# CS9840
# Machine Learning in Computer Vision
Olga Veksler

## Lecture 4

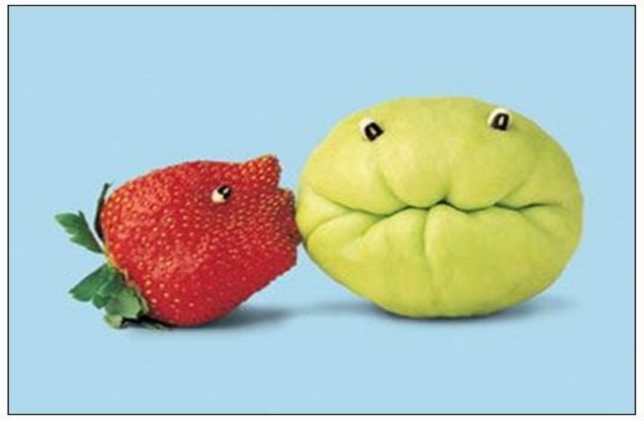
Image Representation

# *Outline*

- How to represent an image as a feature vector?
- Basic image features
  - intensity, color, gradients, response to filter(s)
  - dense (at each pixel)
  - sparse (at a subset of locations)
- Representations
  - pixelwise
  - histogram
    - Global vs. Local histograms
    - Spatial pyramids

# Basic Image Features

- Given image **I**, first compute *basic image features* or *feature responses*
- Then consolidate basic image features into a feature vector ***x*** that represents image **I**
- Simplest basic image feature: intensity of a pixel
  - not enough for most applications
- Other basic image features commonly used:



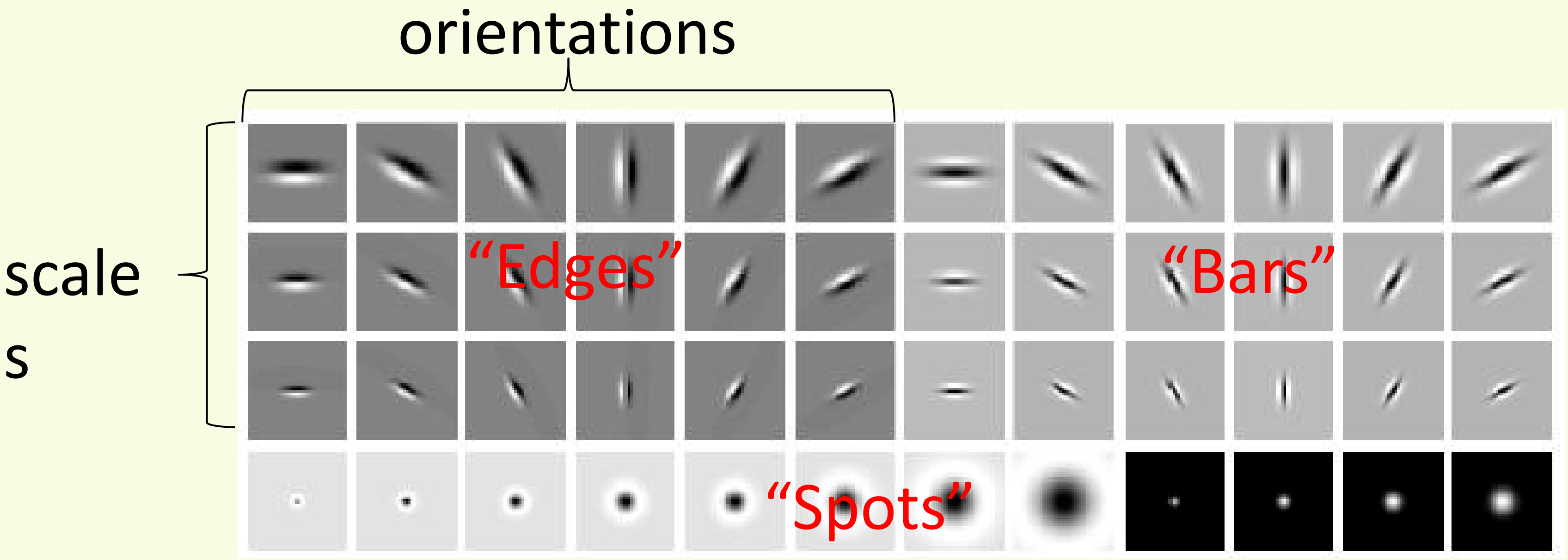
Color:
3 values per pixel

Edges :
1 or 2 values per pixel

Texture:
≈ 48 values per pixel

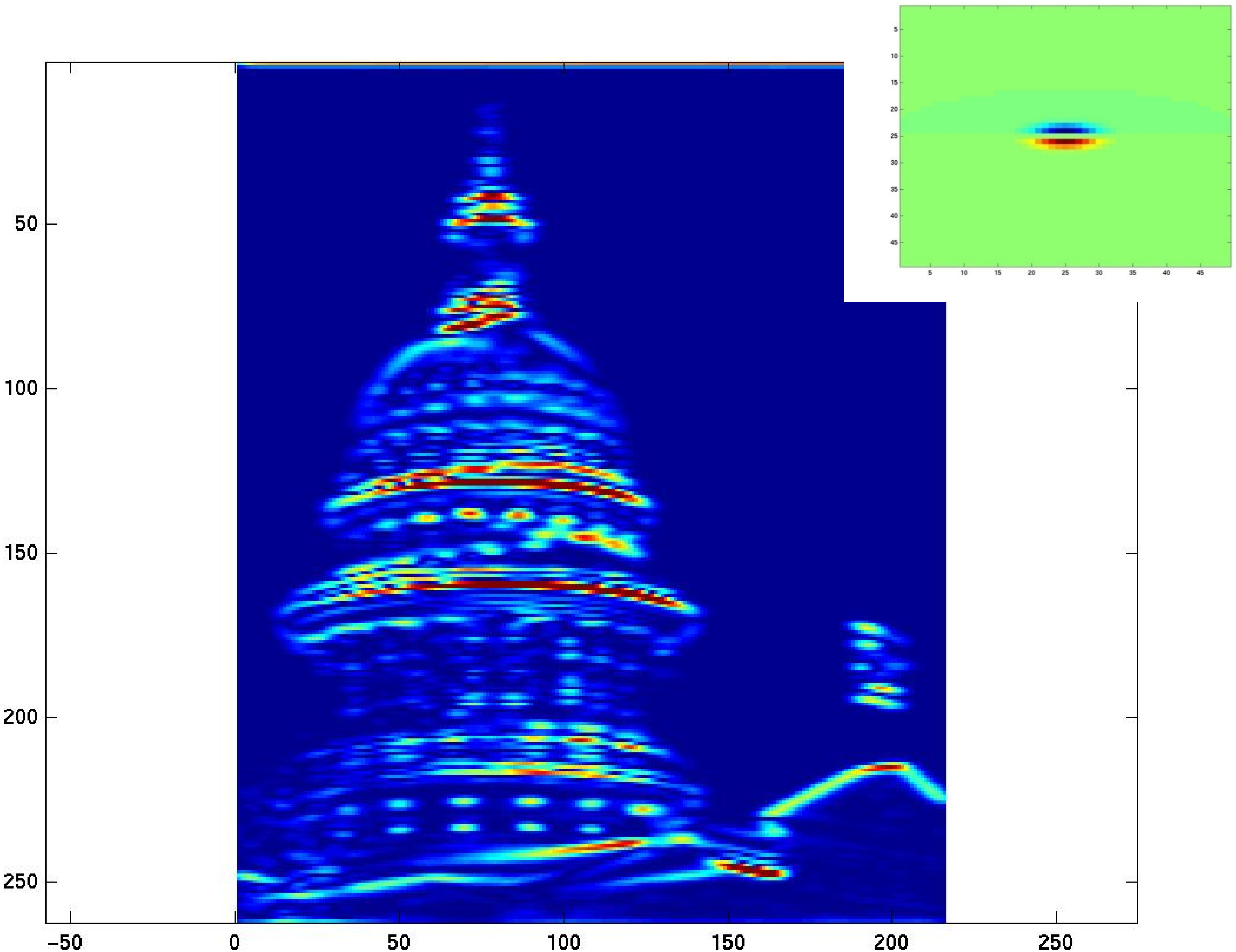
# Extracting Texture (Texture Responses)

- Texture filter bank:

