Computer Vision

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2020-2021

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

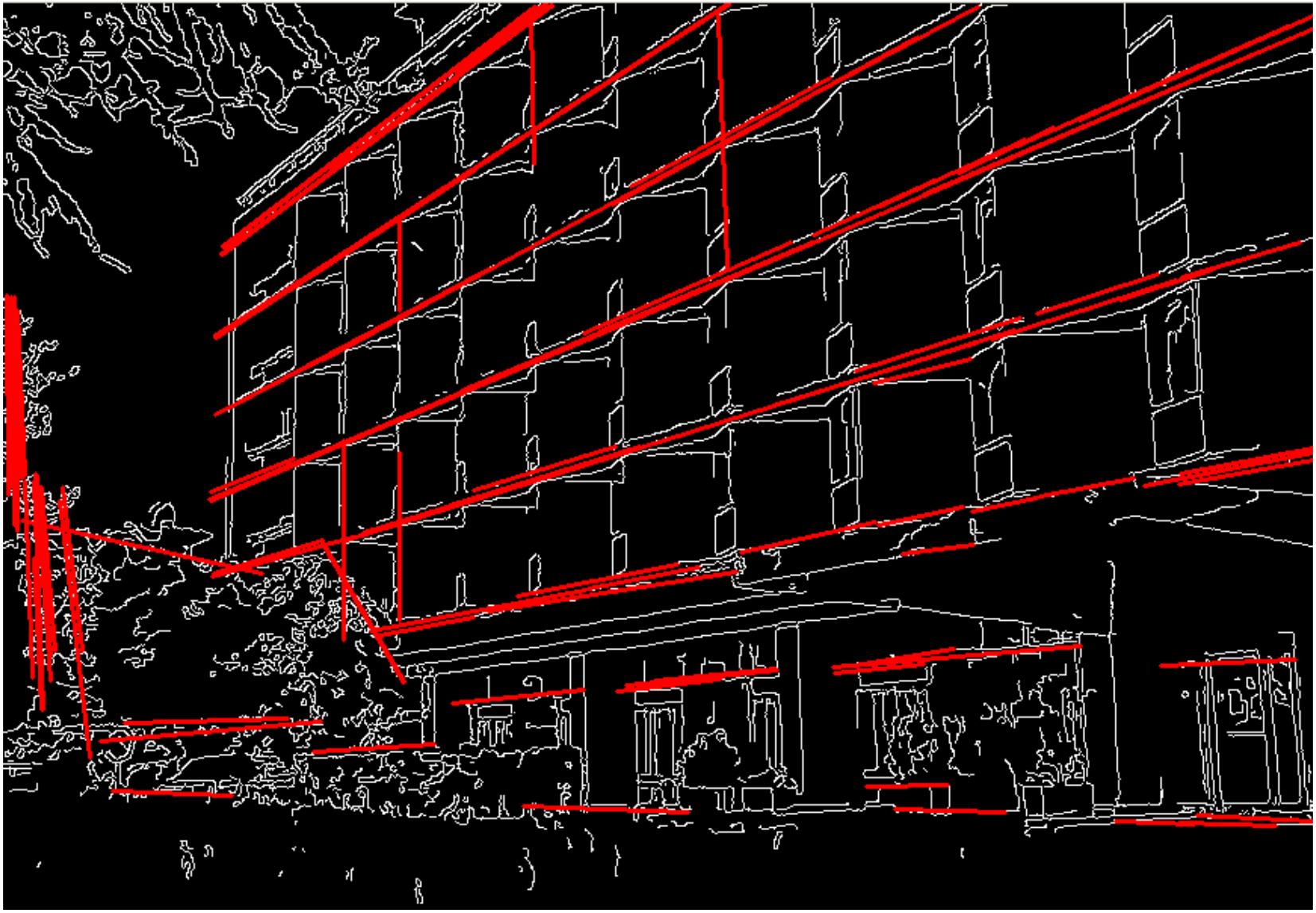
4. **Object Recognition: high – level vision**

   Object classification, object detection, part based models, bovw models

5. **Video understanding**

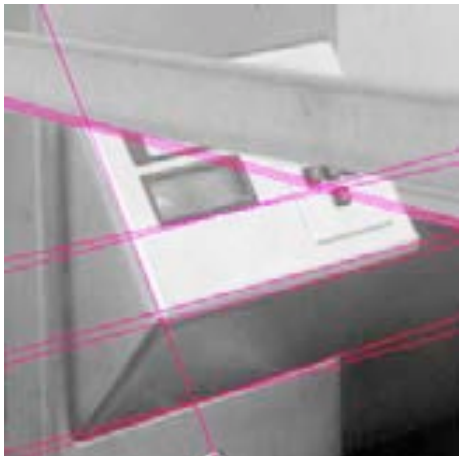
   Object tracking, background subtraction, motion descriptors, optical flow

Fitting



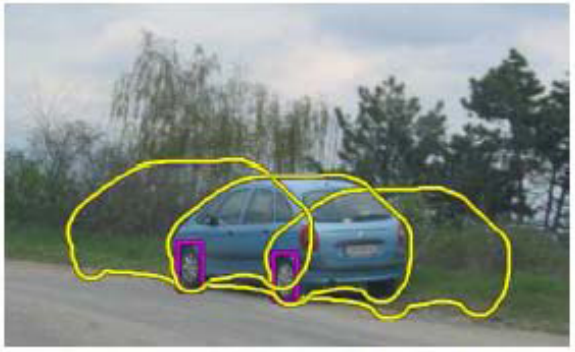
Fitting

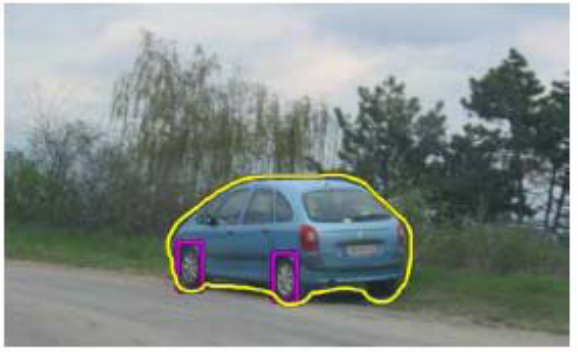
- Choose a *parametric model* to represent a set of features



simple model: lines

simple model: circles

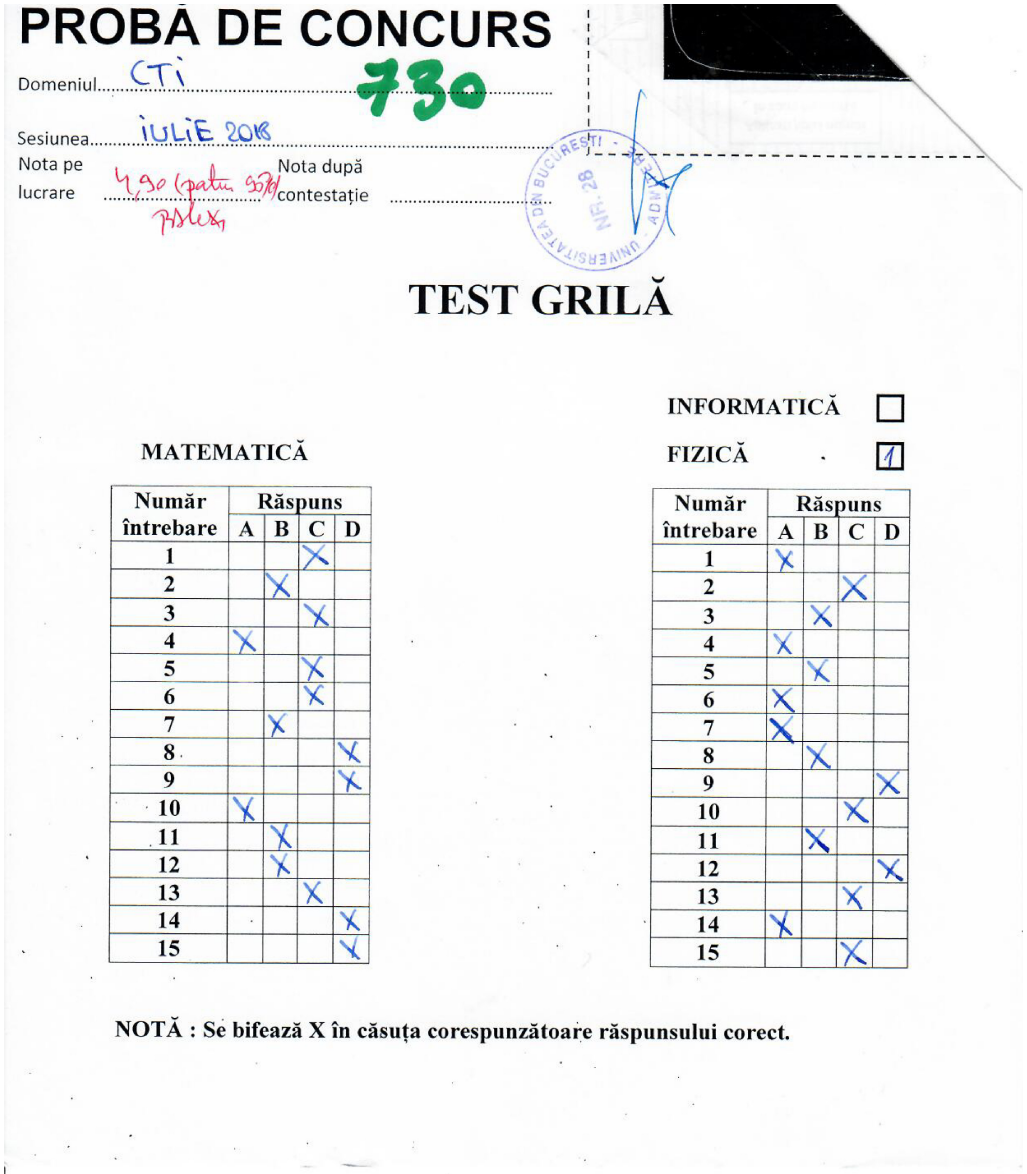




complicated model: car

Slide credit : K. Grauman

Lab 2: Line detection - application



PROBĂ DE CONCURS

Domeniul CTI 730

Sesiunea IULIE 2018

Nota pe lucrare 4,90 (patru 90) Nota după contestație .....

TEST GRILĂ

INFORMATICĂ ☐

FIZICĂ ☑

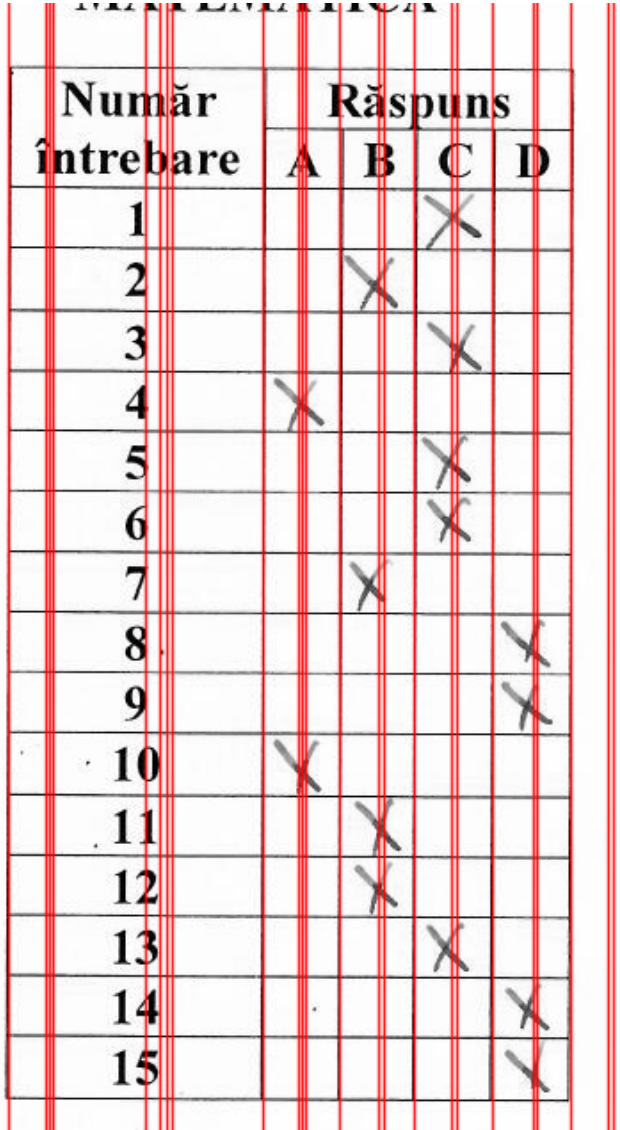
MATEMATICĂ

Număr întrebare	Răspuns A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

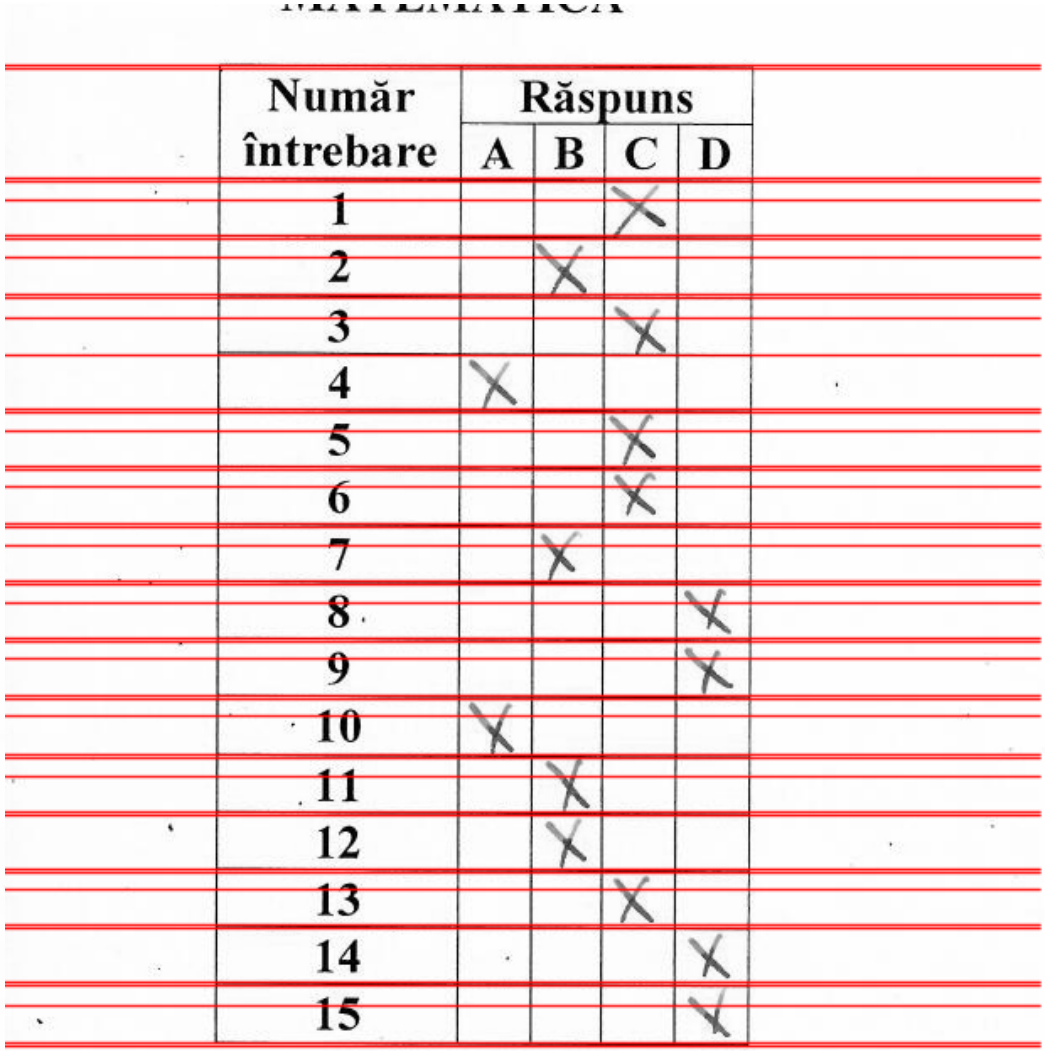
Număr întrebare	Răspuns A	B	C	D
1	X			
2			X	
3		X		
4	X			
5		X		
6	X			
7	X			
8		X		
9				X
10			X	
11		X		
12				X
13			X	
14	X			
15			X	

NOTĂ : Se bifează X în căsuța corespunzătoare răspunsului corect.

Lab 2: Line detection - application

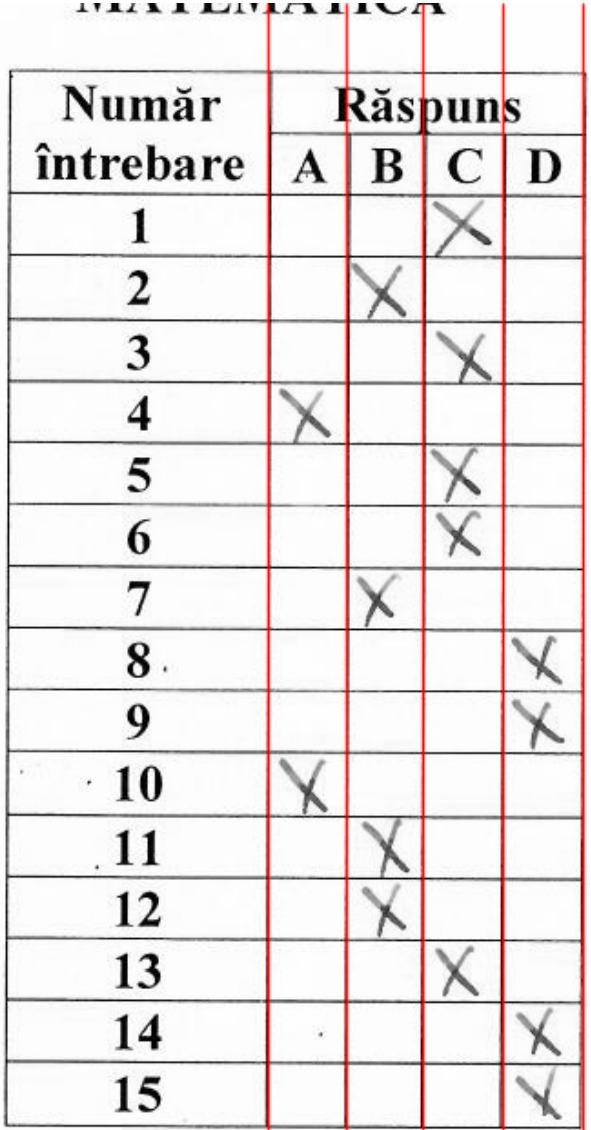


Număr întrebare	Răspuns A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

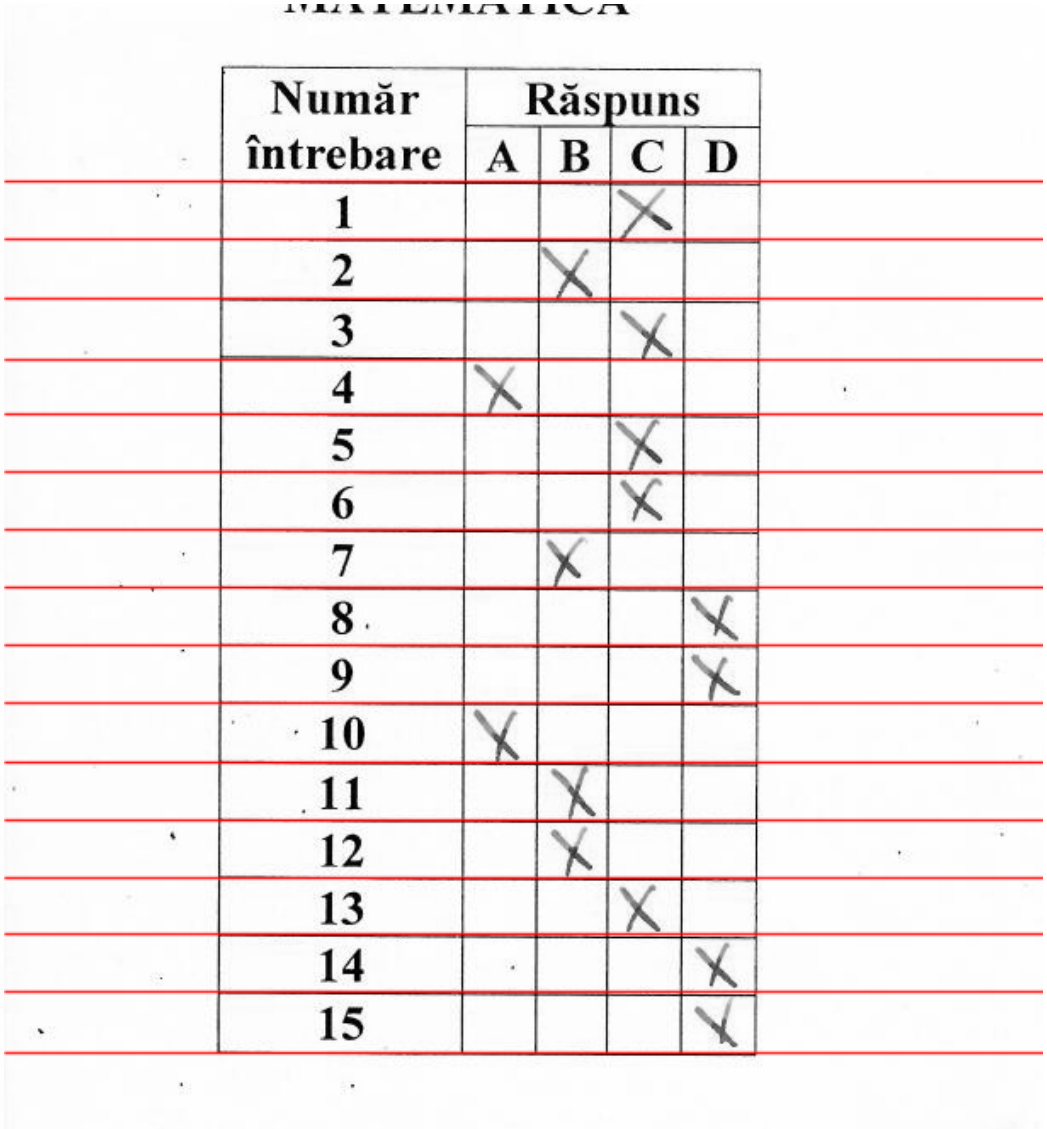


Număr întrebare	Răspuns A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

Lab 2: Line detection - application



Număr întrebare	Răspuns			
	A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X



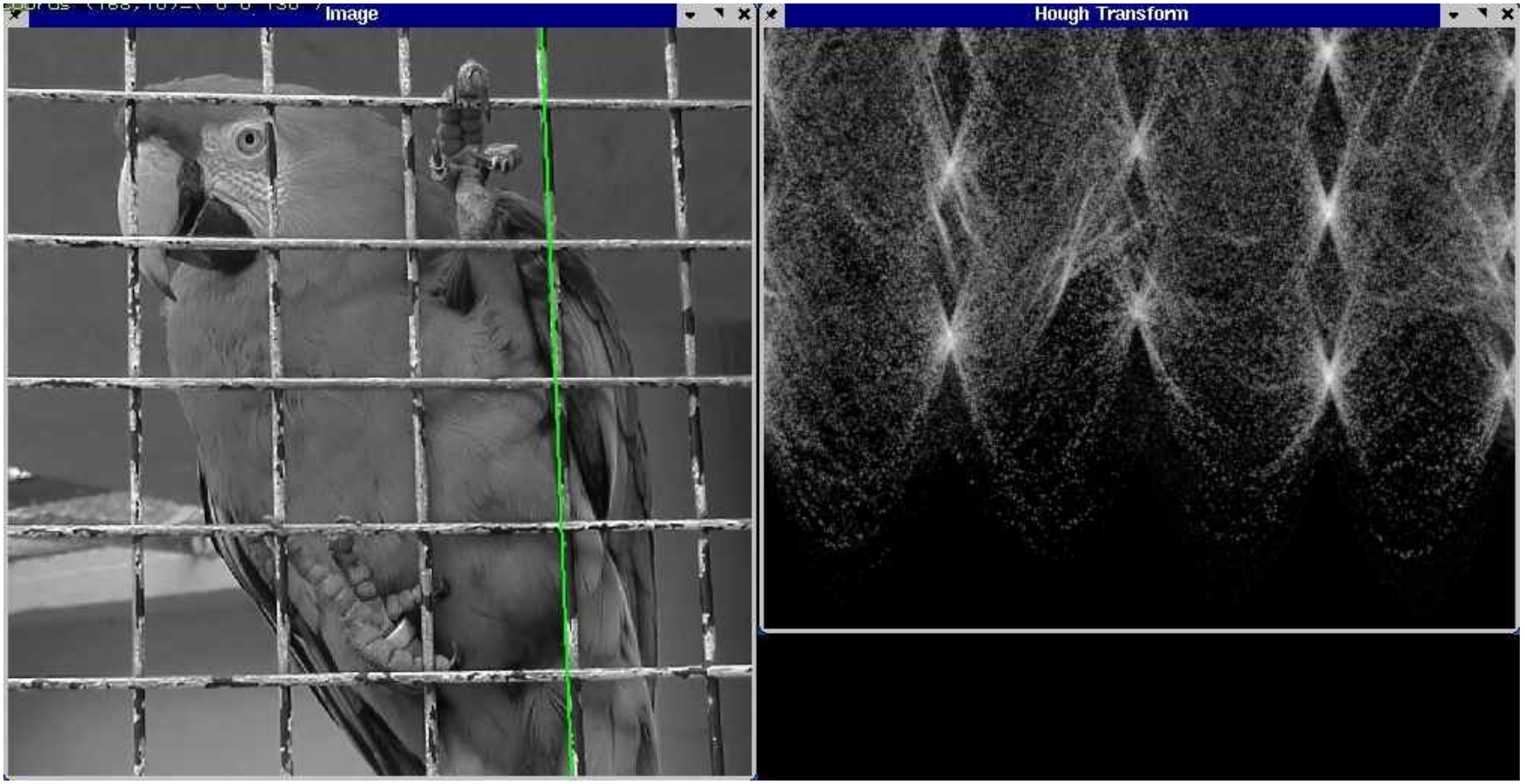
Număr întrebare	Răspuns			
	A	B	C	D
1			X	
2		X		
3			X	
4	X			
5			X	
6			X	
7		X		
8				X
9				X
10	X			
11		X		
12		X		
13			X	
14				X
15				X

