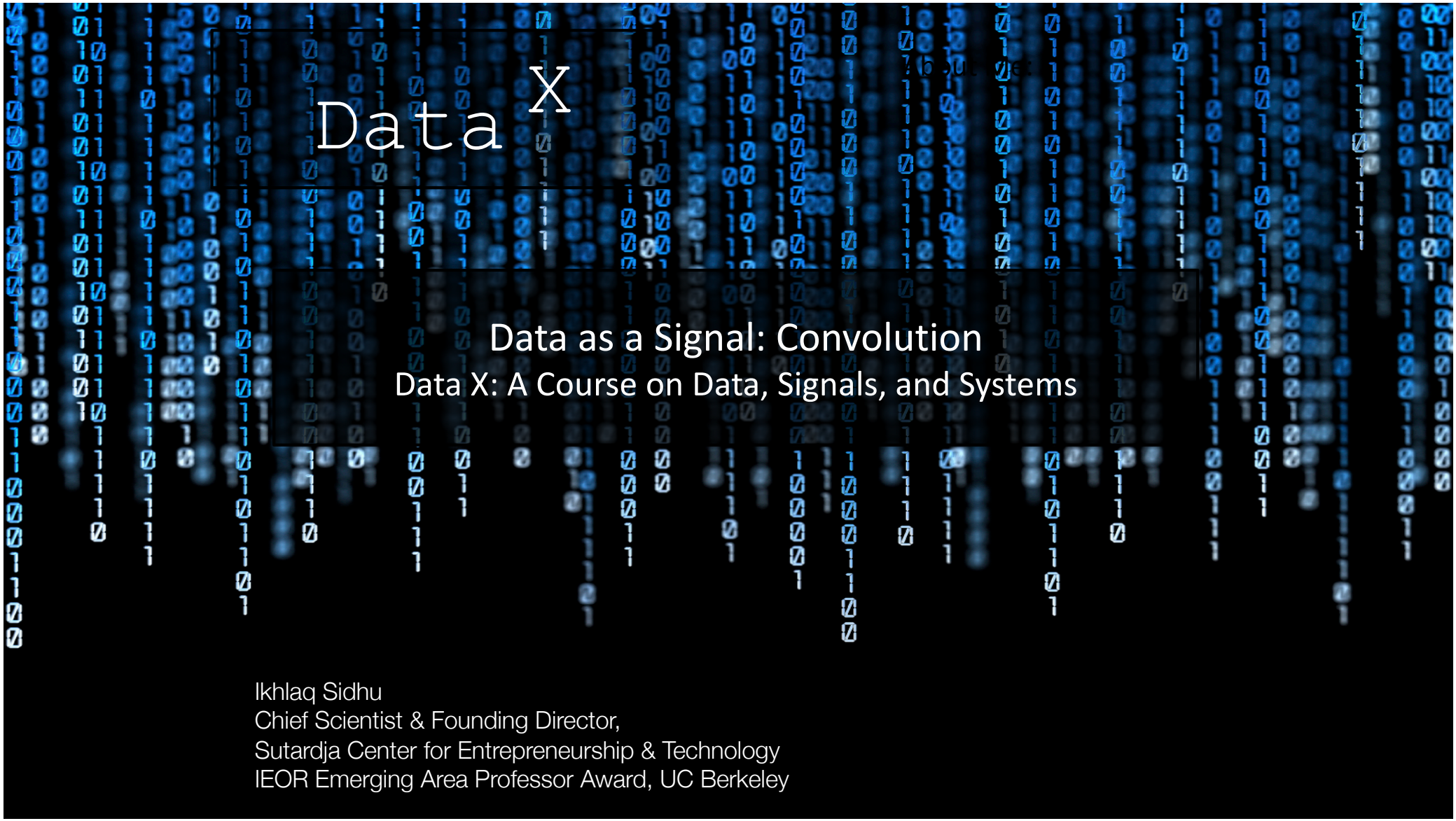


Data as a Signal: Convolution

Data X: A Course on Data, Signals, and Systems

Ikhlaq Sidhu
Chief Scientist & Founding Director,
Sutardja Center for Entrepreneurship & Technology
IEOR Emerging Area Professor Award, UC Berkeley

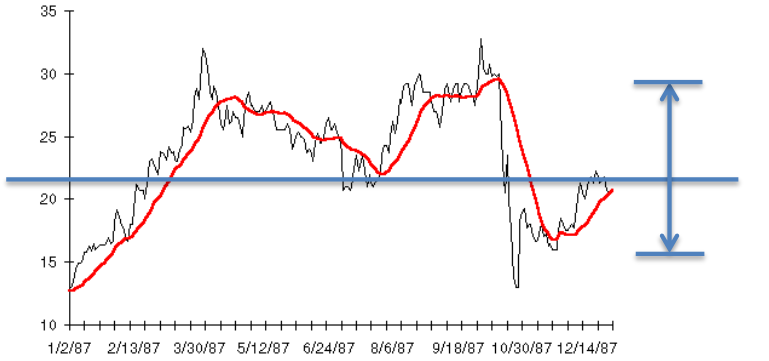
Before We Discuss Convolutional Neural Networks
Lets Discuss Convolution and LTI Systems First

Simple things we can do with a sequence: Mean, Variance

Discrete data

x_n = x1, x2, x3, …

Rec	Observed
1	60.323
2	61.122
3	60.171
4	61.187
5	63.221
6	63.639
7	64.989
8	63.761
9	66.019
10	67.857
11	68.169
12	66.513
13	68.655
14	69.564
15	69.331
16	70.551



variance

mean

Sample Mean	Population Mean
$\bar{x} = \frac{\Sigma x}{n}$	$\mu = \frac{\Sigma x}{N}$

where Σx is sum of all data values

N is number of data items in population

n is number of data items in sample

$$\sigma^2 = \frac{\sum (X-\mu)^2}{N}$$

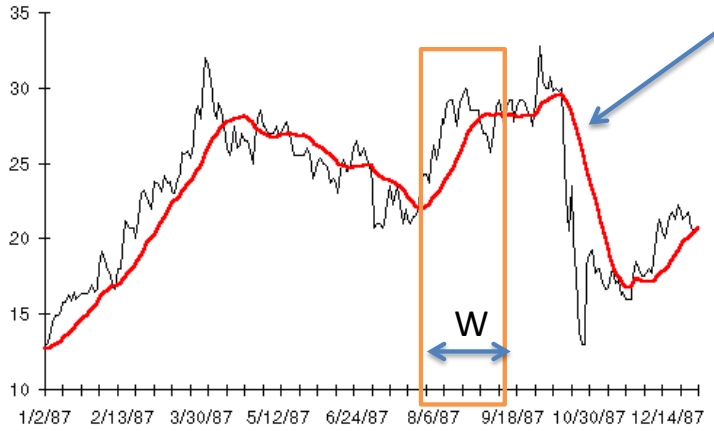
$$s^2 = \frac{\sum_{i=1}^{n} (X_i - X_{avg})^2}{n-1}$$

Moving Average of Sequence

(Simple) Moving Average

- Discrete data x_n = x1, x2, x3, …
- Moving Average

W =size of window
MA(n) is red line, at the right edge of the window



Note:

* It's a function of n (and maybe W)

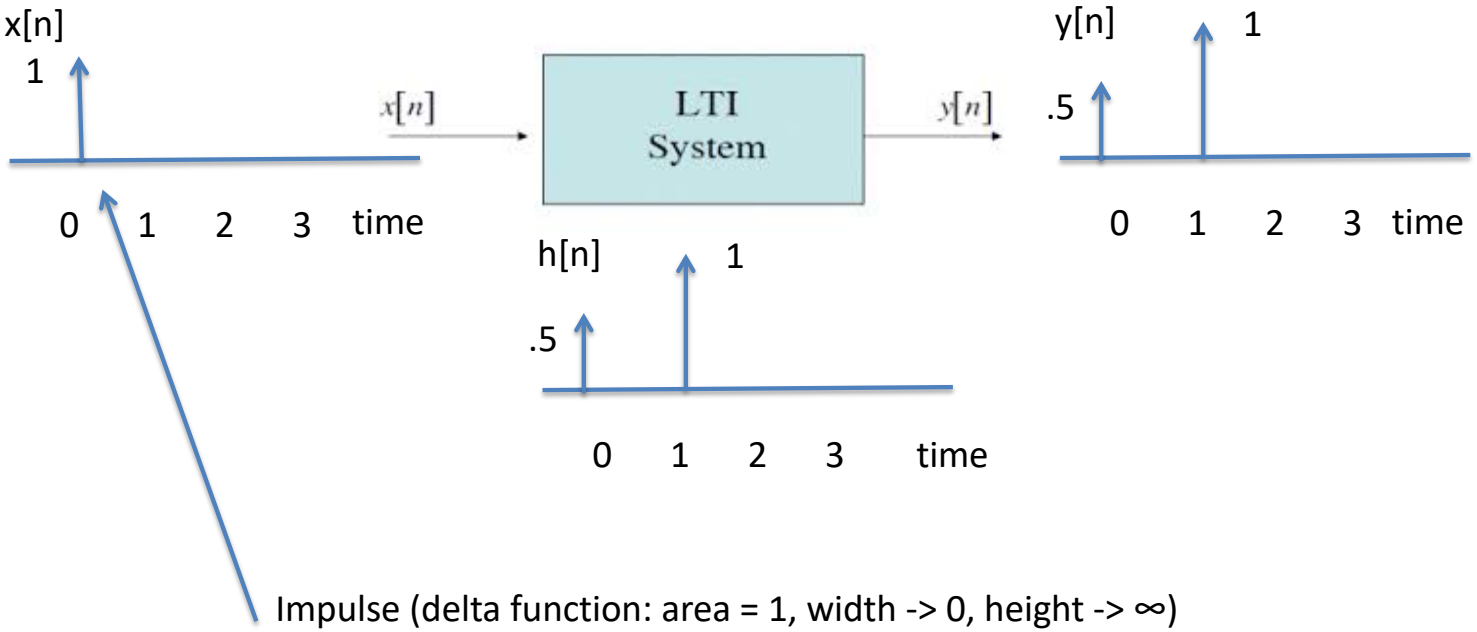
- Its smoother than the original (low pass filter)
- W = window size, can be in the summation

$$MA(n) = \frac{1}{W} \sum_{i=0.}^{W-1} x_{n-i}$$

$$MA(n) = \sum_{i=0.}^{W-1} \frac{x_{n-i}}{W}$$

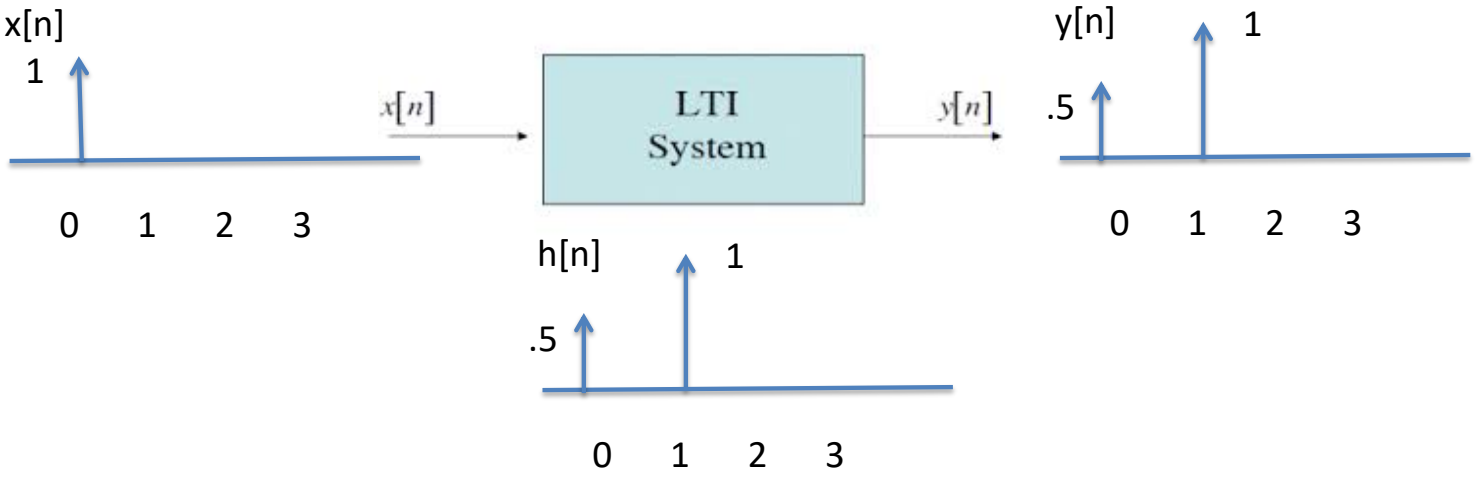
Linear Time Invariant System Approaches

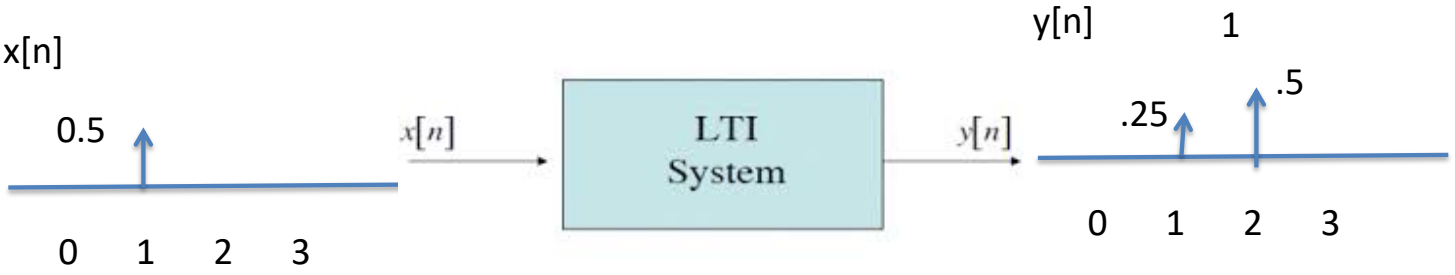
Linear Time Invariant System



Impulse (delta function: area = 1, width -> 0, height -> ∞)
In discrete systems, its just a value of “1”

