Finding lines – part 1

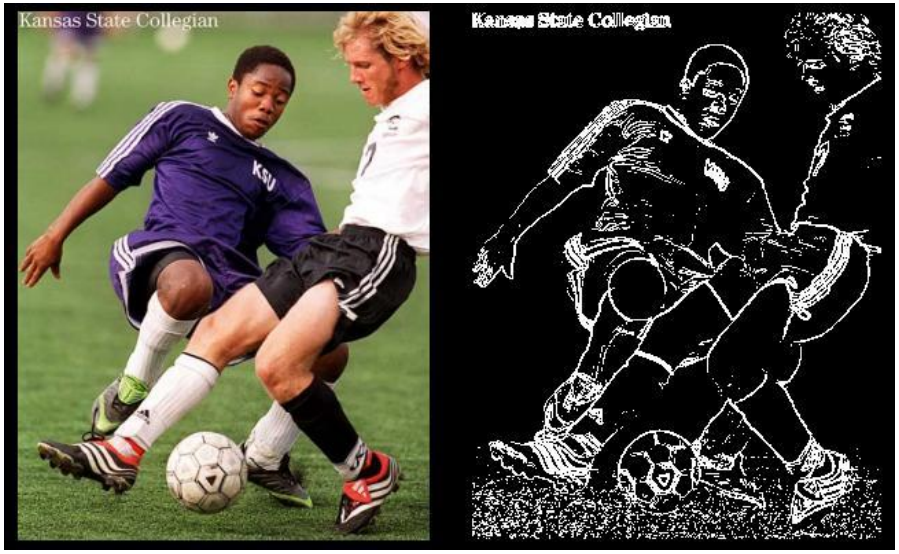
Lihi Zelnik-Manor, Computer Vision

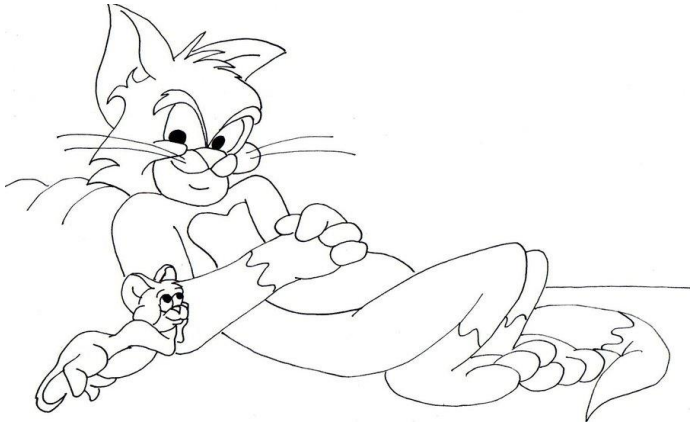
Today

- Edge detection
  - Canny edge detector
  - Berkeley edge probability
- Line fitting
  - Hough transform
  - RANSAC

Why edges?

- We know edges are special



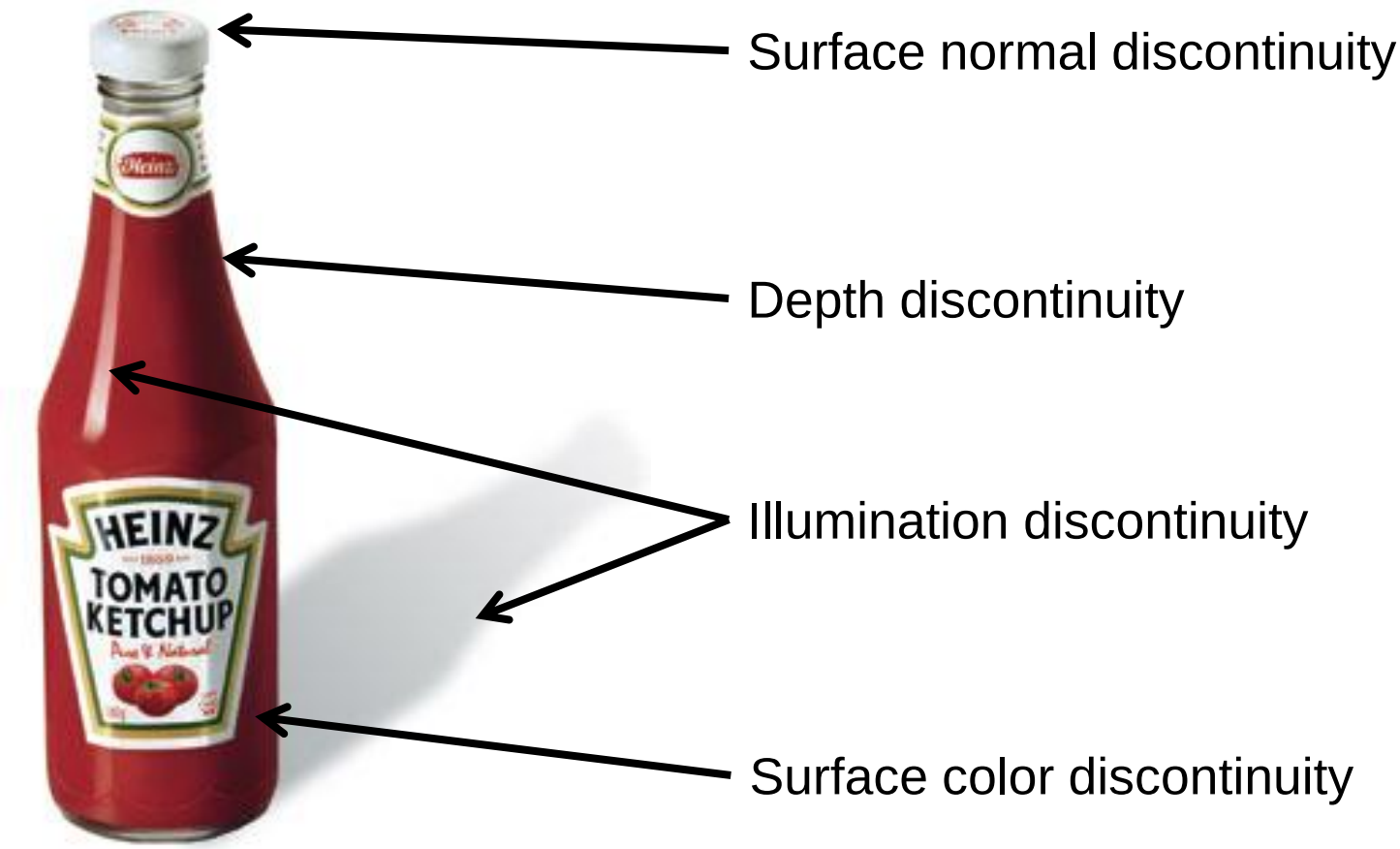


Edge detection - goal

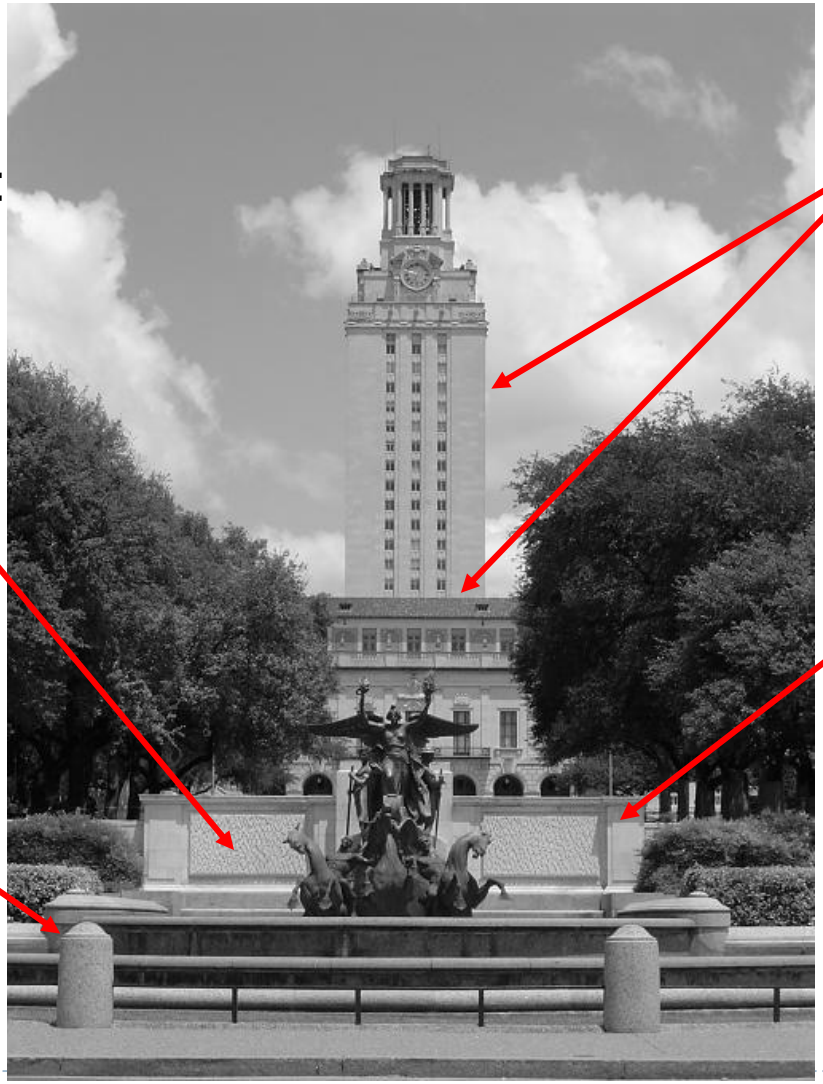
- **Goal**:
  Map image from 2d array of pixels to a set of curves or line segments or contours.
  - Most semantic and shape information from the image can be encoded in the edges
  - A more compact representation than a complete image

- **Ideal:**
  Artist's Line drawing
  (but artists use prior knowledge)

What can cause an edge?



What can cause an edge?



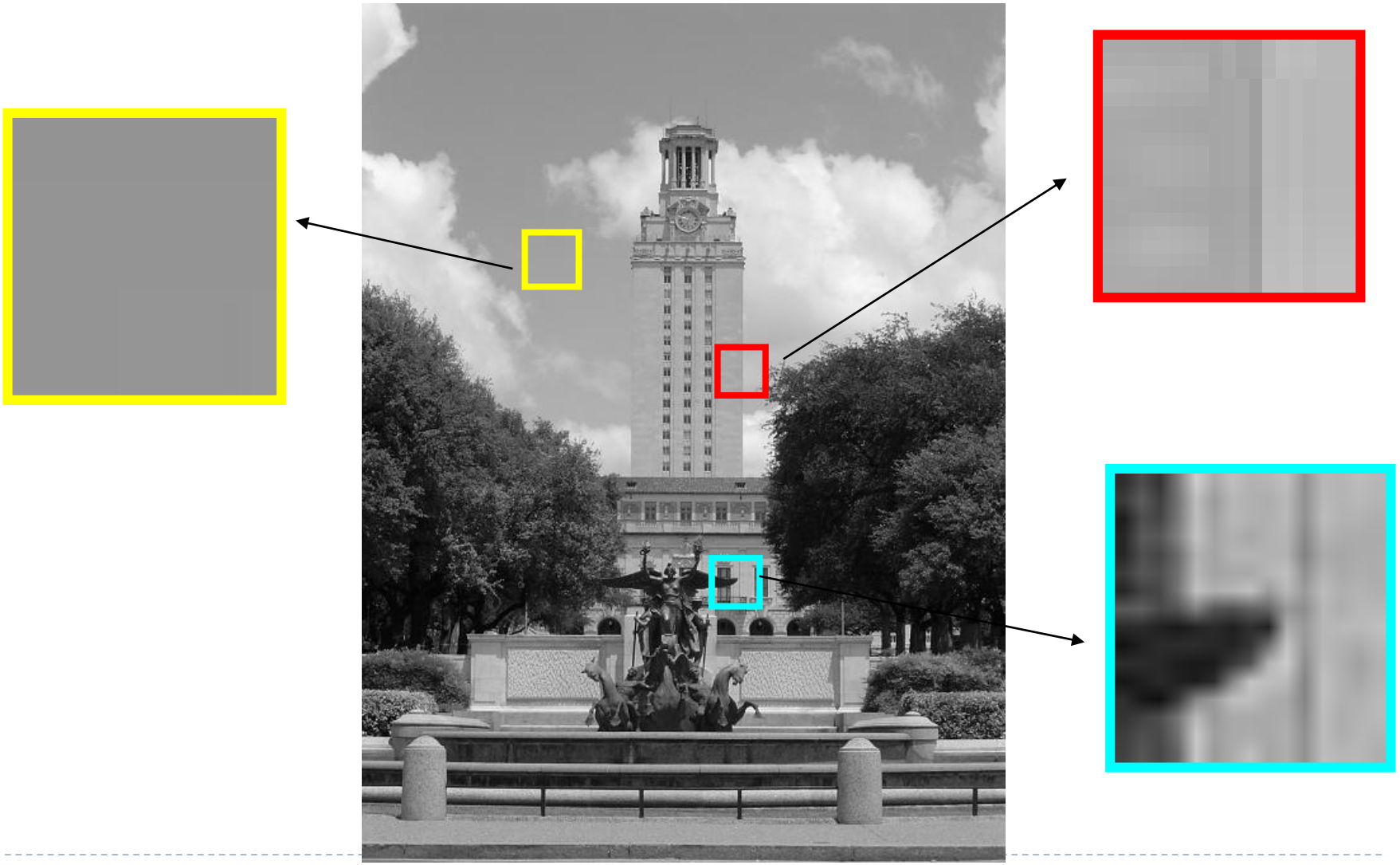
Reflectance change: appearance information, texture

Depth discontinuity: object boundary

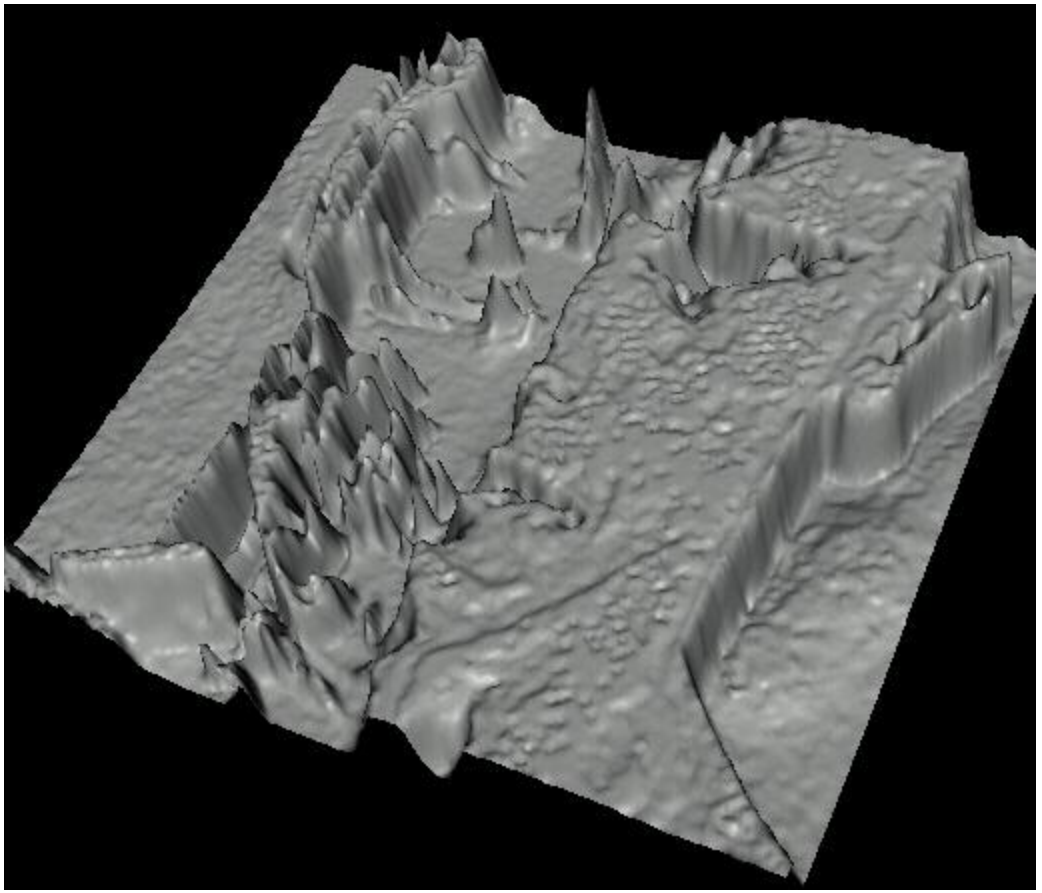
Cast shadows

Change in surface orientation: shape

Contrast and invariance



Edges look like steep cliffs



Source: S. Seitz

Today

- Edge detection
  - Canny edge detector
  - Berkeley edge probability
- Line fitting
  - Hough transform
  - RANSAC

Derivatives and edges

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

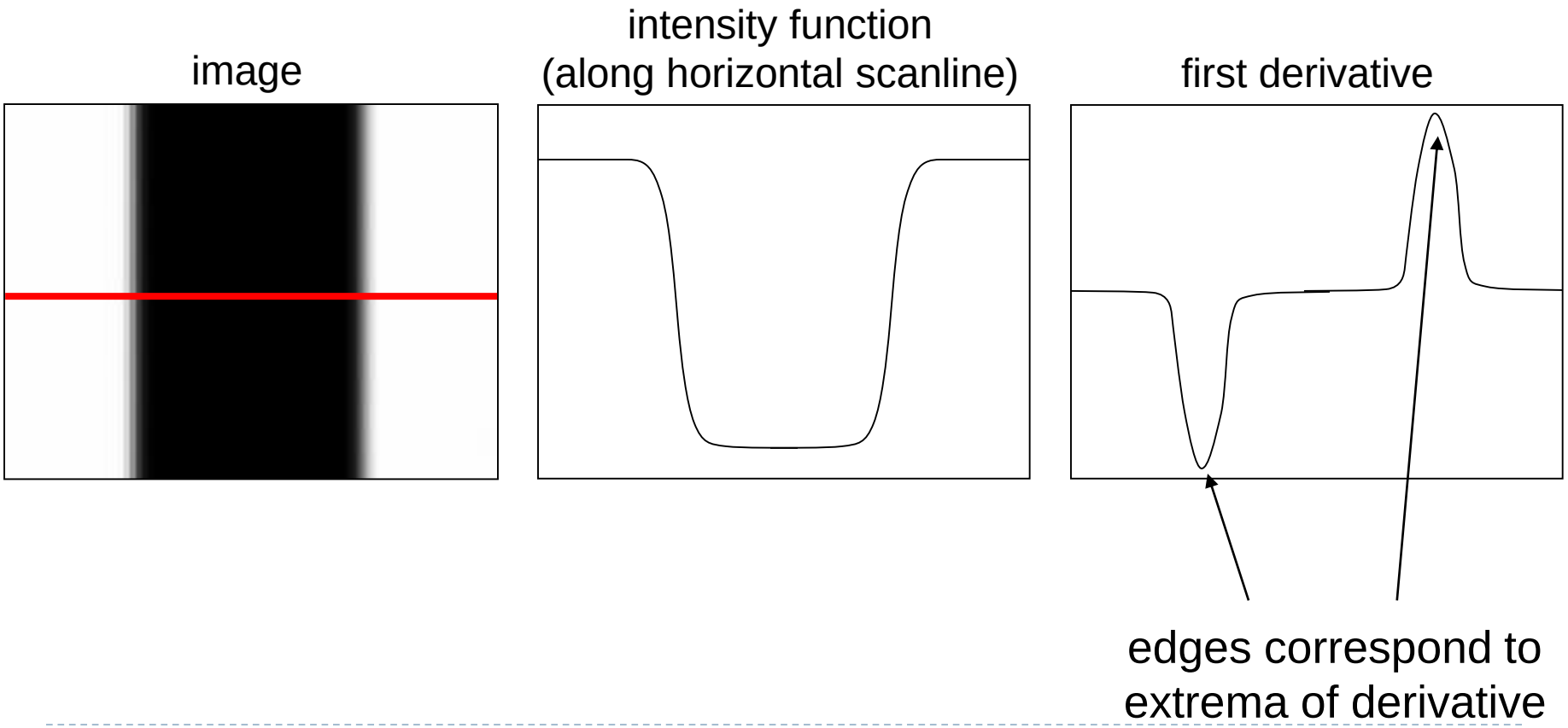
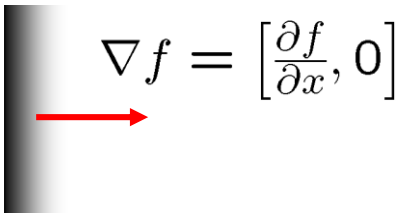


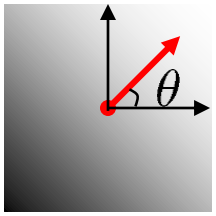
Image gradient

The gradient of an image:

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

The gradient points in the direction of most rapid change in intensity





The gradient direction (orientation of edge normal) is given by:

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

The *edge strength* is given by the gradient magnitude

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

Differentiation and convolution

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

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

For discrete data, we can approximate using finite differences:

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

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

-1	1

Assorted finite difference filters

Prewitt: $M_x =$

-1	0	1
-1	0	1
-1	0	1

; $M_y =$

1	1	1
0	0	0
-1	-1	-1

Sobel: $M_x =$

-1	0	1
-2	0	2
-1	0	1

; $M_y =$

1	2	1
0	0	0
-1	-2	-1

Roberts: $M_x =$

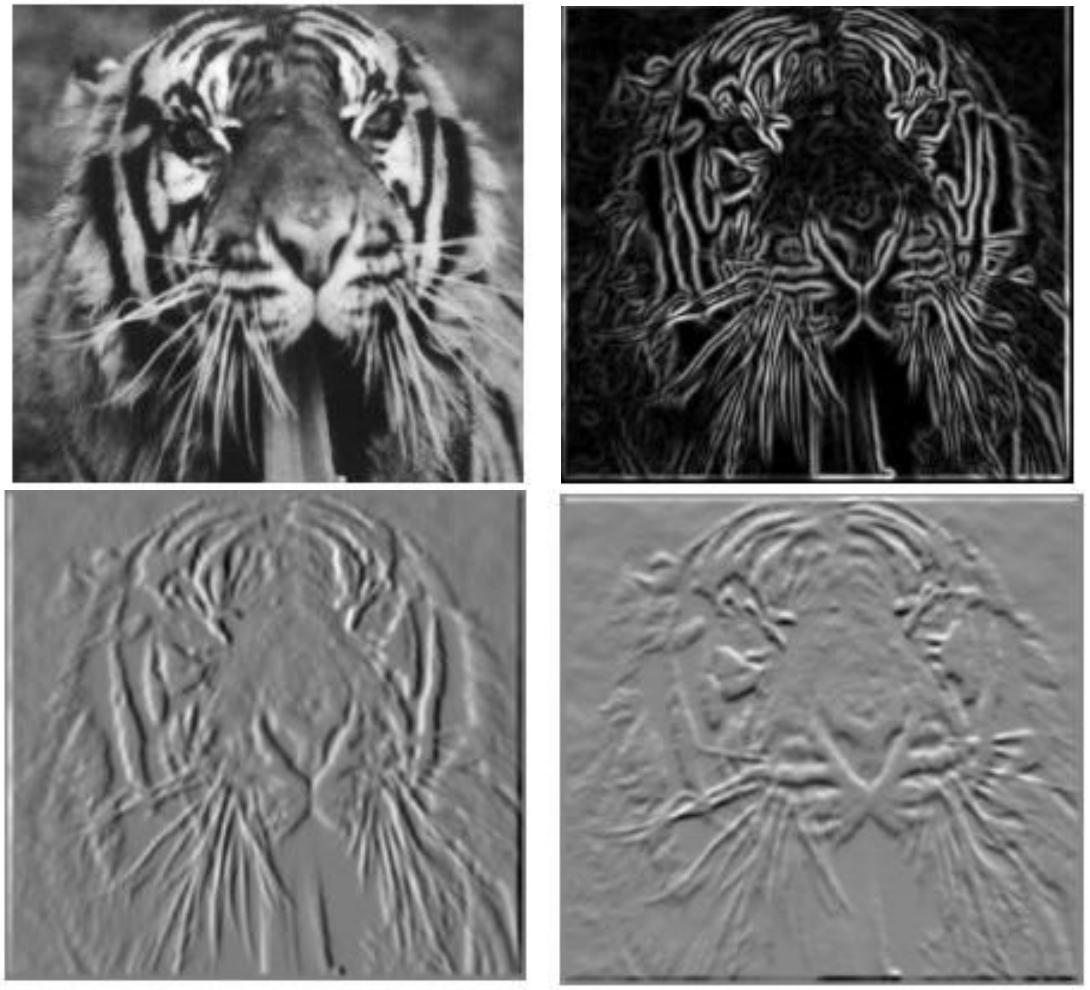
0	1
-1	0

; $M_y =$

1	0
0	-1

```
>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
```

Partial derivatives of an image



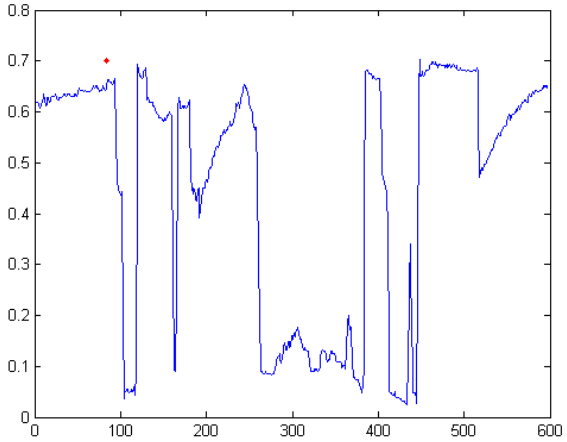
$$\frac{\partial f(x,y)}{\partial x}$$

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

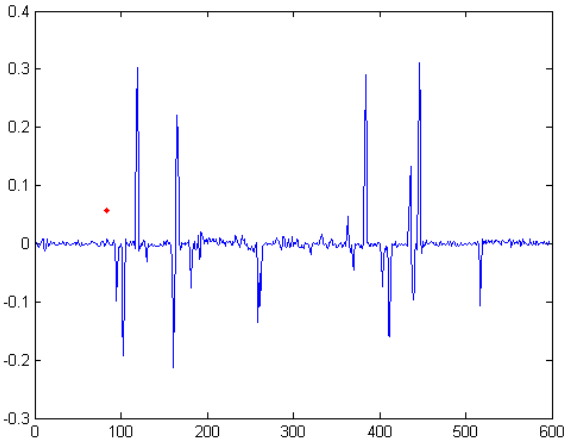
Which shows changes with respect to x? y?

Intensity profile of one row

Intensity



Gradient

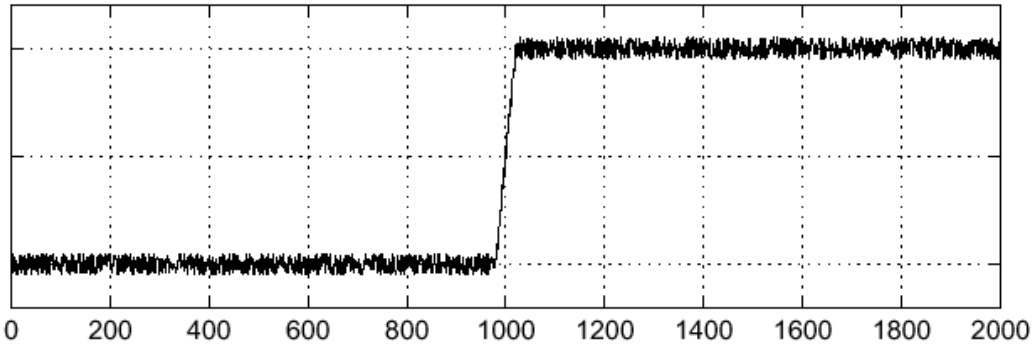


Effects of noise

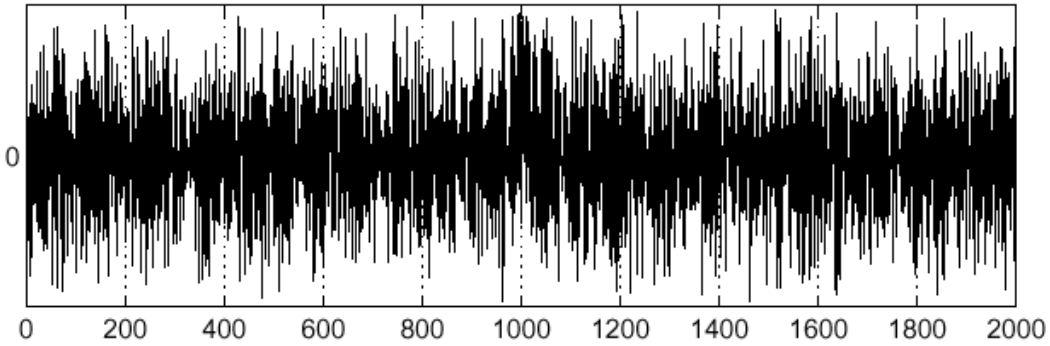
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response

- What can be done?

