

Lecture 7

Textures





Lecture Outline

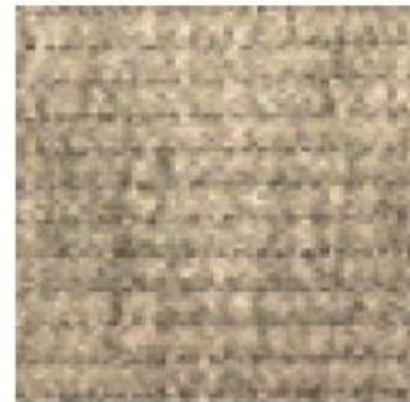
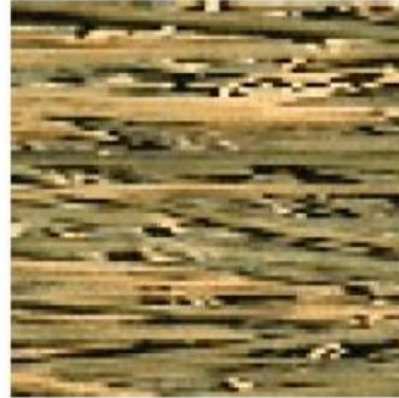
- › Textures
 - Defining textures in images
 - Texture representation, Filter bank
- › Clustering
 - K-means clustering
 - Using clustering to form histograms for texture feature occurrences
- › Applications

What defines a texture?



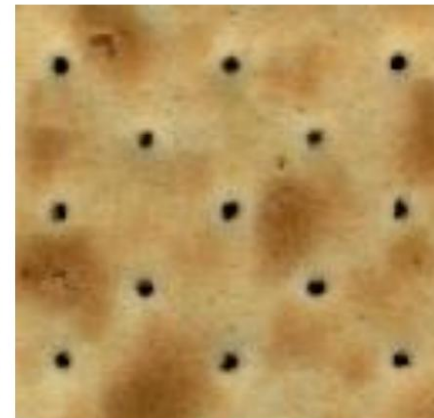
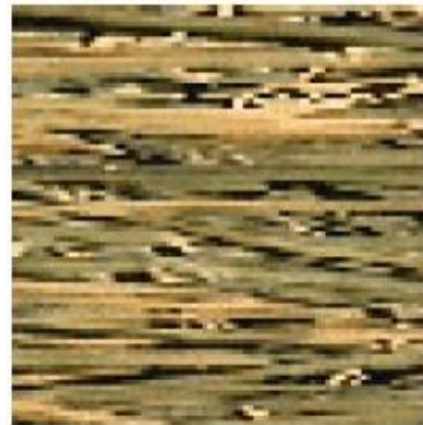
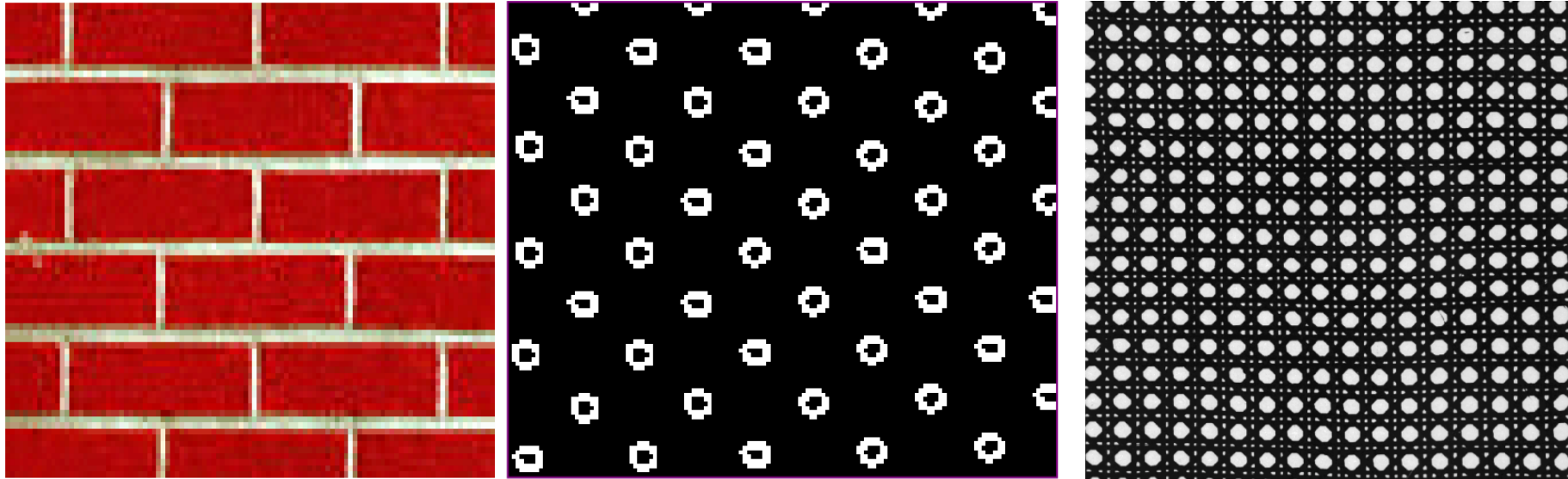