Linear Time Invariant System





Convolution is actually really simple, but powerful

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

- x[[n] = 1 2 3 2 4 0 0 0 …
- h[n] = 1 1 1
- y[n] = 1 1 1 ← h[n-0] x[0]

  2 2 2 ← h[n-1] x[1]

  3 3 3 ← h[n-2] x[2]

  2 2 2 ← h[n-3] x[3]

  4 4 4 ← h[n-4] x[4]

---

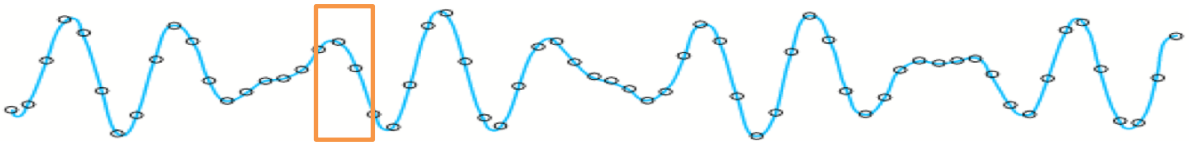
y[n] =1 3 6 7 9 6 4 …

Note: if h[n] = 1/3, 1/3, 1/3, then y[n] = 1/3, 1, 2, 7/3, 3, 2, 4/3

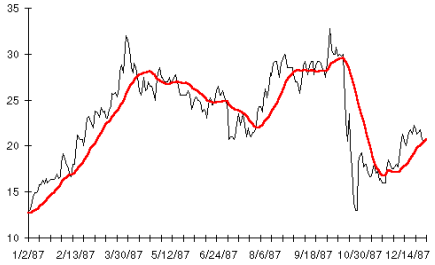
Note, MA with Window = 3 (if divided by 3)

Another Way to Think about Moving Average

x(t)



x(t) → LTI: h(n)=1,1,1,1 → y[n] is similar to moving average with window of 4



x(n) = array with sequence of numbers = [10, 3, 6, 12, …. … 43, 12, 1, 4]

h(n) = impulse response function from linear systems
= [1, 1, 1, 1]

MA(n, W) = y(n) = x(t) * h(t) (convolution)

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

Continuous $$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau = x(t) * h(t)$$

Circular $$(x \circledast y)_n \triangleq \sum_{m=0}^{N-1} x(m)y(n-m)$$

Another Convolution, different impulse response

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

- x[[n] = 1 2 3 2 4 0 0 0 …
- h[n] = 3 2 1
- y[n] = 3 2 1 ← h[n-0] x[0]

  6 4 1 ← h[n-1] x[1]

  9 6 1 ← h[n-2] x[2]

  6 4 1 ← h[n-3] x[1]

  12 8 4 ← h[n-1] x[1]

---

y[n] =3 8 14 13 17 9 4 …

This is called filtering the data with an FIR filter with impulse response h[n] = 3 2 1

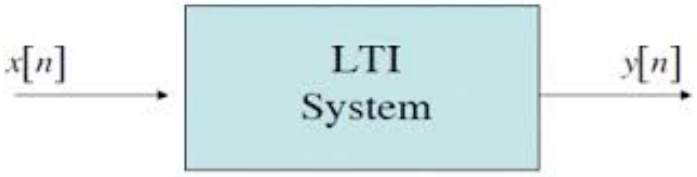
Code Example of Convolution

```
>>> np.convolve([1, 2, 3], [0, 1, 0.5])
array([ 0. ,  1. ,  2.5,  4. ,  1.5])
```

numpy.convolve(a, v, mode='full')[source]

- Returns the discrete, linear convolution of two one-dimensional sequences.
- The convolution operator is often seen in signal processing, where it models the effect of a linear time-invariant system on a signal [R17]. In probability theory, the sum of two independent random variables is distributed according to the convolution of their individual distributions.

Convolution: Why We Care



FIR: Finite Impulse Response Filter

Take Any Input Sequence x[n]
Filter for what you want with h[n]
Get result y[n]

h[n]	**Result**
1/W,1/W,1/W,1/W	Low pass filter or moving average, with 1/W cutoff
(a1,a2,…a10)	Any linear weighted sum
1, -1	Differential
Any sequence	Cross-correlation, allows you to look for similar pattern

Properties of Convolution

EQUATION 7-6
The commutative property of convolution. This states that the order in which signals are convolved can be exchanged.

$$a[n] * b[n] = b[n] * a[n]$$

EQUATION 7-8
The distributive property of convolution describes how parallel systems are analyzed.

$$a[n] * b[n] + a[n] * c[n] = a[n] * (b[n] + c[n])$$

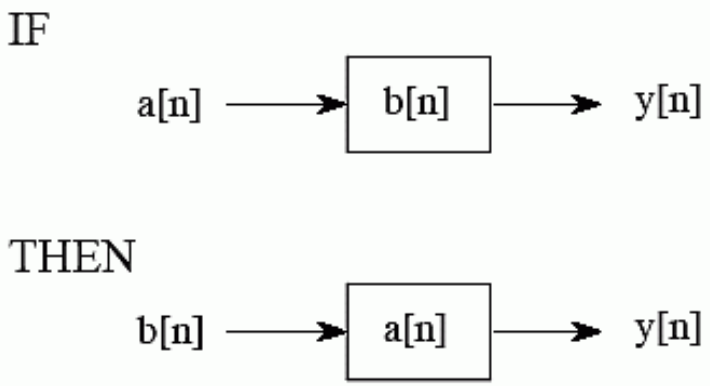


FIGURE 7-8
The commutative property in system theory. The commutative property of convolution allows the input signal and the impulse response of a system to be exchanged without changing the output. While interesting, this usually has no physical significance. (A signal appearing inside of a box, such as b[n] and a[n] in this figure, represent the *impulse response* of the system).

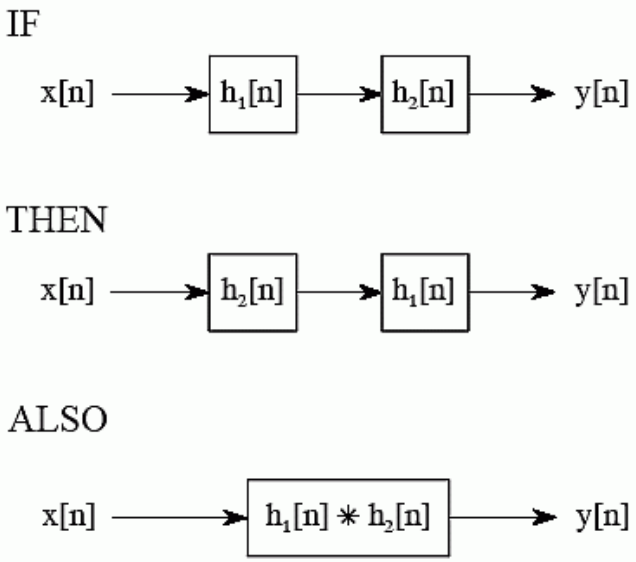


FIGURE 7-9
The associative property in system theory. The associative property provides two important characteristics of *cascaded* linear systems. First, the order of the systems can be rearranged without changing the overall operation of the cascade. Second, two or more systems in a cascade can be replaced by a single system. The impulse response of the replacement system is found by convolving the impulse responses of the stages being replaced.

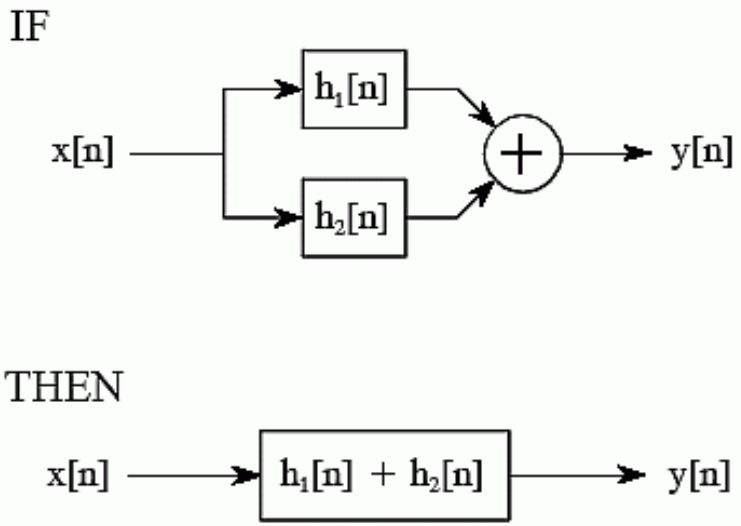
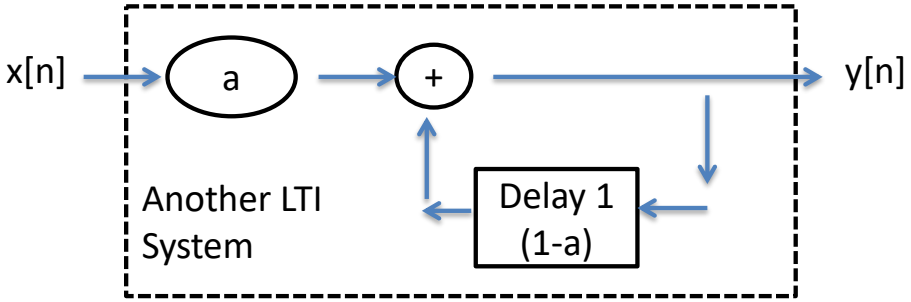


FIGURE 7-10
The distributive property in system theory. The distributive property shows that parallel systems with added outputs can be replaced with a single system. The impulse response of the replacement system is equal to the sum of the impulse responses of all the original systems.

Another LTI System: IIR: Infinite Impulse Response Filter

- Discrete data x_n = x1, x2, x3, …



$$y(n) = \alpha x(n) + (1-\alpha) y(n-1)$$

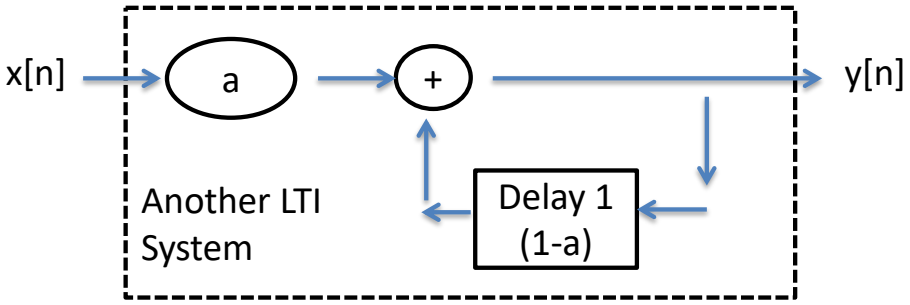
$$y(n) = \alpha x(n) + \alpha^2 x(n-1) + \alpha^3 x(n-2) \ldots$$

Infinitely back in x[n]

Another LTI System: IIR: Infinite Impulse Response Filter

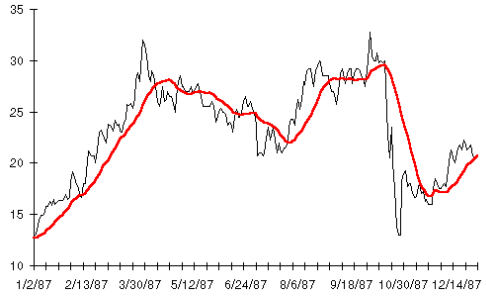
- Discrete data x_n = x1, x2, x3, …

$$y(n) = \alpha x(n) + (1-\alpha) y(n-1)$$



$$y(n) = \alpha x(n) + \alpha^2 x(n-1) + \alpha^3 x(n-2) \ldots$$

Exponential Moving Average



alpha = smoothing factor of 2/(W+1)

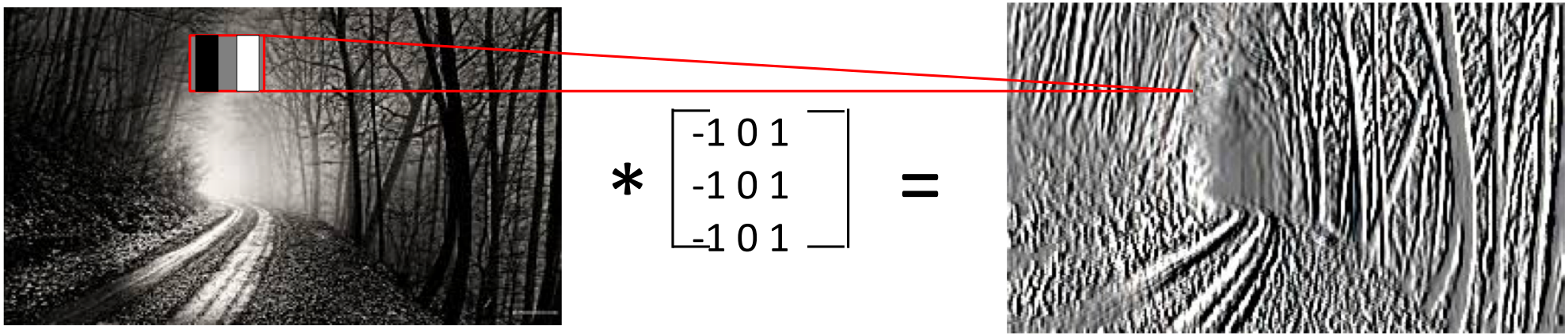
W is number of time periods

Infinitely back in x[n]

With Images
Convolutions are 2D

Examples in this section are from
Alyosha Efros, Slides in Guest Lecture

Convolutional of Two Signals



$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[.,.]$

	0	10							

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

	0	10	20						

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

	0	10	20	30					

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

	0	10	20	30	30				

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[.,.]$

	0	10	20	30	30				
					?				