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response

- What can be done?
  - Smoothing the image should help – it forces pixels to look more like their neighbors

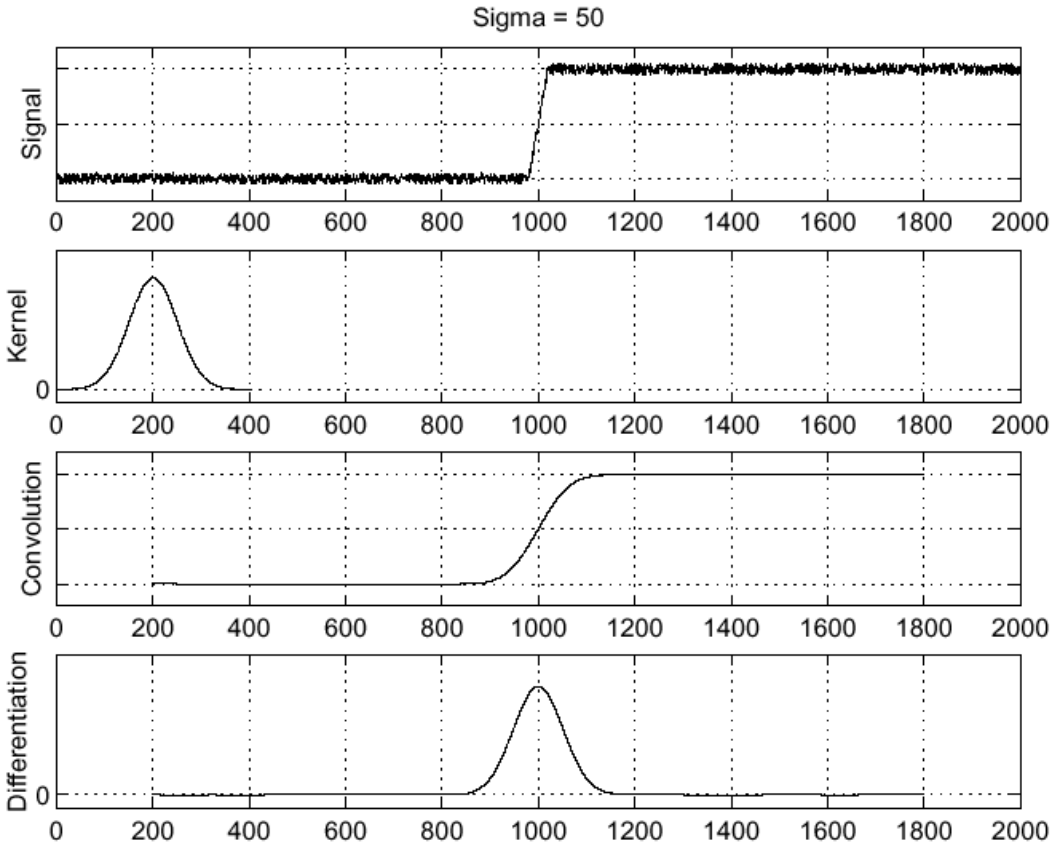
Solution: smooth first

f

g

$f * g$

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



Where is the edge?

Look for peaks in $\frac{d}{dx}(f * g)$

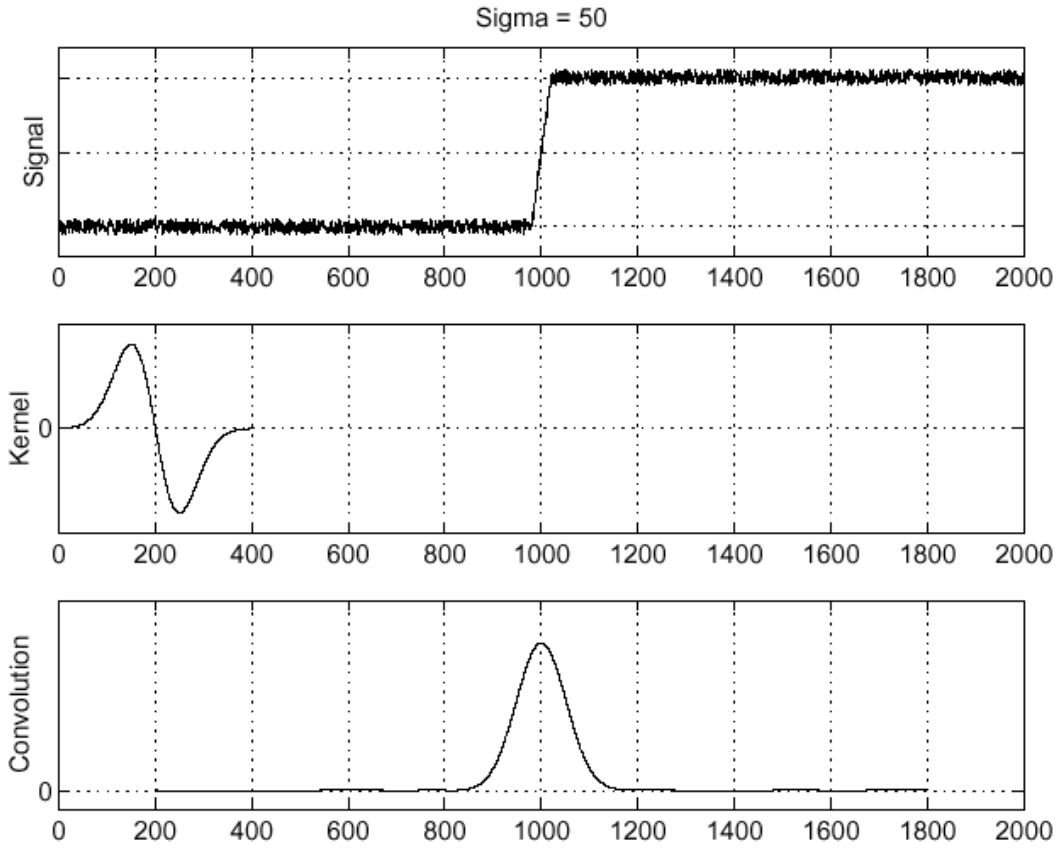
Derivative theorem of convolution

Differentiation property of convolution: $\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$

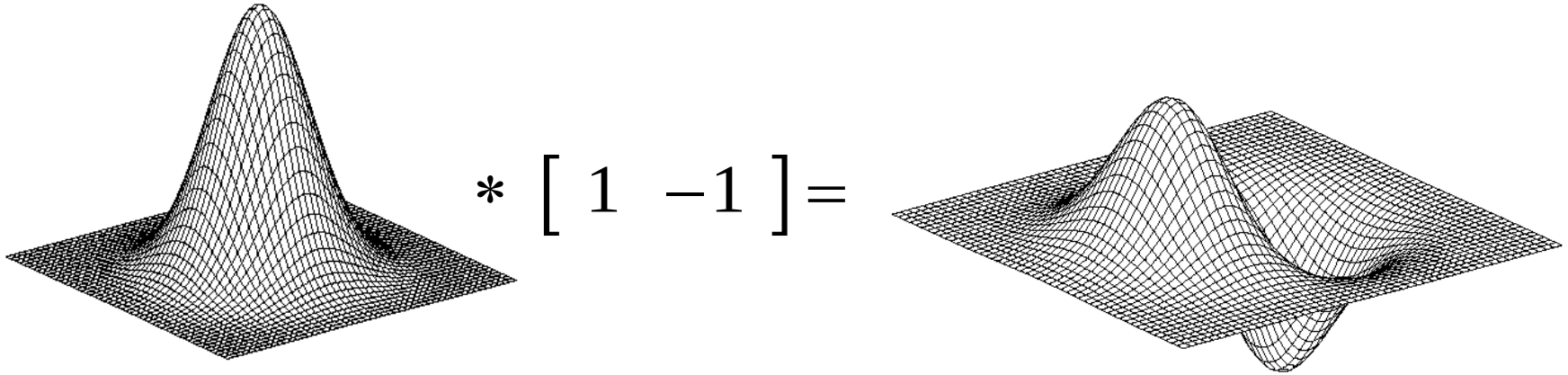
f

$\frac{d}{dx} g$

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

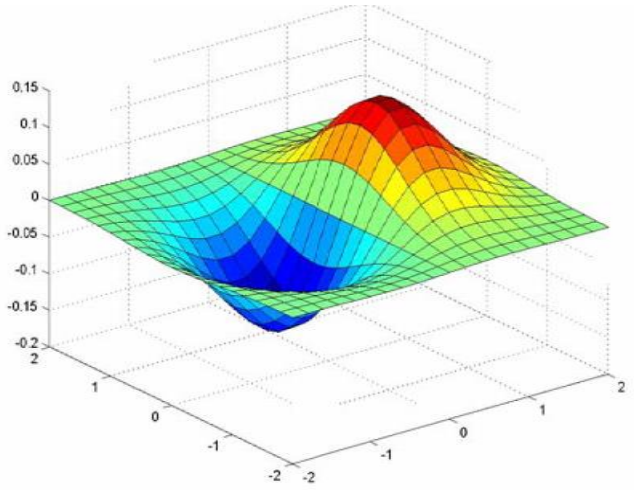


Derivative of Gaussian filter

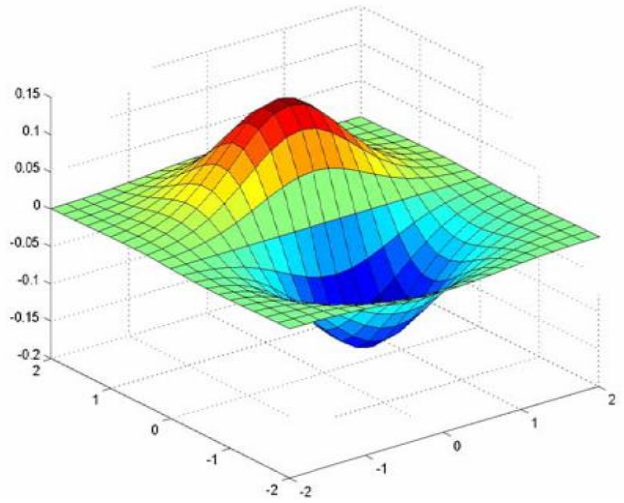


Is this a separable filter?

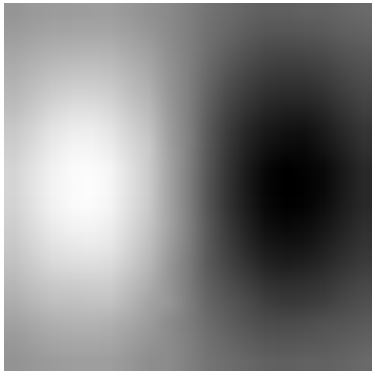
Derivative of Gaussian filters

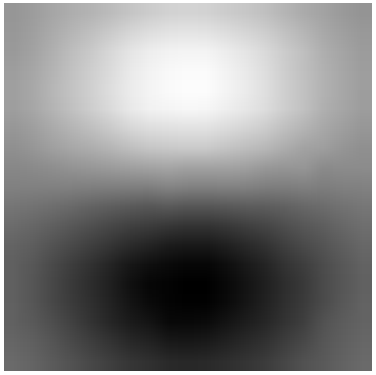


x-direction



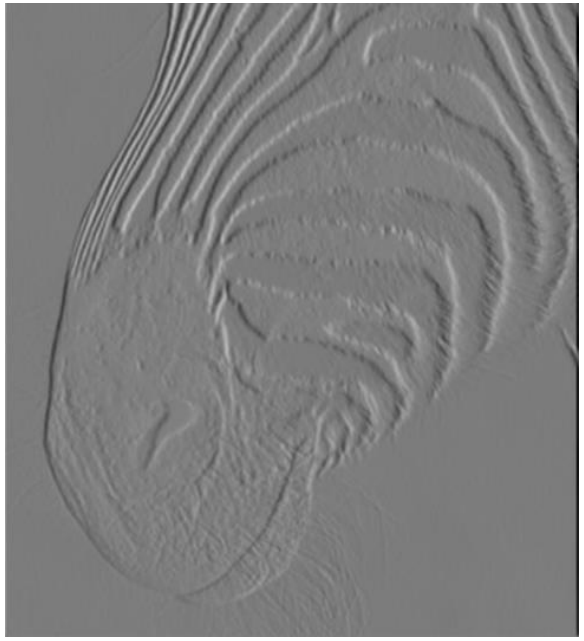
y-direction



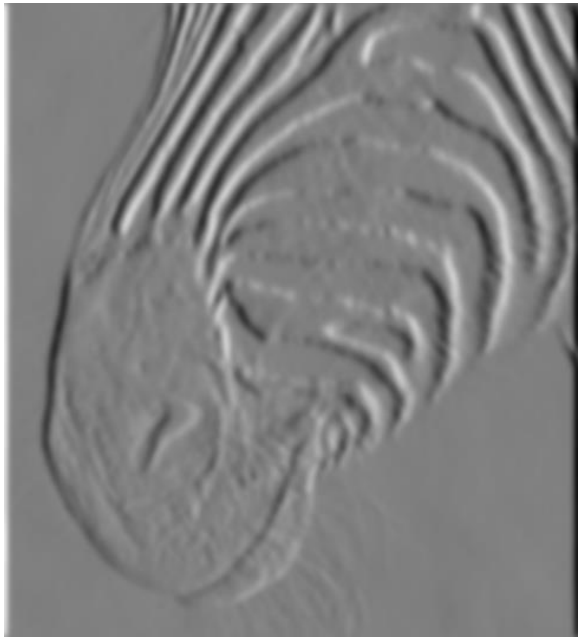


Which one finds horizontal/vertical edges?

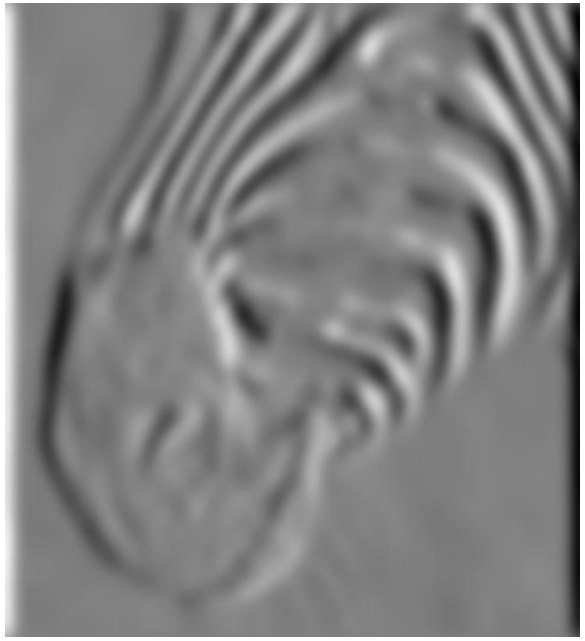
Smoothing tradeoffs



1 pixel



3 pixels



7 pixels

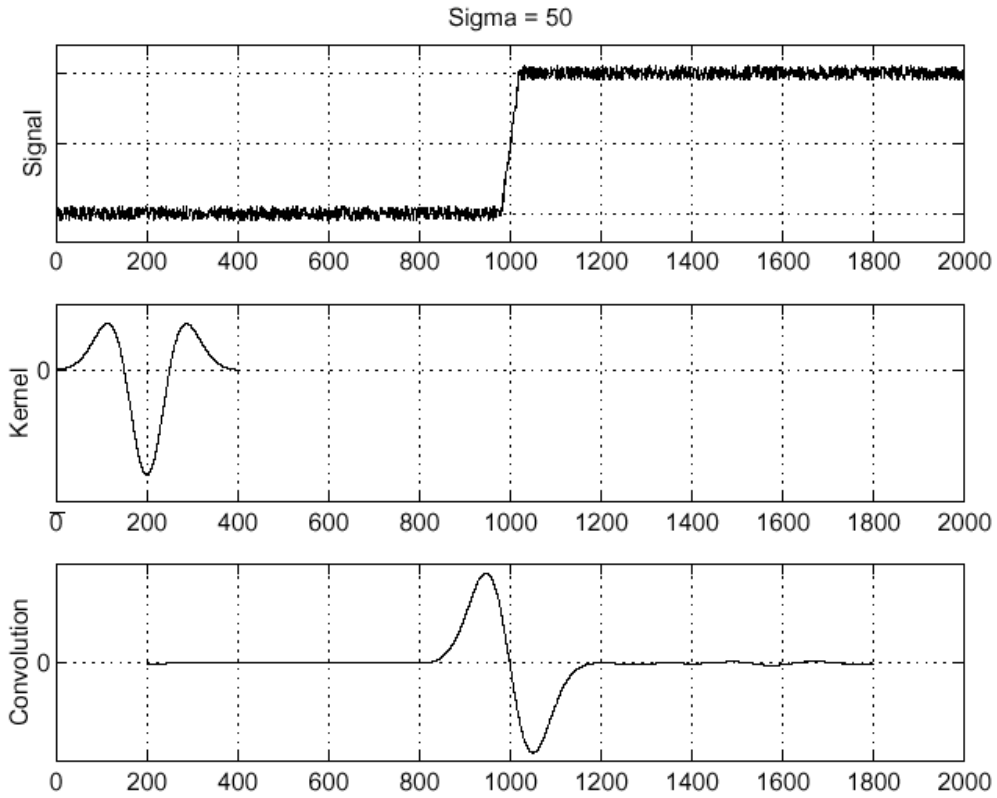
Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Laplacian of Gaussian (LoG filter)

f

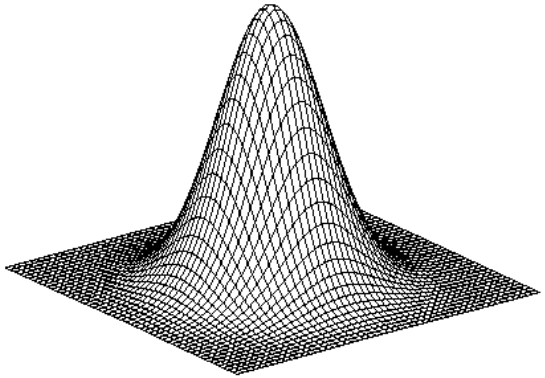
$\frac{d^2}{dx^2} g$

$f * \frac{d^2}{dx^2} g$



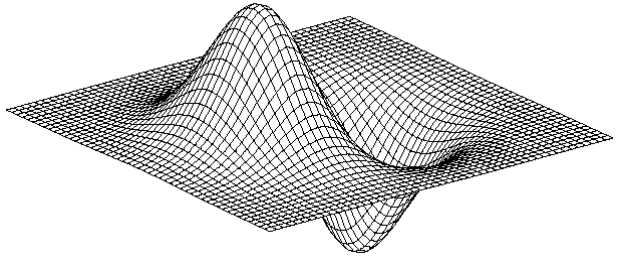
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

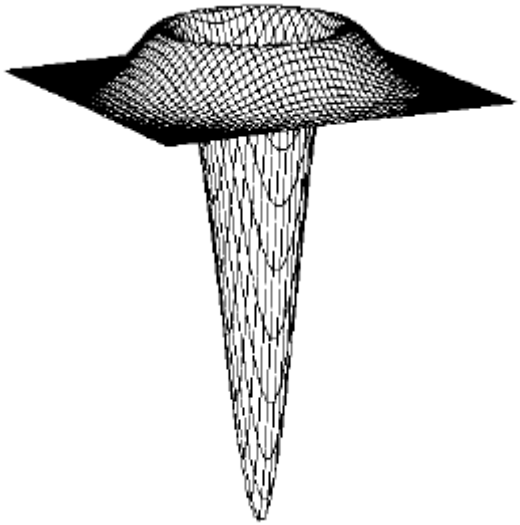
$$g$$



derivative of Gaussian

$$\frac{dg}{dx}$$

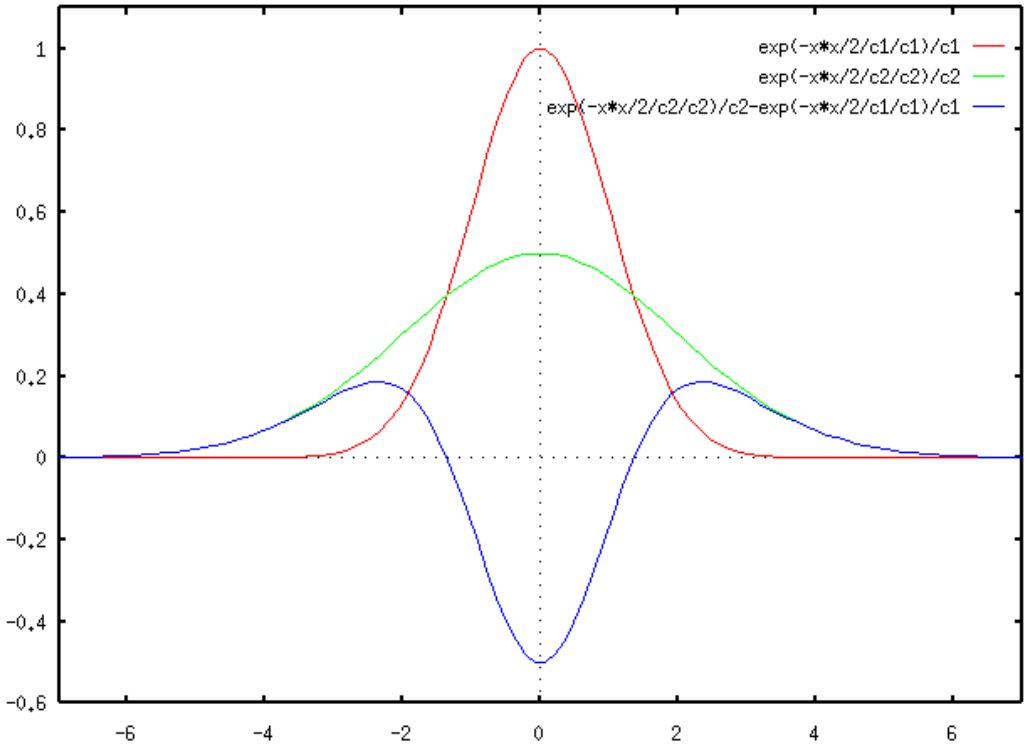
Laplacian of Gaussian



$$\frac{d^2g}{dx^2} + \frac{d^2g}{dy^2}$$

DoG = Difference of Gaussians

- Can approximate Laplacian filter

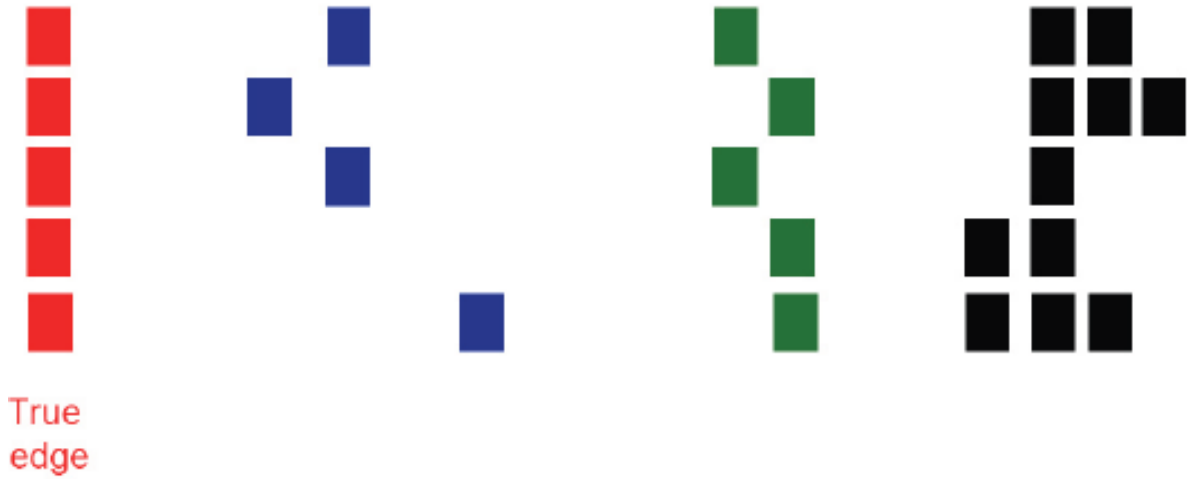


What is a good edge detector?

- **Good detection:**
  - Minimize false positives (wrong detections)
  - Minimize false negatives (missing real edges)
  - Maximize true detections
- **Good localization:**
  - Detected edges should be as close as possible to the true edges
- **Single response:**
  - Return a single detection for each true edge point
- **Connect detections to lines**

What is a good edge detector?

- **Good detection**
- **Good localization**
- **Single response**



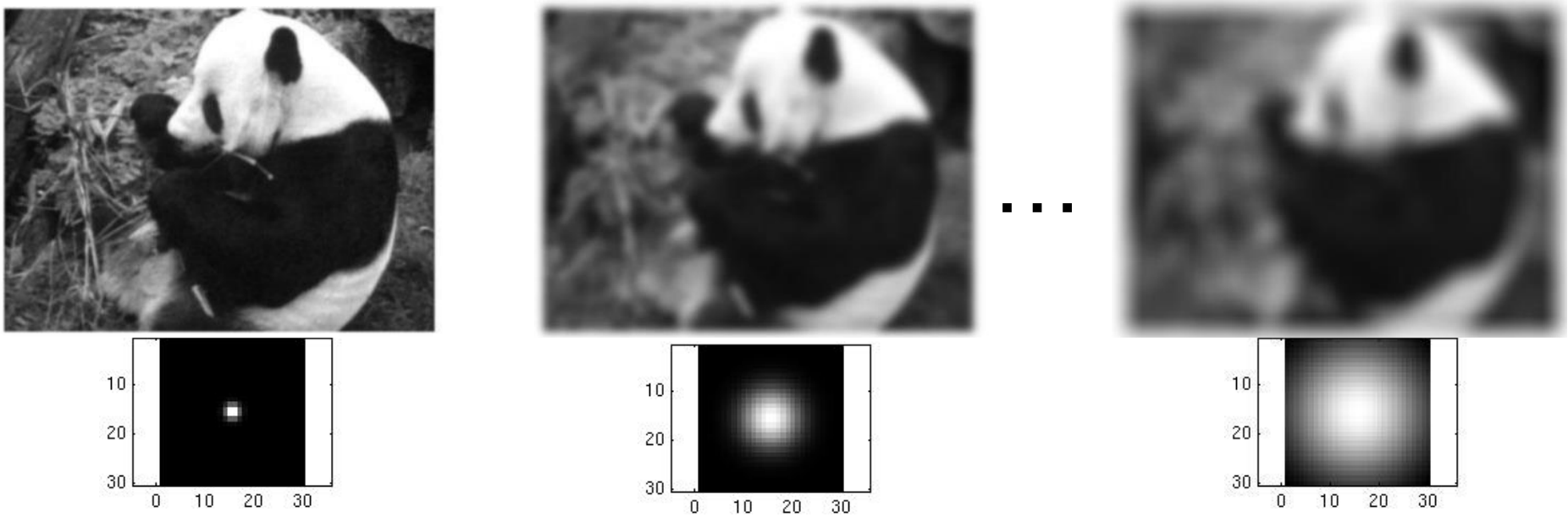
Which of these detections is the best?

What are the parameters?

- Scale
- Threshold

Scale selection

- Recall: We first smooth the image with a Gaussian kernel to reduce noise.
- The scale of the Gaussian determines how much smoothing we apply



Effect of σ on derivatives

- The apparent structures differ depending on Gaussian's scale parameter.

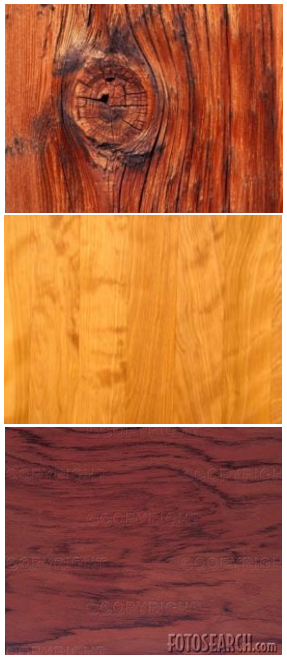
- Large scale: larger scale edges detected
- Small scale: finer features detected

σ = 1 pixel

σ = 3 pixels

How do we chose the scale?

- It depends what we're looking for:



- Too fine of a scale…can't see the forest for the trees.
- Too coarse of a scale…can't tell the maple grain from the cherry.

What are the parameters?