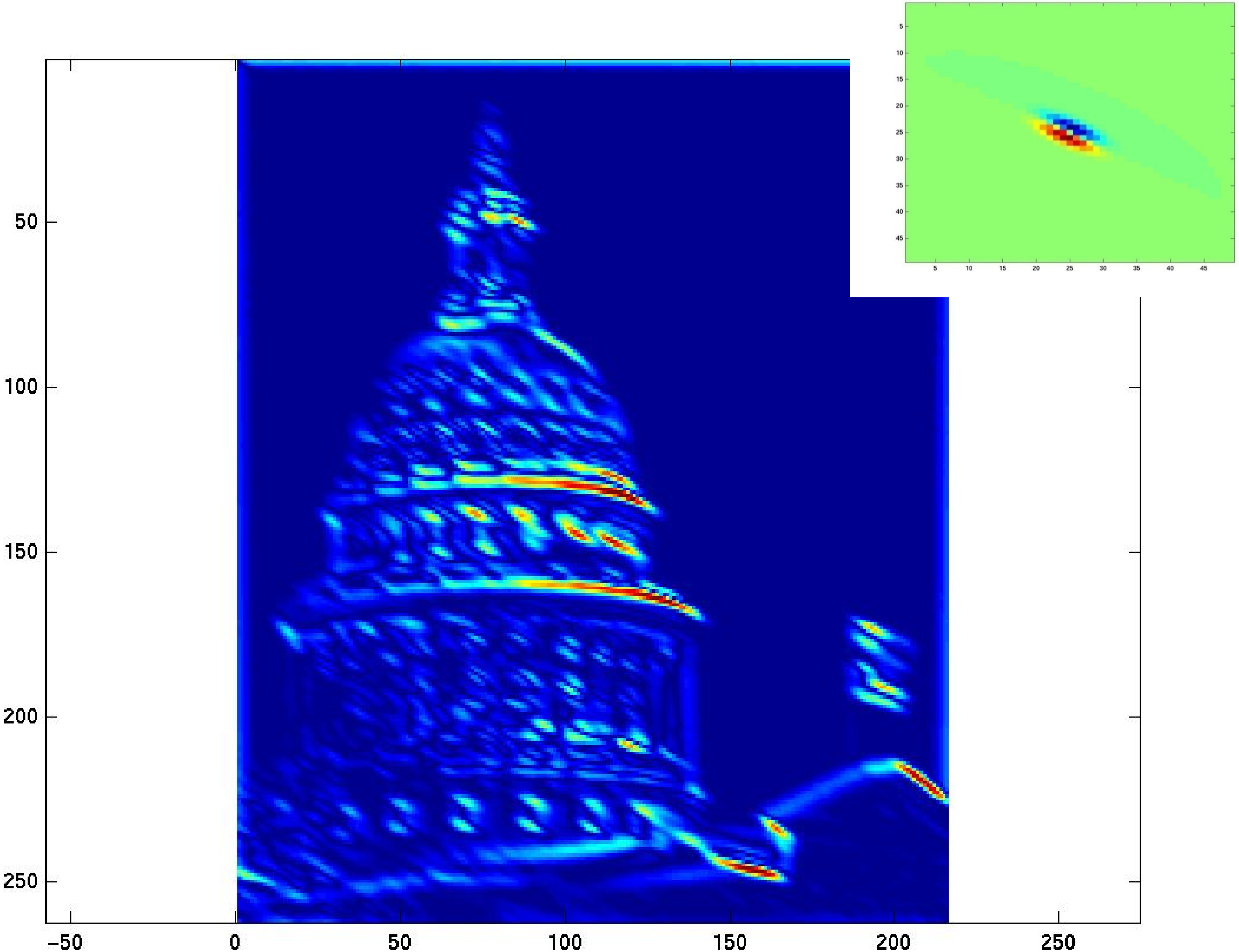



- Convolve image with each filter
  - 48 responses per pixel

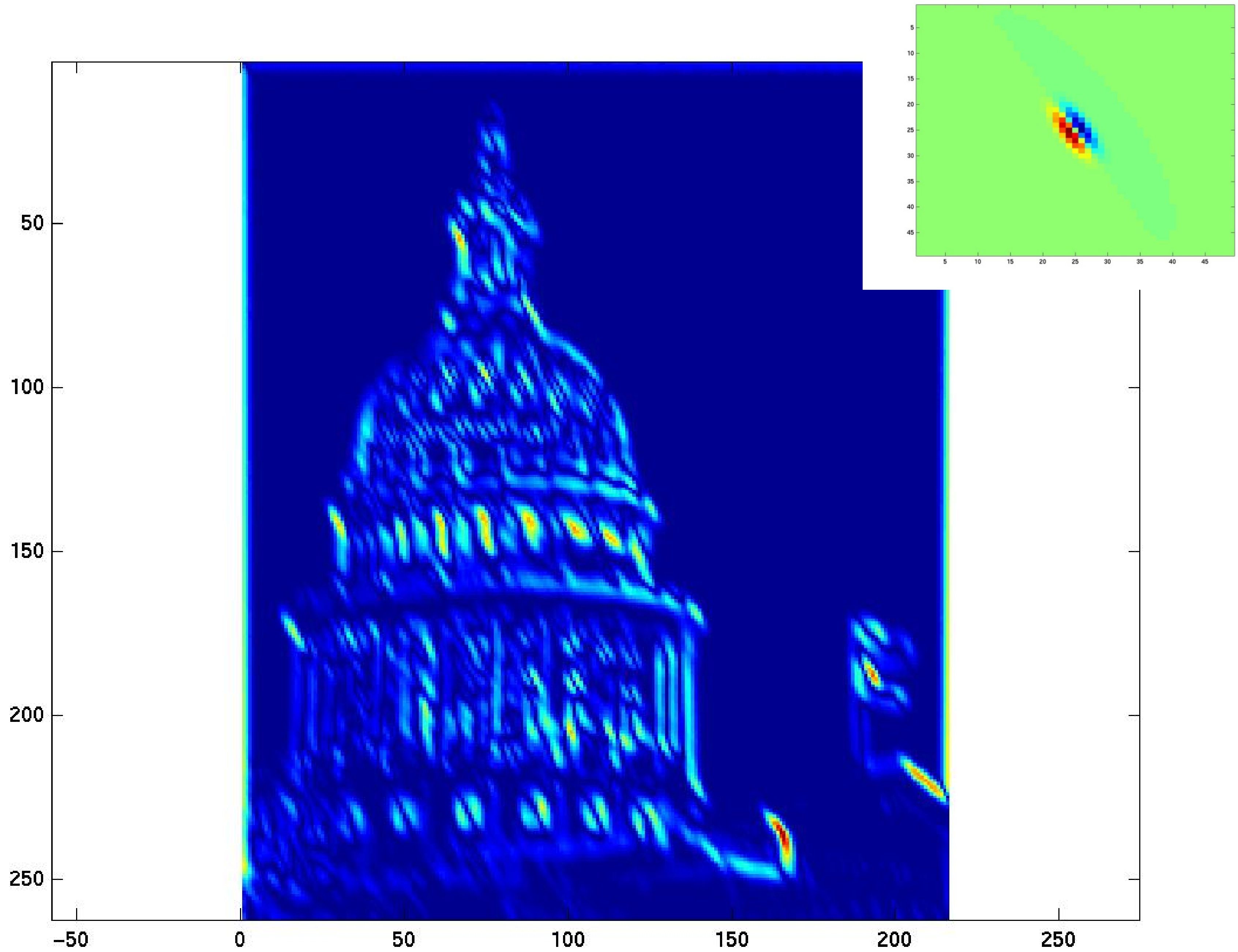
Kristen Grauman



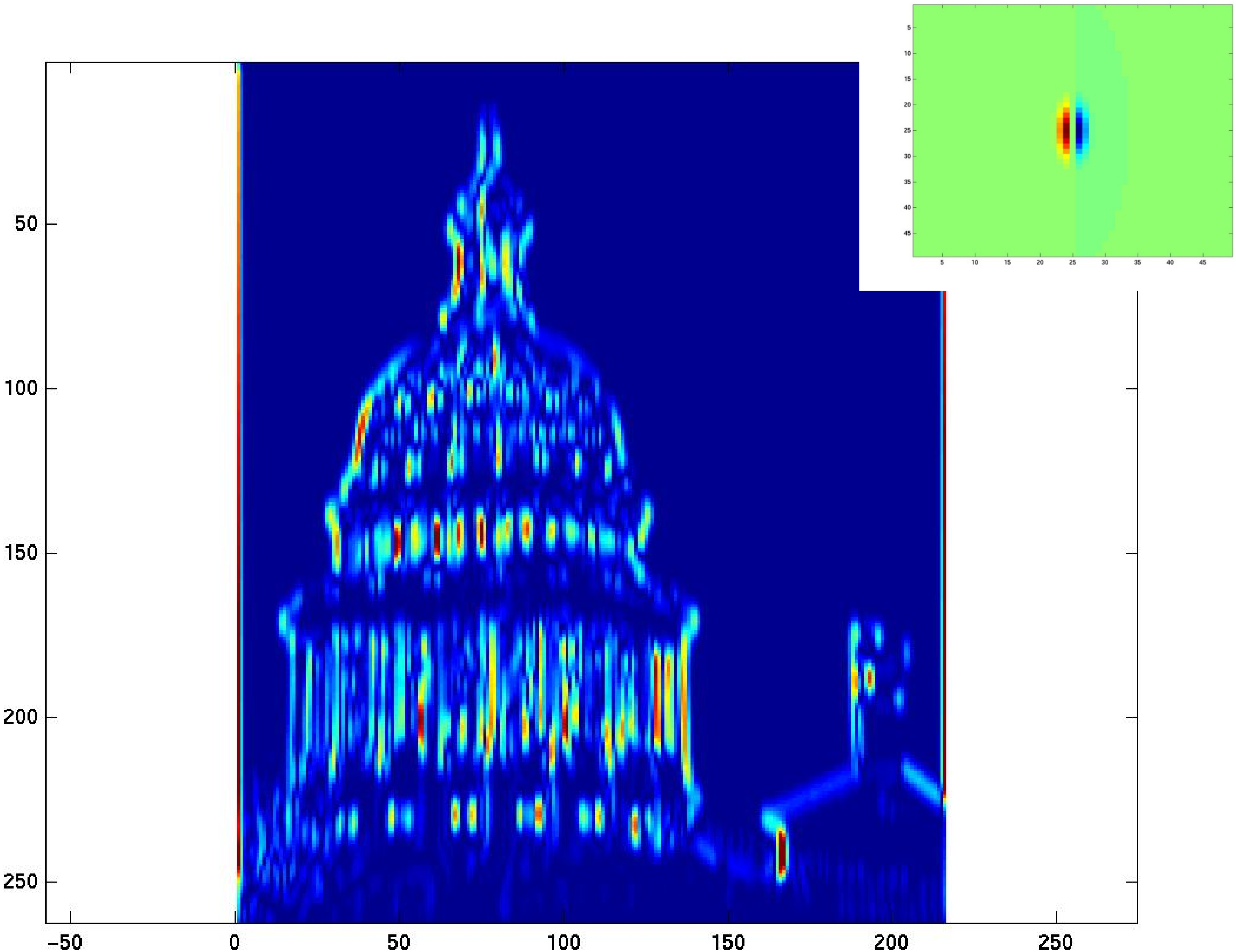

Kristen Grauman



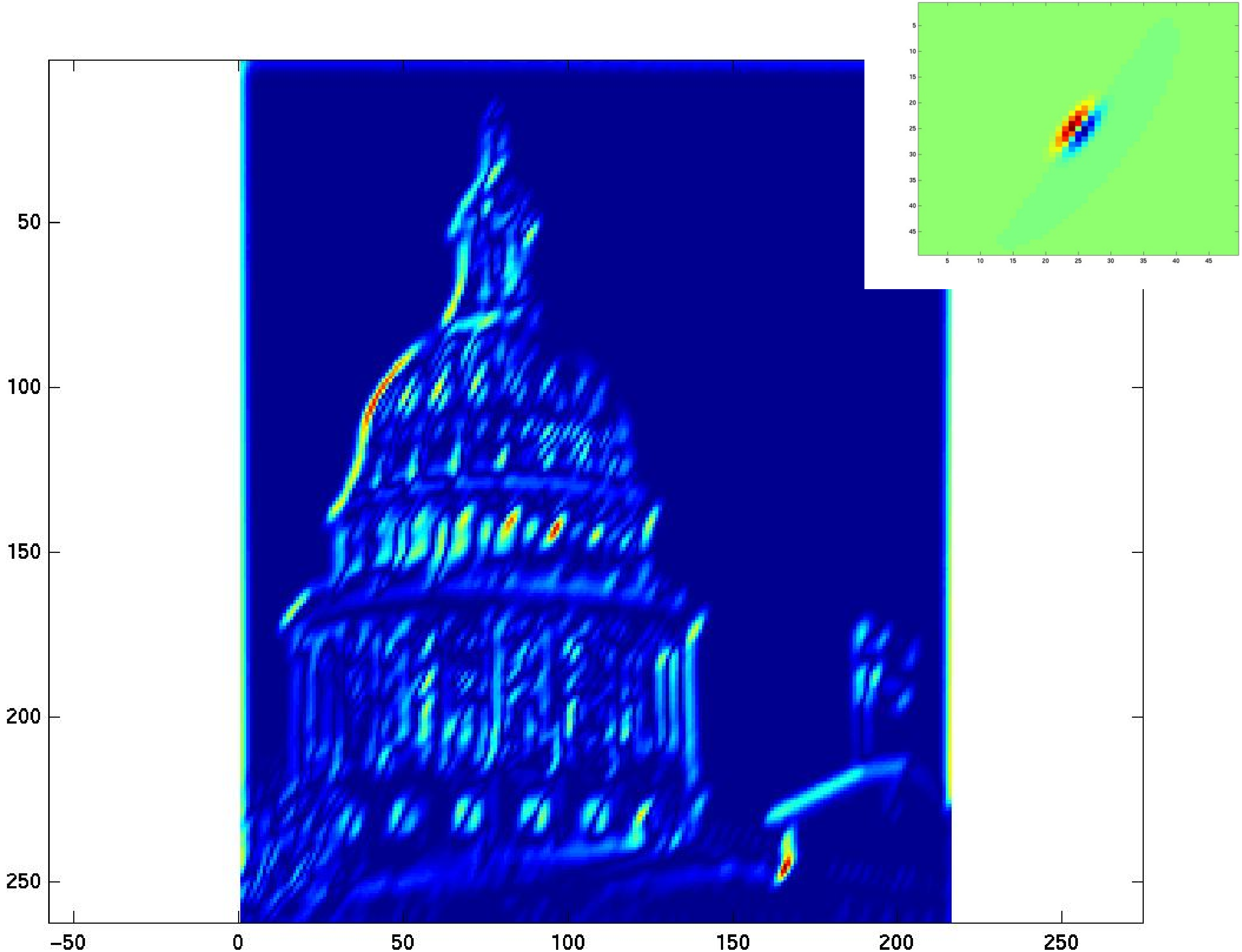

Kristen Grauman



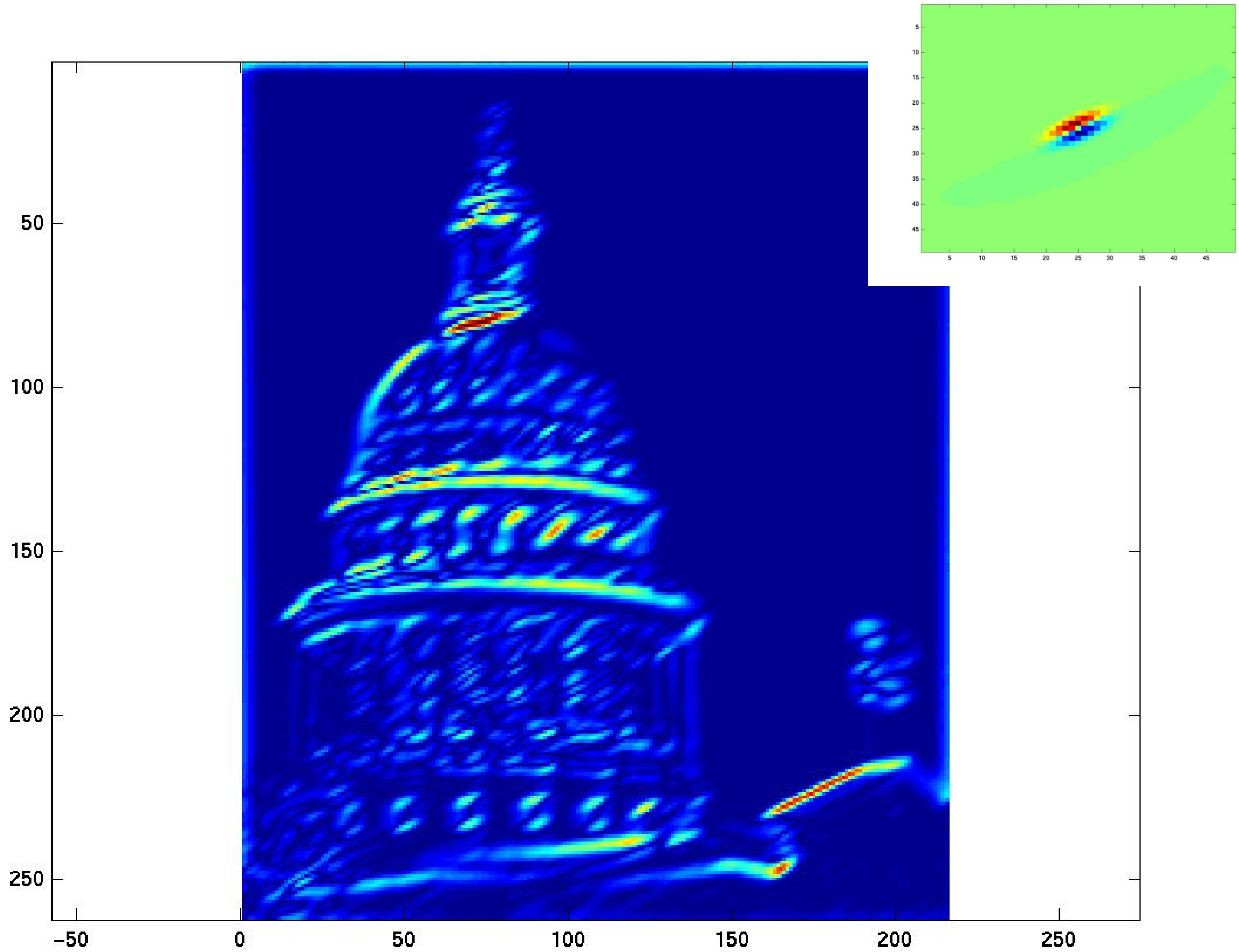

Kristen Grauman



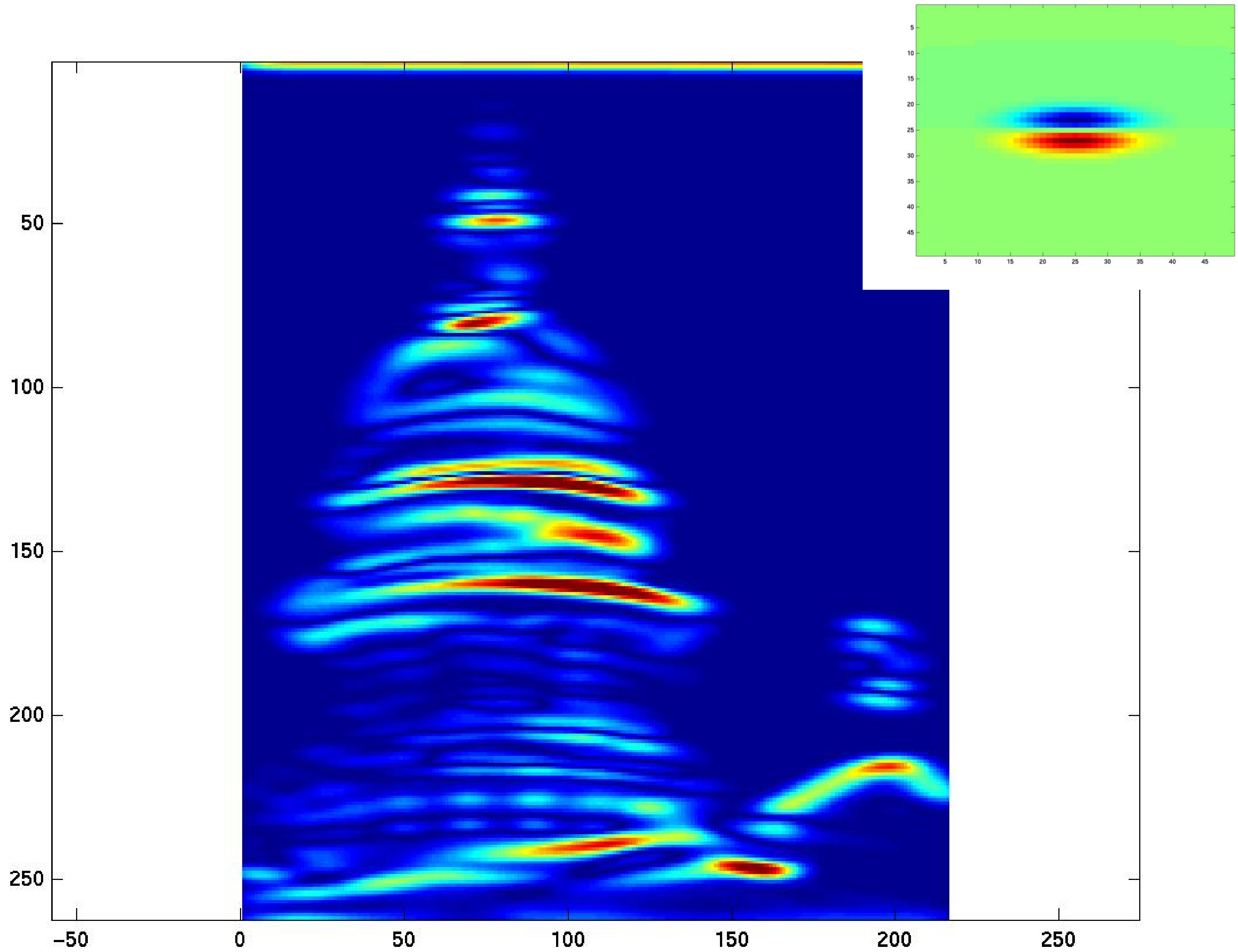

Kristen Grauman



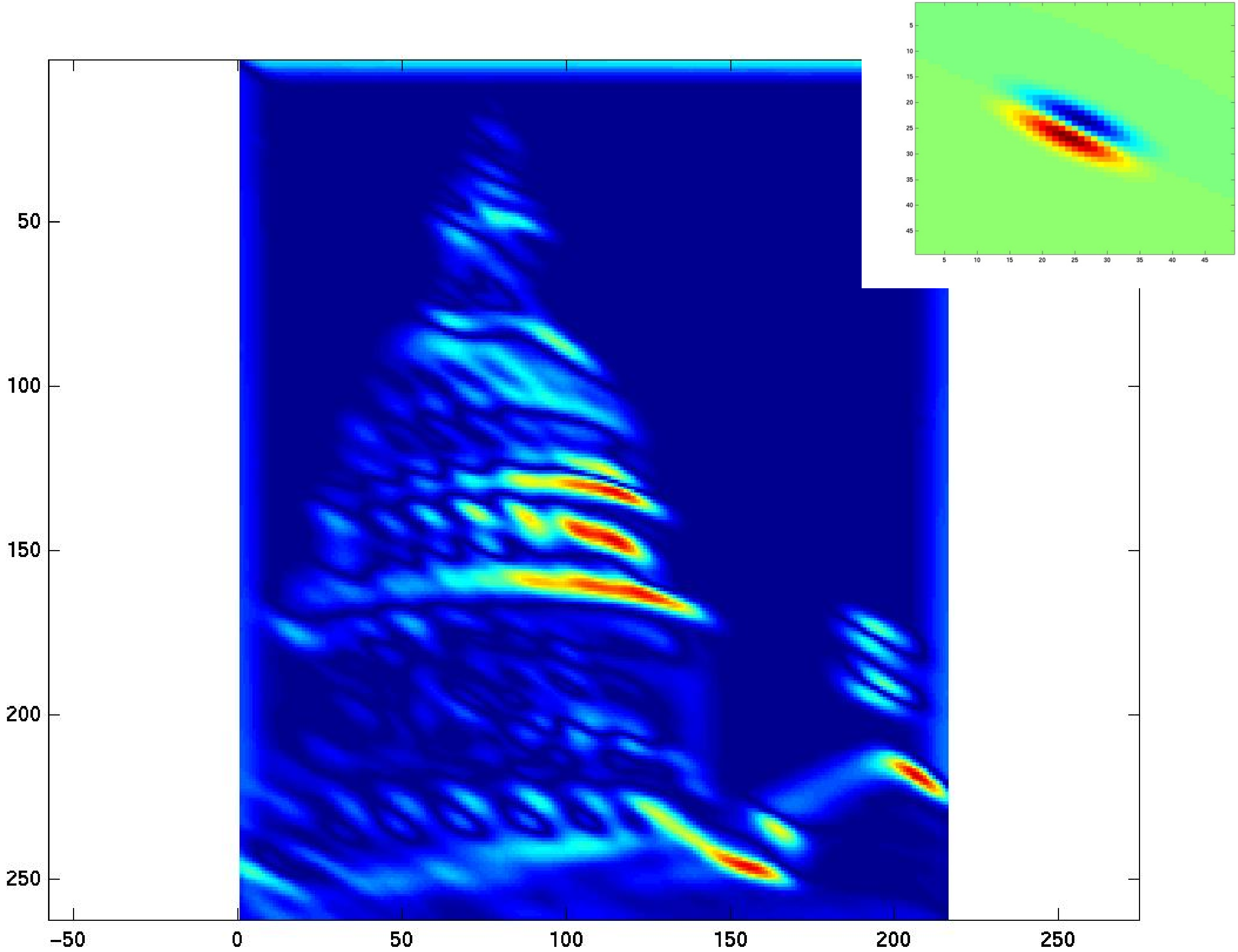

Kristen Grauman



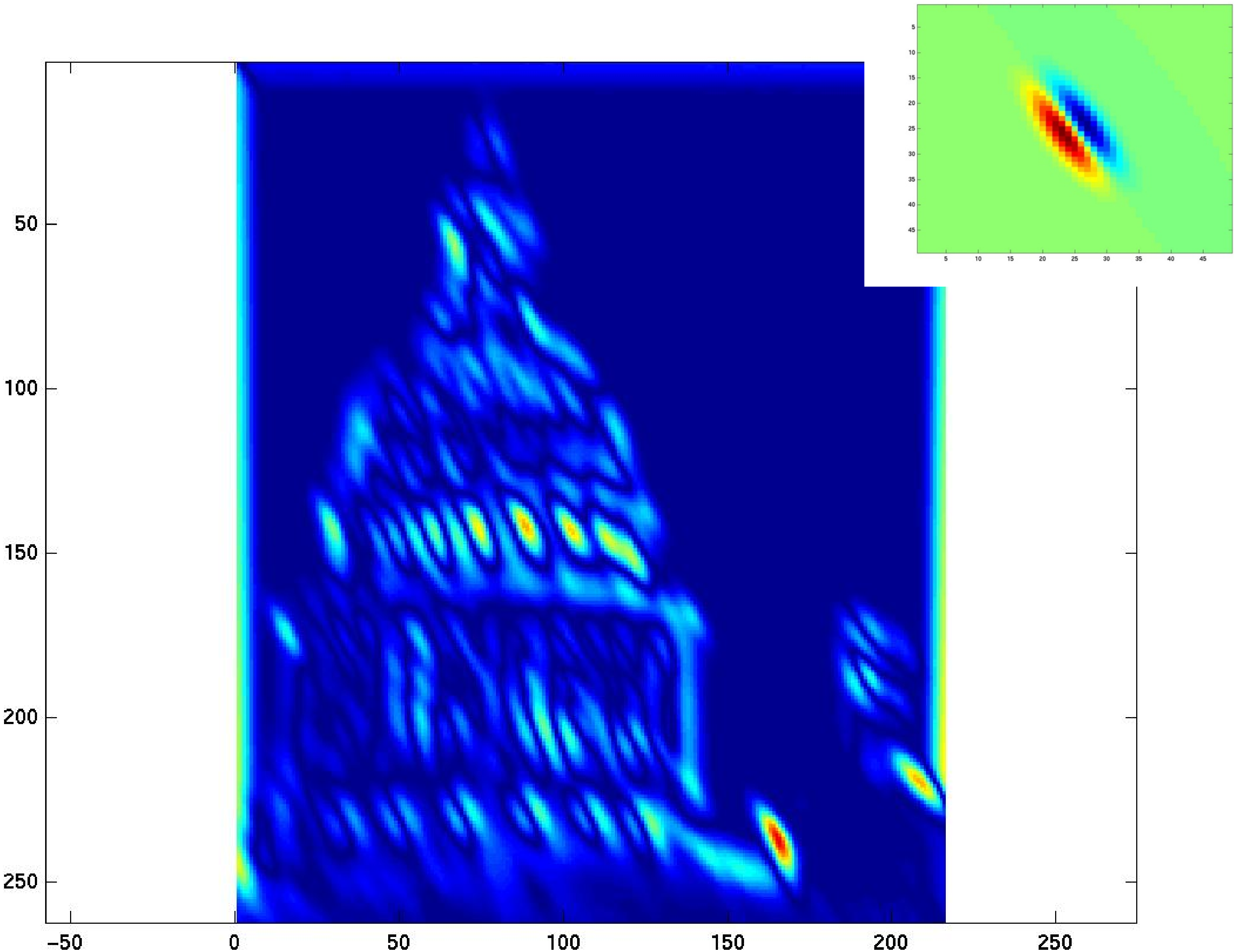

Kristen Grauman



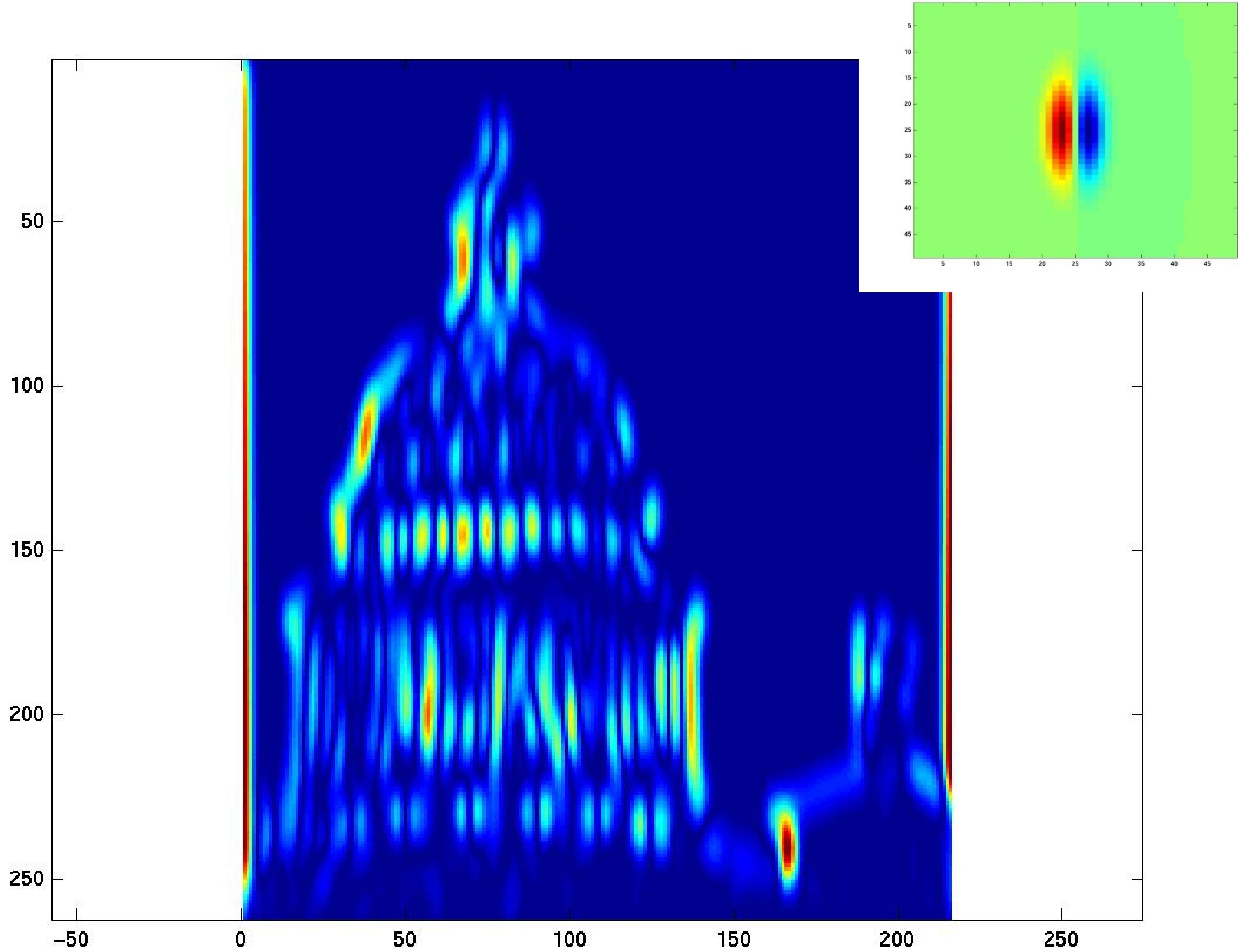

Kristen Grauman



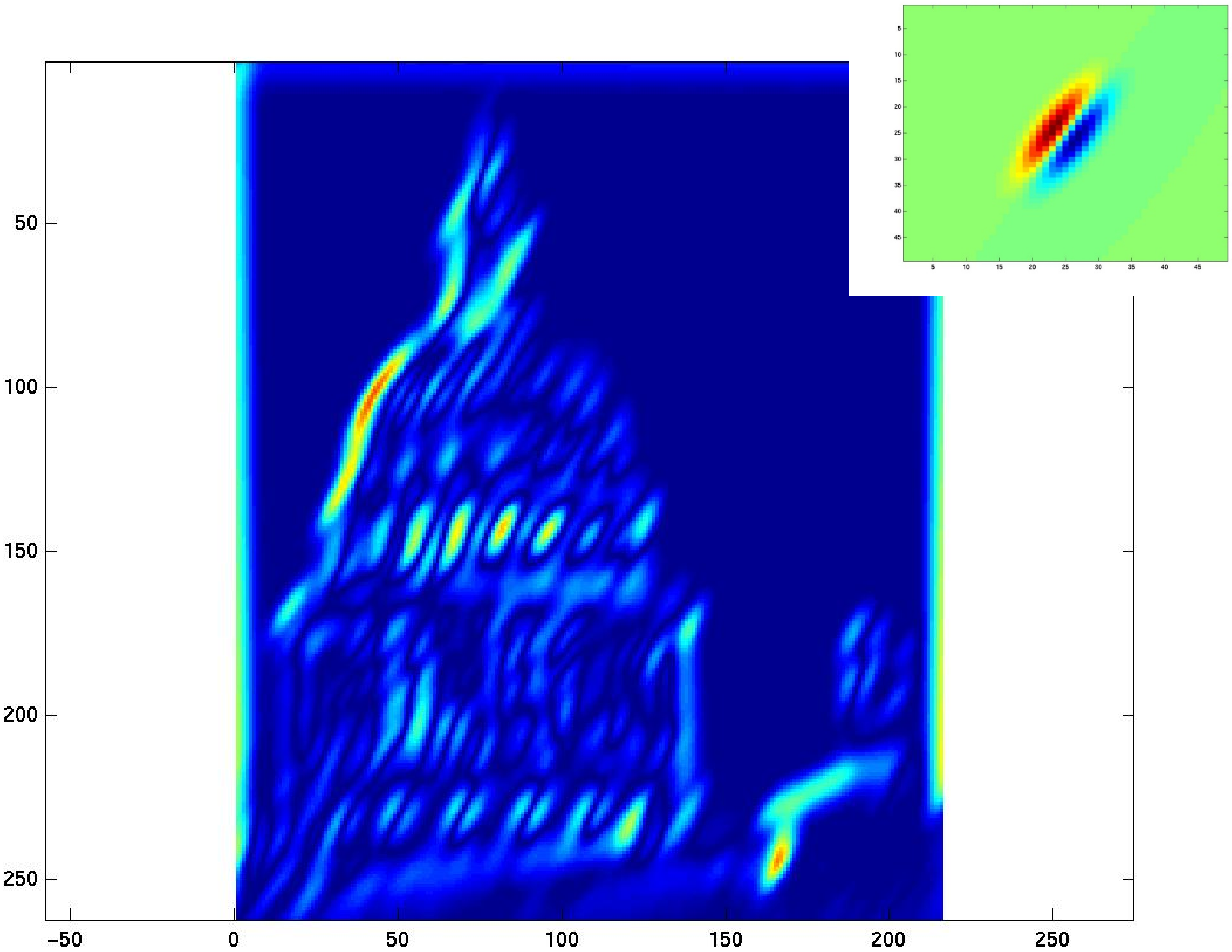

Kristen Grauman



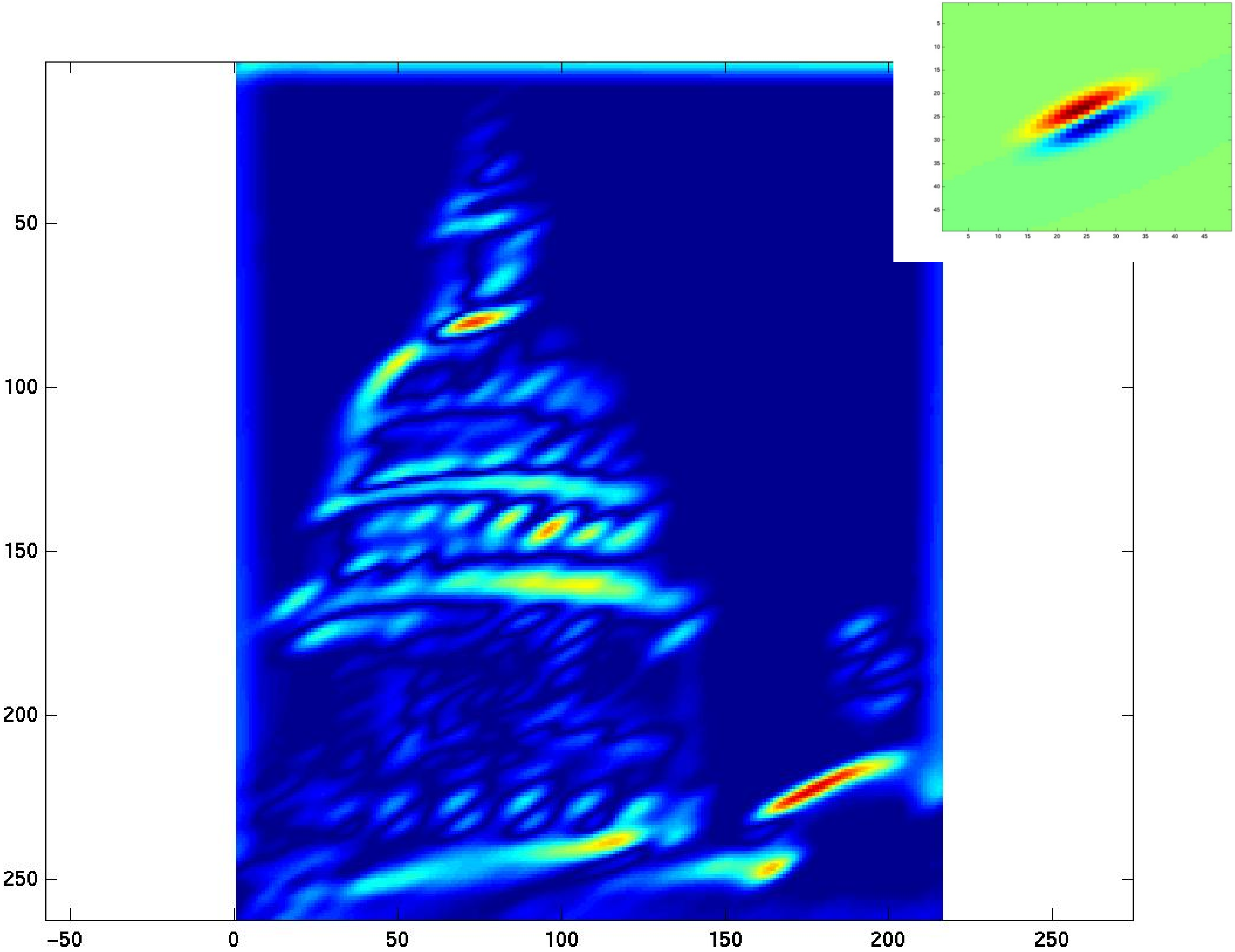

Kristen Grauman



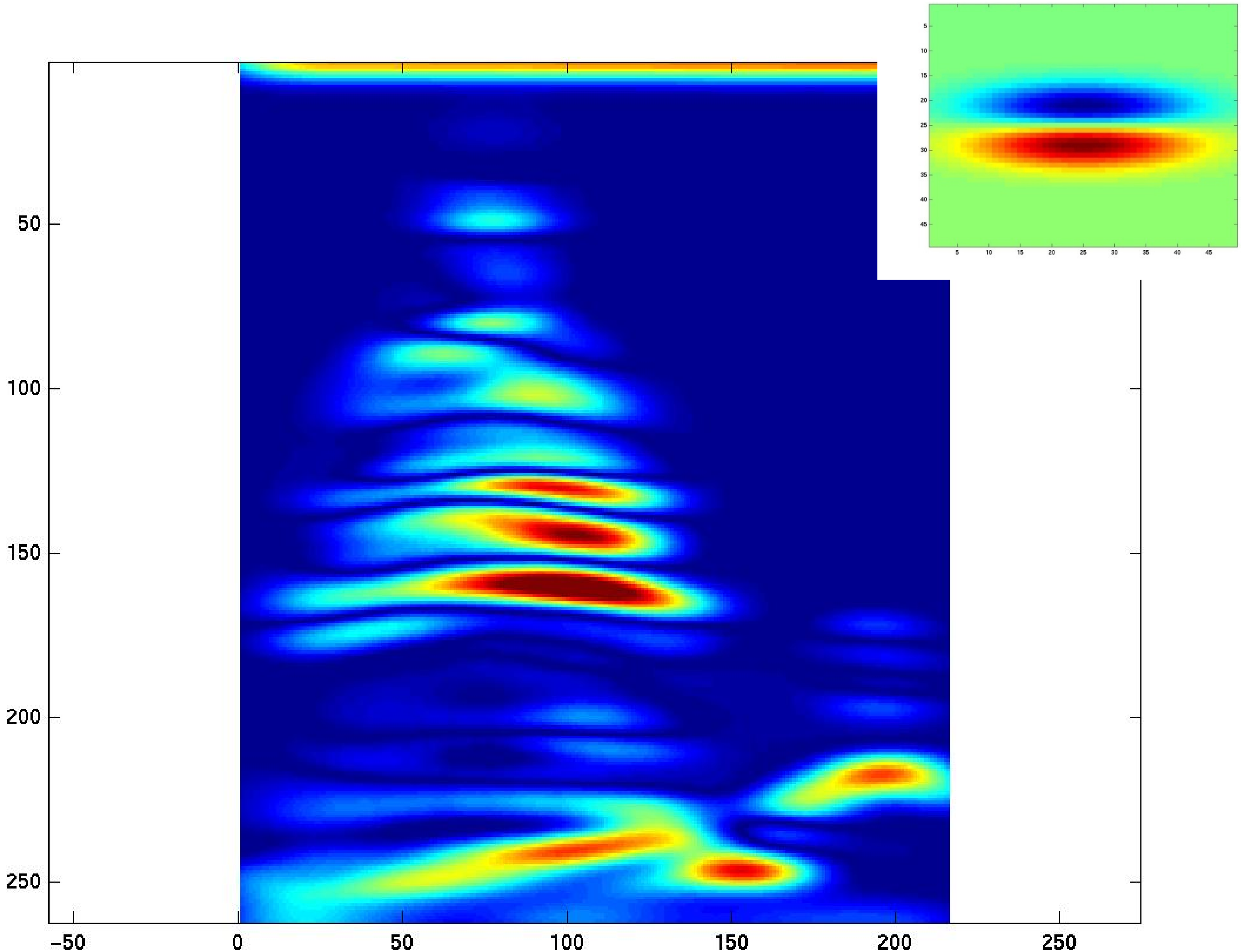

Kristen Grauman



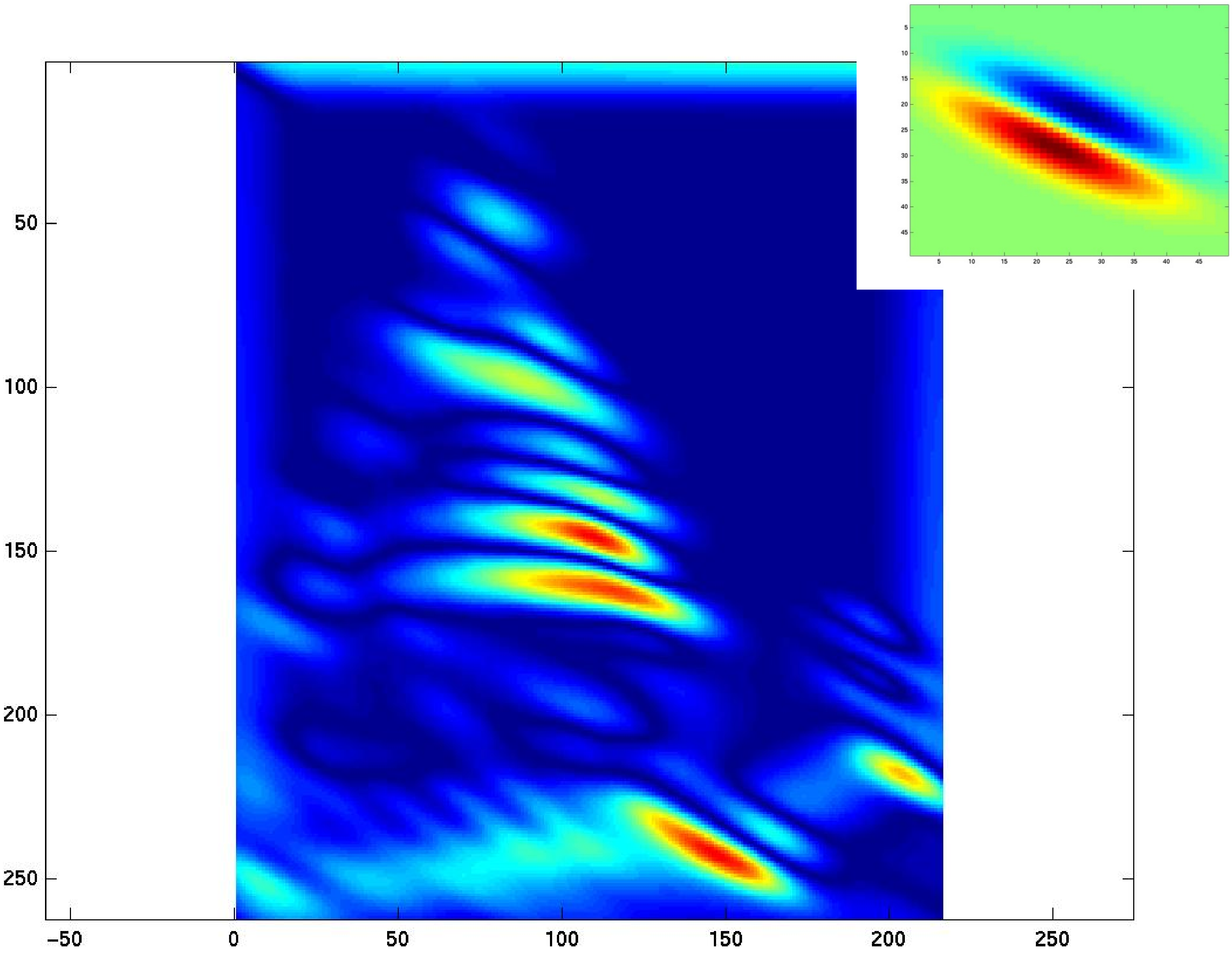

Kristen Grauman



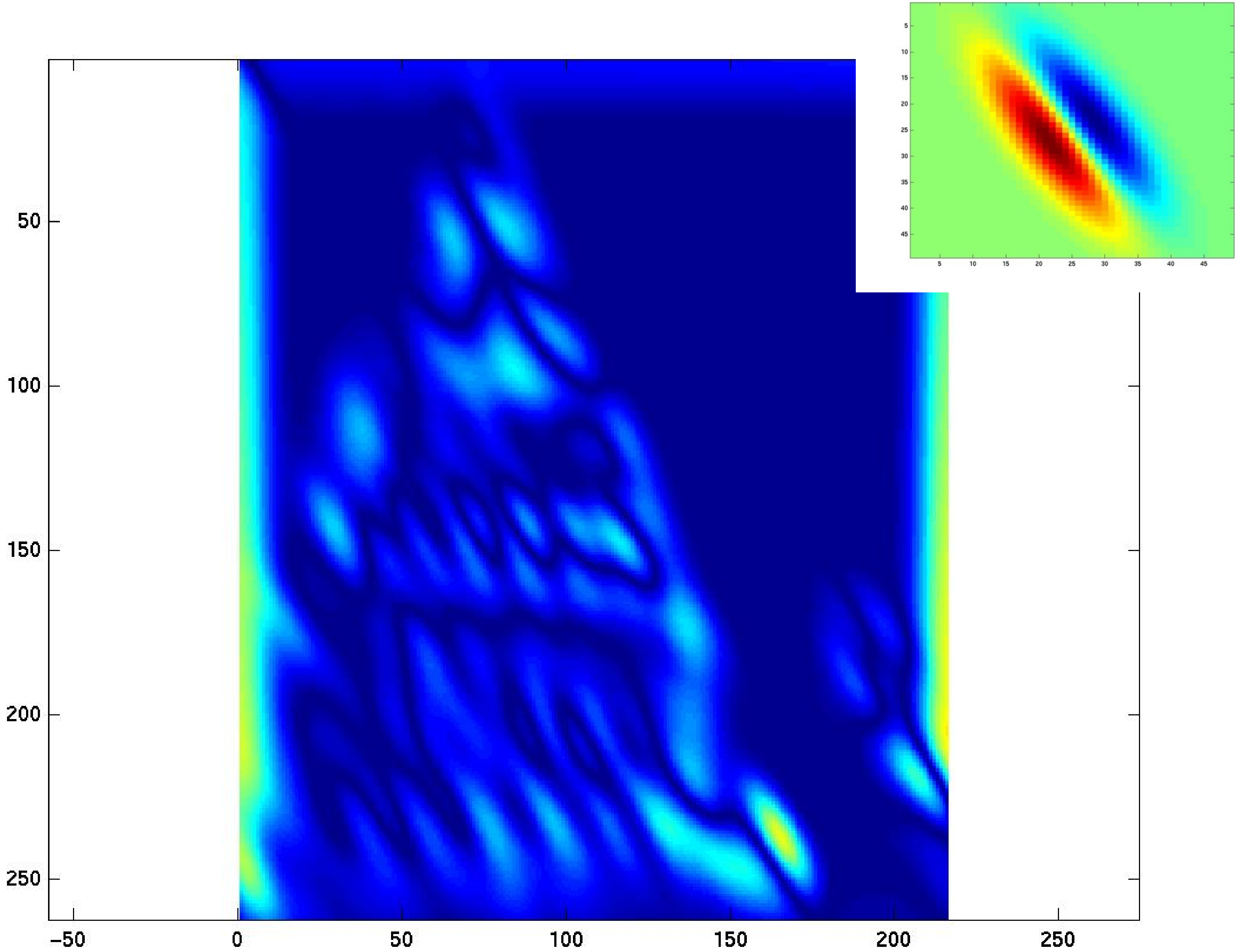

Kristen Grauman



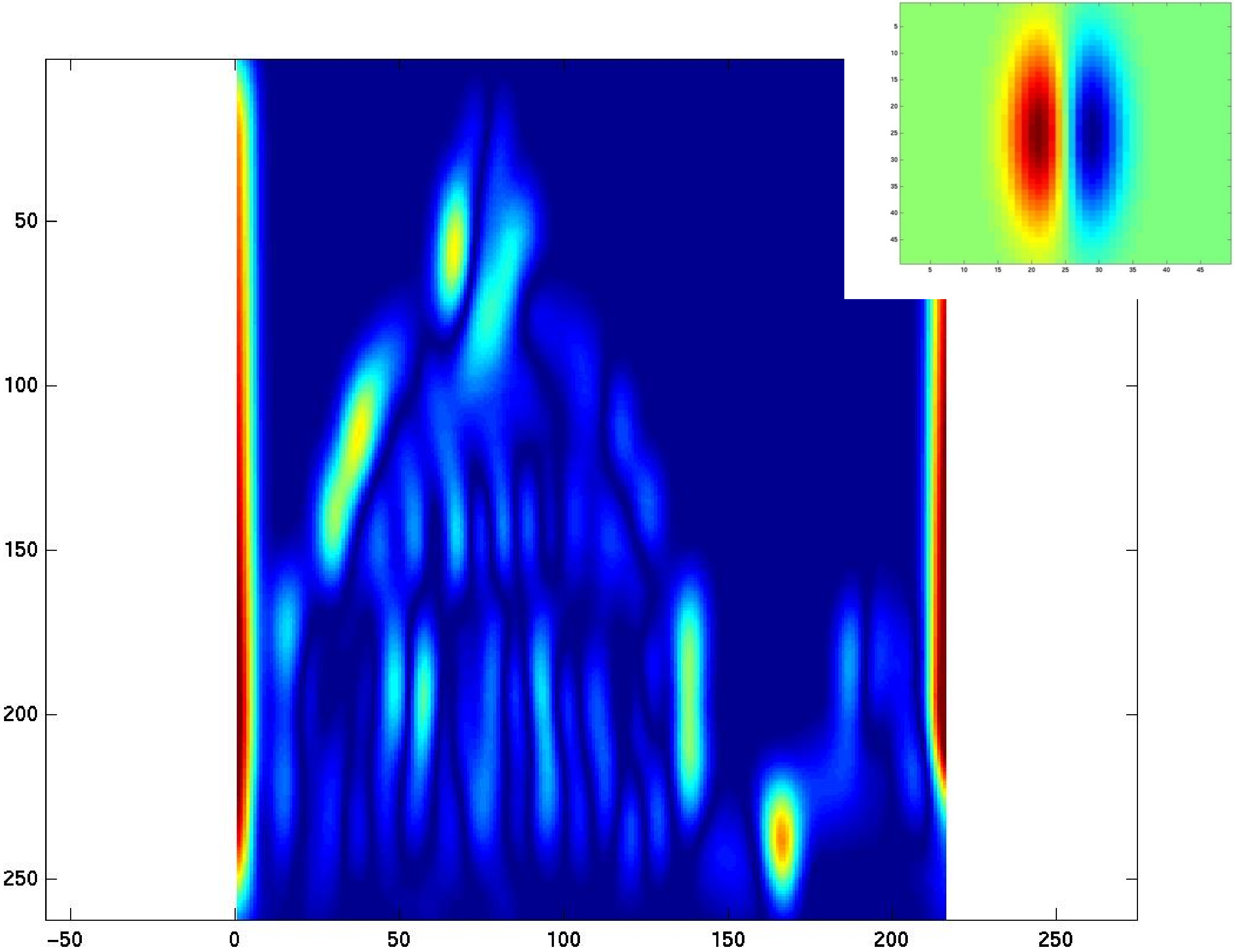

Kristen Grauman



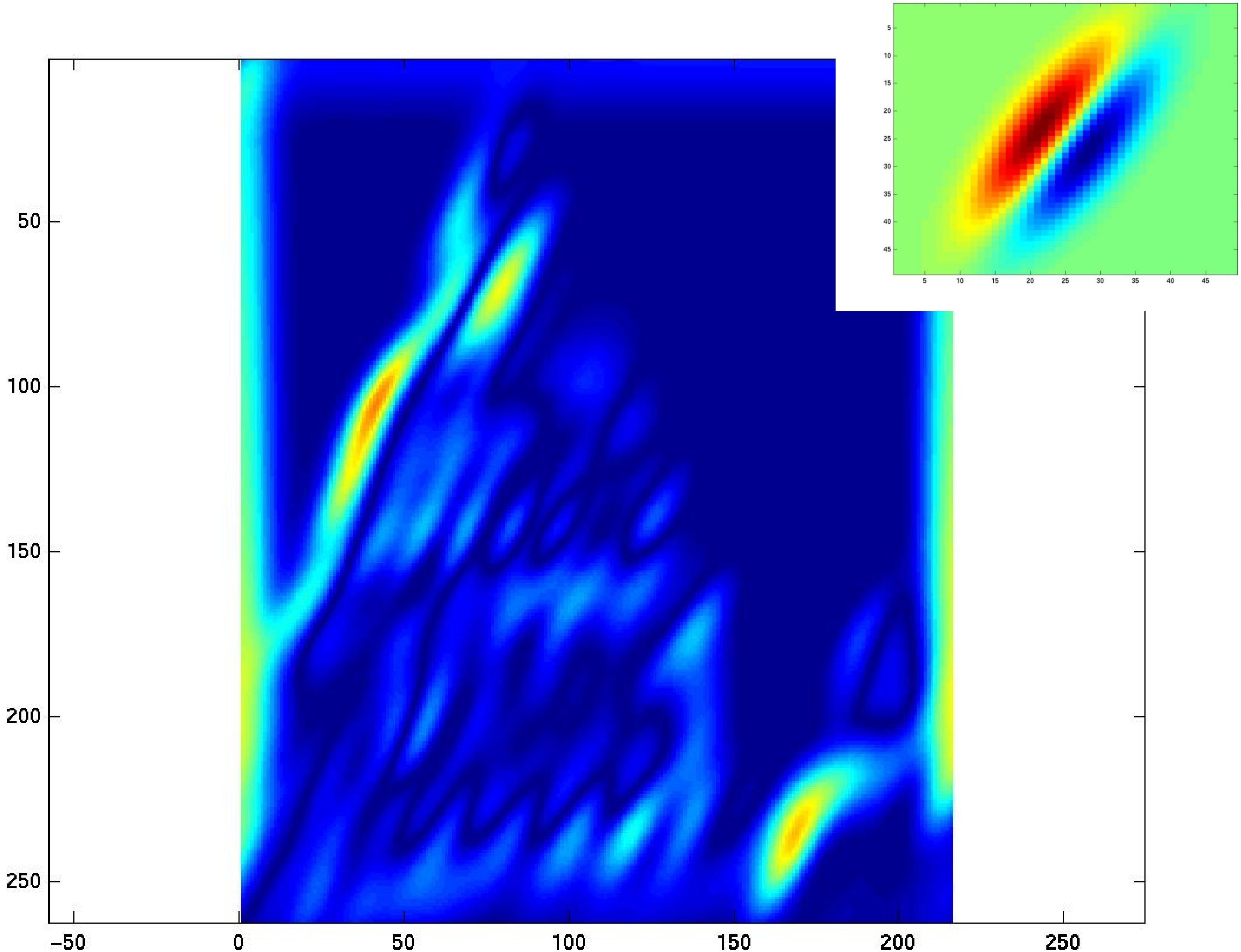

Kristen Grauman



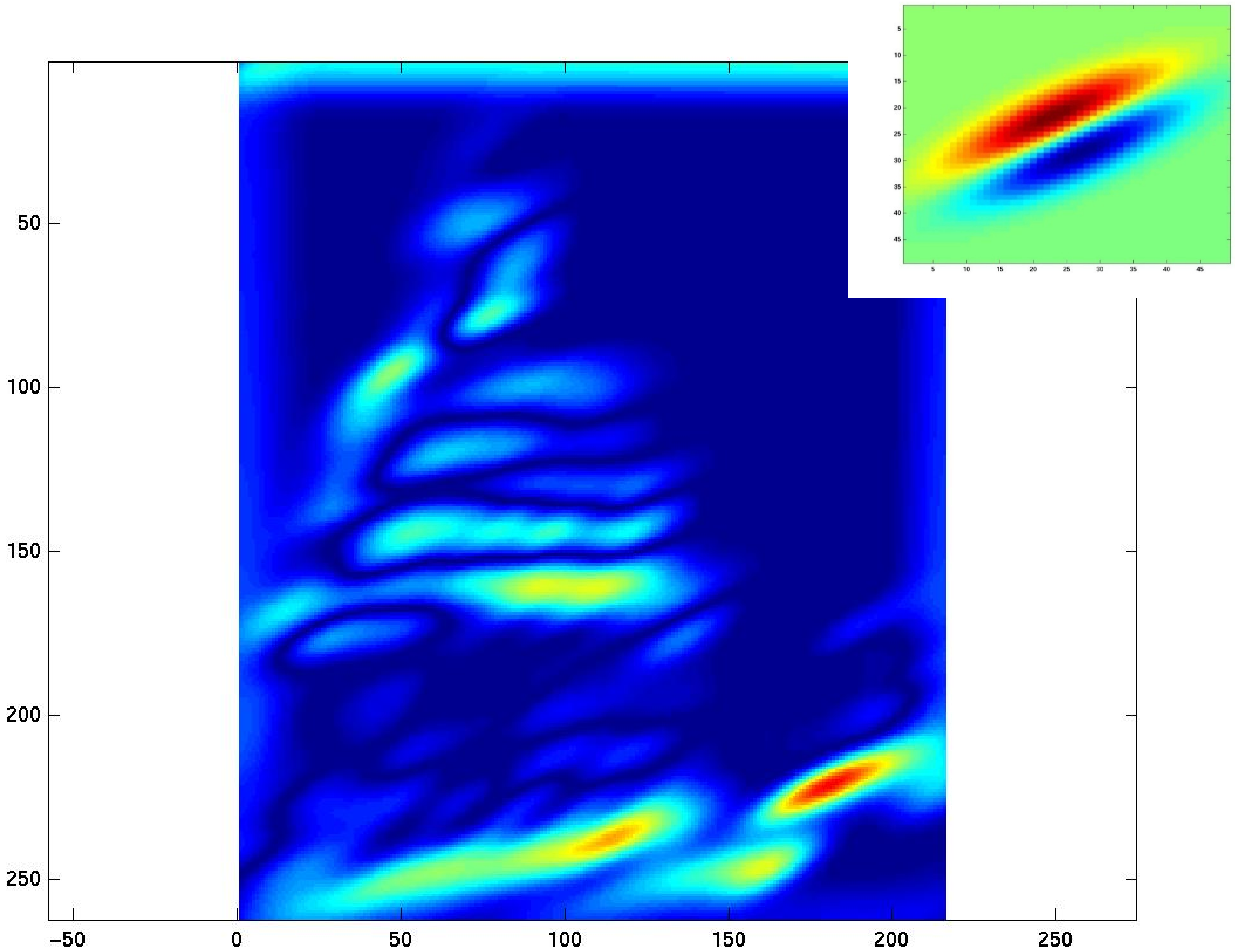

Kristen Grauman



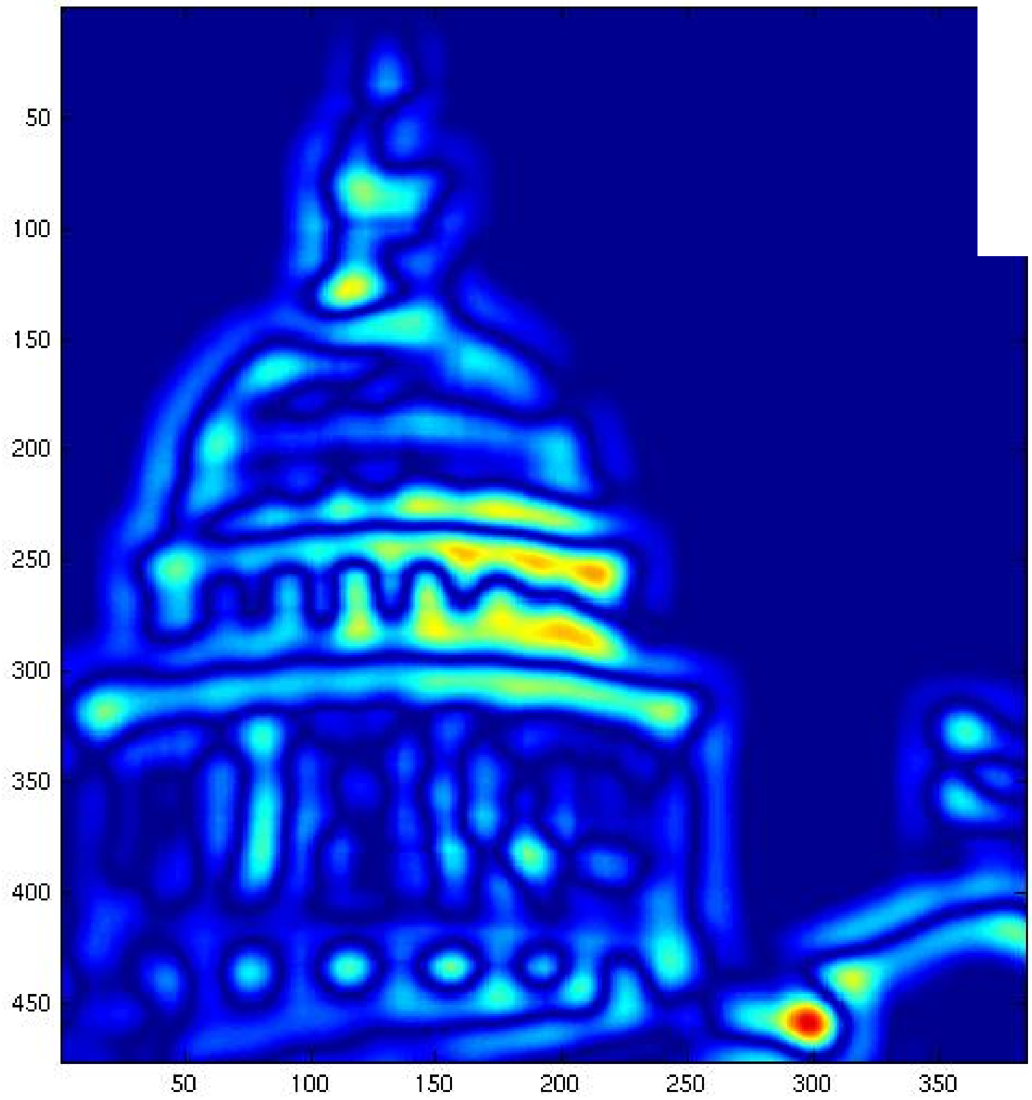

Kristen Grauman

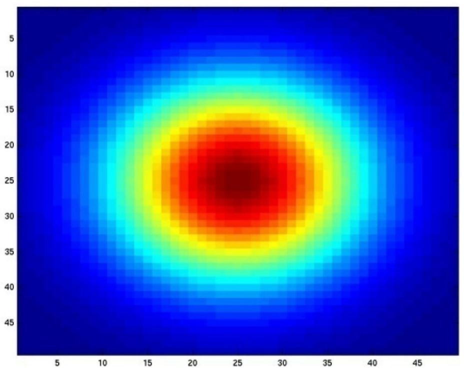