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[.,.]$

	0	10	20	30	30				
						?			
					50				

Image filtering

$h[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

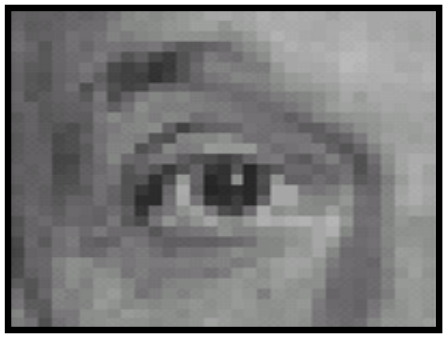
$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot,\cdot]$

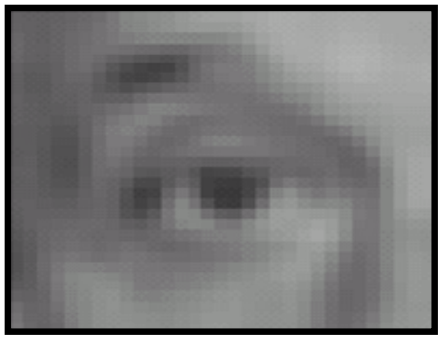
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Linear filters: examples



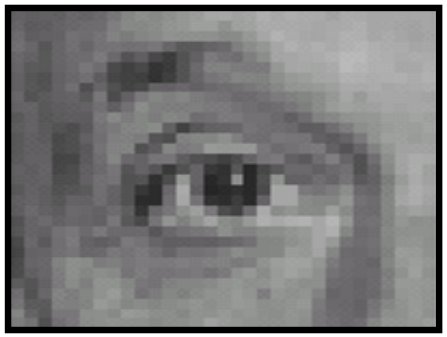
Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} =$$



Blur (with a mean filter)

Practice with linear filters

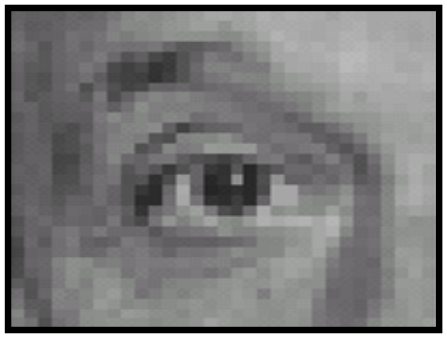


Original

0	0	0
0	1	0
0	0	0

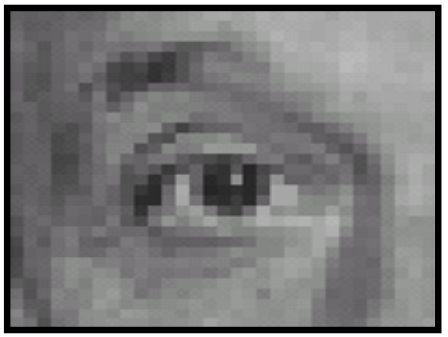
?

Practice with linear filters



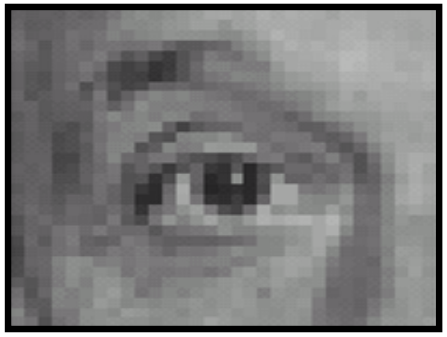
Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters

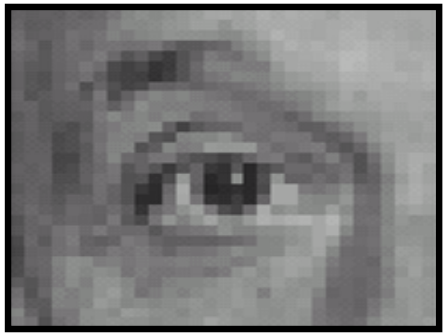


Original

0	0	0
0	0	1
0	0	0

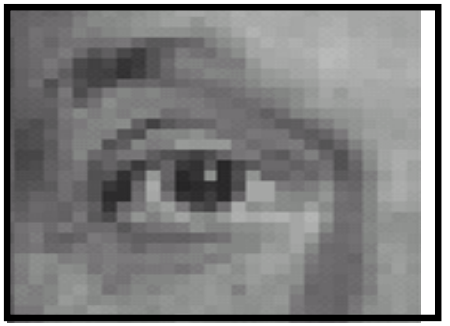
?

Practice with linear filters



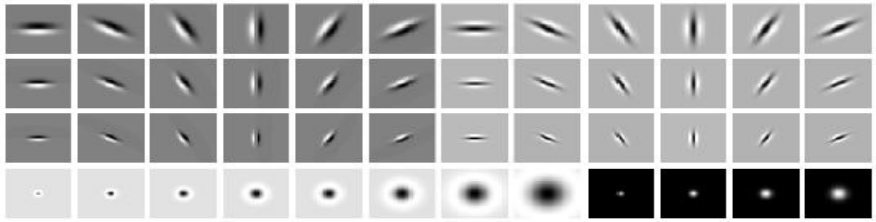
Original

0	0	0
0	0	1
0	0	0

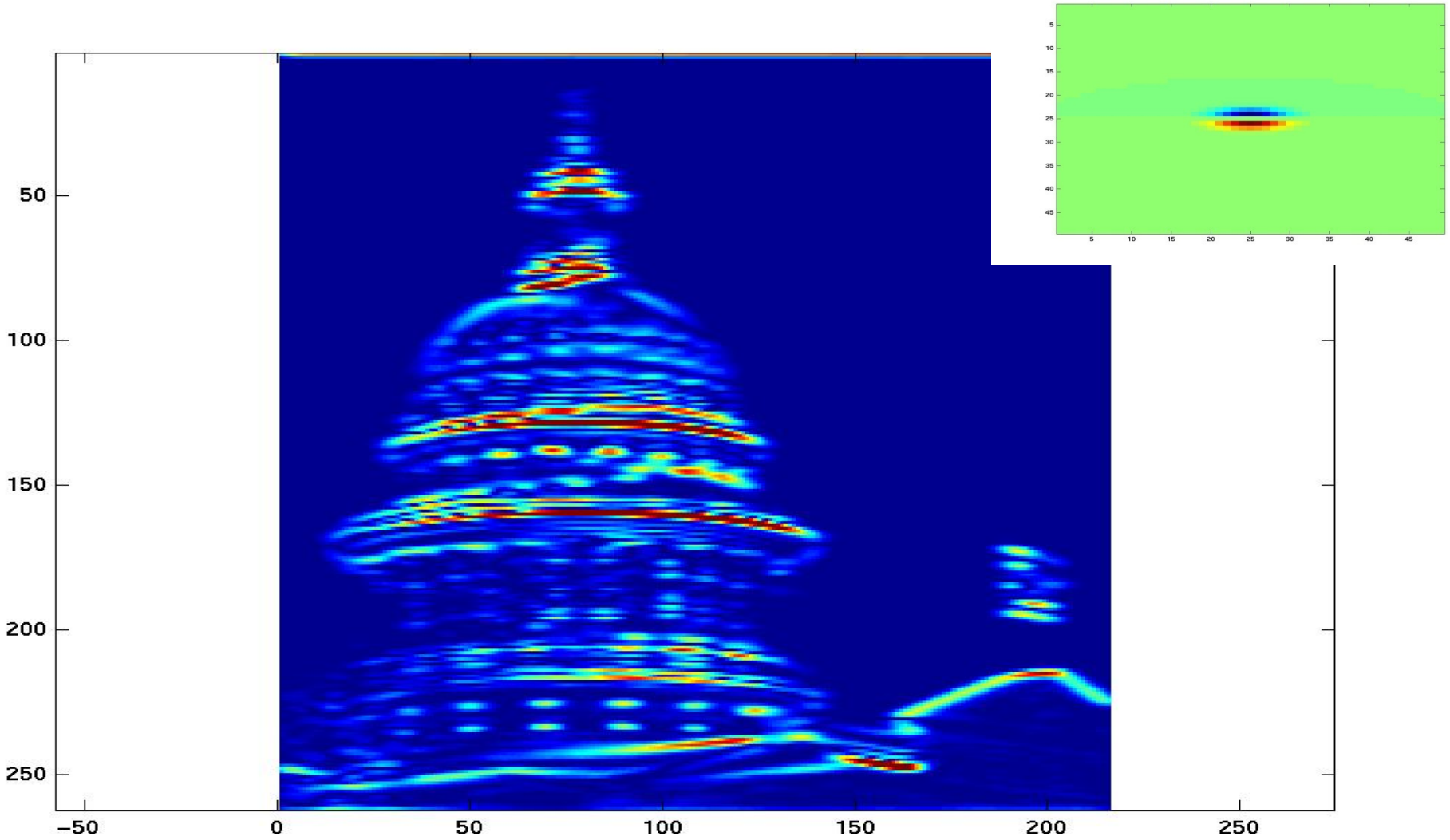


Shifted left
By 1 pixel

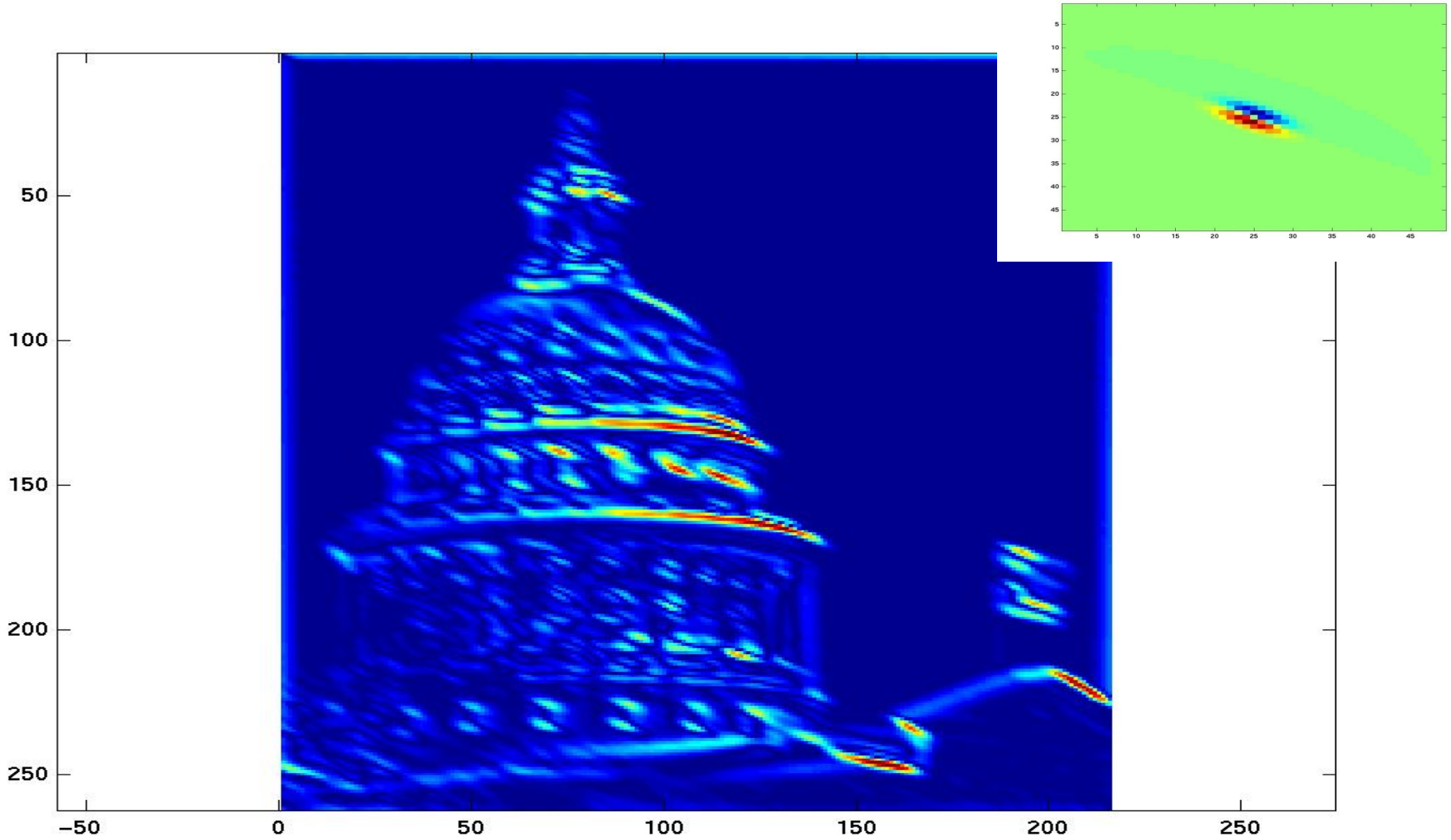
from http://www.texasexplorer.com/austincap2.jpg



