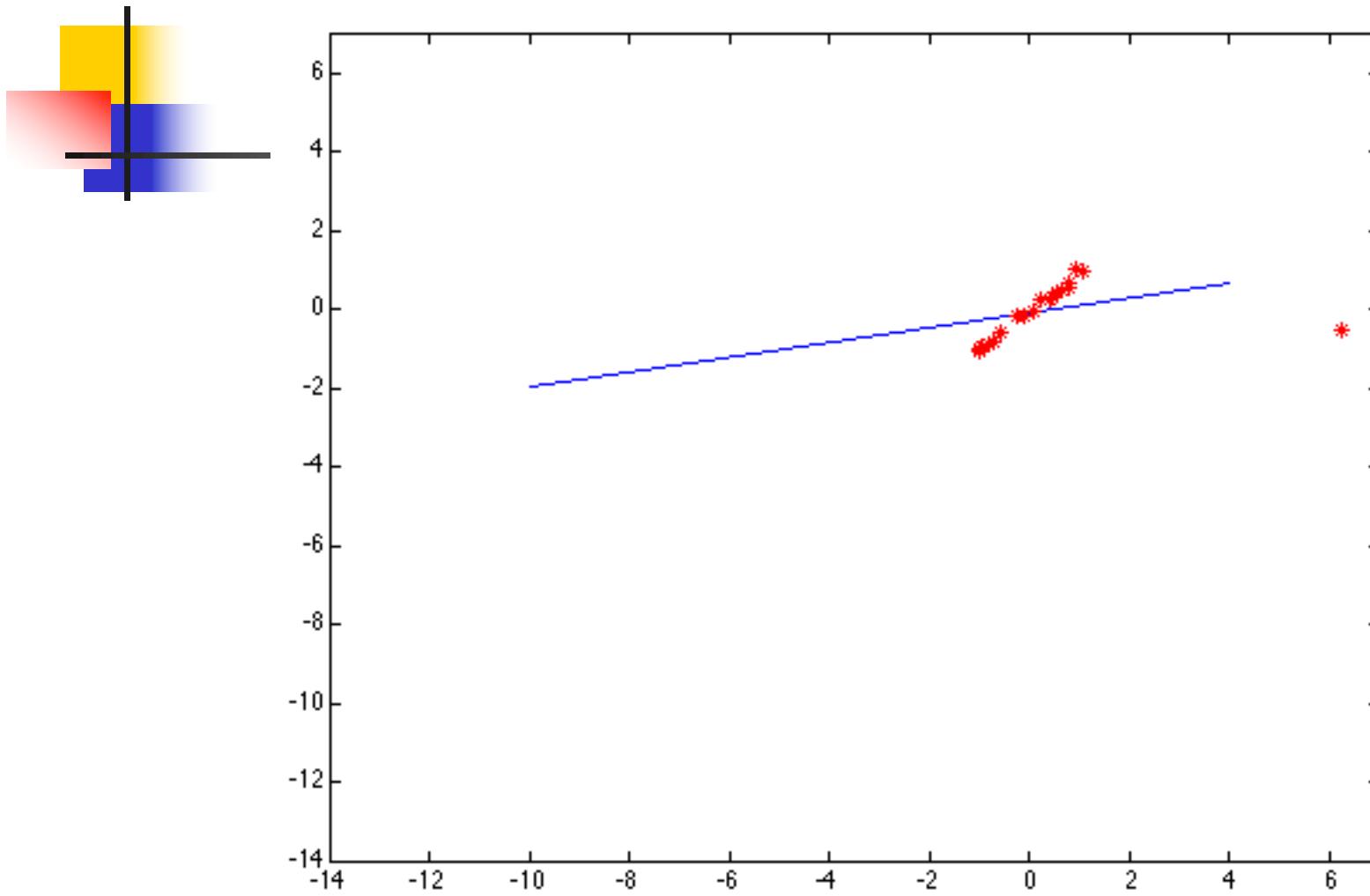
The effect of the outlier is minimized

Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Choosing the scale: Too large

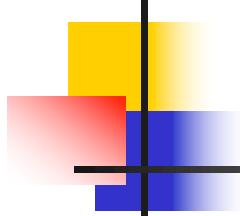


Behaves much the same as least squares

RANSAC

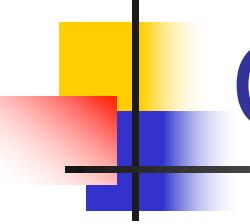
- Robust fitting can deal with a few outliers – what if we have very many?
- Random sample consensus (RANSAC):
Very general framework for model fitting in the presence of outliers
- Outline
 - Choose a small subset of points uniformly at random
 - Fit a model to that subset
 - Find all remaining points that are “close” to the model and reject the rest as outliers
 - Do this many times and choose the best model

M. A. Fischler, R. C. Bolles. [Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography](#). Comm. of the ACM, Vol 24, pp 381-395, 1981.



RANSAC for line fitting

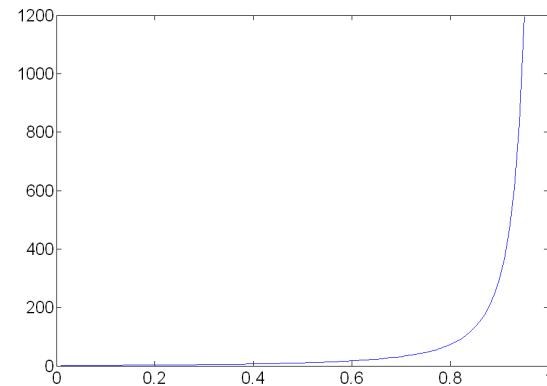
- Repeat N times:
 - Draw s points uniformly at random
 - Fit line to these s points
 - Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than t)
 - If there are d or more inliers, accept the line and refit using all inliers



Choosing the parameters

- Initial number of points s
 - Typically minimum number needed to fit the model
- Distance threshold t
 - Choose t so probability for inlier is p (e.g. 0.95)
 - Zero-mean Gaussian noise with std. dev. σ : $t^2=3.84\sigma^2$
- Number of samples N
 - Choose N so that, with probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)

Choosing the parameters

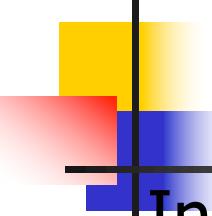


$$\left(1 - \left(1 - e\right)^s\right)^N = 1 - p$$

s	proportion of outliers e							
	5%	10%	20%	25%	30%	40%	50%	
2	2	3	5	6	7	11	17	
3	3	4	7	9	11	19	35	
4	3	5	9	13	17	34	72	
5	4	6	12	17	26	57	146	
6	4	7	16	24	37	97	293	
7	4	8	20	33	54	163	588	
8	5	9	26	44	78	272	1177	

$$N = \log(1 - p) / \log\left(1 - \left(1 - e\right)^s\right)$$

Probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)

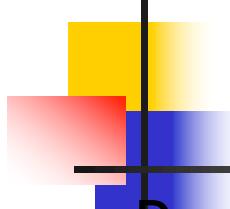


Choosing the parameters

- Initial number of points s
 - Typically minimum number needed to fit the model
- Distance threshold t
 - Choose t so probability for inlier is p (e.g. 0.95)
 - Zero-mean Gaussian noise with std. dev. σ : $t^2=3.84\sigma^2$
- Number of samples N
 - Choose N so that, with probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)
- Consensus set size d
 - Should match expected inlier ratio

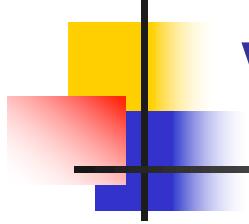
Adaptively determining the number of samples

- Inlier ratio e is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield $e=0.2$
- Adaptive procedure:
 - $N=\infty$, $\text{sample_count} = 0$
 - While $N > \text{sample_count}$
 - Choose a sample and count the number of inliers
 - Set $e = 1 - (\text{number of inliers})/(\text{total number of points})$
 - Recompute N from e :
$$N = \log(1-p)/\log(1-(1-e)^s)$$
 - Increment the sample_count by 1



RANSAC pros and cons

- Pros
 - Simple and general
 - Applicable to many different problems
 - Often works well in practice
- Cons
 - Lots of parameters to tune
 - Can't always get a good initialization of the model based on the minimum number of samples
 - Sometimes too many iterations are required
 - Can fail for extremely low inlier ratios
 - We can often do better than brute-force sampling



Voting schemes

- Let each feature vote for all the models that are compatible with it
- Hopefully the noise features will not vote consistently for any single model
- Missing data doesn't matter as long as there are enough features remaining to agree on a good model

Hough transform

- An early type of voting scheme
- General outline:
 - Discretize parameter space into bins
 - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point.
 - Find bins that have the most votes