Fitting: challenges

- 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

Slide credit : S. Lazebnik

Fitting: The Hough transform



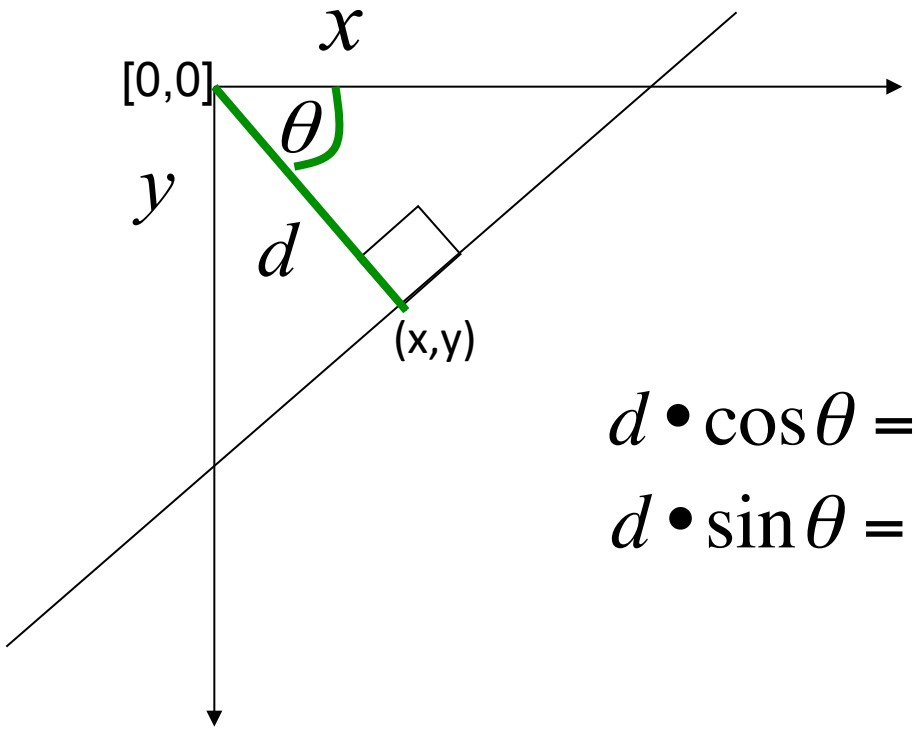
Slide credit : S. Lazebnik

Polar representation

Problems with the (m,b) space:

- Unbounded parameter domains
- Vertical lines require infinite m

Alternative: polar representation



d: distance from origin to line (length of the perpendicular)

θ: angle between the perpendicular and Ox axis

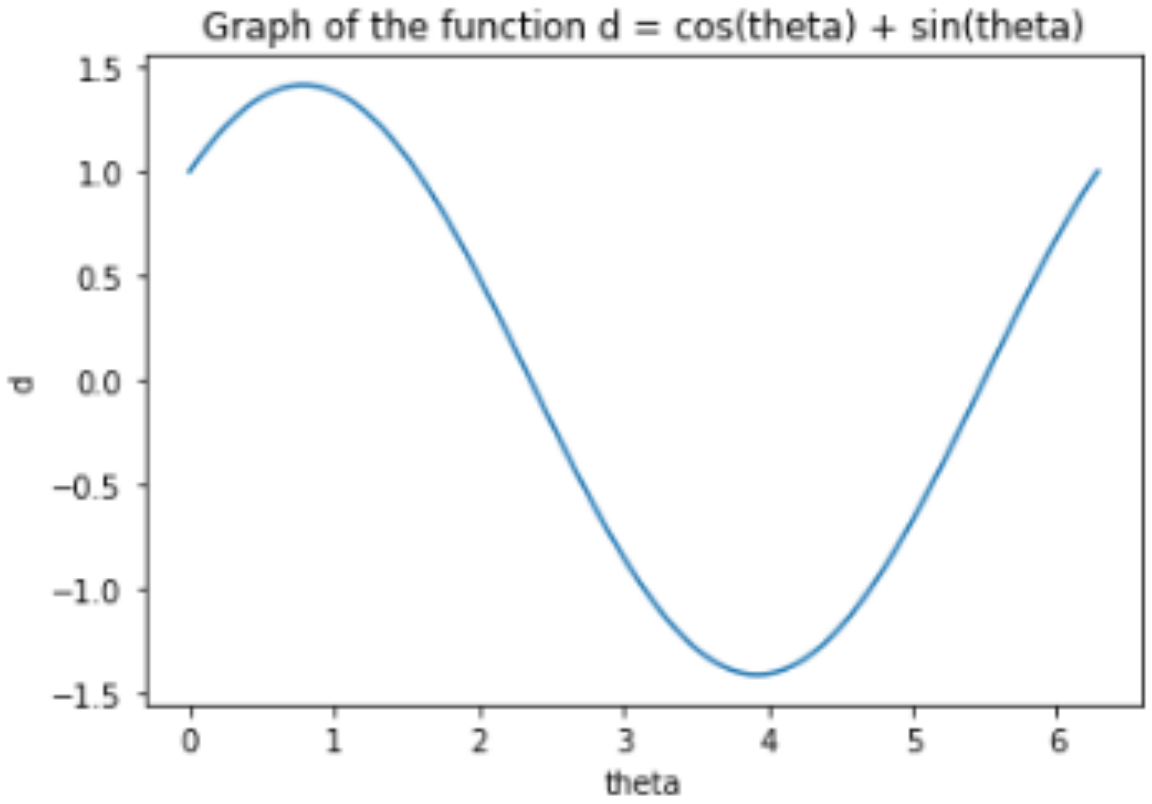
$$d \bullet \cos\theta = x \qquad d \bullet (\cos\theta)^2 = x \bullet \cos\theta$$

$$d \bullet \sin\theta = y \qquad d \bullet (\sin\theta)^2 = y \bullet \sin\theta$$

$$d = x \bullet \cos\theta + y \bullet \sin\theta$$

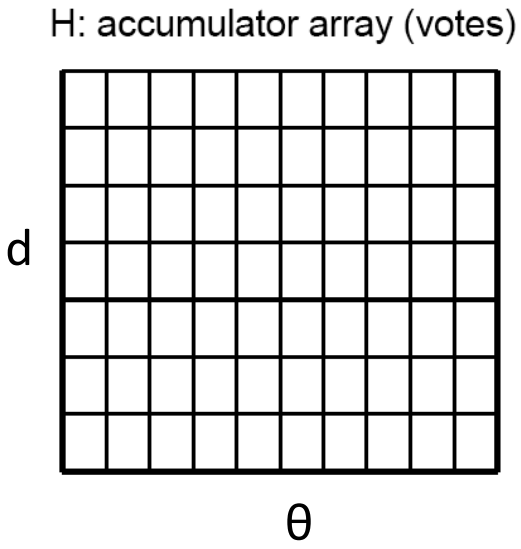
Point in the image space → sinusoid curve in the Hough space

```python
import numpy as np
import matplotlib.pyplot as plt
x, y = 1,1
theta = np.linspace(0,2*np.pi,100)
d = x * np.cos(theta) + y * np.sin(theta)
plt.figure,
plt.plot(theta, d)
plt.xlabel('theta')
plt.ylabel('d')
plt.title('Graph of the function d = cos(theta) + sin(theta)')
plt.show()
```



Hough transform-algorithm outline

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

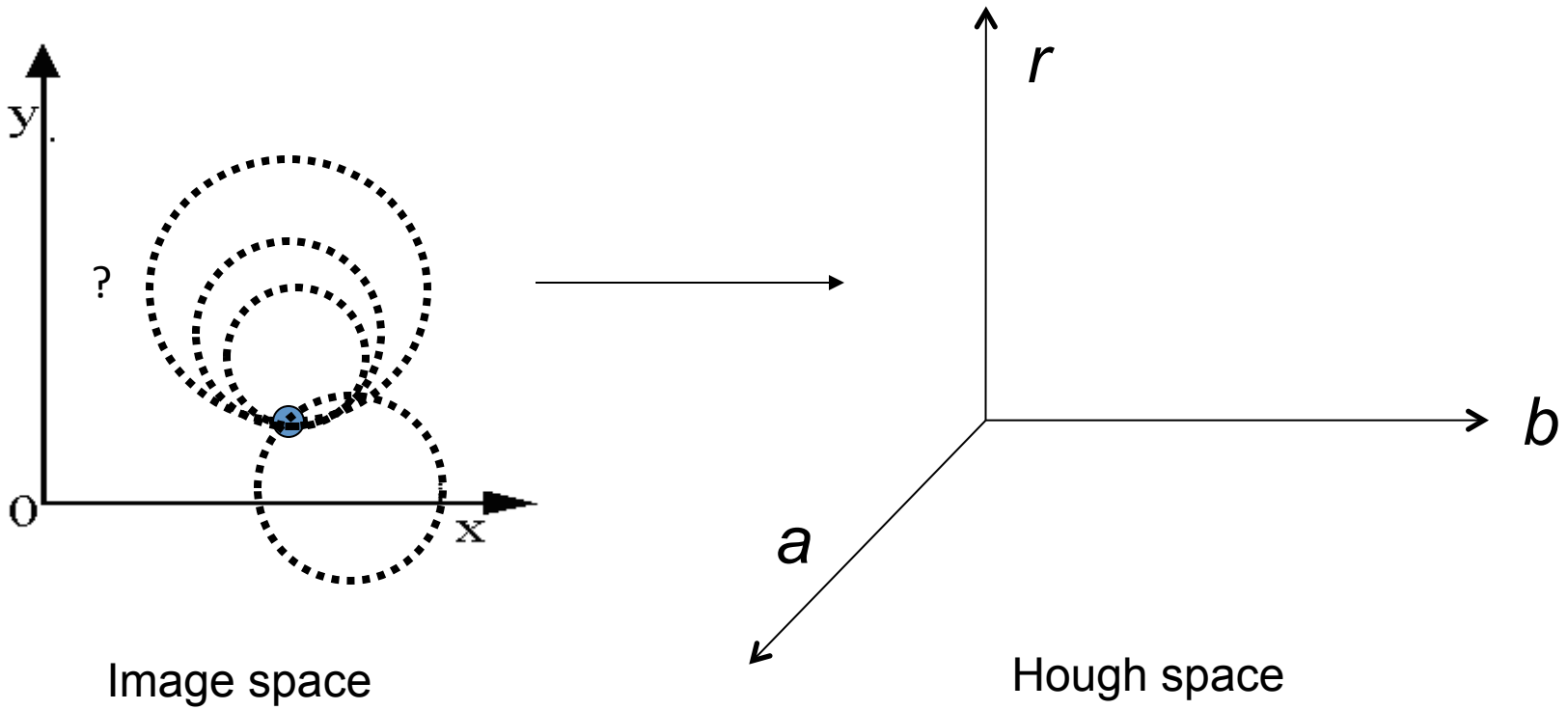


Hough transform for circles

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

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

- For unknown radius r



Slide credit: Kristen Grauman

Hough transform for circles

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

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

- For unknown radius r

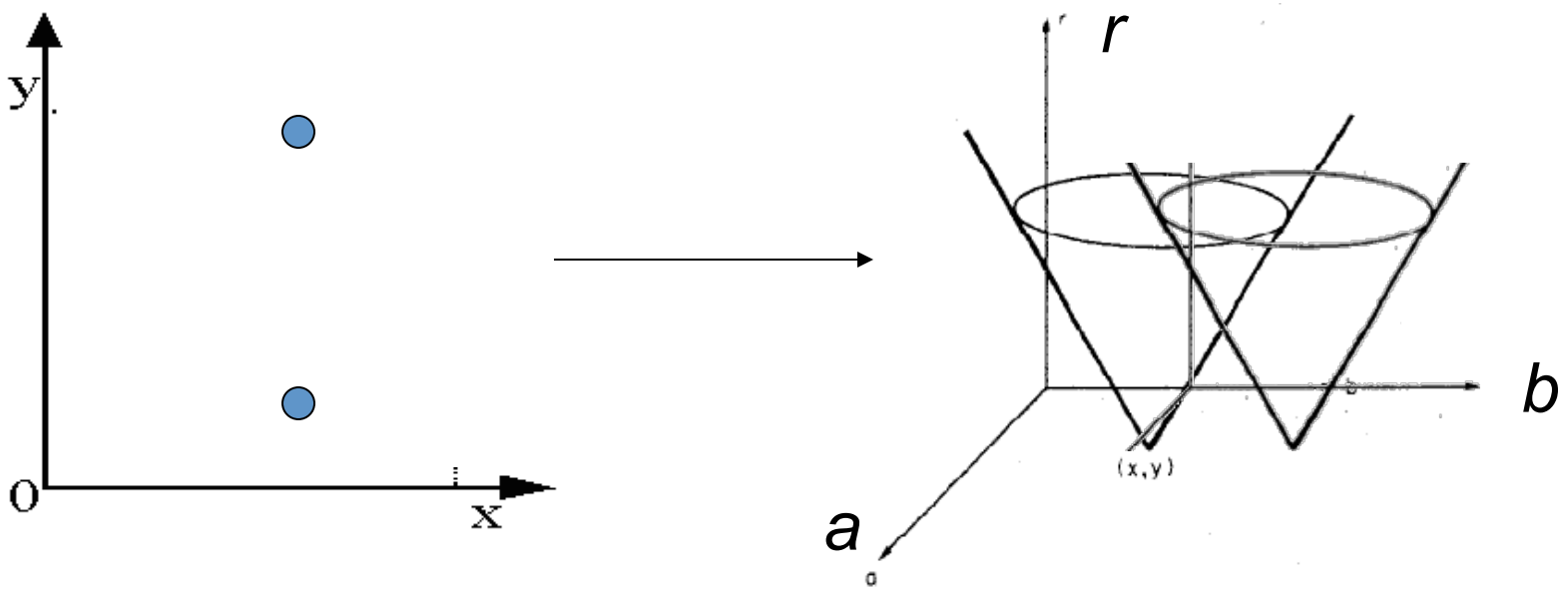


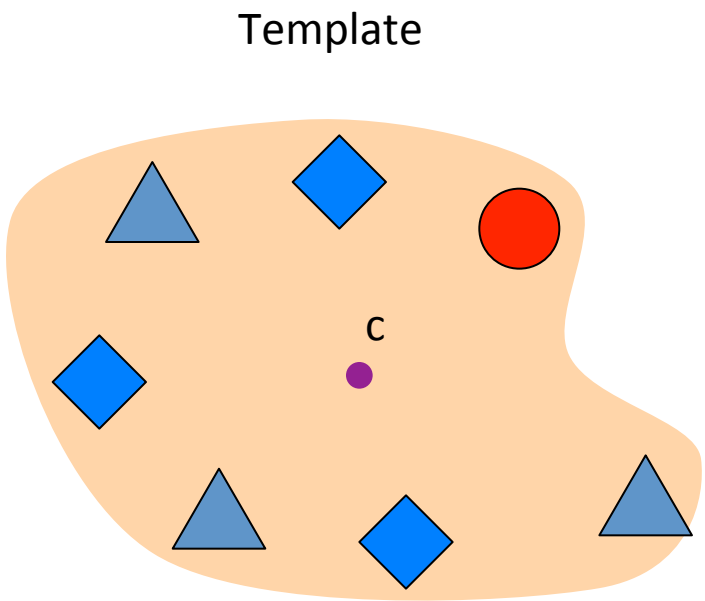
Image space

Hough space

Slide credit: Kristen Grauman

Generalized Hough transform

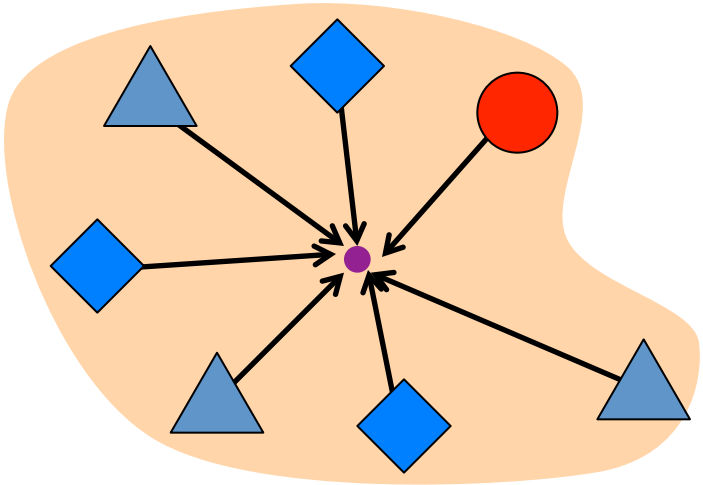
- We want to find a template defined by its reference point (center) and several distinct types of landmark points in stable spatial configuration



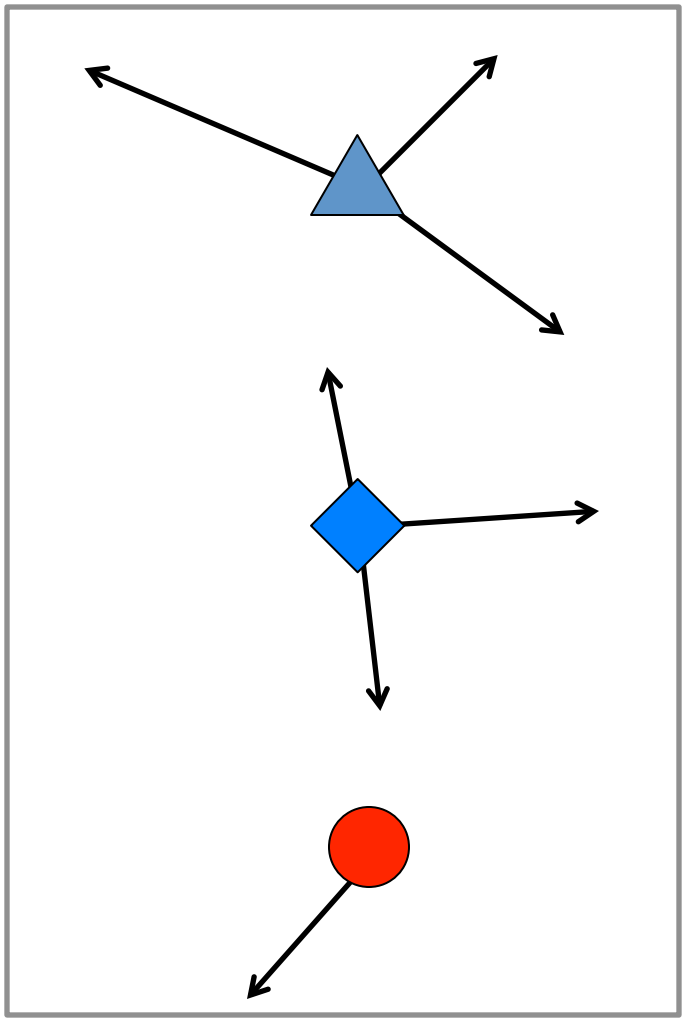
Generalized Hough transform

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

Template

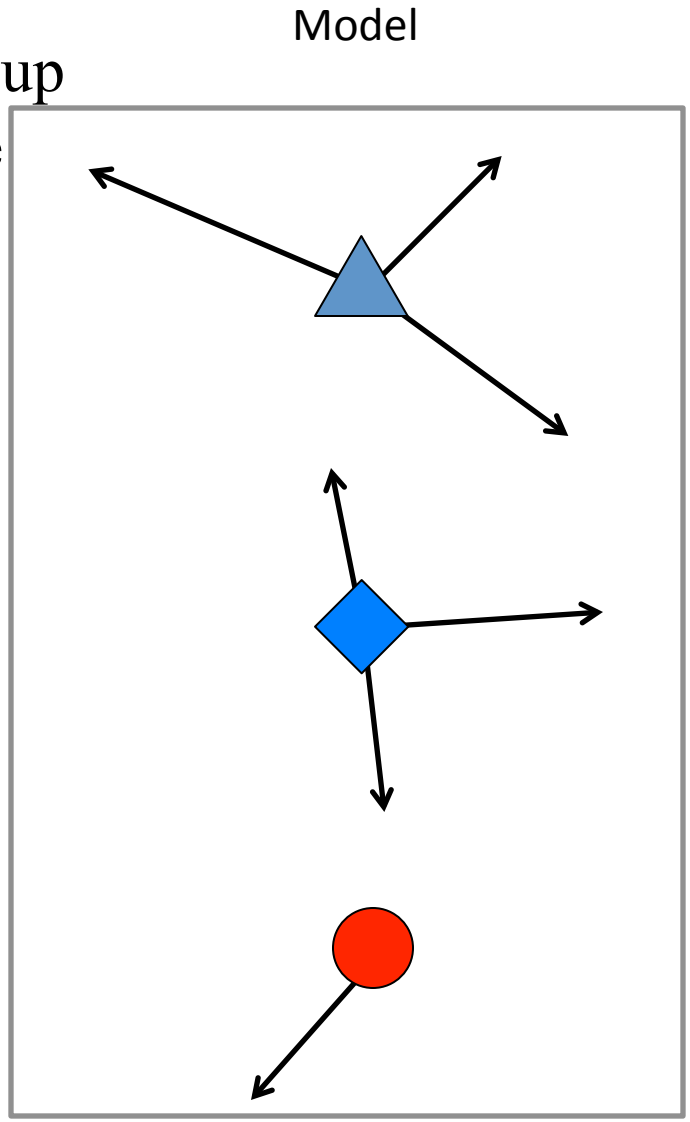


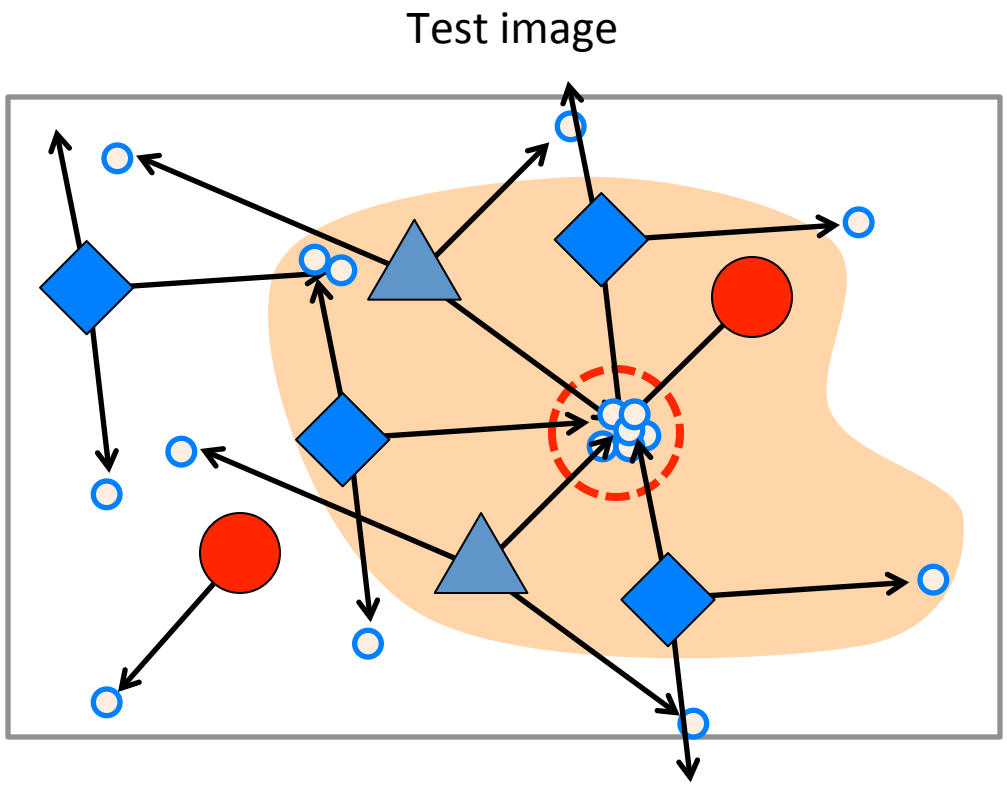
Model



Generalized Hough transform

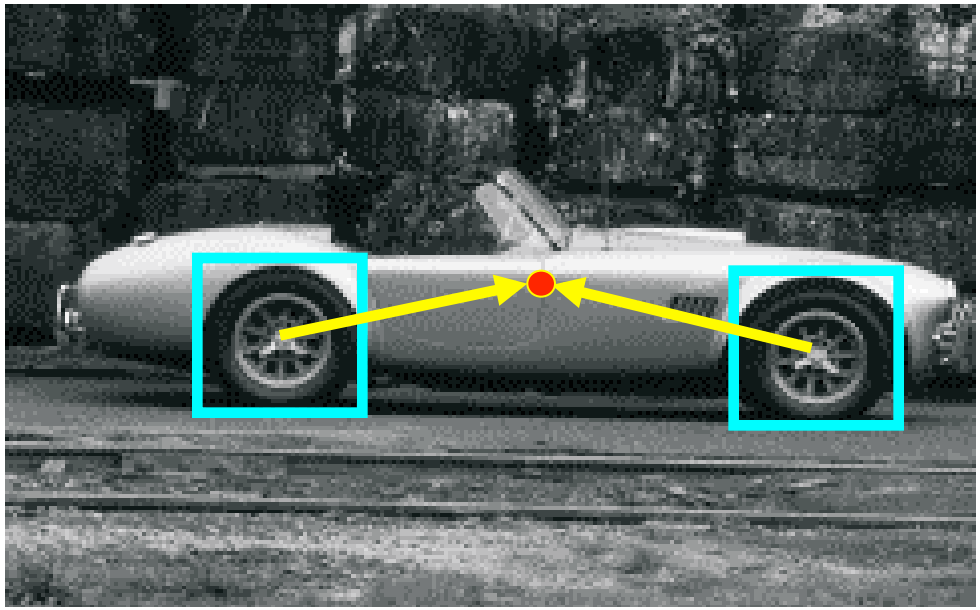
- Detecting the template:
  - For each feature in a new image, look up that feature type in the model and vote for the possible center locations associated with that type in the model