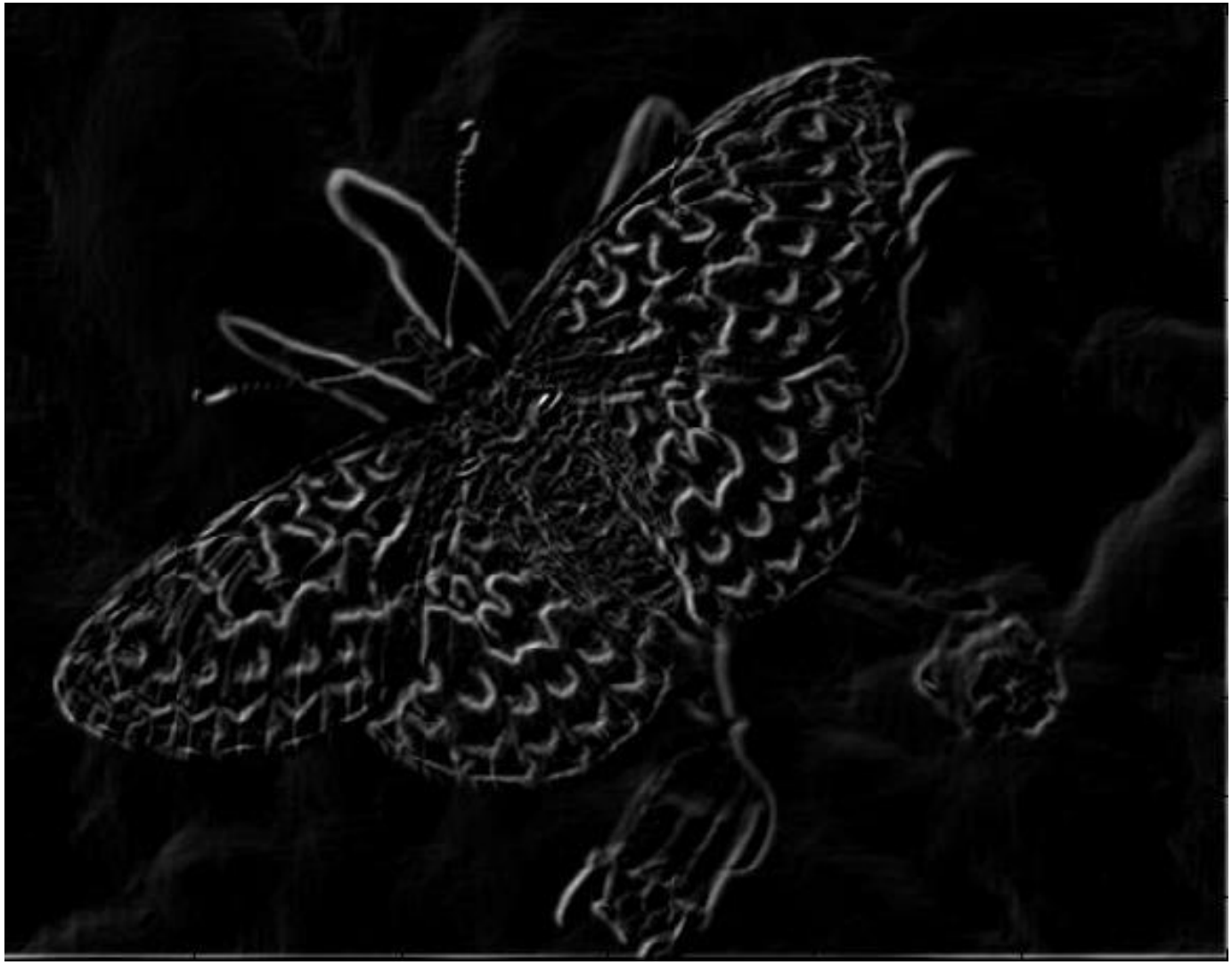
- Scale
- Threshold

Thresholding

- Choose a threshold value
- Set any pixels less than *thresh* to zero (off)
- Set any pixels greater than or equal to *thresh* to one (on)

Original image

Gradient magnitude



Using a low threshold

Using a higher threshold

The Canny edge detector

- Probably the most widely used edge detector in computer vision

- Key idea:
  detect step-edges that are corrupted by additive Gaussian noise

- Theorem:
  Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization

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

Canny edge detector

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

- MATLAB: `edge(image, 'canny');`
- `>>help edge`

Example - input

original image (Lena)

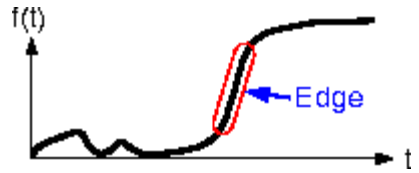
Example – step 1

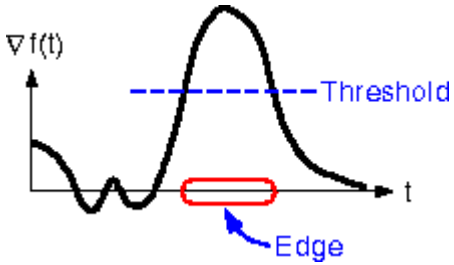
norm of the gradient

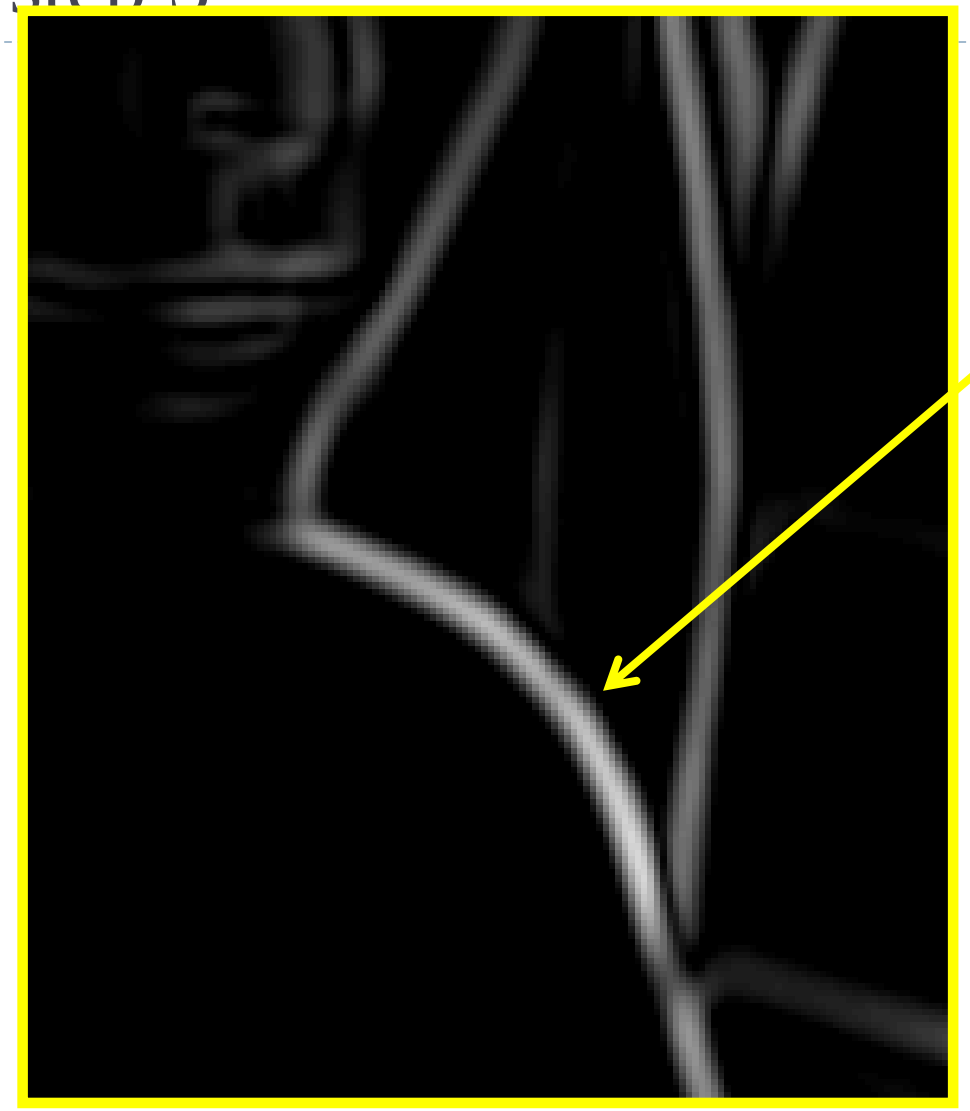
Example – step 2

thresholding

Example – step 3





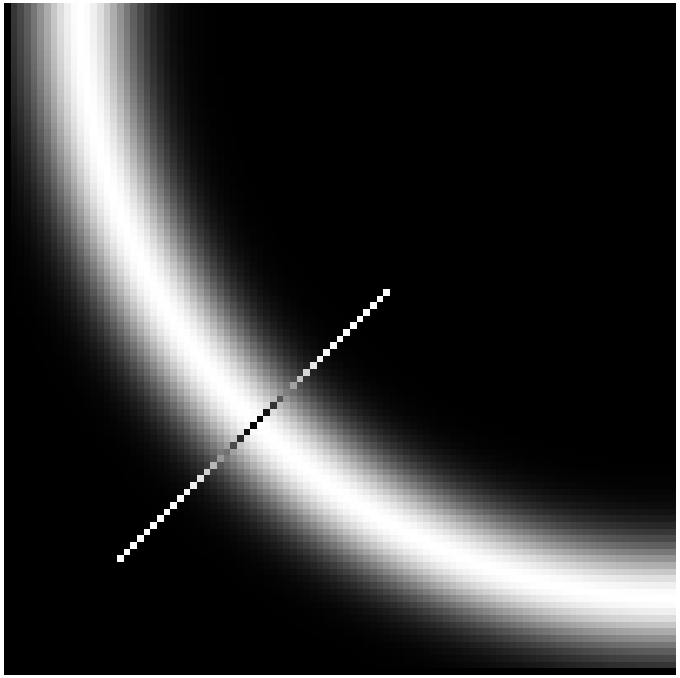


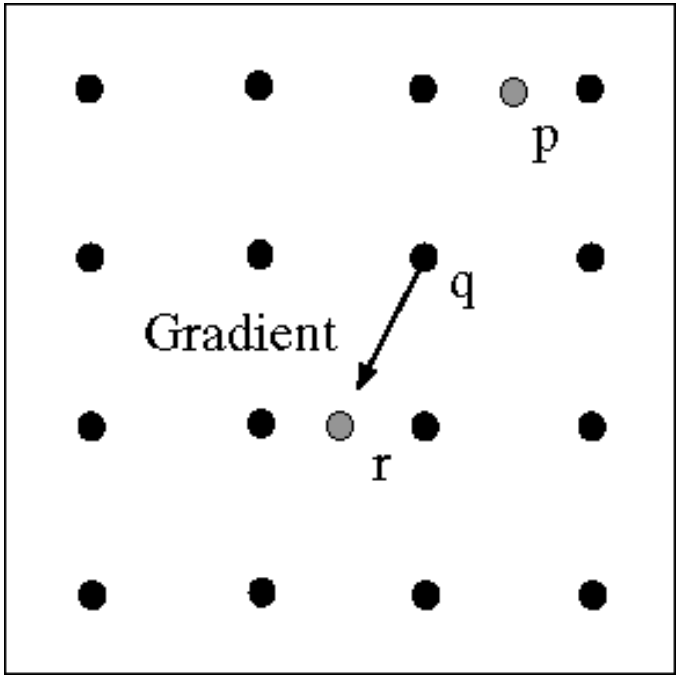
How to turn these thick regions of the gradient into curves?

Non-maximum suppression

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

- requires checking interpolated pixels *p* and *r*





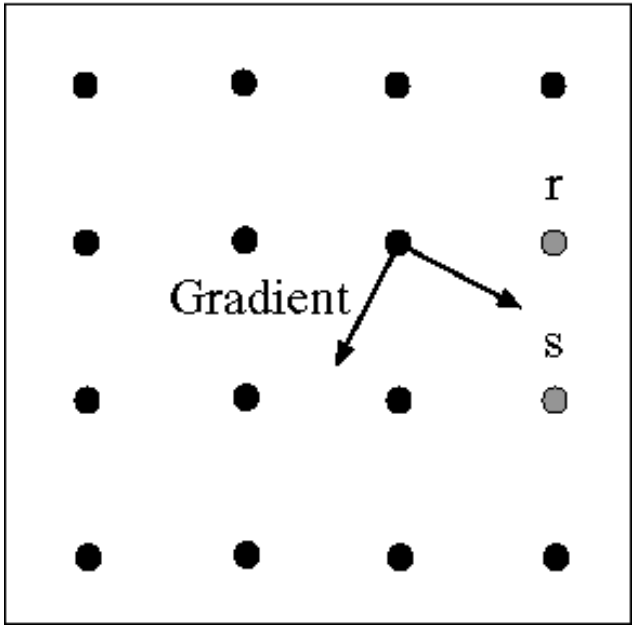
Example – step 3

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

Thinning (non-maximum supression)

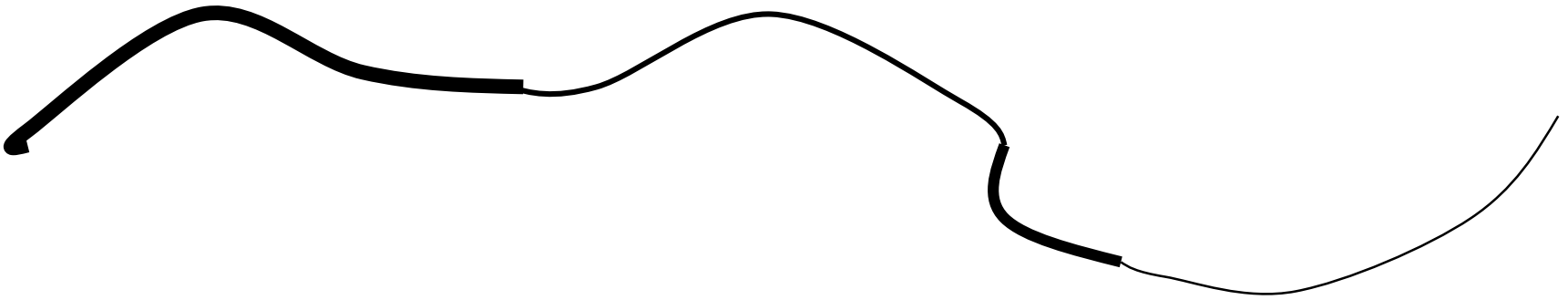
Edge linking

- Assume the marked point is an edge point.
- Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either *r* or *s*).



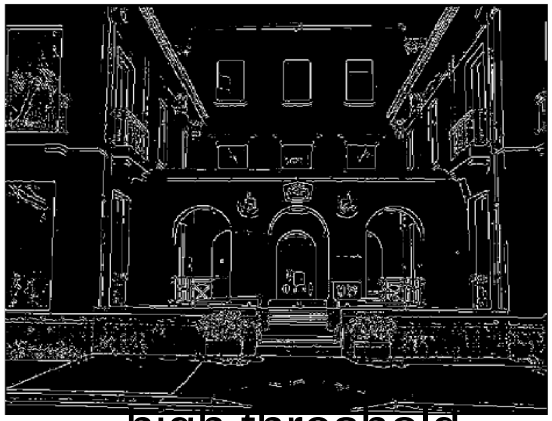
Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Hysteresis thresholding

original image



high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold

Hysteresis thresholding

high threshold
(strong edges)

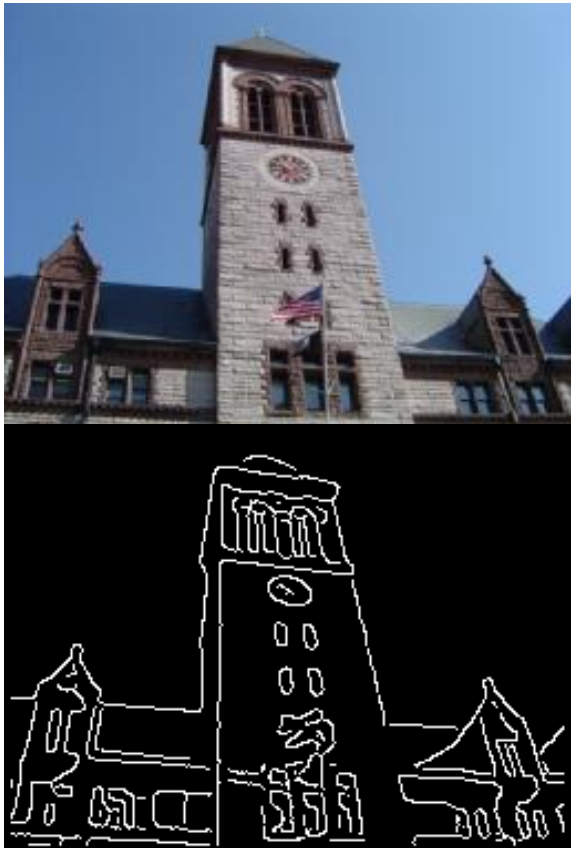
Hysteresis thresholding

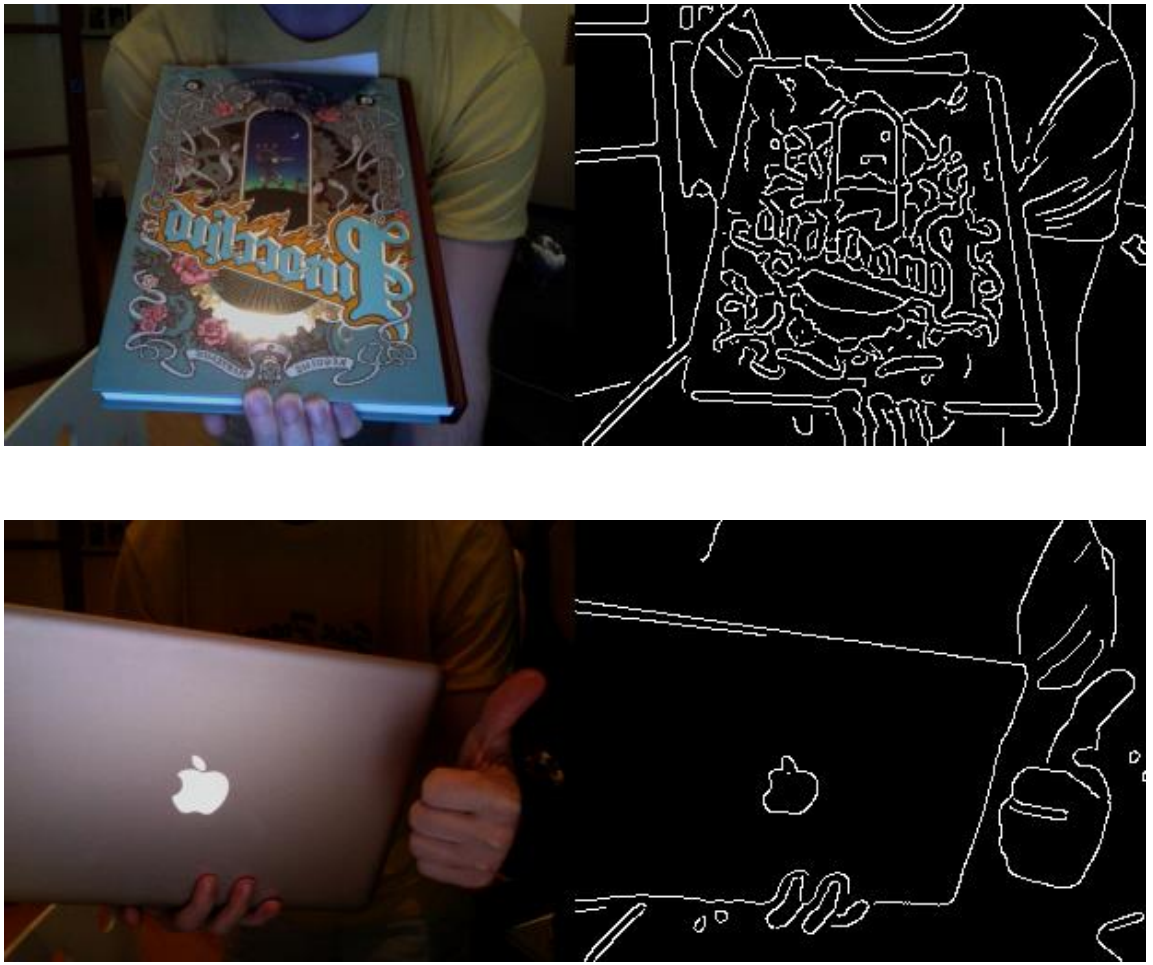
low threshold
(weak edges)

Hysteresis thresholding

hysteresis threshold

Canny - results





Object boundaries vs. edges

Background

Texture

Shadows

Today

- Edge detection
  - Canny edge detector
  - Berkeley edge probability
- Line fitting
  - Hough transform
  - RANSAC

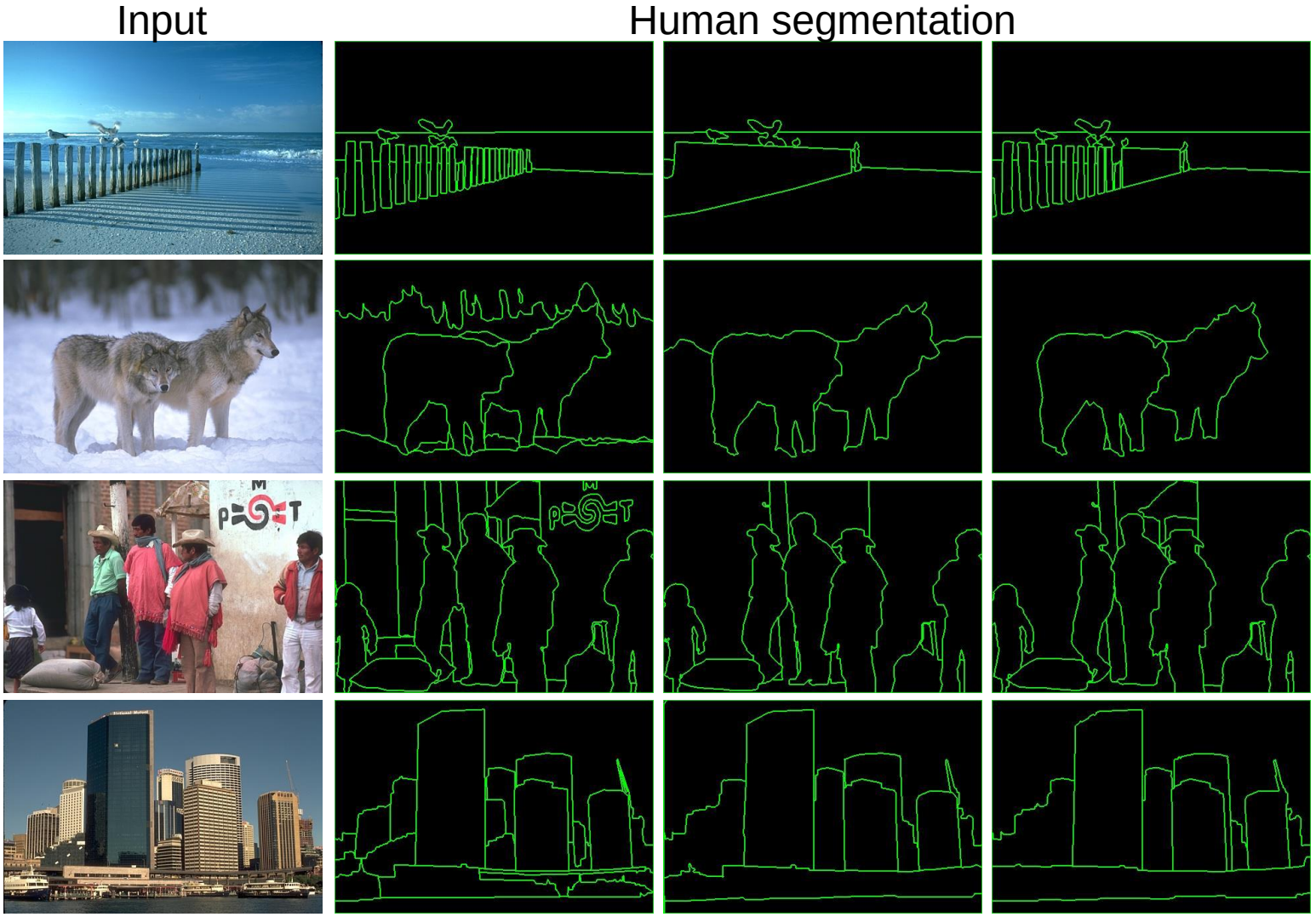
At Berkeley:

1. Collect Data Set of Human segmented images
2. Learn Local Boundary Model for combining brightness, color and texture
3. Global framework to capture closure, continuity
4. Detect and localize junctions
5. Integrate low, mid and high-level information for grouping and figure-ground segmentation

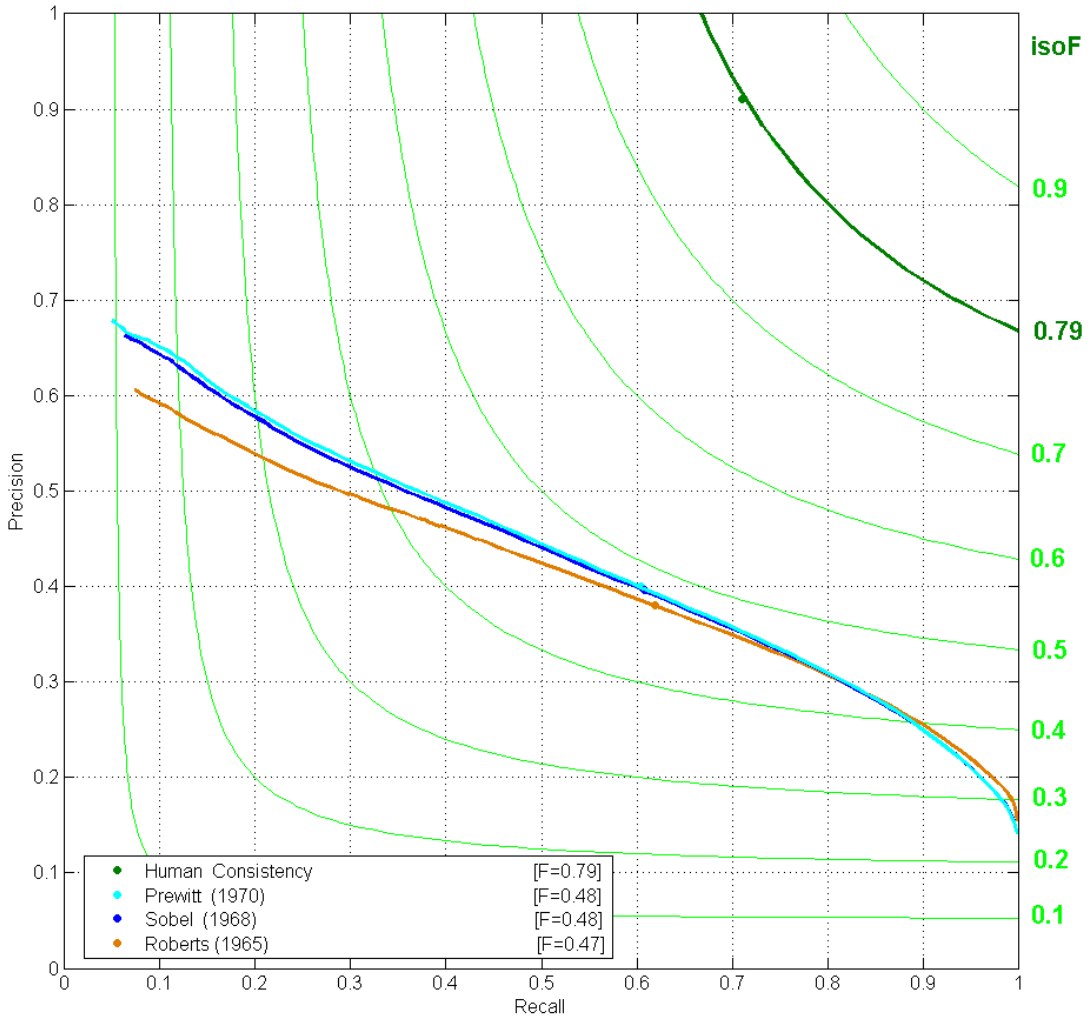
D. Martin, C. Fowlkes, D. Tal, J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics", ICCV, 2001

htpp://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

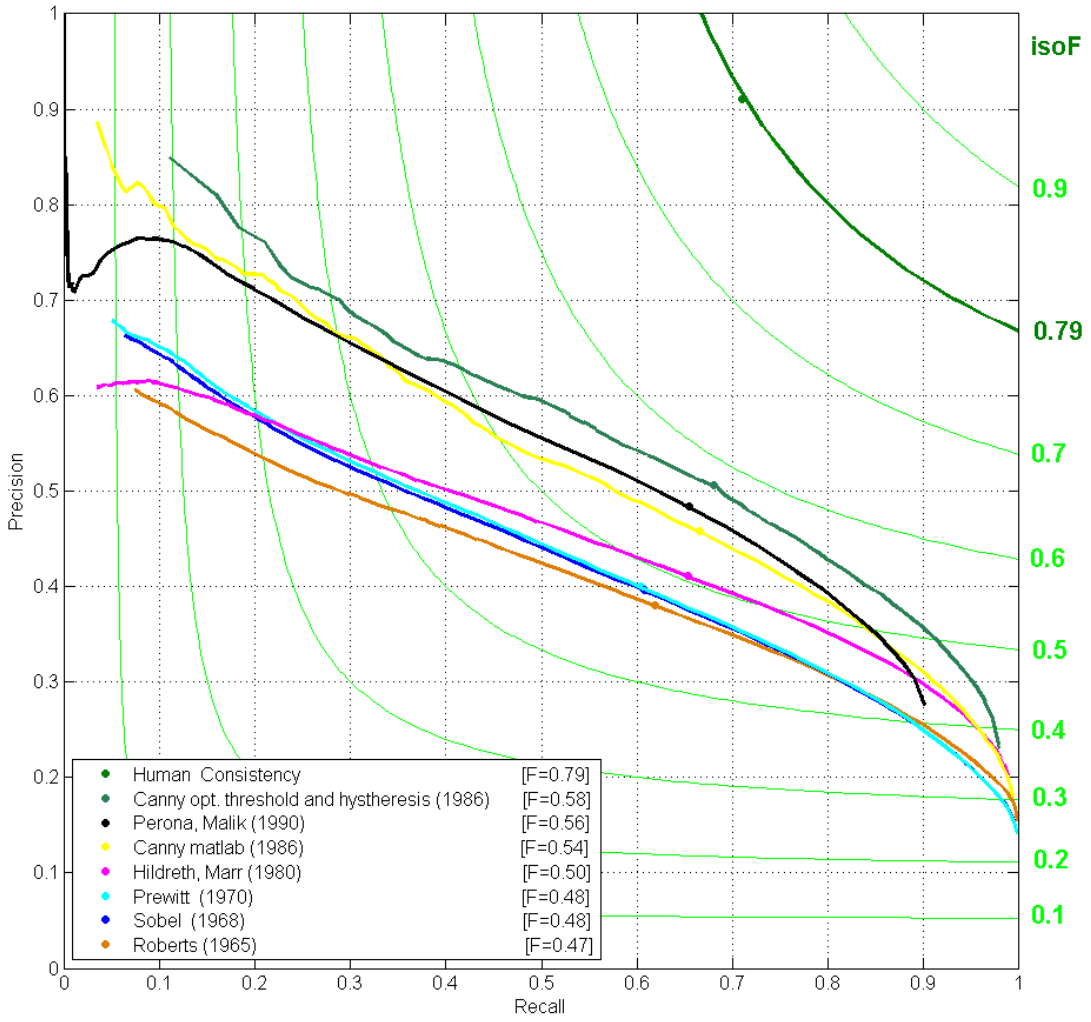
Berkeley Segmentation DataSet [BSDS]