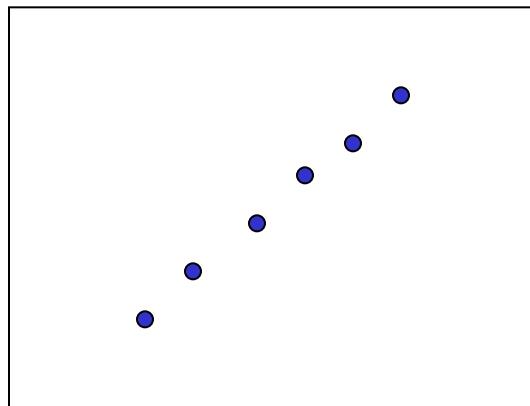
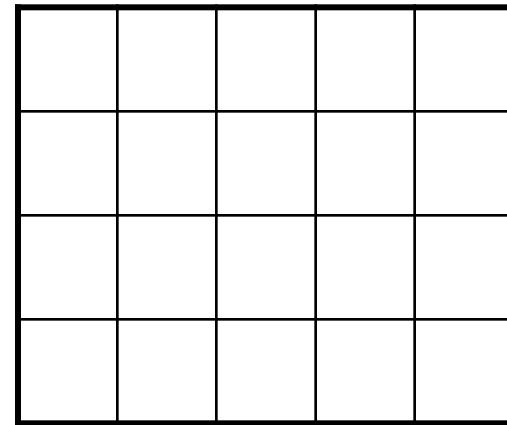


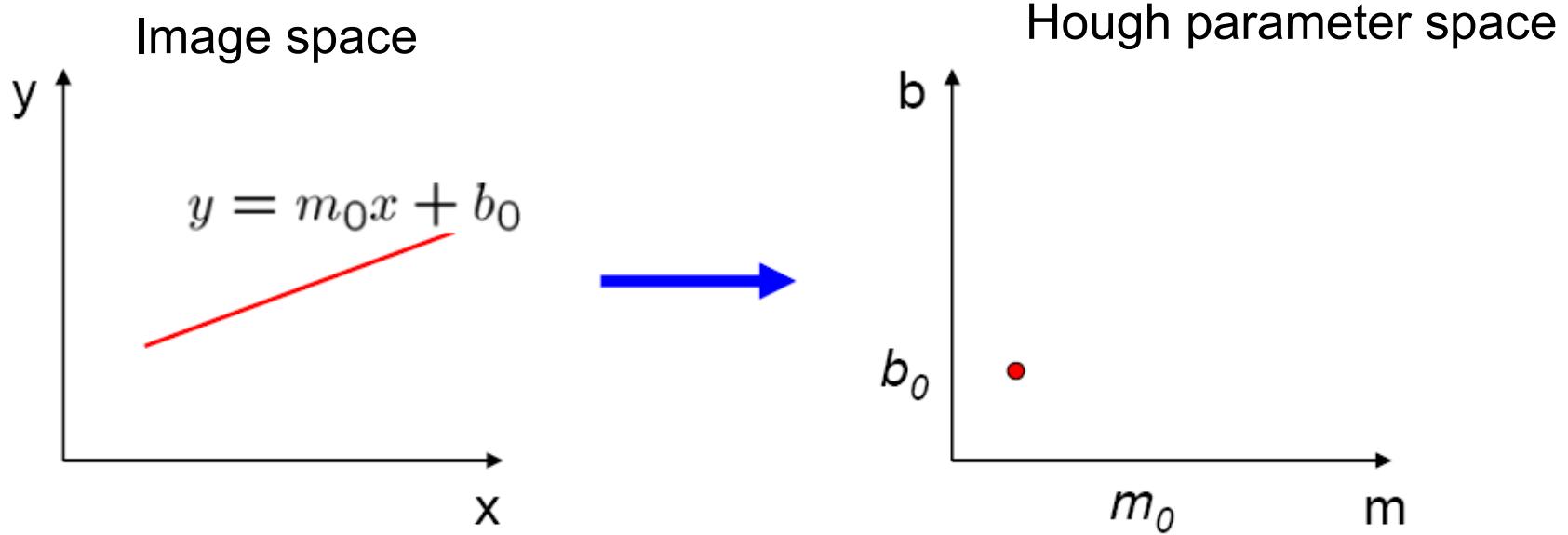
Image space



Hough parameter space

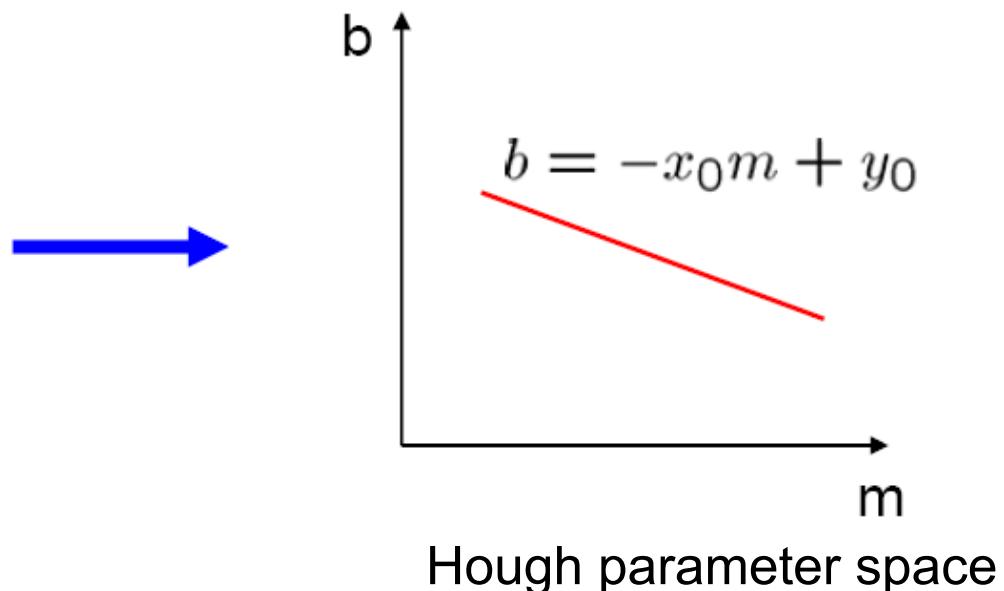
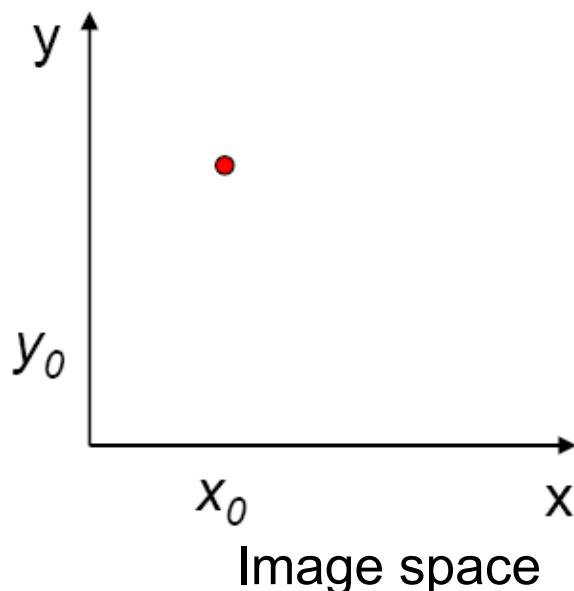
Parameter space representation

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



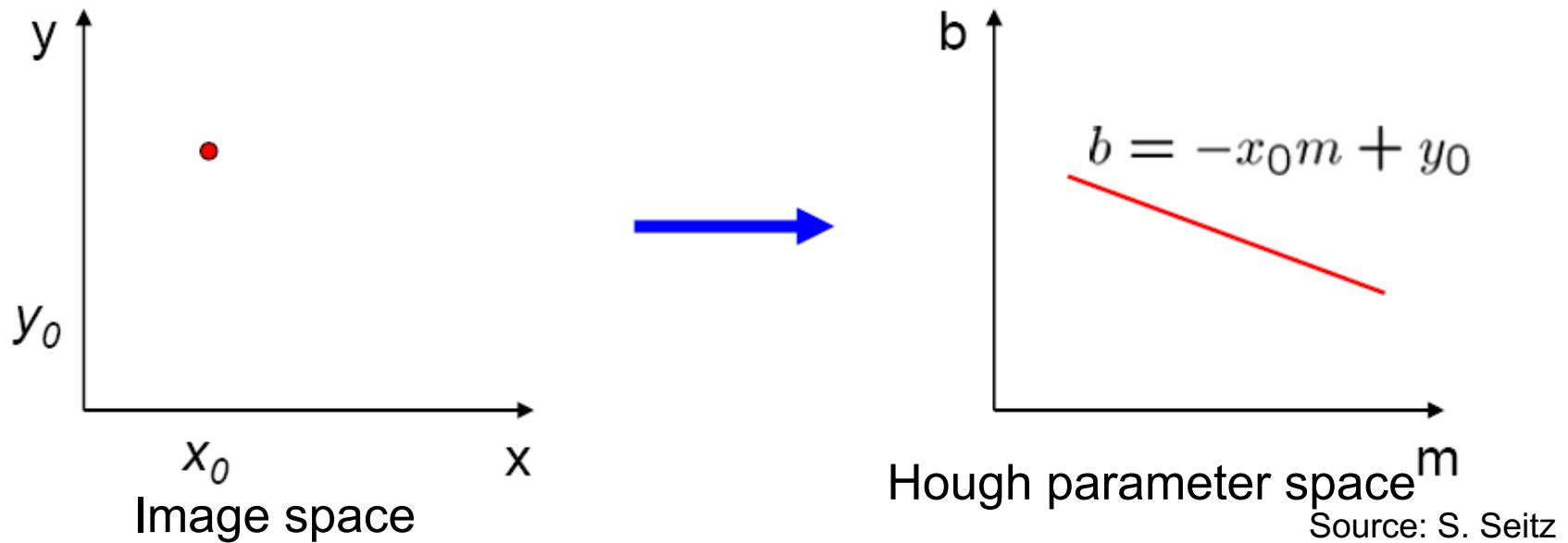
Parameter space representation

- What does a point (x_0, y_0) in the image space map to in the Hough space?