Application in recognition

- Index displacements by “visual codeword”



training image

visual codeword with displacement vectors

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

Application in recognition

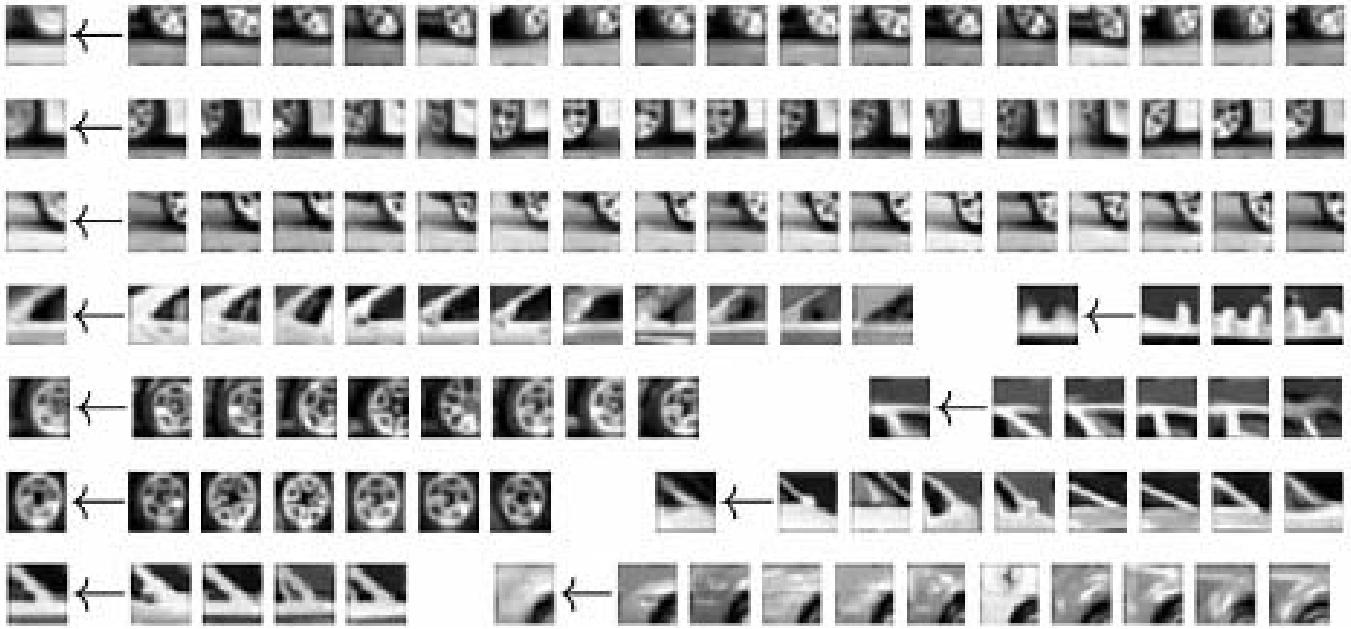
- Index displacements by “visual codeword”

test image

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

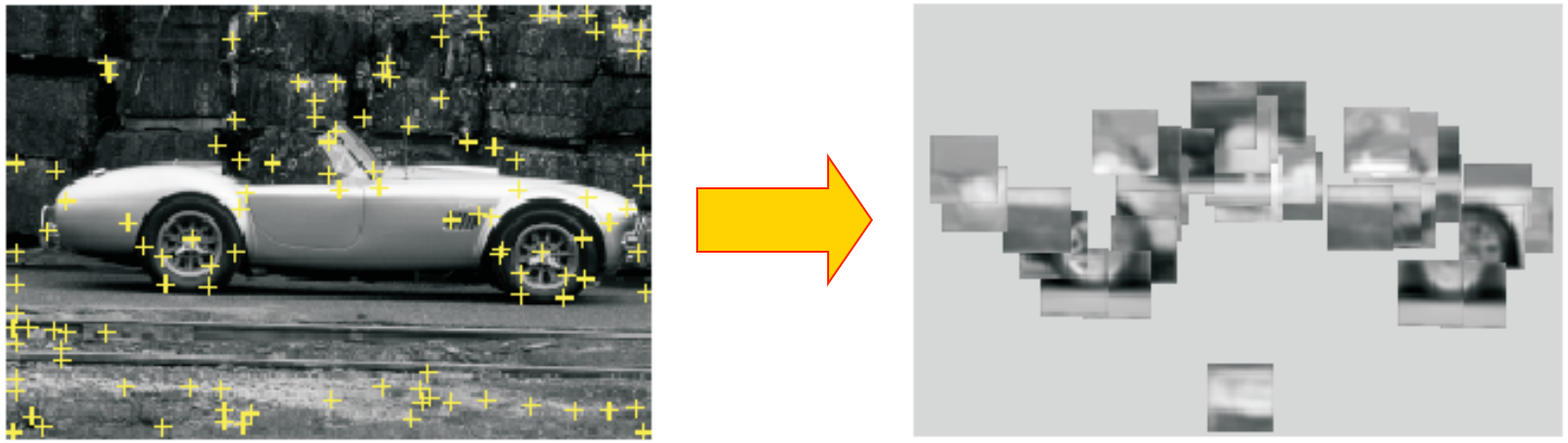
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering



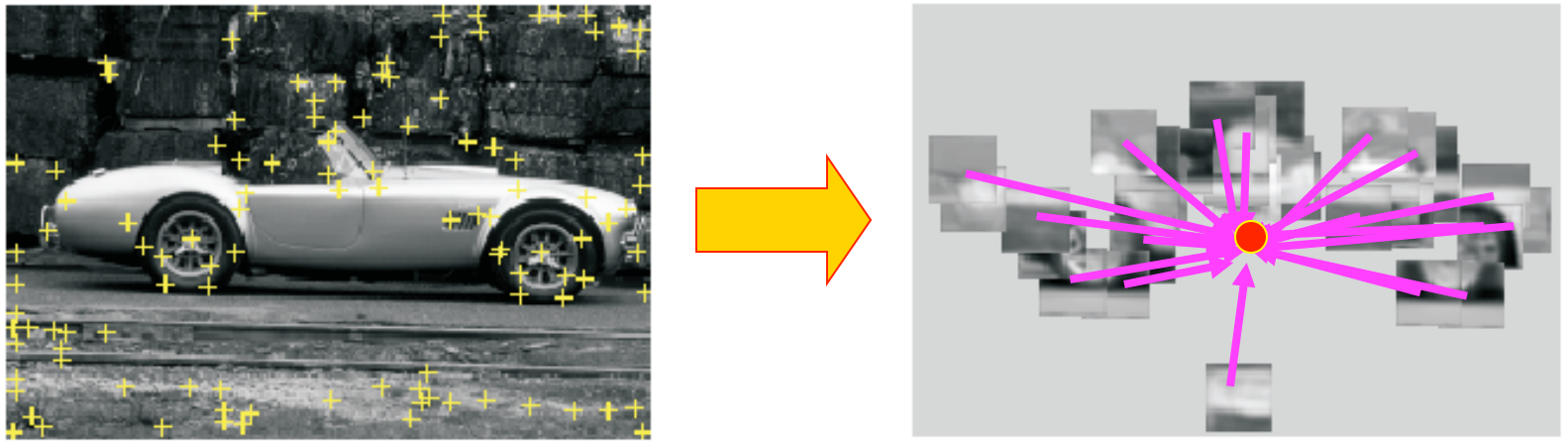
Implicit shape models: Training

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



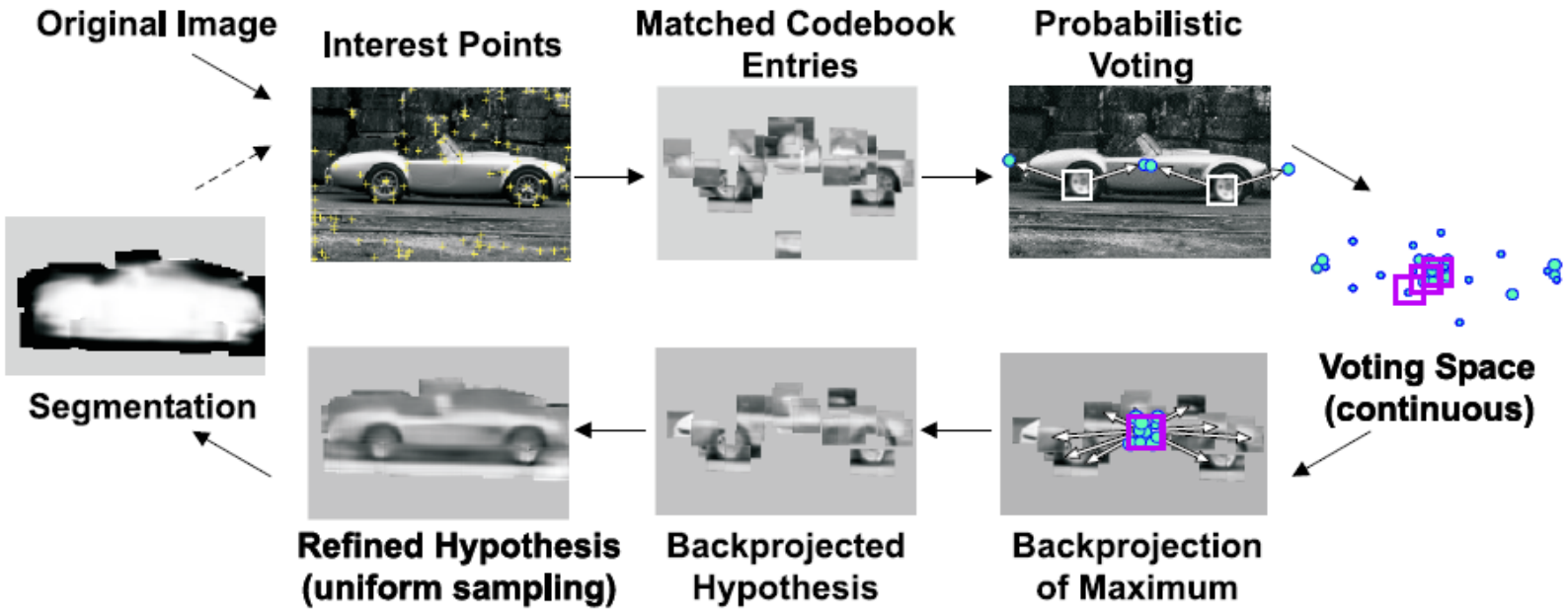
Implicit shape models: Training

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

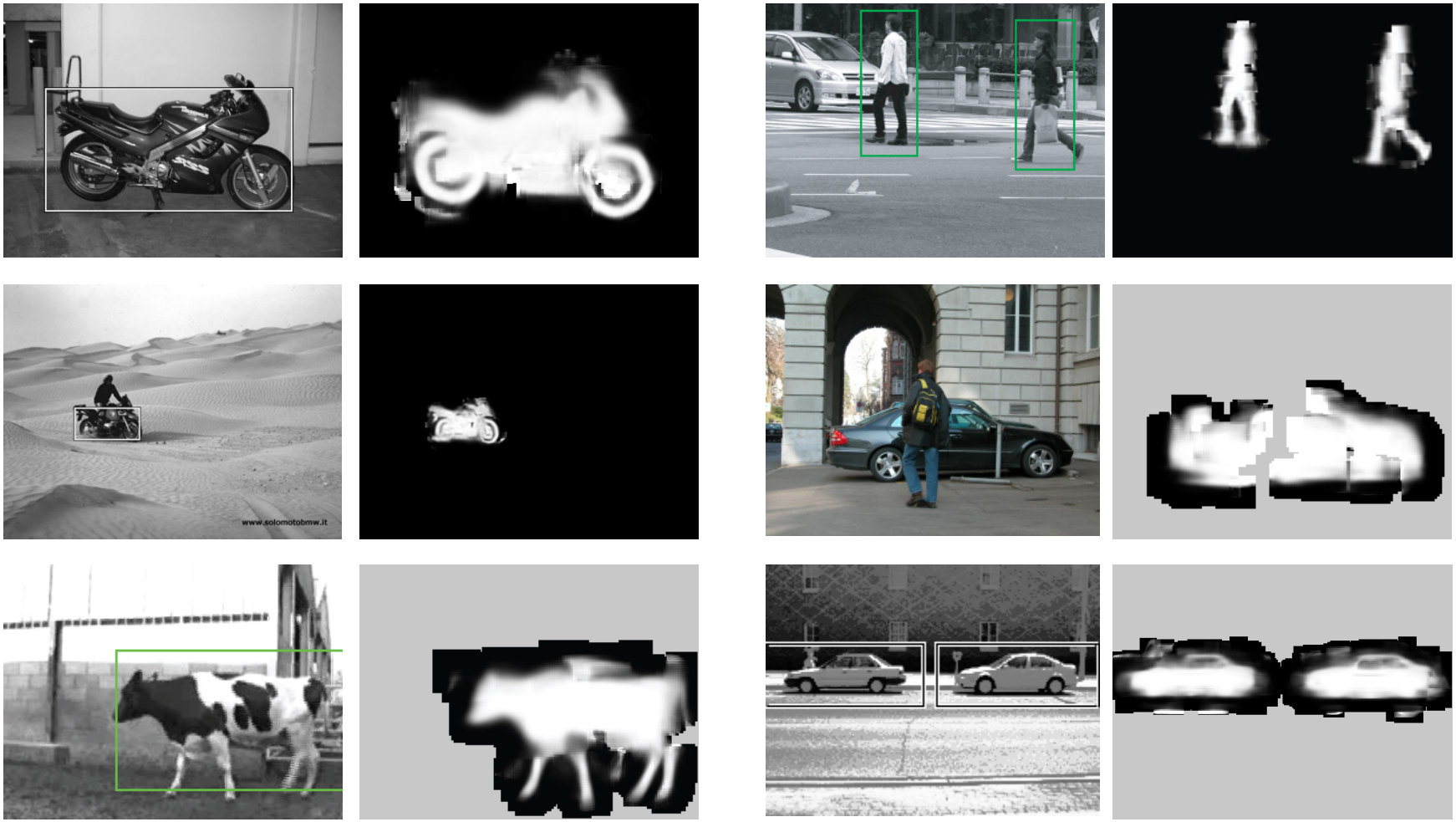


Implicit shape models: Testing

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

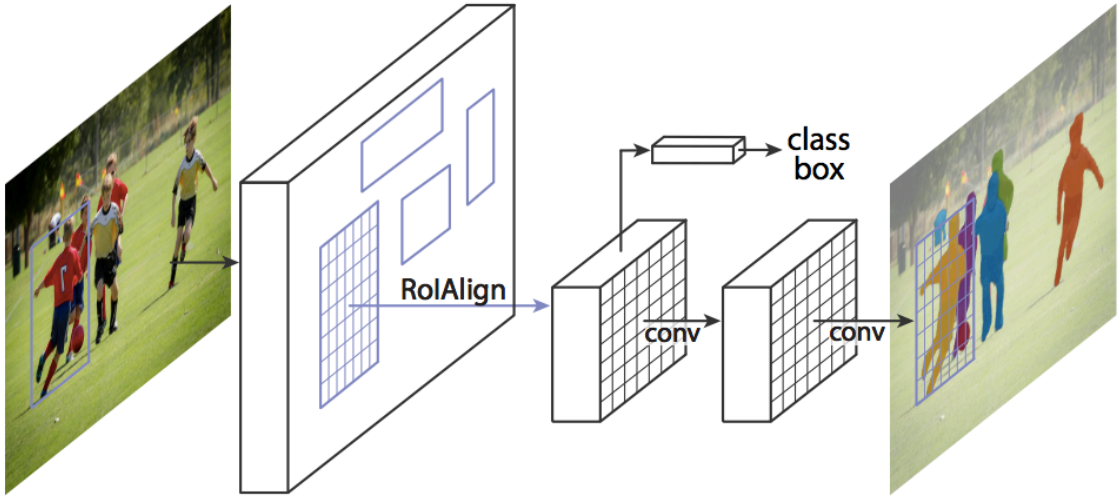


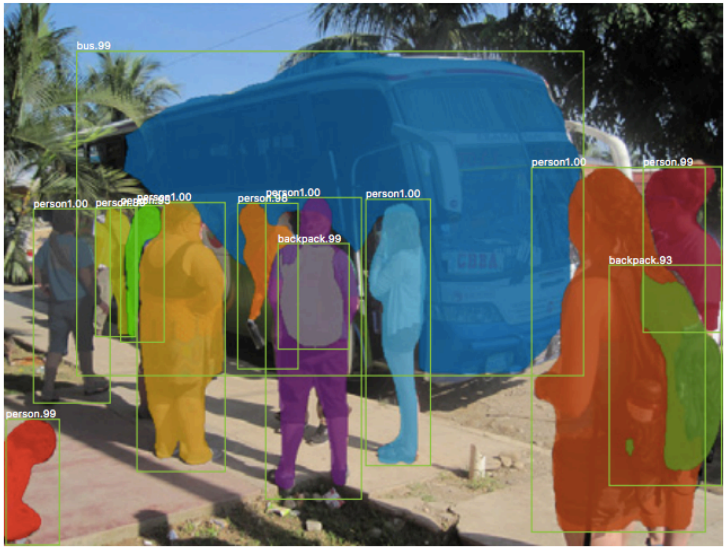
Additional examples

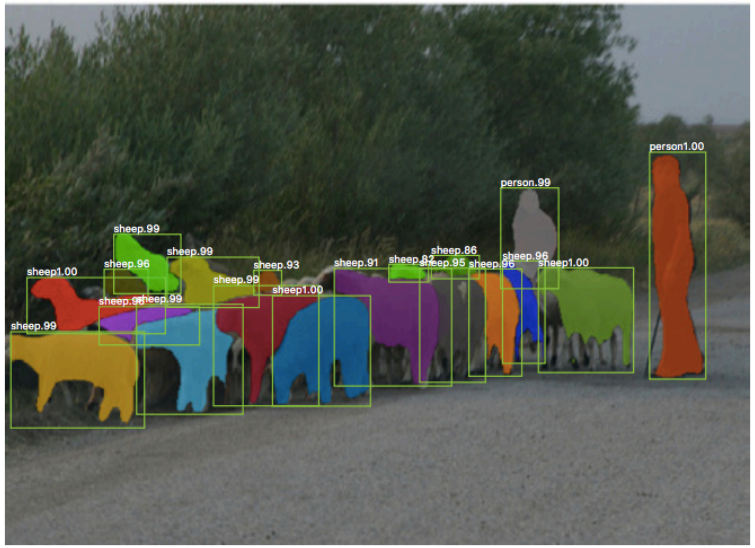


B. Leibe, A. Leonardis, and B. Schiele,
Robust Object Detection with Interleaved Categorization and Segmentation,
IJCV 77 (1-3), pp. 259-289, 2008.

These days: Mask R-CNN







K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017

Fitting: challenges

- 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

Slide credit : S. Lazebnik

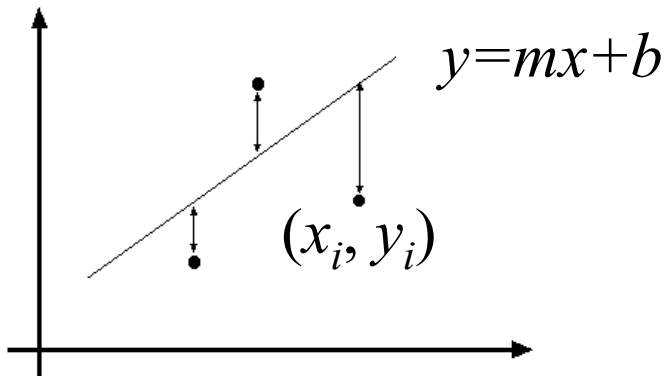
Least squares line fitting

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Line equation: $y_i = m x_i + b$

Find (m, b) to minimize

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



$$E = \|Y - XB\|^2 \quad \text{where} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \quad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

$$E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

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

$$X^T XB = X^T Y$$ *Normal equations:* least squares solution to $XB=Y$

$$B = \left(X^T X\right)^{-1} X^T Y$$ *Solution, if $X^T X$ is invertible*

Slide credit : S. Lazebnik

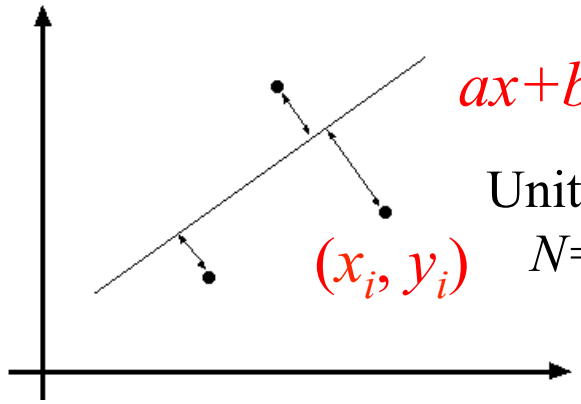
Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines

Slide credit : S. Lazebnik

Total least squares

Distance between point (x_i, y_i) and line $ax+by-d=0$ ($a^2+b^2=1$): $|ax_i + by_i - d|$



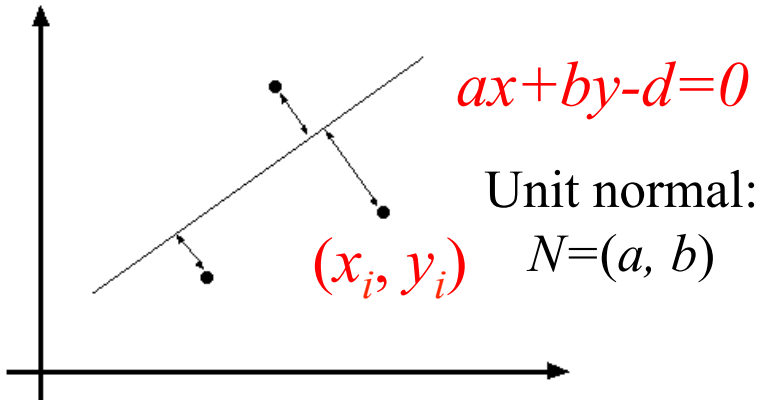
Slide credit : S. Lazebnik

Total least squares

Distance between point (x_i, y_i) and line $ax+by-d=0$ ($a^2+b^2=1$): $|ax_i + by_i - d|$

Find (a, b, d) to minimize the sum of squared *perpendicular* distances

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

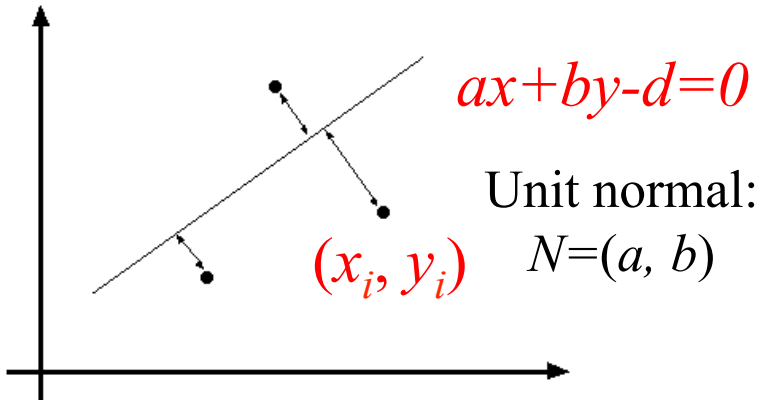


Slide credit : S. Lazebnik

Total least squares

Distance between point (x_i, y_i) and line $ax+by-d=0$ ($a^2+b^2=1$): $|ax_i + by_i - d|$

Find (a, b, d) to minimize the sum of squared *perpendicular* distances



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

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

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

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

$$\frac{dE}{dN} = 2(U^T U)N = 0$$

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

Slide credit : S. Lazebnik

Total least squares

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

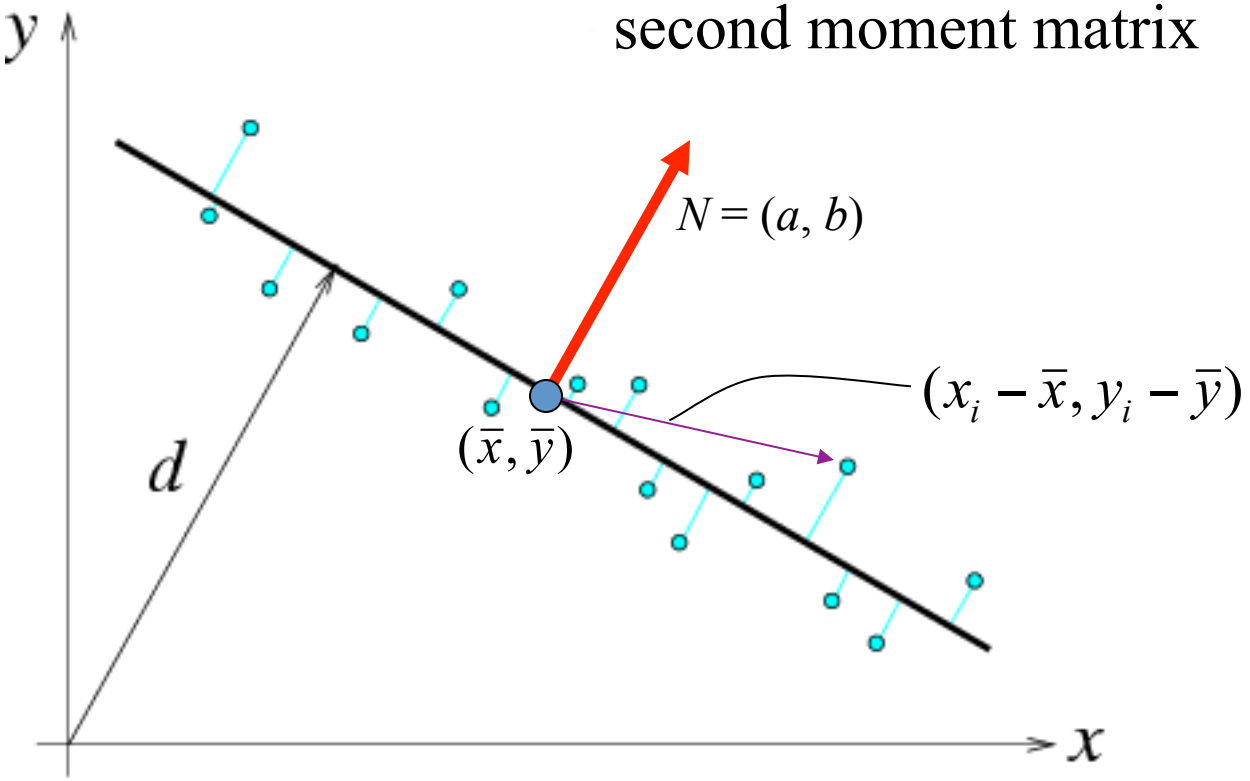
second moment matrix

Slide credit : S. Lazebnik

Total least squares

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

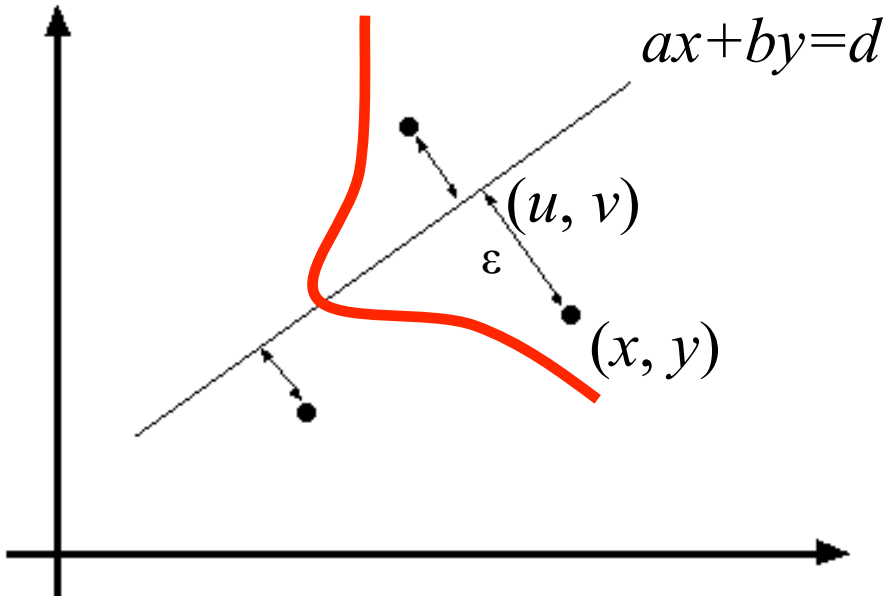
second moment matrix



Slide credit : S. Lazebnik

Least squares as likelihood maximization

- **Generative model**: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line



$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}$$

point on the line

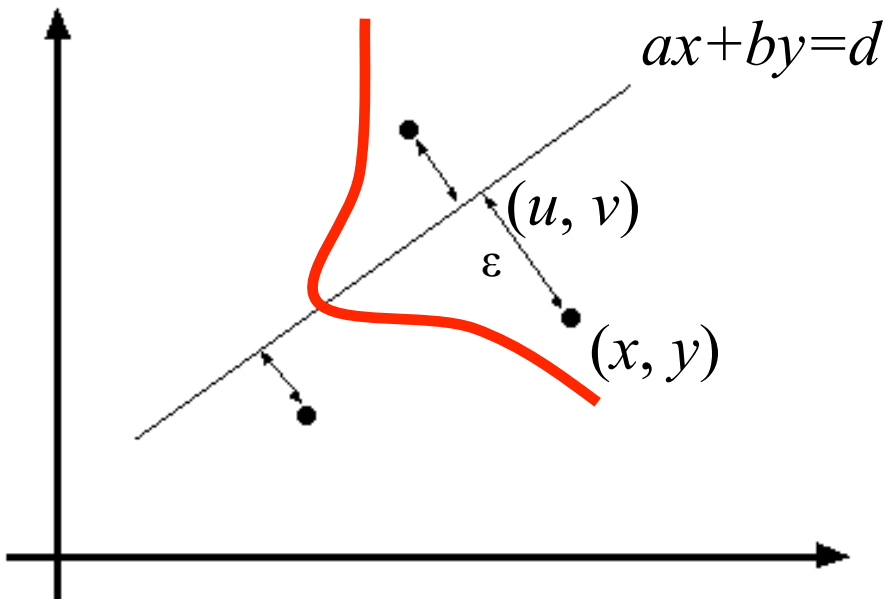
noise: sampled from zero-mean Gaussian with std. dev. σ

normal direction

Slide credit : S. Lazebnik

Least squares as likelihood maximization

- **Generative model**: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line



$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}$$

Likelihood of points given line parameters (*a*, *b*, *d*):