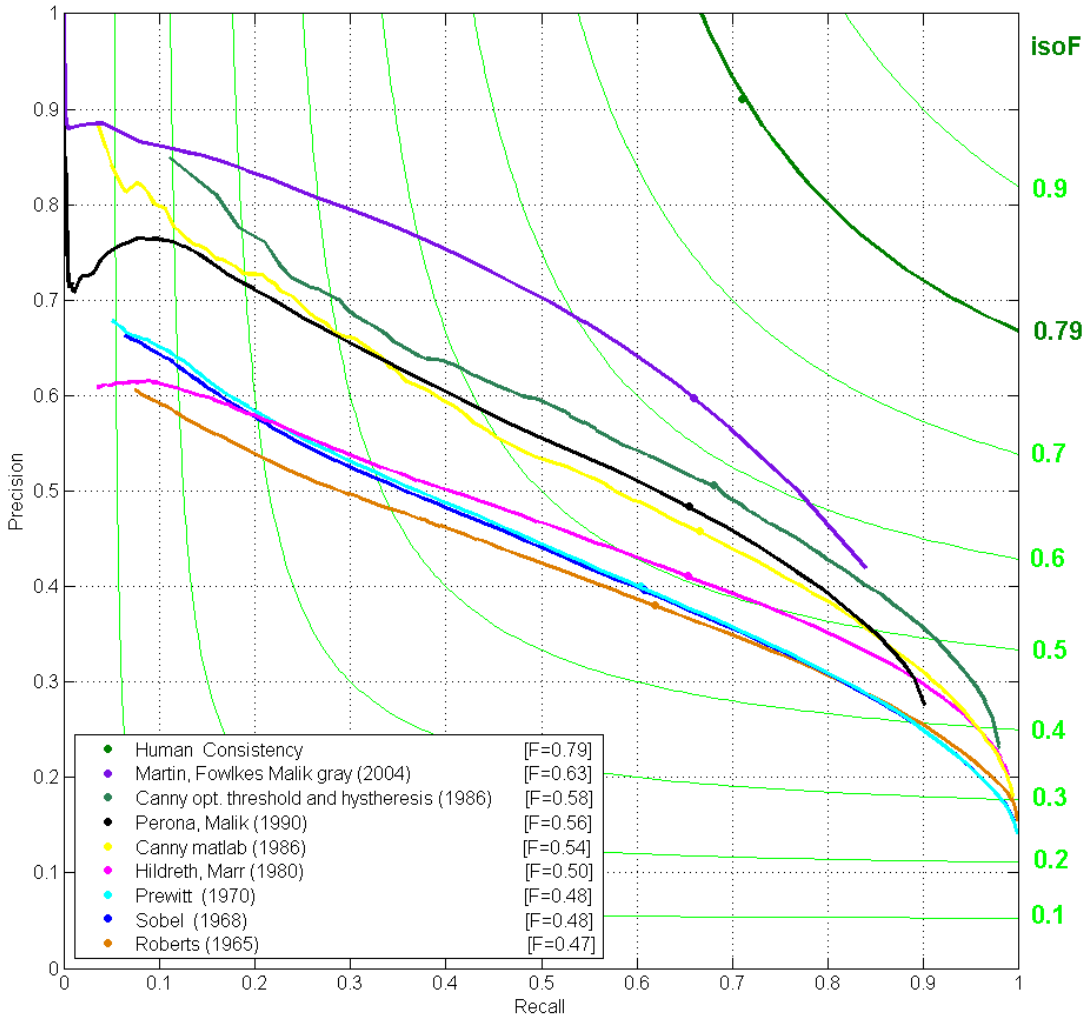
Contour detection ~1970



Contour detection ~1990

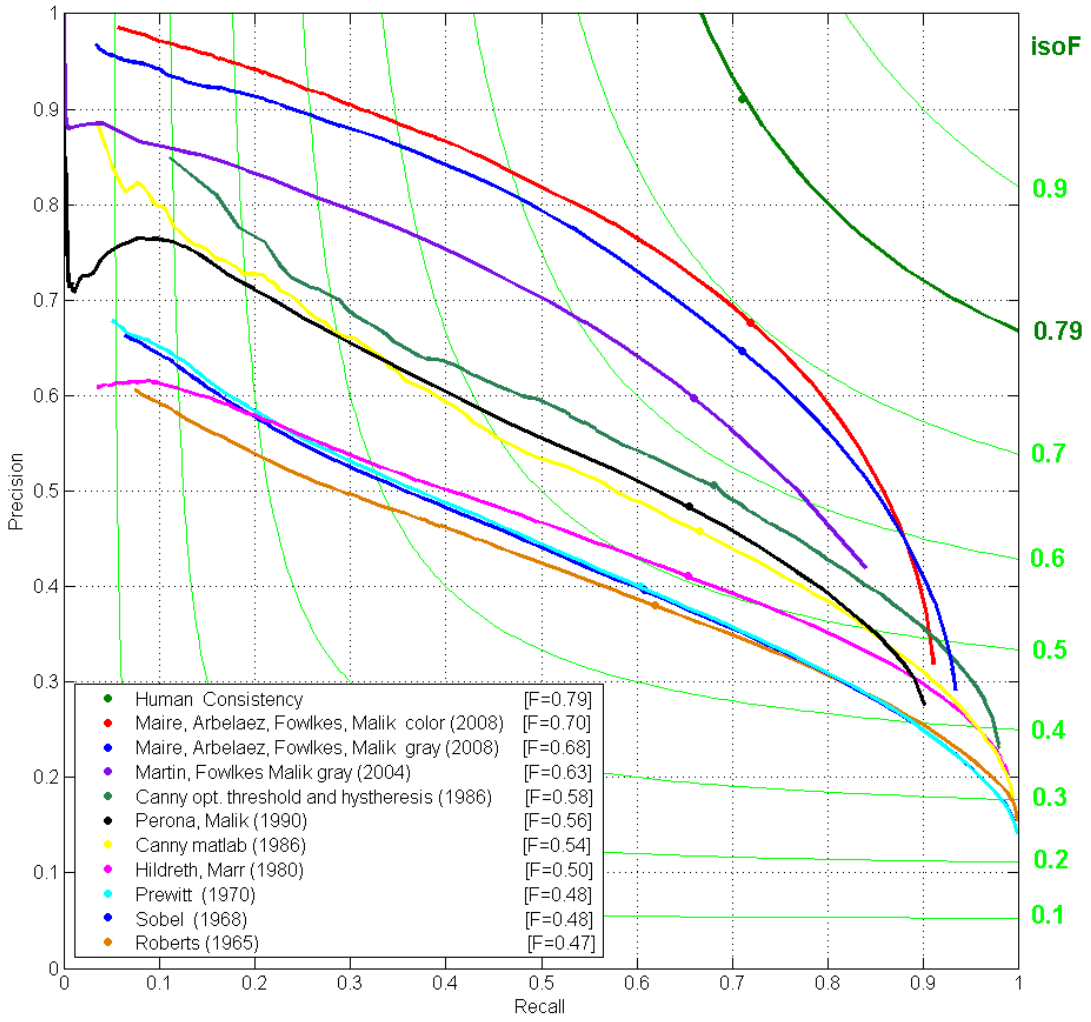


Contour detection ~2004

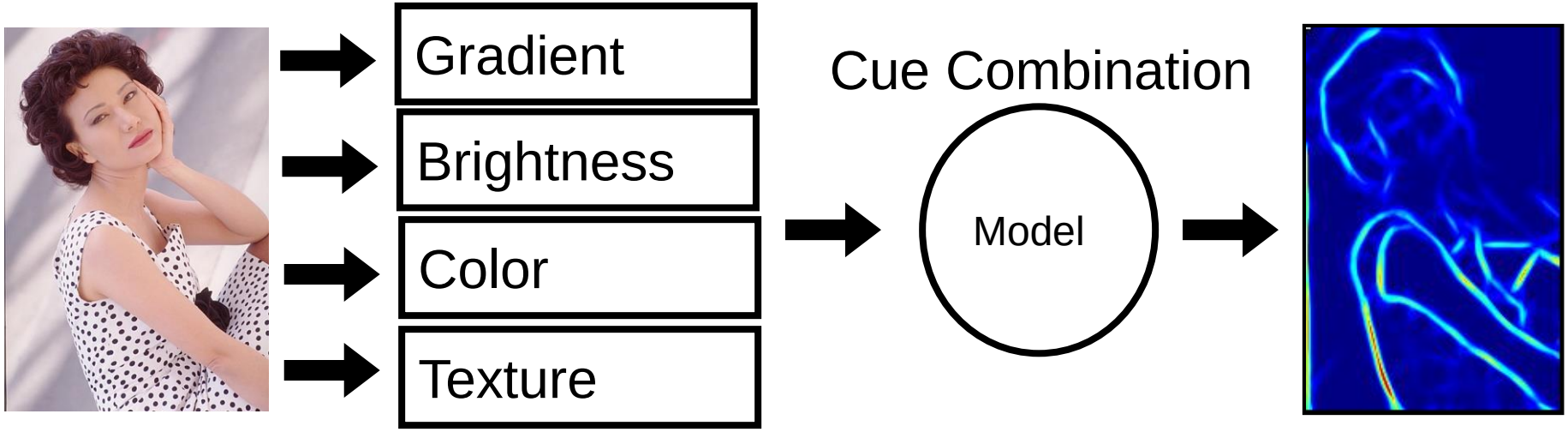


Contour detection ~2008

Include global cues



Martin, Fowlkes, Malik PAMI 04



- Goal: learn the posterior probability of a boundary $P_b(x,y,\theta)$ from local information only

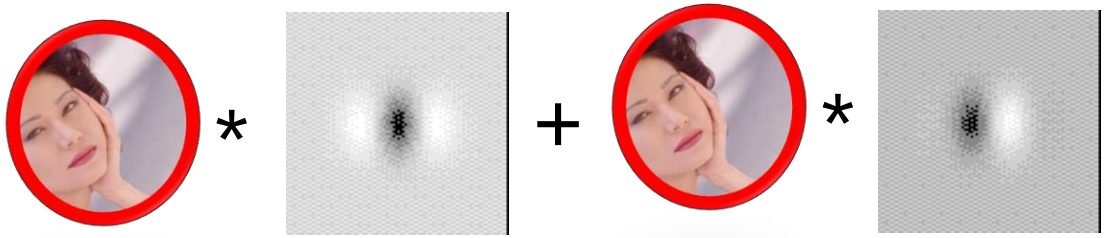
- Challenges:
  - computing the cues,
  - cue combination

Pb: gradient cue

Image

OE =

Oriented Energy

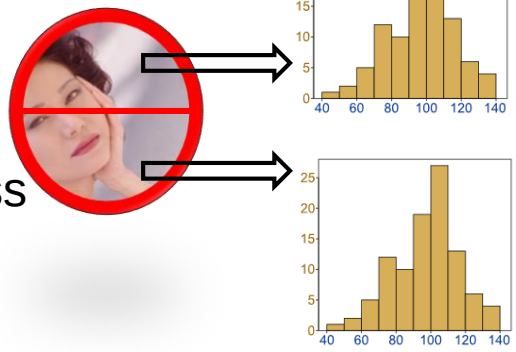


Pb: brightness and color cues

Image

BG =

Brightness Gradient

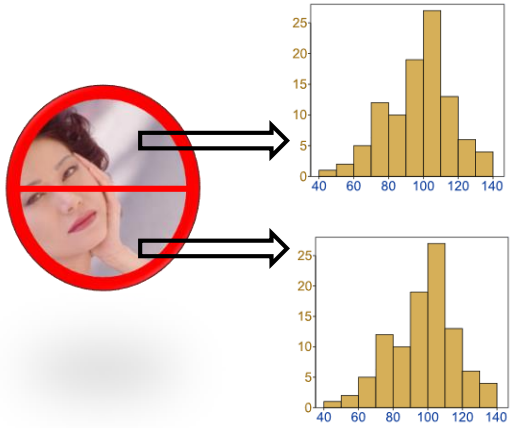


Difference of L* distributions

$BG = \chi^2(h1(L),h2(L))$

CG =

Color Gradient



Difference of a* distributions

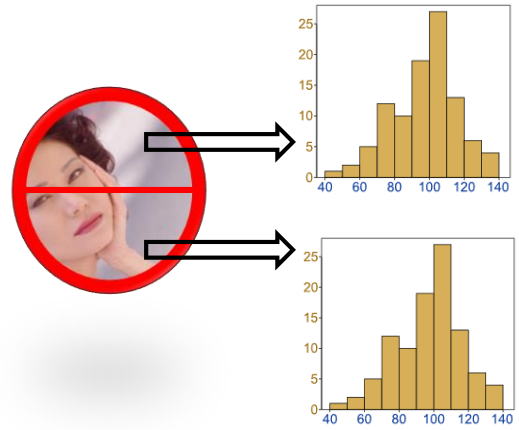
$CG = \chi^2(h1(a),h2(a)) + \chi^2(h1(b),h2(b))$

Pb: texture cue

Image

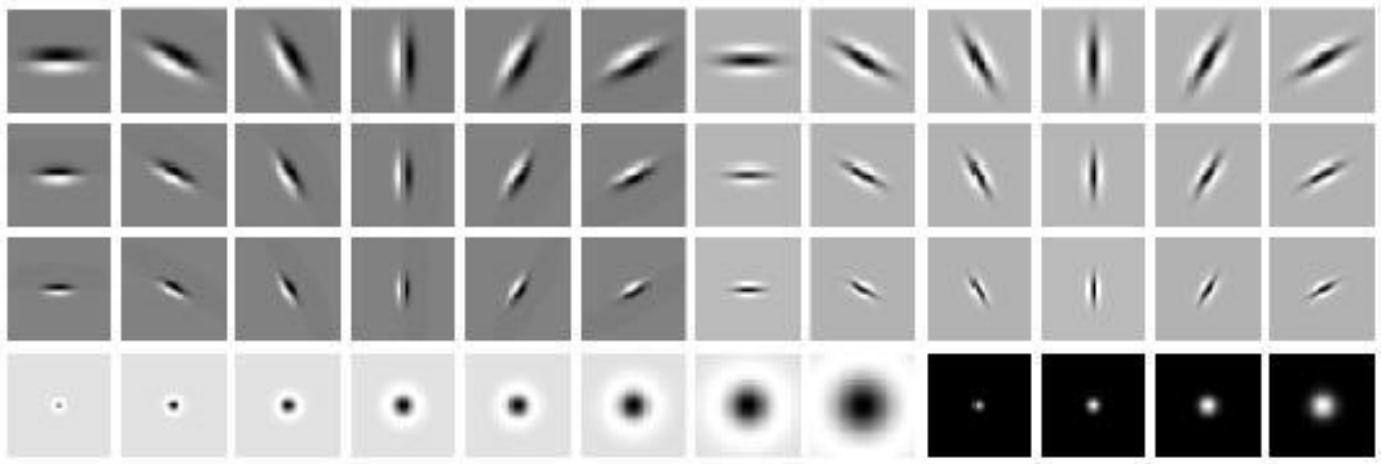
TG =

Texture Gradient



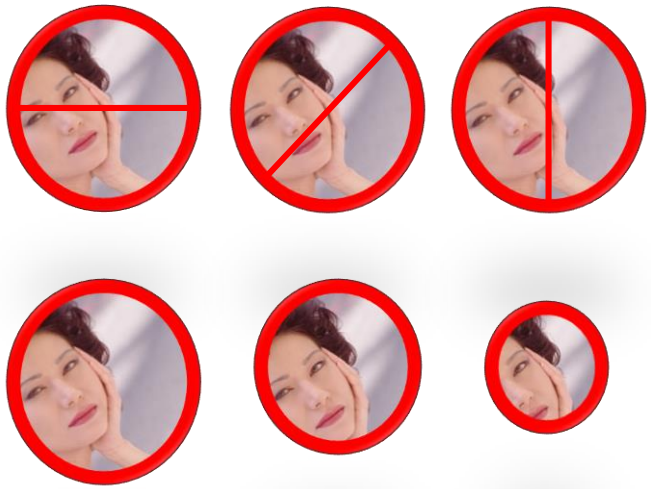
Difference of filter distributions

$TG=\sum_f \chi^2(h1(f),h2(f))$



Pb: cue design

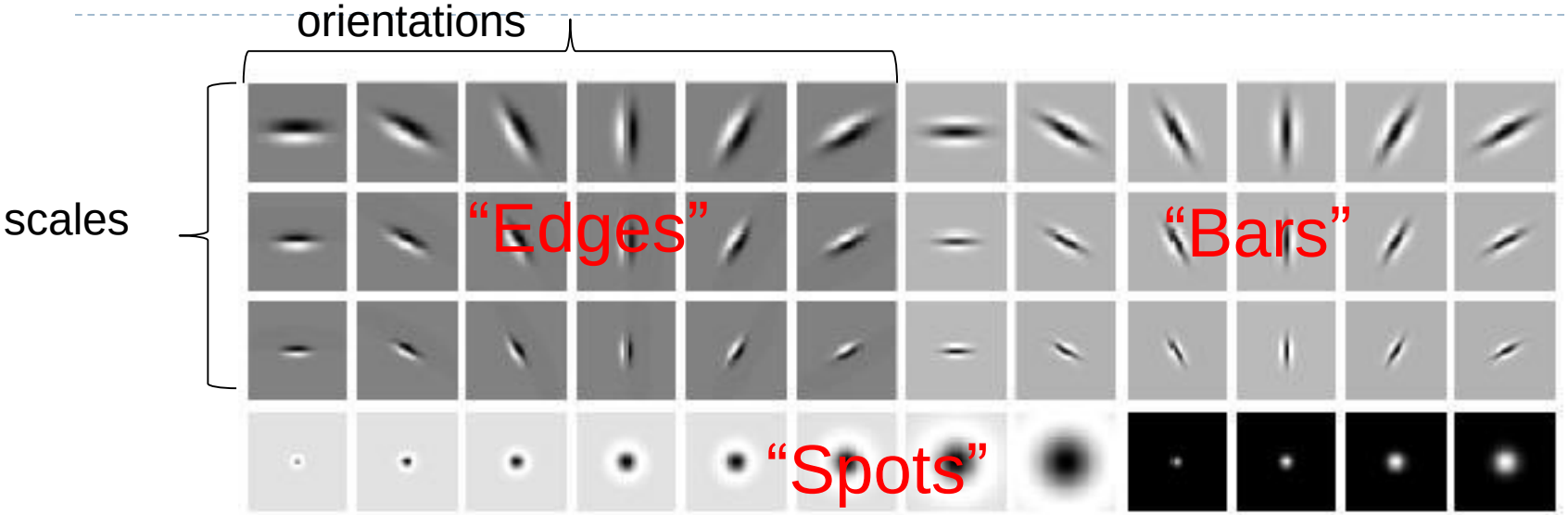
Image



Multiple orientations

Multiple disk radii

Filter banks



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

Matlab code available for these examples:
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Filter bank

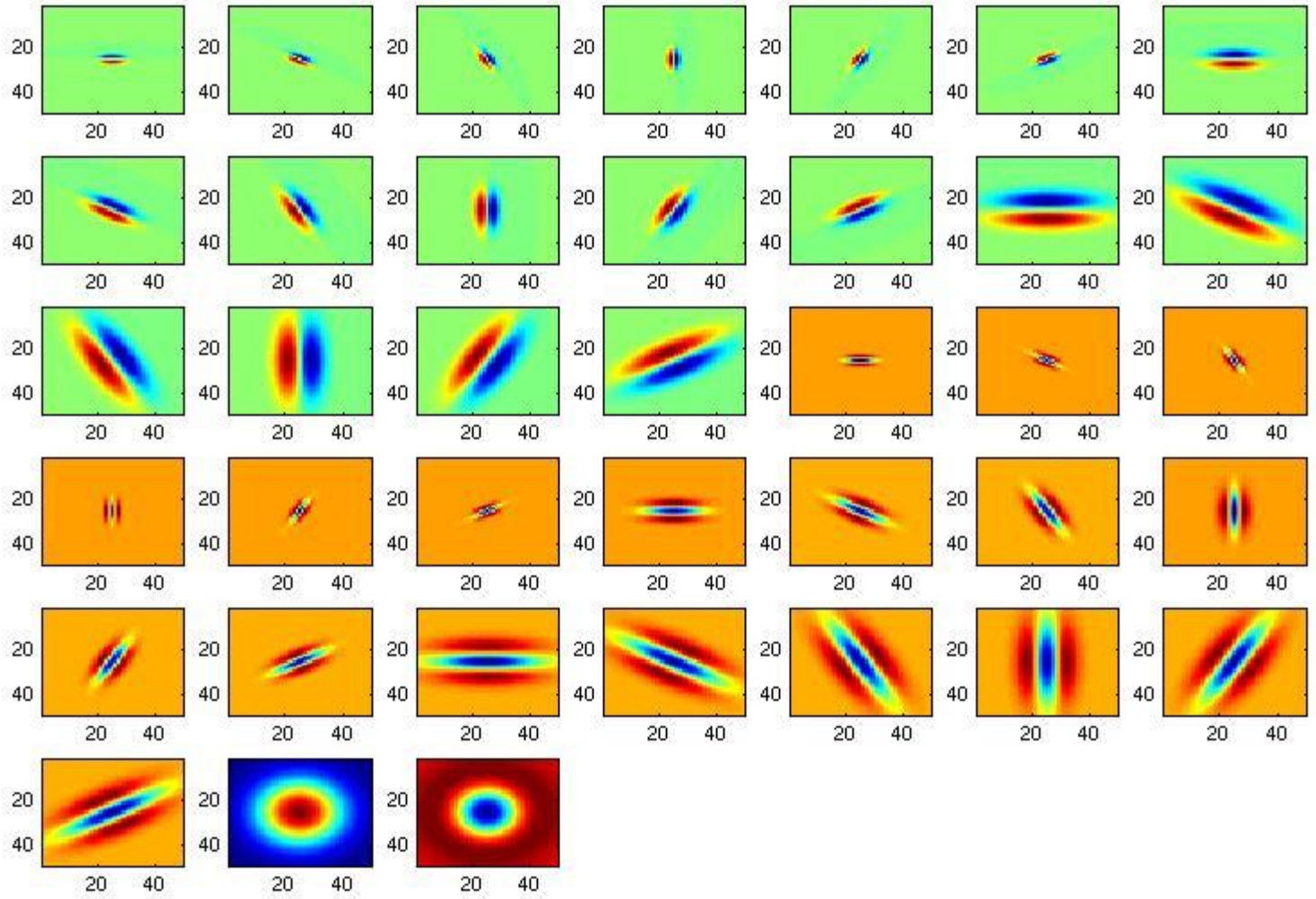
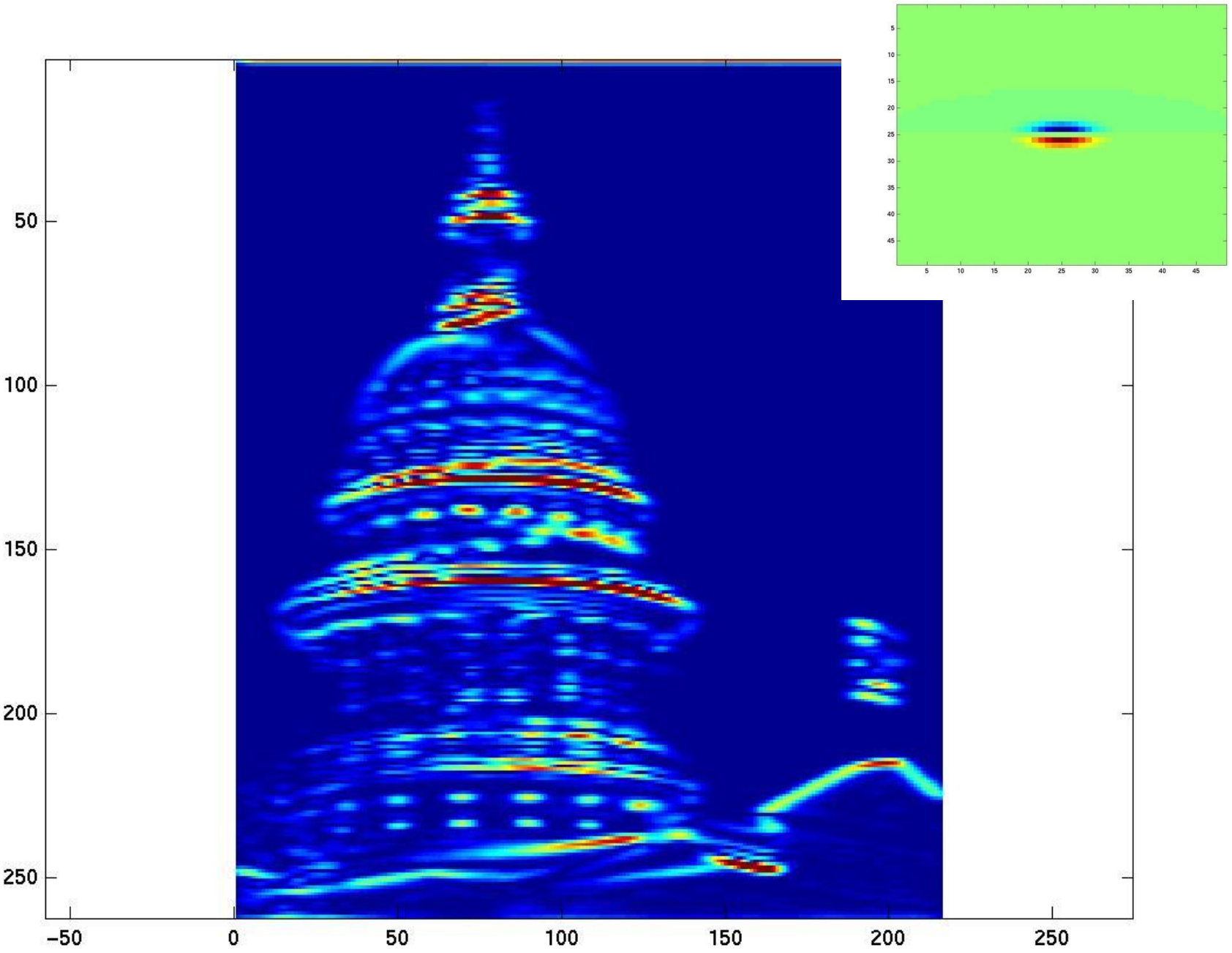
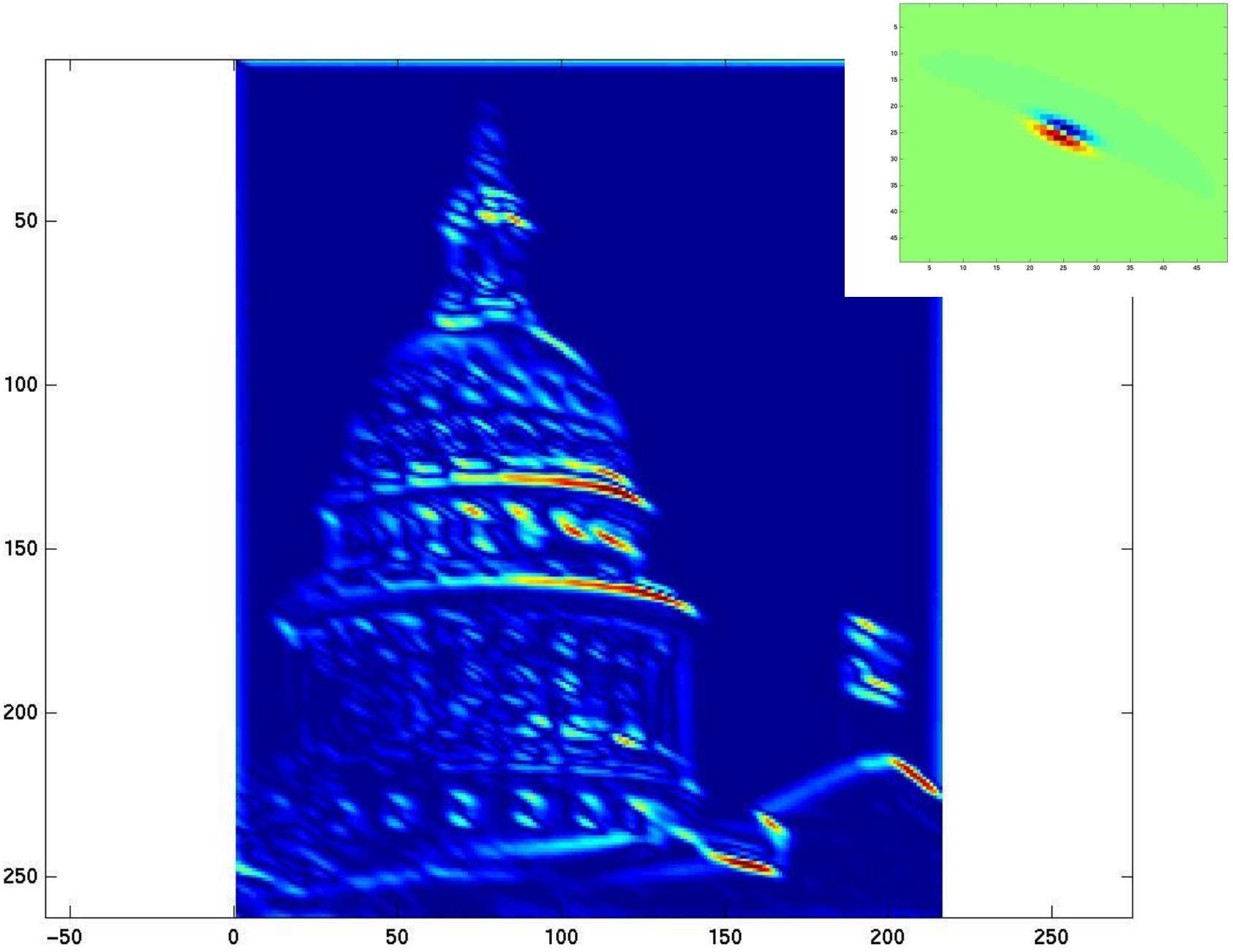
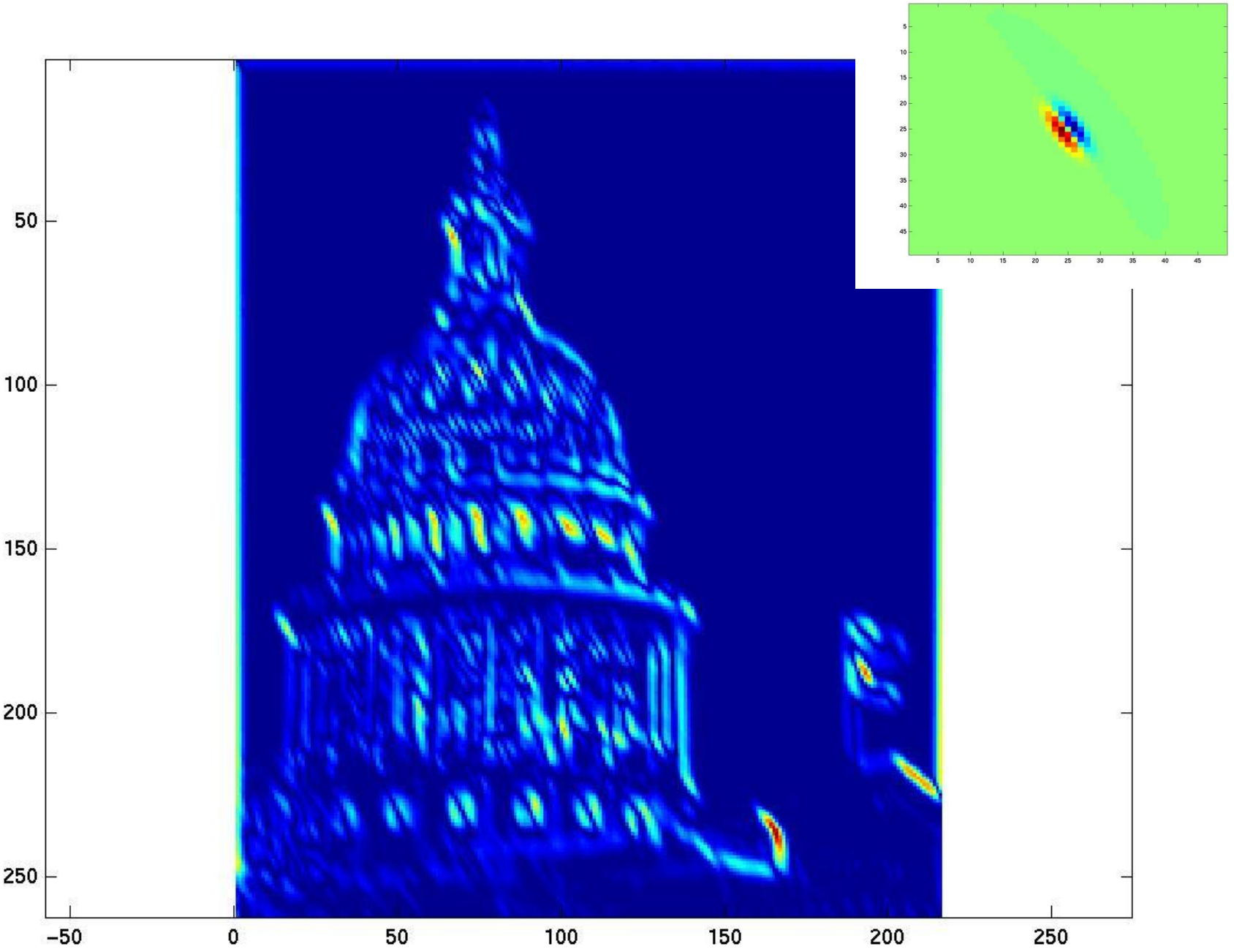
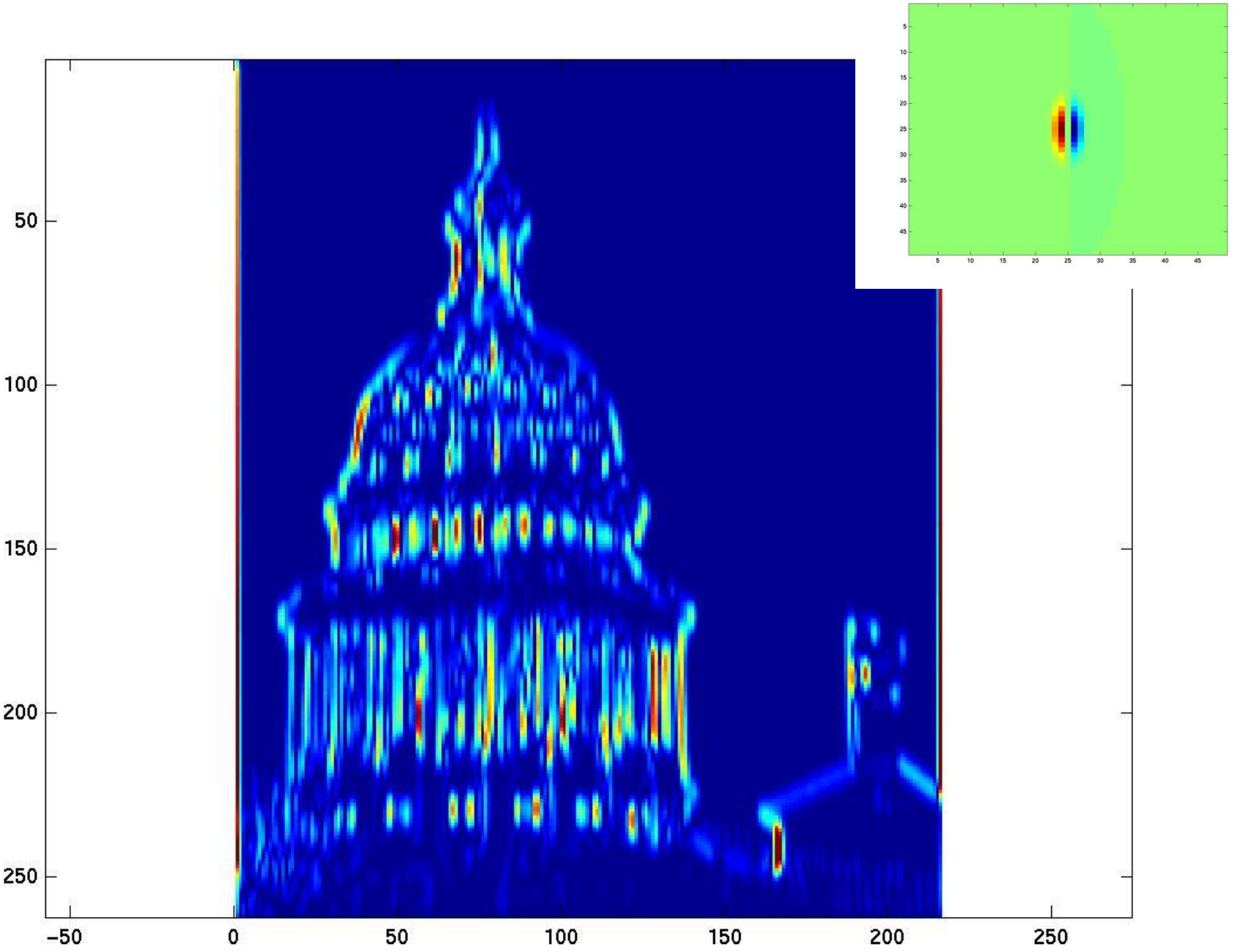