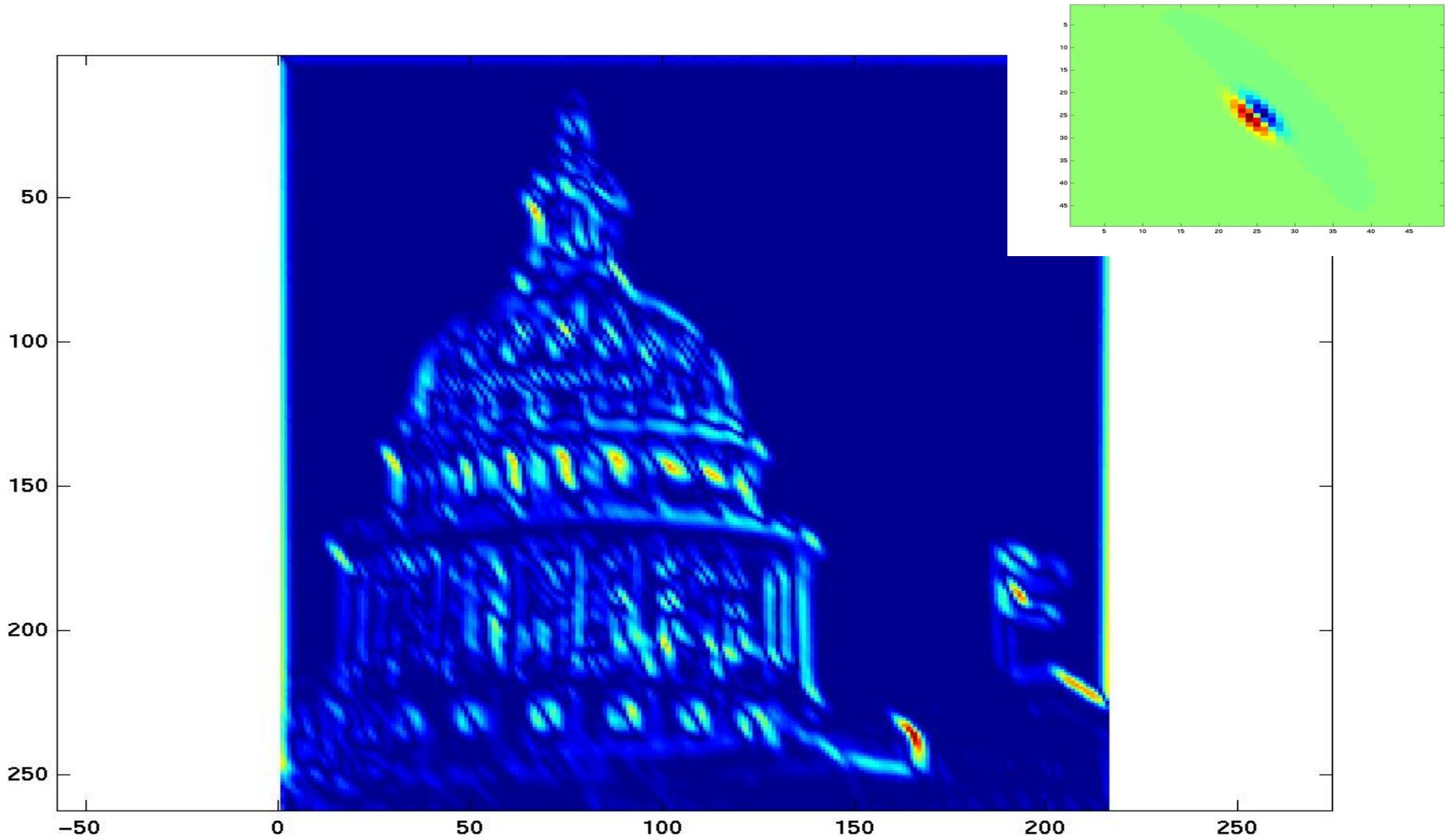
Data



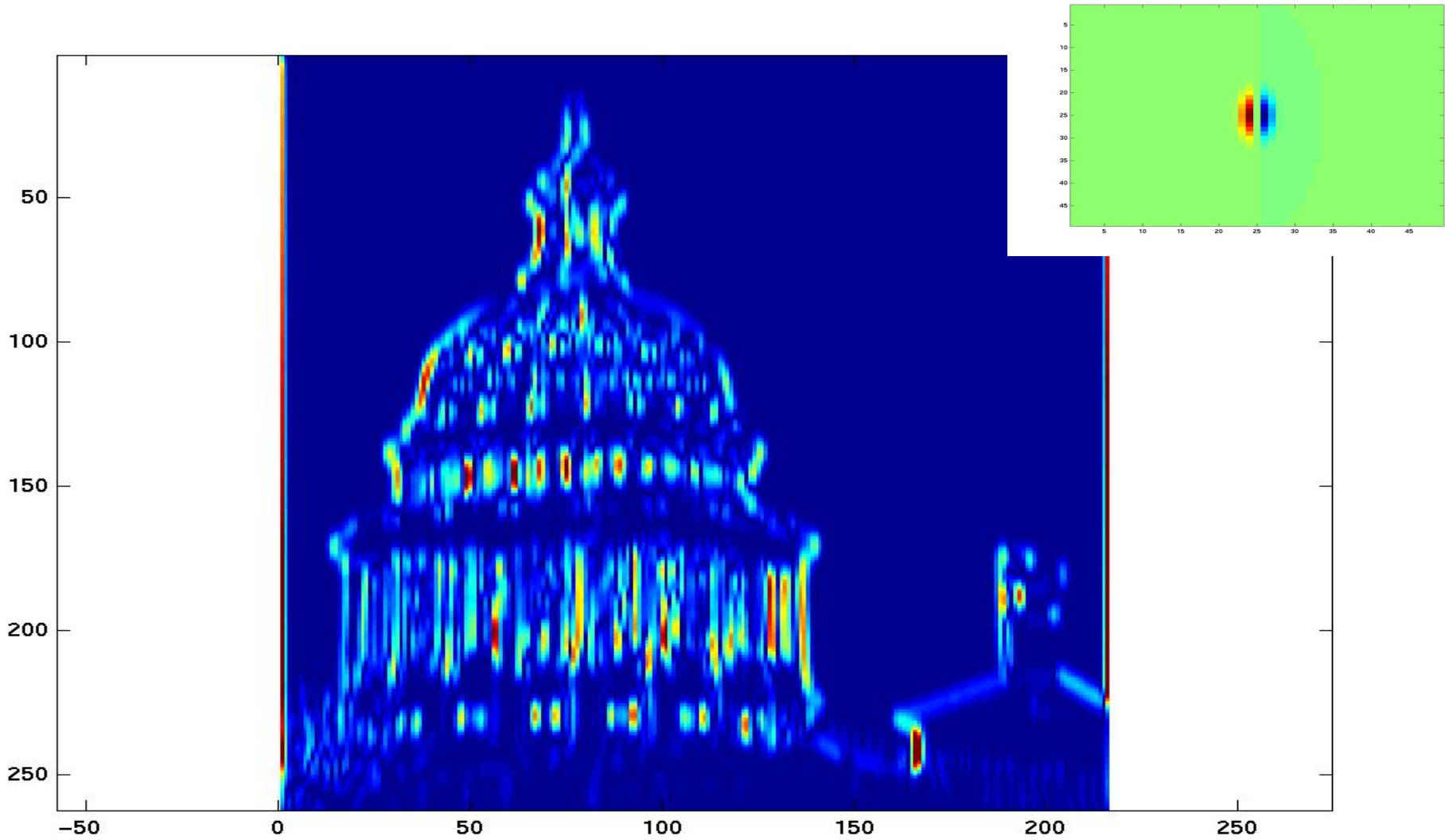
Kristen Grauman



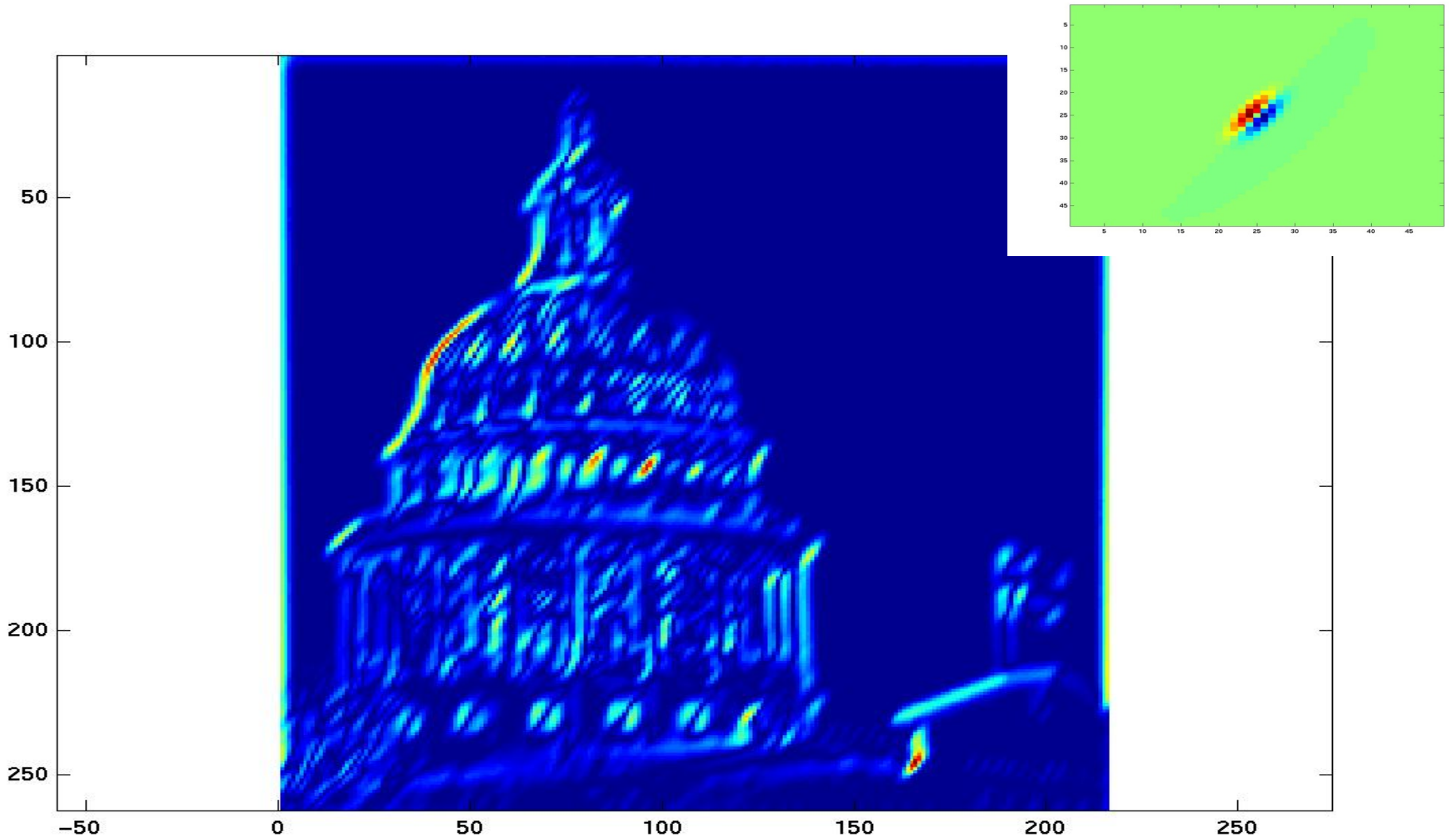
Kristen Grauman



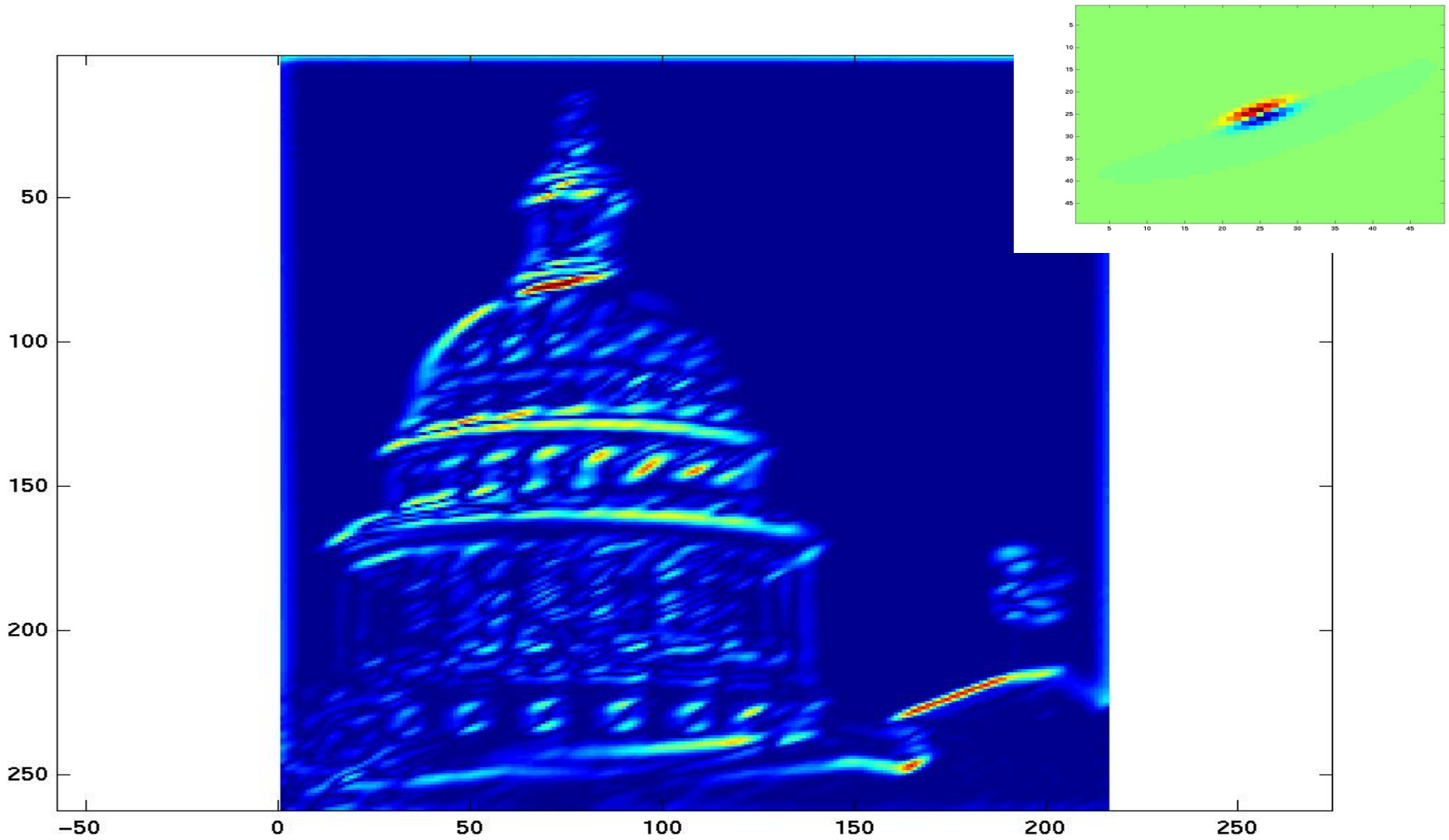
Kristen Grauman



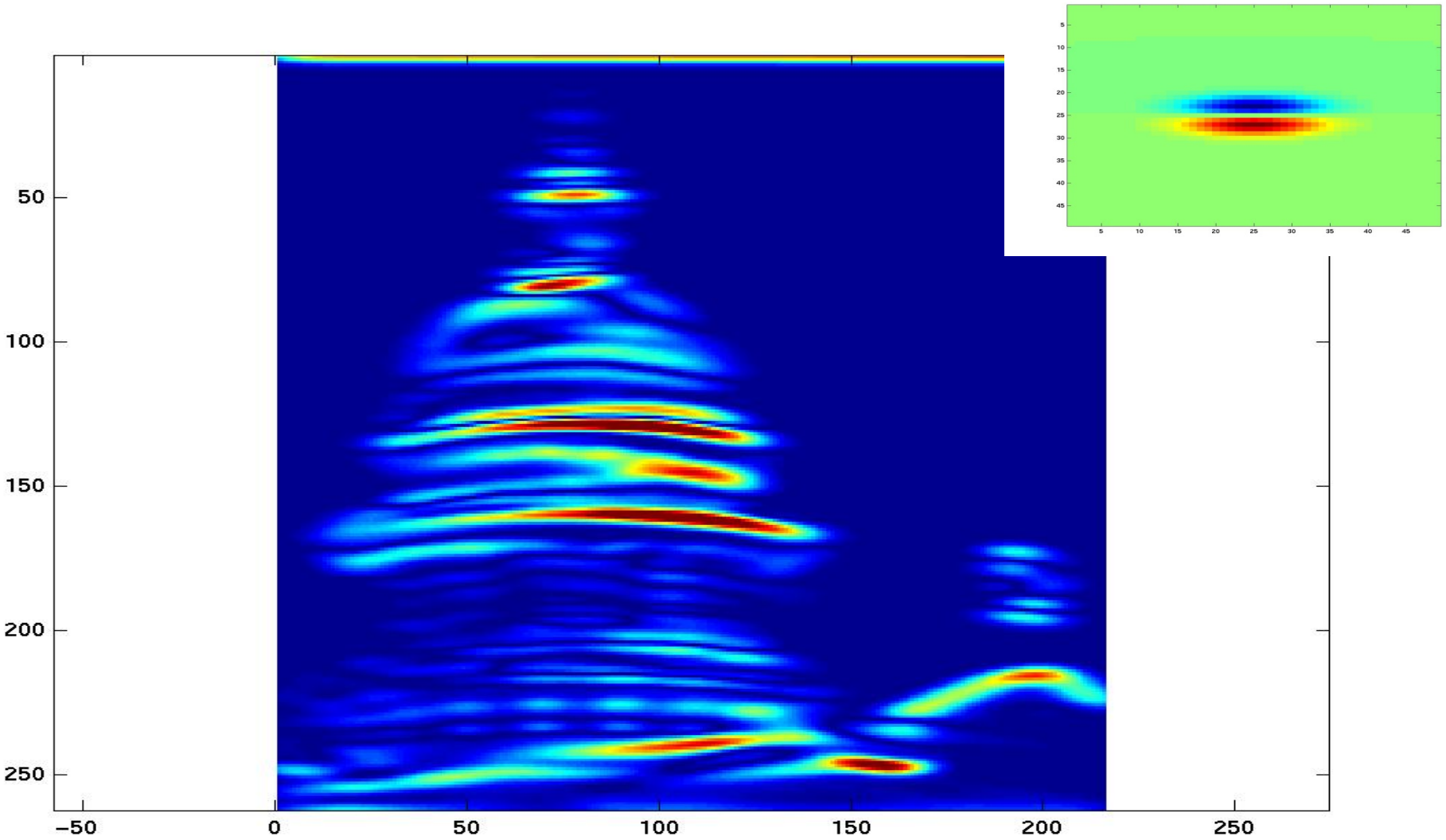
