Visual Textures

CS 4243 Computer Vision & Pattern Recognition

Angela Yao

Recap & Outline

Last Week

- Goal of segmentation is to separate image into coherent regions
- Treat segmentation as a clustering problem
- K-means clustering for segmentation
- Mean-shift clustering for segmentation

Today's Lecture

- Visual textures: property indicative of material and appearance
- Filter banks as feature representations
- Representing texture with textons and histograms of textons
- Role of texture and texture boundaries in perceived contours

The Big Picture

We have looked at so far

- how digital images are represented
- point-wise and filtering operations to adjust the appearance and extract low-level structures

1- or 3-channel matrices of intensity values

brightening, contrast, equalization, etc.

regions & boundaries

- boundaries: image gradients, edges (via Canny), lines (via Hough transform)
- regions: based on intensity or colour (via k-means clustering)

textures

Can we do better?

mean-shift clustering

What kind of response will we get with an edge detector for these images?

Image Edges

≠

Perceived Boundaries

relies on texture

Dictionary

Search for a word

texture

/ˈtɛkstʃə/

noun

the feel, appearance, or consistency of a surface or a substance.
"skin texture and tone"
synonyms: feel, touch; More

verb

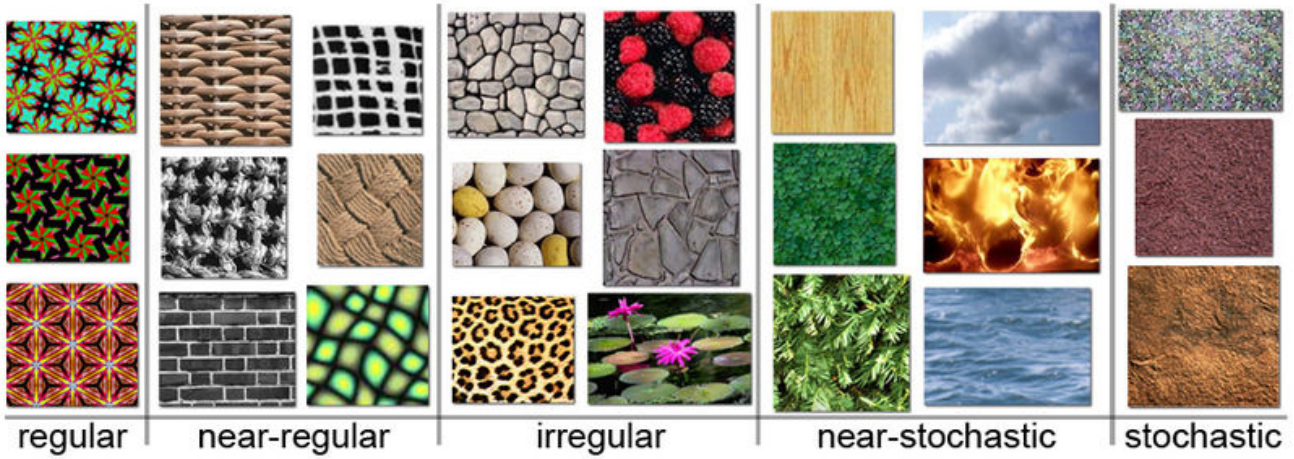
give (a surface) a rough or raised texture.
"wallcoverings which create a textured finish"

We typically associate textures with physical attributes and how a surface may feel

Because this is a computer vision course, we want to talk about the appearance aspect

Image Textures

A pattern with repeating elements. Repetitions can have some variation, but can still separate what repeats vs. what stays the same



Visually, a texture must be perceived as a pattern instead of individual elements

vs

Example textures (from Wikipedia)

Objects vs. Texture

Often, the same thing can be an object or a texture, depending on the scale of consideration.

Adapted from T. Darrell

Why Analyze Visual Textures?

For visual perception:

- Indicative of material properties
- Important appearance cue (shapes, boundaries, textures), especially if shape is similar across objects

For computer vision:

- Representation-wise, we want a feature one step above basic building block of colour, simple filters, and or edges.

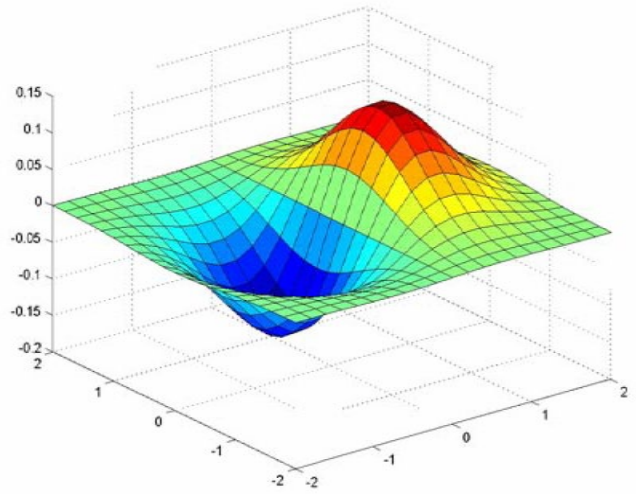
Adapted from T. Darrell

Filter Banks

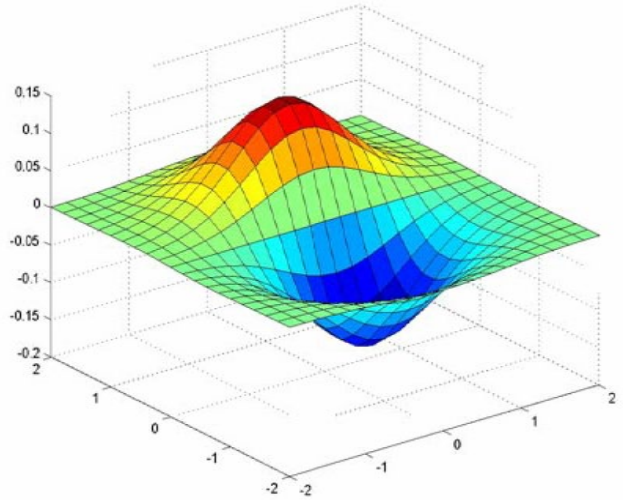
Gabor filters, filter bank response as a representation

How Should We Represent Textures?

Remember our every-so-handy derivative of Gaussian filters?

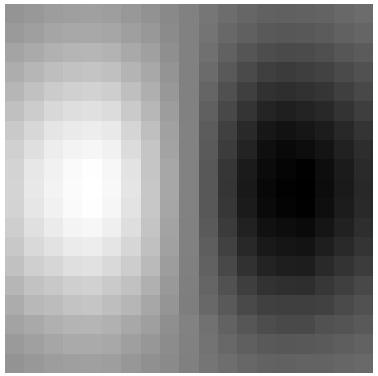


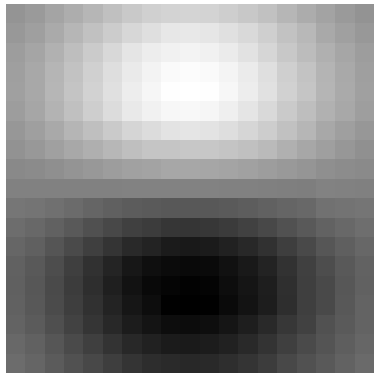
x-direction



y-direction

To which gradient direction do these filters correspond?