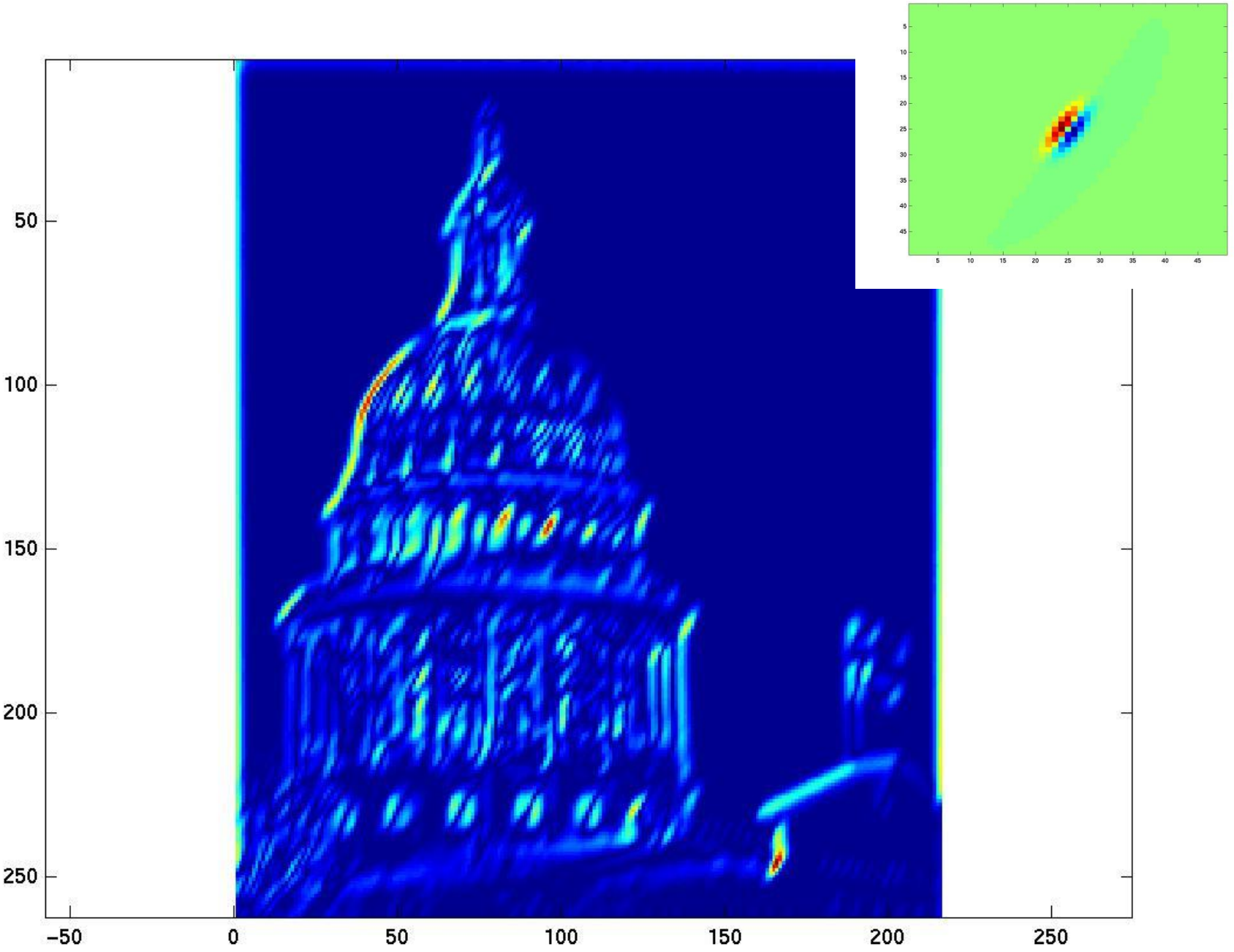
Image from http://www.texasexplorer.com/austincap2.jpg

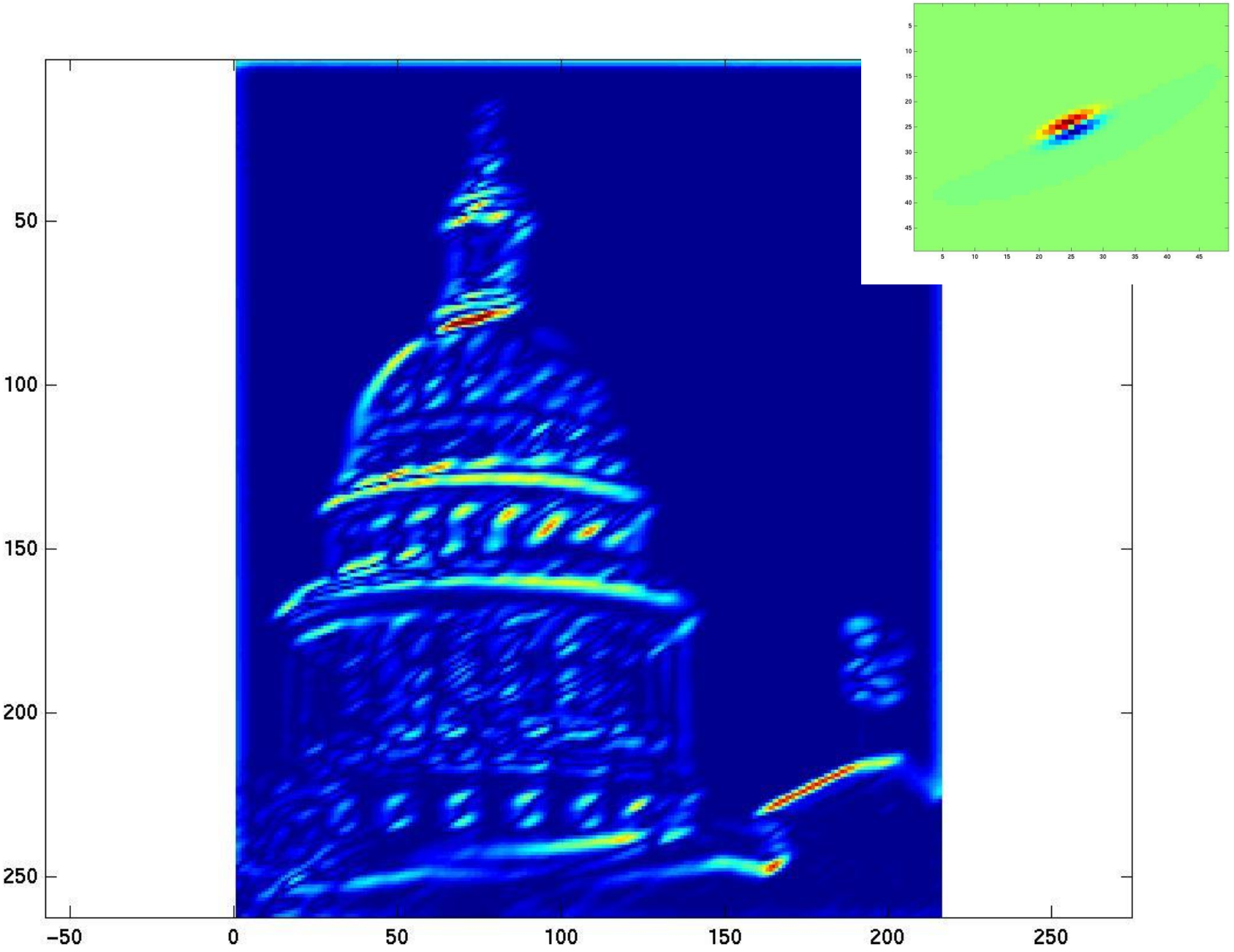

50
100
150
200
250
−50
0
50
100
150
200
250

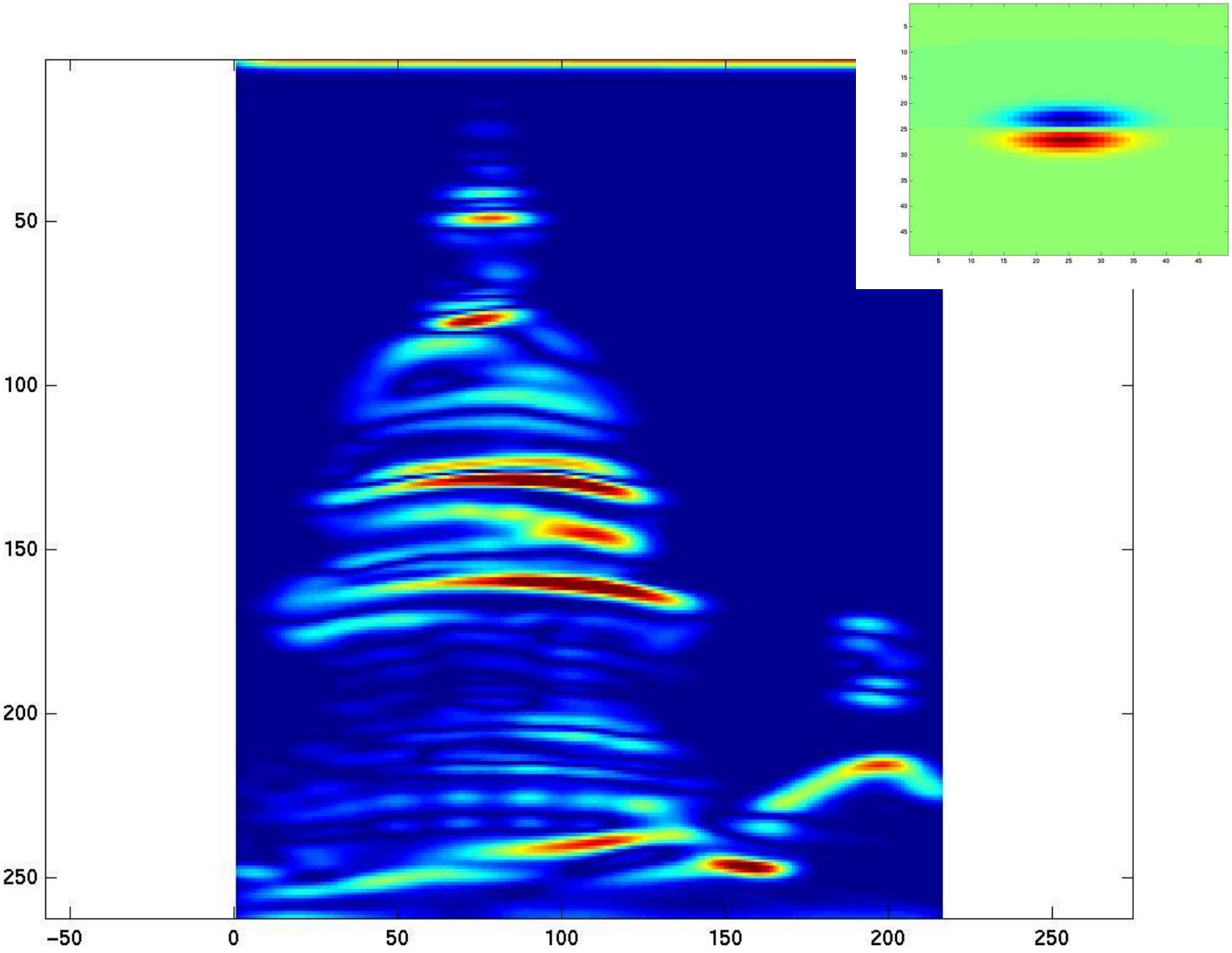

5
10
15
20
25
30
35
40
45
5
10
15
20
25
30
35
40
45
50
100
150
200
250
−50
0
50
100
150
200
250