Kristen Grauman



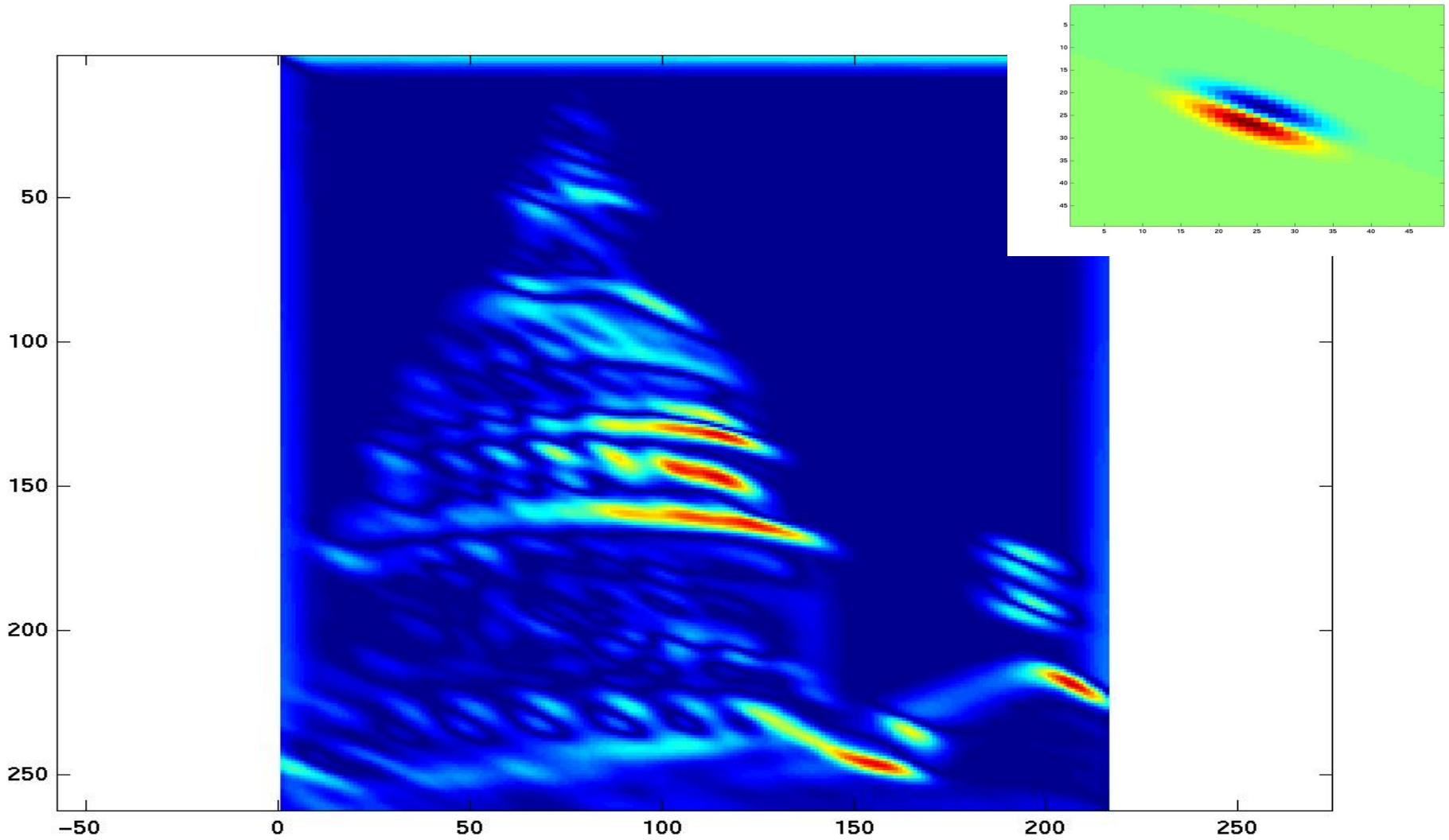
Kristen Grauman



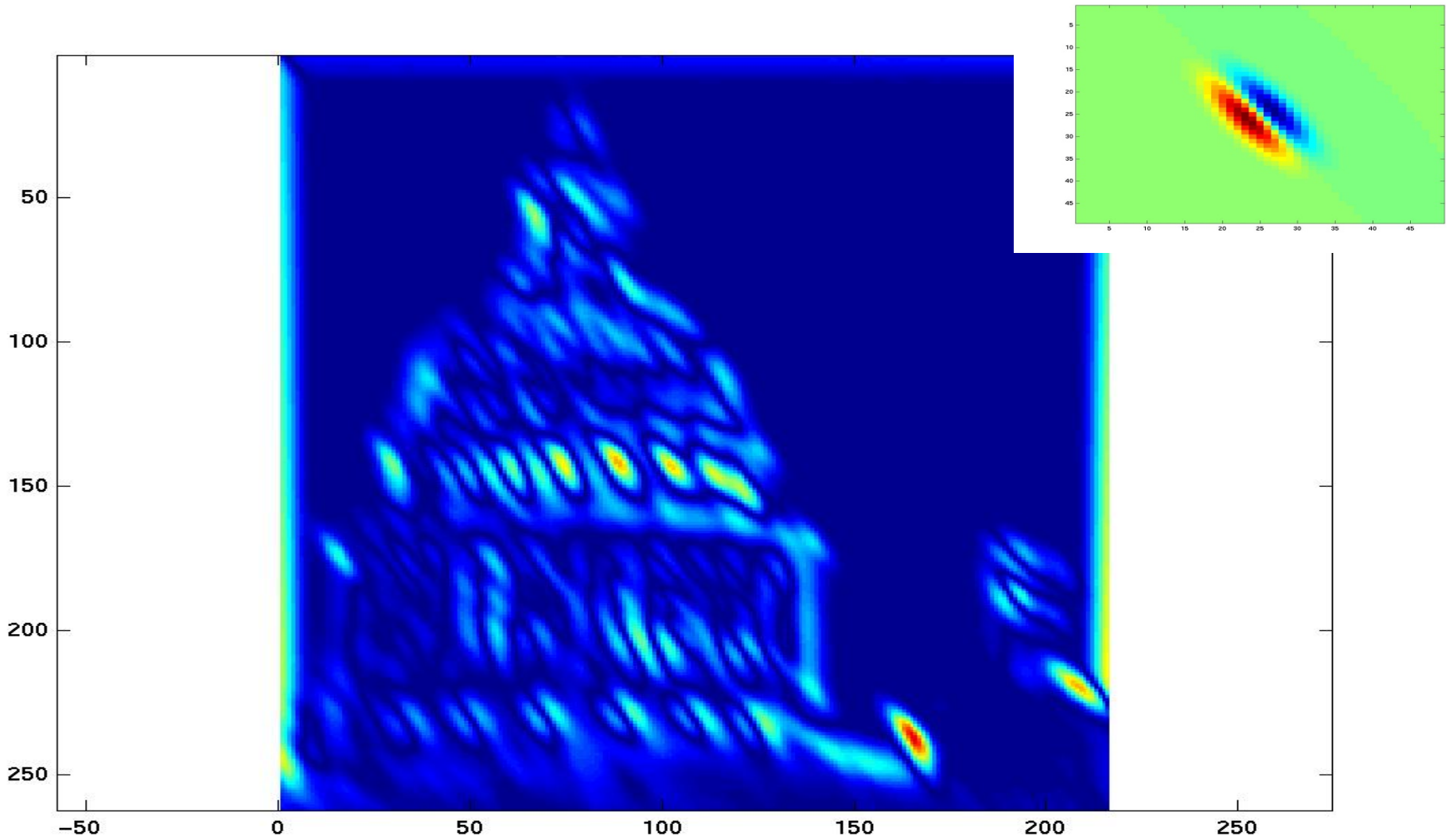
Kristen Grauman



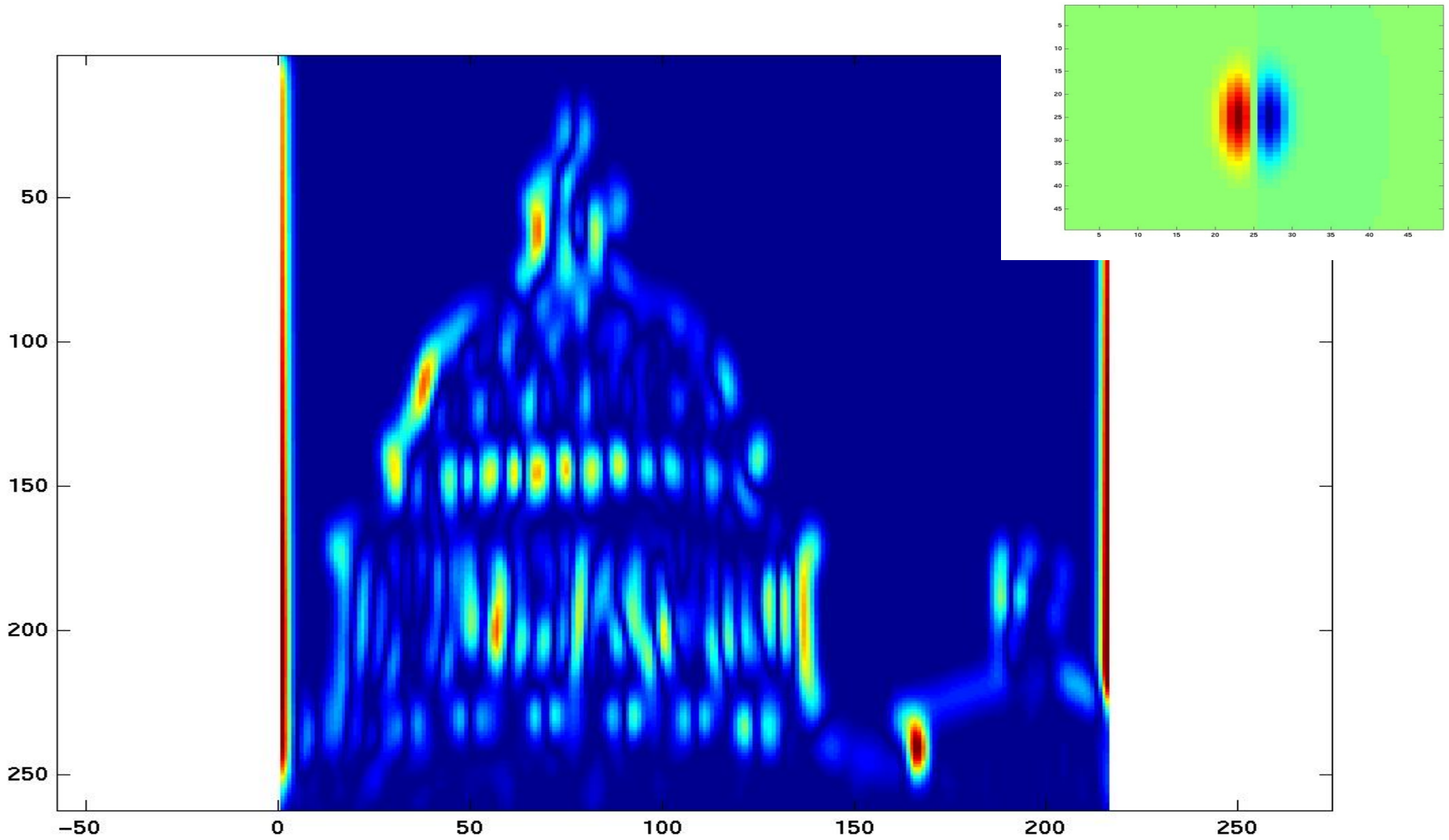
Kristen Grauman



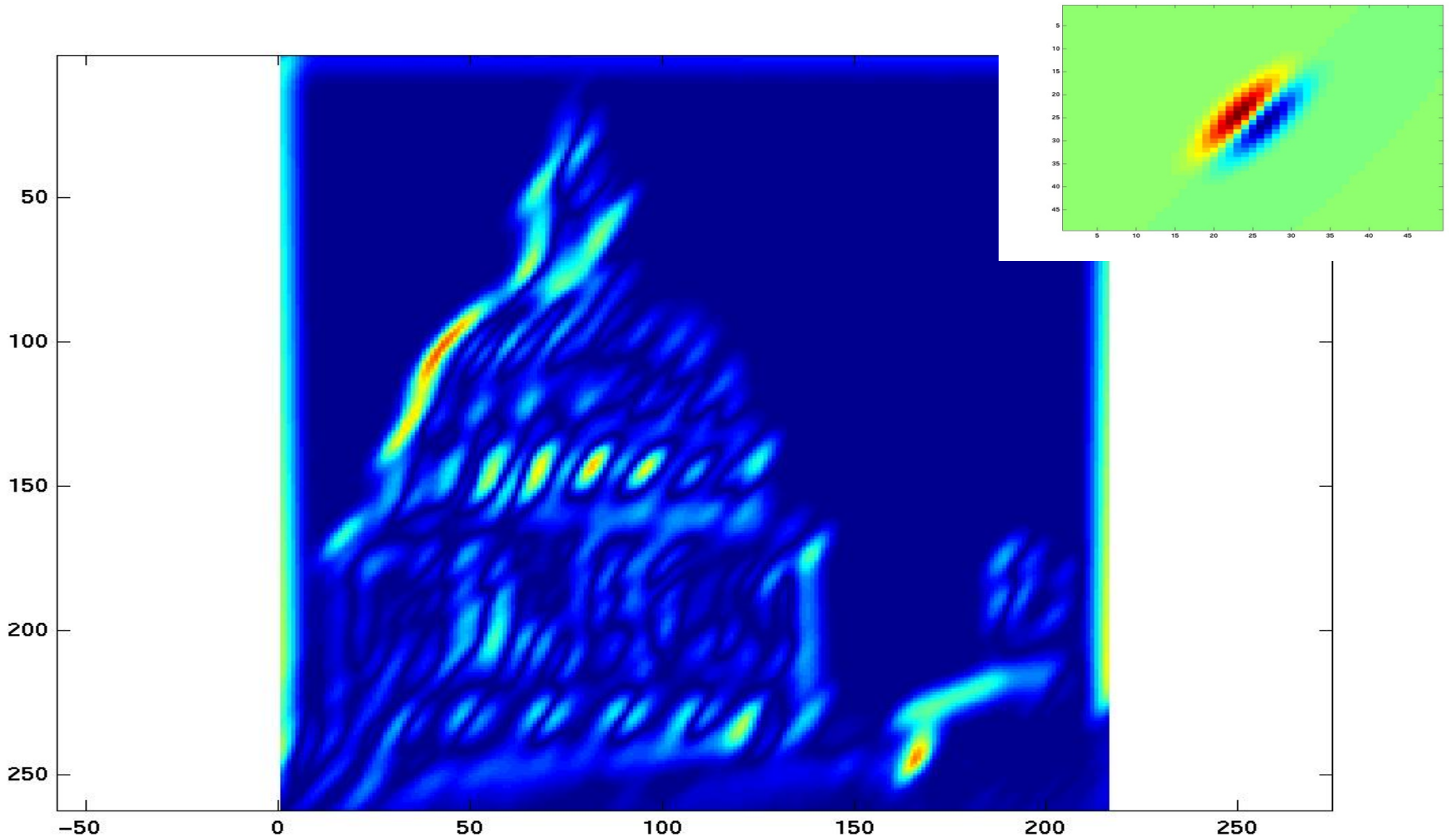
Kristen Grauman