Kristen Grauman



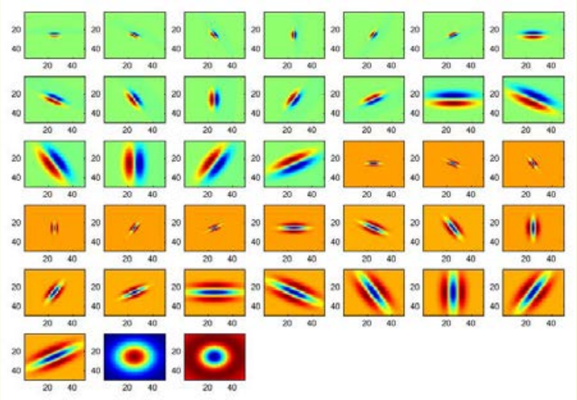

Kristen Grauman

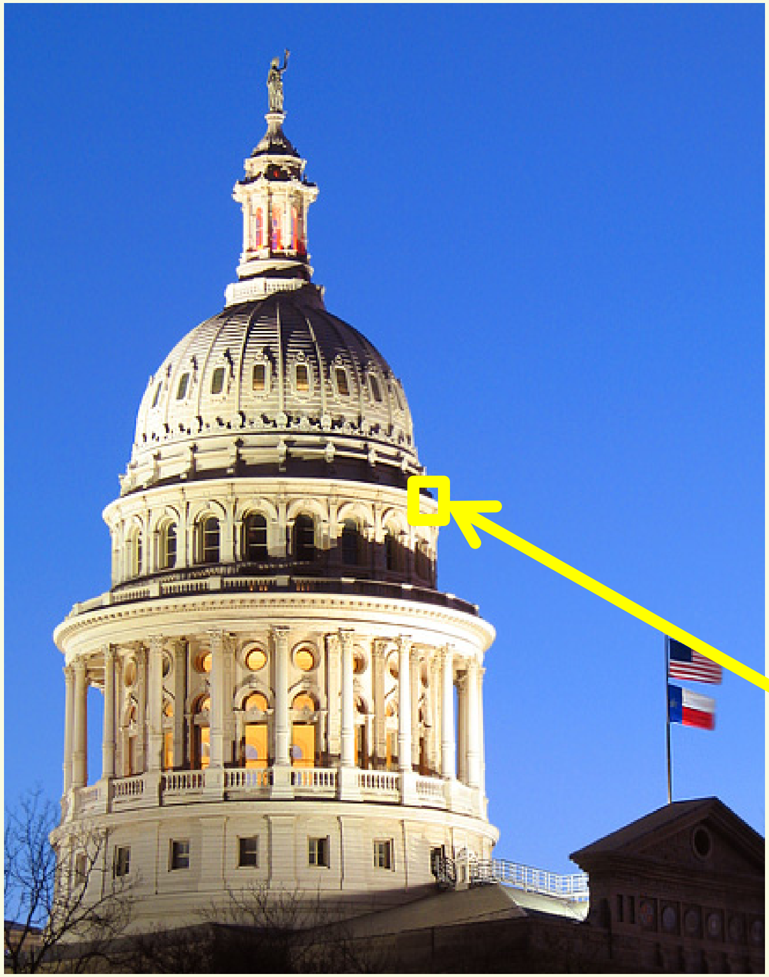



Kristen Grauman

# Extracting Texture

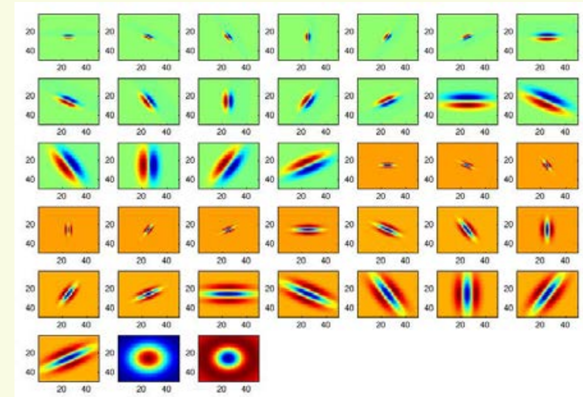


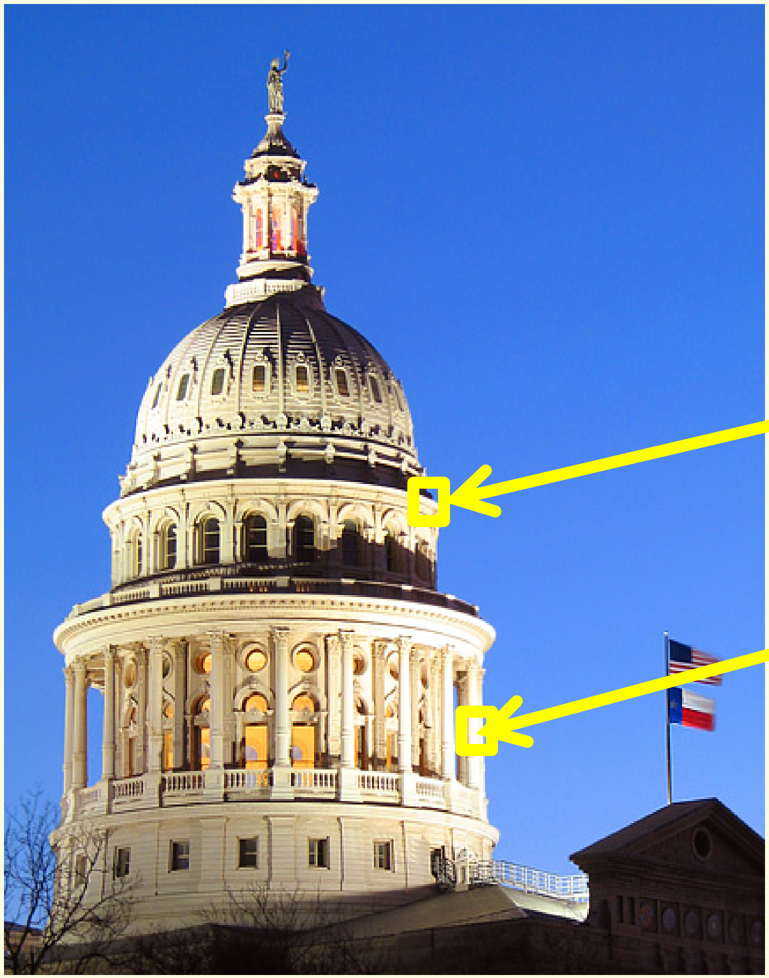


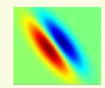
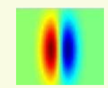
Form a feature vector from the list of responses at each pixel

$[r_1, r_2, \ldots, r_{38}]$

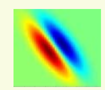
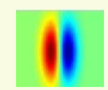
# Extracting Texture





[$r_1$,…, large, …., small, …, $r_{48}$]

[$r_1$,…, small, …., large, …, $r_{48}$]

# Right features depend on what you want to know

- Object: 2D shape
  - Local shape info, shading, shadows, texture
- Scene : overall layout
  - linear perspective, gradients
- Material properties: albedo, feel, hardness, …
  - Color, texture
- Motion
  - Optical flow, tracked points

Slide by Derek Hoiem

# Pixelwise Representation

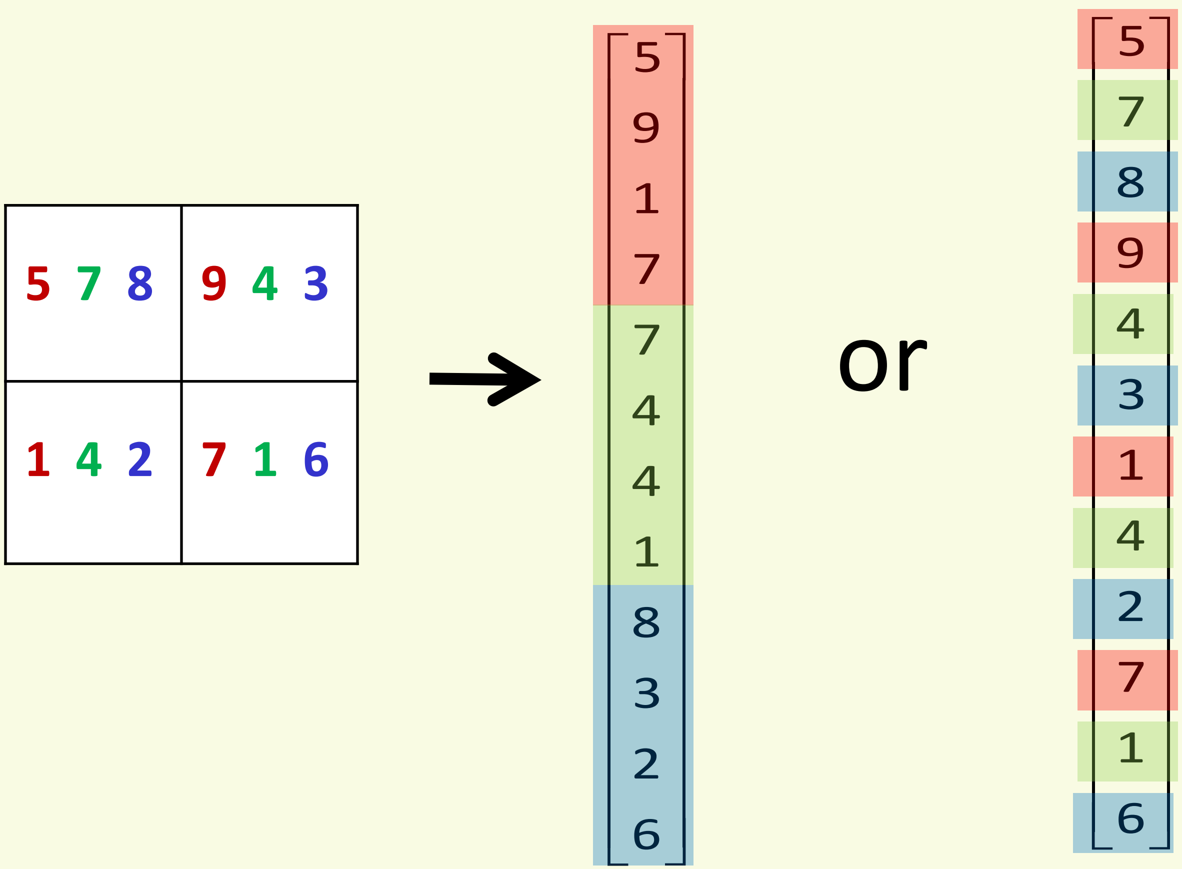
- Pile basic image feature values into one vector, say row order
- Example: intensity as a basic image feature
  - one value per pixel

| | | |
|---|---|---|