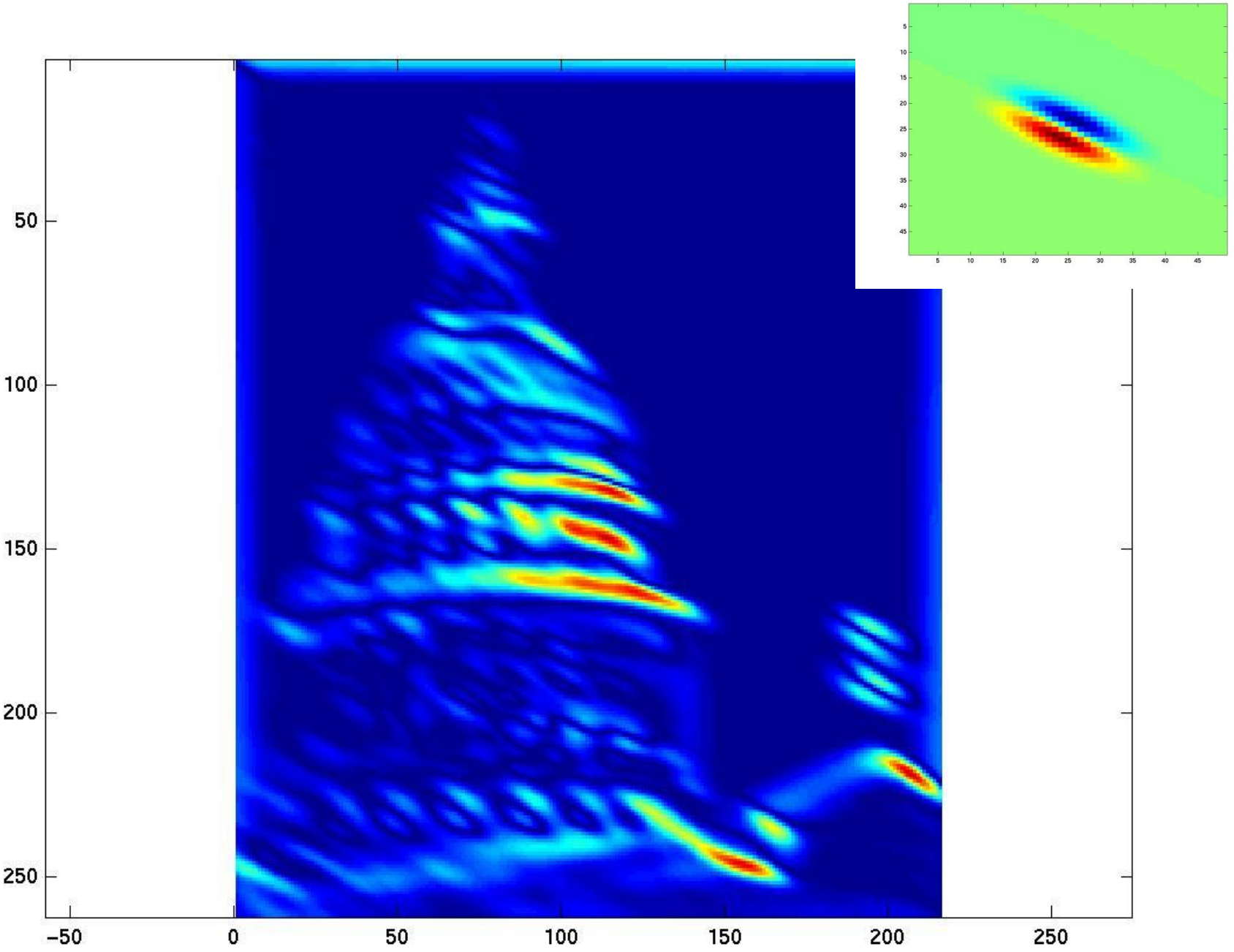

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

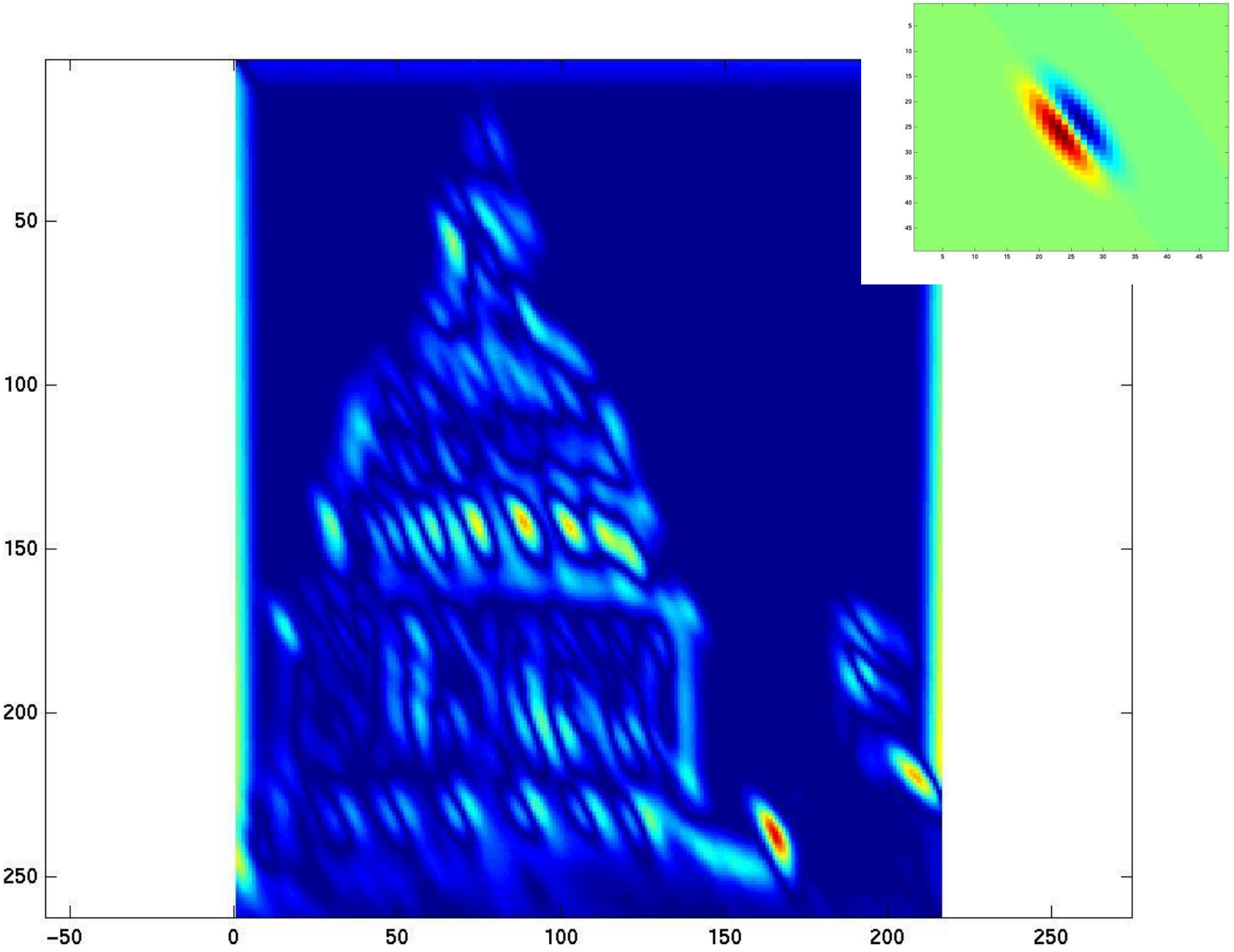

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

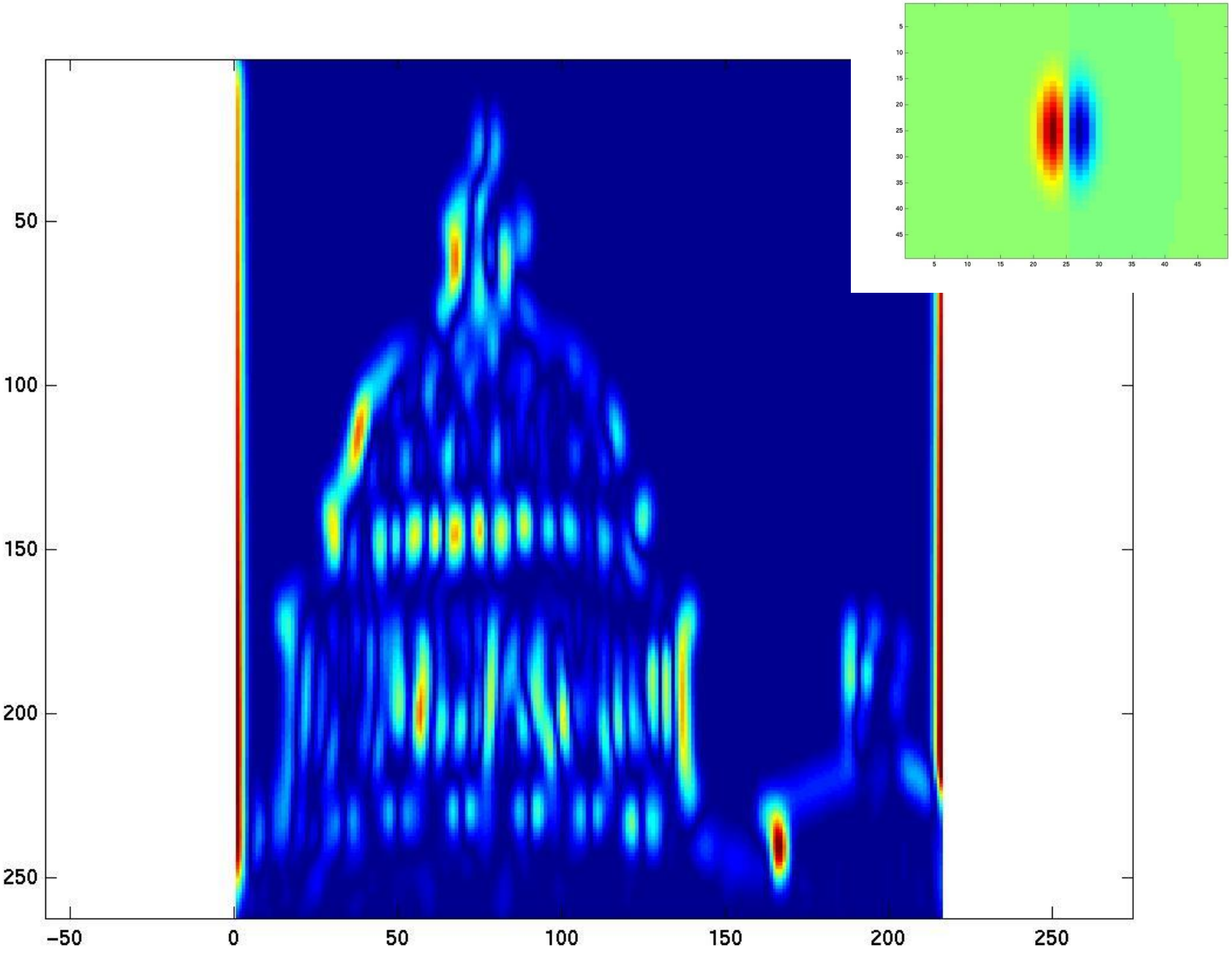

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

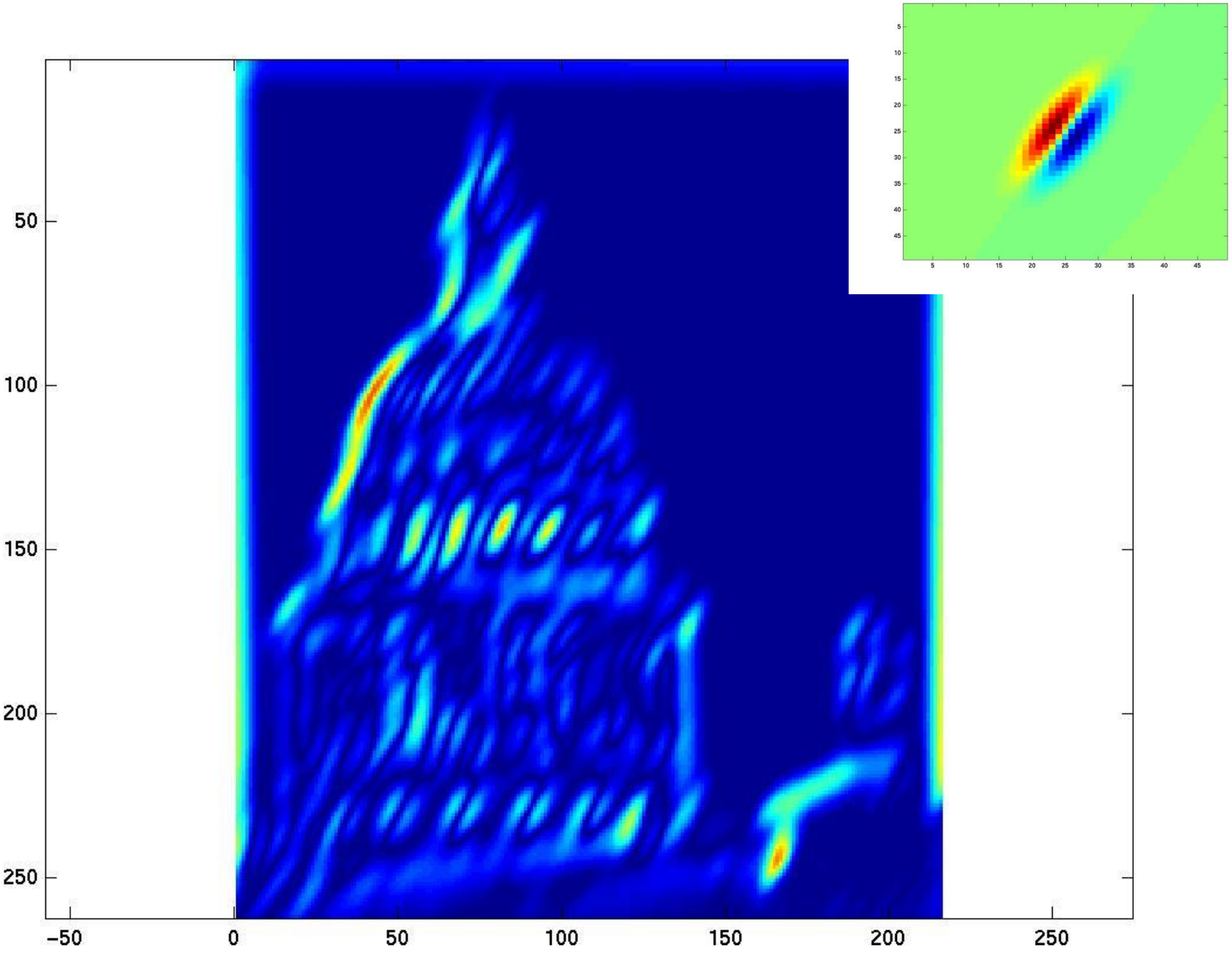

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

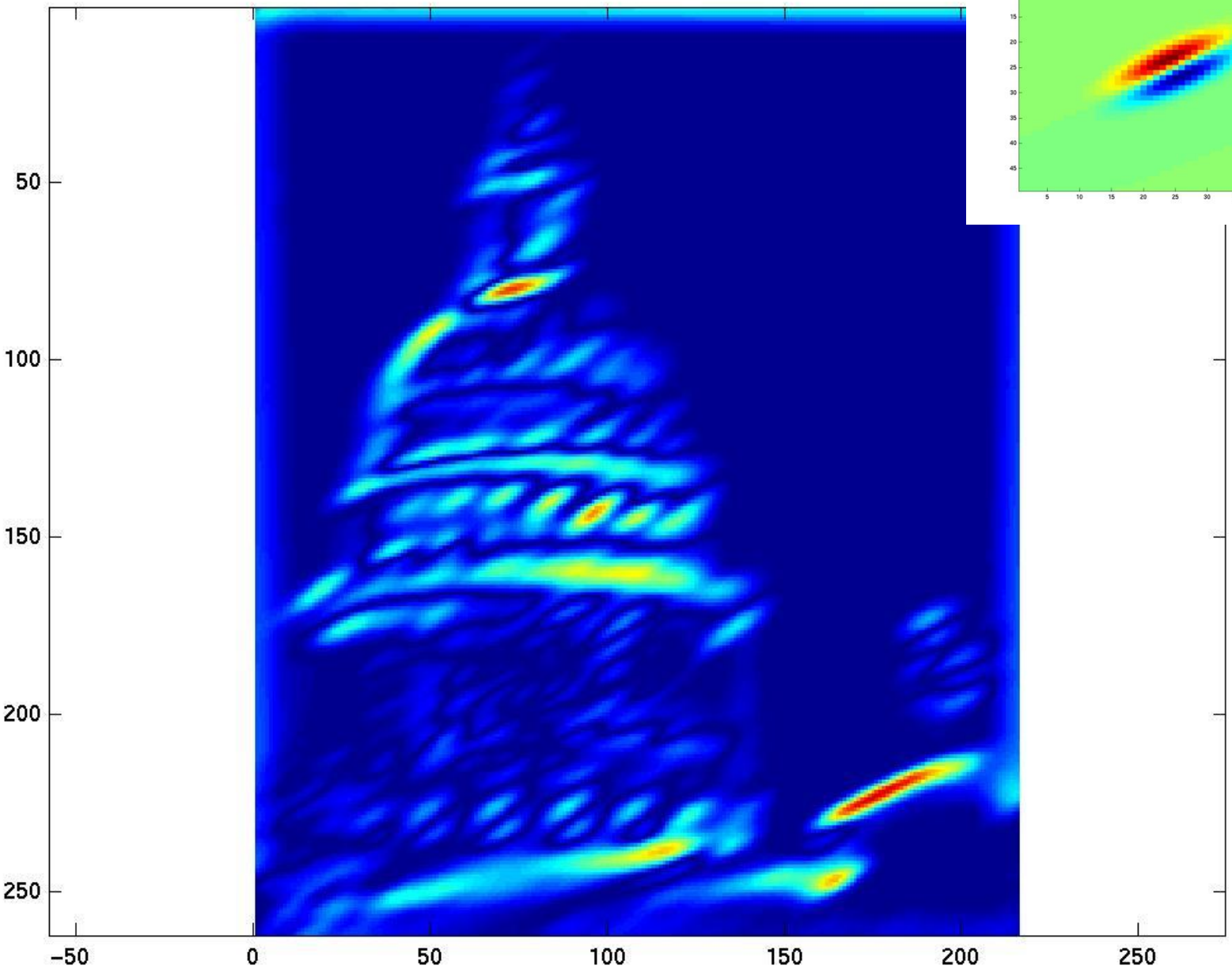

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

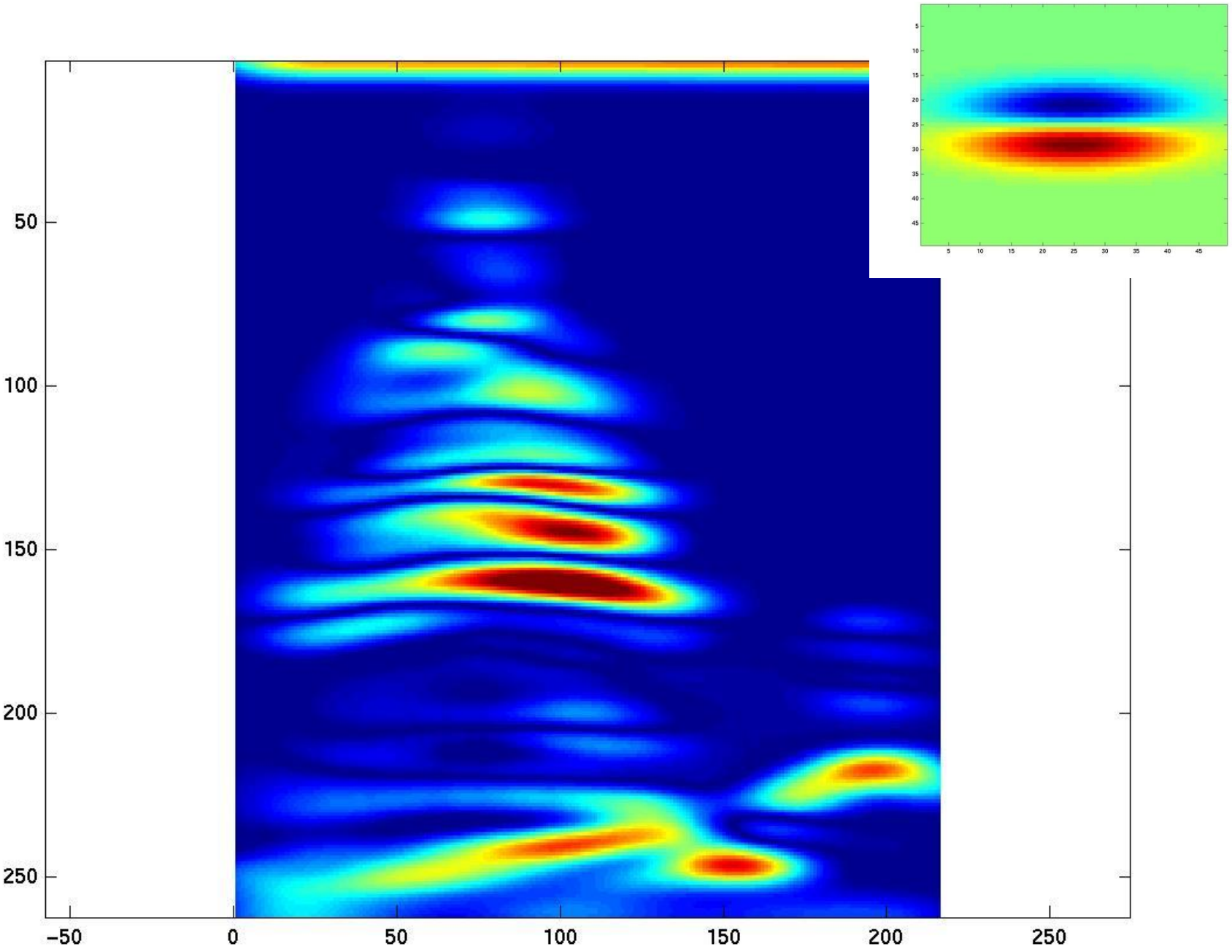

5
10
15
20
25
30
35
40
45
5
10
15
20
25
30
35
40
45
50
100
150
200
250
−50
0
50
100
150
200
250

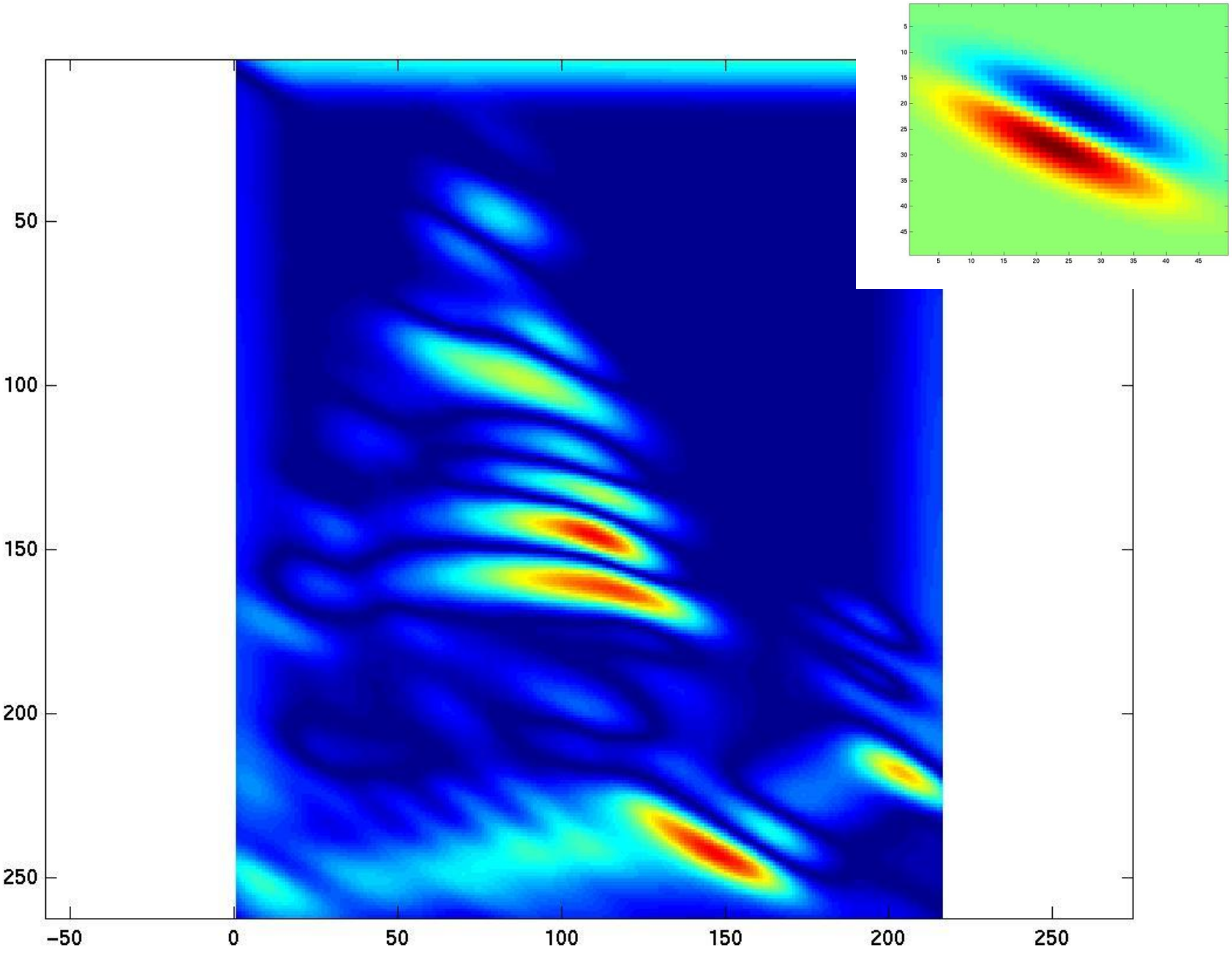

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

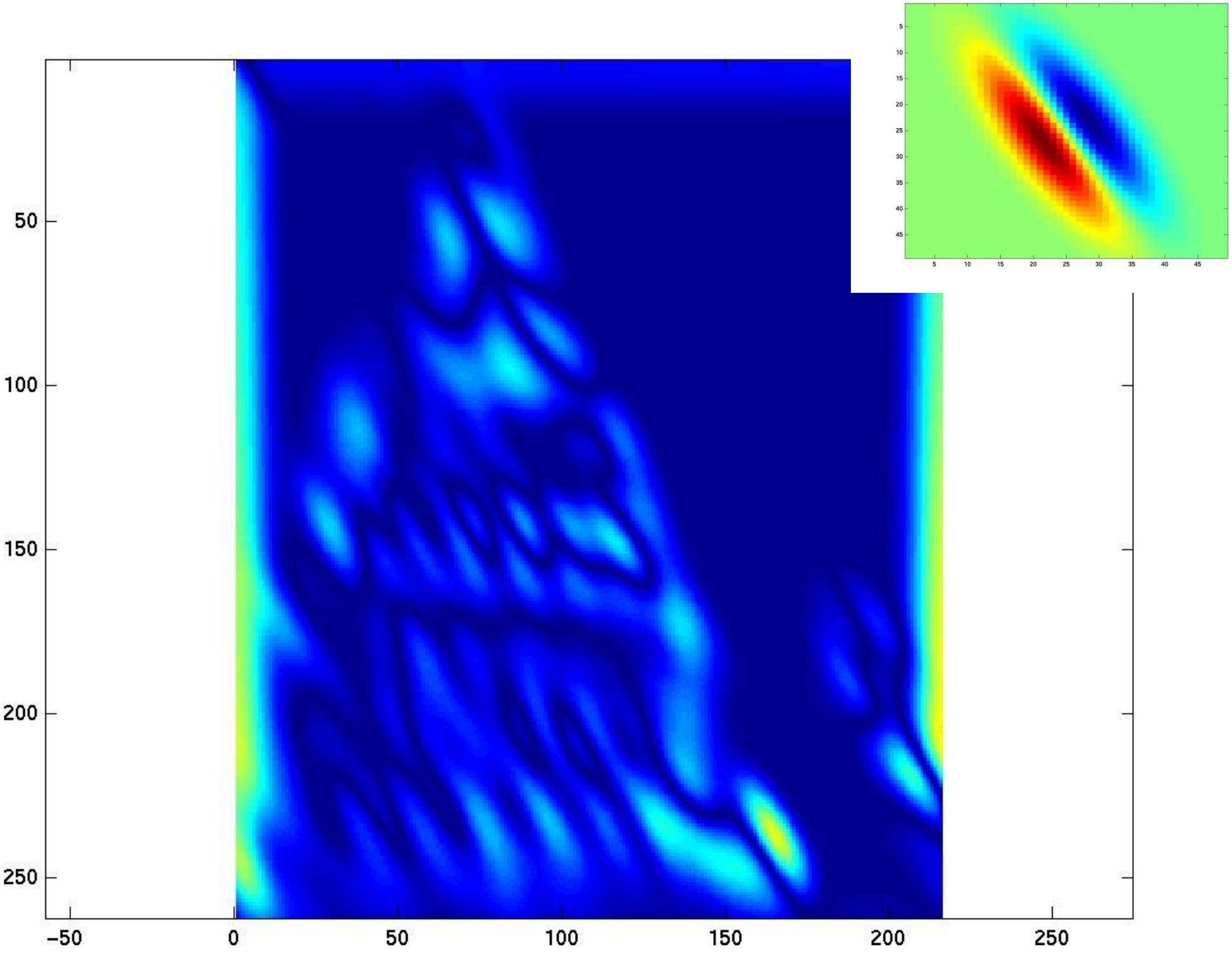

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

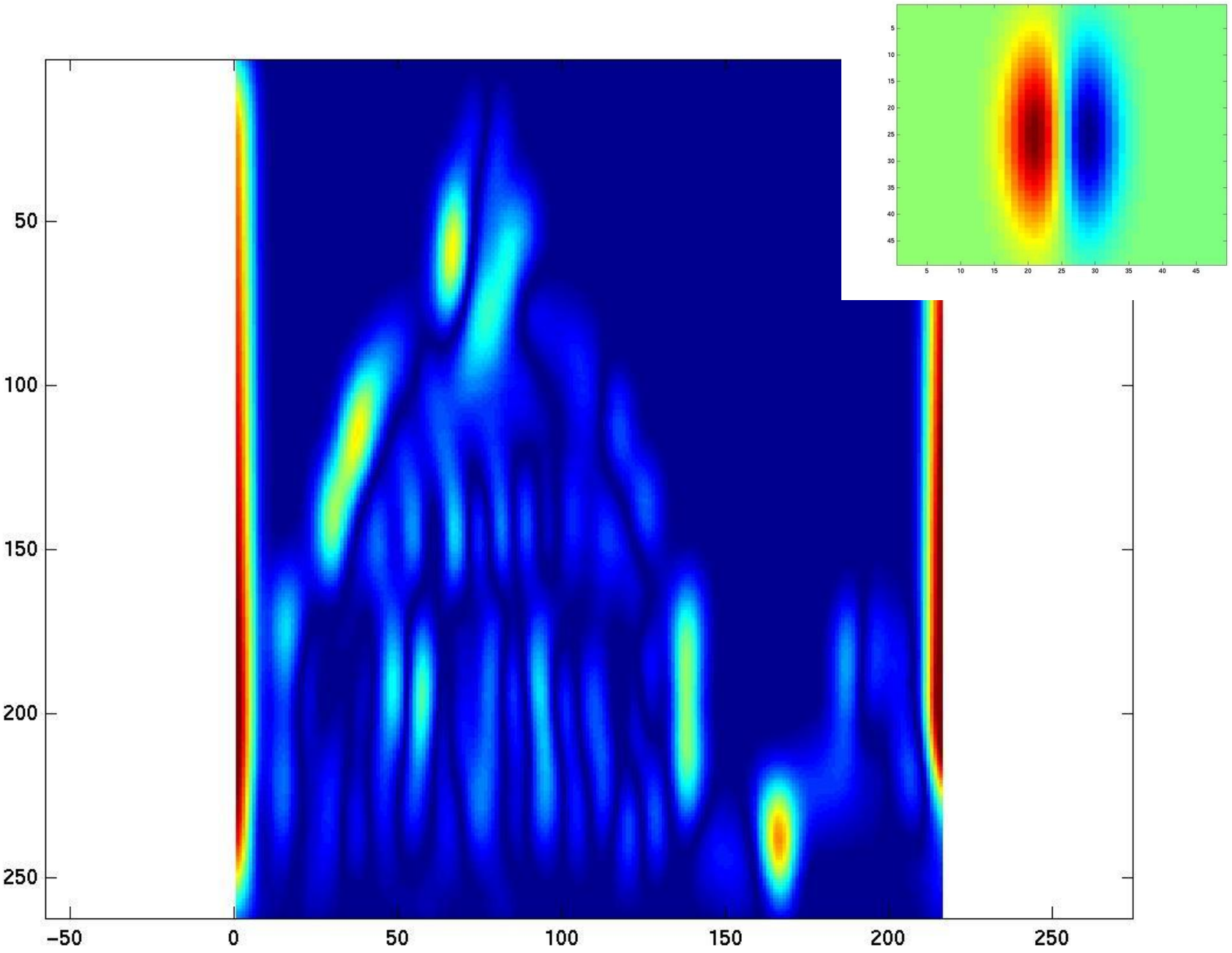

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

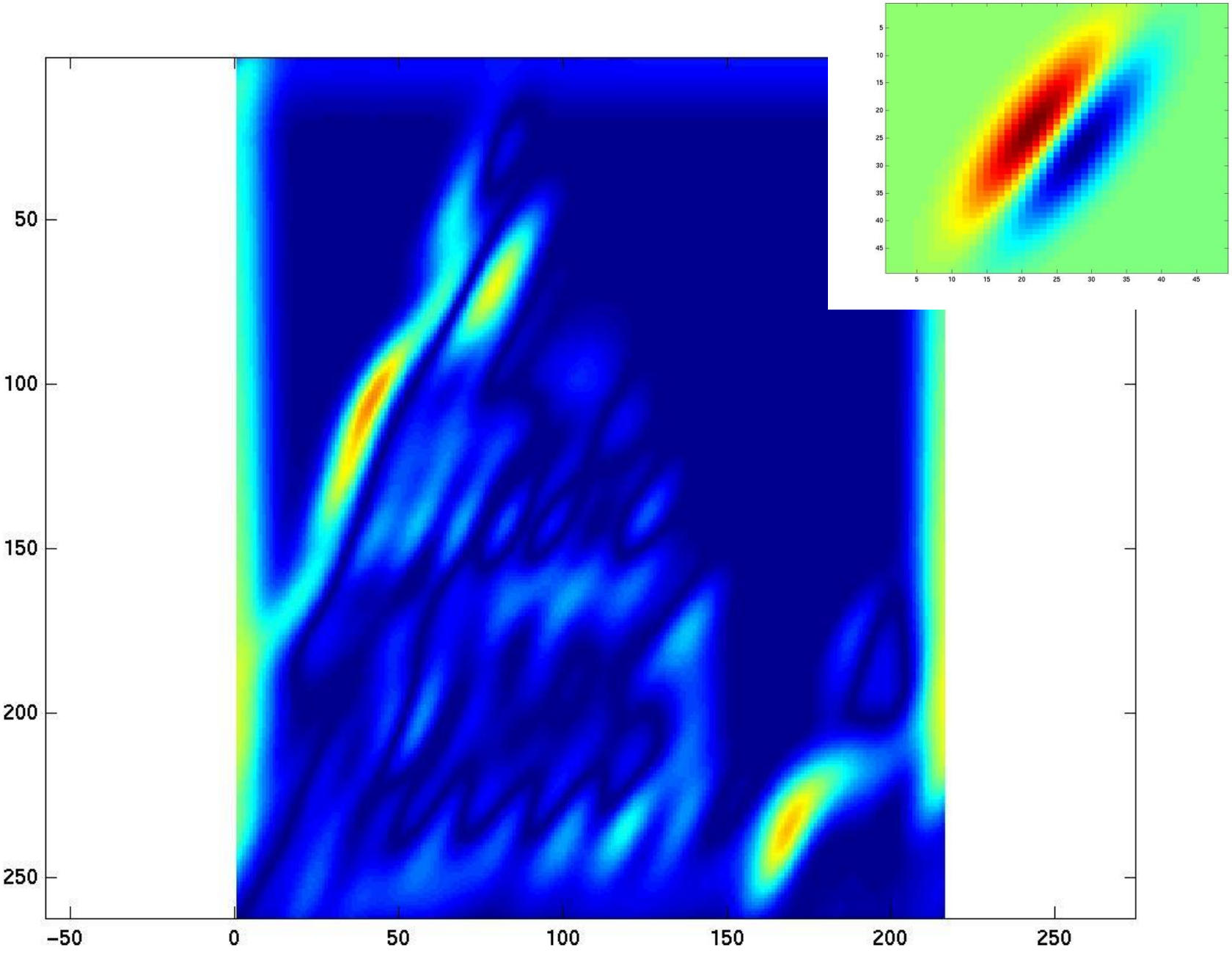

50
100
150
200
250
−50
0
50
100
150
200
250

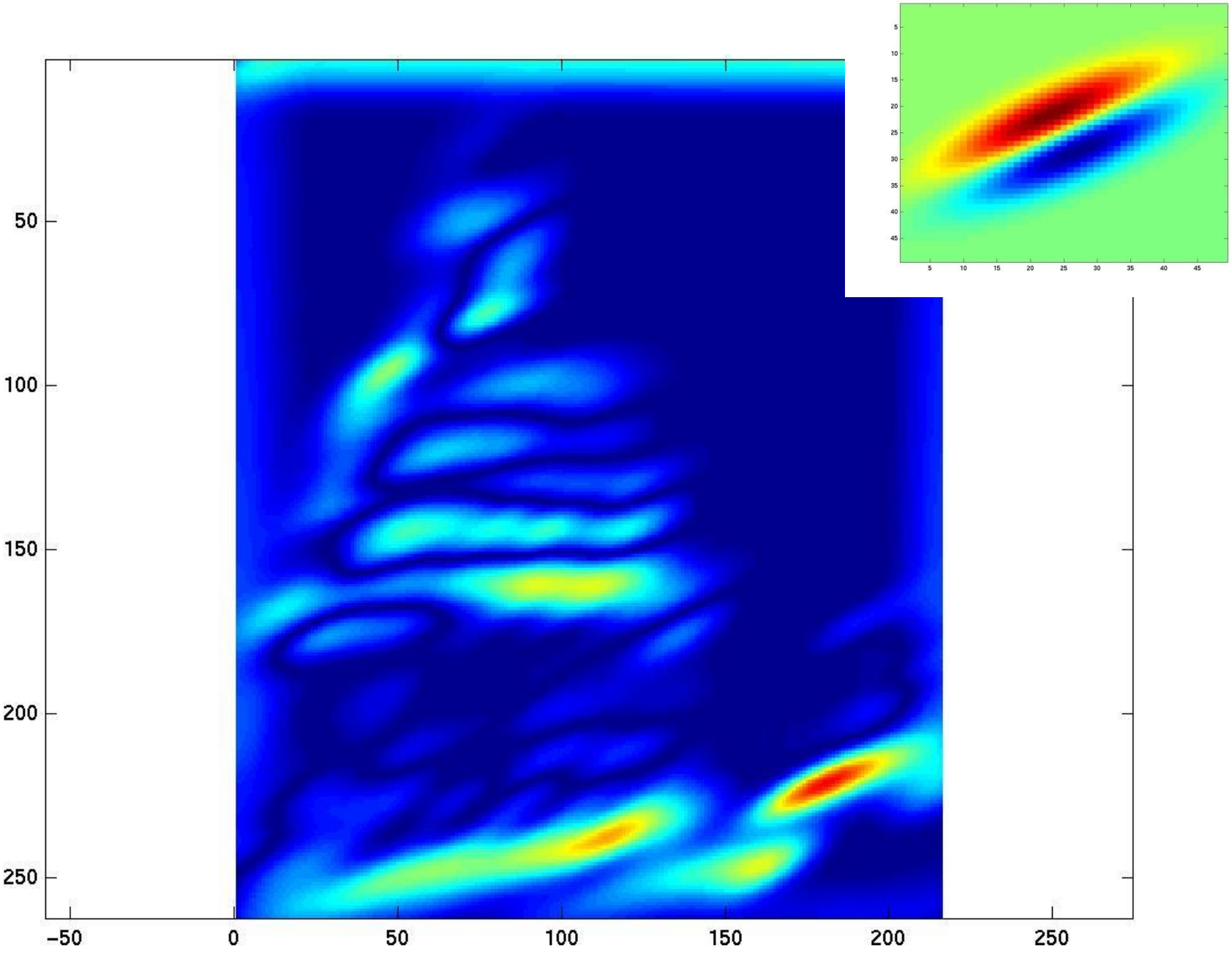

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

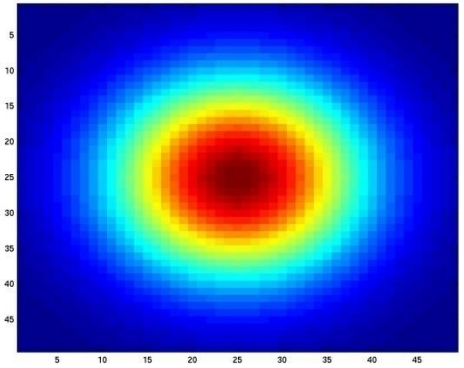

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45


5
10
15
20
25
30
35
40
45
5
10
15
20
25
30
35
40
45
50
100
150
200
250
−50
0
50
100
150
200
250