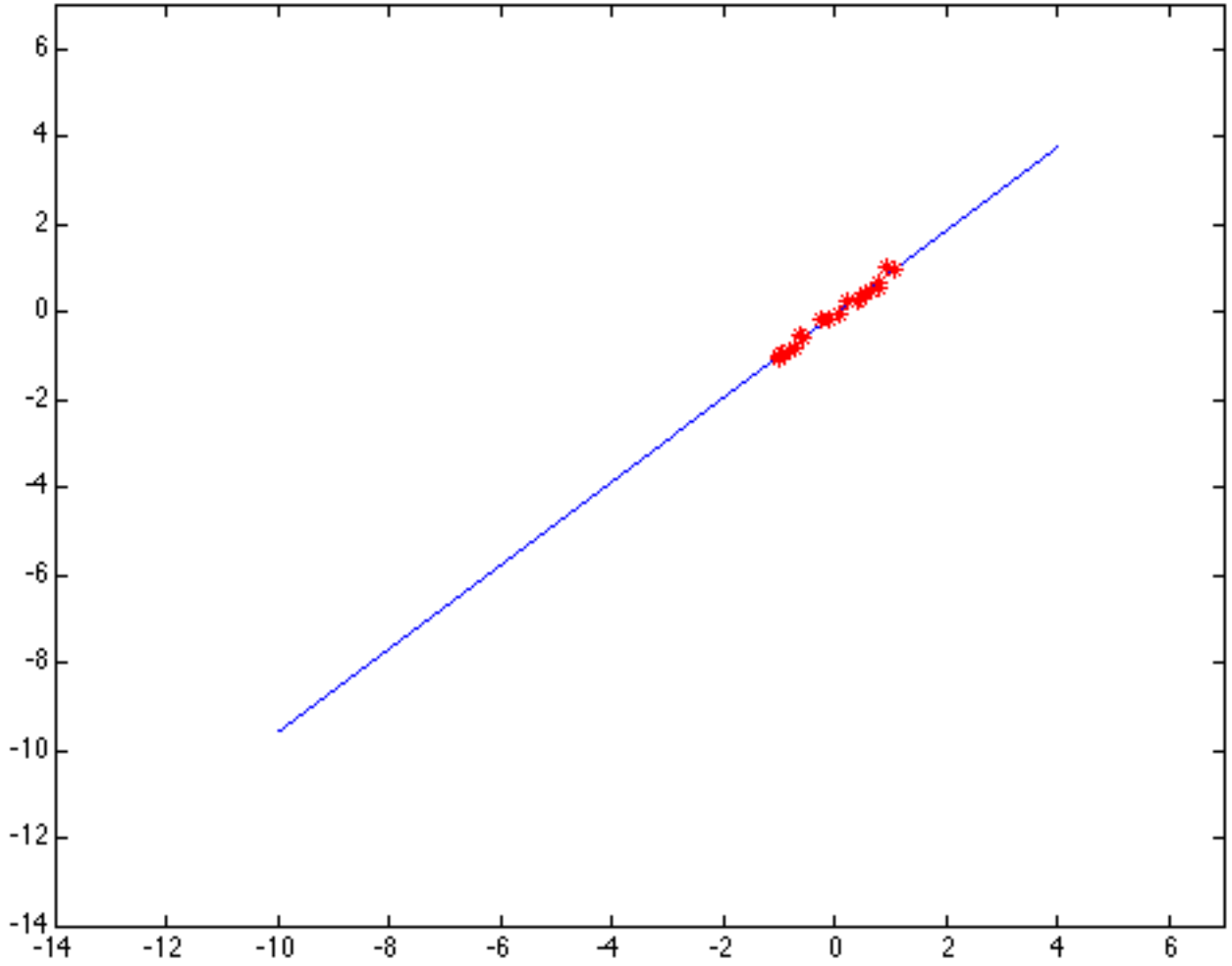
$$P(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = \prod_{i=1}^{n} P(x_i, y_i \mid a, b, d) \propto \prod_{i=1}^{n} \exp\left(-\frac{(ax_i + by_i - d)^2}{2\sigma^2} \right)$$

Log-likelihood: $$L(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (ax_i + by_i - d)^2$$

Slide credit : S. Lazebnik

Least squares: Robustness to noise

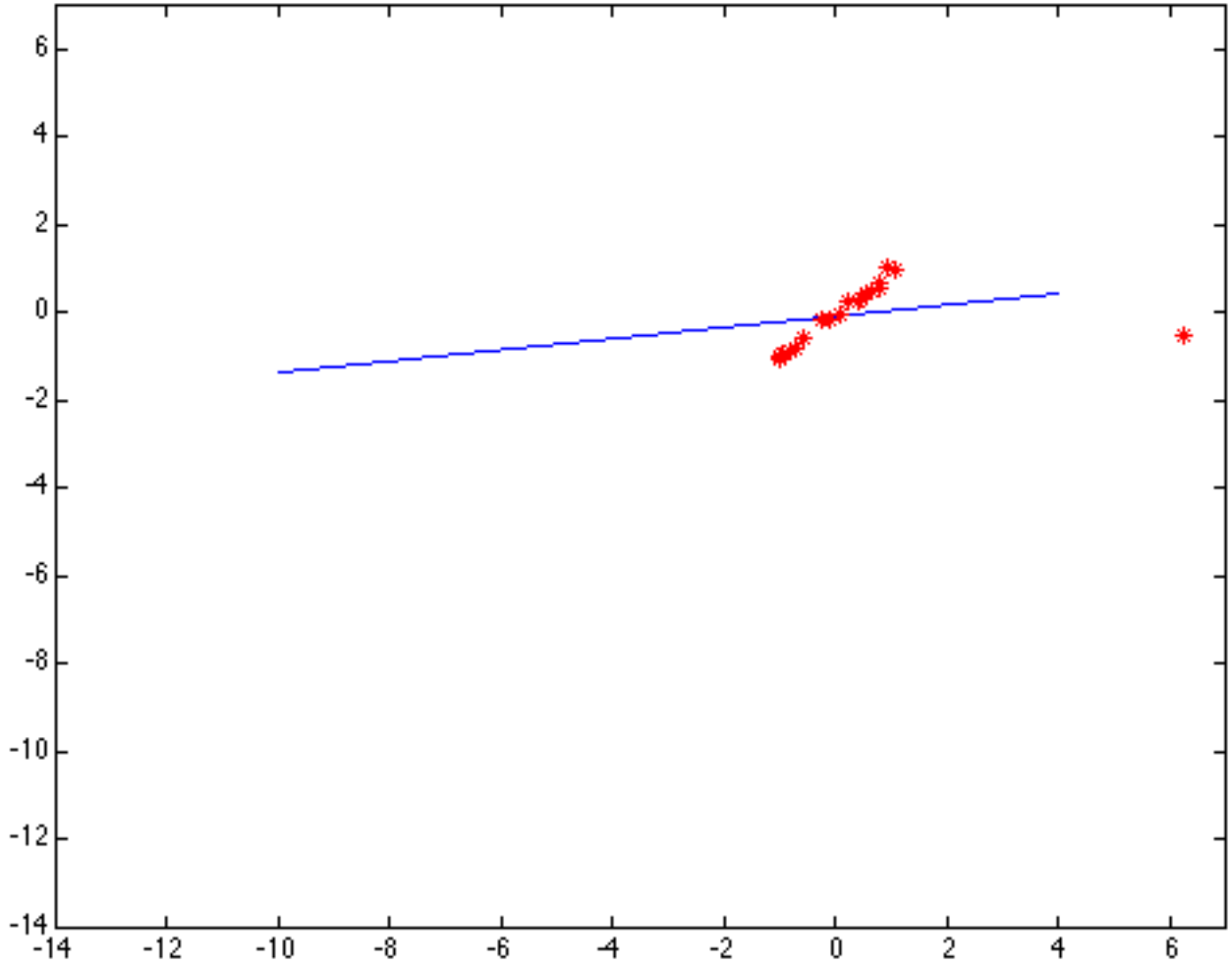
Least squares fit to the red points:



Slide credit : S. Lazebnik

Least squares: Robustness to noise

Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

Slide credit : S. Lazebnik

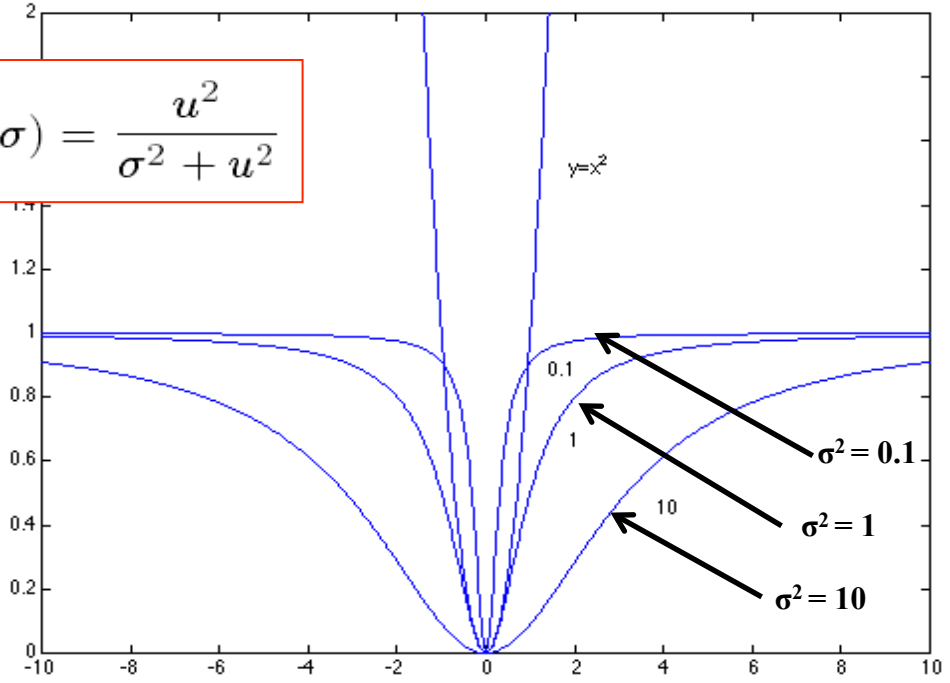
Robust estimators

- General approach: find model parameters θ that minimize

$$\sum_i \rho\left(r_i\left(x_i, \theta\right); \sigma\right)$$

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

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



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

Slide credit : S. Lazebnik

Robust estimators

- General approach: find model parameters θ that minimize

$$\sum_i \rho\left(r_i\left(x_i, \theta\right); \sigma\right)$$

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

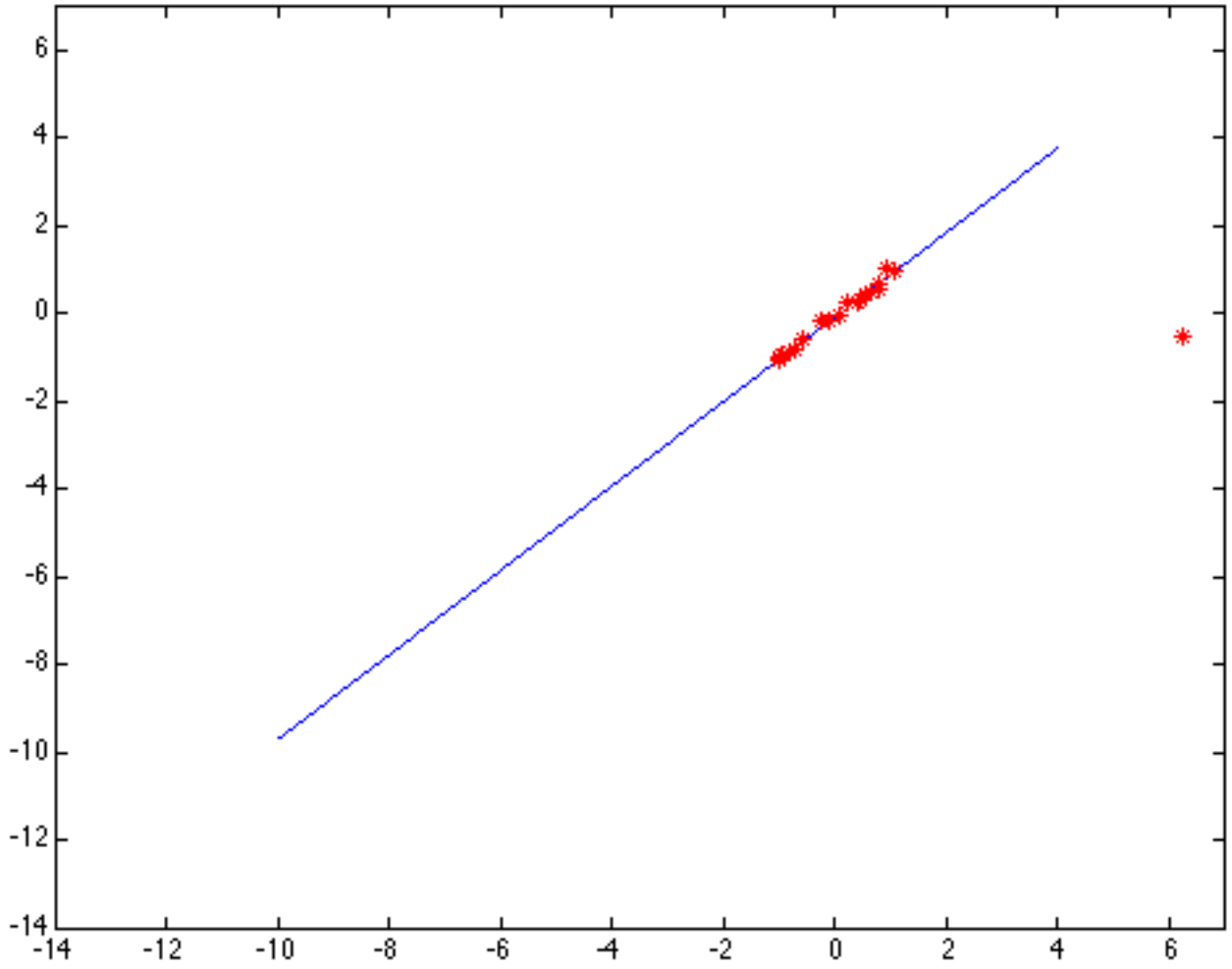
- Robust fitting is a nonlinear optimization problem that must be solved iteratively

- Least squares solution can be used for initialization

- Scale of robust function should be chosen carefully

Slide credit : S. Lazebnik

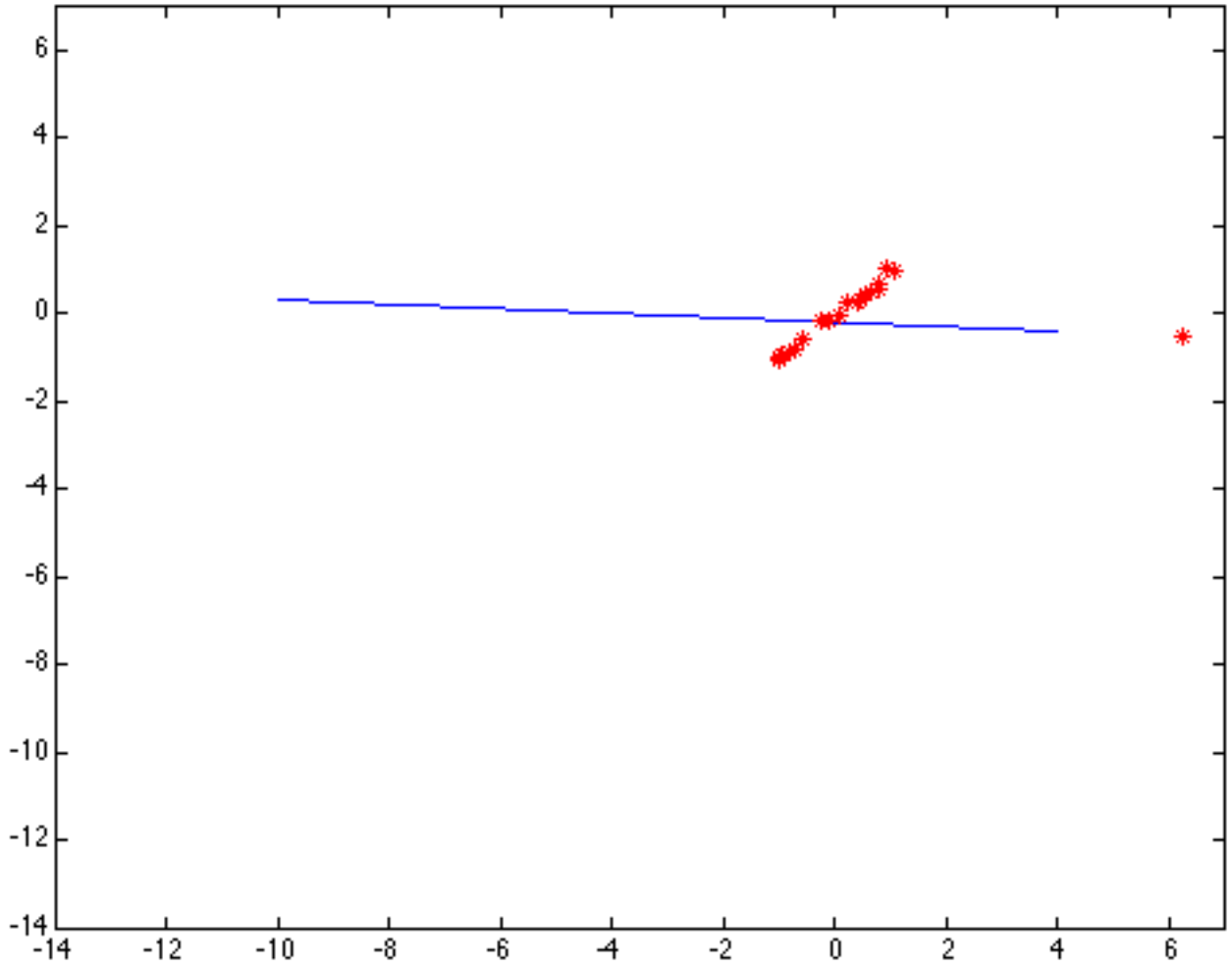
Choosing the scale: just right



The effect of the outlier is minimized

Slide credit : S. Lazebnik

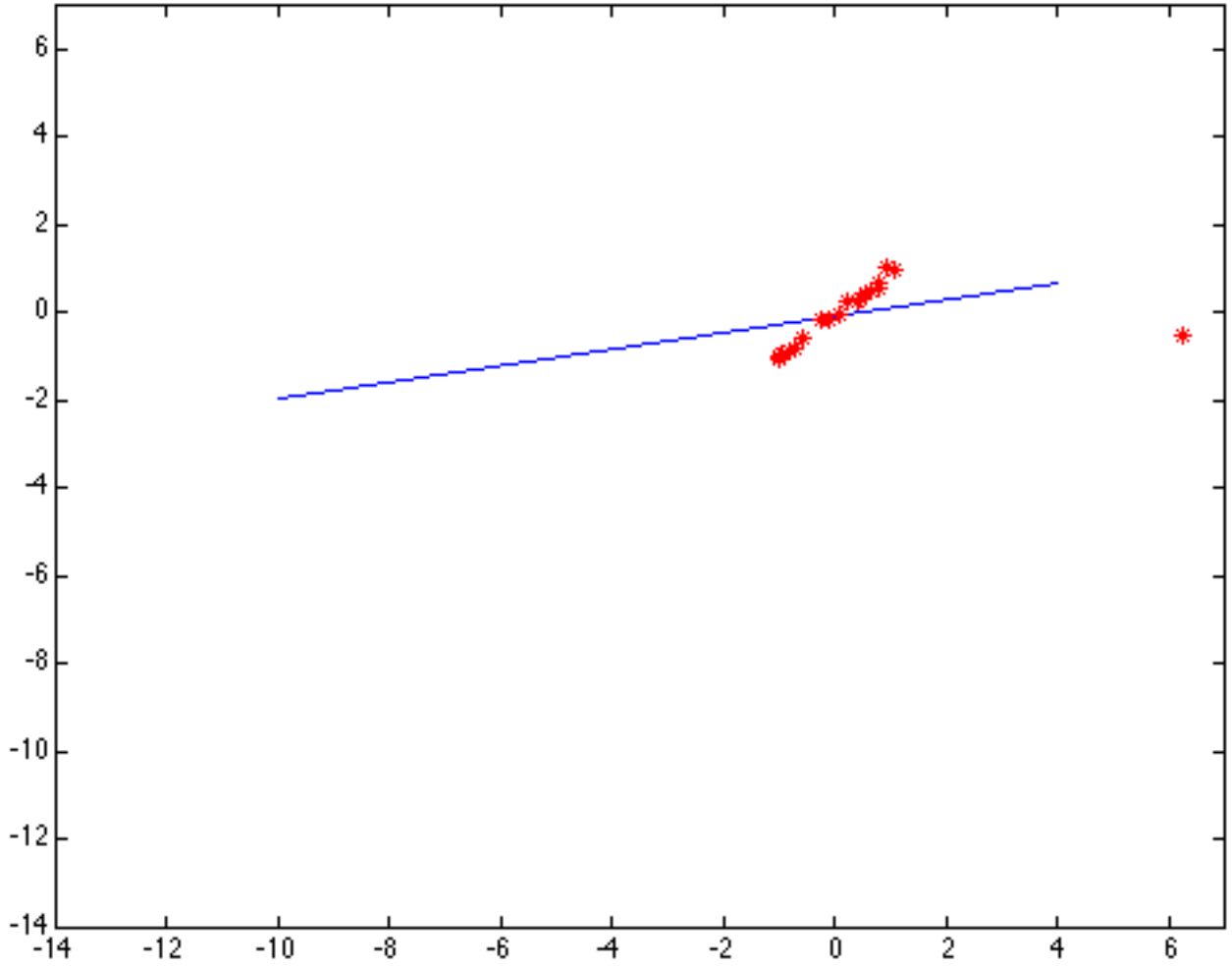
Choosing the scale: too small



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

Slide credit : S. Lazebnik

Choosing the scale: too large



Behaves much the same as least squares

Slide credit : S. Lazebnik

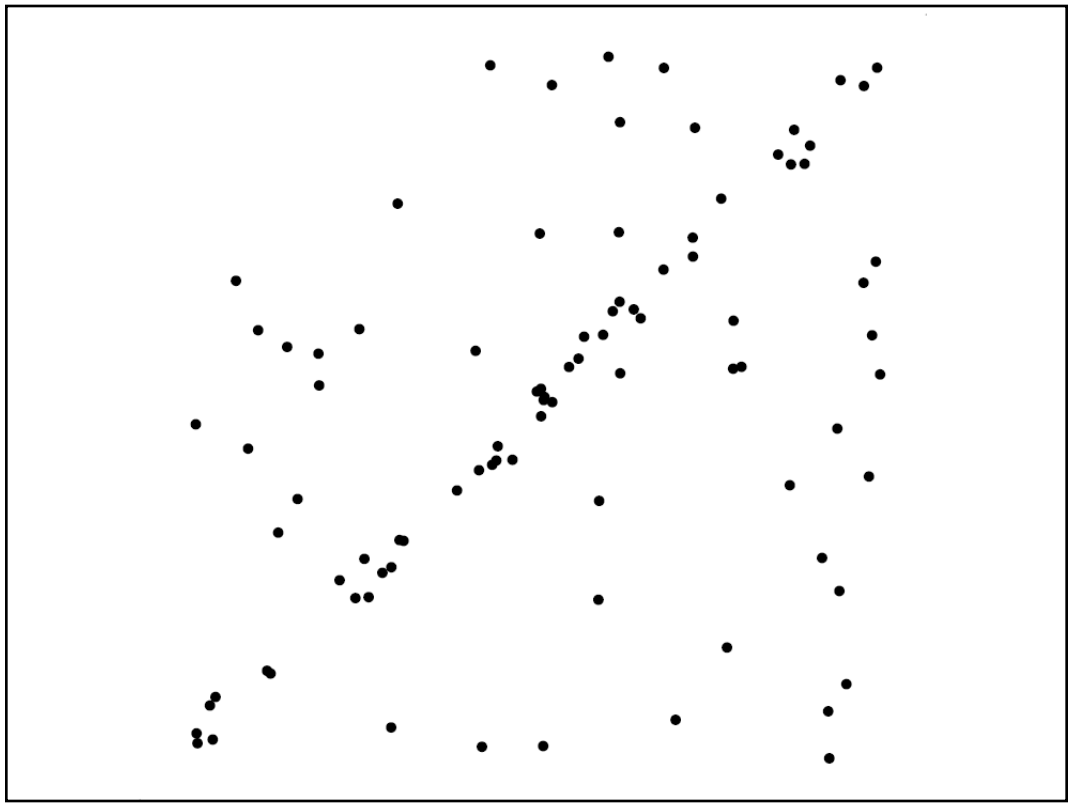
RANSAC

- Robust fitting can deal with a few outliers – what if we have 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. Bolle
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.