| 10 | 12 | 30 |
| 20 | 17 | 21 |
| 33 | 91 | 24 |

$$\rightarrow \begin{bmatrix} 10 \\ 12 \\ 30 \\ 20 \\ 17 \\ 21 \\ 33 \\ 91 \\ 24 \end{bmatrix}$$

# Pixelwise Representation

- Color as a basic image feature
  - three values per pixel
- Pile all color channel into one vector



# Pixelwise Representation

- Filter responses as a basic image feature
  - **n** values per pixel, **n** is the number of filters
- Pile each filter channel into one vector

| | |
|---|---|
| 5 7 8 3 | 9 4 3 2 |
| 1 4 2 1 | 7 1 6 0 |

$$\rightarrow \begin{bmatrix} 5 \\ 9 \\ 1 \\ 7 \\ 7 \\ 4 \\ 4 \\ 1 \\ 8 \\ 3 \\ 2 \\ 6 \\ 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

# Pixel Representations

- Small change in image appearance

Slide by Erik Learned-Miller

# Pixel Representations

- Leads to a large change in feature vector

difference image

| | | |
|---|---|---|
| 10 | 12 | 30 |
| 20 | 17 | 21 |
| 33 | 91 | 24 |

| | | |
|---|---|---|
| 9 | 10 | 12 |
| 19 | 20 | 17 |
| 32 | 33 | 91 |

[10 12 30 20 17 21 33 91 24]

[9 10 12 19 20 17 32 33 91]

Slide modified from Erik Learned-Miller

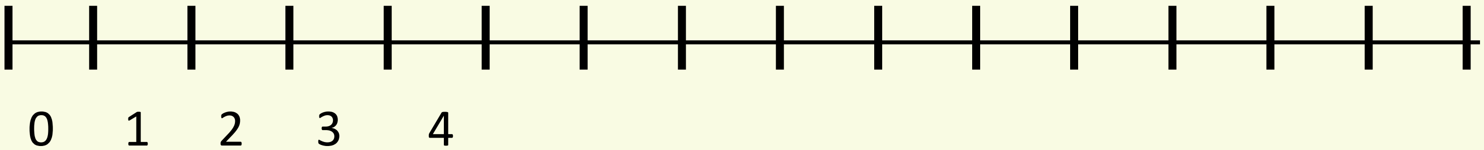
# Pixel Representations

- Pixelwise representations:
  *overly sensitive to position*
- Nevertheless it has been successfully used in applications
  - eigenfaces, first successful face detection system

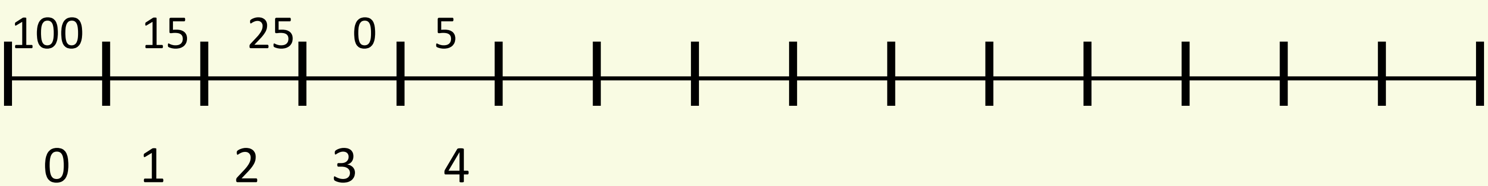
Slide by Erik Learned-Miller

# Global Intensity Histogram

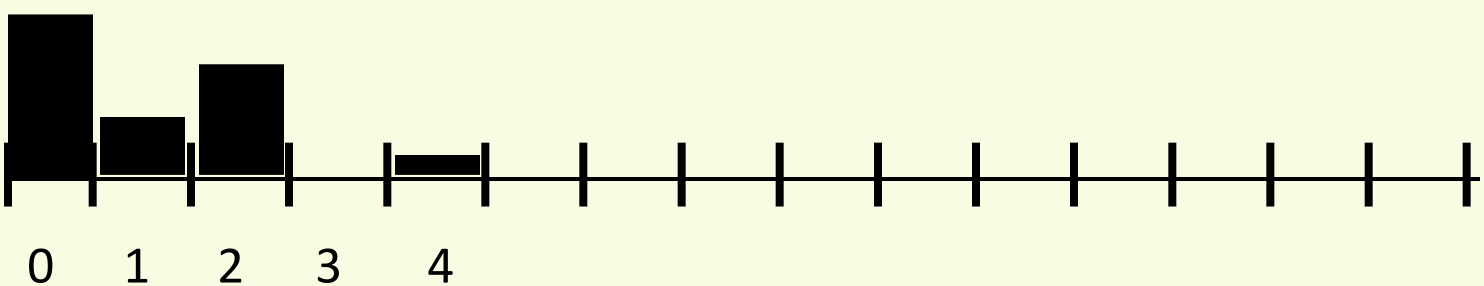
- Think of each intensity value as a "bin"



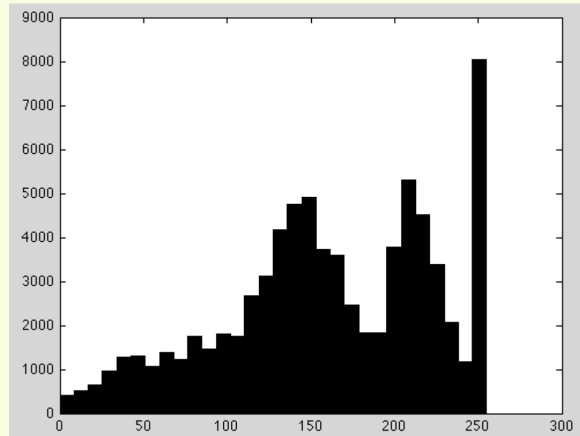

- Histogram counts the number of values that fall in each bin




- Visual plot:



# Global Intensity Histogram



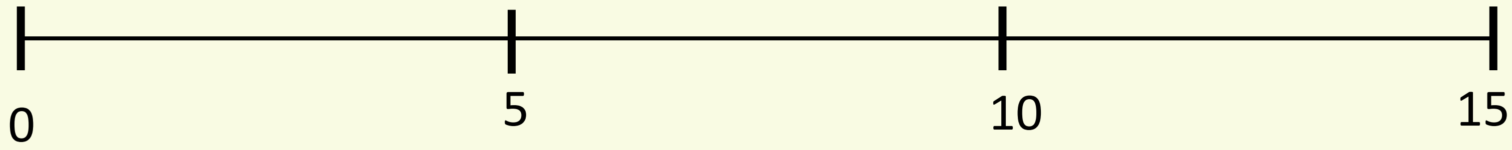

$$\begin{bmatrix} 5 \\ 10 \\ 4 \\ \dots \\ 100 \end{bmatrix}$$

- Insensitive to changes in pixel location
- Often use normalized histogram
  - sums up to 1

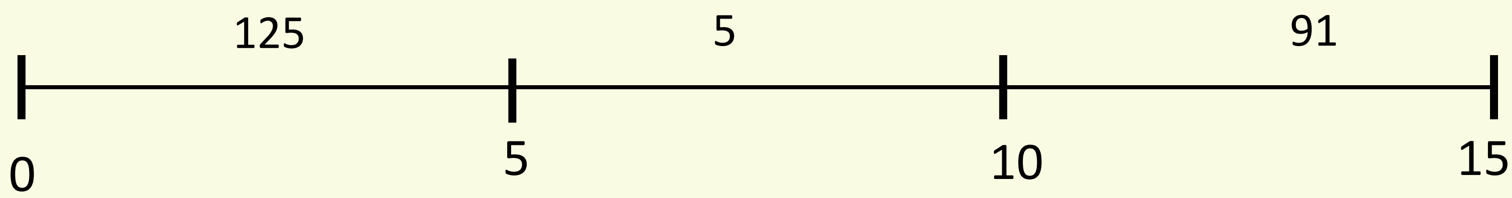
$$\begin{bmatrix} 0.04 \\ 0.08 \\ 0.03 \\ \dots \\ 0.8 \end{bmatrix}$$

# Global Intensity Histogram Quantization

- Can quantize intensities (larger bins)




- Histogram: count number of values that fall in each bin



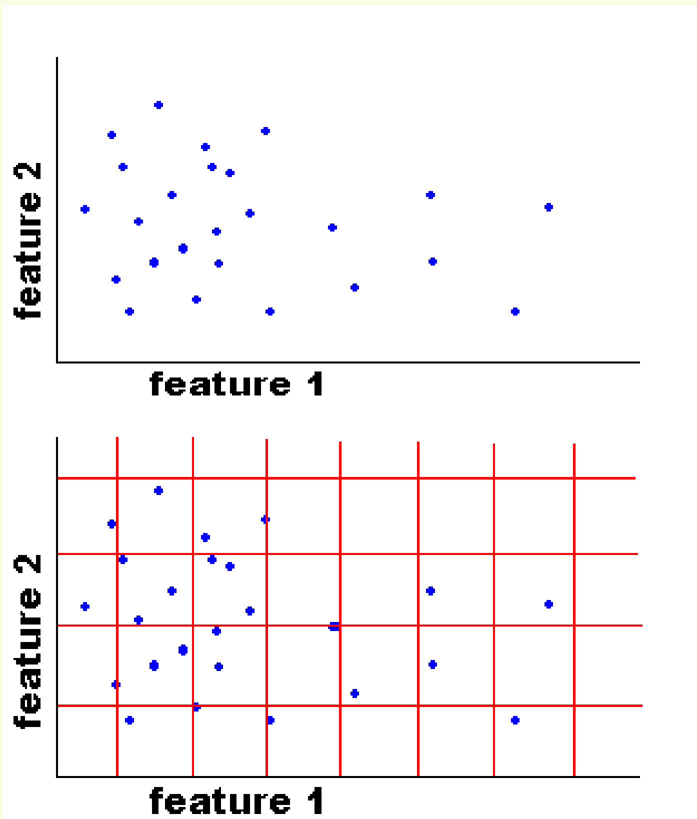

- Quantization
  - helps to improve efficiency
  - groups similar values together (i.e. removes fine distinction)
    - may help for recognition