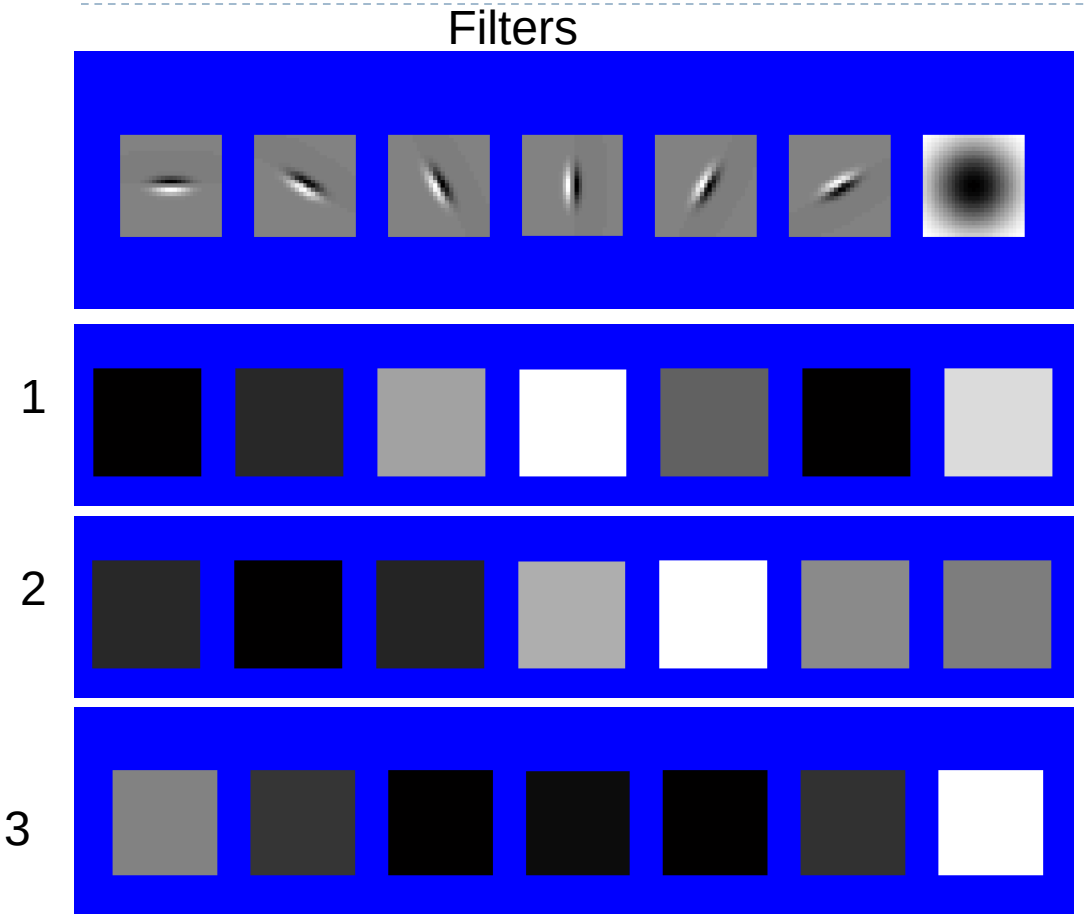

50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

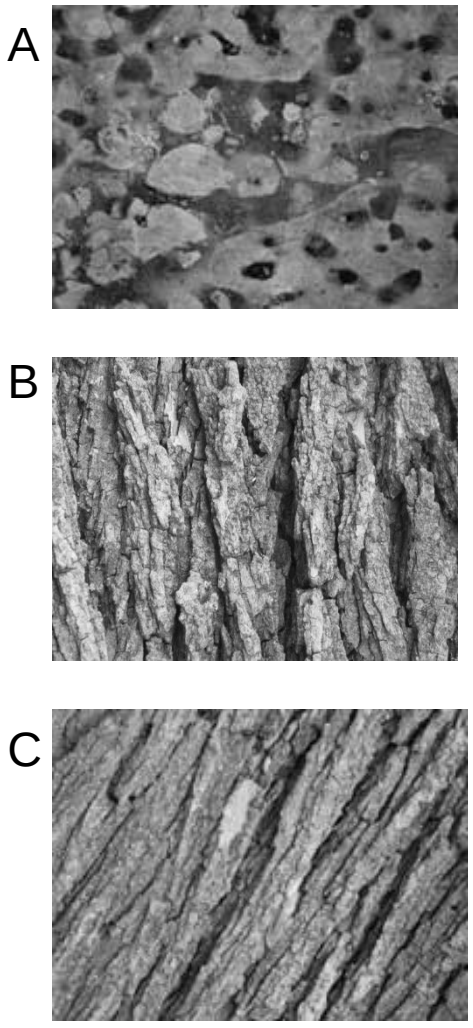

5
10
15
20
25
30
35
40
45
5
10
15
20
25
30
35
40
45
50
100
150
200
250
−50
0
50
100
150
200
250


50
100
150
200
250
−50
0
50
100
150
200
250
5
10
15
20
25
30
35
40
45

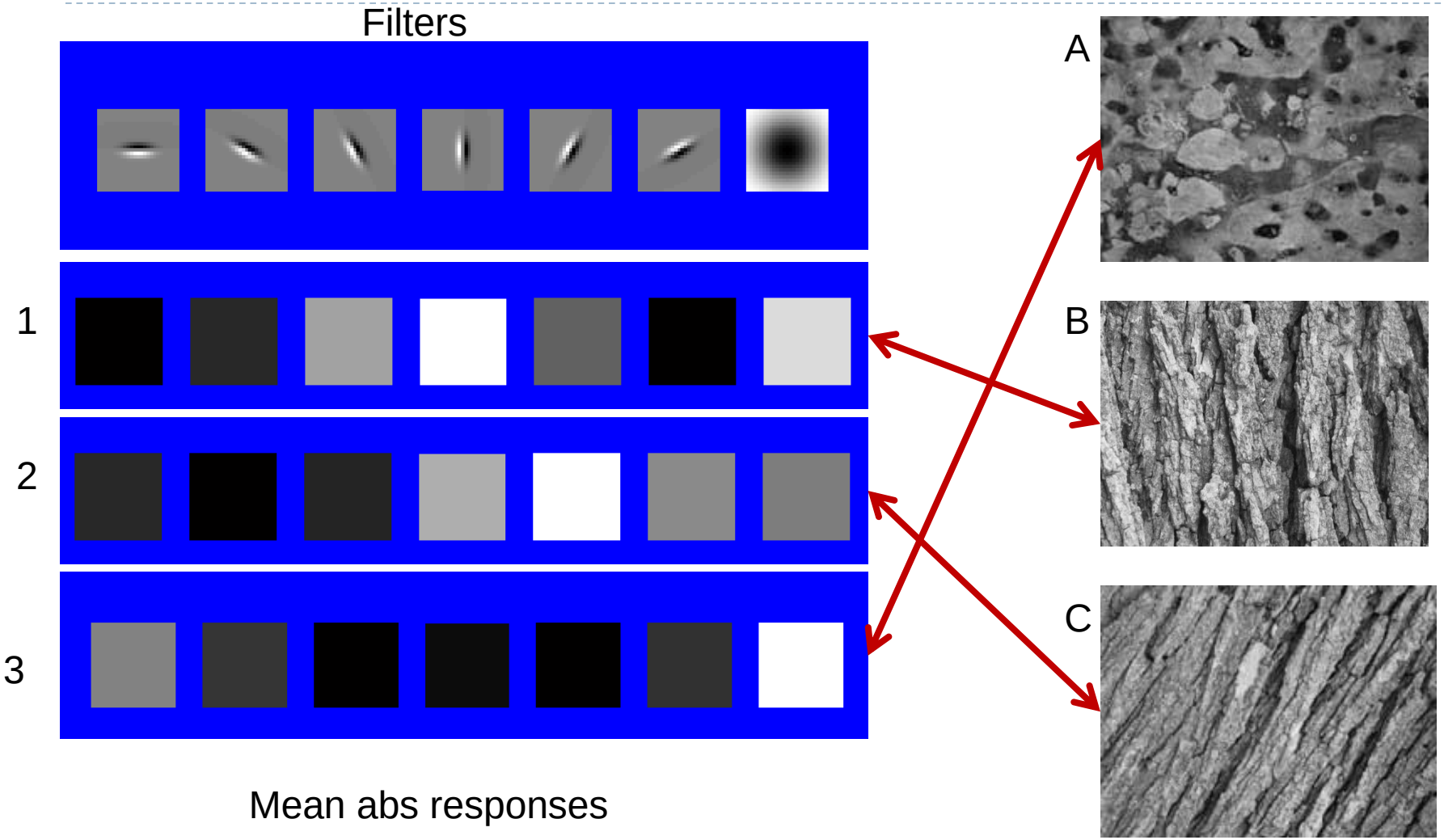

5
10
15
20
25
30
35
40
45
5
10
15
20
25
30
35
40
45

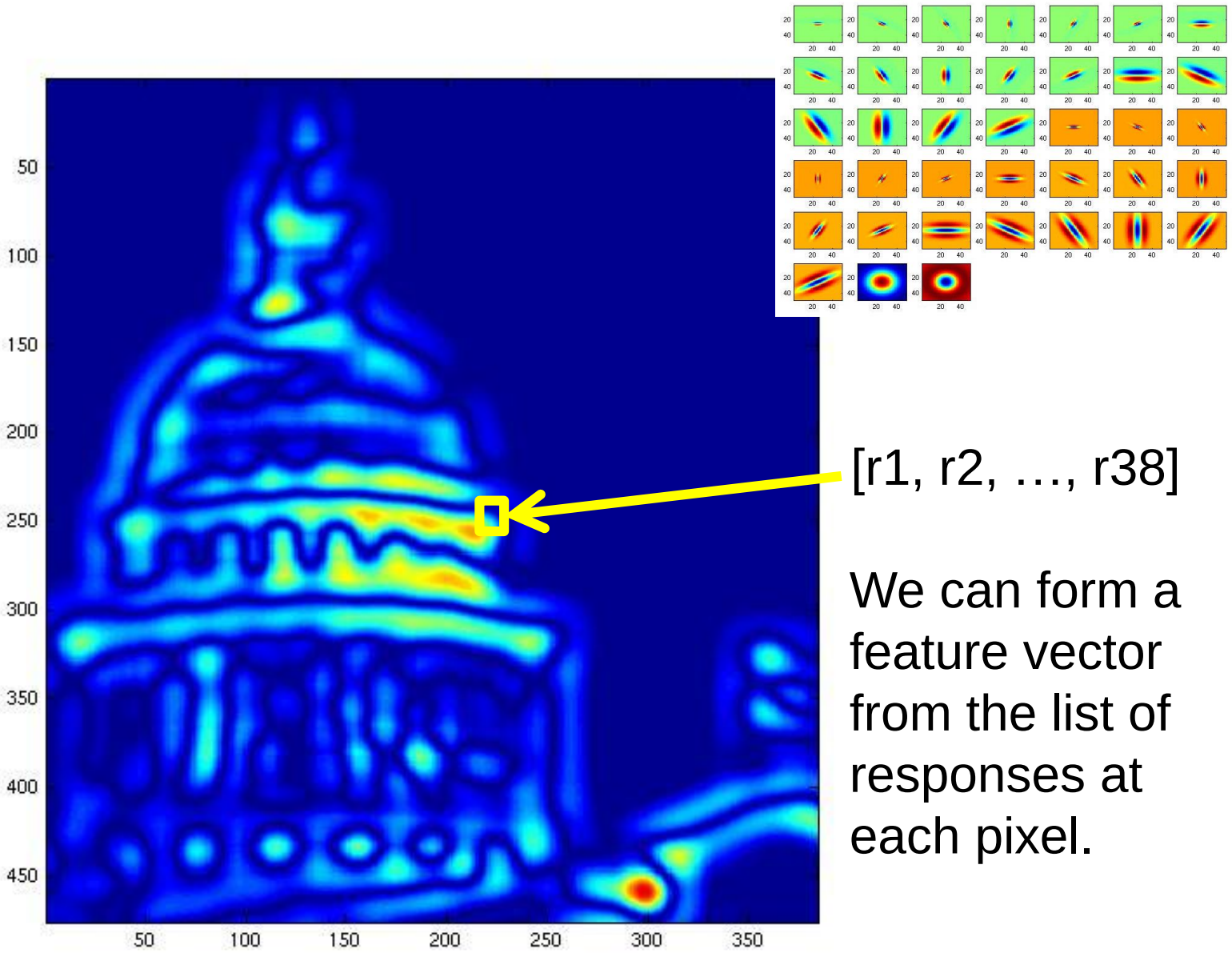
You try: Can you match the texture to the response?



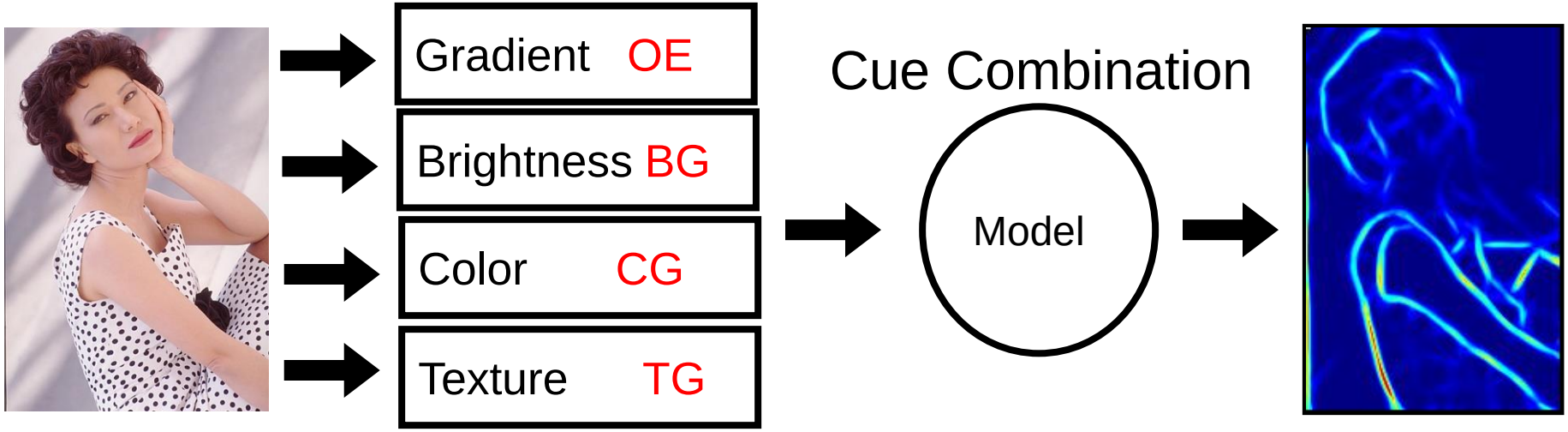


You try: Can you match the texture to the response?




50
100
150
200
250
300
350
400
450
50
100
150
200
250
300
350
[r1, r2, …, r38]
We can form a feature vector from the list of responses at each pixel.

Martin, Fowlkes, Malik PAMI 04



- Goal: learn the posterior probability of a boundary $P_b(x,y,\theta)$ from local information only

- Challenges:
  - computing the cues,
  - cue combination

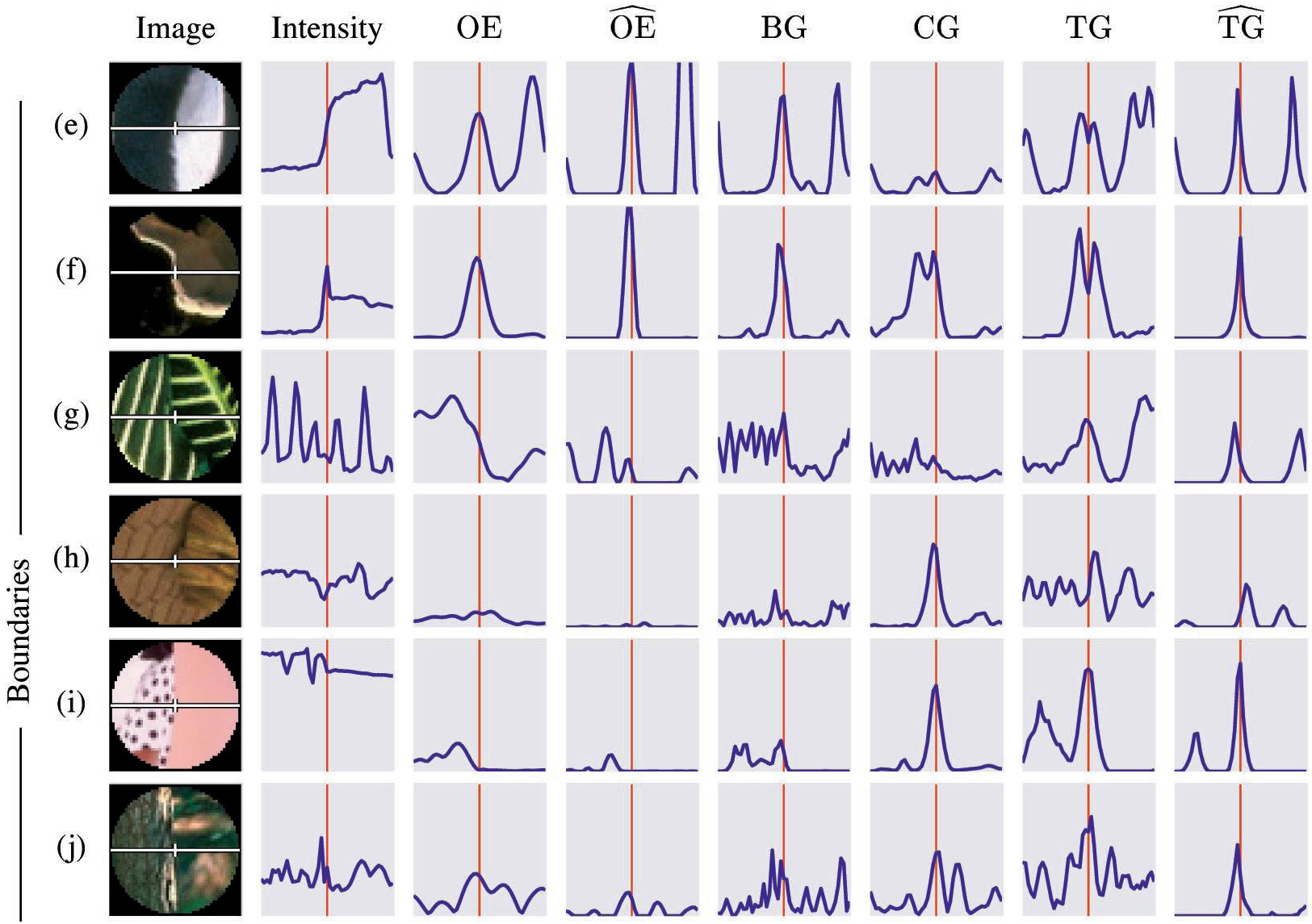


Image
Intensity
OE
$\widehat{\text{OE}}$
BG
CG
TG
$\widehat{\text{TG}}$
Boundaries
(e)
(f)
(g)
(h)
(i)
(j)

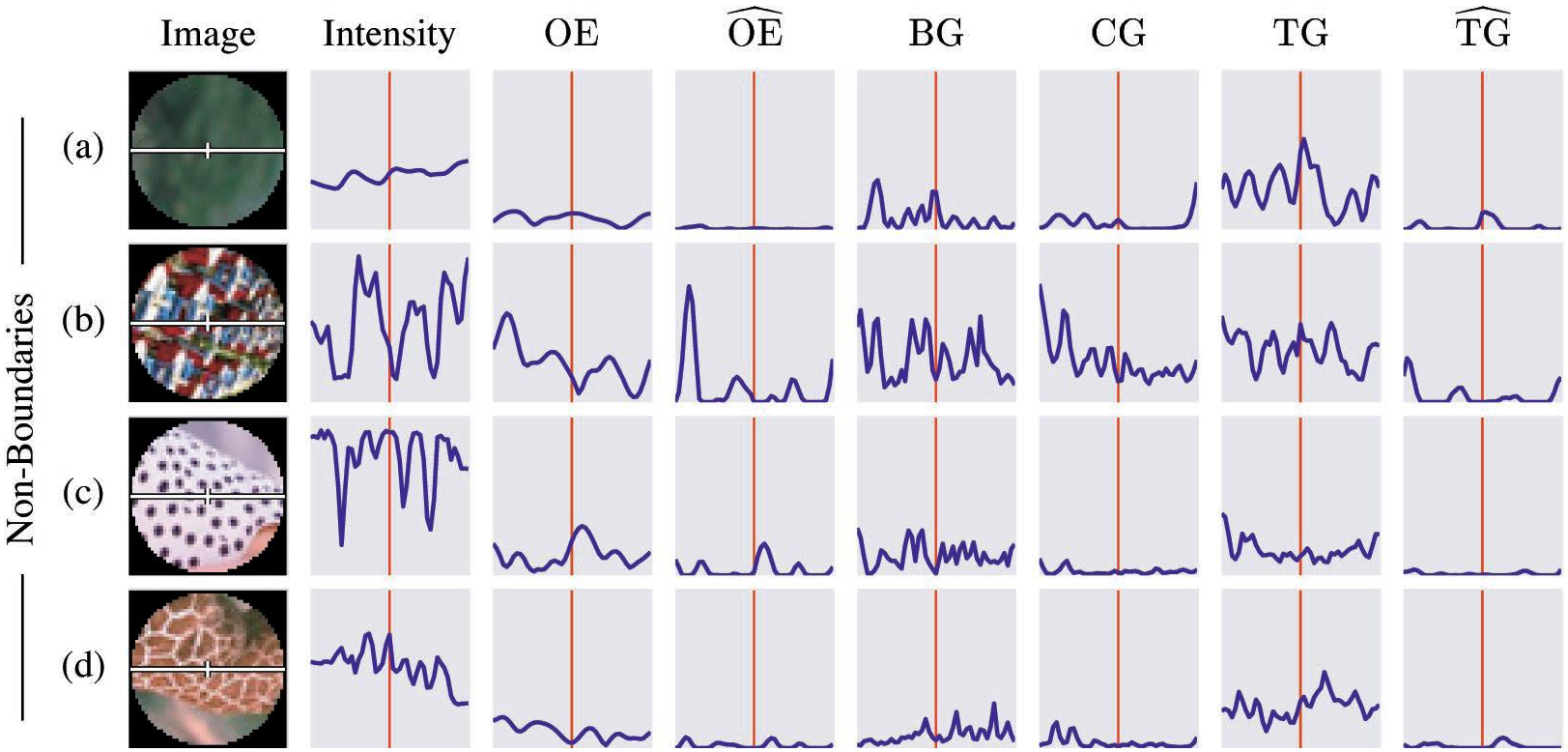


Image
Intensity
OE
$\widehat{\text{OE}}$
BG
CG
TG
$\widehat{\text{TG}}$
Non-Boundaries
(a)
(b)
(c)
(d)

Parameter tuning (cue optimization)

- Scale
- Disk radius
- Number of bins in histogram

Trained on labeled data

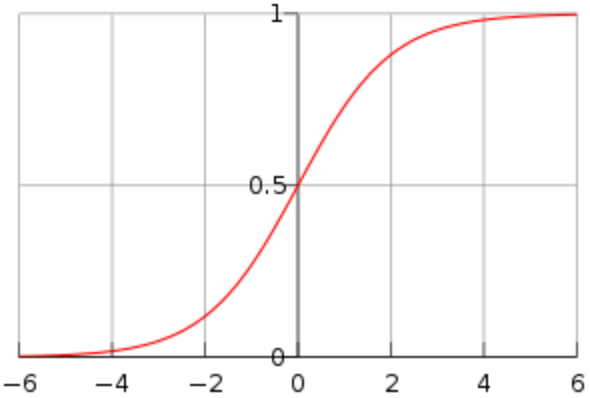
Cue combination

- Tested many options:
  - Logistic regression – the winner!
  - SVM (support vector machines)
  - Hierarchical Mixture of Experts
  - Classification trees

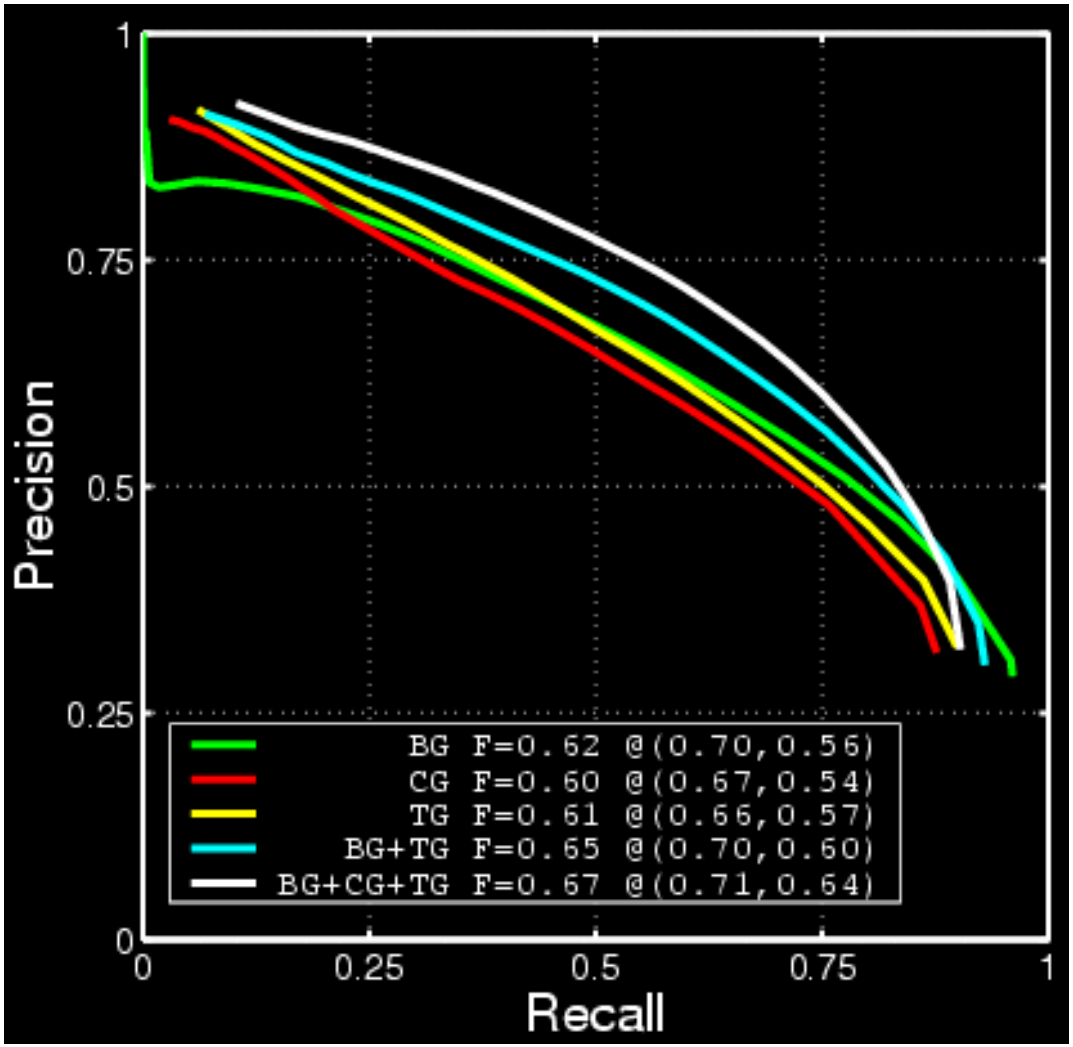
$$f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k,$$