Textures





Regular & Irregular patterns





Texture-related tasks

› **Shape from texture**

- Estimate surface orientation or shape from image texture

› **Segmentation/classification** from texture cues

- Analyze, represent texture
- Group image regions with consistent texture

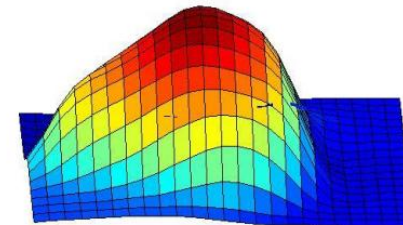
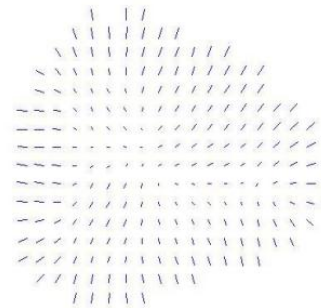
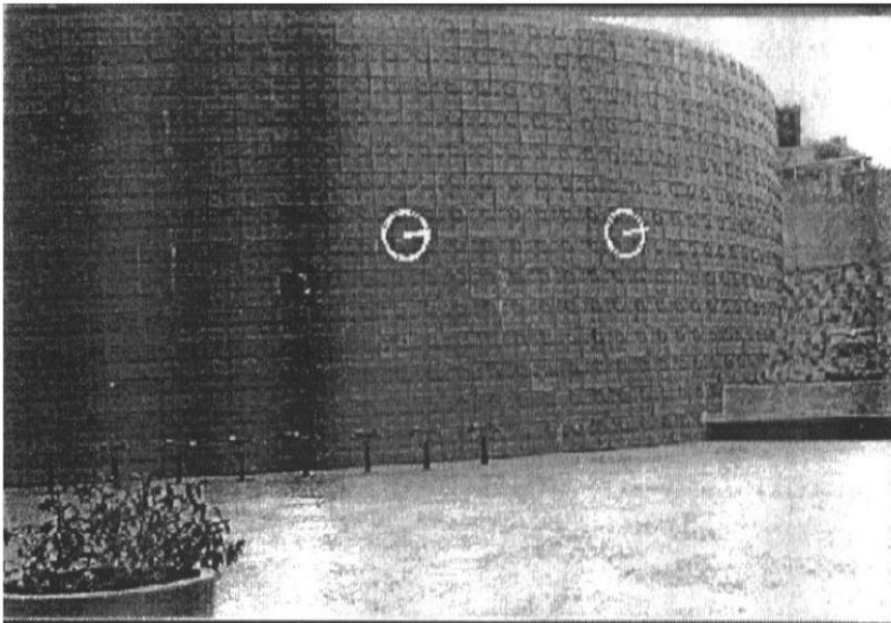
› **Synthesis**