# Multi-Dimensional Histograms

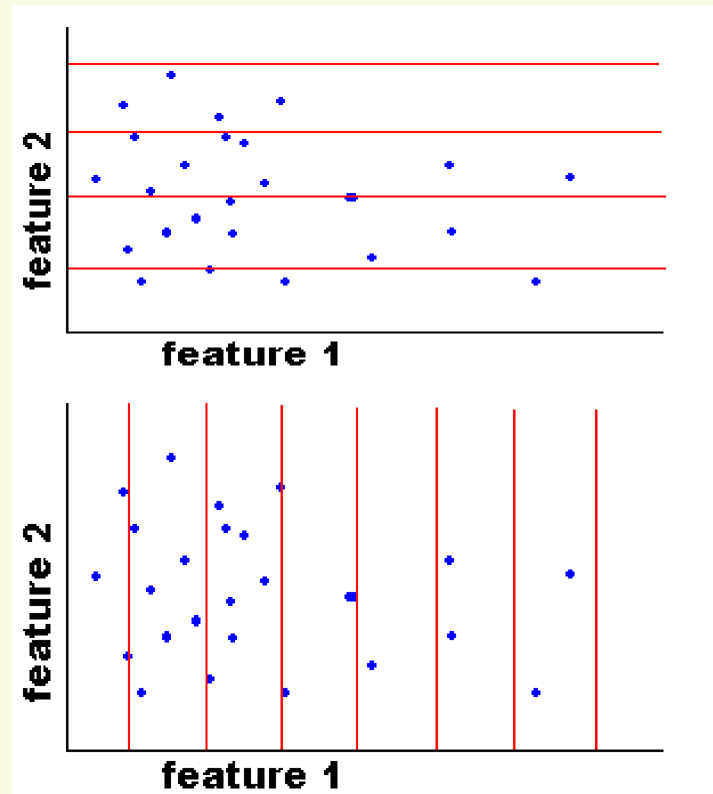
- Basic image features most often multi-dimensional
  - color, texture, optical flow,  etc.
- How to build histogram?
- Have to quantize, too sparse without quantization

# How to Quantize Multi-Dimensional Data?

1. Joint histogram
   - Need lots of data to avoid empty bins
   - Make bins coarse to simulate lots of data $\rightarrow$ loose resolution



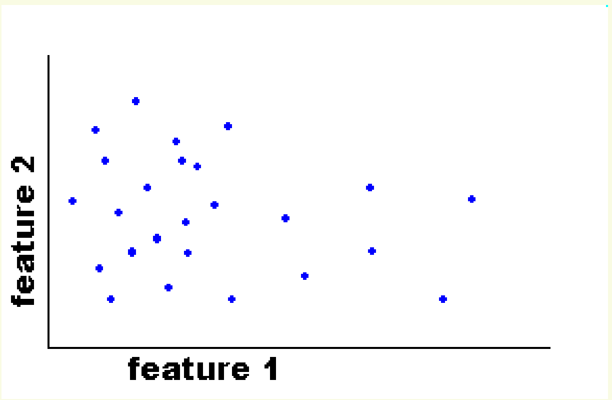

2. Marginal histogram
   - more data per bin than joint histogram
   - works best for independent features
     - loose correlation information

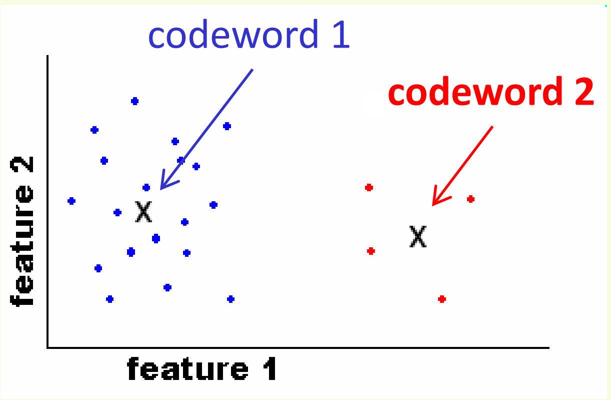


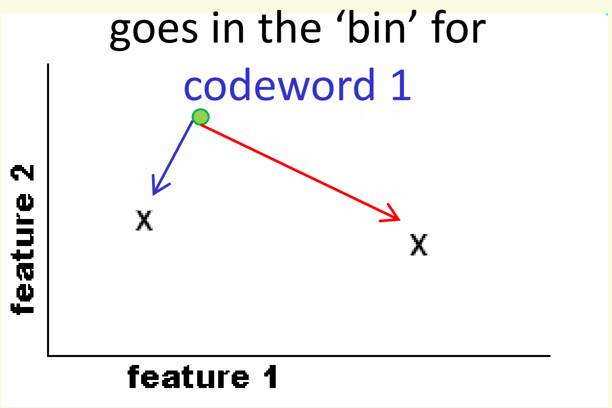

slide credit: Dave Kauchak

# Histograms based on Irregular Partitioning

- Irregular quantization (clustering) gives meaningful bins that adapt to data
    - k-means clustering, etc.



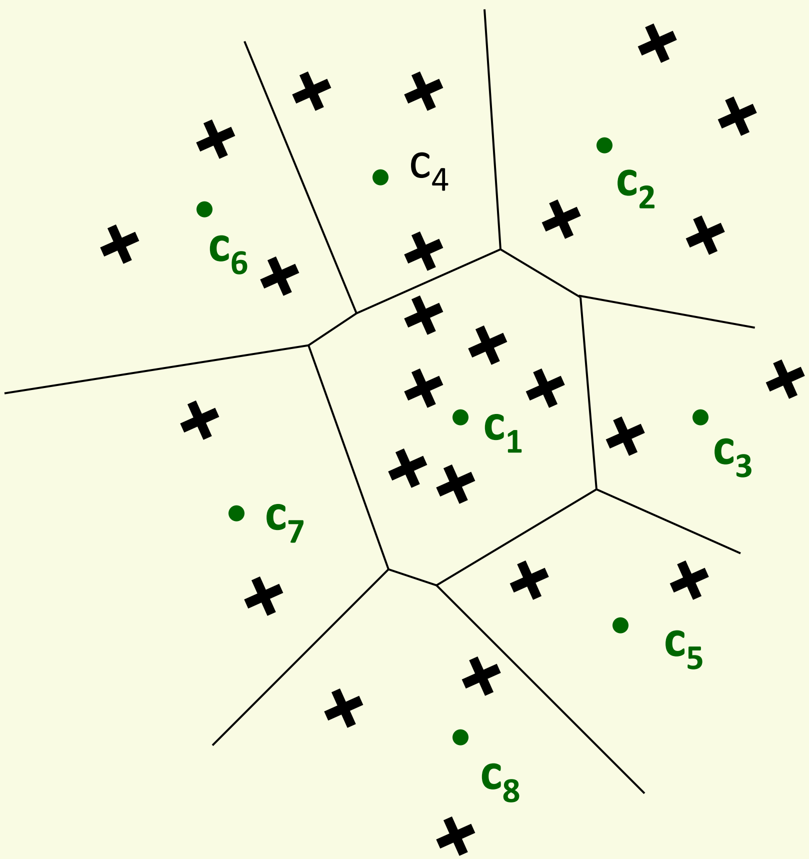



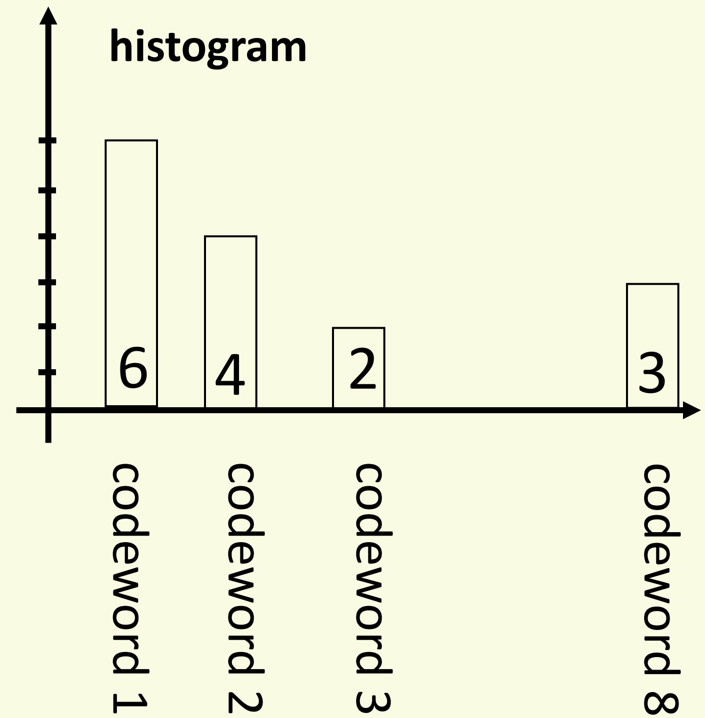

- Cluster centers are called **codewords**
- A sample is identified (assigned to) with the closest codeword
- Build histogram over the codeword
    - count how many samples are closest to codeword 1, codeword 2, etc.
- Need to store only the codewords

Slide Credit: Dave Kauchak

# Encoding Image *I* as Feature Vector

- Pre-computed code-words in green
- Extract 2D features from image *I*



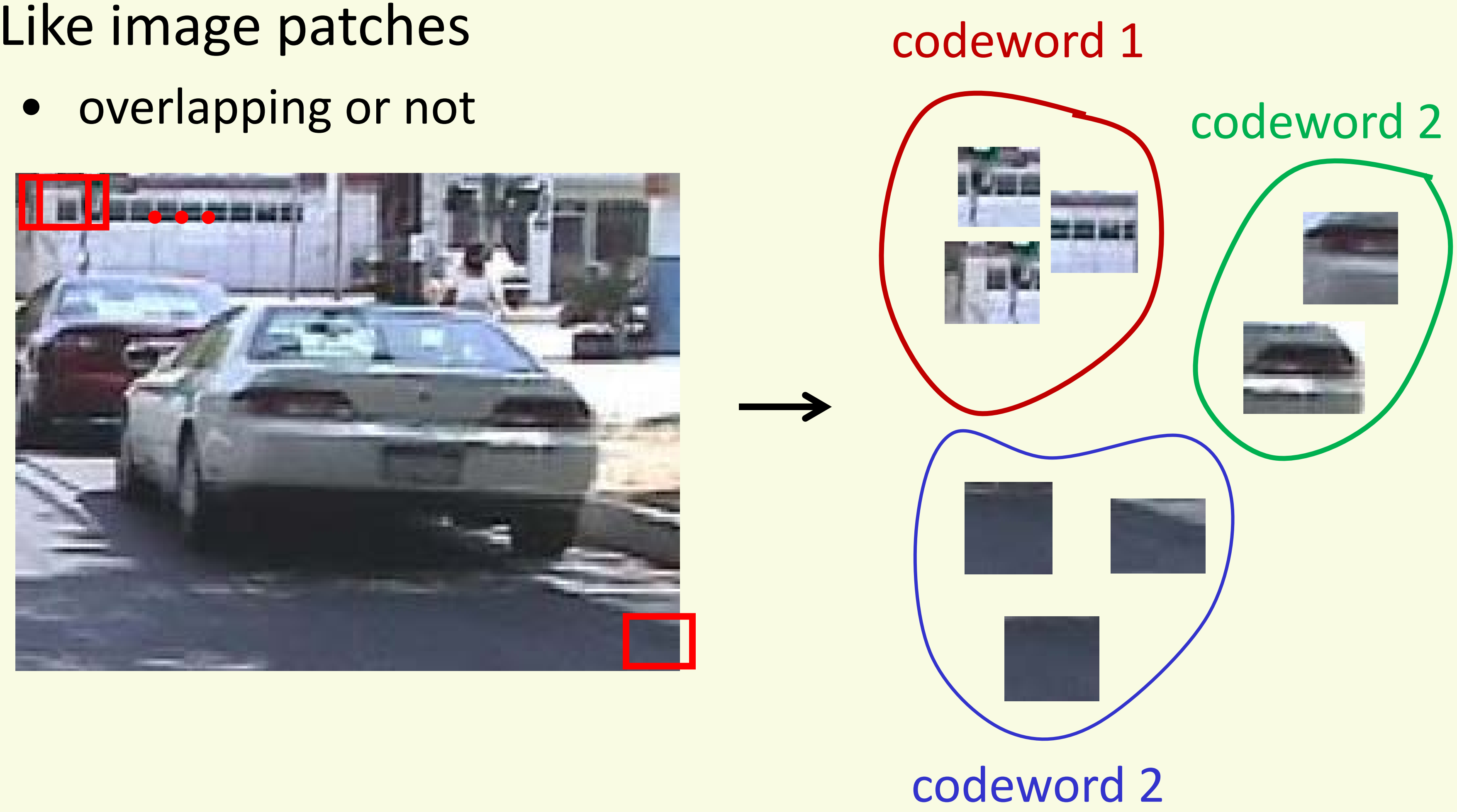




- Feature vector that represents image *I*
  - can also normalize it

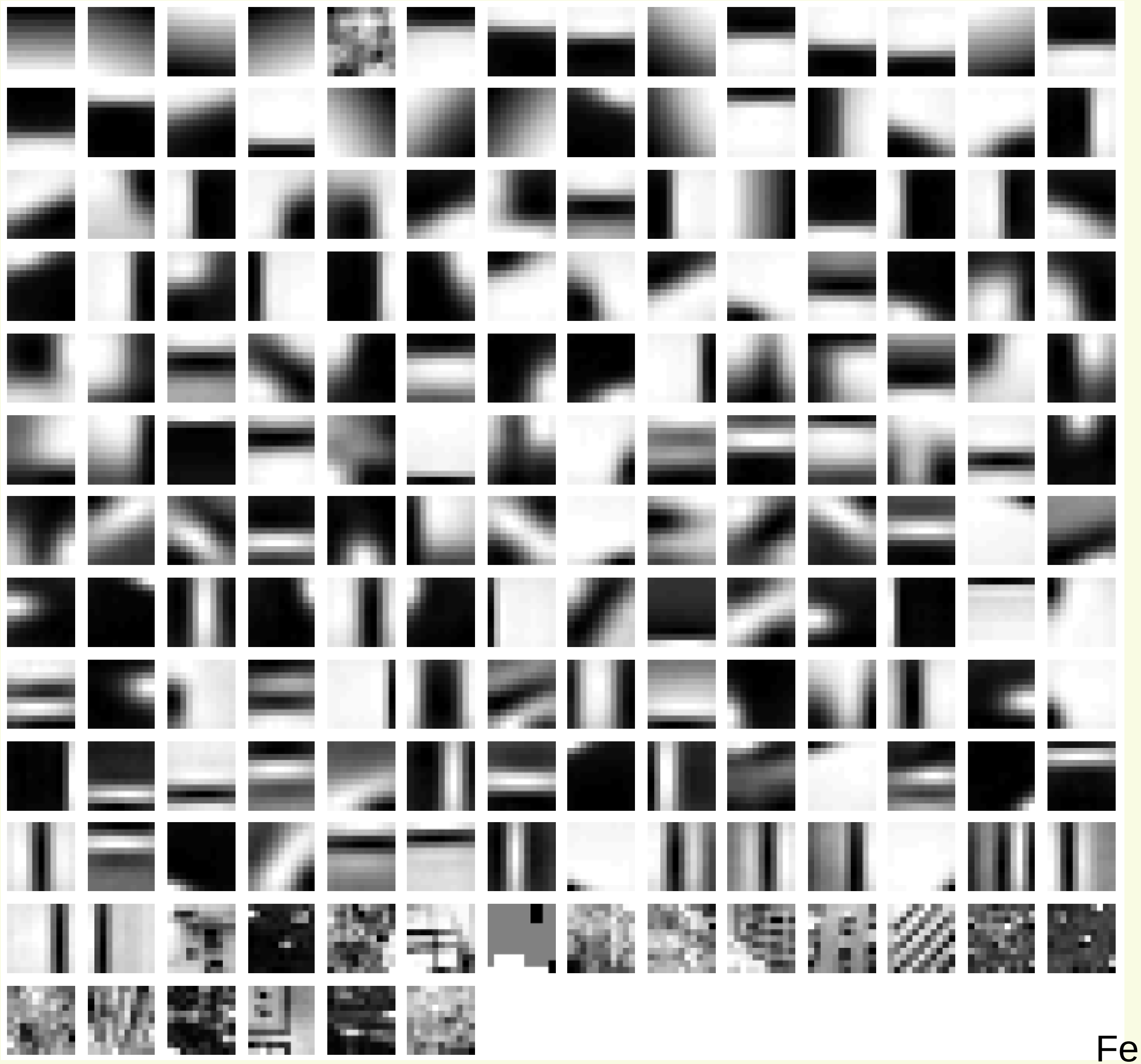
$$\begin{bmatrix} 6 \\ 4 \\ 2 \\ \dots \\ 3 \end{bmatrix}$$

# Clustered Patches

- So far clustered feature responses at each pixel
- Can cluster other things
- Like image patches
    - overlapping or not