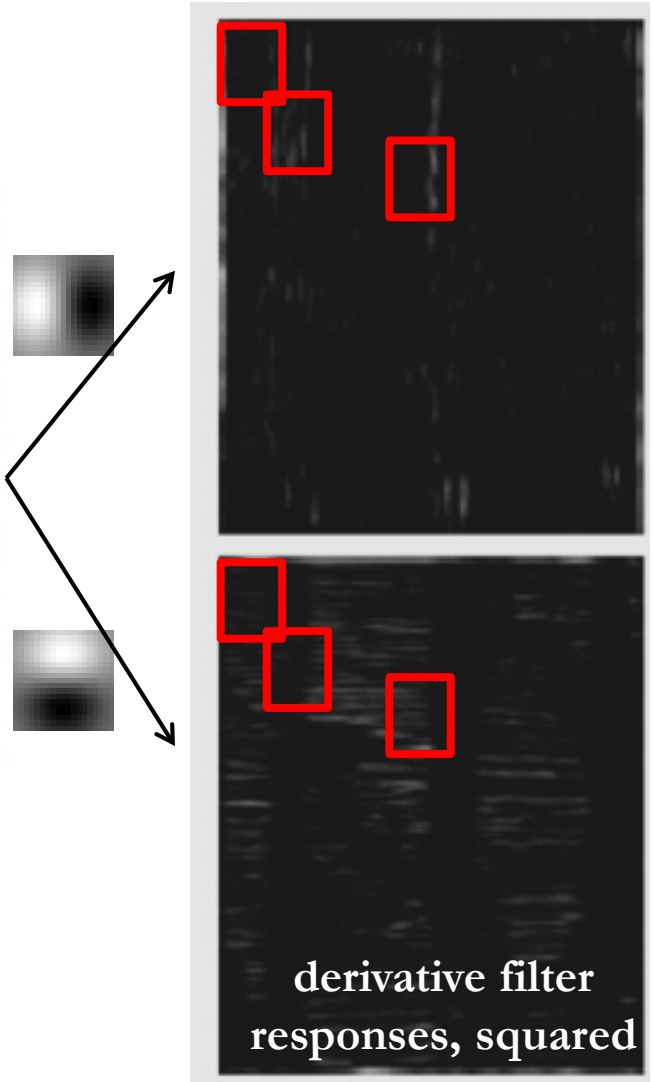




Adapted from L. Lazebnik

Texture representation: example

original image



derivative filter responses, squared

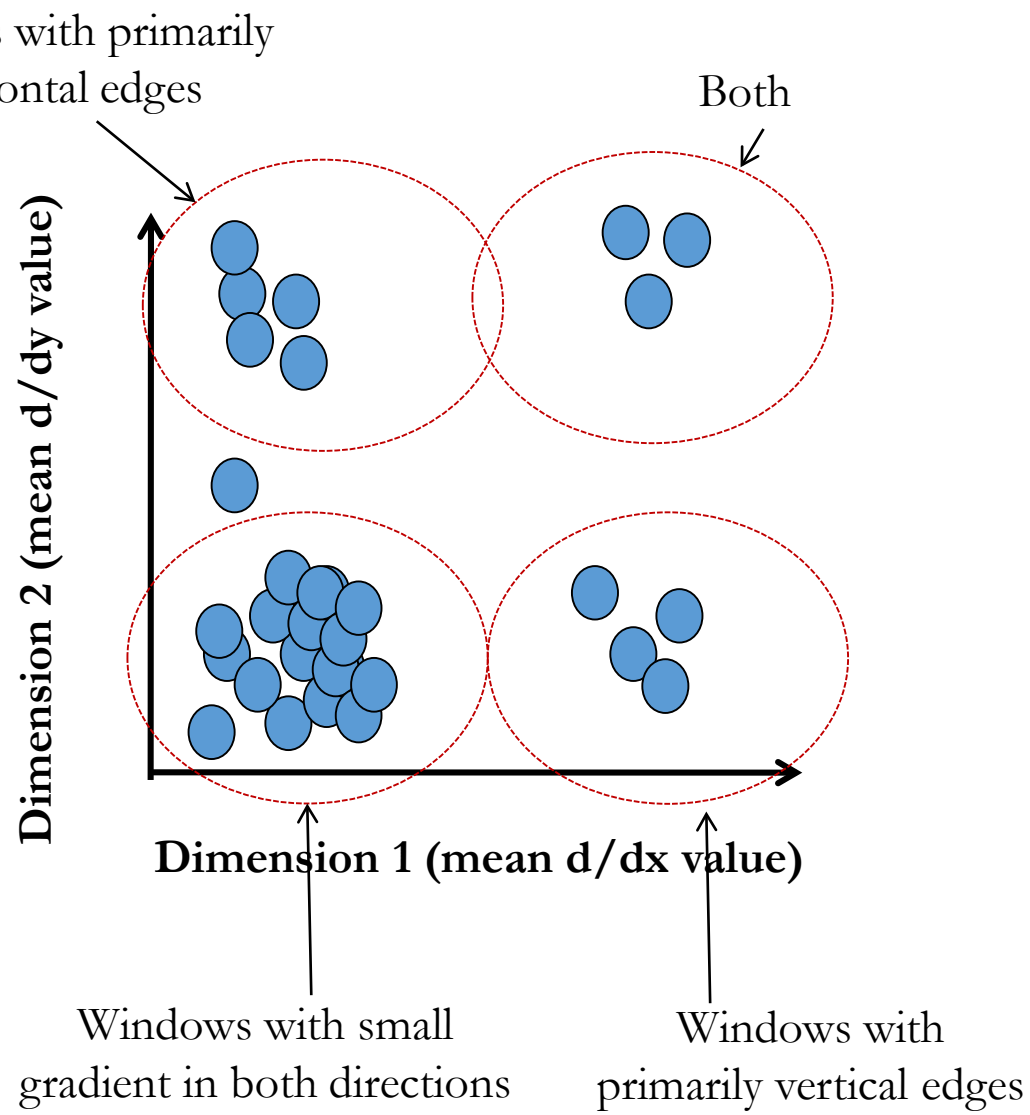
	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20	

statistics to summarize patterns in small windows

Slide Credit: K. Grauman

Texture representation: example

visualization of assignment to texture "types"



	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20

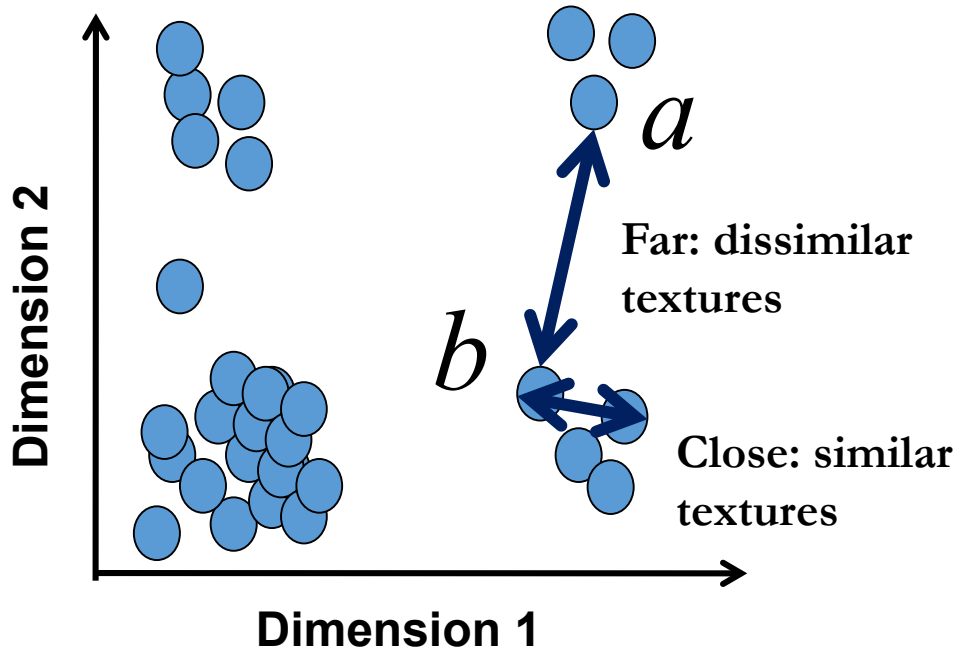
⋮

statistics to summarize patterns in small windows

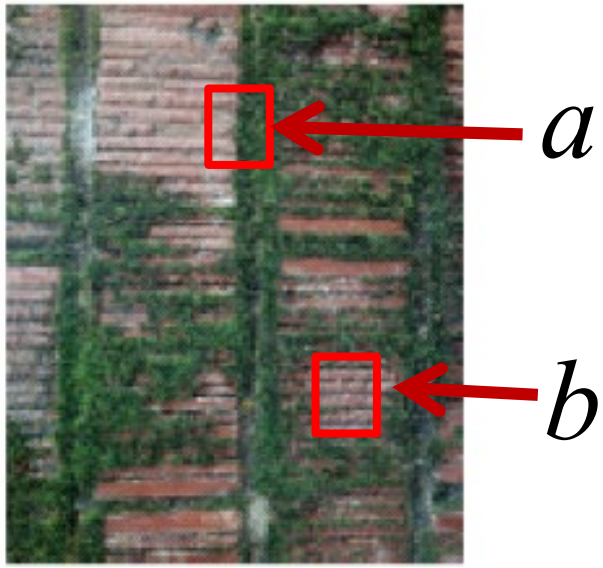
Slide Credit: K. Grauman

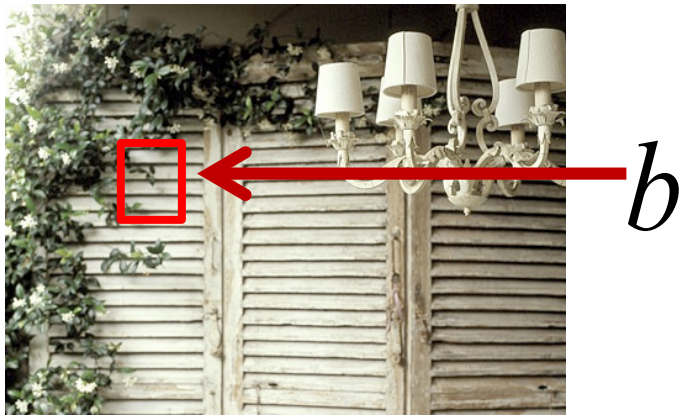
Texture representation: example

Distance reveals how dissimilar texture from window **a** is from texture in window **b**.



$$D(a,b) = \|a - b\|$$
$$= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

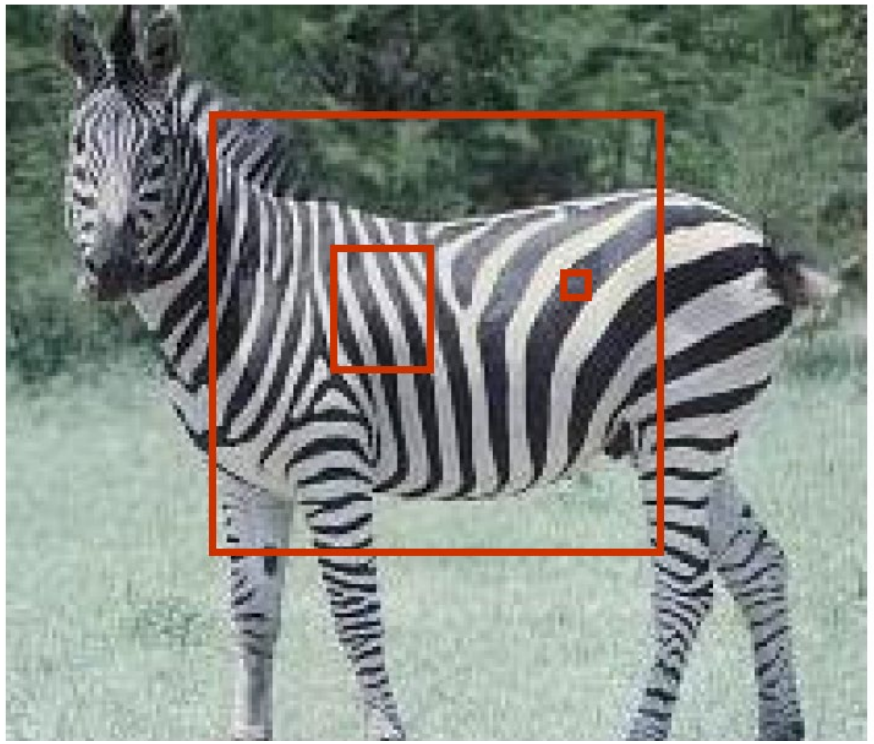




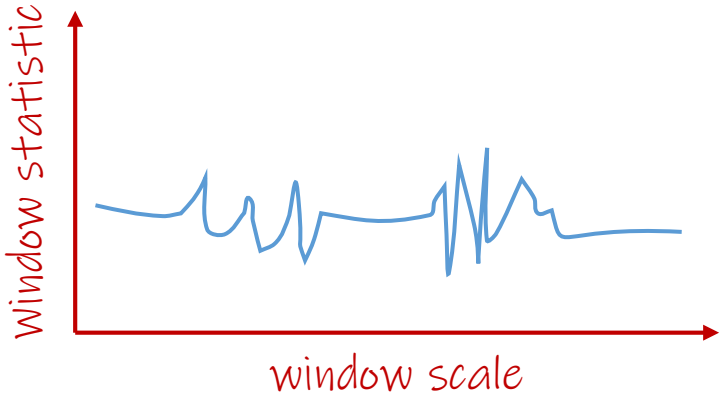
Slide Credit: K. Grauman

Texture representation: window scale

How do we know the relevant window size for collecting statistics?