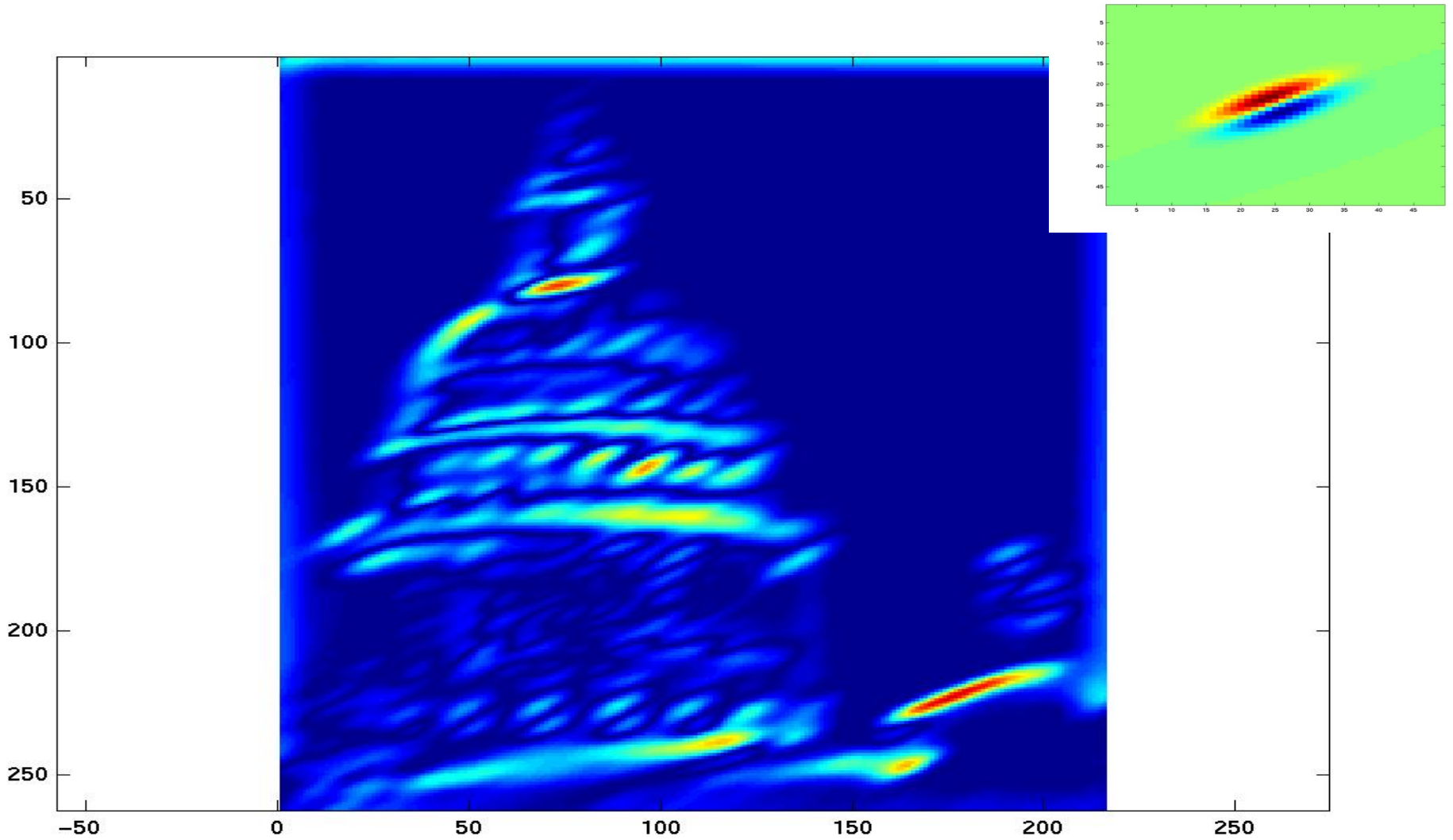
Kristen Grauman



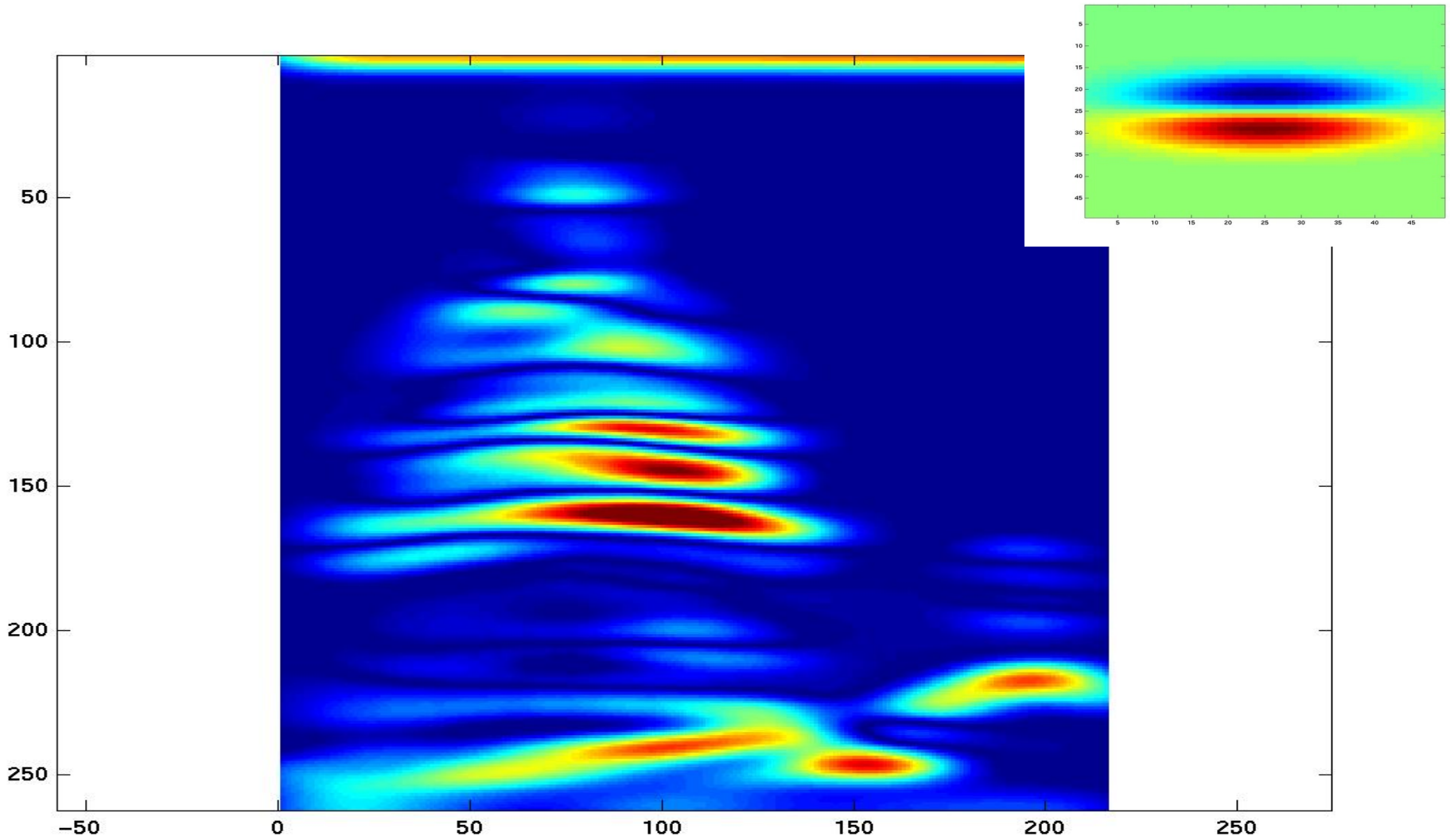
Kristen Grauman



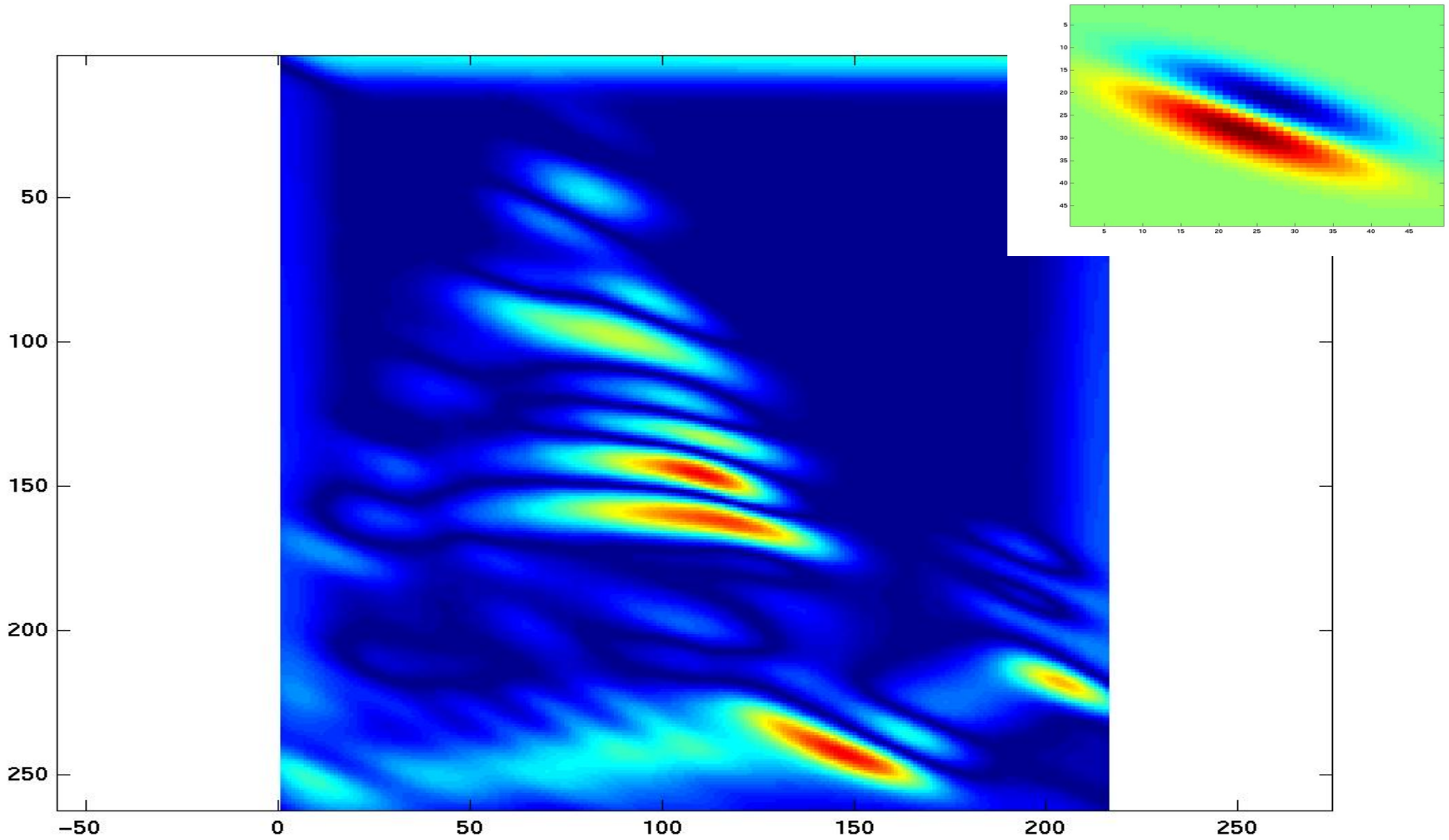
Kristen Grauman



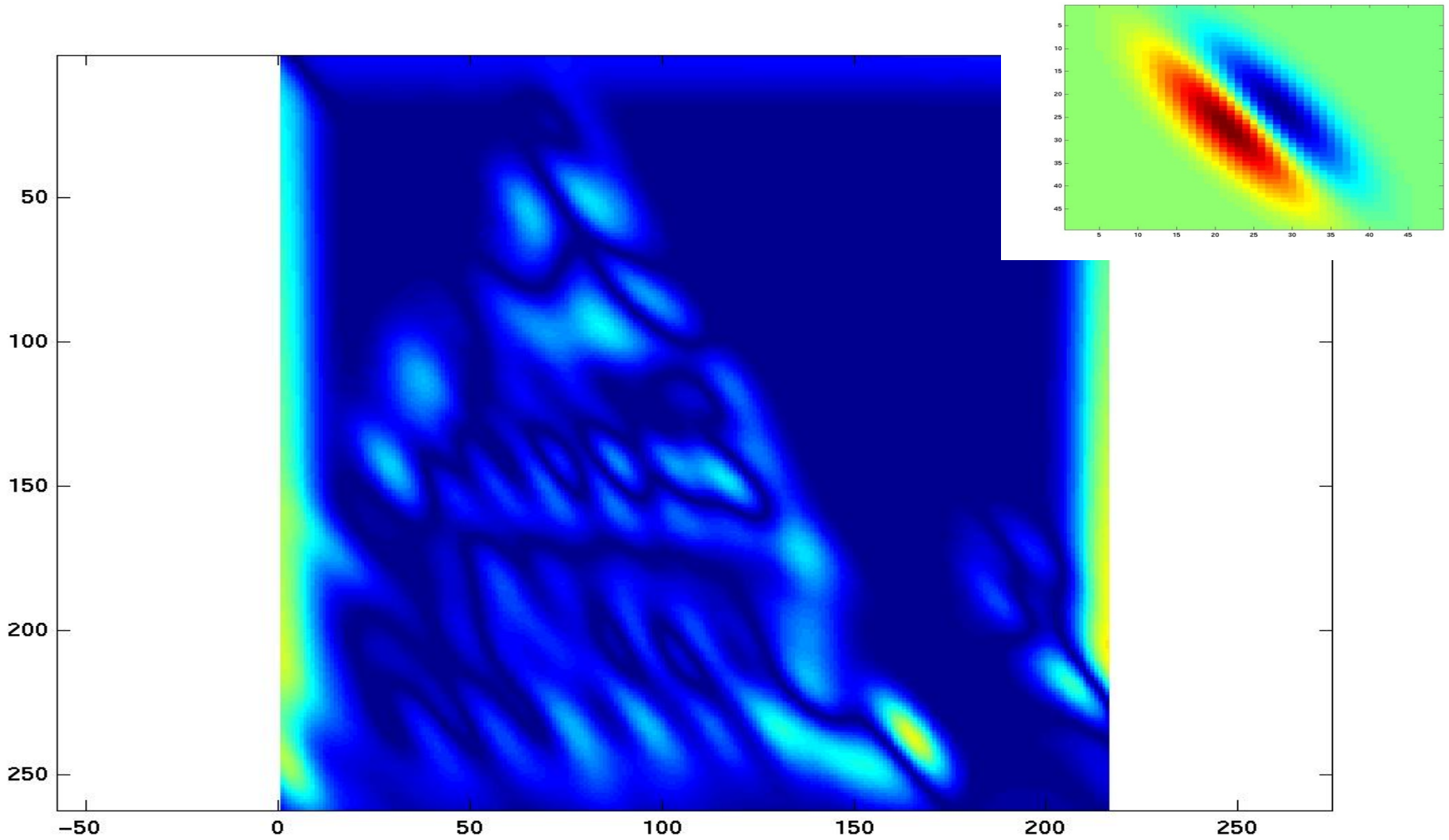
Kristen Grauman