Parameter space representation

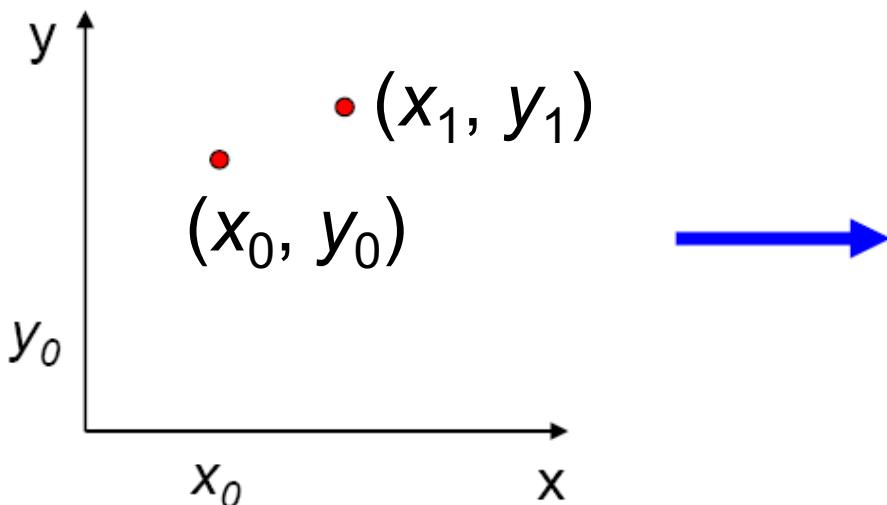
- What does a point (x_0, y_0) in the image space map to in the Hough space?
 - Answer: the solutions of $b = -x_0 m + y_0$
 - This is a line in Hough space



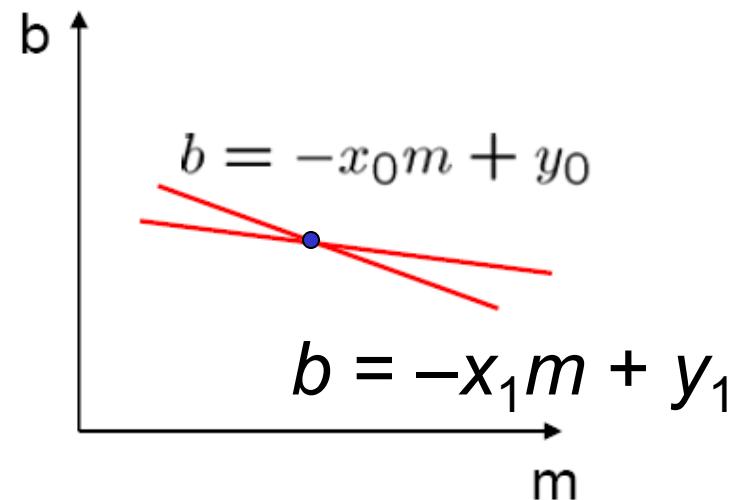
Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1) ?

Image space

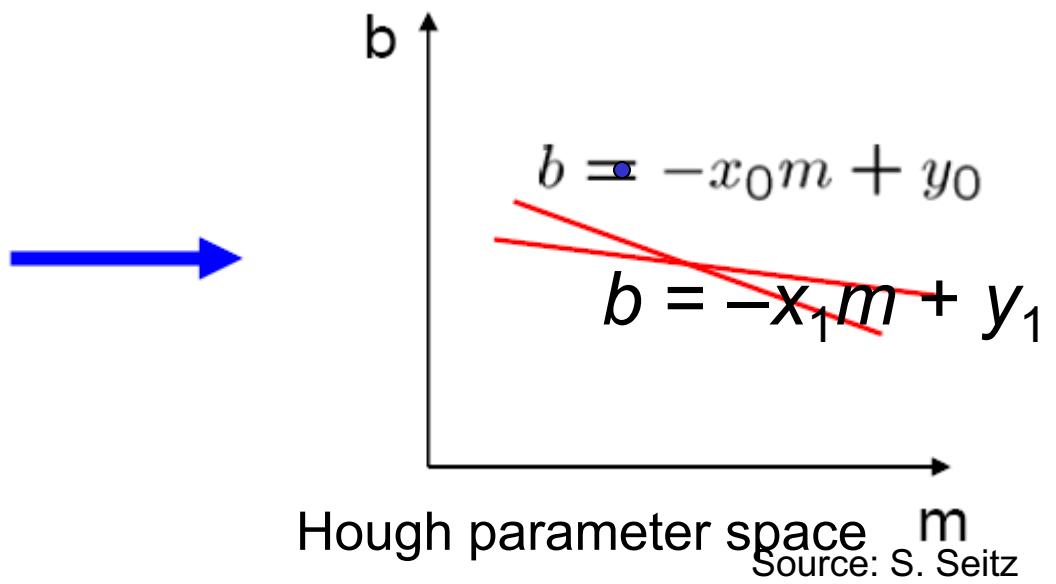
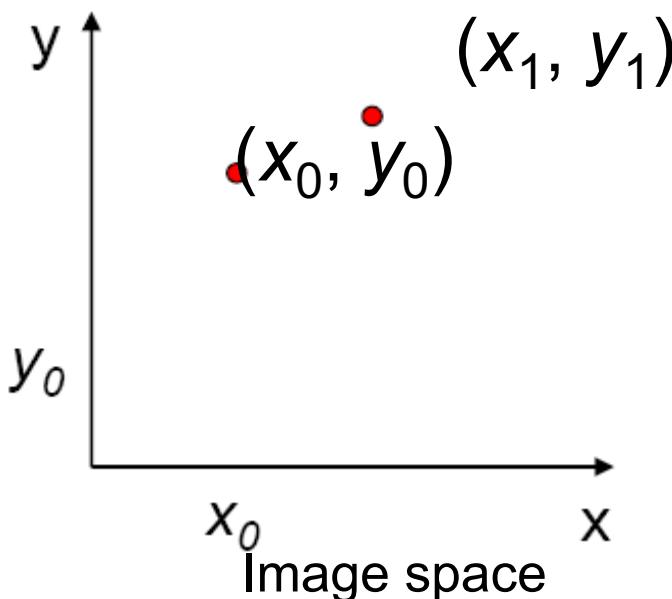


Hough parameter space

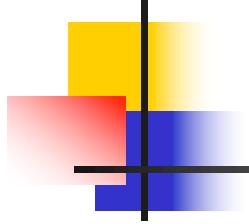


Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1) ?
 - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$



Hough parameter space Source: S. Seitz



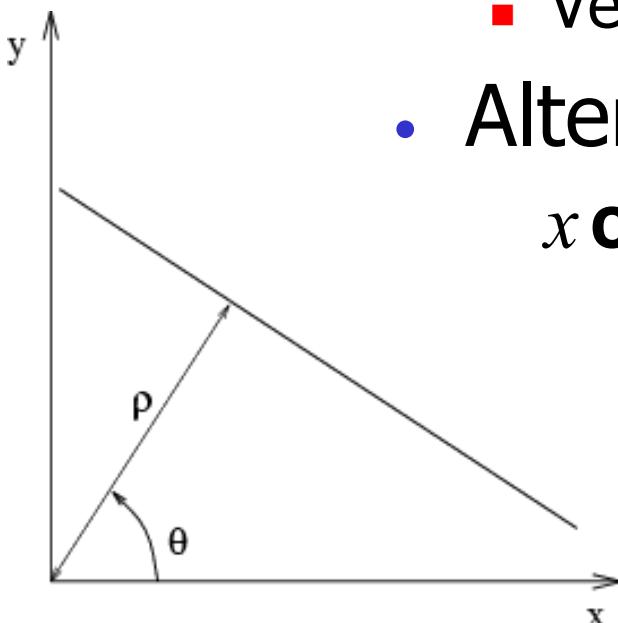
Parameter space representation

- Problems with the (m,b) space:
 - Unbounded parameter domain
 - Vertical lines require infinite m

Parameter space representation

- Problems with the (m,b) space:
 - Unbounded parameter domain
 - Vertical lines require infinite m
- Alternative: polar representation

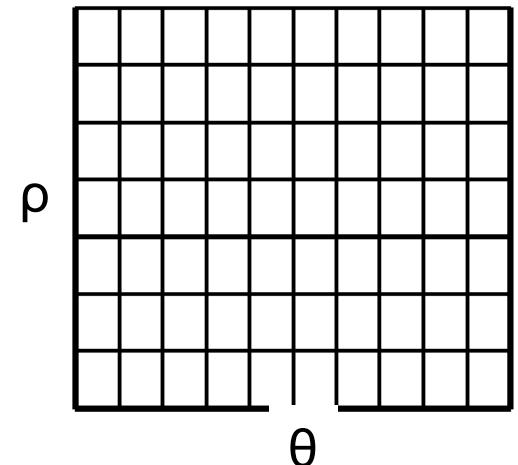
$$x \cos \theta + y \sin \theta = \rho$$