Perform scale selection by looking for window scale where statistic (texture representation) does not change much.



Adapted from K. Grauman

Filter Responses as Representations

- Previous example used two filters, and resulted in a 2-dimensional feature vector to represent texture in a window.
  - mean x- and y-gradients revealed something about local structure.
- Generalize to apply multiple filters: a "filter bank"
- For *d* filters, feature vectors are now *d*-dimensional.
  - still can think of nearness and farness in feature space

A measure of distance; simplest is L2/Euclidean

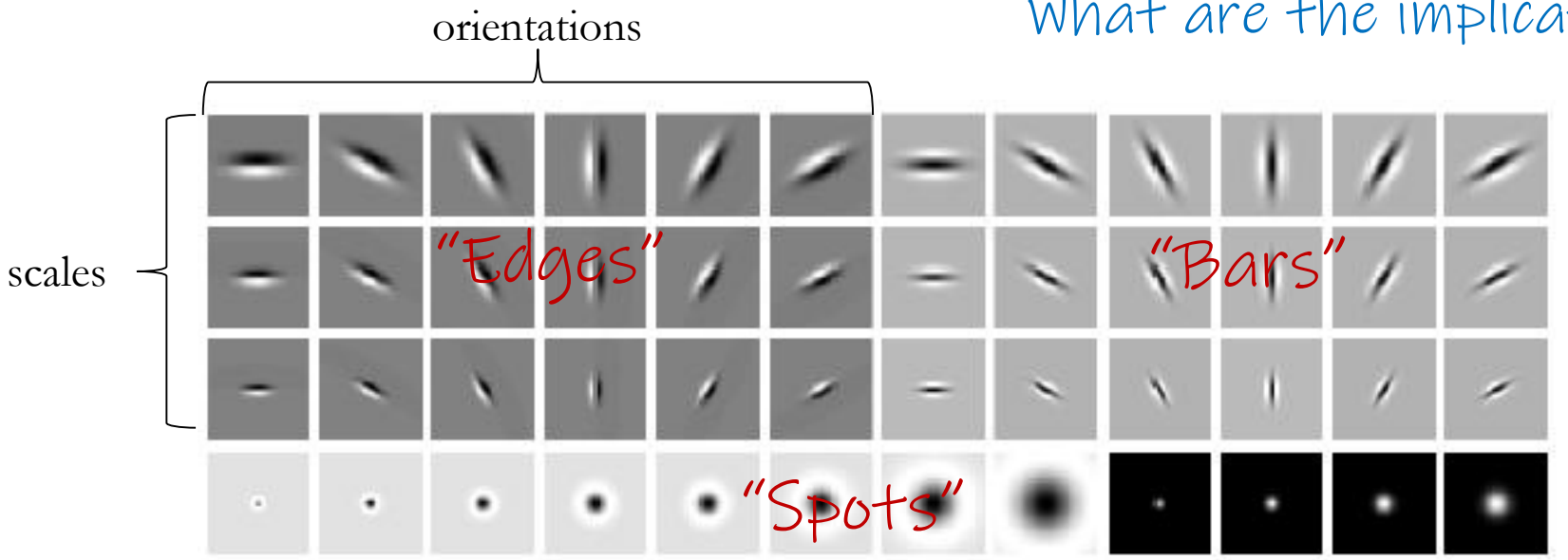
$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

HWQ: How is this distance measure impacted if the range of values for a_1 and b_1 are 100x or 1000x that of a_2 and b_2? What can we do to mitigate these effects?

Adapted from K. Grauman

Filter Banks

How many filters do we want, and how different should they be? What are the implications?



What filters to put in the filter bank?

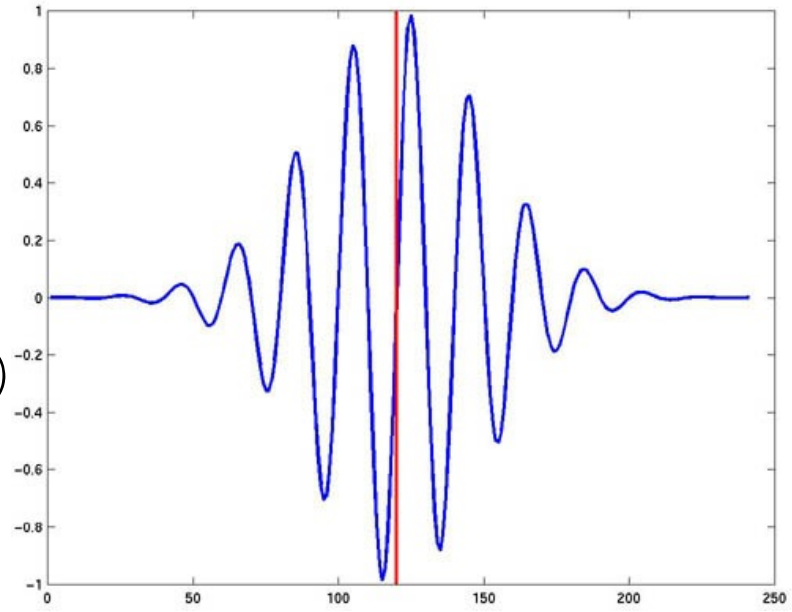
Typically, a combination of different patterns at various scales and orientations.

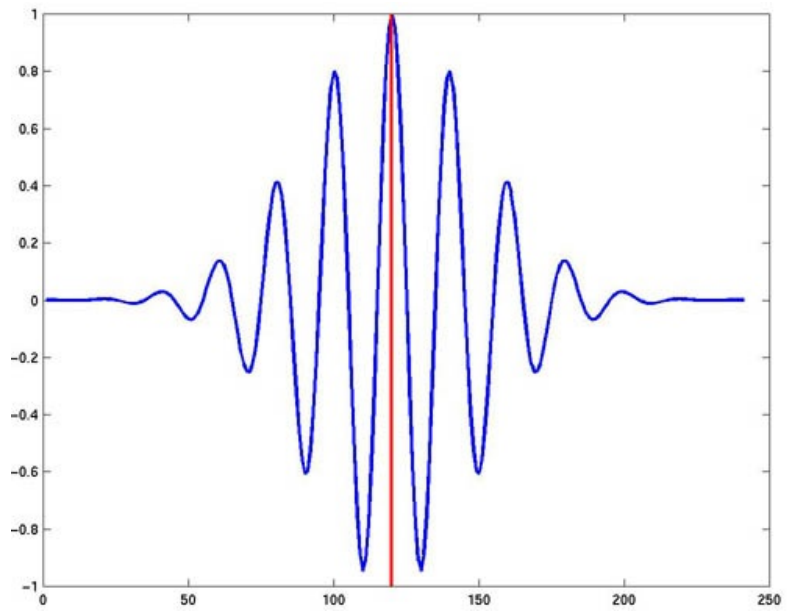
Expressing filters mathematically? A convenient option is Gabor Filters.

Adapted from K. Grauman

Gabor Filters (1D examples)

Gabor filters combine sinusoids with an exponential (Gaussian) envelope.





rate of decay for exponential

frequency of sinusoid

f corresponds to $\frac{1}{\lambda}$ on slide 17

$$e^{-\frac{x^2}{2\sigma^2}} \sin(2\pi f x) \qquad e^{-\frac{x^2}{2\sigma^2}} \cos(2\pi f x)$$