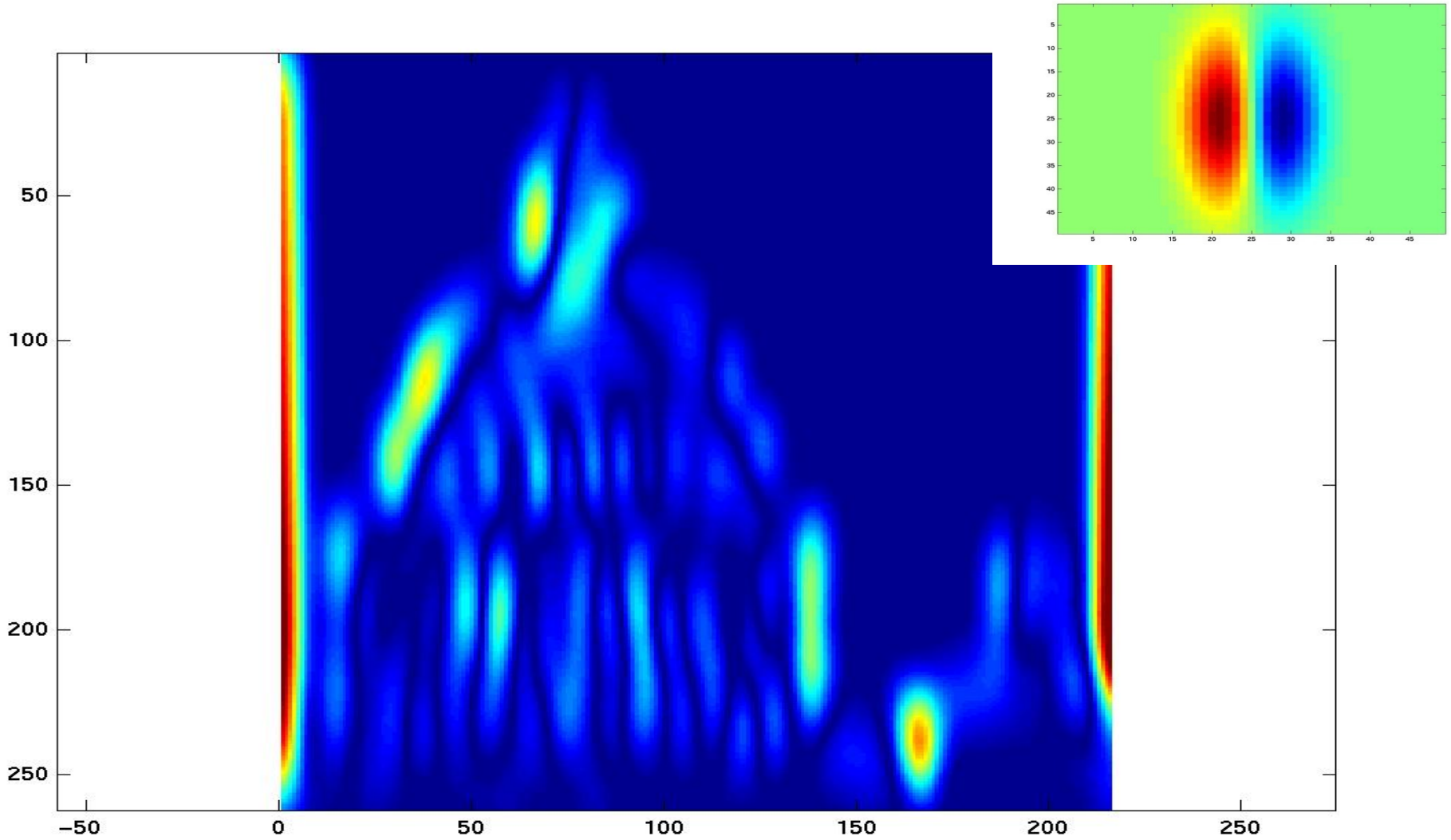
Kristen Grauman



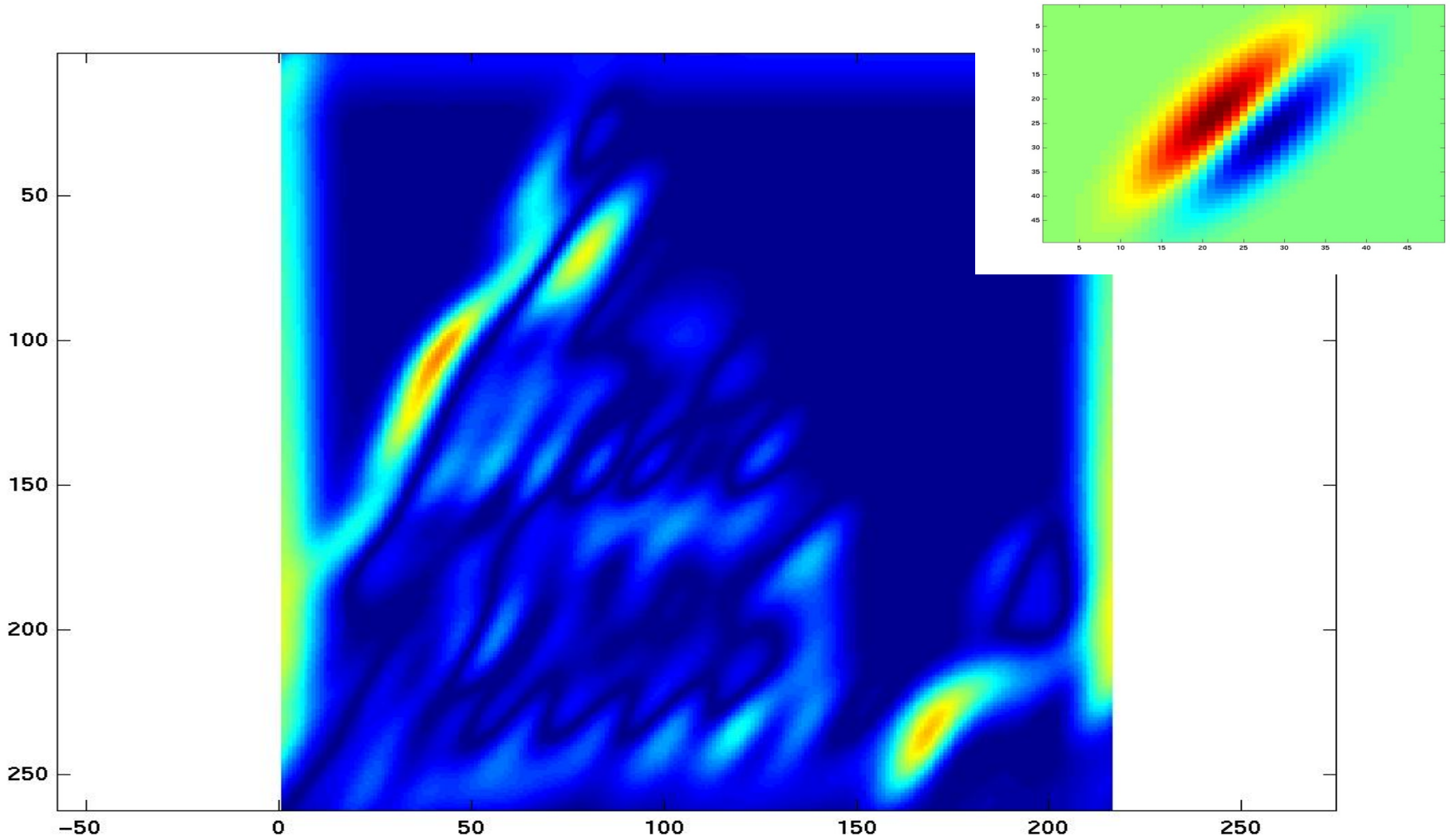
Kristen Grauman



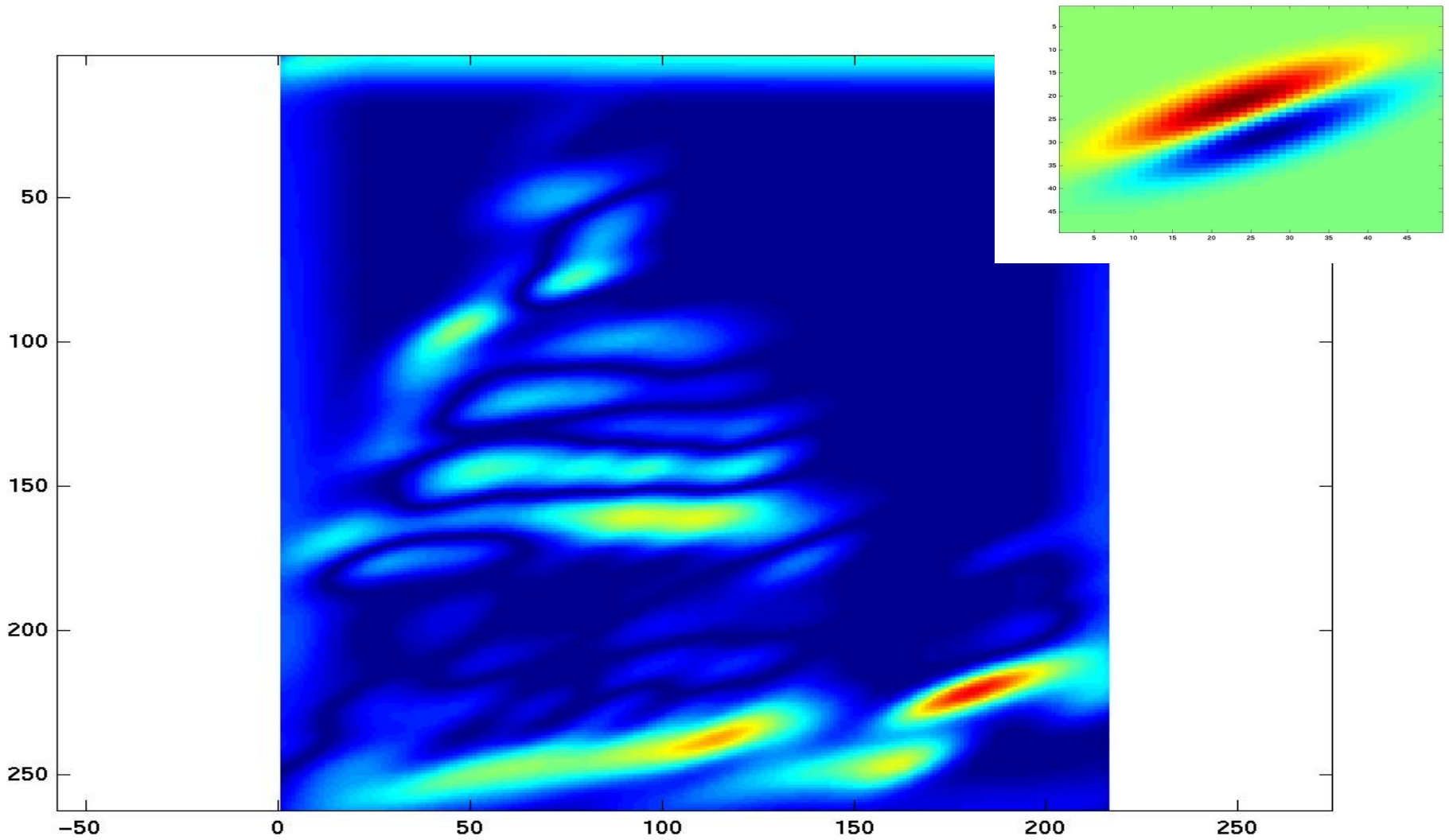
Kristen Grauman



Kristen Grauman



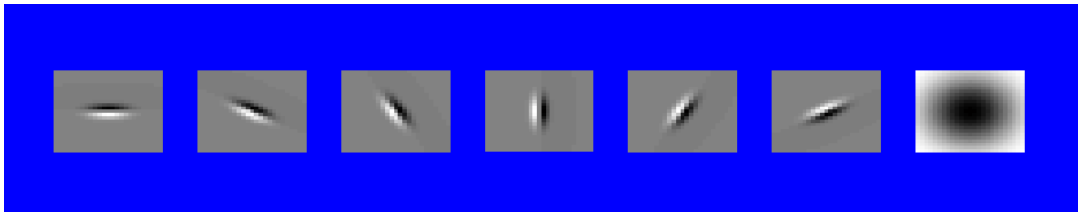
Kristen Grauman



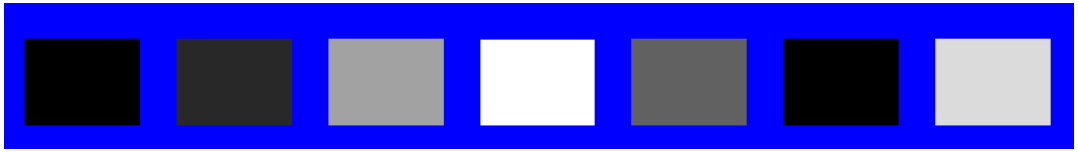