- Generate new texture patches/images given some examples



Texture-related tasks

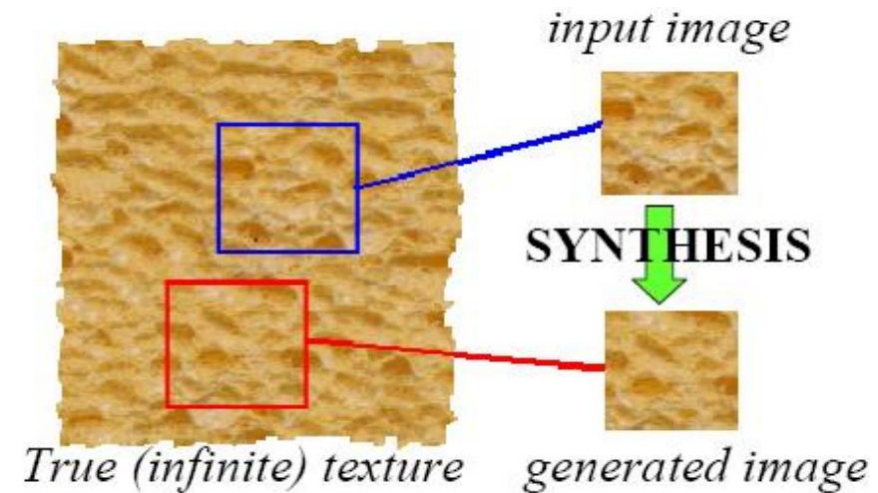
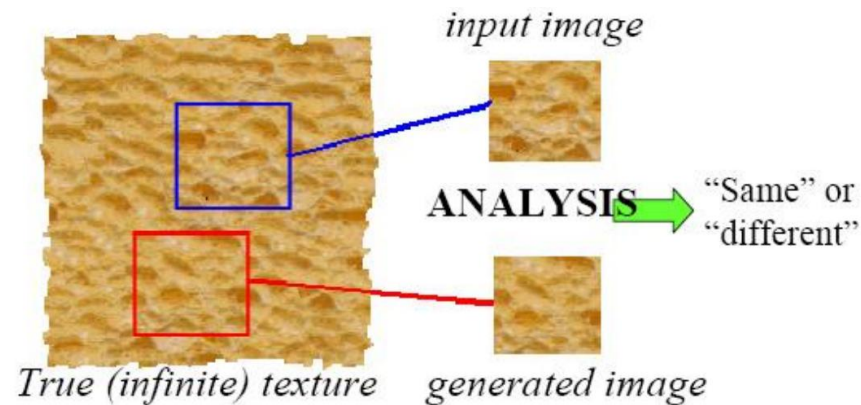
- › Shape from texture
 - Estimate surface orientation or shape from image texture





Analysis vs. Synthesis

- › **Why** analyse texture?
- › **Why** synthesize texture?