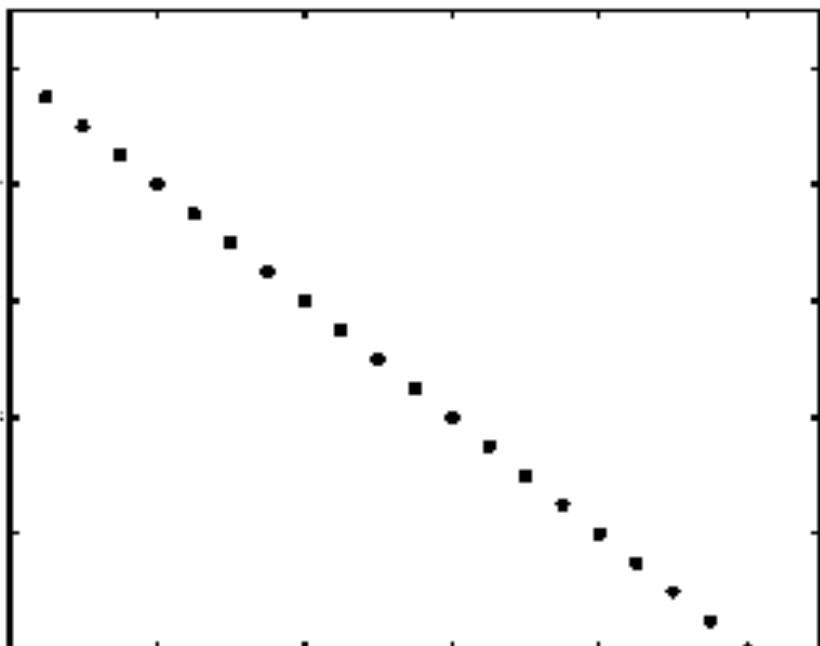
Each point will add a sinusoid in the (θ, ρ) parameter space

Algorithm outline

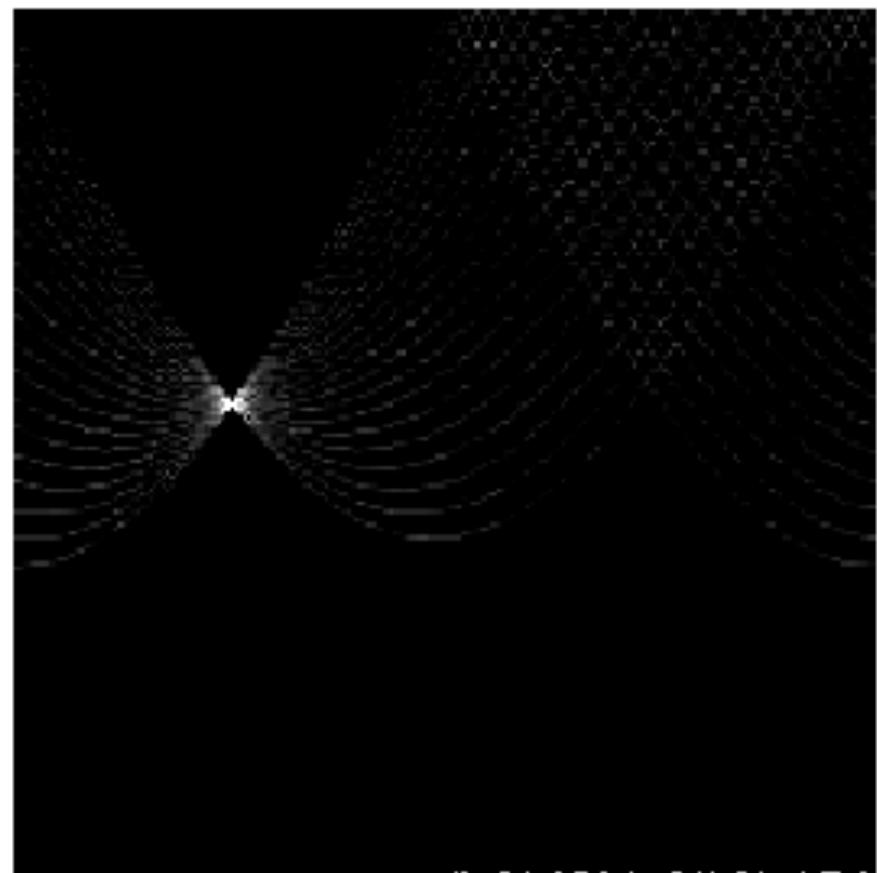
- Initialize accumulator H to all zeros
- For each edge point (x, y) in the image
 - For $\theta = 0$ to 180
 - $\rho = x \cos \theta + y \sin \theta$
 - $H(\theta, \rho) = H(\theta, \rho) + 1$
 - end
 - end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
 - The detected line in the image is given by
$$\rho = x \cos \theta + y \sin \theta$$