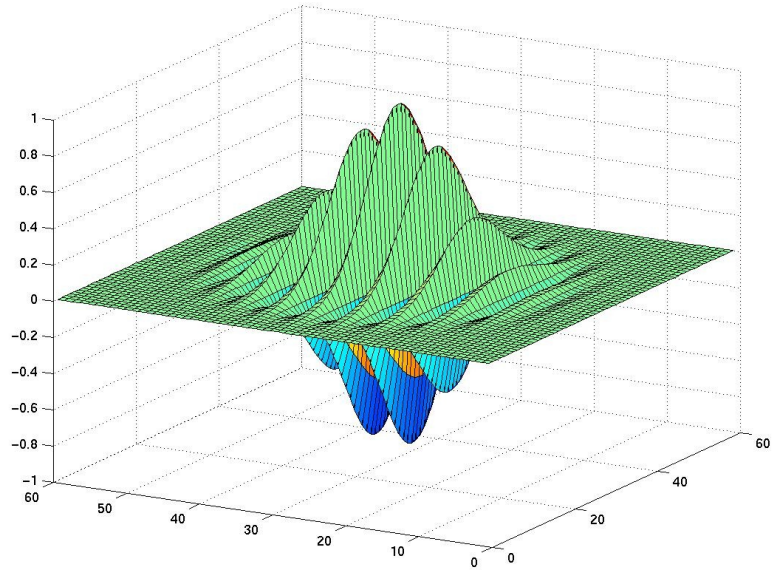
exponential envelope

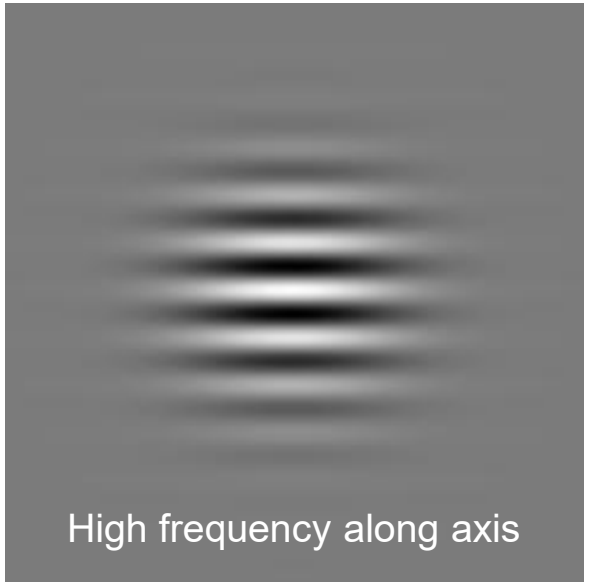
sinusoid

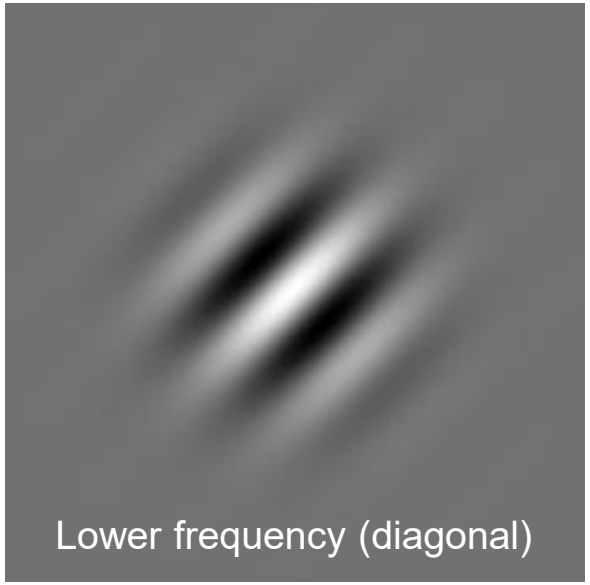
2D Gabor Filters

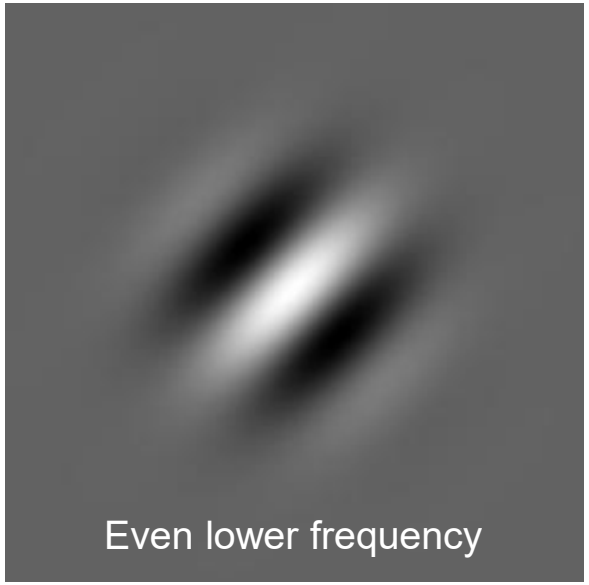
$$f_{mn} = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{m^2+n^2}{2\sigma^2}\right] \sin\left[\frac{2\pi(\cos[\omega]m + \sin[\omega]n)}{\lambda} + \phi\right]$$

This is just the real component, used for the odd filter; can also have a version using cosine, for the even filter.



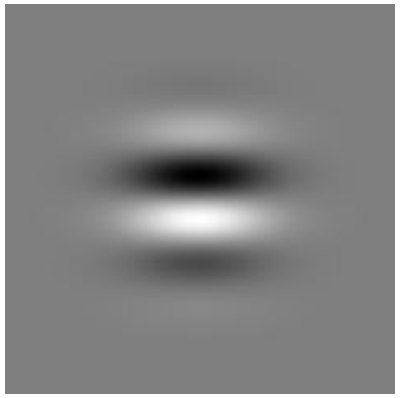




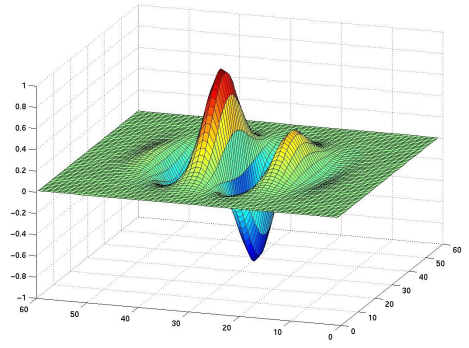


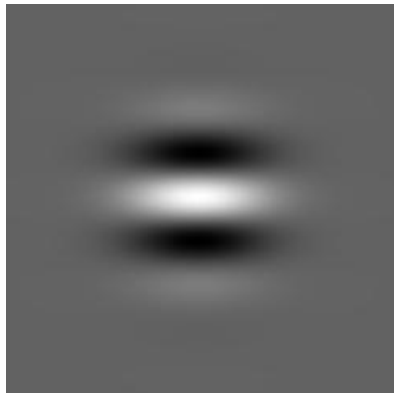
Slide credit: K. Kitani

Approximate Derivatives

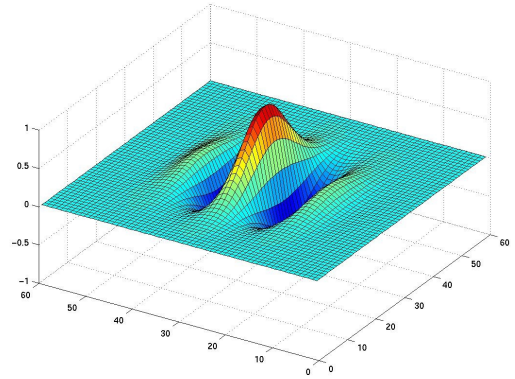


odd
Gabor filter



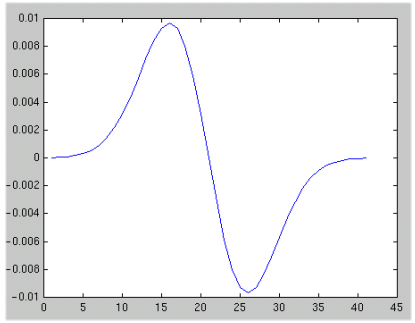


even
Gabor filter

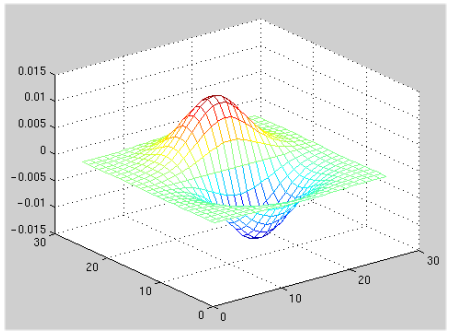


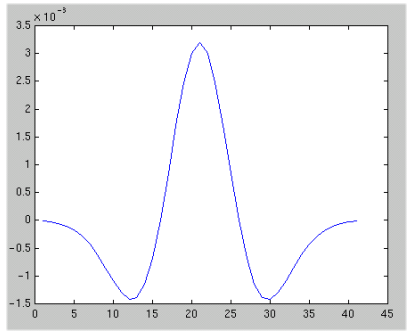
… looks a lot like…

… looks a lot like…

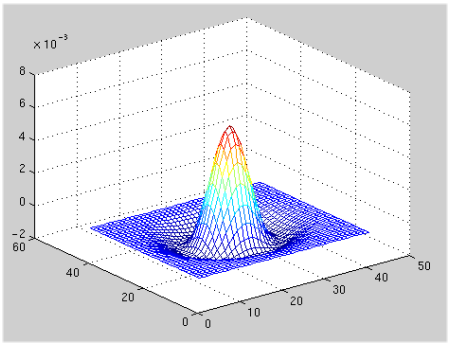


Gaussian
Derivative





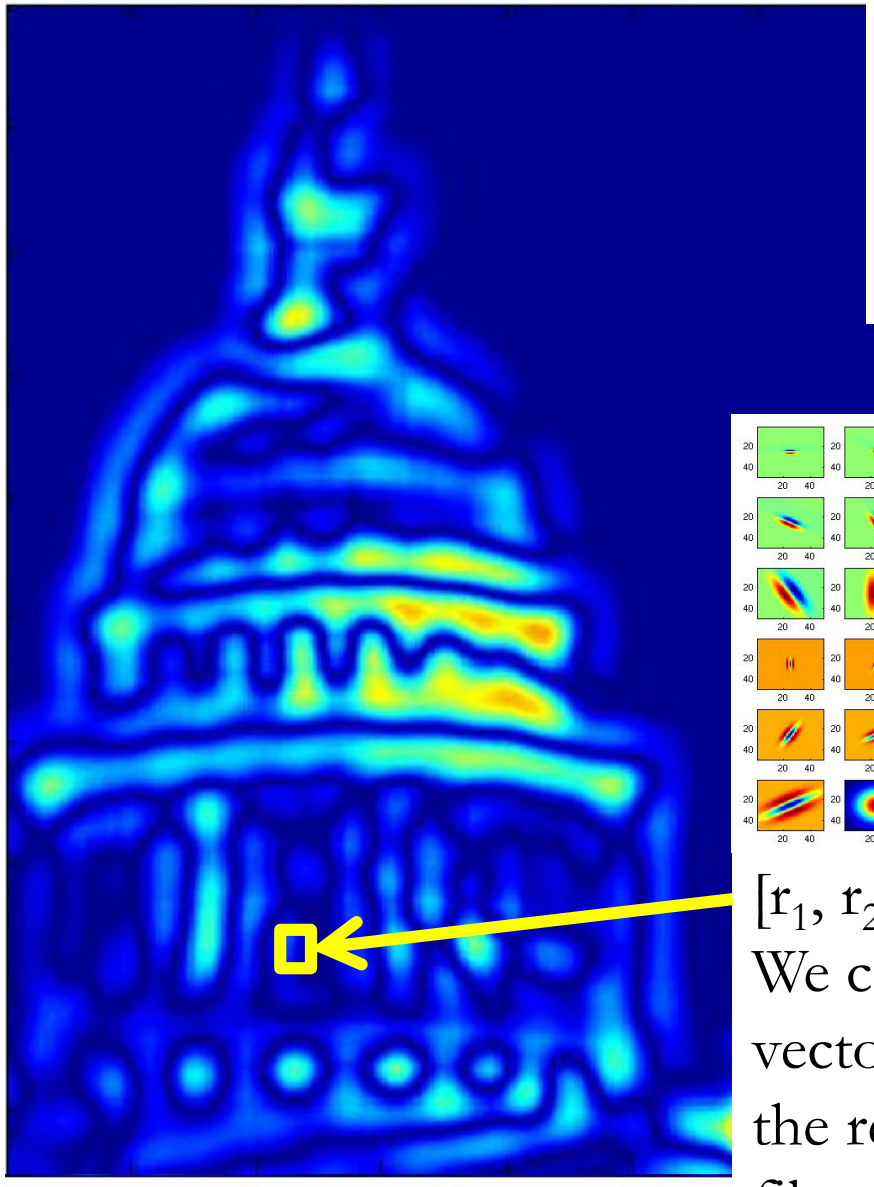
Laplacian of
Gaussian
(2nd derivative)