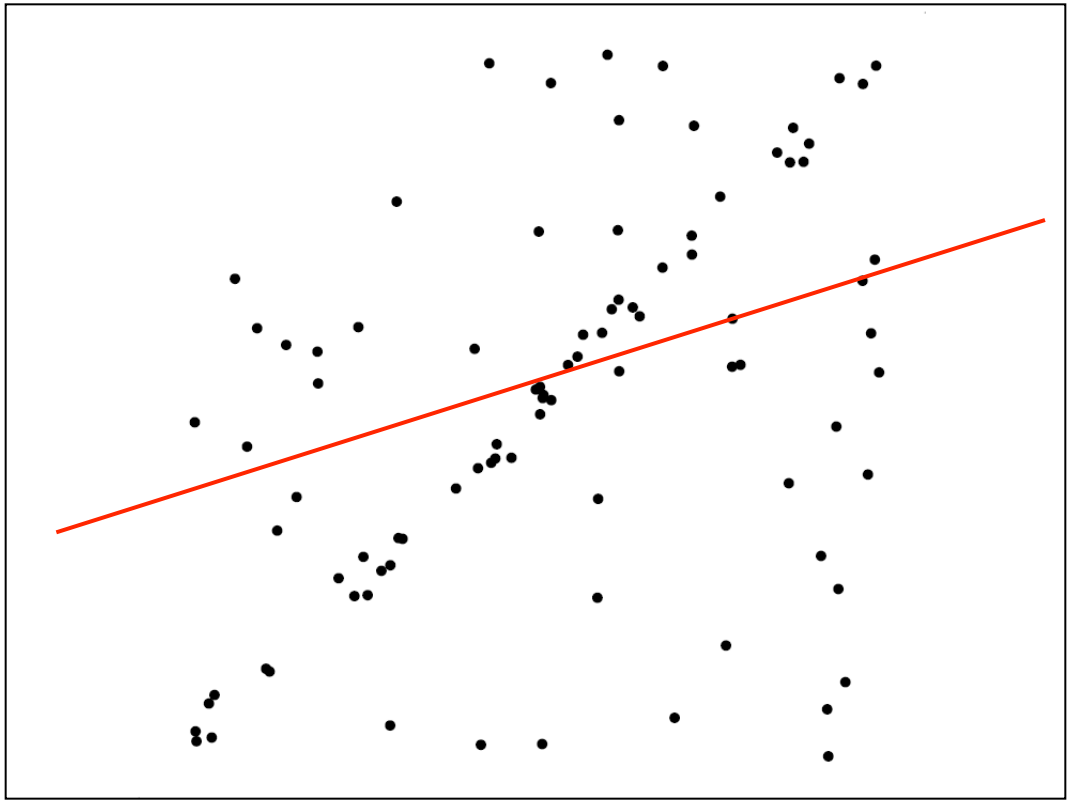
Slide credit : S. Lazebnik

RANSAC for line fitting example



Slide credit: R. Raguram

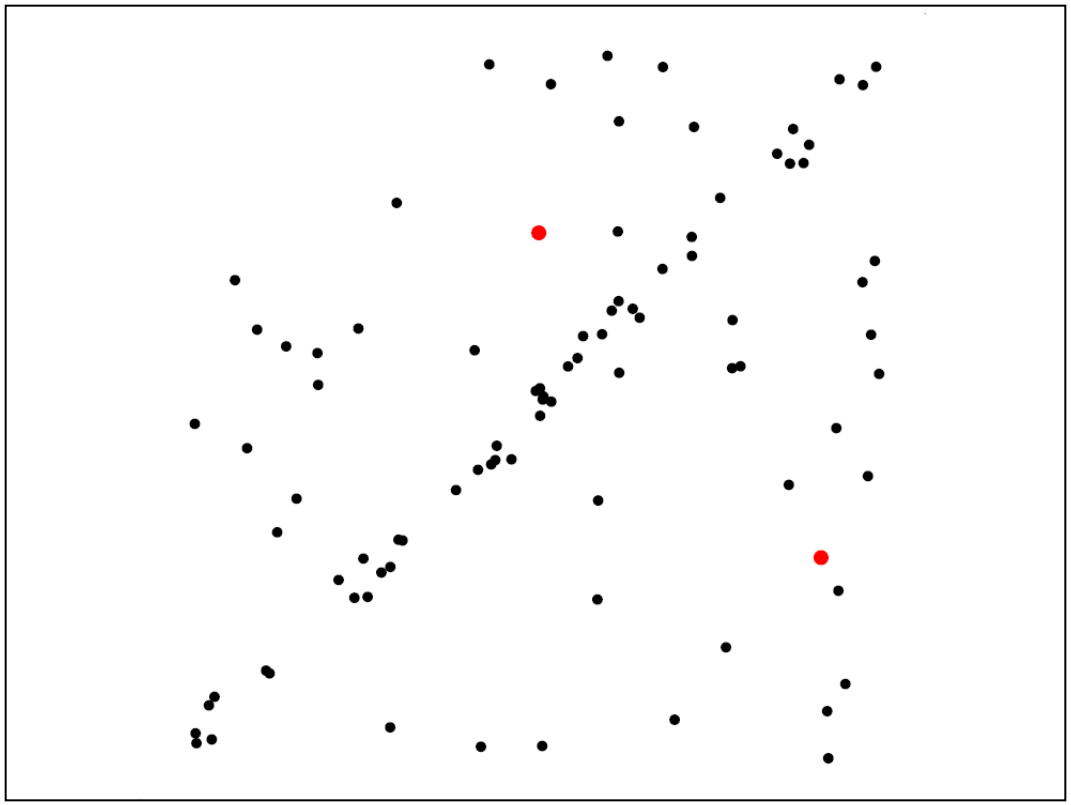
RANSAC for line fitting example



Least-squares fit

Slide credit: R. Raguram

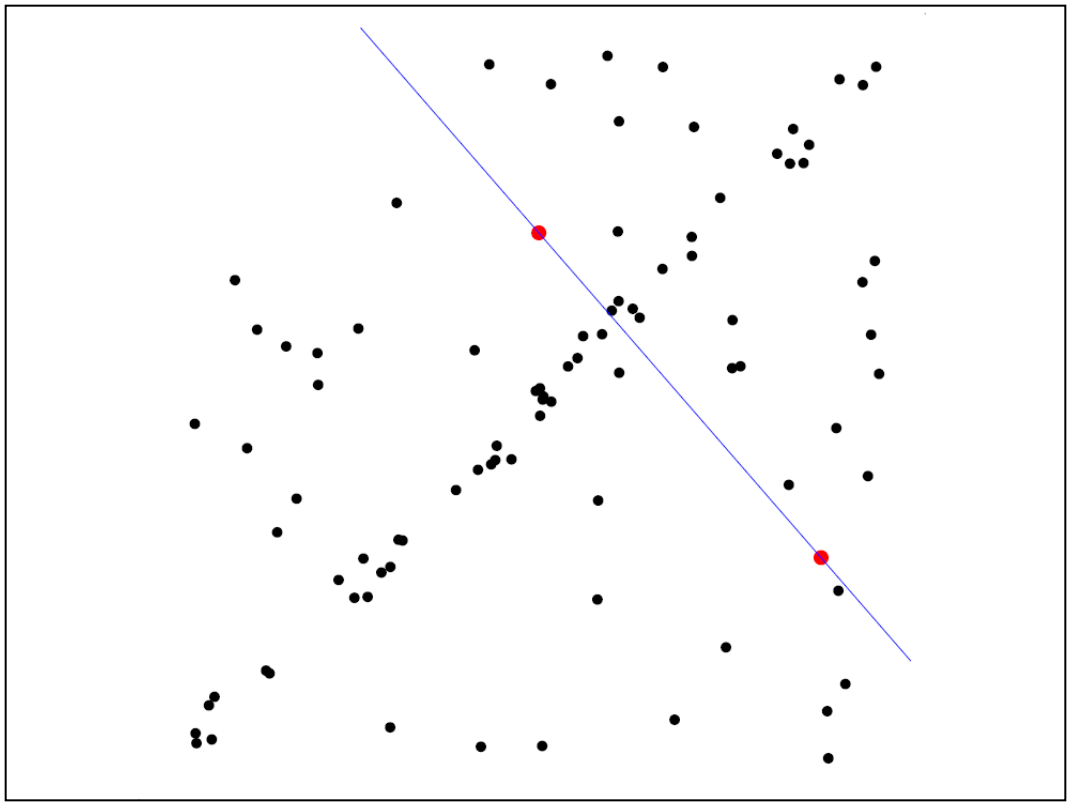
RANSAC for line fitting example



1. Randomly select minimal subset of points

Slide credit: R. Raguram

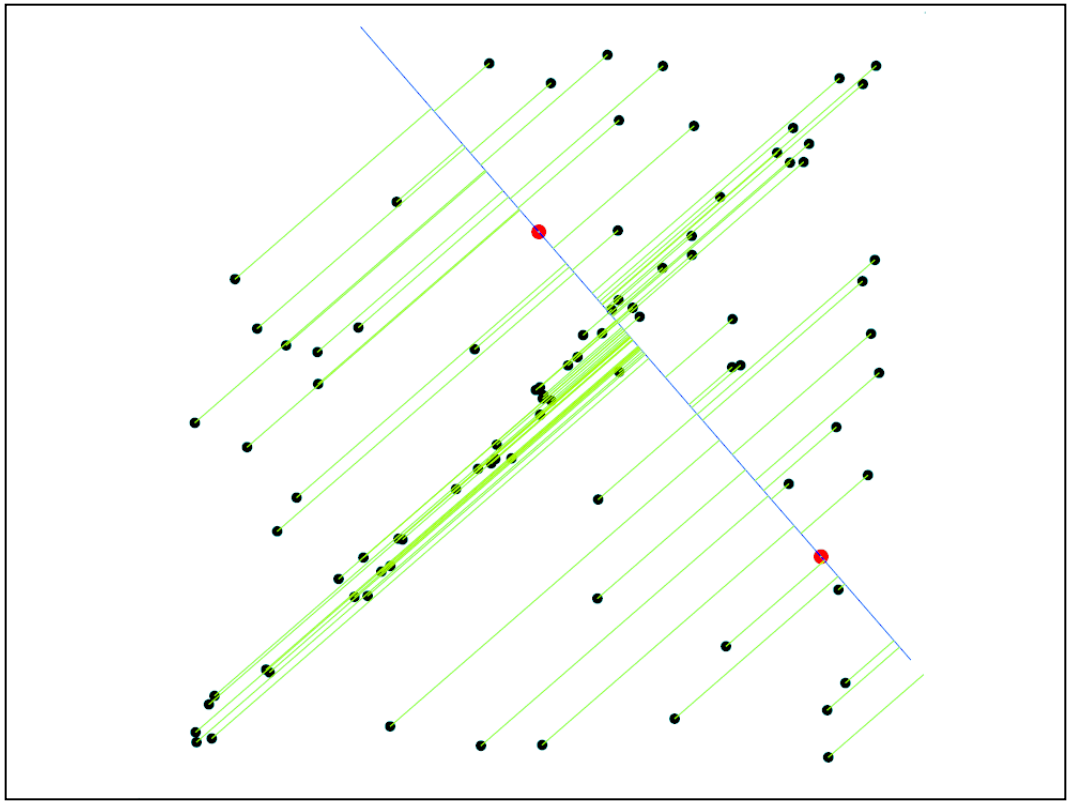
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model

Slide credit: R. Raguram

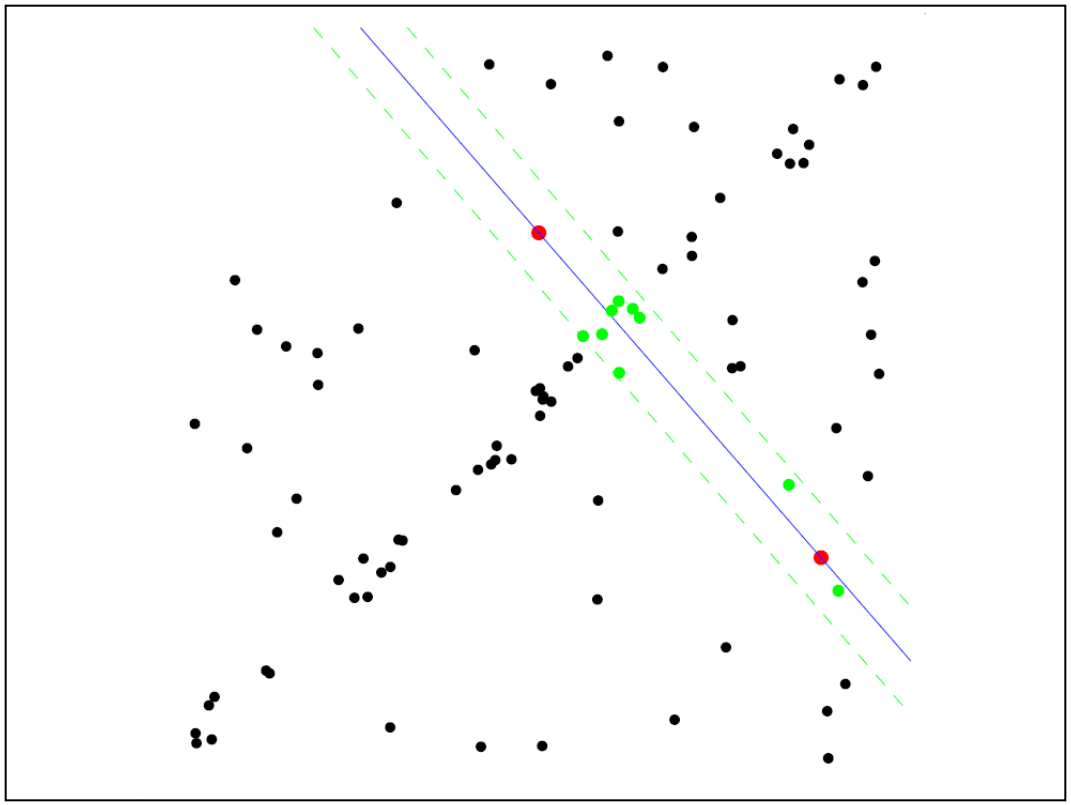
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Slide credit: R. Raguram

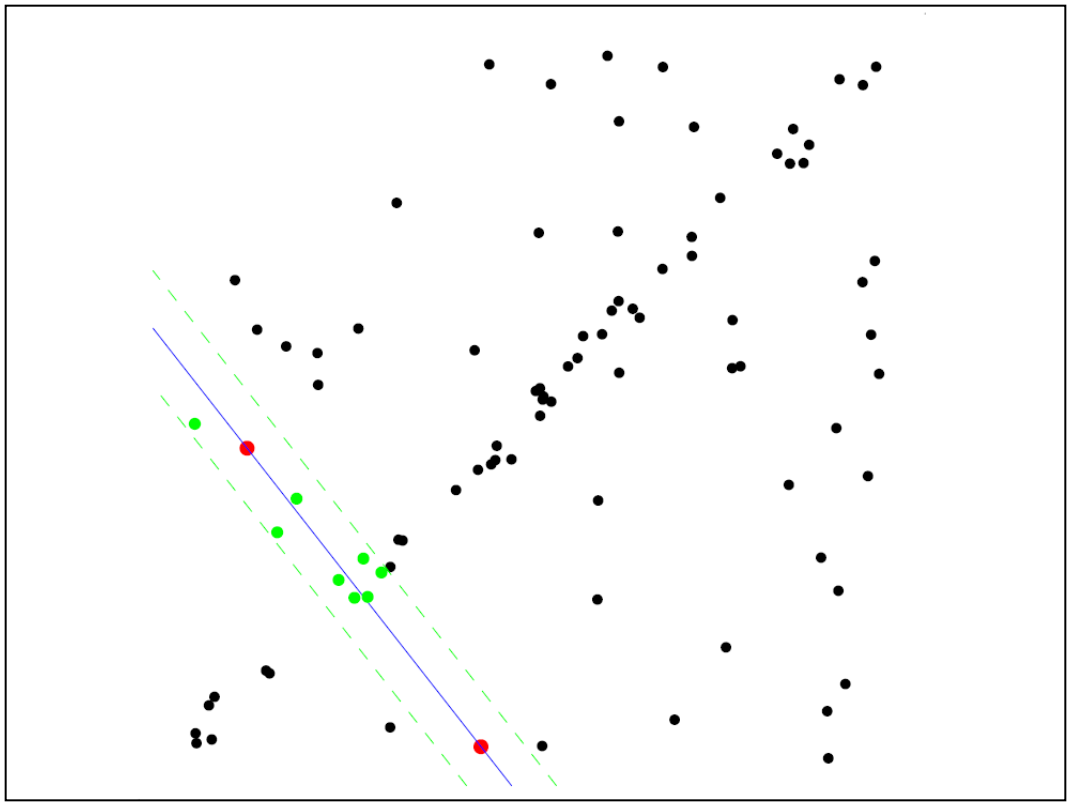
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Slide credit: R. Raguram

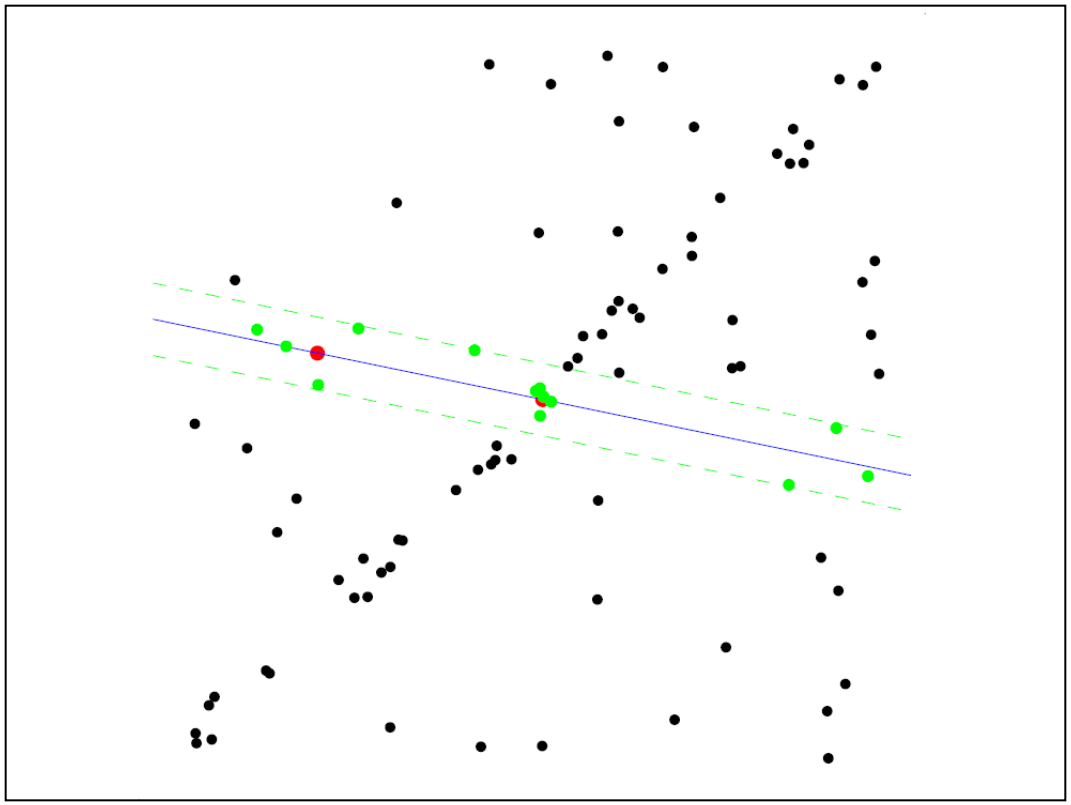
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

RANSAC for line fitting example

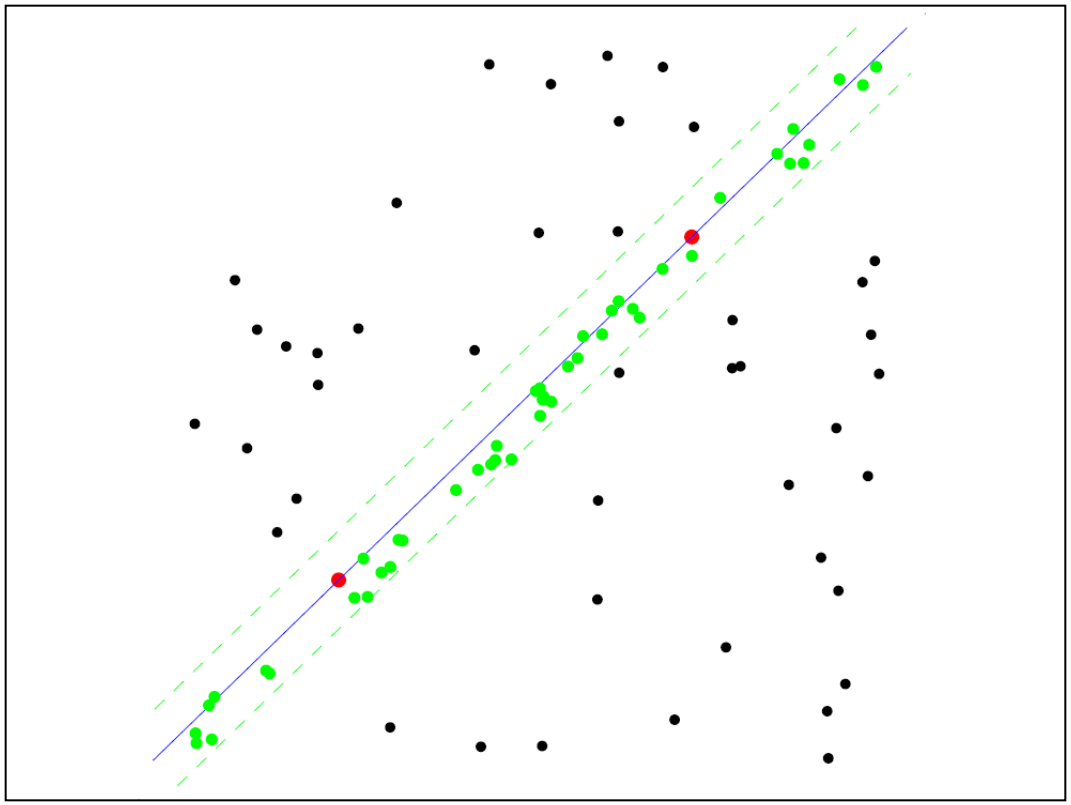


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

RANSAC for line fitting example

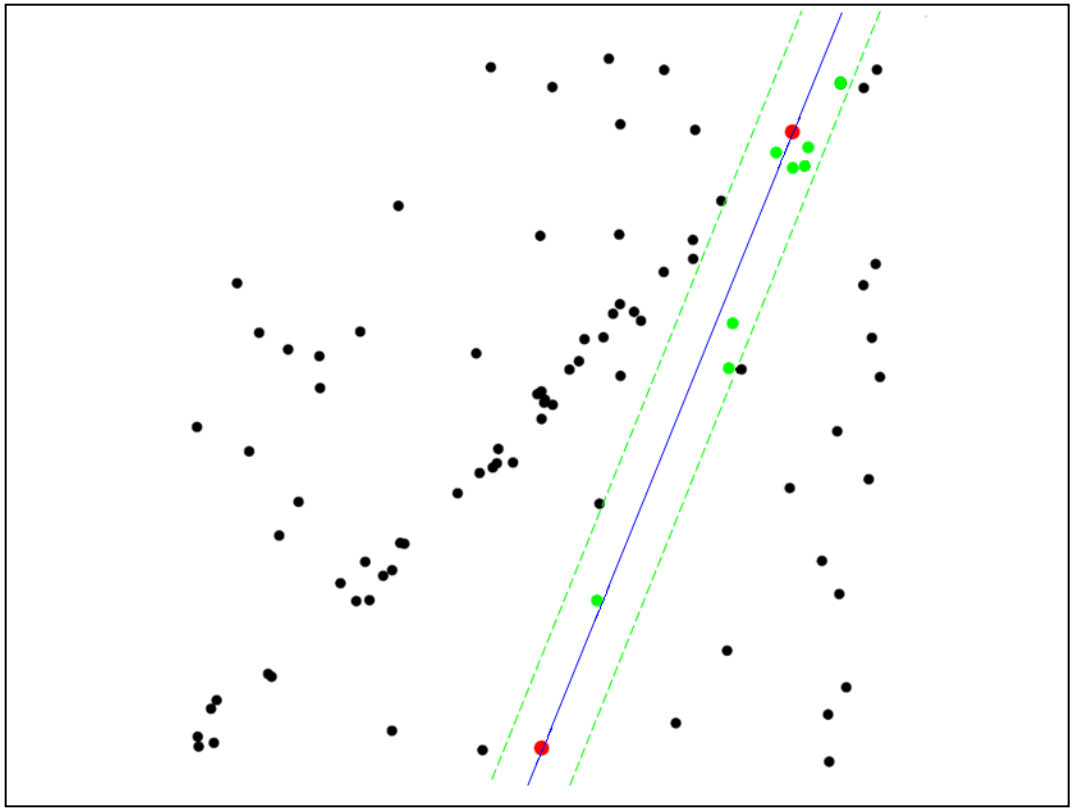
Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

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

Slide credit: M. Pollefeys

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: $t^2=3.84\sigma^2$, $t=1.9\sigma$
- 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*)

Slide credit: M. Pollefeys

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)

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

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

	proportion of outliers e						
s	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

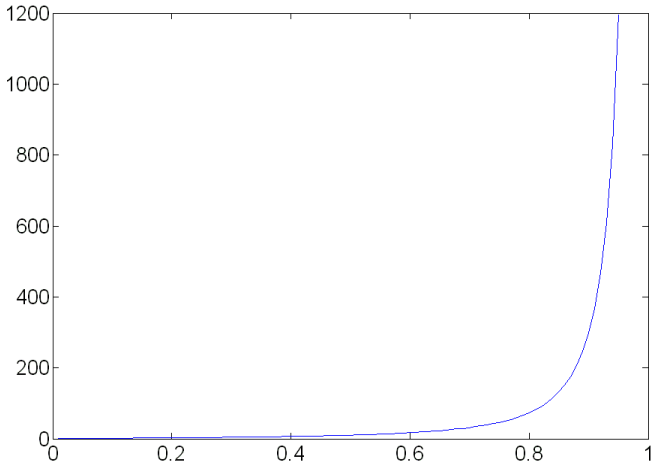
Slide credit: M. Pollefeys

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*)