Basic illustration



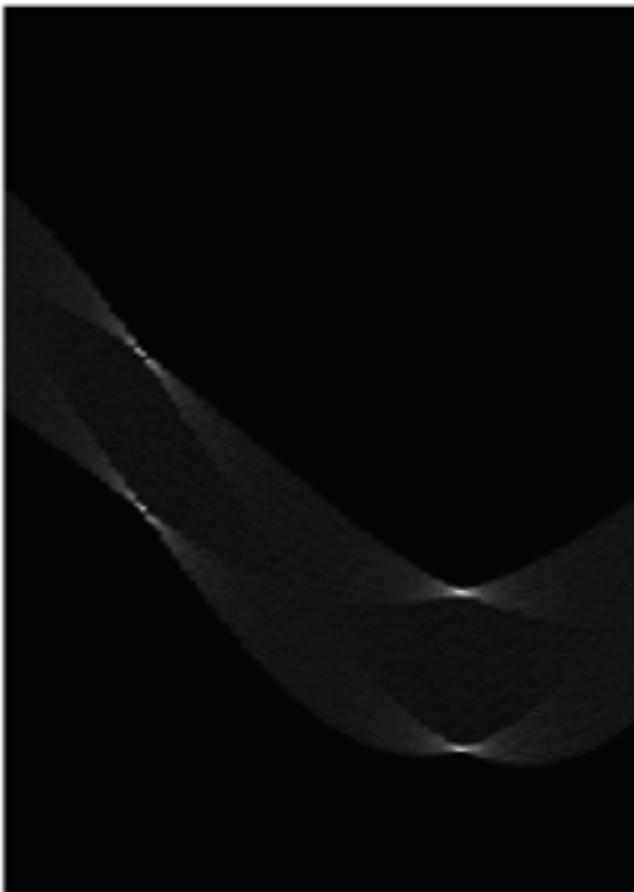
features



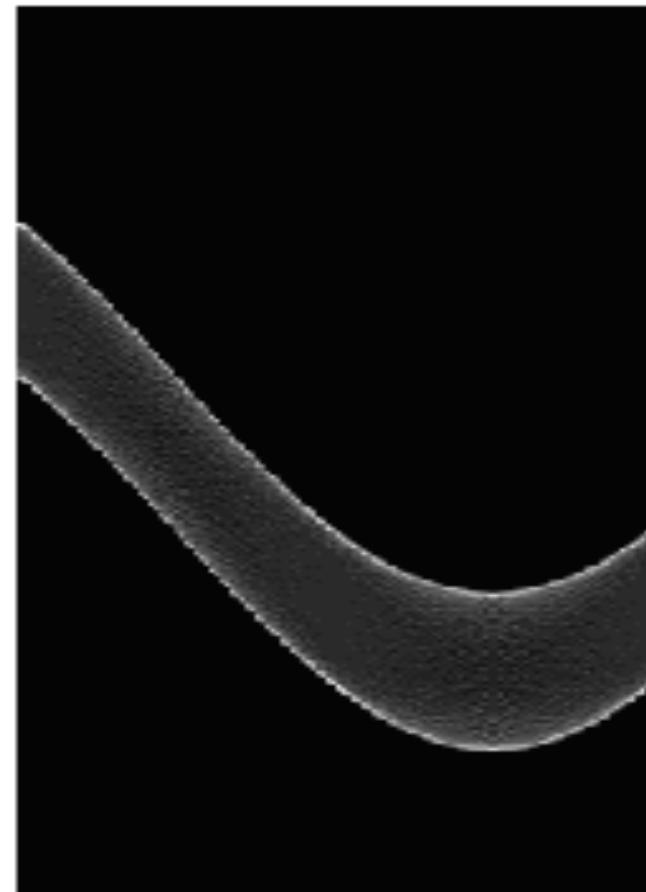
votes

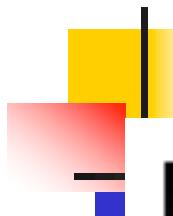
Other shapes

Square

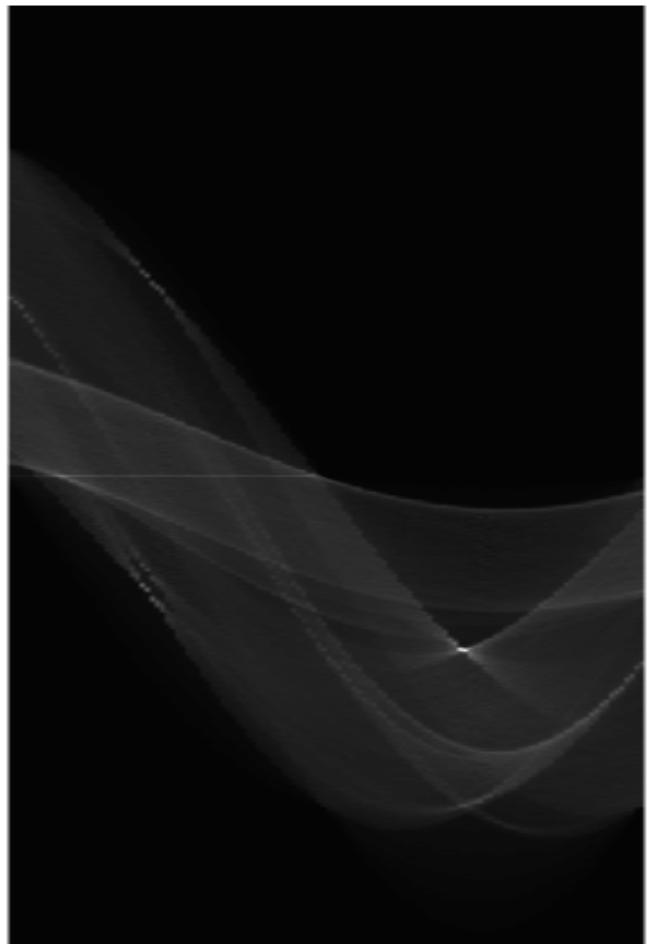
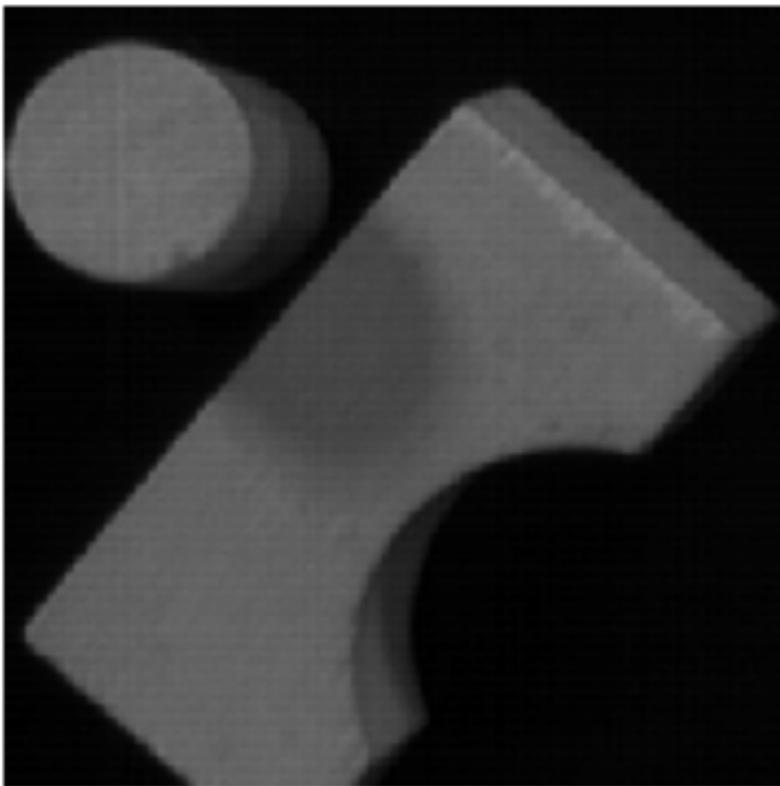