Texture-related tasks

> Shape from texture

- Estimate surface orientation or shape from image texture

> **Segmentation/classification** from texture cues

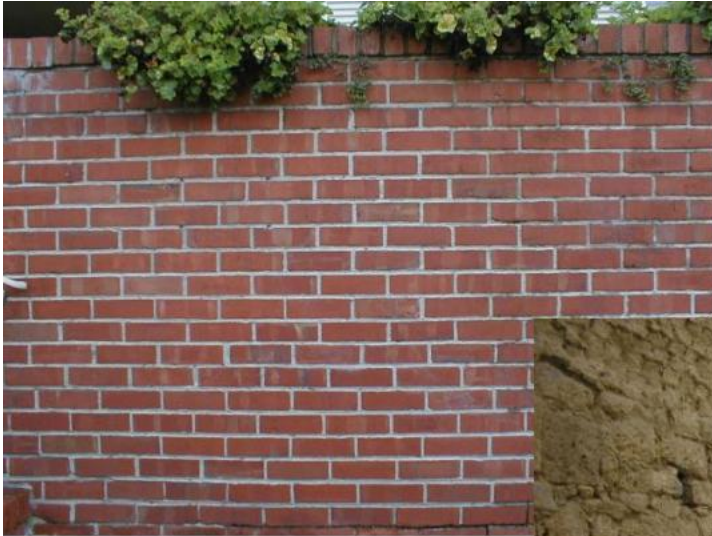
- Analyze, represent texture
- Group image regions with consistent texture

> Synthesis

- Generate new texture patches/images given some examples



Textures from walls





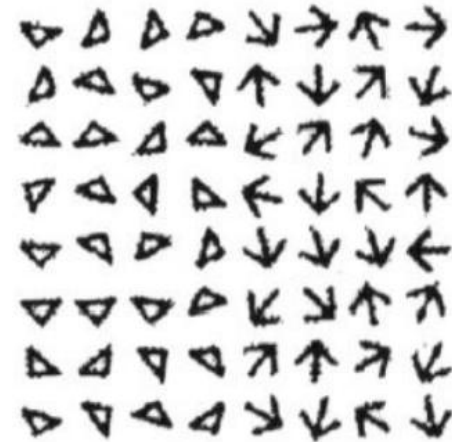
Textures from animals





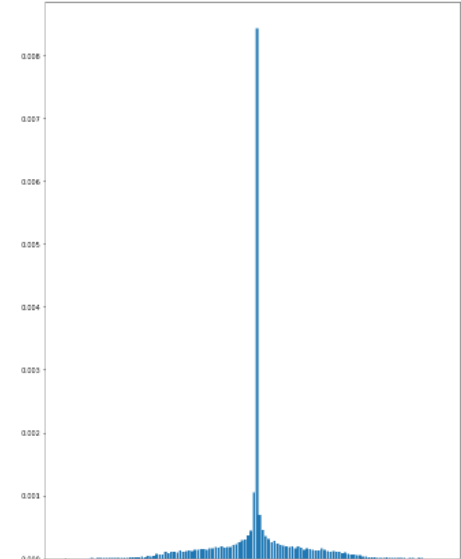
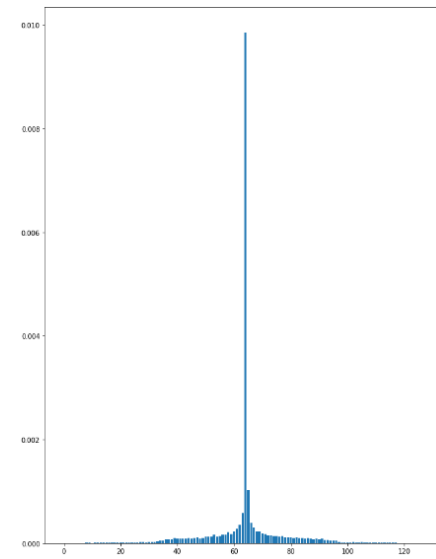
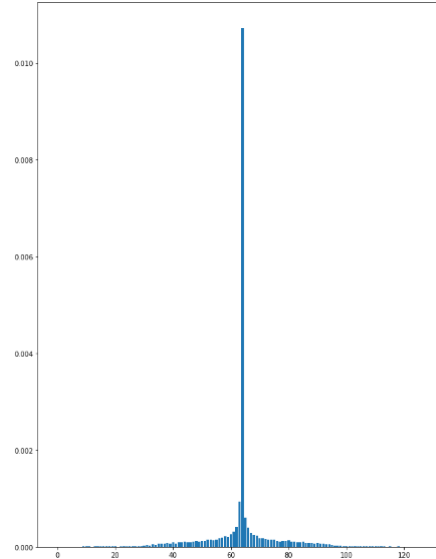
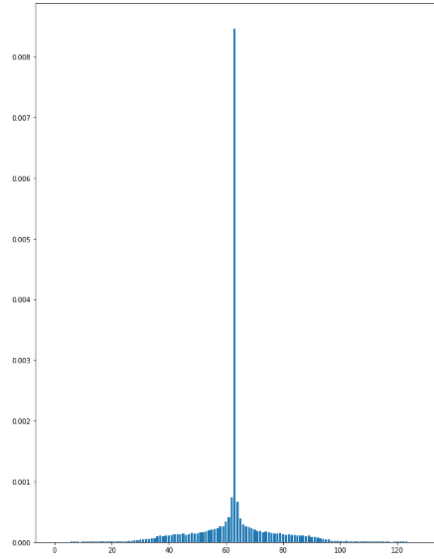
Are edges sufficient?