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

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



Slide credit: M. Pollefeys

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

Slide credit: M. Pollefeys

Adaptively determining the number of samples

- Outlier 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:
  - $N = \infty$, *sample_count* =0
  - While $N >$ *sample_count*
    - Choose a sample and count the number of inliers
    - If inlier ratio is highest of any found so far, set e = 1 – (number of inliers)/(total number of points)
    - Recompute *N* from *e:*

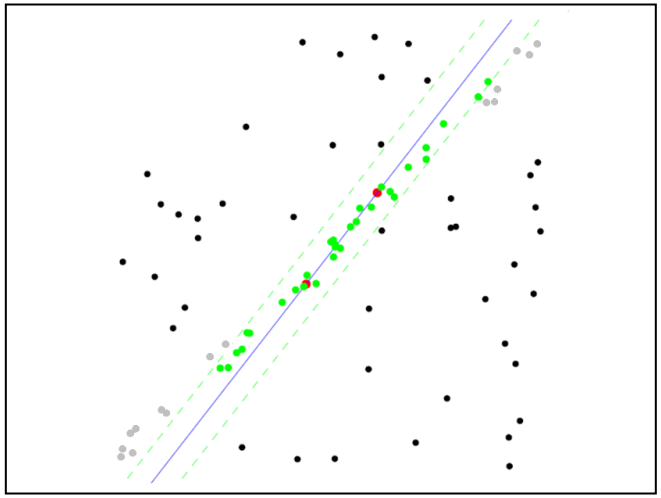
$$N = \log(1 - p) / \log(1 - (1 - e)^s)$$

    - Increment the *sample_count* by 1

Slide credit: M. Pollefeys

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Doesn't work well for low inlier ratios (too many iterations, or can fail completely)
  - Can't always get a good initialization of the model based on the minimum number of samples



Slide credit : S. Lazebnik

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

4. **Object Recognition: high – level vision**

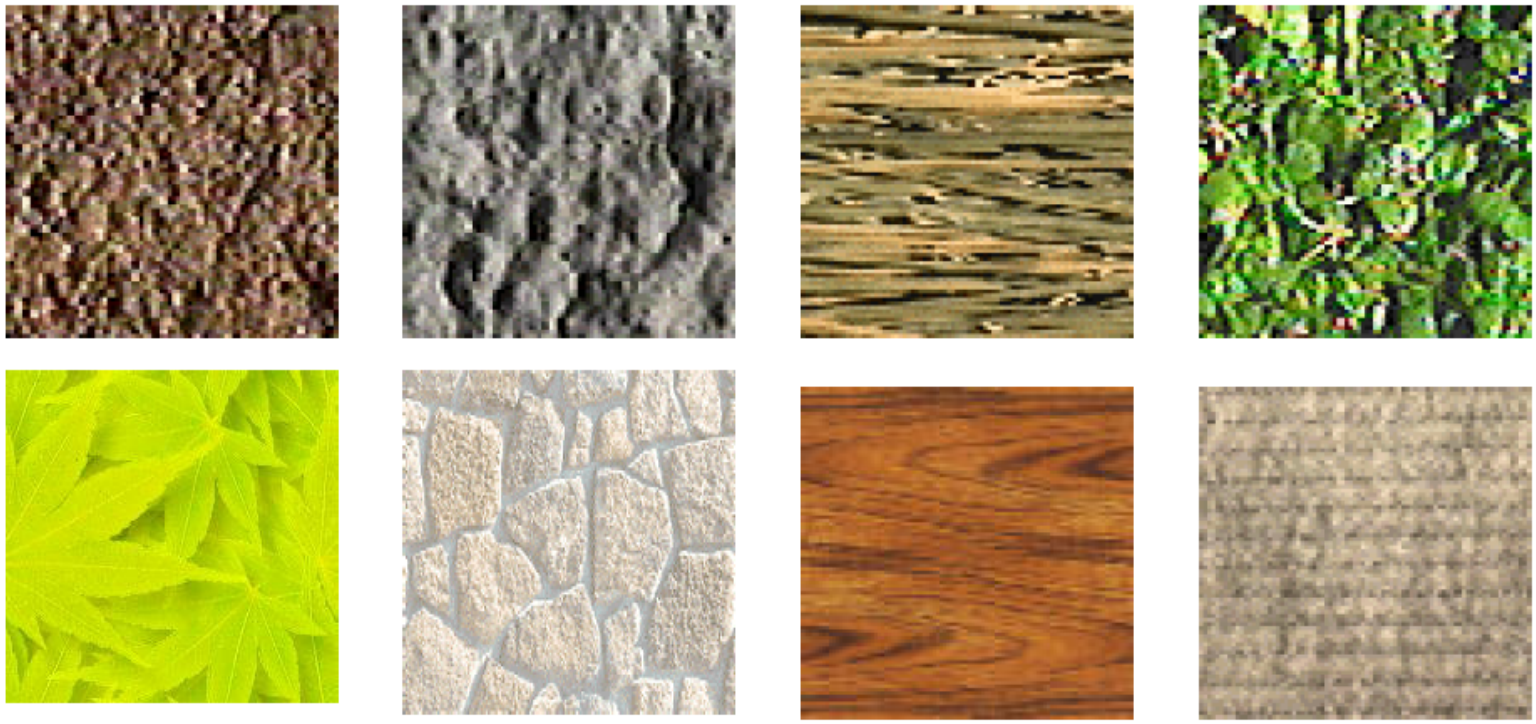
   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

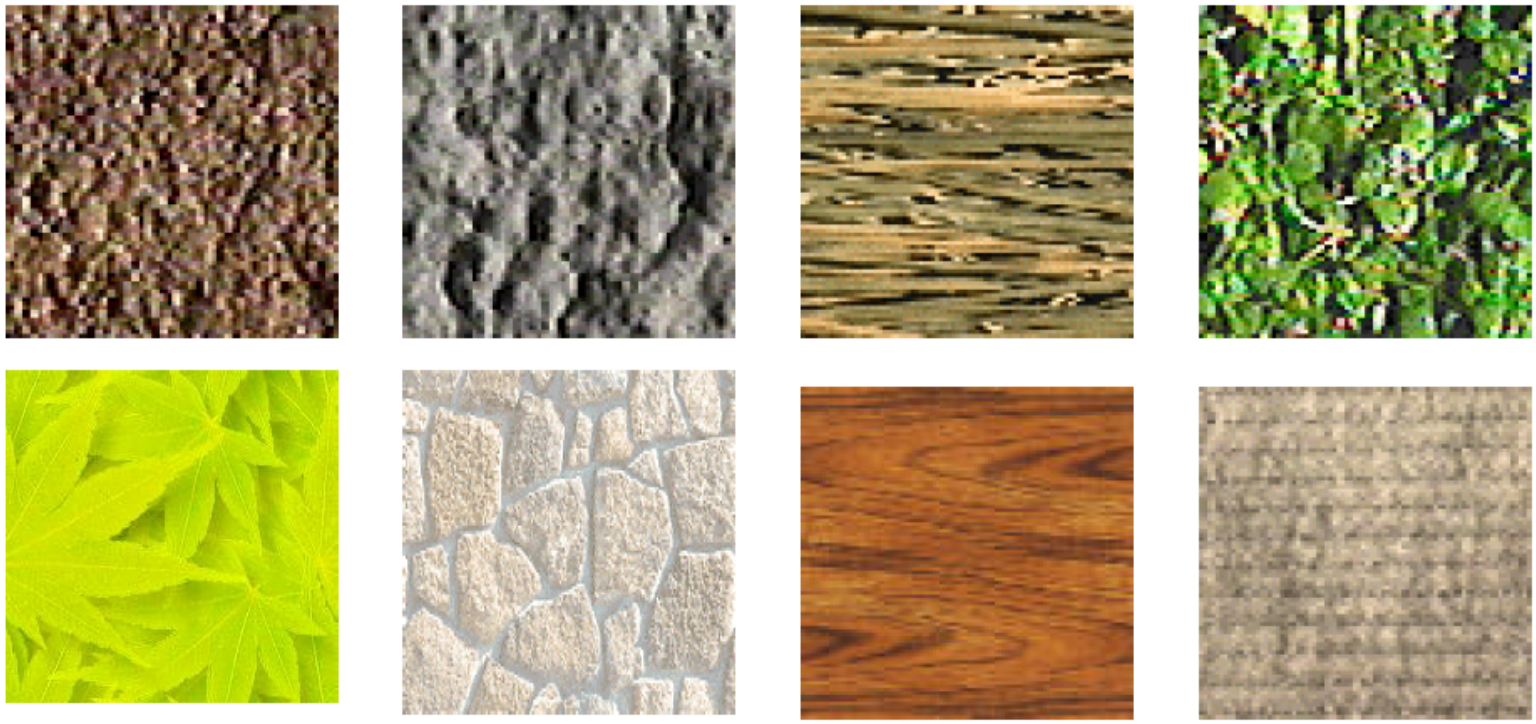
Texture

Texture



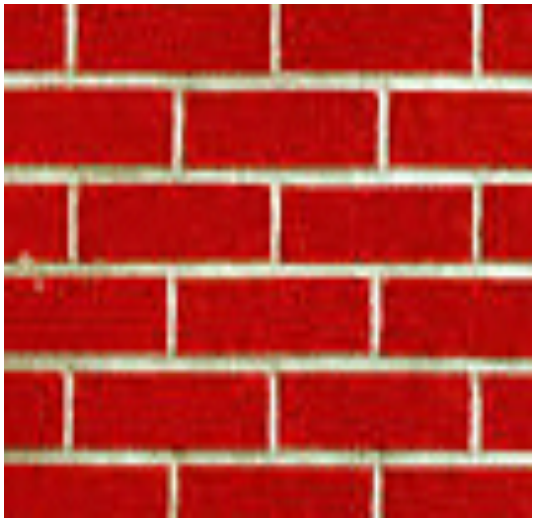
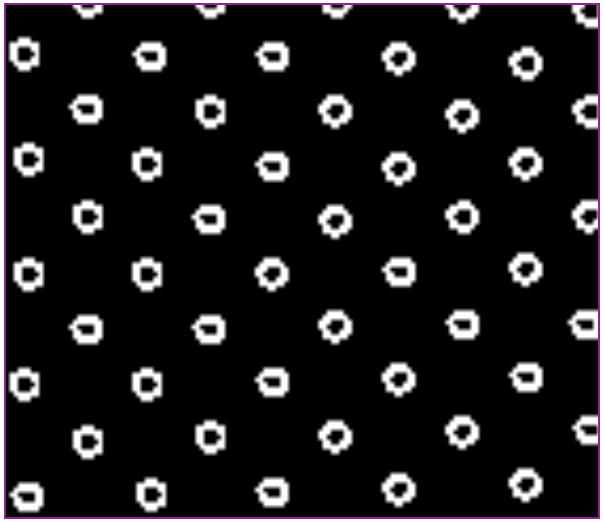
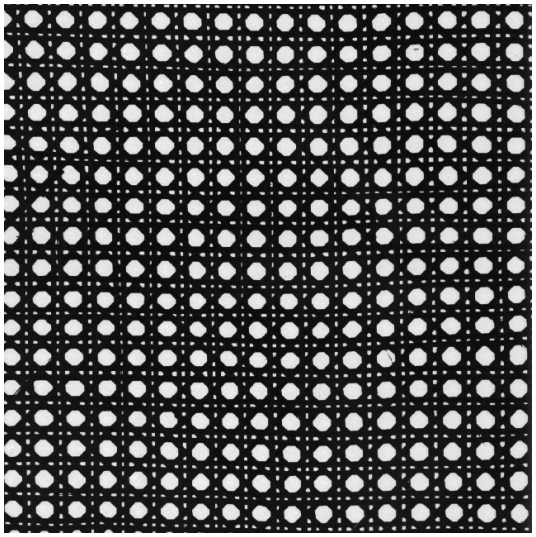
What defines a texture?

Texture



Similar structures of pixels (patterns) that repeat themselves
Similar aspect at a fixe scale
Follow some statistics properties

Includes: more regular patterns

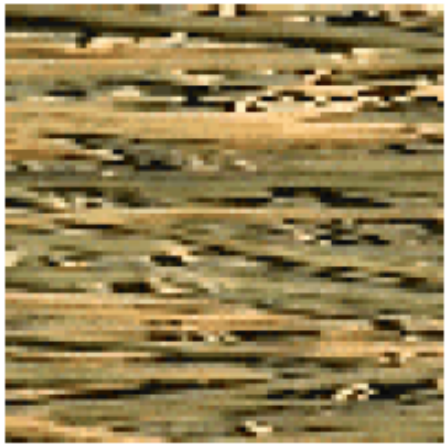
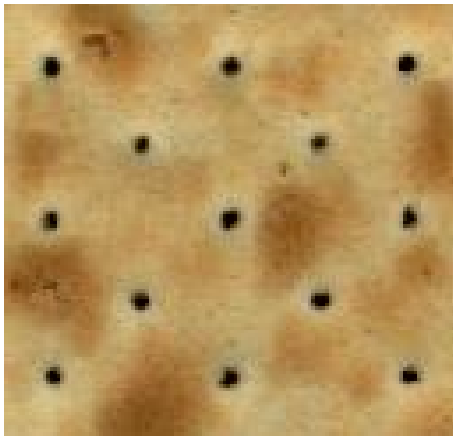




Texture = set of ***texels*** in some regular pattern

Texels = texture elements

Includes: more random patterns



Texture = set of ***texels*** in some regular or repeated pattern

Texels = texture elements

Scale and texture

Leaf - object

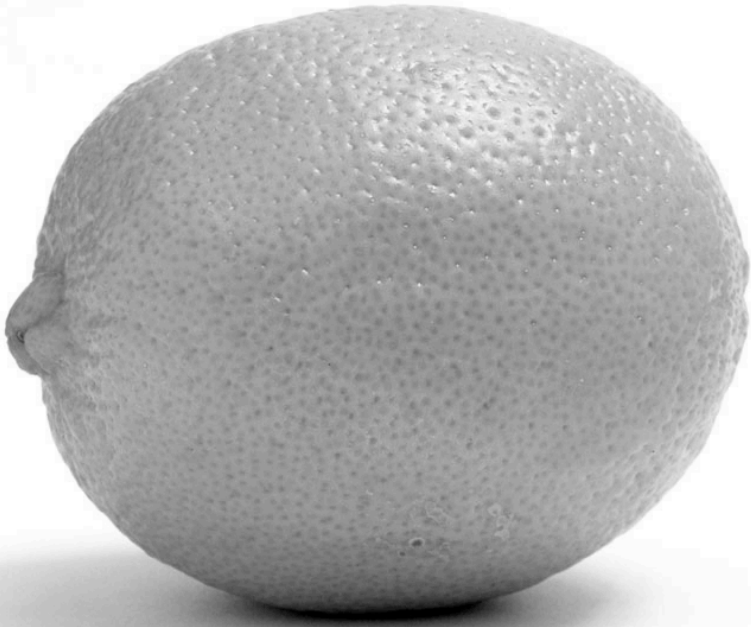
Tree - texture

Why analyze texture?

Importance to perception:

- Often indicative of a material's properties
- Can be important appearance cue, especially if shape is similar across objects
- Aim to distinguish between shape, boundaries, and texture

Slide credit: Kristen Grauman



Slide credit: Kristen Grauman

http://animals.nationalgeographic.com/

Slide credit: Kristen Grauman

Why analyze texture?

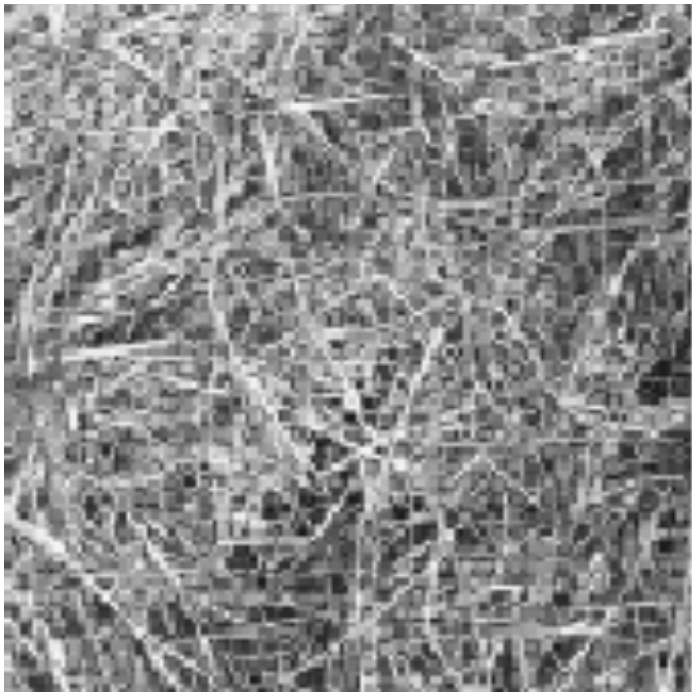
Importance to perception:

- Often indicative of a material's properties
- Can be important appearance cue, especially if shape is similar across objects
- Aim to distinguish between shape, boundaries, and texture

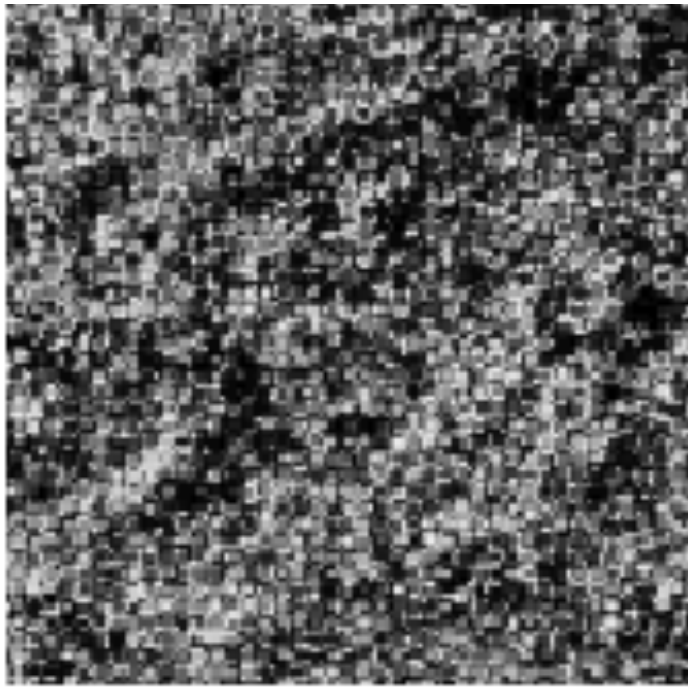
Technically:

- Representation-wise, we want a feature one step above "building blocks" of filters, edges.

Natural Textures



grass



leaves

What/Where are the texels?

The Case for Statistical Texture

- Segmenting out texels is difficult or impossible in real images.
- Numeric quantities or statistics that describe a texture can be computed from the grayscale (or colors) images alone.
- This approach is less intuitive, but is computationally efficient.
- It can be used for both classification and segmentation.

Some Simple Statistical Texture Measures

Edge Density and Direction

- Use an edge detector as the first step in texture analysis.
- The number of edge pixels in a fixed-size region tells us how busy that region is.
- The directions of the edges also help characterize the texture

Edge-based Texture Measures

1. edgeness per unit area

$$F_{edgeness} = |\{ p \mid gradient\_magnitude(p) \geq threshold \}| / N$$

where N is the size of the unit area

2. edge magnitude and direction histograms

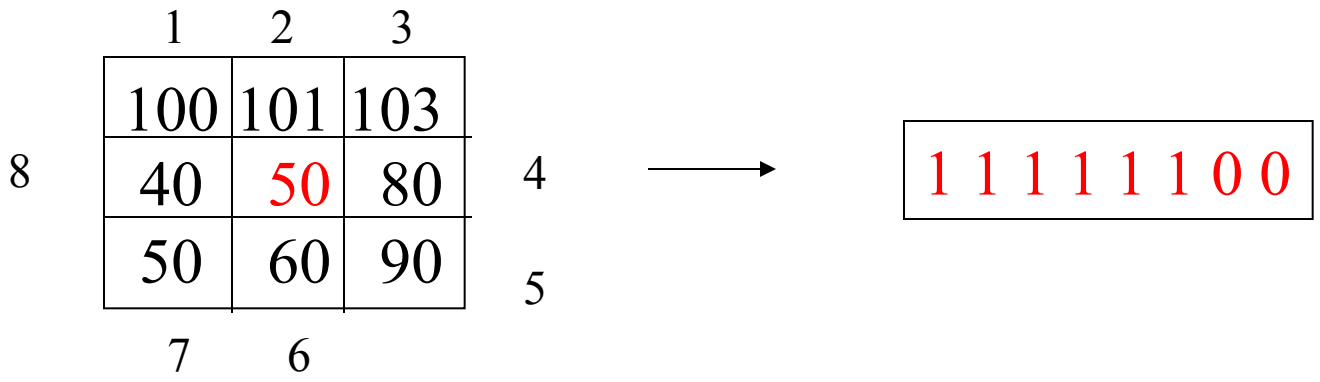
$$F_{magdir} = (H_{magnitude}, H_{direction})$$

where these are the normalized histograms of gradient magnitudes and gradient directions, respectively.

Local Binary Pattern Measure

- For each pixel p, create an 8-bit number $b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8$, where $b_i = 0$ if neighbor *i* has value less than or equal to p's value and 1 otherwise.

- Represent the texture in the image (or a region) by the histogram of these numbers.



Co-occurrence Matrix Features

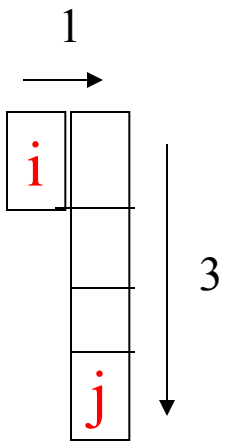
A co-occurrence matrix is a 2D array C in which

- Both the rows and columns represent a set of possible image values.
- $C_d(i,j)$ indicates how many times value i co-occurs with value j in a particular spatial relationship d.
- The spatial relationship is specified by a vector d = (dr,dc).

Co-occurrence Example

1	1	0	0
1	1	0	0
0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2

grayscale image



d = (3,1)

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

C_d

co-occurrence matrix

From C_d we can compute N_d, the normalized co-occurrence matrix, where each value is divided by the sum of all the values.

Co-occurrence Features

- extract numerical features from the co-occurrence matrix which can be used for representing and comparing textures

$$
\begin{aligned}
Energy &= \sum_i \sum_j N_d^2(i,j) \\
Entropy &= -\sum_i \sum_j N_d(i,j) log_2 N_d(i,j) \\
Contrast &= \sum_i \sum_j (i-j)^2 N_d(i,j) \\
Homogeneity &= \sum_i \sum_j \frac{N_d(i,j)}{1+\mid i-j \mid} \\
Correlation &= \frac{\sum_i \sum_j (i-\mu_i)(j-\mu_j) N_d(i,j)}{\sigma_i \sigma_j}
\end{aligned}
\tag{7.11}
$$

where μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of the row and column

But how do you choose d?

- This is actually a critical question with **all** the statistical texture methods.
- Are the "texels" tiny, medium, large, all three …?
- Not really a solved problem.

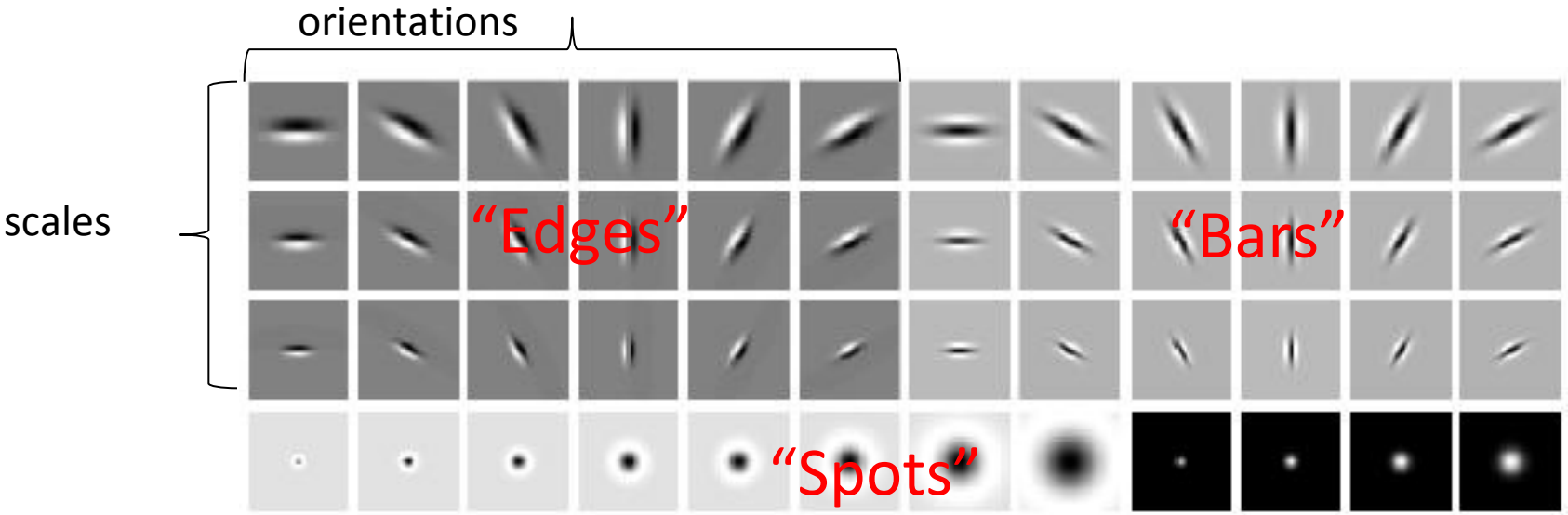
Alternative approach

Signal-processing-based algorithms use texture filters applied to the image to create filtered images from which texture features are computed.