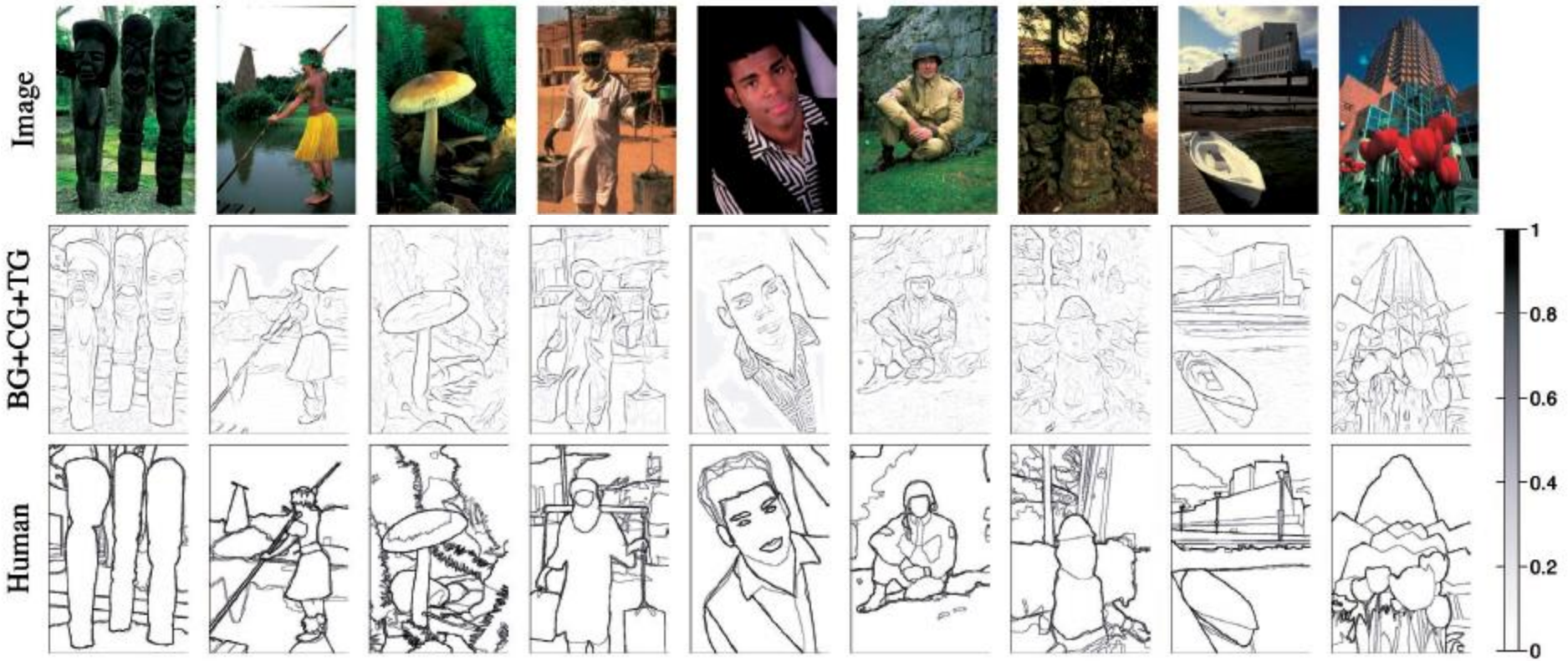
The logistic function



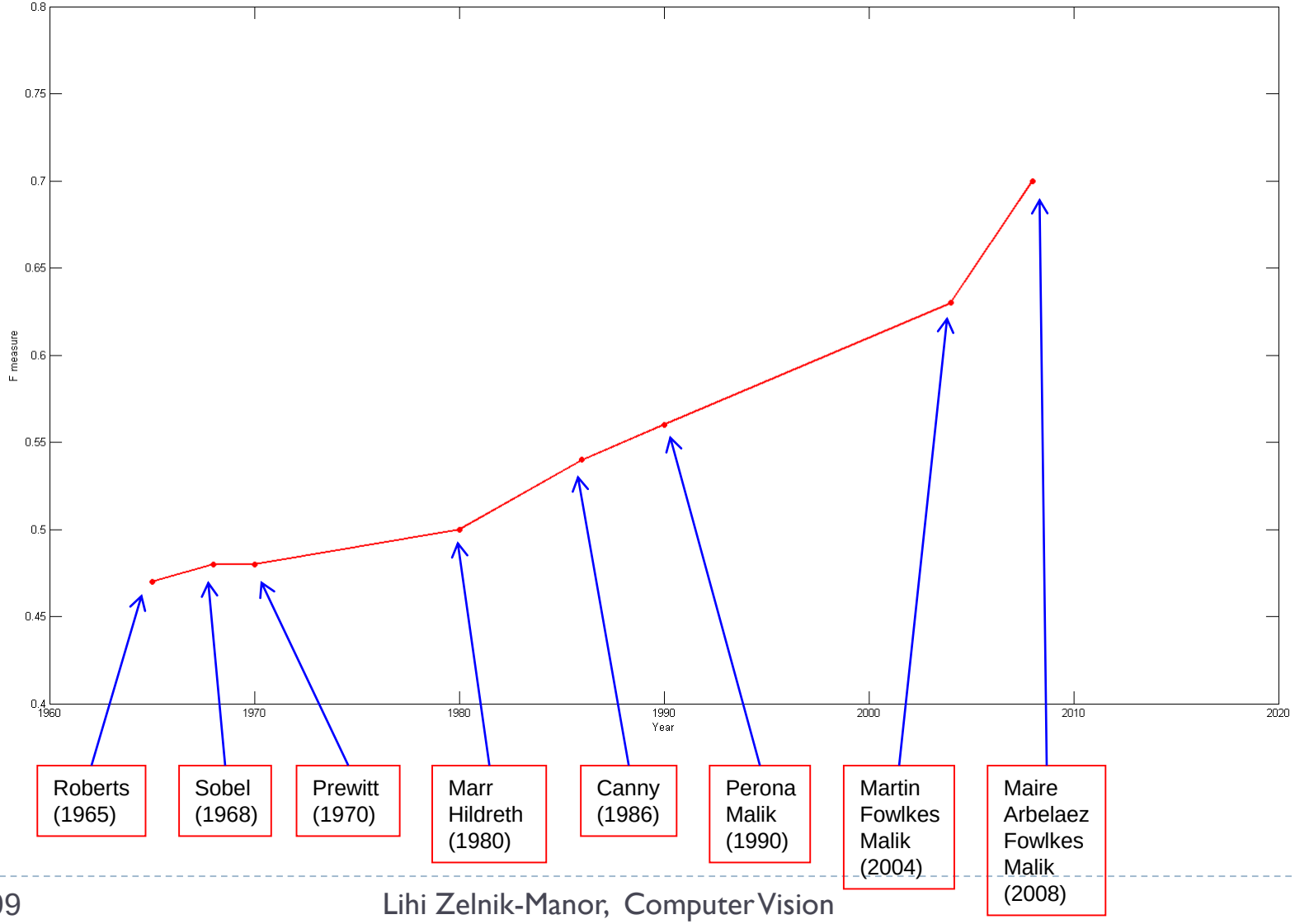
Various cue combinations



Example results



Forty years of contour detection



Global pB boundary detector

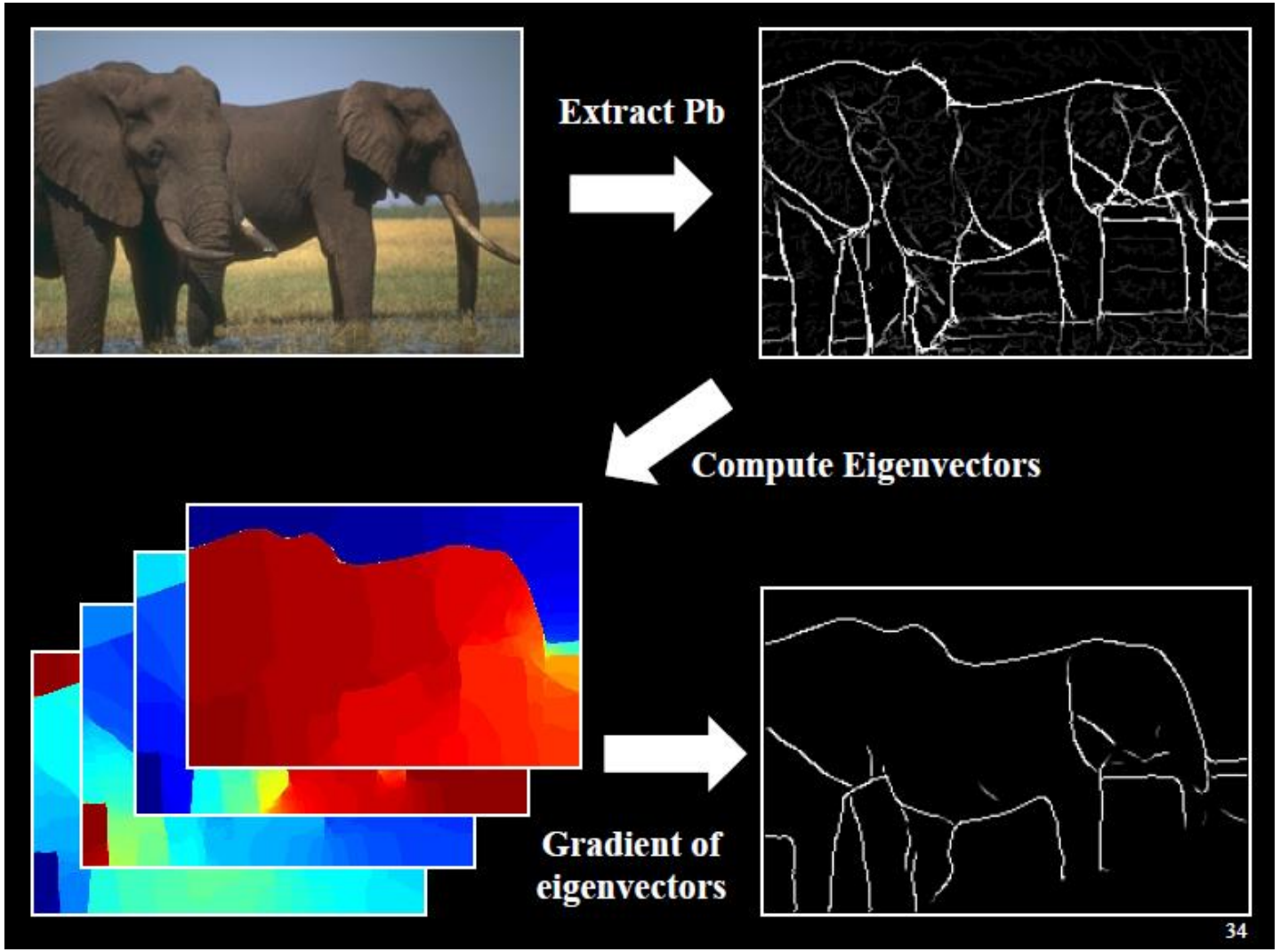


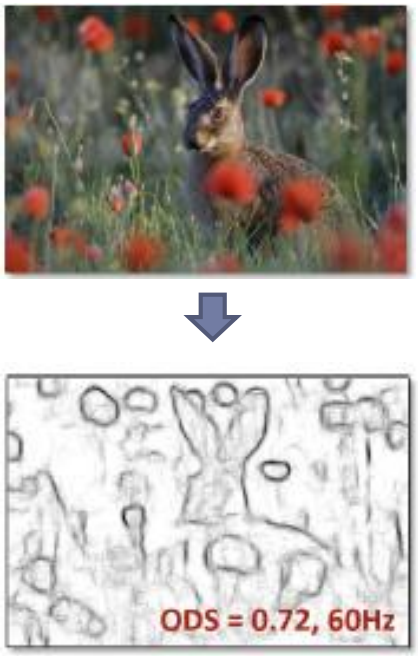
Figure from Fowlkes

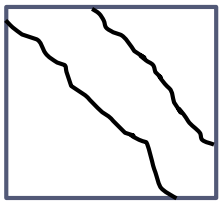
Edge Detection with Structured Random Forests (Dollar Zitnick ICCV 2013)

- Goal: quickly predict whether each pixel is an edge

- Insights
  - Predictions can be learned from training data
  - Predictions for nearby pixels should not be independent

- Solution
  - Train structured random forests to split data into patches with similar boundaries based on features
  - Predict boundaries at patch level, rather than pixel level, and aggregate (average votes)



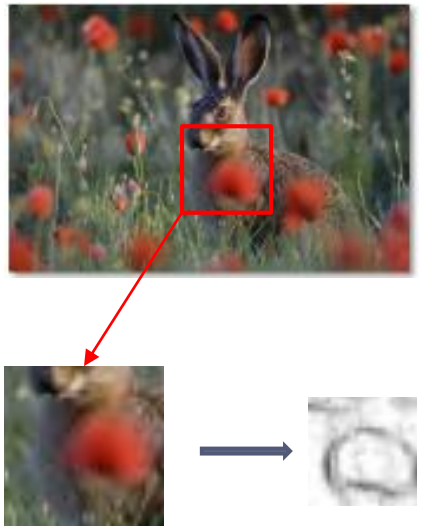


Boundaries in patch

http://research.microsoft.com/pubs/202540/DollarICCV13edges.pdf

Edge Detection with Structured Random Forests

- Algorithm
  1. Extract overlapping 32x32 patches at three scales
  2. Features are pixel values and pairwise differences in feature maps (LUV color, gradient magnitude, oriented gradient)
  3. Predict T boundary maps in the central 16x16 region using T trained decision trees
  4. Average predictions for each pixel across all patches



Edge Detection with Structured Random Forests

Results

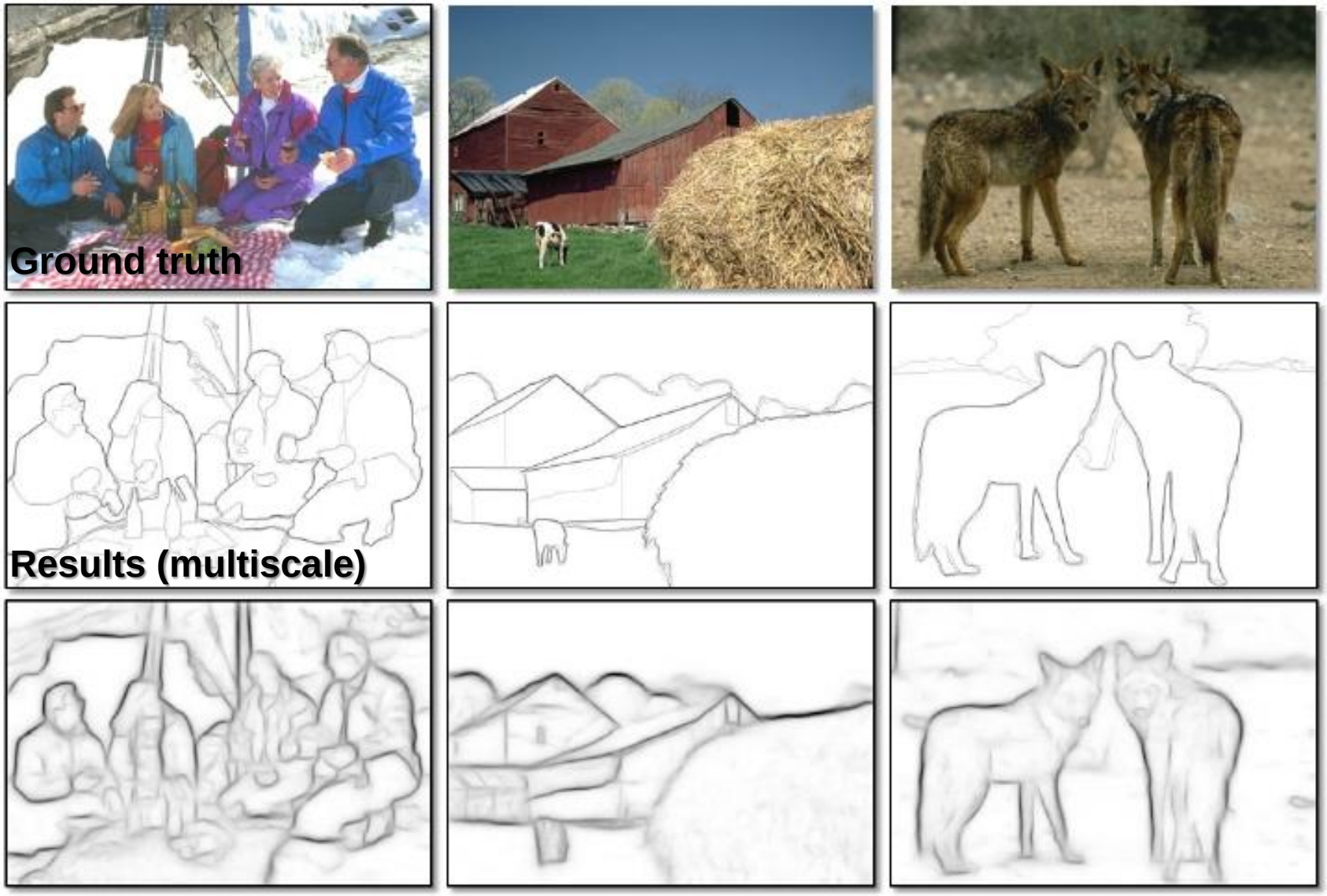
BSDS 500

	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.60	.64	.58	15
Felz-Hutt [11]	.61	.64	.56	10
Hidayat-Green [16]	.62†	-	-	20
BEL [9]	.66†	-	-	1/10
gPb + GPU [6]	.70†	-	-	1/2‡
gPb [1]	.71	.74	.65	1/240
gPb-owt-ucm [1]	.73	**.76**	.73	1/240
Sketch tokens [21]	.73	.75	**.78**	1
SCG [31]	**.74**	**.76**	.77	1/280
SE-SS, T=1	.72	.74	.77	**60**
SE-SS, T=4	.73	.75	.77	30
SE-MS, T=4	**.74**	**.76**	**.78**	6

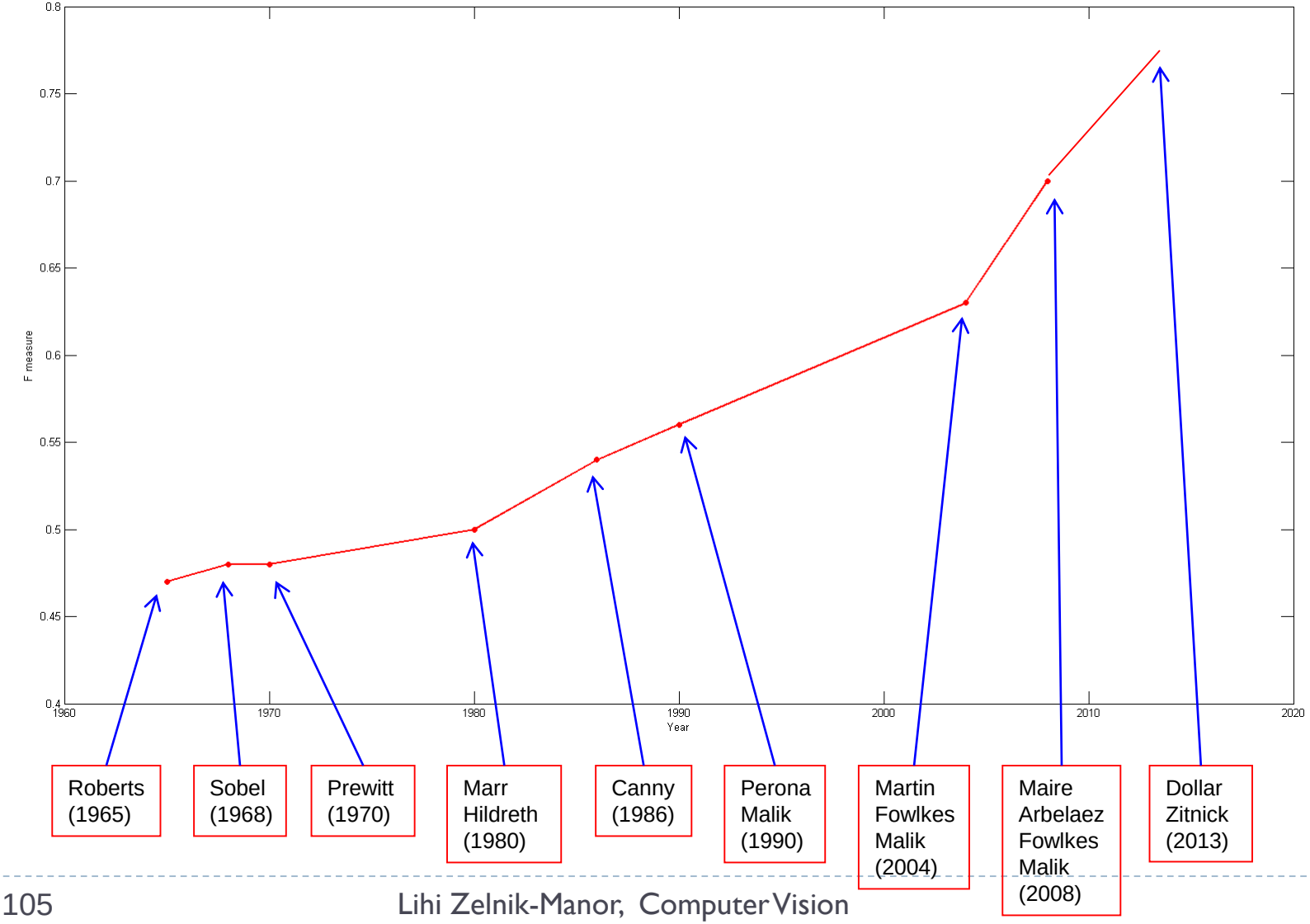
NYU Depth dataset edges

	ODS	OIS	AP	FPS
gPb [1] (rgb)	.51	.52	.37	1/240
SCG [31] (rgb)	.55	.57	.46	1/280
SE-SS (rgb)	.58	.59	.53	**30**
SE-MS (rgb)	**.60**	**.61**	**.56**	6
gPb [1] (depth)	.44	.46	.28	1/240
SCG [31] (depth)	.53	.54	.45	1/280
SE-SS (depth)	.57	.58	.54	**30**
SE-MS (depth)	**.58**	**.59**	**.57**	6
gPb [1] (rgbd)	.53	.54	.40	1/240
SCG [31] (rgbd)	.62	.63	.54	1/280
SE-SS (rgbd)	.62	.63	.59	**25**
SE-MS (rgbd)	**.64**	**.65**	**.63**	5

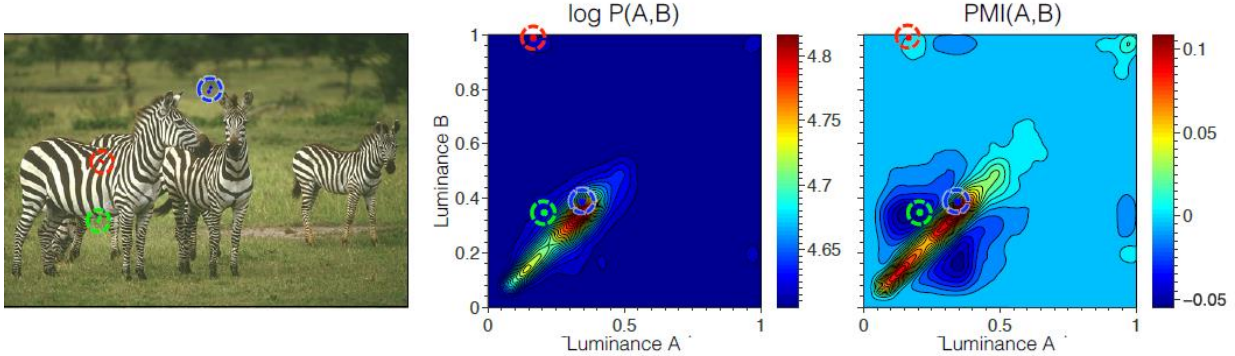
Edge Detection with Structured Random Forests



Forty years of contour detection



Crisp Boundary Detection using Pointwise Mutual Information (Isola et al. ECCV 2014)



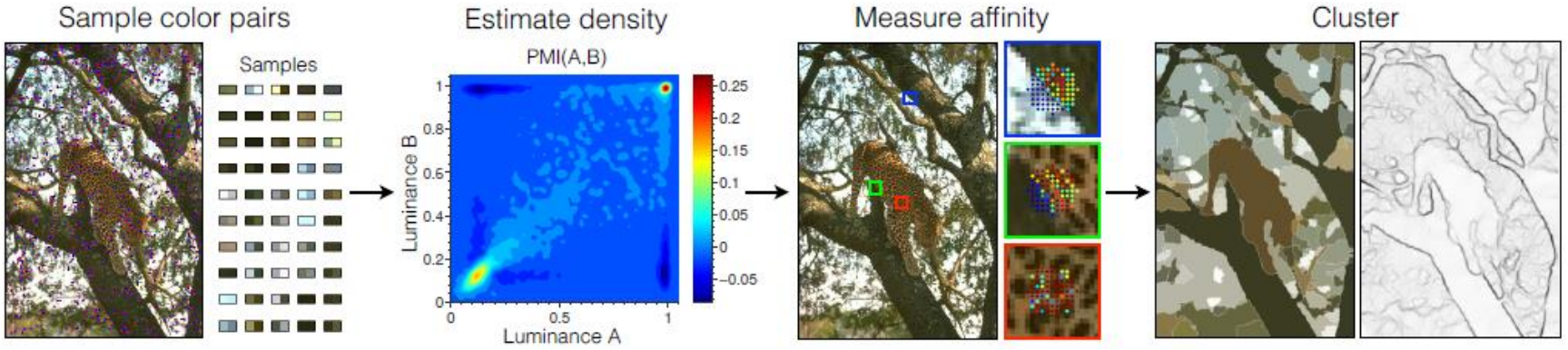
$$\mathrm{PMI}_\rho(A,B) = \log \frac{P(A,B)^\rho}{P(A)P(B)}$$

Pixel color combinations that are unlikely to be together are edges

Algorithm:

Kernel density estimation

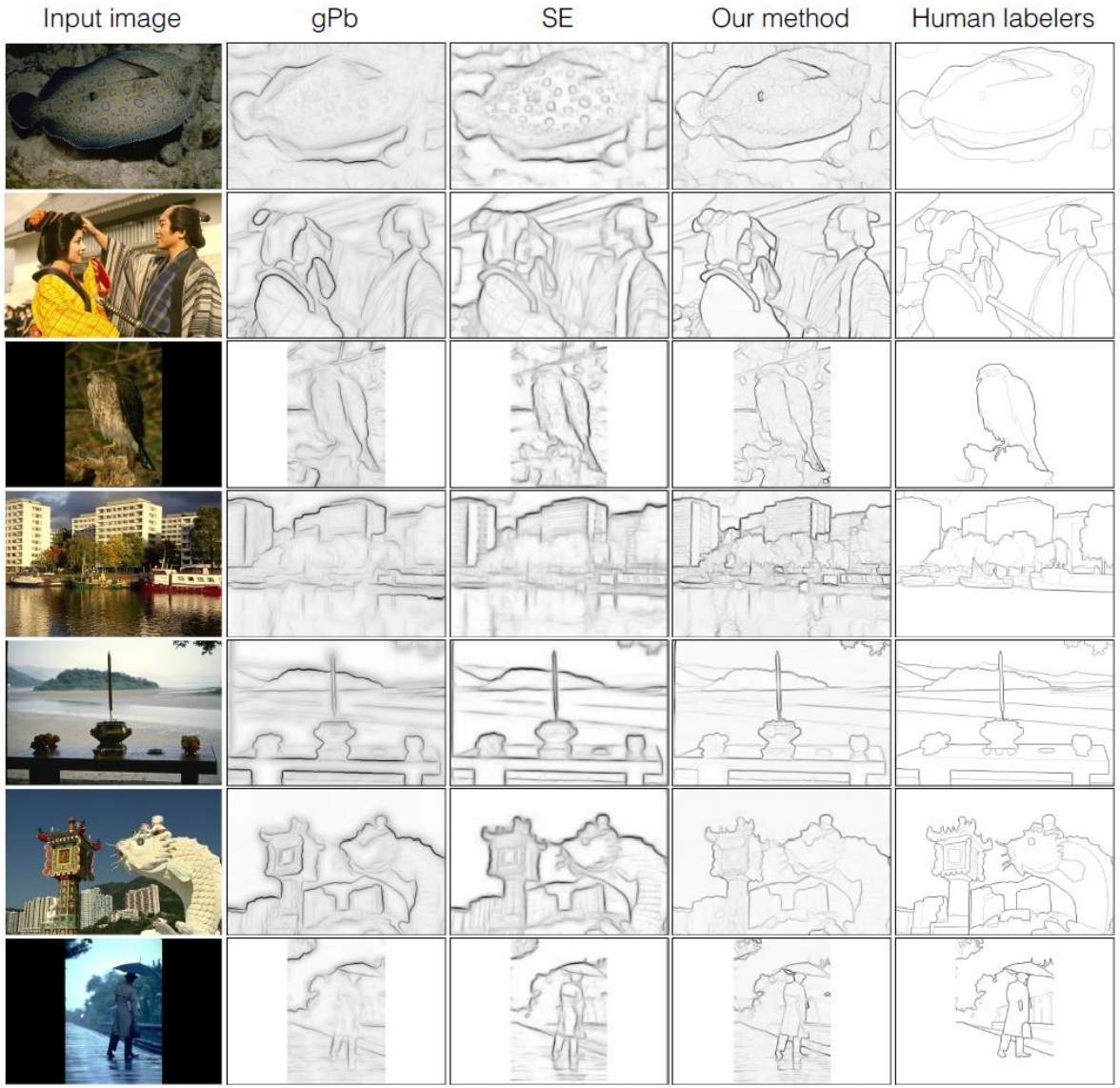
Spectral clustering



http://web.mit.edu/phillipi/www/publications/crisp_boundaries.pdf

Crisp Boundary Detection using Pointwise Mutual Information

Algorithm	ODS	OIS	AP
Canny [14]	0.60	0.63	0.58
Mean Shift [36]	0.64	0.68	0.56
NCuts [37]	0.64	0.68	0.45
Felz-Hutt [38]	0.61	0.64	0.56
gPb [1]	0.71	0.74	0.65
gPb-owt-ucm [1]	0.73	0.76	0.73
SCG [9]	**0.74**	0.76	0.77
Sketch Tokens [7]	0.73	0.75	0.78
SE [8]	**0.74**	0.76	0.78
Our method – SS, color only	0.72	0.75	0.77
Our method – SS	0.73	0.76	**0.79**
Our method – MS	**0.74**	**0.77**	0.78



Slide Credits

- Trevor Darell
- Kristen Grauman
- Jitendra Malik
- FeiFei Li
- Derek Hoiem
- and Seitz, Marschner, Lazebnik and others

End – finding lines

Now you know how it works