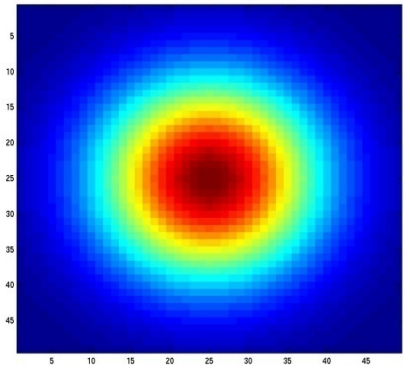


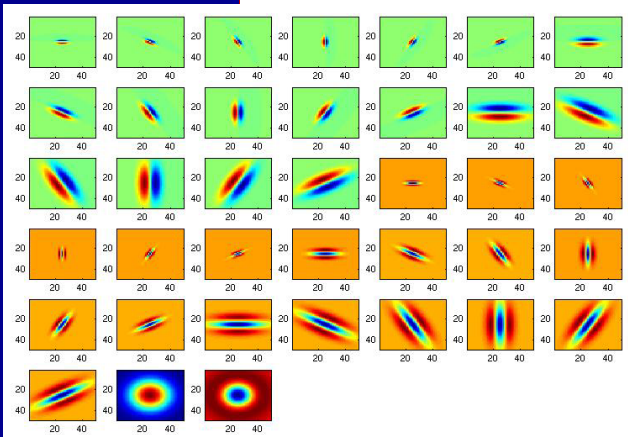
If scale is small compared to the frequency, the Gabor filters can be used as approximate derivative operators!

Slide credit: K. Kitani

Magnitude of Response

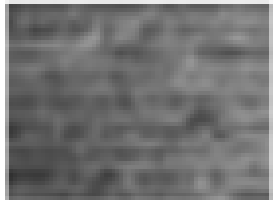






$[r_1, r_2, \ldots, r_{38}]$

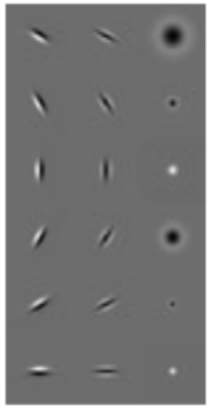
We can form a feature vector at each pixel from the responses to each filter in the filterbank.



magnified image

original image

filter bank



filter responses

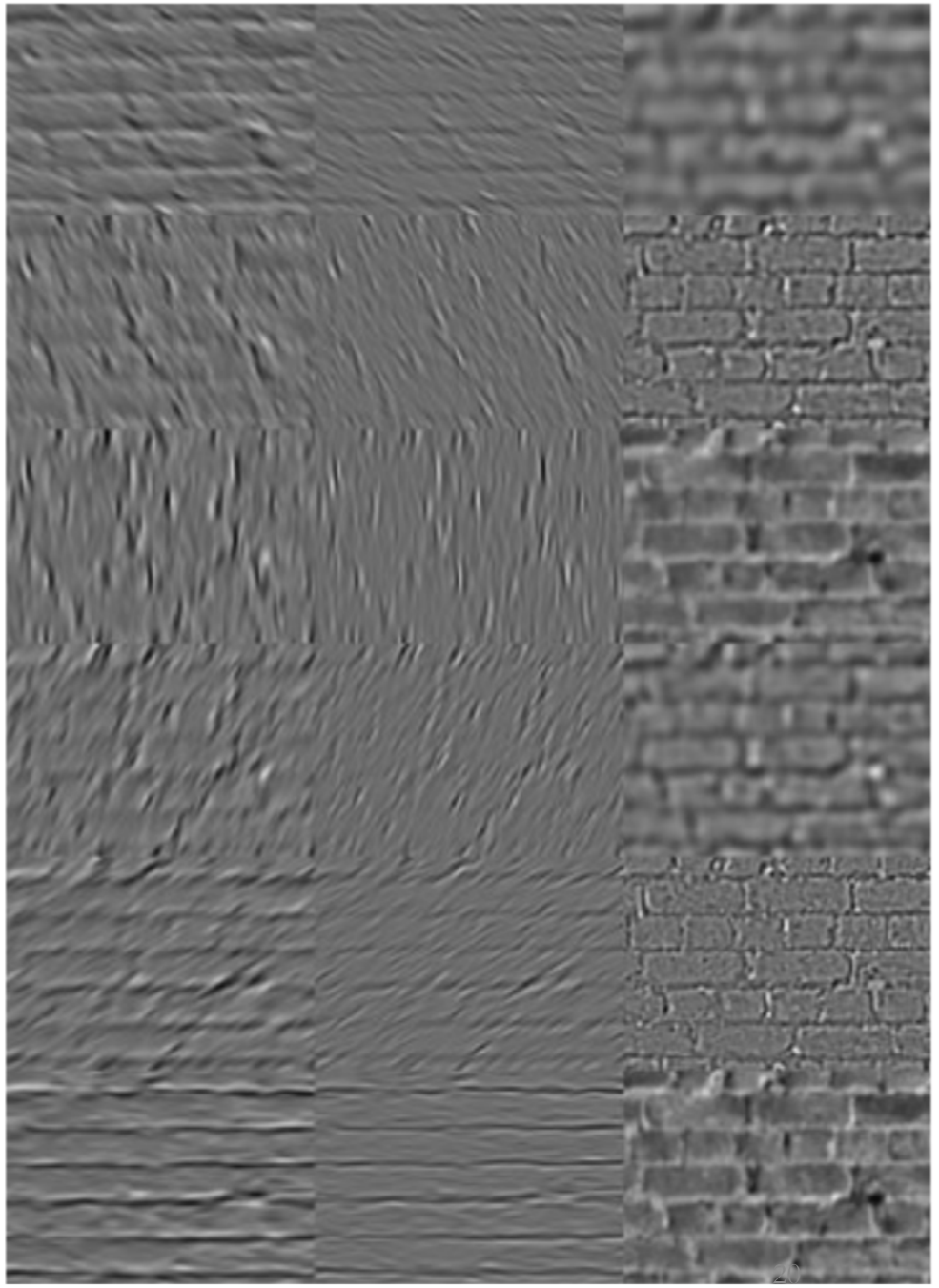


FIGURE 6.5: Filter responses for the oriented filters of Figure 6.4, applied to an image of a wall. At the **center**, we show the filters for reference (but not to scale, because they would be too small to resolve). The responses are laid out in the same way that the filters are (i.e., the response map on the top left corresponds to the filter on the top left, and so on). For reference, we show the image at the **left**. The image of the wall is small, so that the filters respond to structures that are relatively large; compare with Figure 6.6, which shows responses to a larger image of the wall, where the filters respond to smaller structures. These are filters of a fixed size, applied to a small version of the image, and so are equivalent to large-scale filters applied to the original version. Notice the strong response to the vertical and horizontal lines of mortar between the bricks, which are at about the scale of the bar filters. All response values are shown on the same intensity scale: lighter is positive, darker is negative, and mid-gray is zero.

Images from Forsyth & Ponce, CV: A Modern Approach

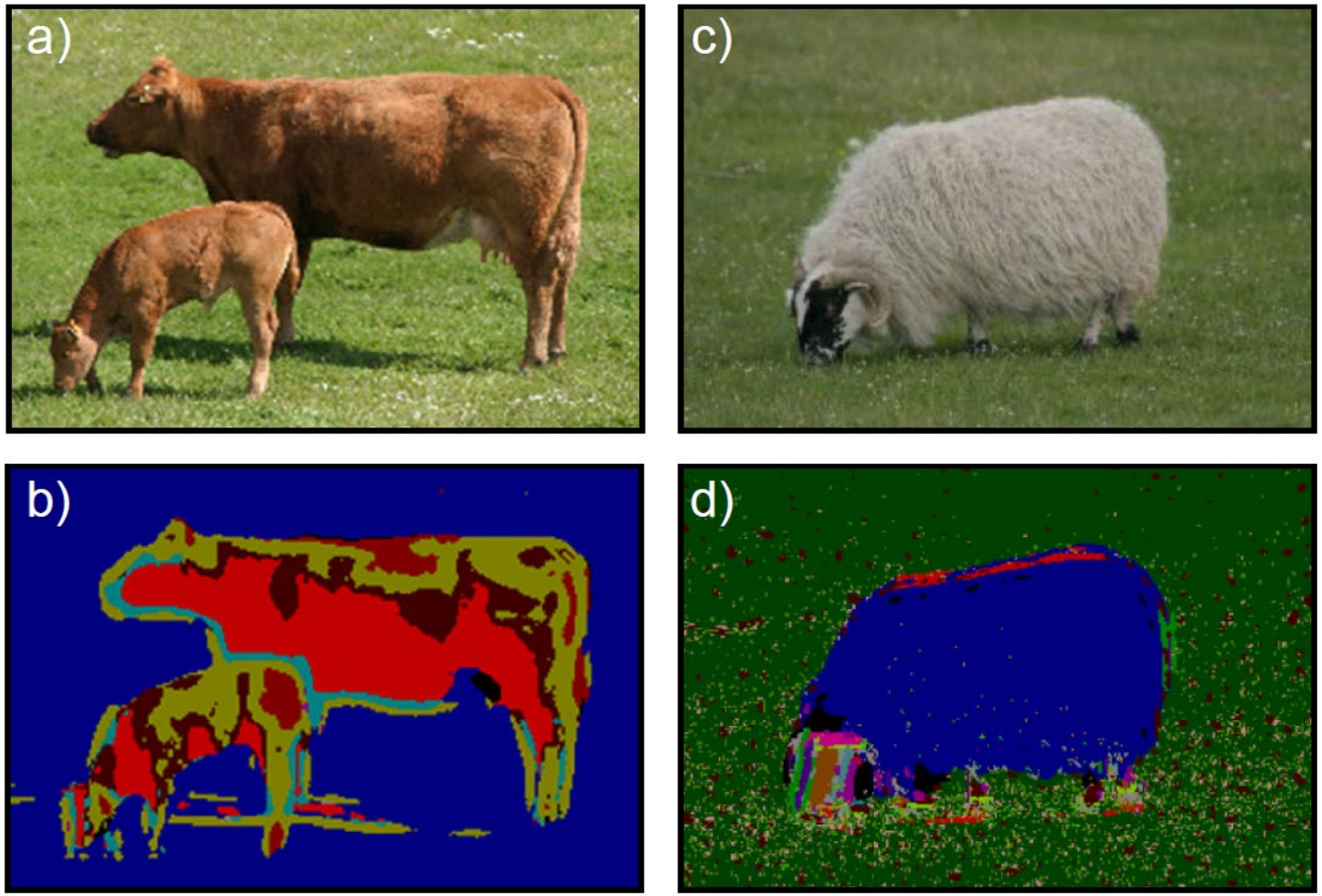
Textons

How to build a texton dictionary; histogram of textons
Texture boundaries,

Textons

Characterize texture by replacing each pixel with integer representing texture 'type'.