Circle

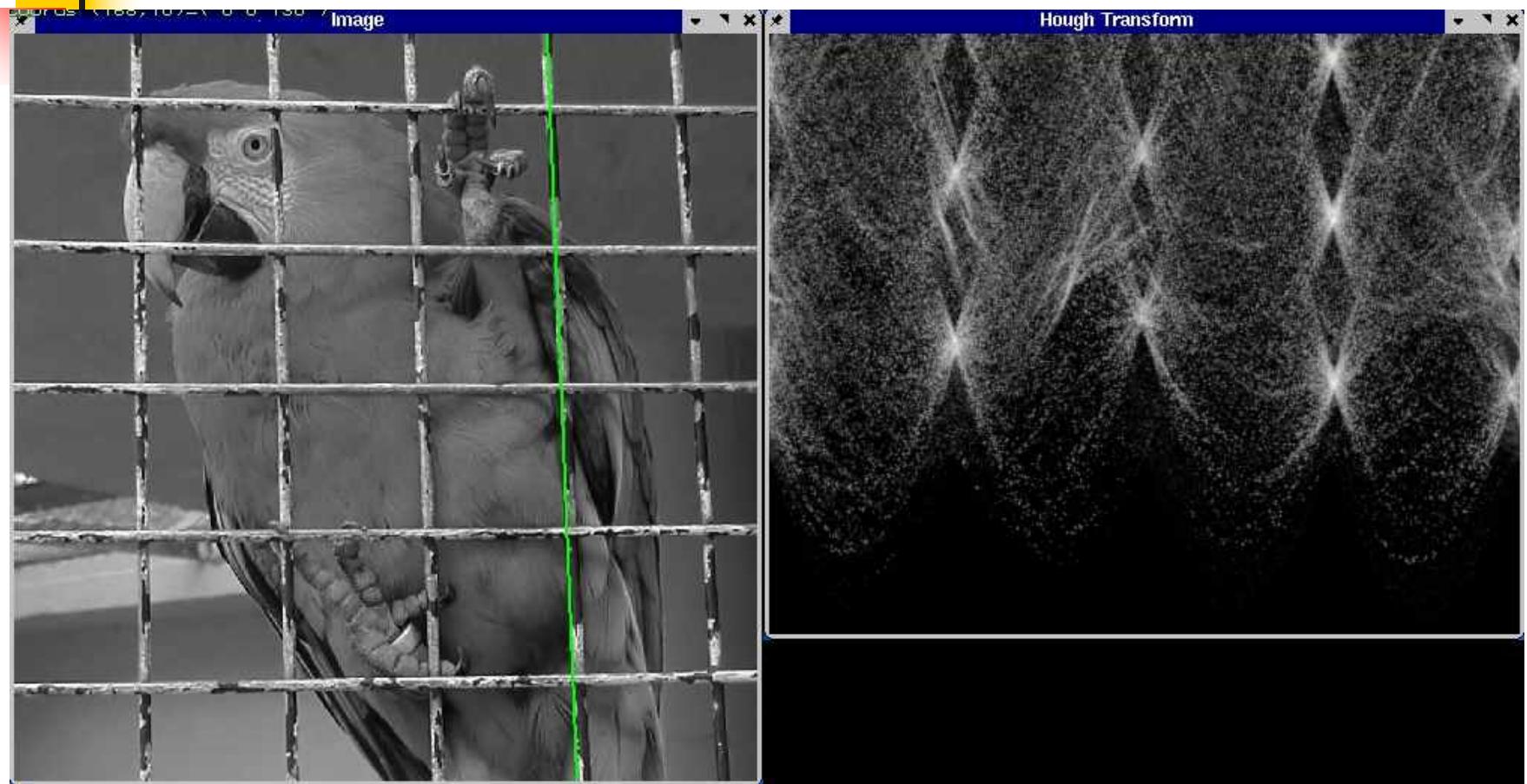


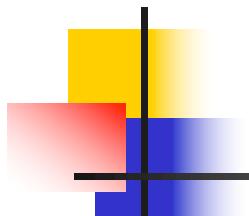


Several lines

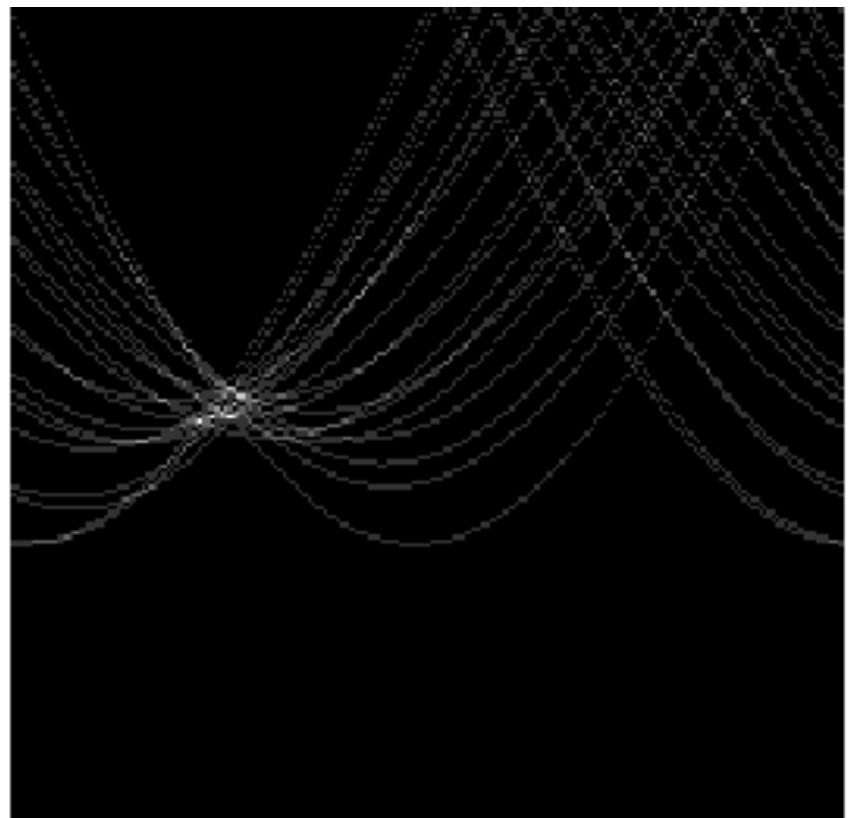
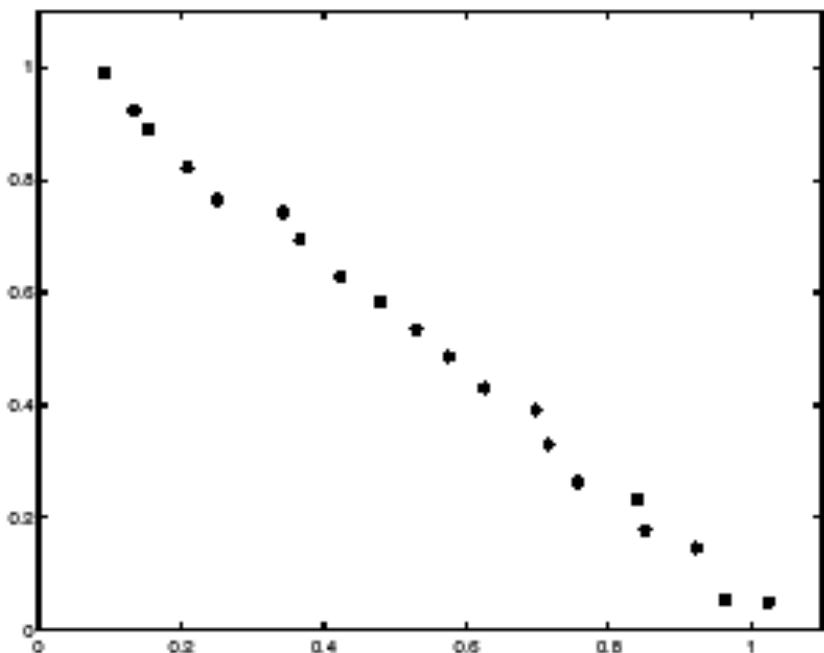


A more complicated image



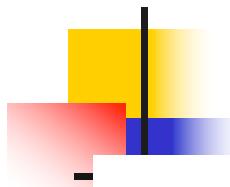


Effect of noise

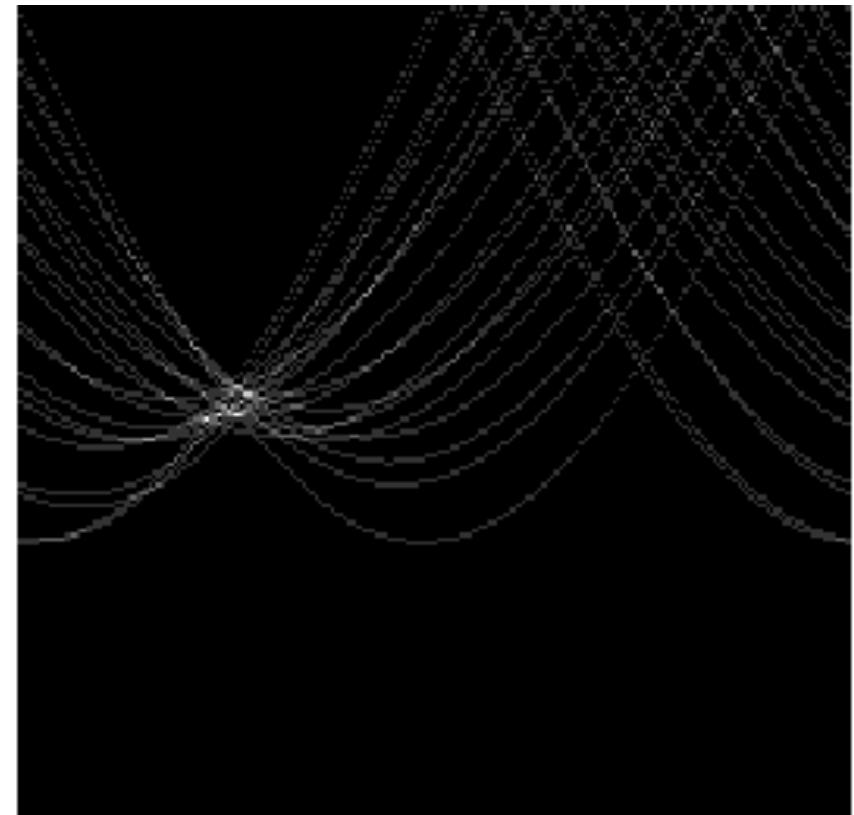
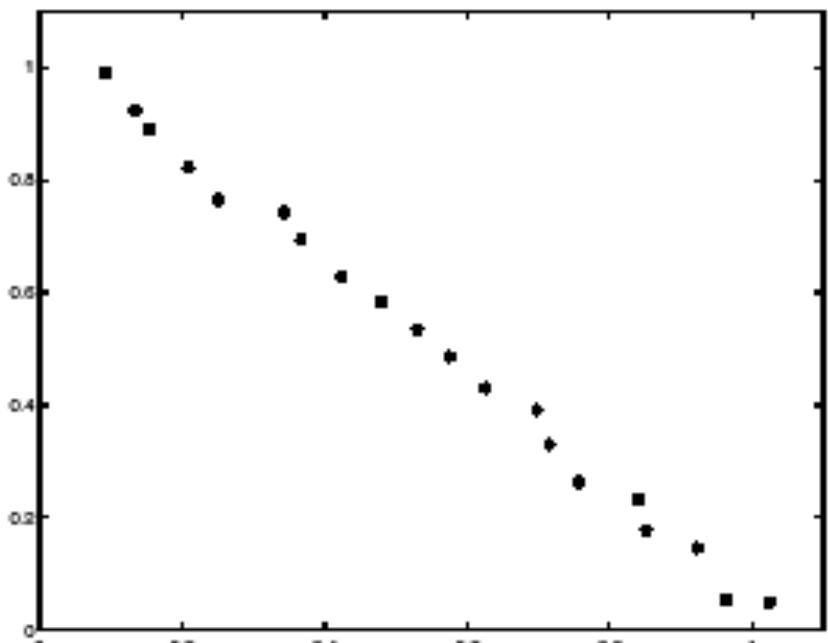