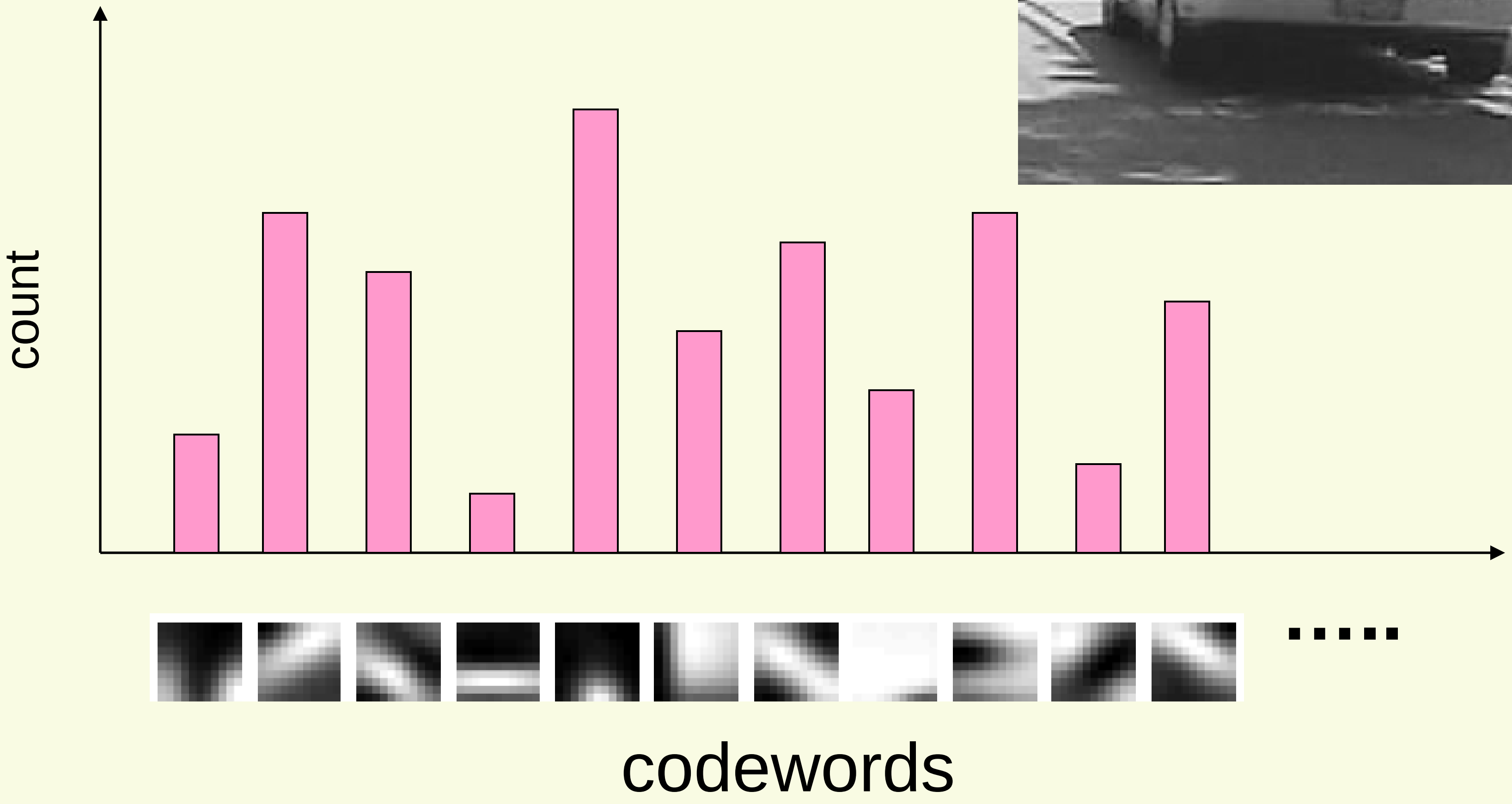


# Clustered Image Patches



Fei-Fei et al. 2005

# Feature Vector for image *I*




A. Efros

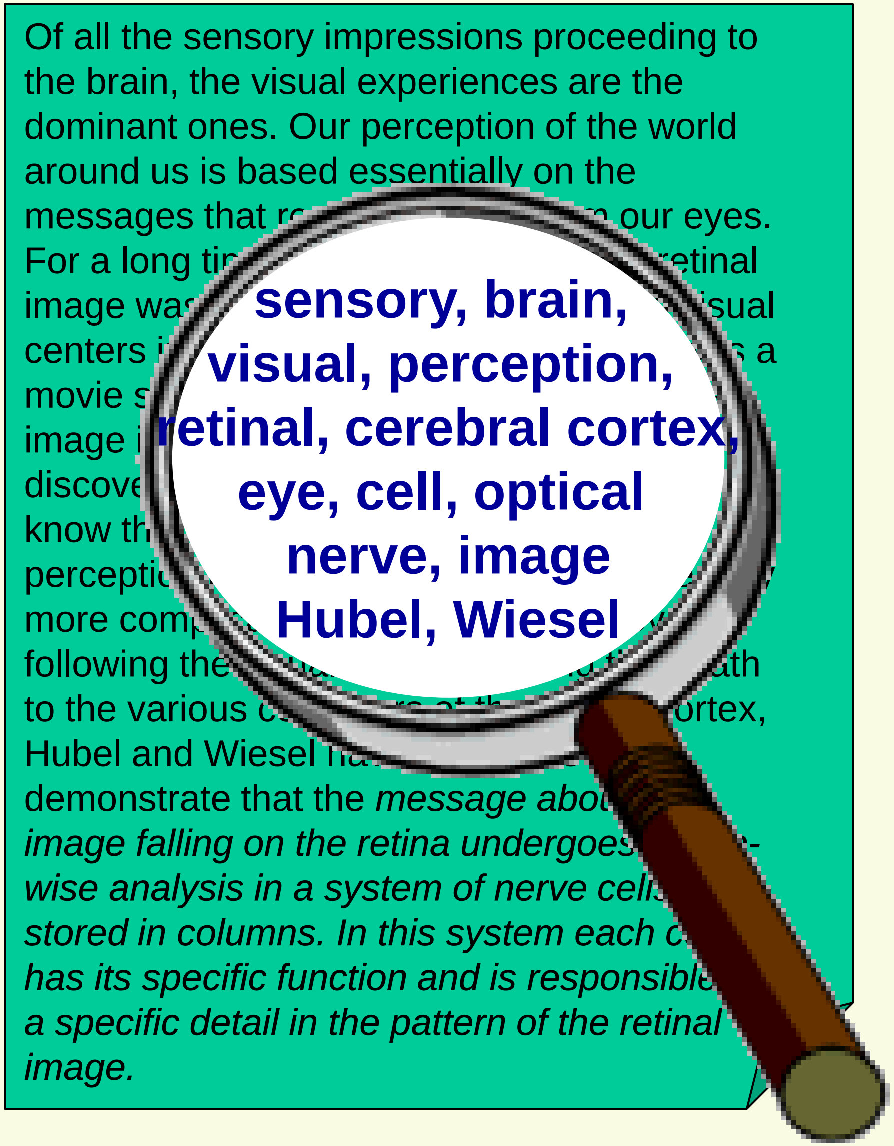
# Codewords

- Find codewords on training data, not just one image
- Usually use only a subset of training data for speed

- But not on test data

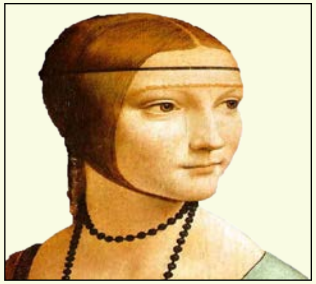
# Analogy to documents: Bag of Words

- Inspiration comes from text classification

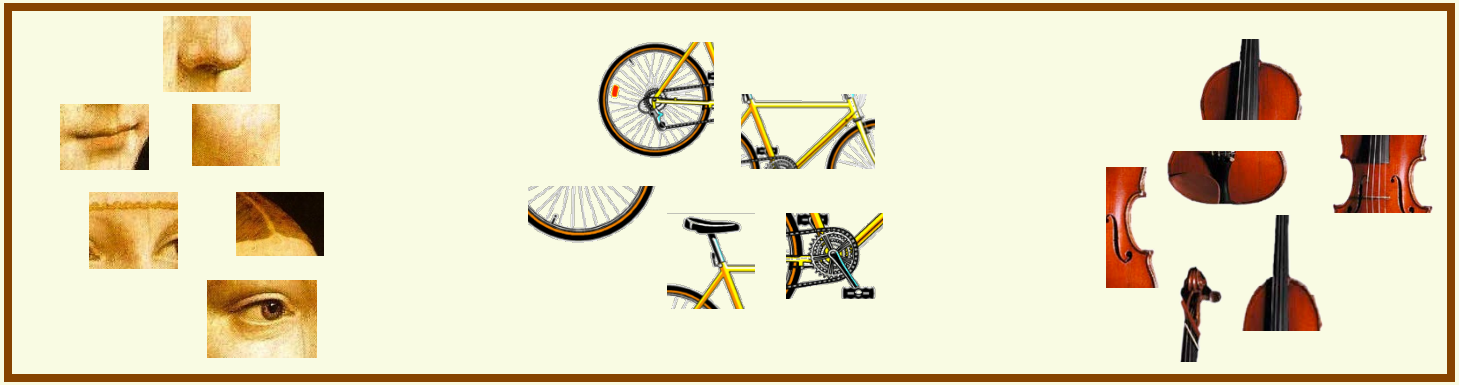



# Bag of visual words

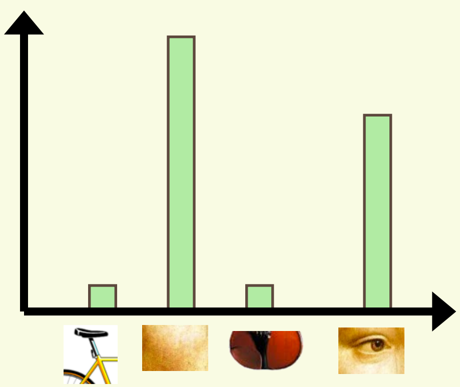
- Training images

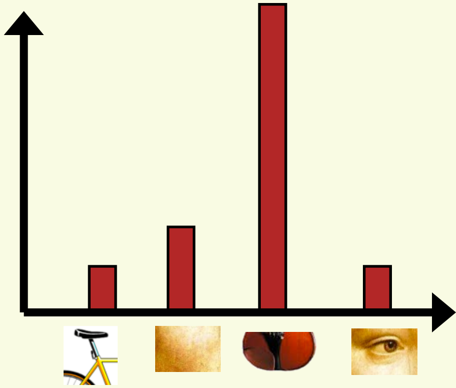


- **codewords** or **visual words**



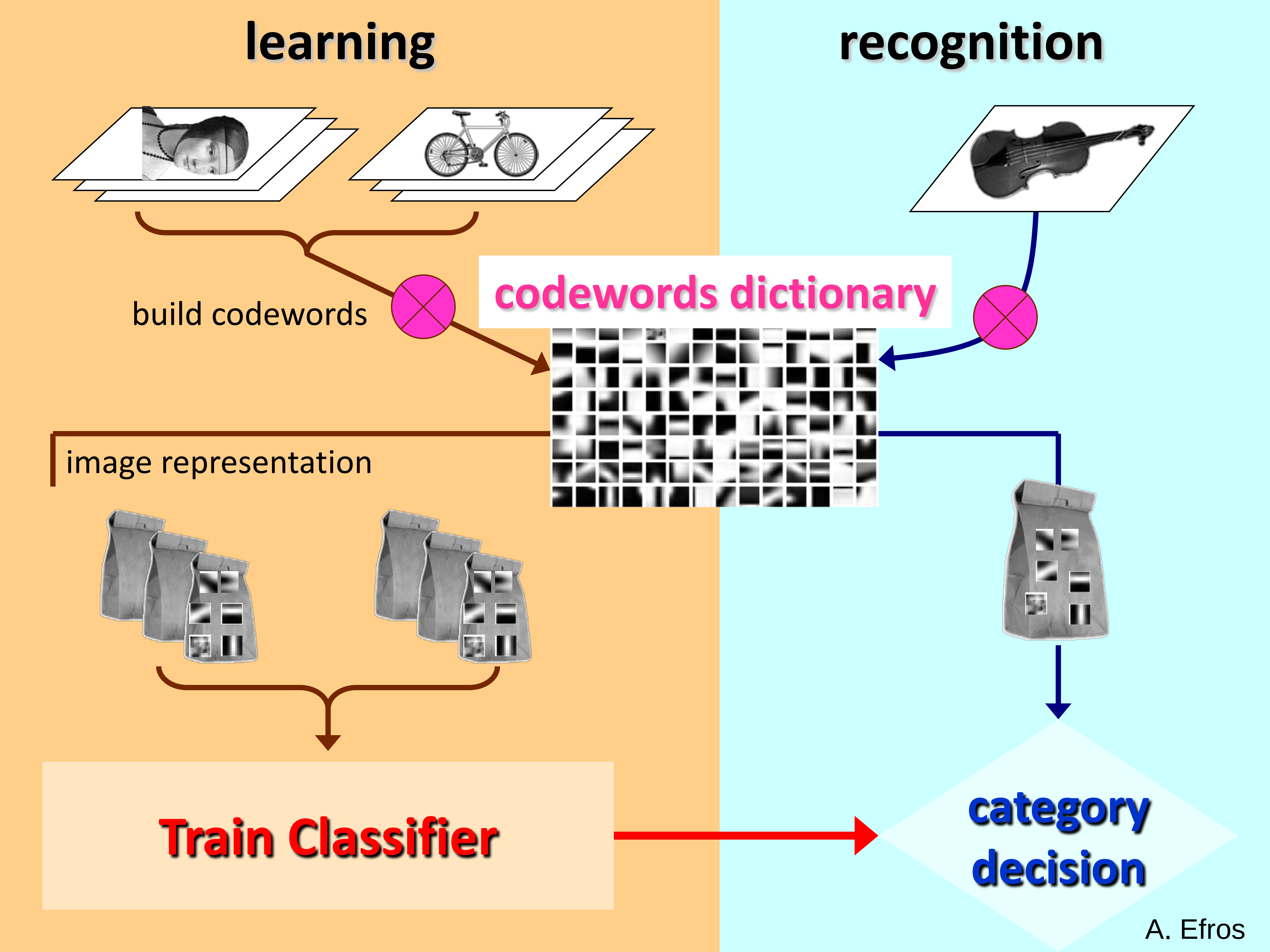
- Bow histogram





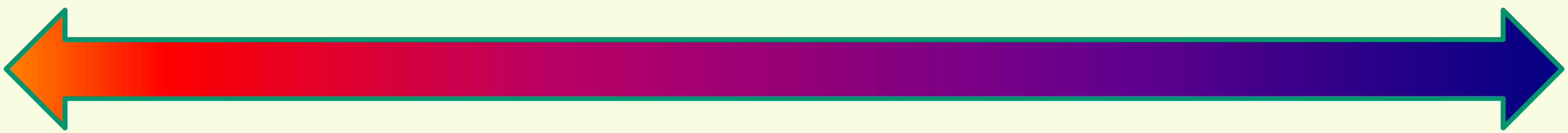
**codewords**

Slide by Derek Hoiem


learning
recognition
build codewords
codewords dictionary
image representation
Train Classifier
category decision
A. Efros

# Histograms: Implementation issues

- Quantization
  - Grids: fast but applicable only with few dimensions
  - Clustering: slower but can quantize data in higher dimension
- How many bins (clusters)?

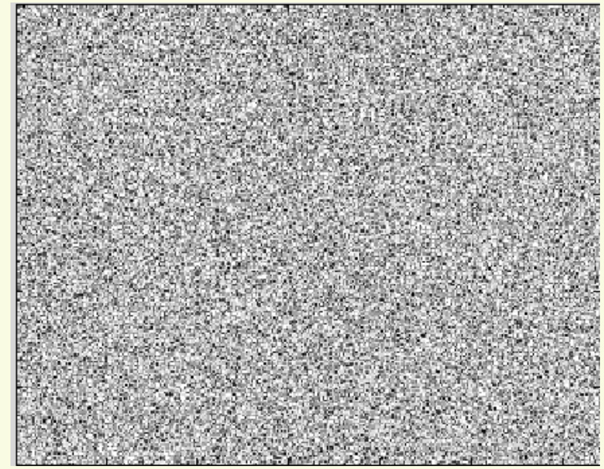


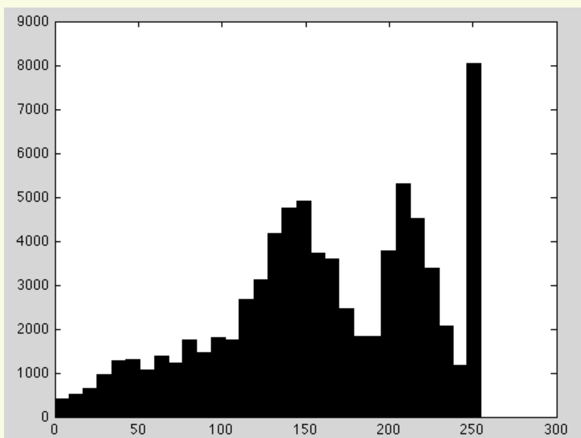
Few Bins
Need less data
Coarser representation
If too coarse, distinction is lost

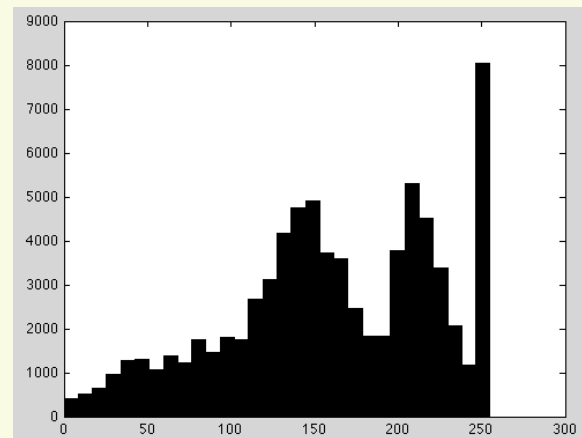
Many Bins
Need more data
Finer representation
If too fine, more distinction than necessary

Slide Credit: Derek Hoiem

# Problem with Global Histogram



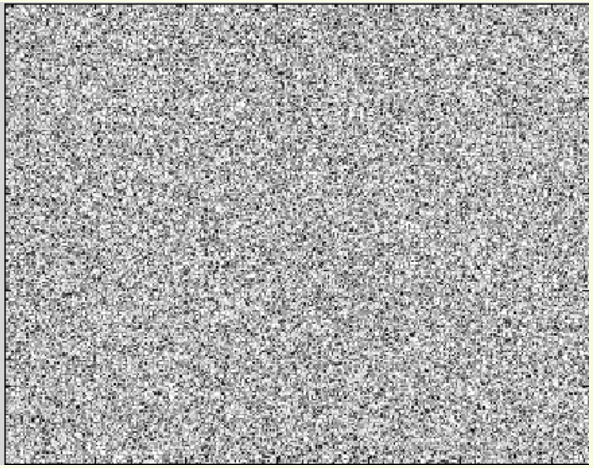



- Identical feature vectors!

Slide by Erik Learned-Miller

# Problem with Global Histogram



Have equal histograms!

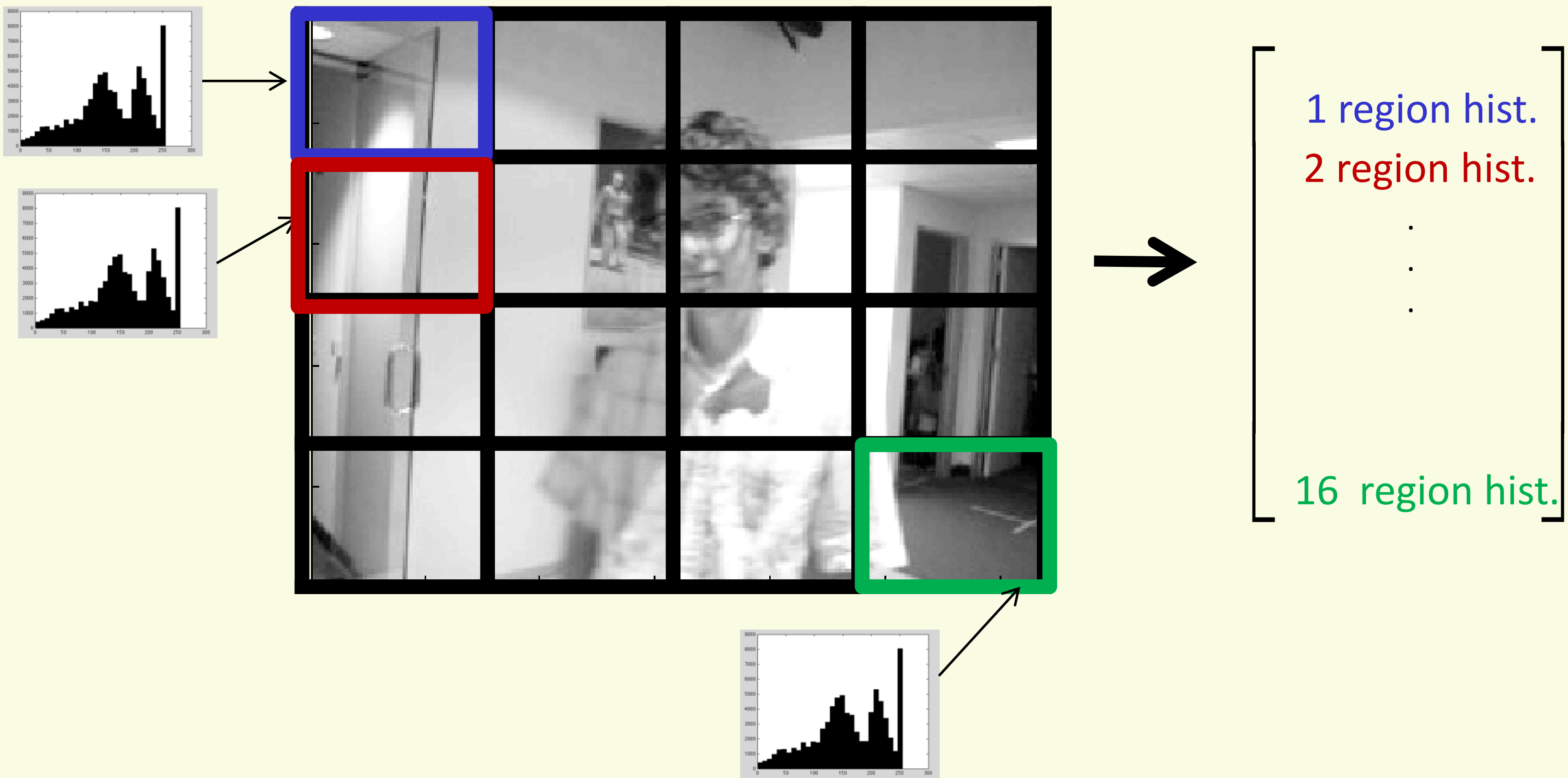
# Conclusions

1. Pixel representations:
   *overly sensitive to position*
2. Global histogram representations:
   *under-sensitive to position*