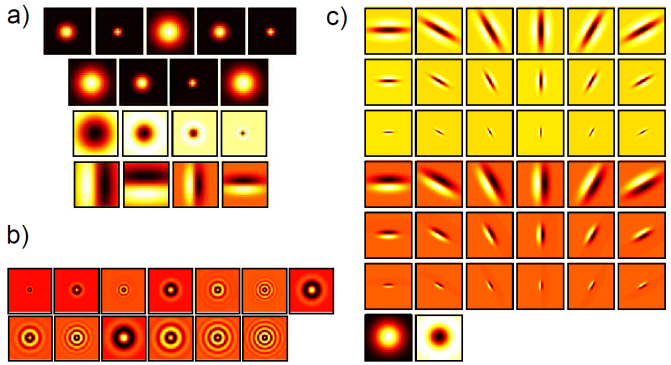
Each colours denotes a 'type', based on a characteristic response to the filter bank.



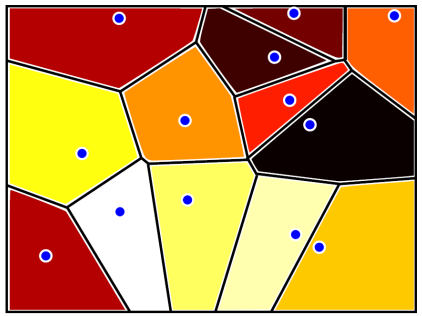
Computer vision: models, learning and inference. ©2011 Simon J.D. Prince

Computing Textons

1. Apply filter bank to (training) images.



2. Cluster in feature space and store cluster centers. This forms a texton dictionary.



3. For new (test) image, filter image with same filter bank to get feature representations for each pixel. Based on feature vectors at each pixel, assign to the nearest cluster. The cluster ID is the texton ID.

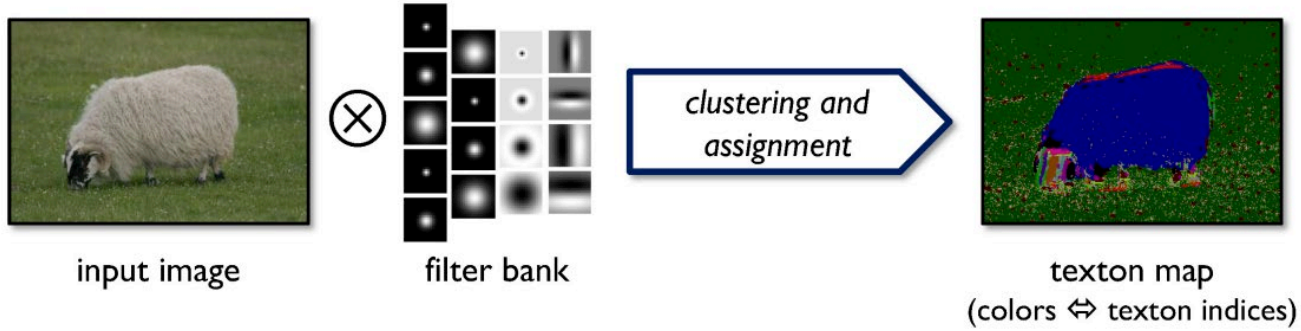
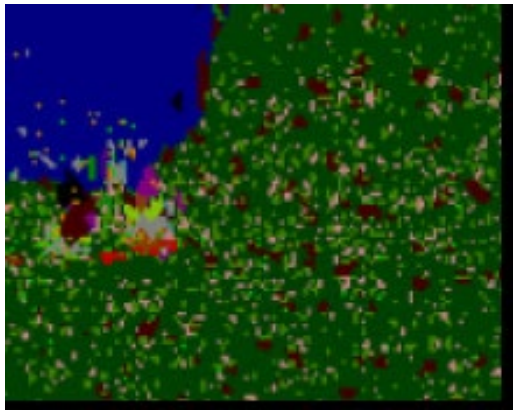


Figure 8: **The process of image textonization.** An input image is convolved with a filter-bank. The filter responses for all pixels in training images are clustered. Finally, each pixel is assigned a texton index corresponding to the nearest cluster center to its filter responses.

Computer vision: models, learning and inference. ©2011 Simon J.D. Prince

Histogram of Textons

- Each pixel can be represented with an ID from a learned texton dictionary.
- Texture is defined by the spatial arrangement of texton IDs, i.e. a first order statistic of texton distribution.
- For a given region, compute a *histogram* of textons as the representation: vector storing number of occurrences of each texton



Texton Histograms for Classification

How should we compare two histograms?

Compare and assign to label of sample with the smallest distance.

http://www.cs.columbia.edu/CAVE

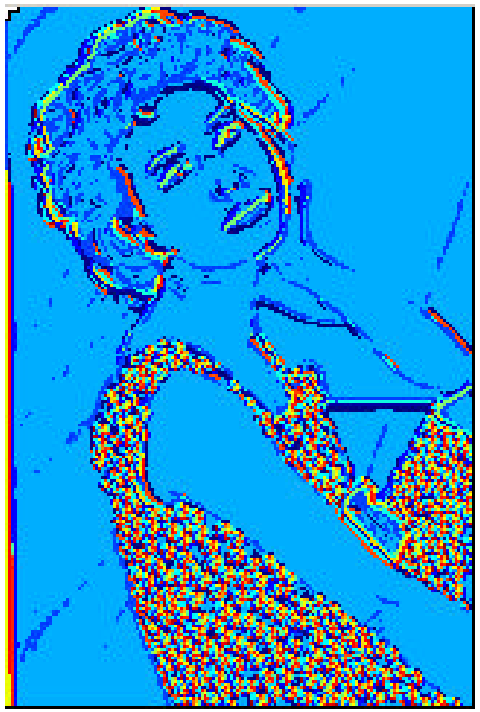
(Leung & Malik, 1999)

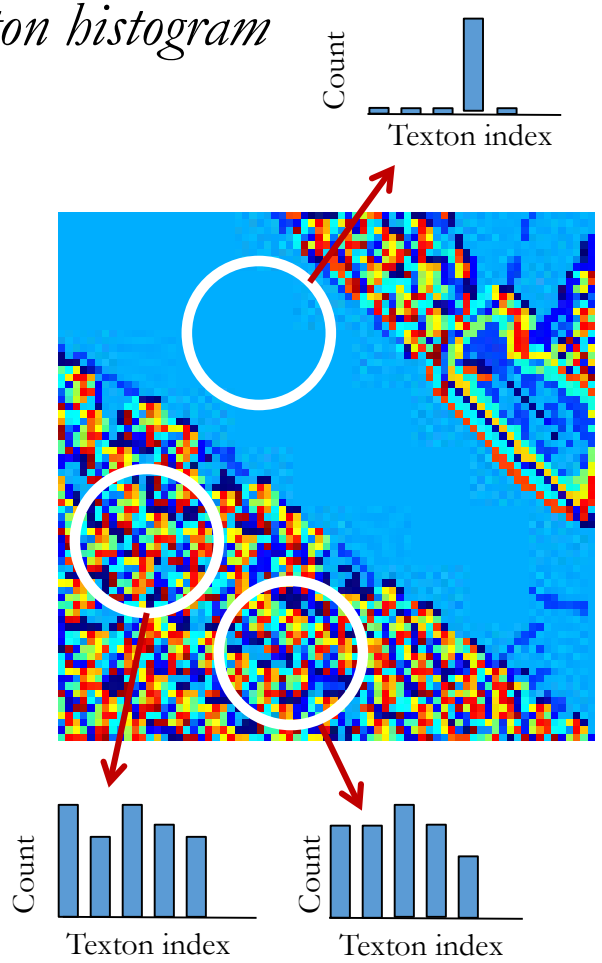
Texton Histograms for Segmentation

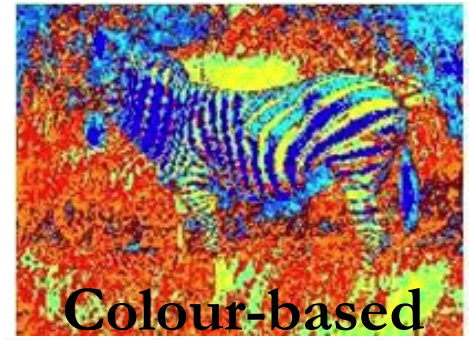
Find "textons" by **clustering** vectors of filter bank outputs.
Describe texture in a window based on *texton histogram*

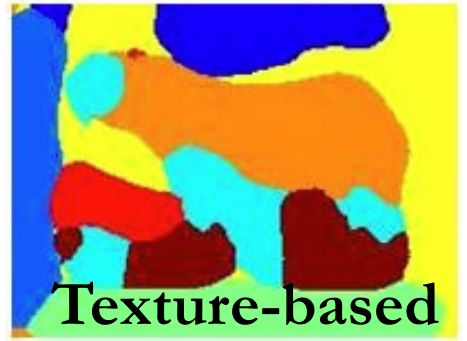
Image

Texton map









Use textons as a feature for segmentation.