features

votes



Effect of noise

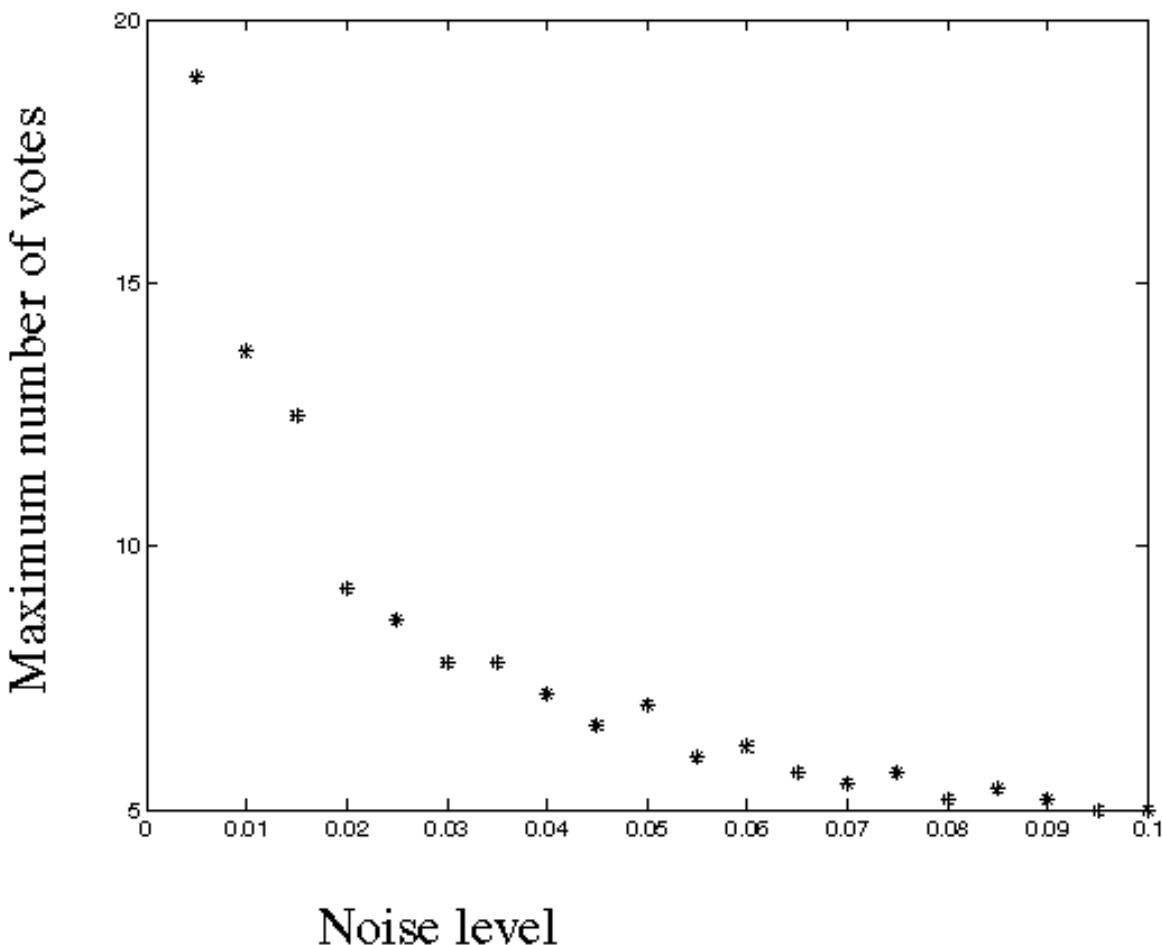


features

votes

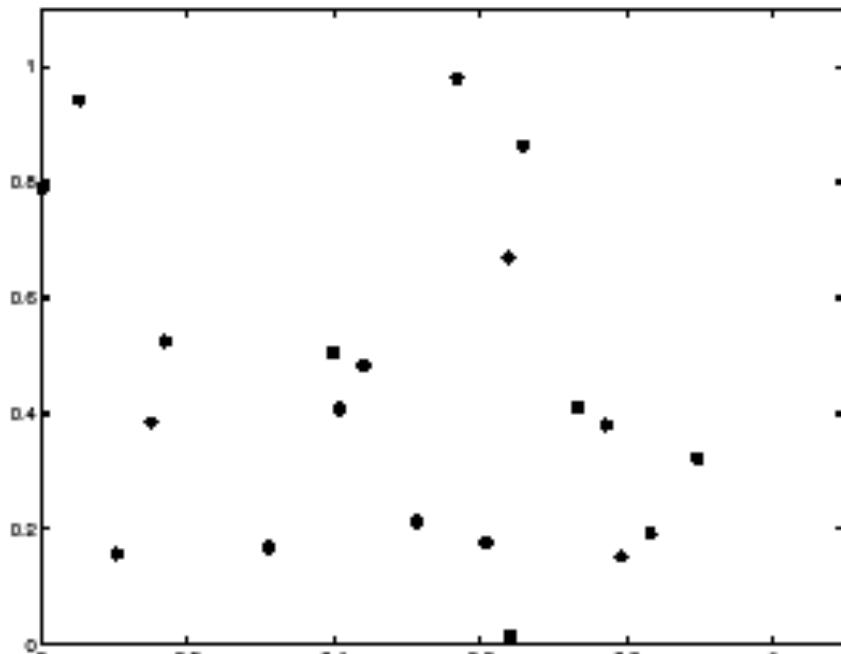
- Peak gets fuzzy and hard to locate

Effect of noise

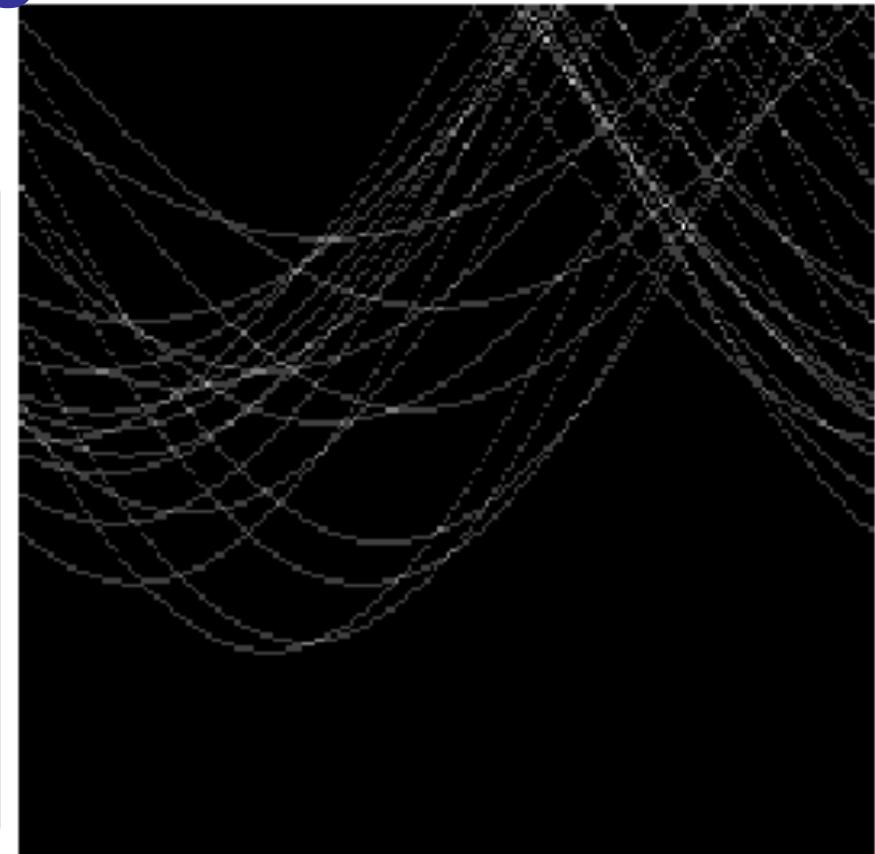


- Number of votes for a line of 20 points with increasing noise.

Random points



features

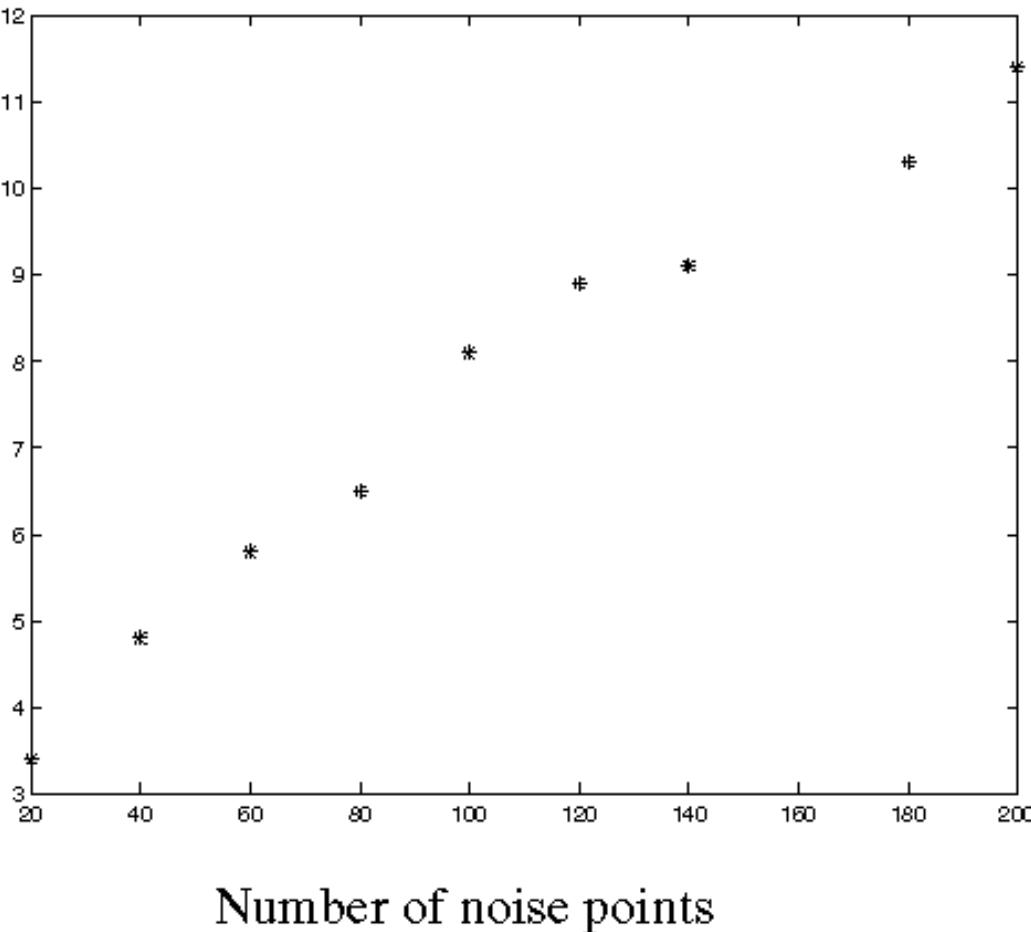


votes

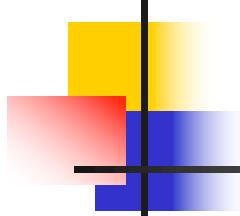
- Uniform noise can lead to spurious peaks in the array

Random points

Maximum number of votes

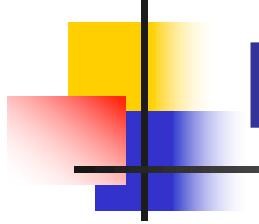


- As the level of uniform noise increases, the maximum number of votes increases too!