Textures are made up of repeated local patterns, so:

- Find the patterns
  - Use filters that look like patterns (spots, bars, raw patches…)
  - Consider magnitude of response
- Describe their statistics within each local window, e.g.,
  - Mean, standard deviation
  - Histogram
  - Histogram of "prototypical" feature occurrences

Filter banks

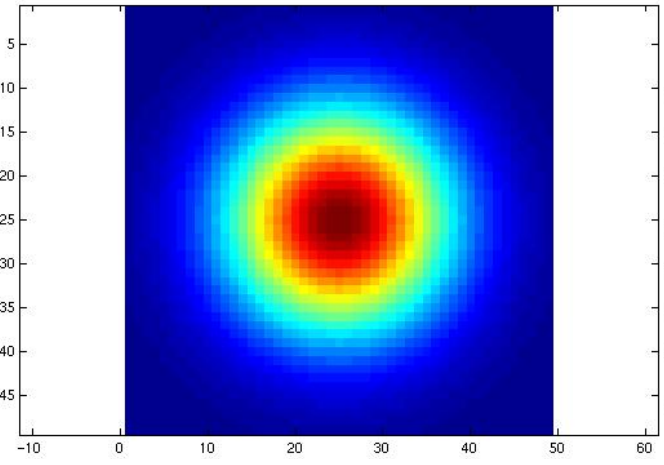


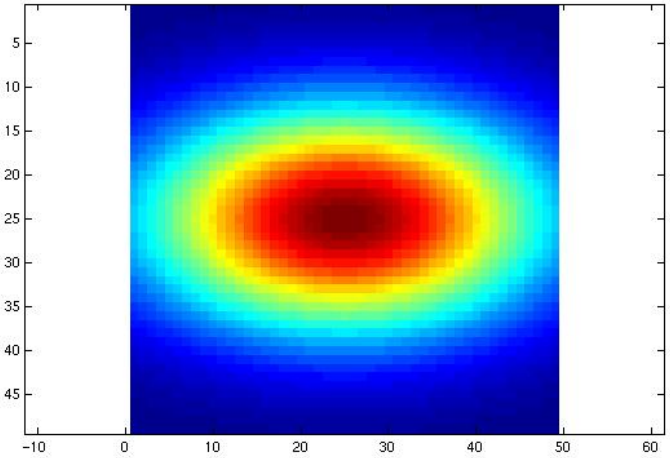
- What filters to put in the bank?
  - Typically we want a combination of scales and orientations, different types of patterns.

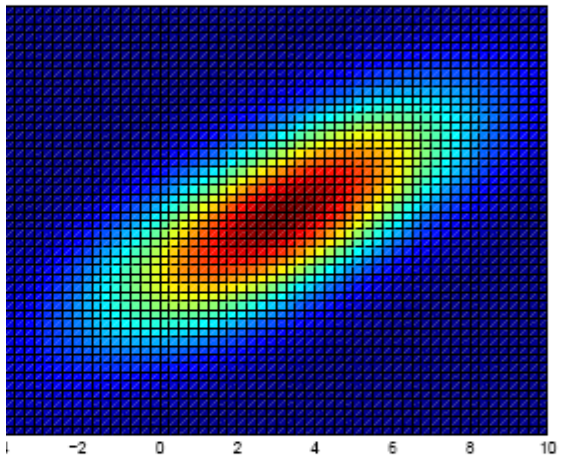
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Multivariate Gaussian

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$



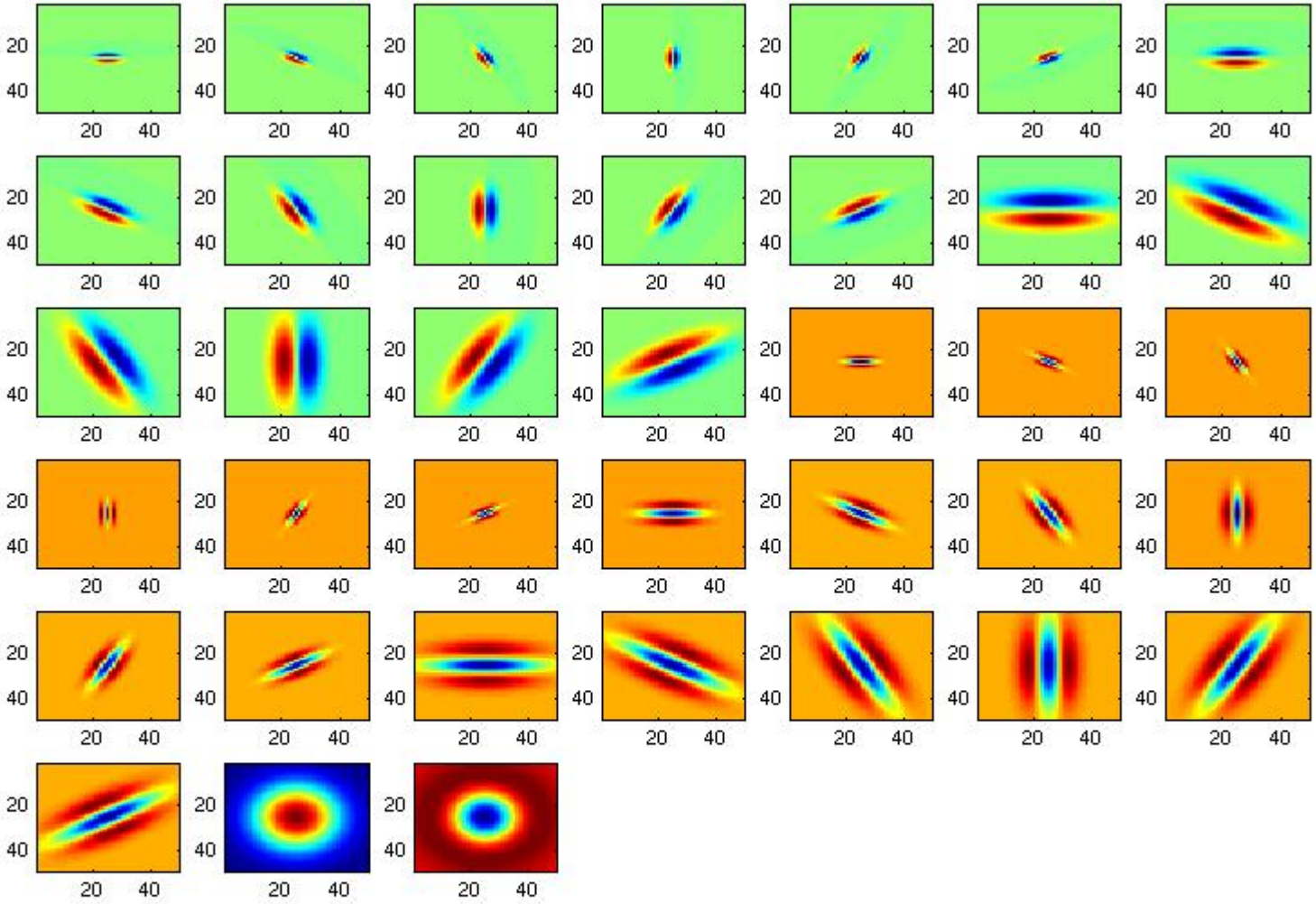




$$\Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 16 & 0 \\ 0 & 9 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$