Adapted from Lana Lazebnik

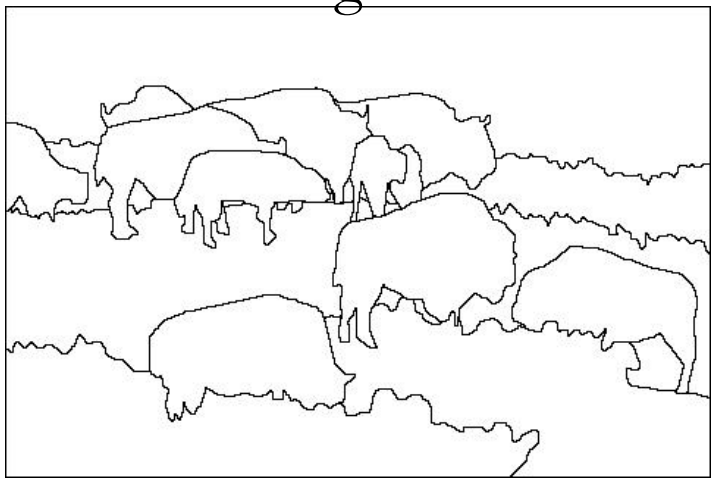
Perceived Boundaries

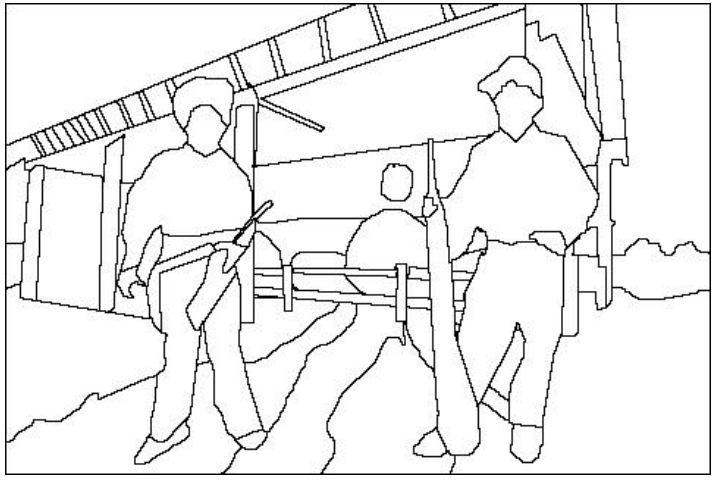
Role of texture in perceived contours;
texture gradients; learned edges.

Low-Level Edges vs. Perceived Boundaries

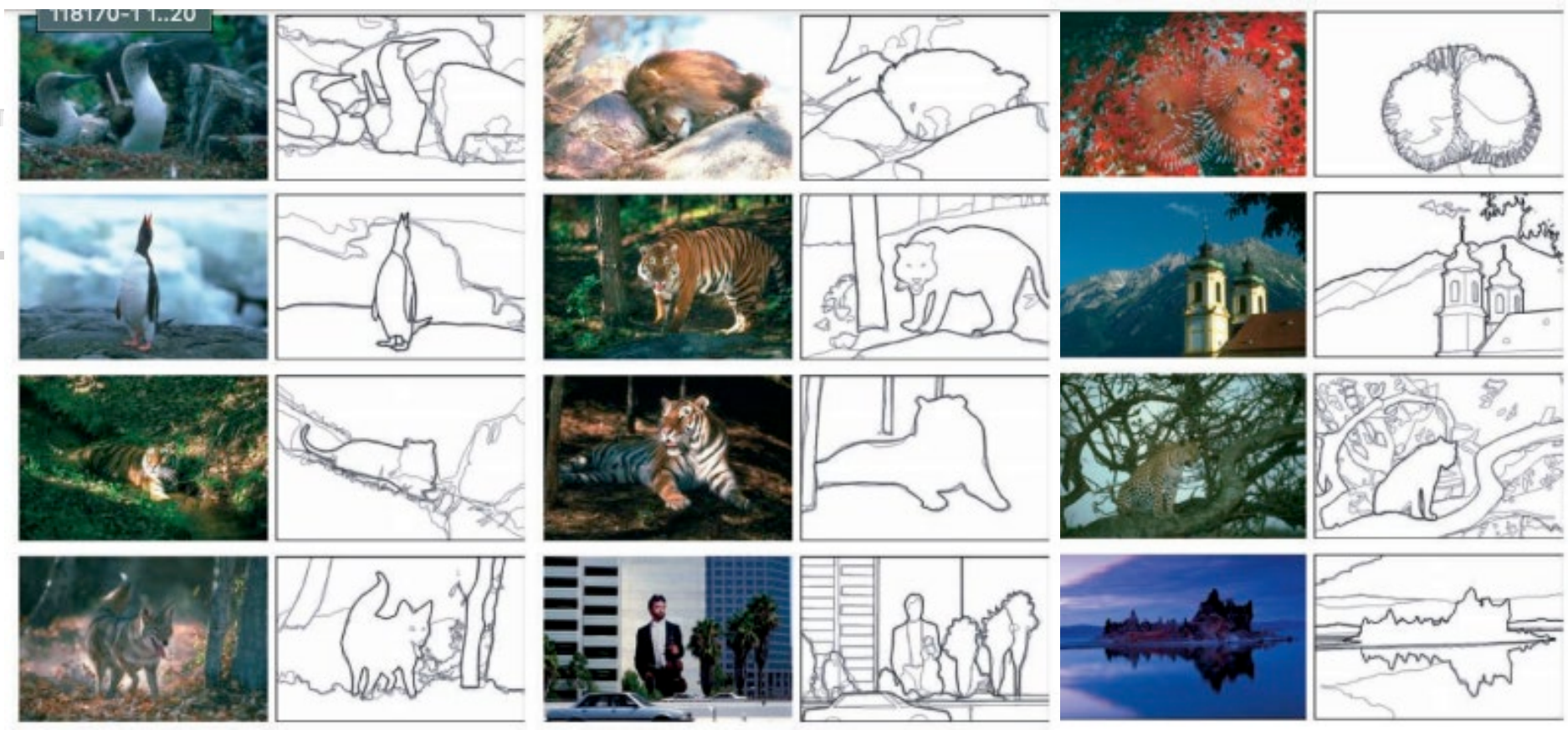
gradient magnitude

human segmentation





Learn boundaries from humans!

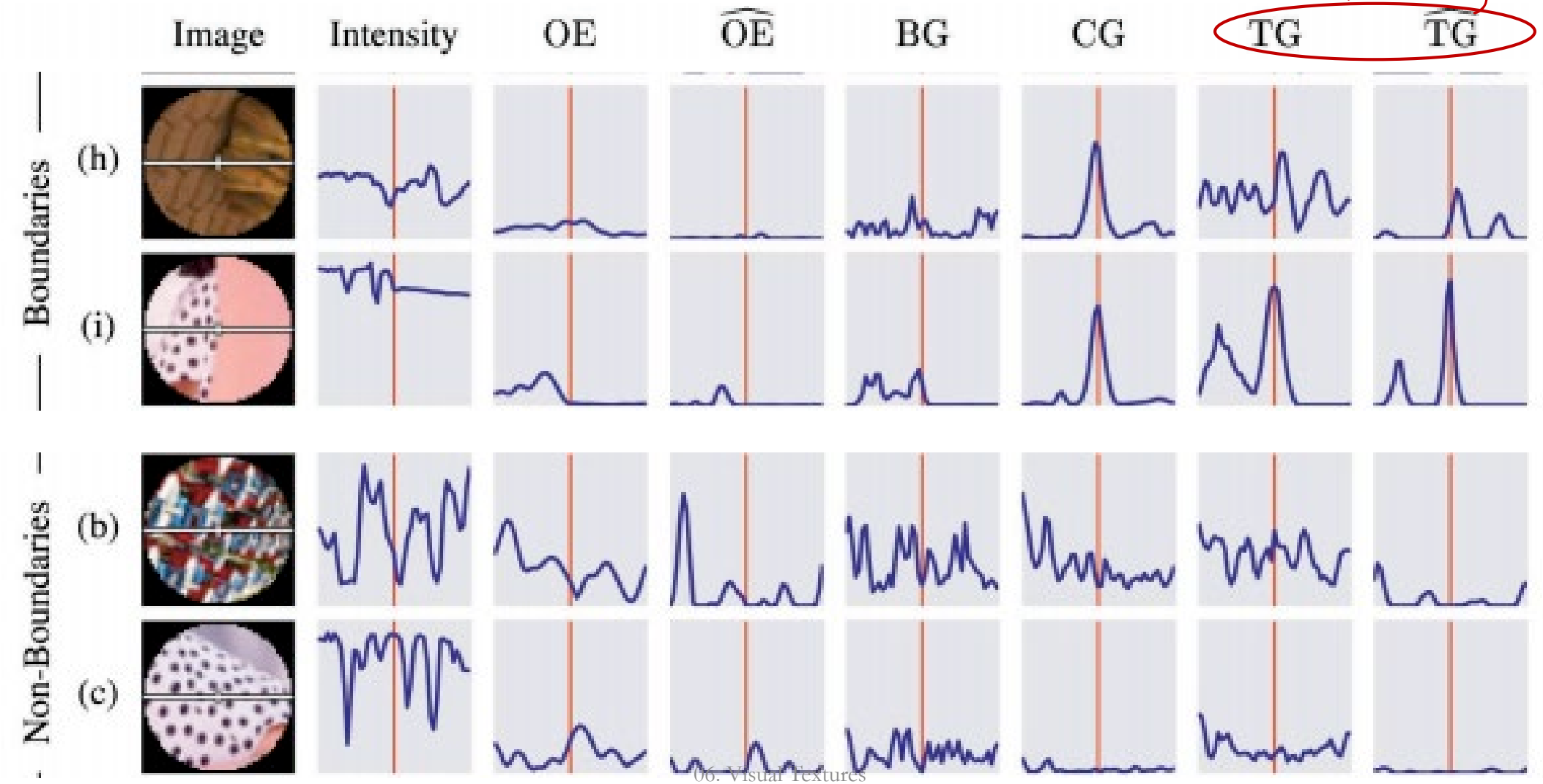


precision-recall curves showing that the resulting detector significantly outperforms existing approaches. Our two main results are 1) that cue combination can be performed adequately with a simple linear model and 2) that a proper, explicit treatment of texture is required to detect boundaries in natural images.

https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/mfm-pami-boundary.pdf

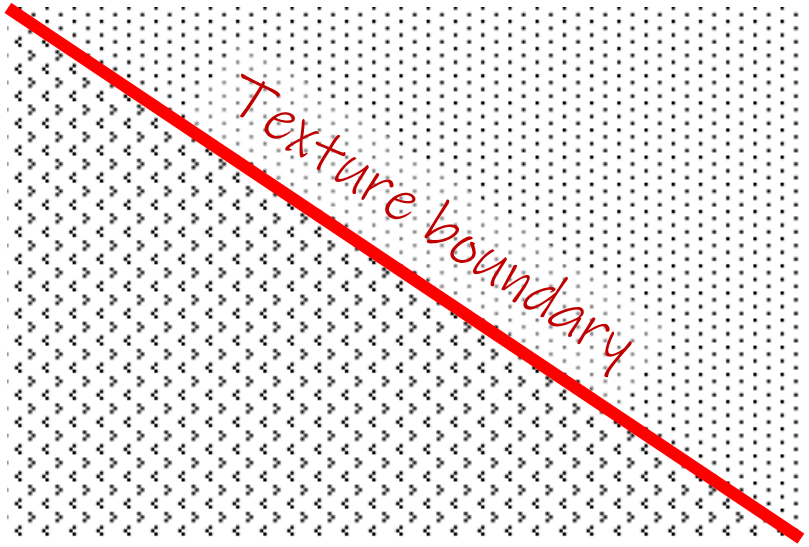
Which Features Indicate Perceived Edges?