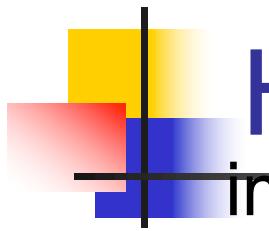
Dealing with noise

- 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
 - Take only edge points with significant gradient magnitude



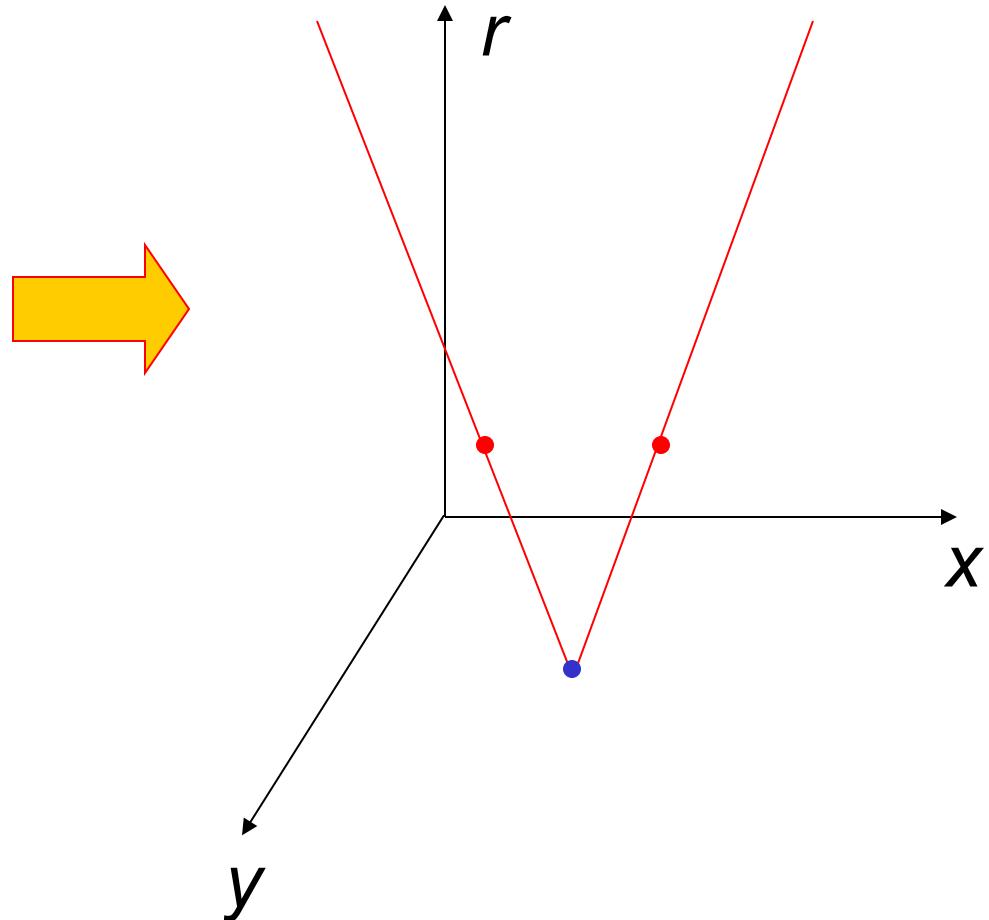
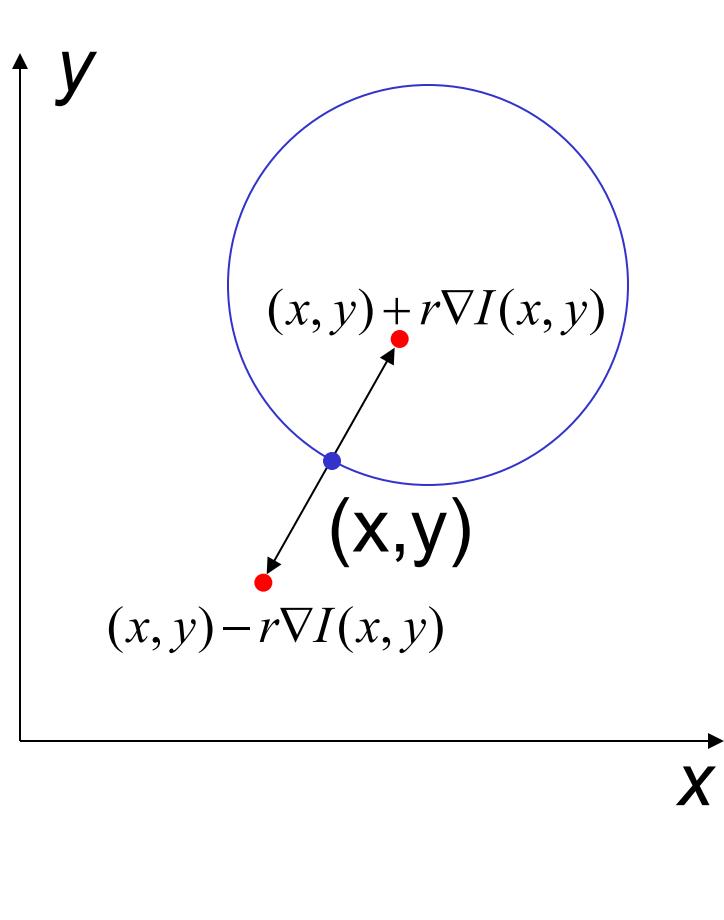
Hough transform for circles

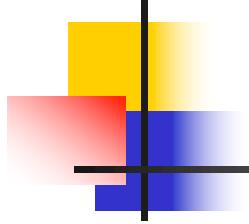
- How many dimensions will the parameter space have?
- Given an oriented edge point, what are all possible bins that it can vote for?



Hough transform for circles

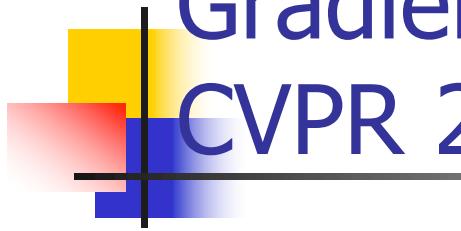
image space Hough parameter space





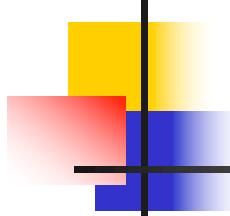
Region descriptors

- Patch descriptors (HOG)
- Image / Sub-Image descriptors (BoVW)



Patch Descriptor: Histogram of Gradients (HoG) (N. Dalal and B. Triggs CVPR 2005)

- Compute centered horizontal and vertical gradients with no smoothing.
- Compute gradient orientation and magnitudes,
- For color image, pick the color channel with the highest gradient magnitude for each pixel.
- For a 64x128 image, divide the image into 16x16 blocks of 50% overlap. → $7 \times 15 = 105$ blocks in total



Histogram of Gradients (HoG)

- Each block: 2x2 cells with size 8x8.
- Quantize the gradient orientation into 9 bins.
- The vote is the gradient magnitude.
- Interpolate votes between neighboring bin center.
- The vote can also be weighted with Gaussian to down-weight the pixels near the edges of the block.
- Concatenate histograms → Feature dimension:
 $105 \times 4 \times 9 = 3,780$

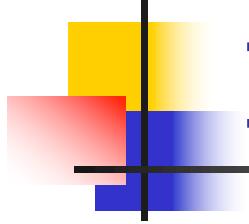


Image / Object Descriptor

- Bag of visual words (Sivic et. al., ICCV'05)
 - Compute key-point based feature descriptors.
 - Quantize them (clustering) to form a finite set of representative descriptors (visual words).
 - Represent by a histogram of visual words.

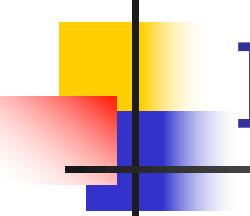
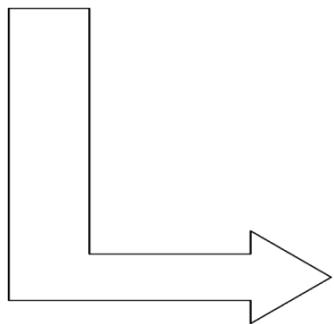


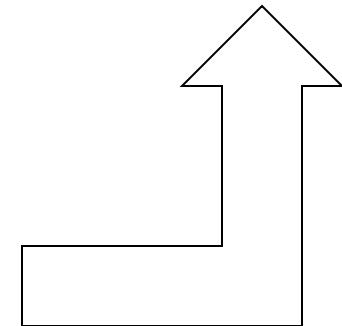
Image Analysis: A brief outline

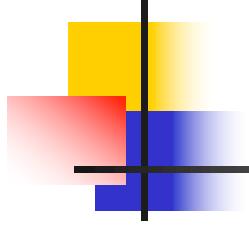
Pre-processing
(Filtering, Enhancement, ...)

End Analysis
(Classification,
Matching, Retrieval)



Feature Extraction
And Description
(SIFT, SURF, HOG,
BoVW)





Thank you!