Kristen Grauman

Can you match the texture to the response?

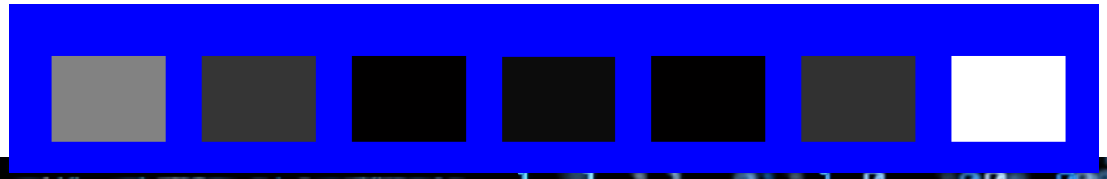
Filters



1

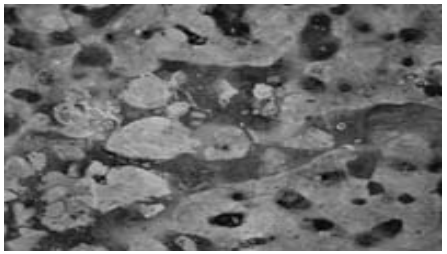


2



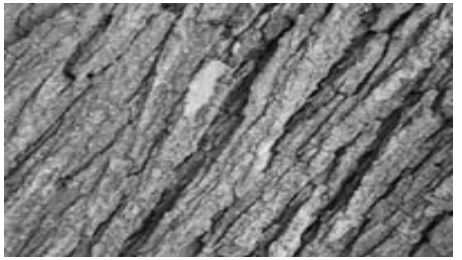
3

A



B

C



Data X

End of Section