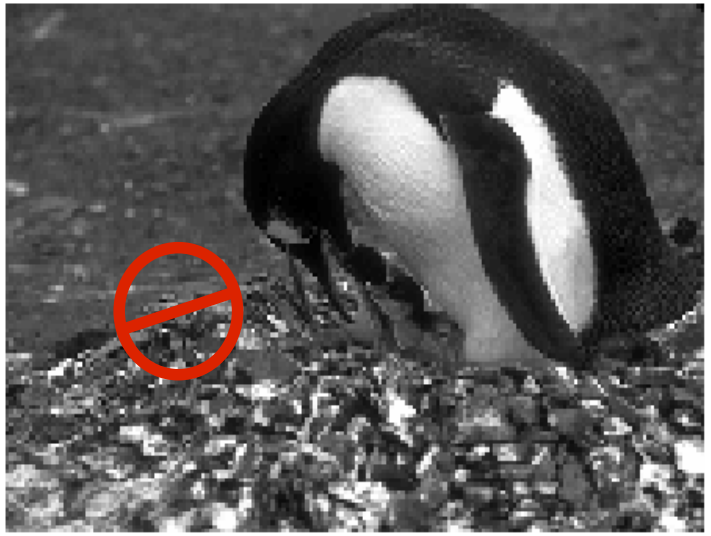
Texture gradients



https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/mfm-pami-boundary.pdf

Using Textures to Identify Boundaries

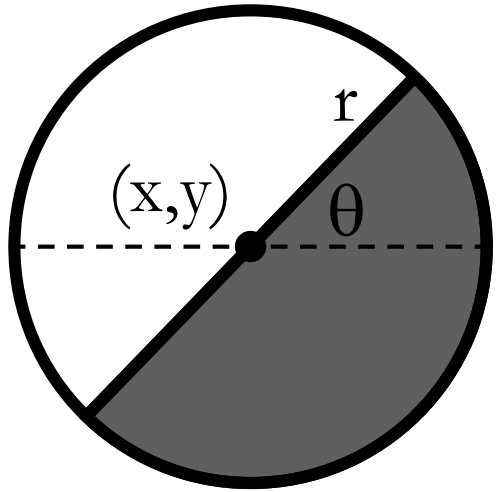




- Regions with very distinct textures are separated by a texture boundary.
- To identify texture boundaries at each location in a scene
  - Consider a disc, split into two halves by a diameter of a particular orientation
  - Measure the difference in texture between the two halves by comparing texton histograms
  - Try all possible orientations

Texture Boundaries (Gradients)

- Texture Gradient TG(x,y,r,θ)
- In each half, compute histogram of textons
- high distance between histograms, suggests more likely boundaries

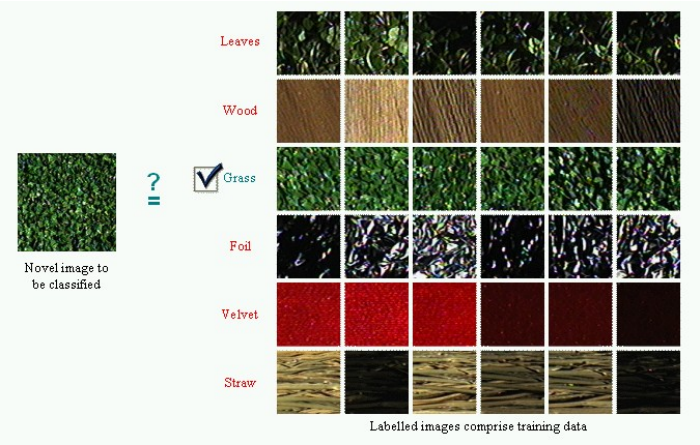


Summary

- Visual textures are a useful property that is often indicative of materials, appearance, etc.
- Filter banks can measure redundant variety of structures in local neighborhood and result in multi-dimensional feature spaces
- Texture representations via filter bank response "summarize" repeating patterns of local structure. They can be used for a variety of applications including segmentation, classification, retrieval, etc.
- Texture plays a strong part in perceived boundaries, can be incorporated into boundary detection algorithms

Supplement

Comparison of material classification vs. segmentation



	Material Classification	Image Segmentation
Input	Image (h x w)	Image (h x w),
Output	Single label: material type	Segmentation image: (h x w)
Algorithm:	Nearest-neighbor classification	Clustering (e.g. k-means)
Data Sample	(h x w) image is represented as a single data point to be classified	each pixel is a single data point to be clustered
Feature Representation of each data sample (input to algorithm)	texton histogram, where histogramming is done over entire h x w image	Several possibilities a. RGB + (x,y); we covered this in L5 b. spatially smoothed filter bank response ← done in demo c. texton histogram (histogramming over a window, e.g. of 5x5 or 7x7 window) ← also done in demo but does not work well. Probably could be improved if one tunes the filters, cluster numbers and window sizes.

Material Classification

Nearest Neighbour Classification

Training set of M samples

- M samples are each represented by feature f, so we have $\{f_1, f_2 \dots, f_M\}$
- the M samples each have some class y (type of material), so we have $\{y_1, y_2, \dots, y_M\}$

Test sample ($M+1^{th}$ sample)

- Represented by same feature as training, f_{M+1}
- We want to find the class y_{M+1}

To classify sample i, we simply give i the label of sample j which has the lowest distance $d(f_i, f_j)$ for j = 1 … M:

$$y_i = y_j, \qquad \text{where } j = \underset{j}{\operatorname{argmin}}\, d(f_i, f_j), \qquad \text{for } j = 1 \dots M$$

Feature Representation

Feature f is the texton histogram. How to compute f? Suppose the training images are all (h x w). We are given a filter bank with D filter kernels $\{k_1, \dots, k_D\}$. Build dictionary of N textons. N does not equal D; often N > D. This implies that f is of size 1xN.

Represent training set M (end up with M texton histograms)

1. Apply filter bank to M training images: input is (M x hw); output is M x hw x D
2. Cluster pixel-wise filter responses into N clusters via k-means, each cluster is one texton:
   - Input: M x hw x D data points; Output: M x hw x 1 texton labels, each label is a value of 1 - N
   - Store N cluster centers to use for test images
3. Take histogram of textons over each training sample; Input: M x hw texton labels; Output: M histograms,
   - each histogram is 1xN (1 bin per texton / cluster); the total sum of elements per histogram is hw;
   - each image is now represented by a 1xN feature, where each dimension of the feature is a count of the number of times that texton occurs in the image. Often, it is a good idea to normalize the histogram instead of using raw counts (esp. if we need to later compare images of different sizes)

Represent test image (with same textons as the training data)

1. Apply filter bank to test image: input is 1 x hw image; Output: 1 x hw x D filter responses
2. Assign textons: input 1 x hw x D filter response; output: 1 x hw x 1 texton labels
   - Use nearest neighbor to assign each pixel's filter response to the nearest of the N cluster centers
   - The cluster ID is the texton ID
3. Take histogram of textons: input: 1 x hw x 1 texton labels, output: 1 histogram
   - Histogram has same N bins as the training images; (optionally normalize histogram)
   - can consider test image now as being represented by a 1xN feature

Image Segmentation

Clustering (e.g. k-means)

No training vs. testing, just the single image to be clustered of size h x w

- Each pixel is each represented by feature *f*, so we have $\{f_{11}, f_{12}, \ldots, f_{1w}, \ldots, f_{h1}, \ldots, f_{hw}\}$
- Cluster the h x w samples into k clusters
- Replace each sample with cluster ID to get segmented image

** note: texton histograms, while good for material classification, are a poor feature choice for segmentation since histograms are not very (locally) precise since they are aggregated from some window. It is better to directly use the spatially smoothed feature response.

Feature Representation

There are many options for feature *f*. Consider the following versions:

A. Colour + position, so f is 5-dimensional (r,g,b,x,y); directly cluster as per Lecture 6

B. Filter bank response, based on a filter bank of D filter kernels $\{k_1, \ldots, k_D\}$

1. Apply filter bank to image to be segmented: input is (h x w); output is h x w x D
2. (optionally) spatially smooth the filter output: input and output are both (h x w x D)
   - Apply blurring kernel (e.g. box, or Gaussian) to each of the D filter responses individually
3. Cluster filter outputs, where each feature f is a D-dimensional feature

C. Texton histogram based on a local region using a texton dictionary of N textons

1. Apply filter bank to image to be segmented: input is (h x w); output is h x w x D
2. Cluster pixel-wise filter responses into N clusters via k-means, each cluster is one texton: input is hw x D filter responses, output is hw x 1 texton labels
3. Take a histogram of textons within a local window (of e.g. 5 x 5 or 7x7): input is hw x 1 texton labels, output: hw x N
   - Each pixel is represented by a histogram (taken at local window, centered at that pixel) with N bins
4. Cluster pixels, using histogram as a feature, i.e. each f is an N-dimensional feature