Slide credit: Kristen Grauman

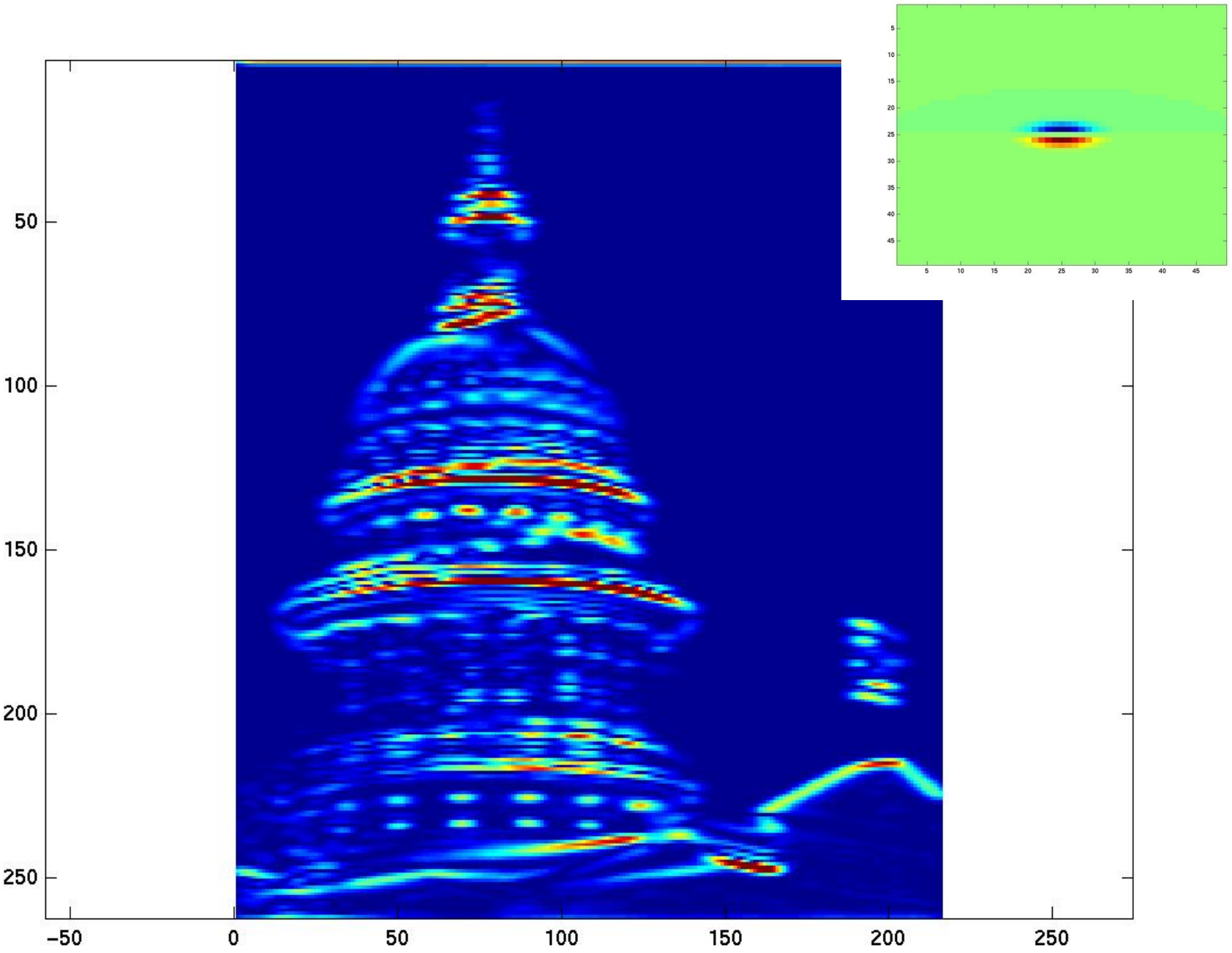
Filter bank



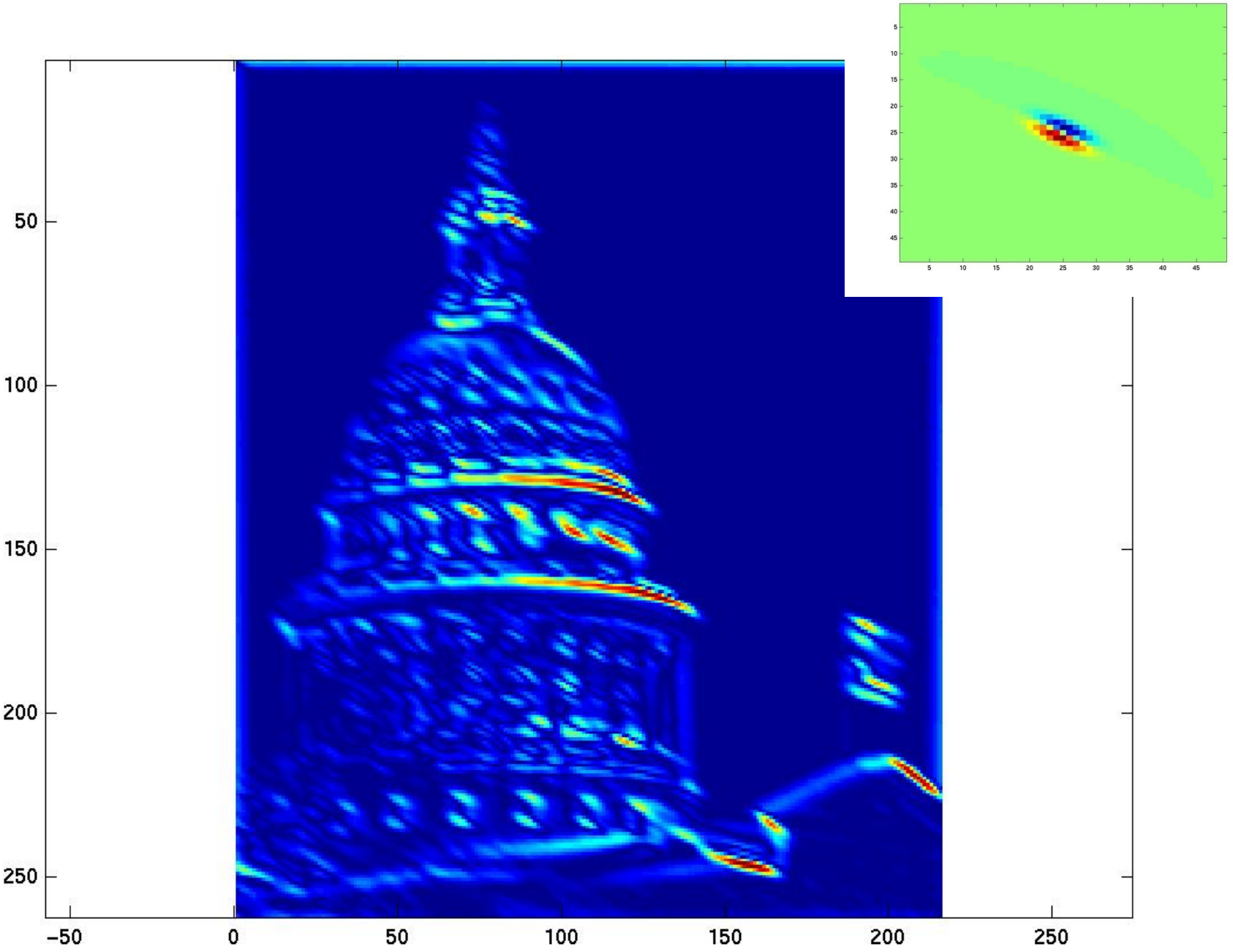
Slide credit: Kristen Grauman

Image from http://www.texasexplorer.com/austincap2.jpg

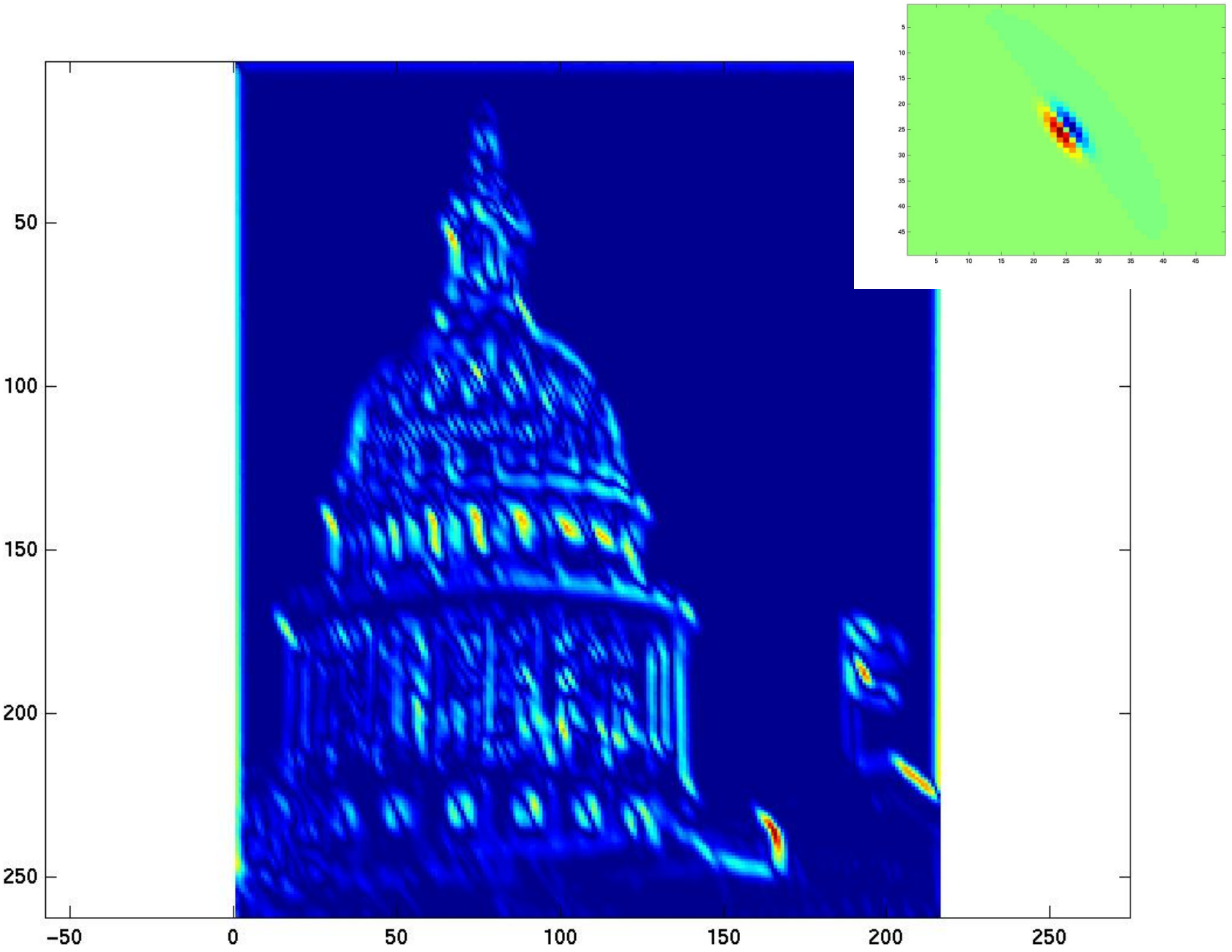
Slide credit: Kristen Grauman



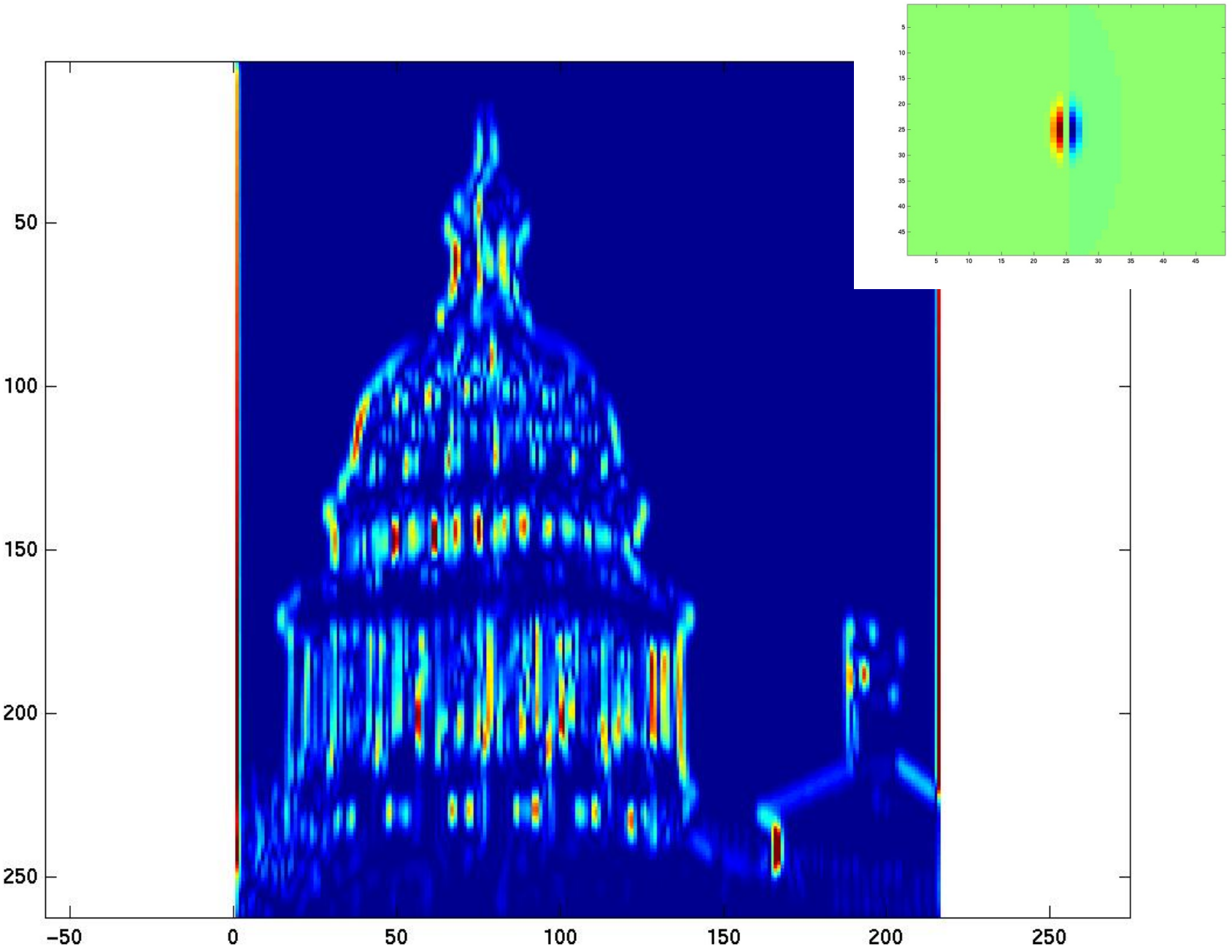
Slide credit: Kristen Grauman



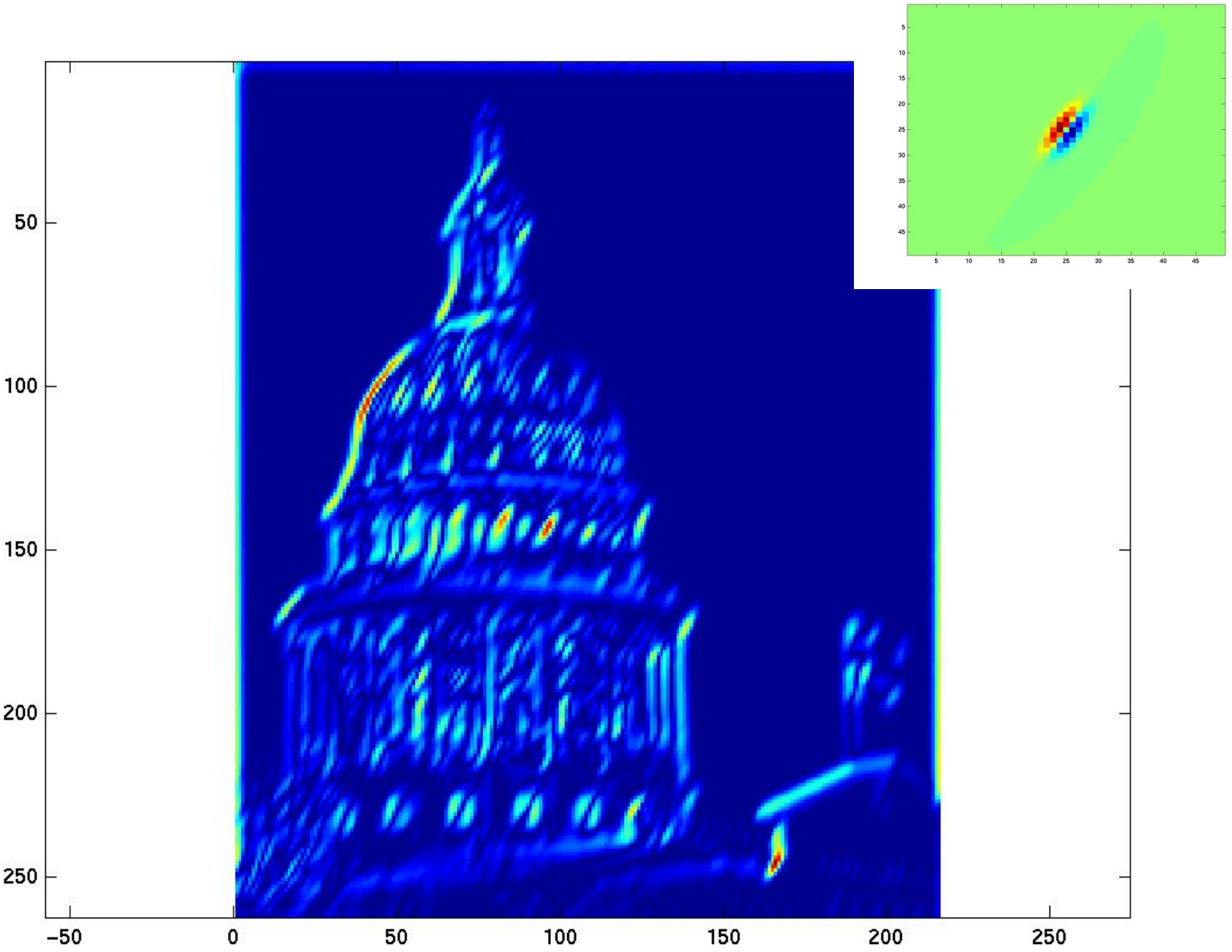
Slide credit: Kristen Grauman



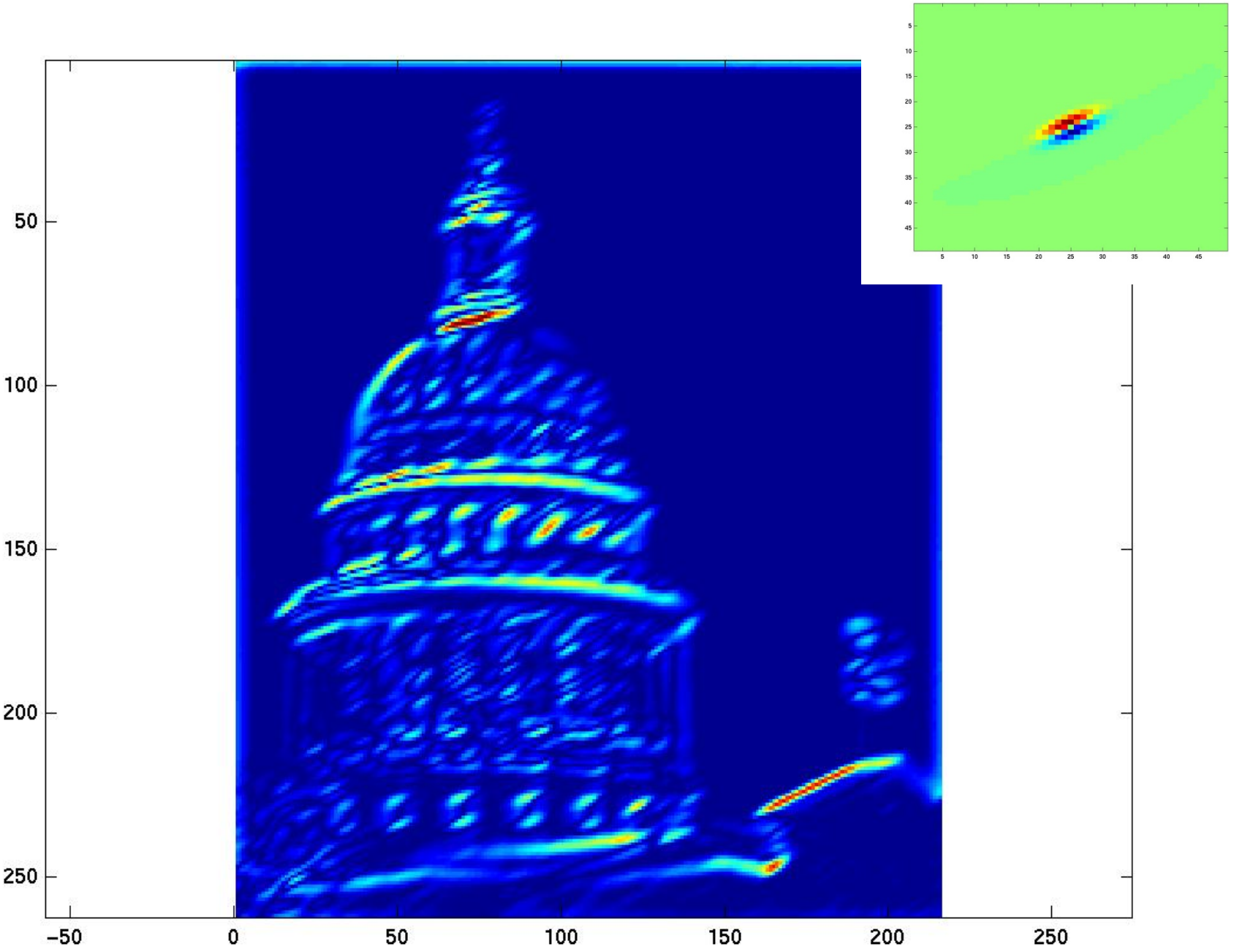
Slide credit: Kristen Grauman



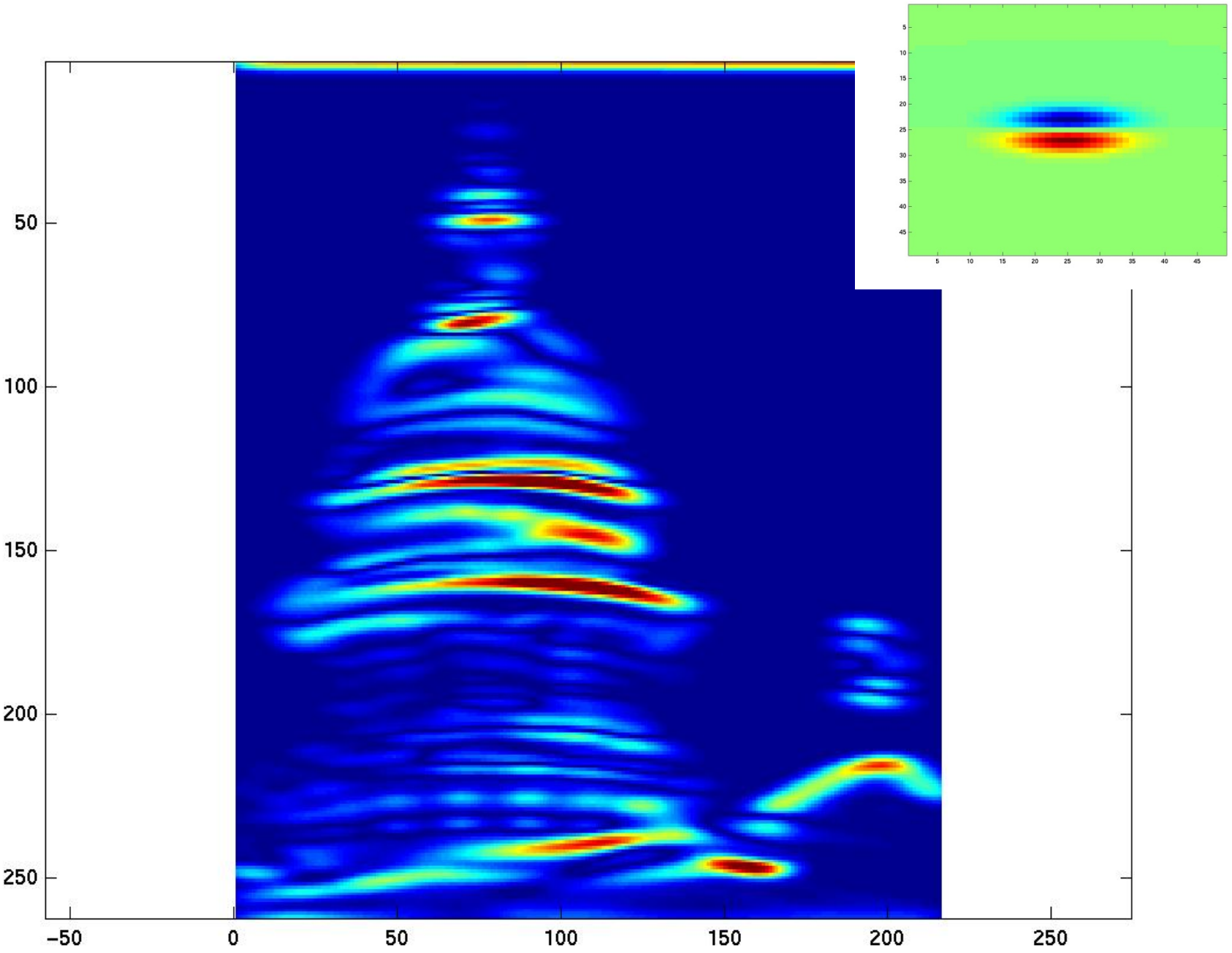
Slide credit: Kristen Grauman



Slide credit: Kristen Grauman



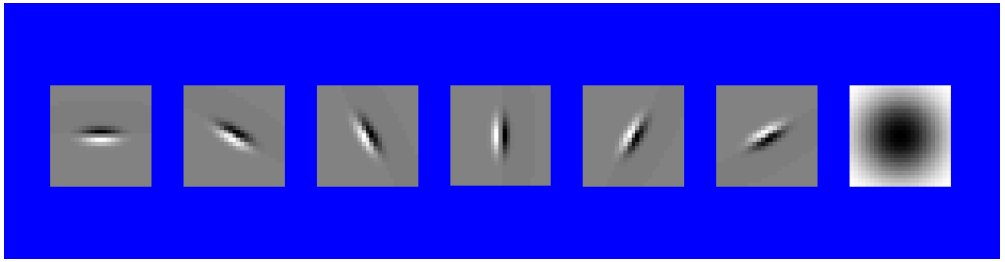
Slide credit: Kristen Grauman



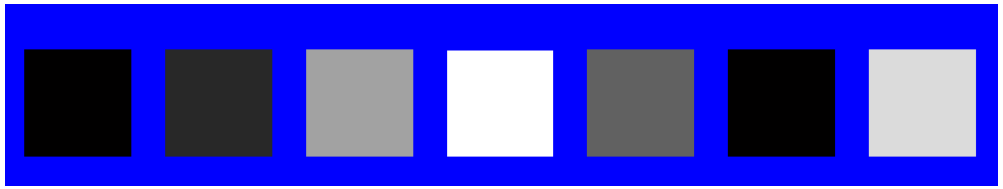
Slide credit: Kristen Grauman

You try: Can you match the texture to the response?

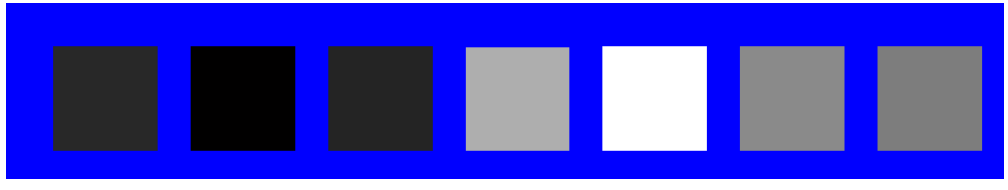
Filters



1



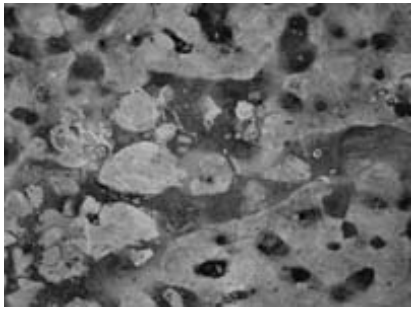
2



3

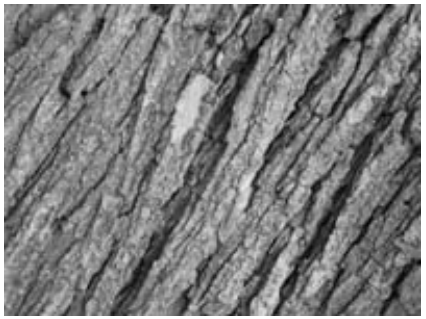
Mean abs responses

A



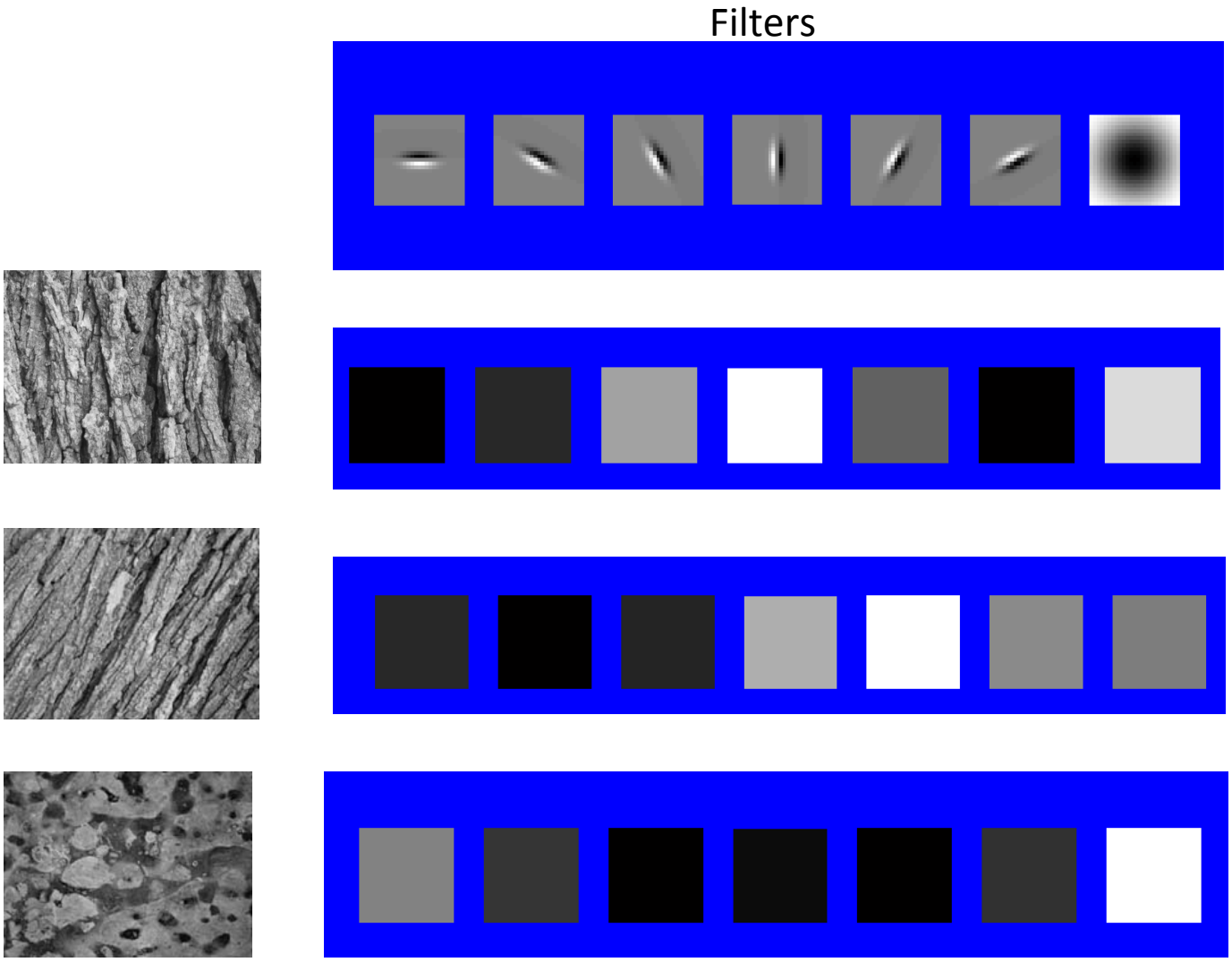
B

C

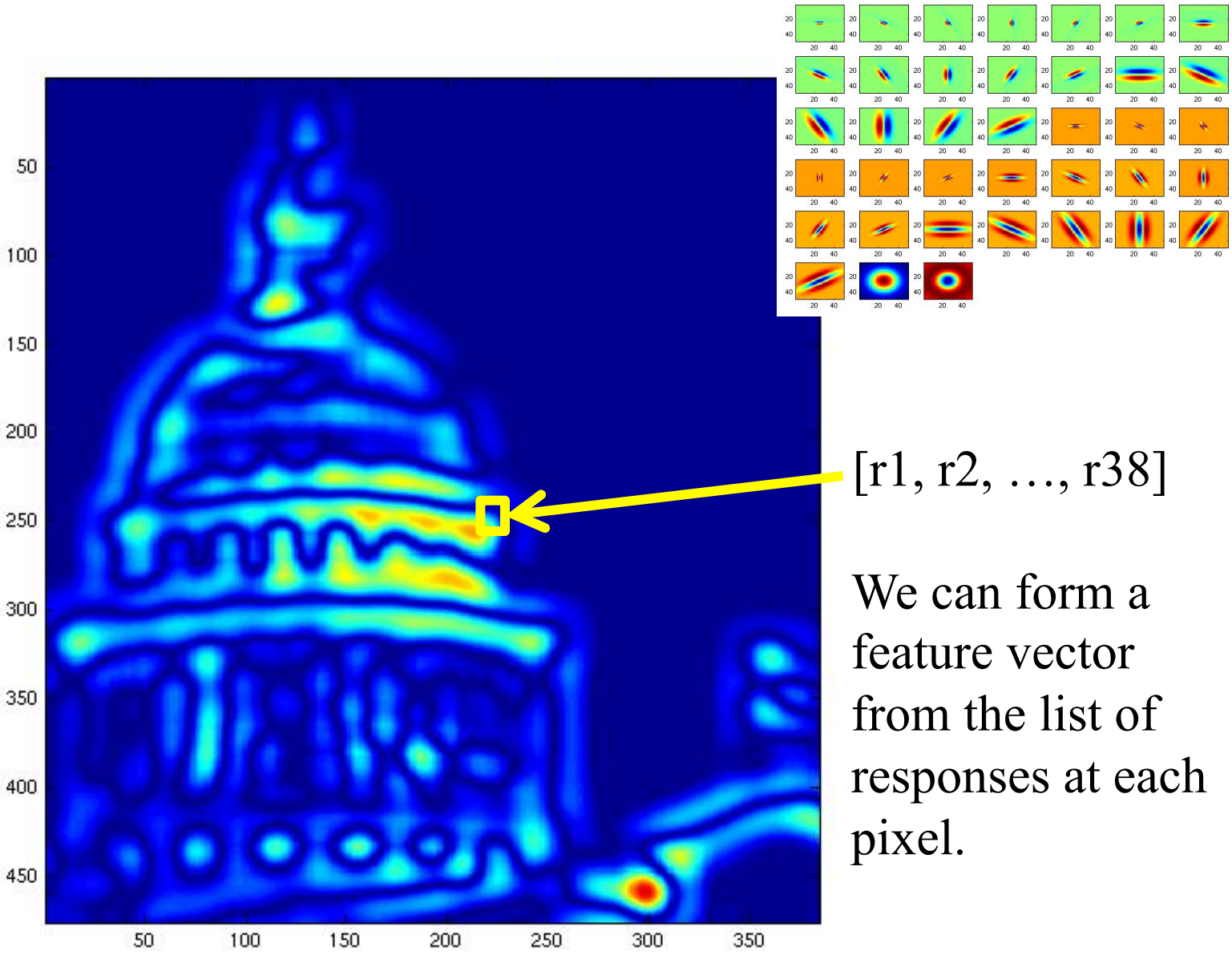


Slide credit Derek Hoiem

Representing texture by mean abs response



Slide credit Derek Hoiem

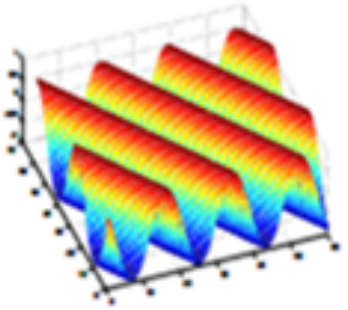


[r1, r2, …, r38]

We can form a feature vector from the list of responses at each pixel.

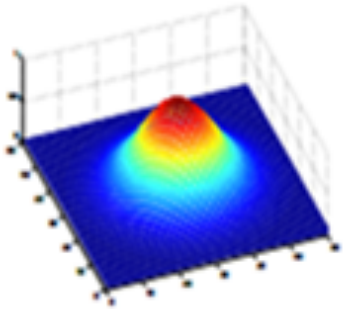
Slide credit: Kristen Grauman

Gabor Filters



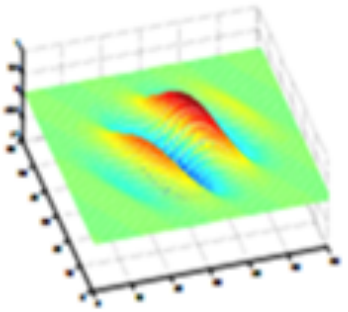
(a)

A Sinusoid oriented 30^0 with X-axis



(b)

A 2-D Gaussian

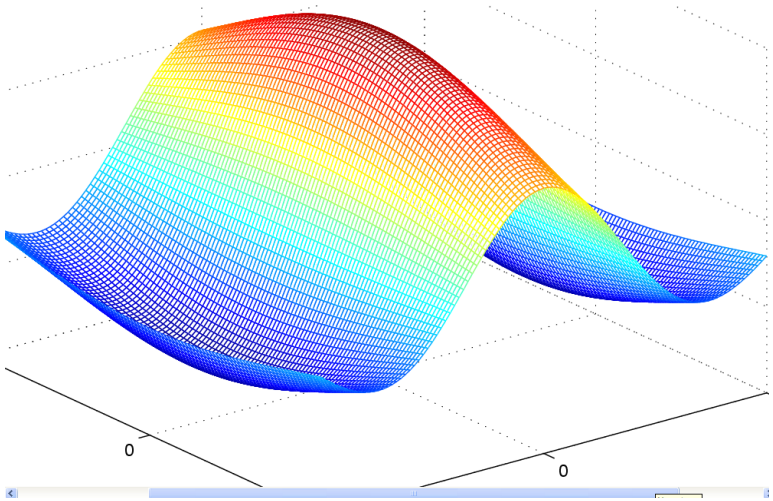


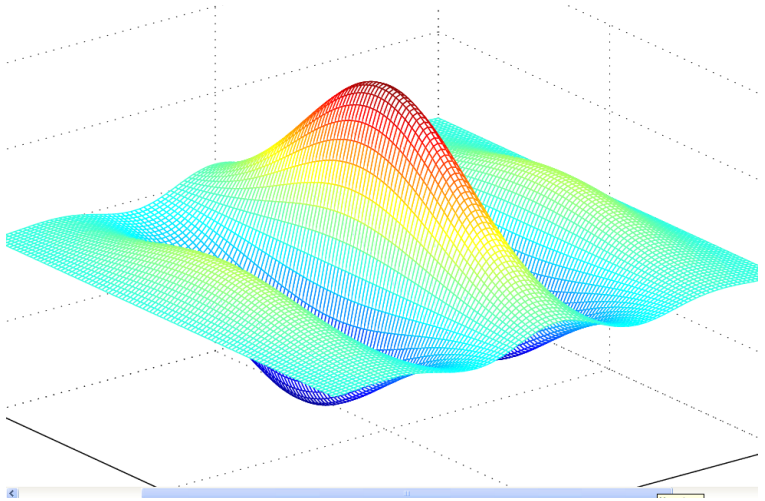
(c)

The corresponding 2-D Gabor filter

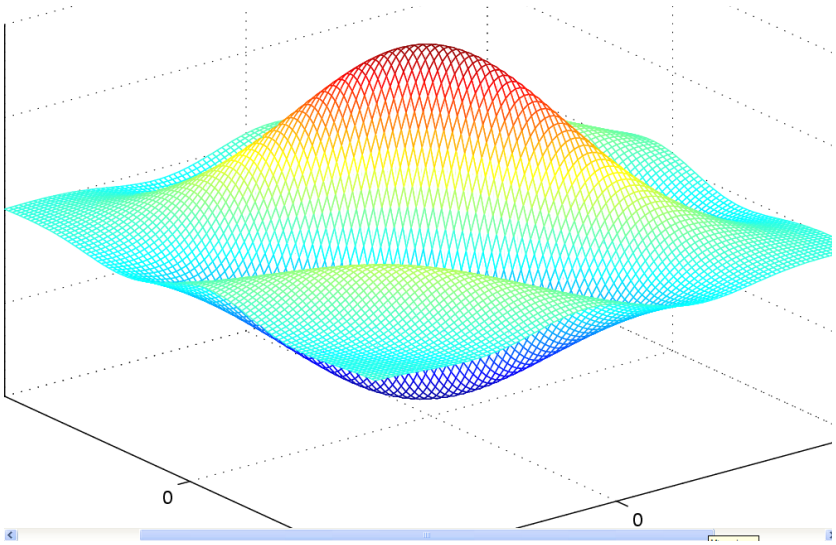
Gabor Filters

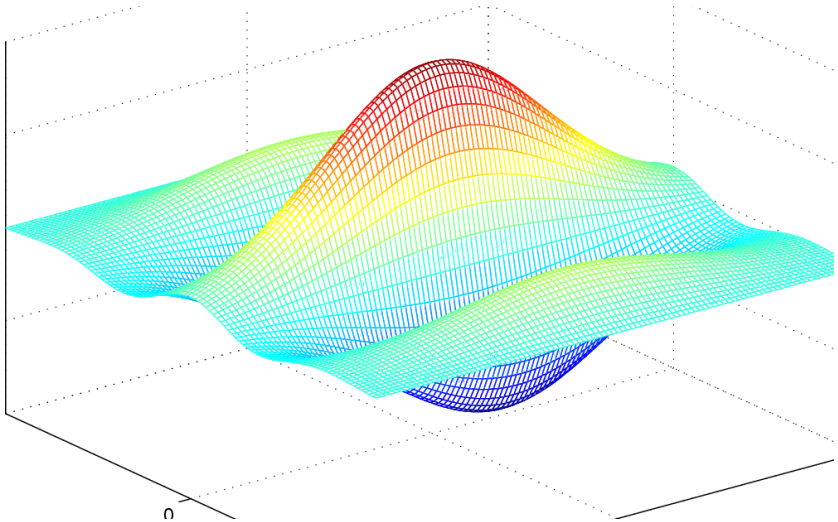
- Wavelets at different frequencies and different orientations



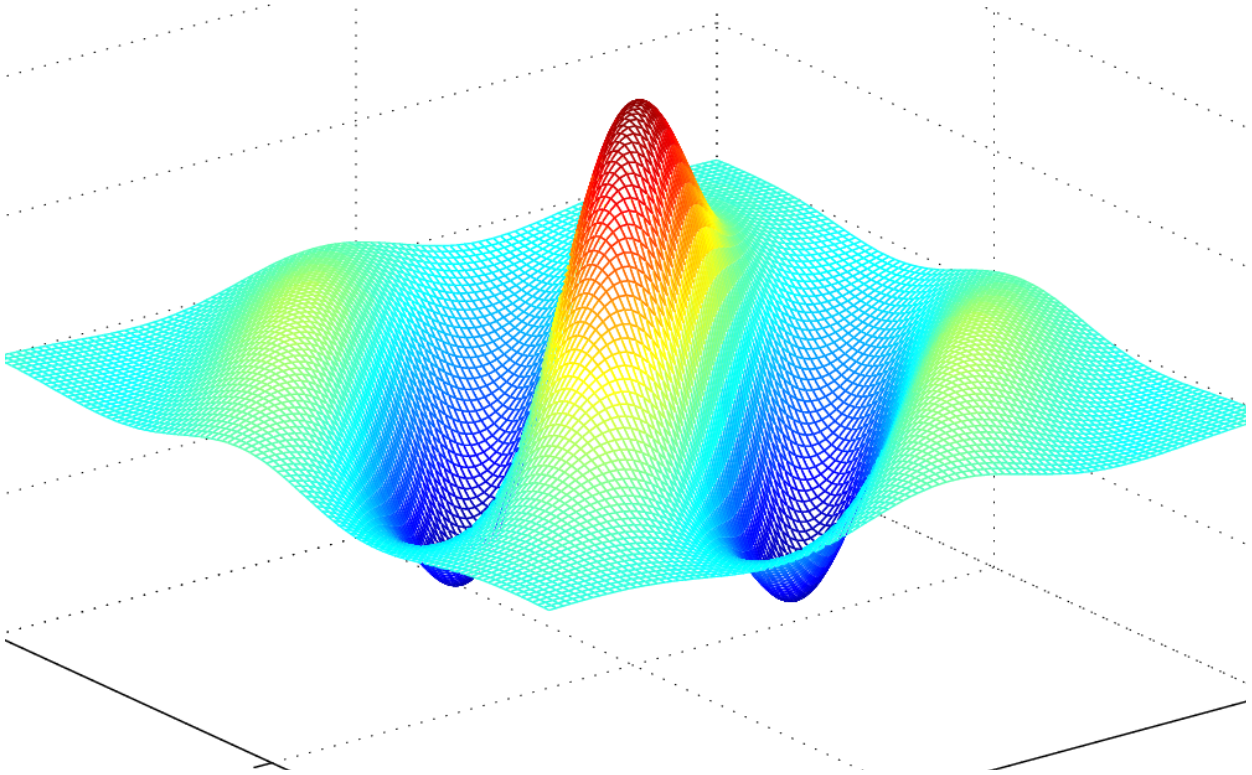


Gabor Filters





Gabor Filters



Gabor filter values visualization by Juergen Mueller.

variable naming conventions as used by http://en.wikipedia.org/wiki/Gabor_filter.
Red: positive values. Blue: negative values

1. Wavelength is changed.
1. wavelength (lambda) : 5.00
2. orientation (theta) : 0.79
3. gaussvar   (sigma) : 20.00
4. phaseoffset (phi)   : 0.00
5. aspectratio (gamma) : 0.50

min gabor val : -0.91
max gabor val : 1.00