Slide by Erik Learned-Miller

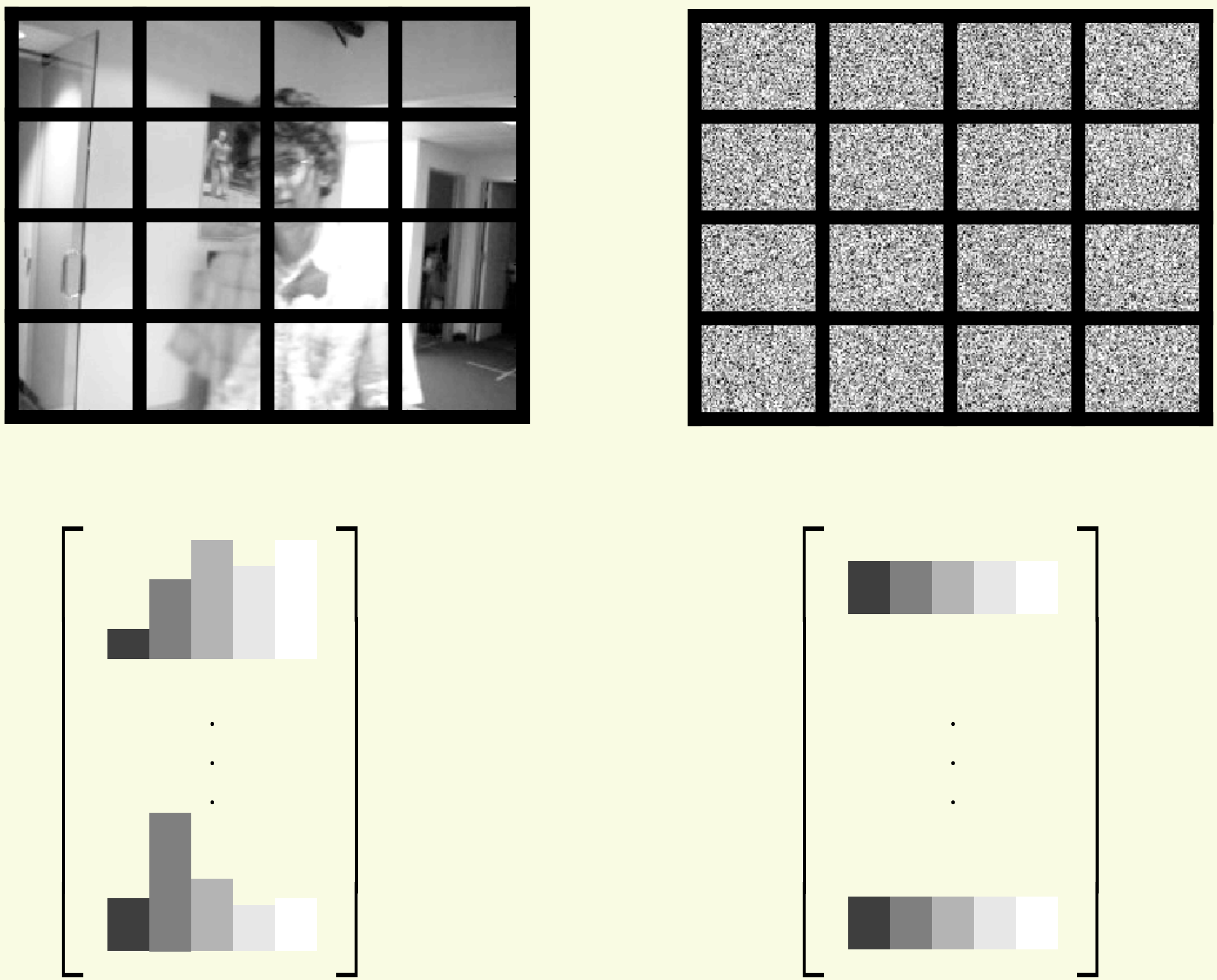
# A Compromise: A local histogram

A separate (normalized) histogram for each region

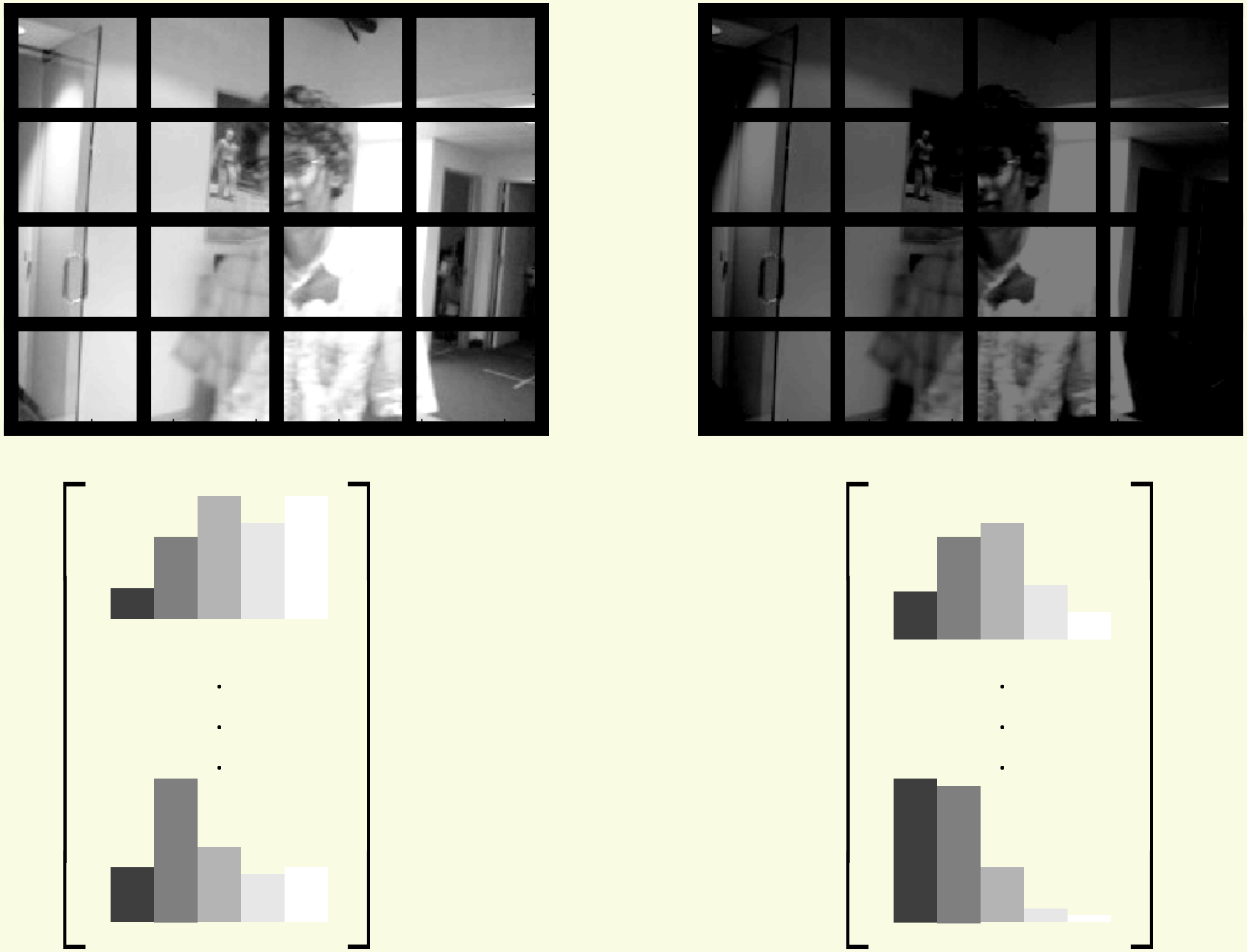



Slide by Erik Learned-Miller

# Local Intensity Histogram

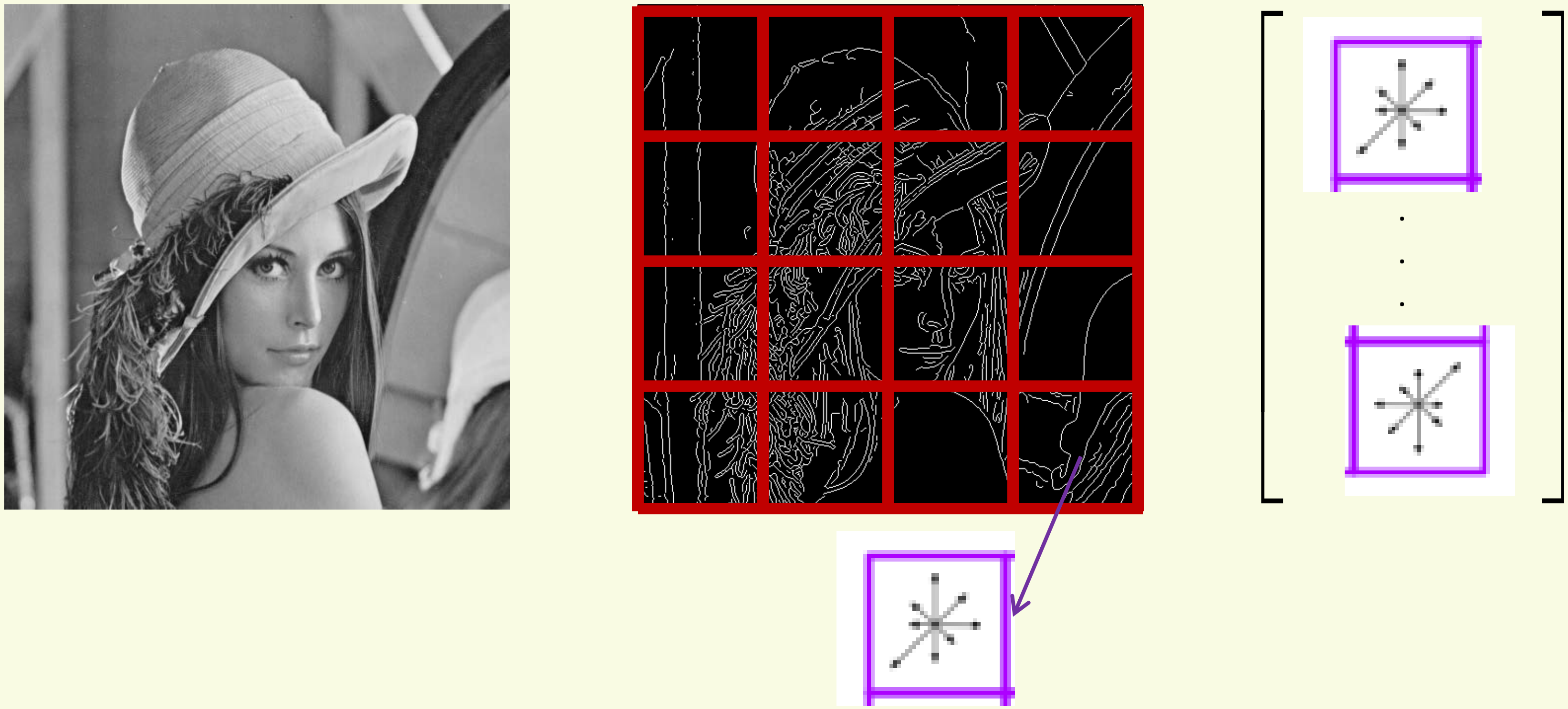


# Local Intensity Histogram



- Intensity histogram is sensitive to lighting changes

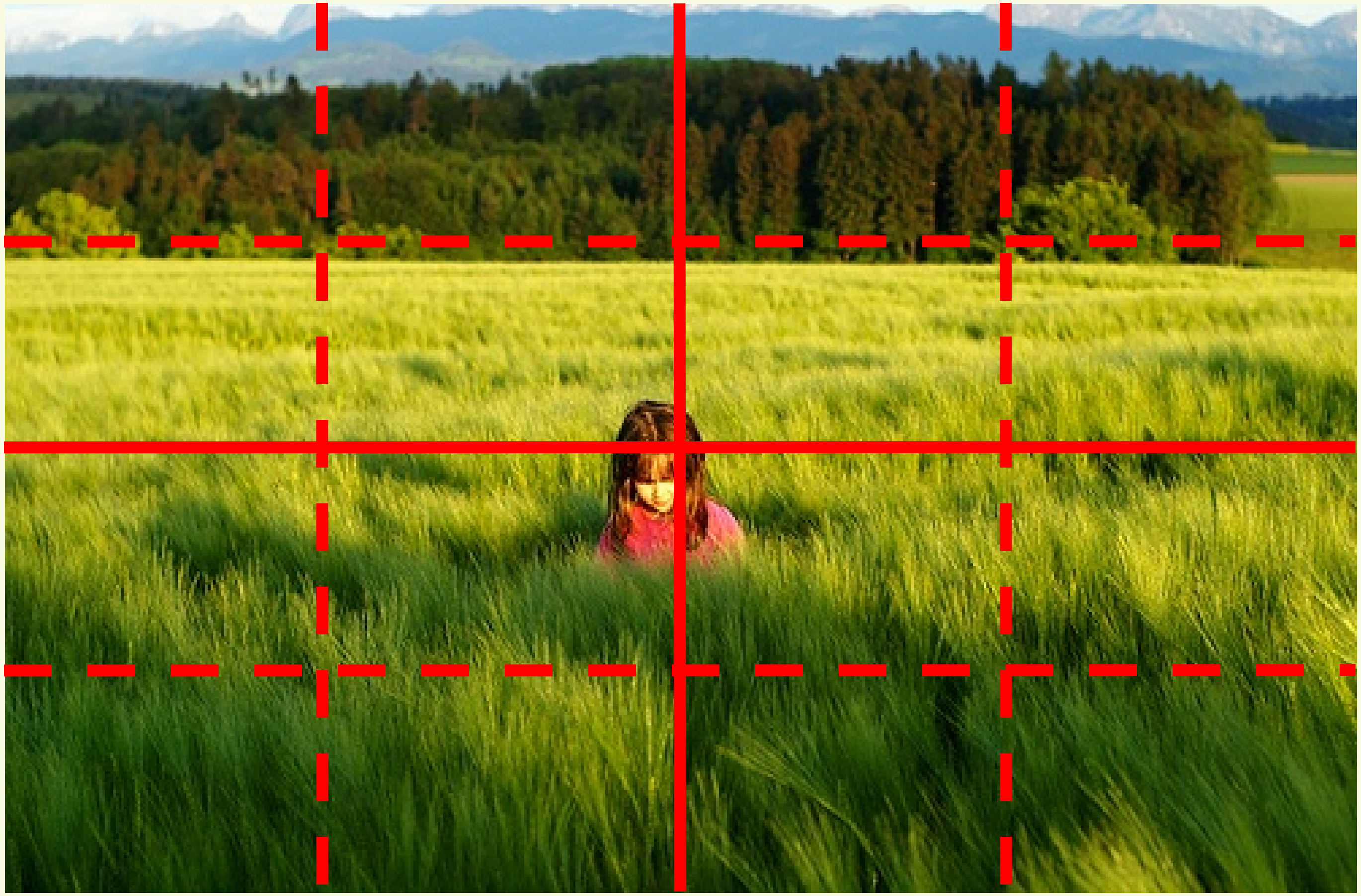
# Local Edge Orientation Histogram



- Edges are not as sensitive to lighting changes
- Compute histogram of edges
  - typically consider only edge orientation
- How do we choose the right box size?

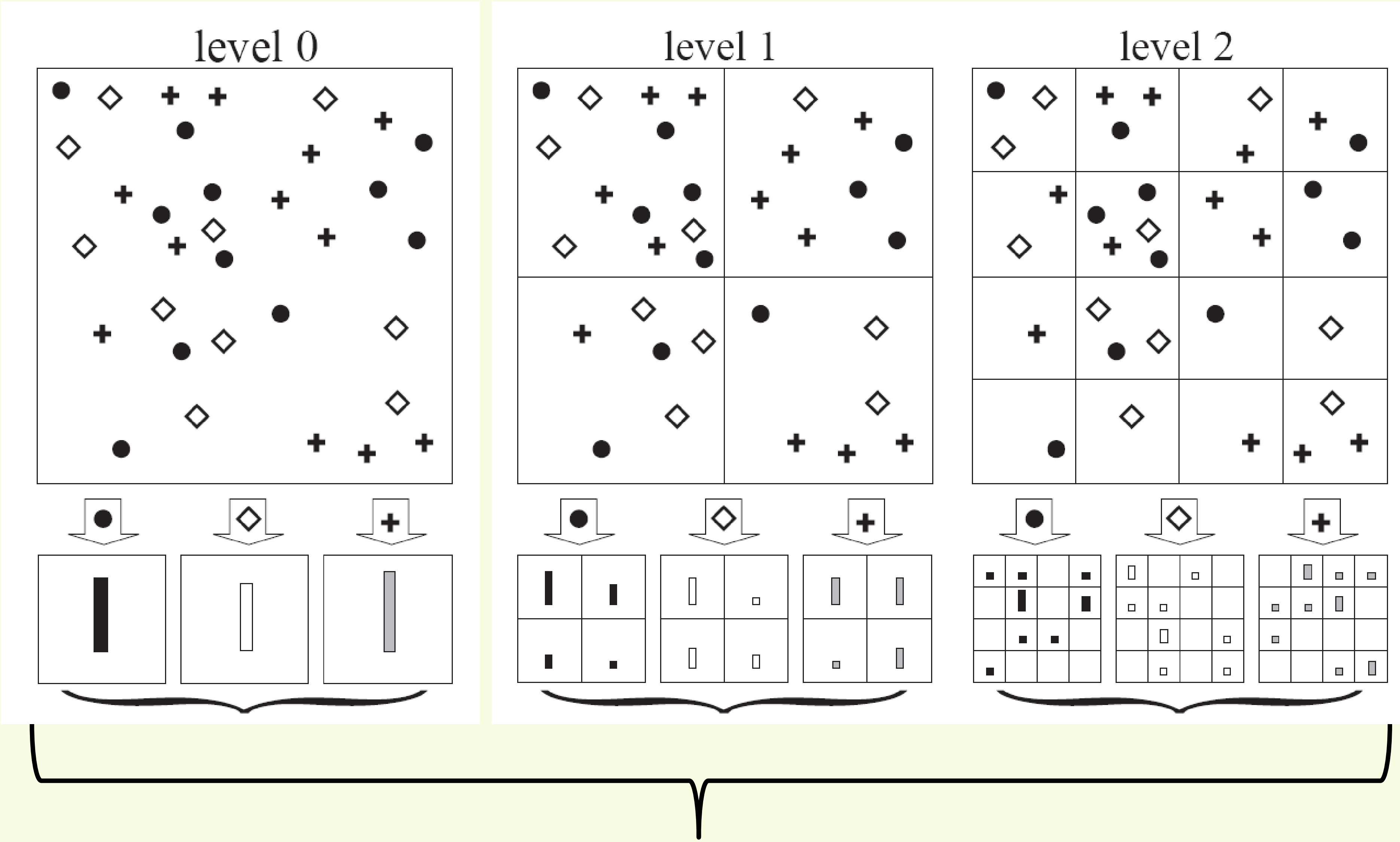
# Spatial pyramid

- Use boxes of different sizes!



Slide Credit: Derek Hoiem

# Spatial Pyramid




These get piled up into one feature vector

Slide Credit: Derek Hoiem

# Other Representations

- Many image representation schemes are based on histogram of
  - texture
  - corner features
  - SIFT features
  - etc.
- There are other ways to represent an image as a feature vector