- › What kind of response will we get with an edge detector for these images?
- › Is it **good enough** to “represent” the image content?





Are edges sufficient?





What about this image?





Why analyse texture?

> Important to perception:

- Indicates material's properties
- Appearance cue, especially if shape is similar (e.g. orange vs. apple)

> In technical terms...

- **Representation**-wise, we want a feature one step above “basic building blocks” of filter outputs, edges, etc.

How to represent these textures as data?



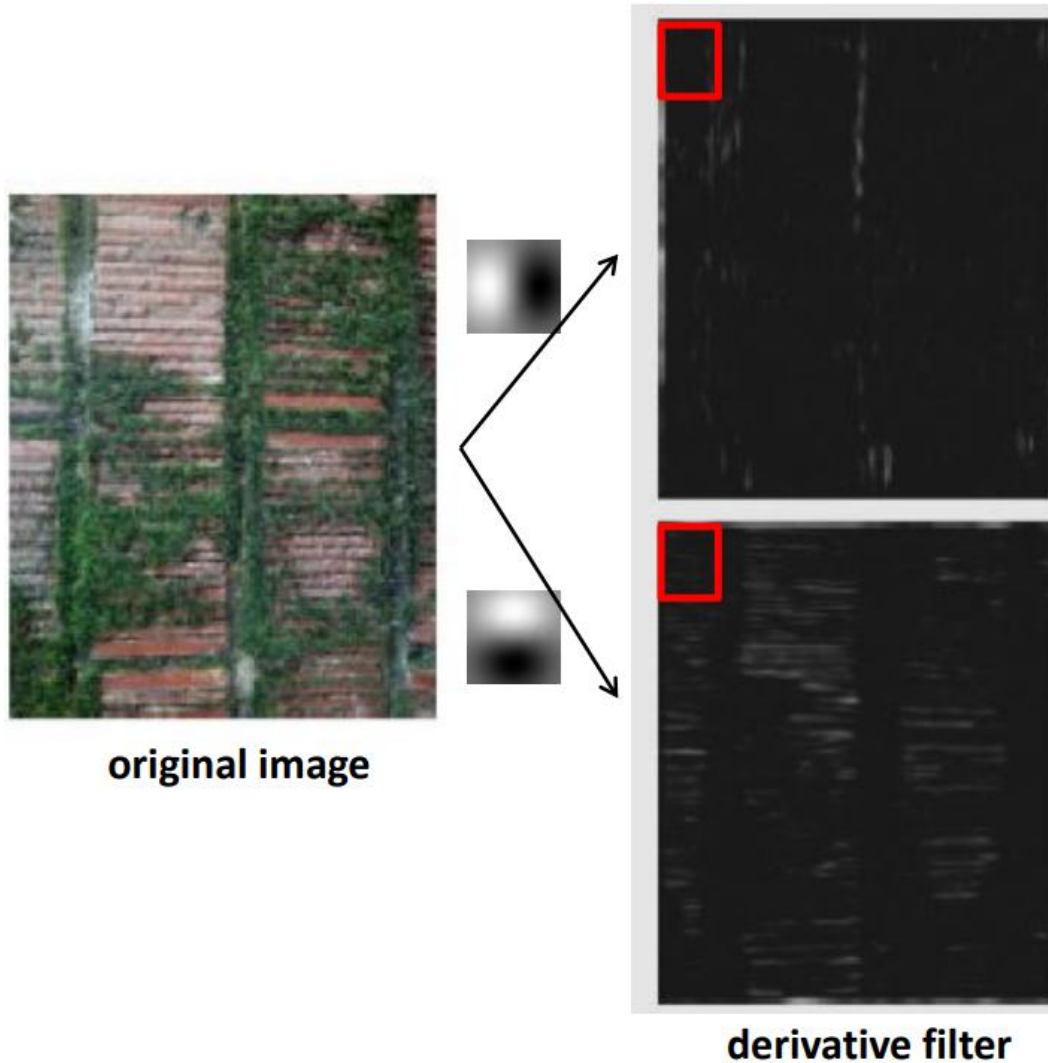


Texture Representation

- › Textures are made up of **repeated local patterns**
 - **Find** these “patterns”
 - › Use filters that LOOK like patterns (spots, bars, raw patches, etc.)
 - › Consider magnitude of response of these patterns
 - **Describe** their statistics within each local window (or “neighbourhood”)
 - › Mean, standard deviation
 - › (and at a higher level..) Histogram of feature occurrences



Texture Representation



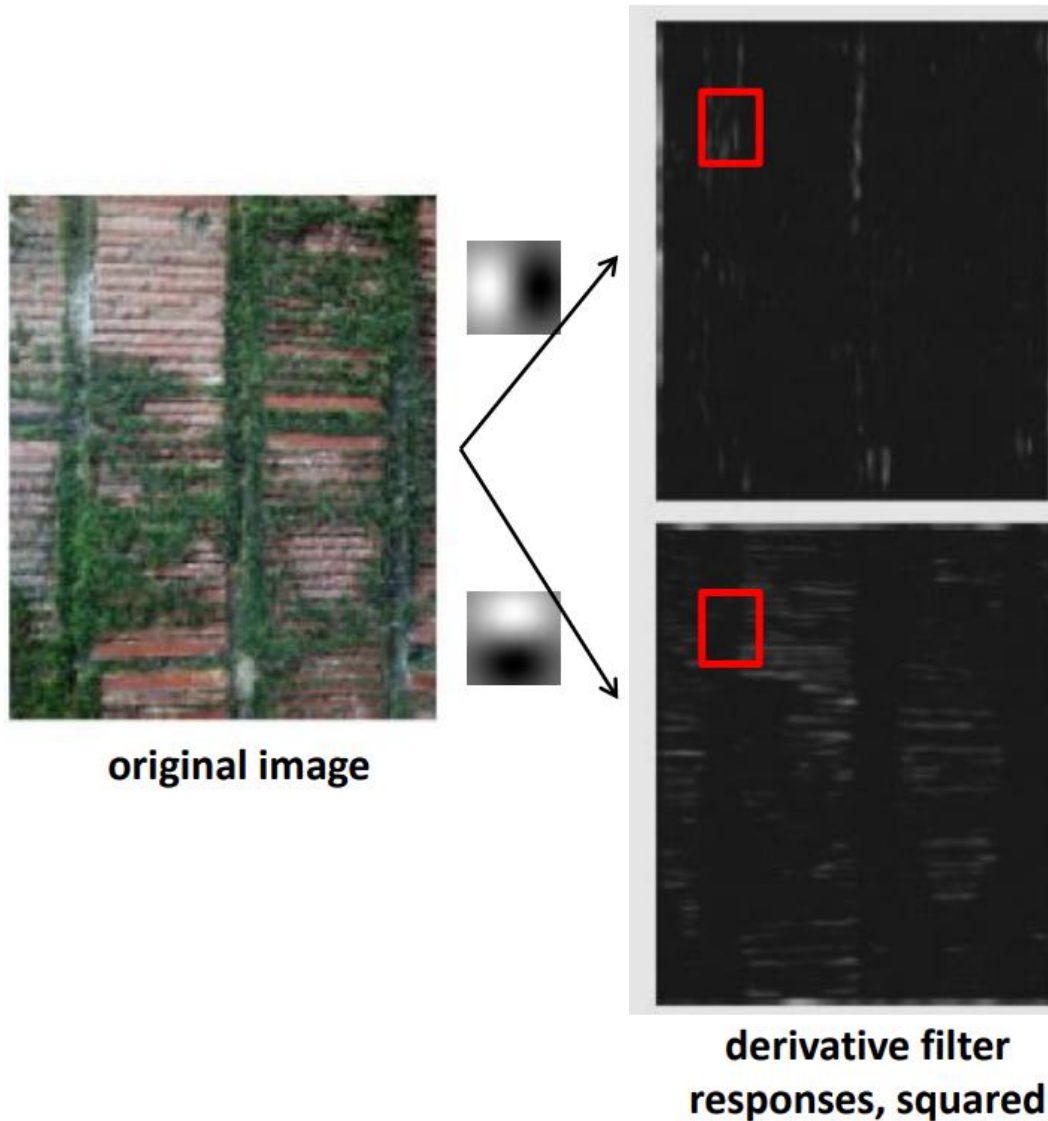
	<u>Mean d/dx value</u>	<u>Mean d/dy value</u>
Win. #1	4	10

•
•
•

statistics to
summarize patterns
in small windows



Texture Representation



	<u>Mean d/dx value</u>	<u>Mean d/dy value</u>
Win. #1	4	10
Win. #2	18	7

•
•
•

statistics to
summarize patterns
in small windows



Texture Representation



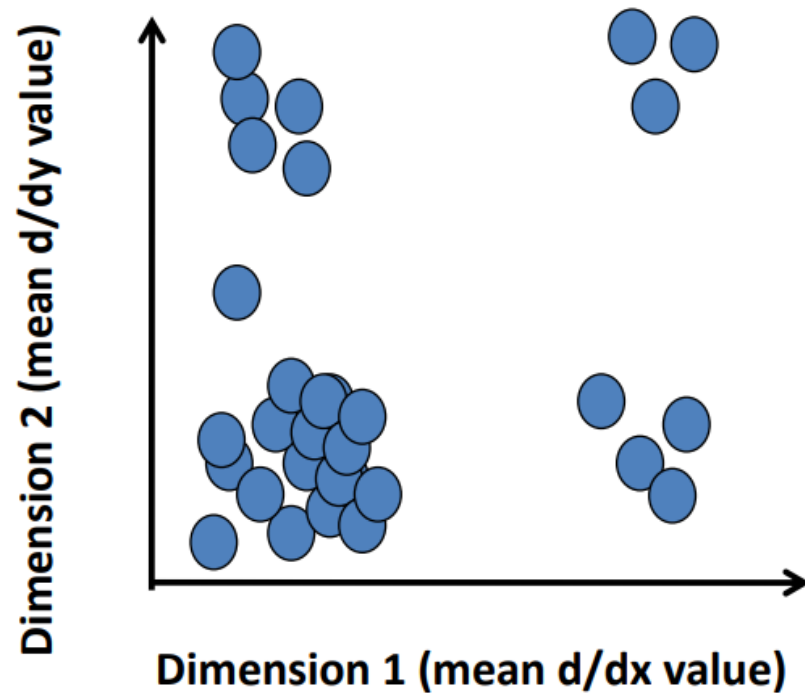
	<u>Mean d/dx value</u>	<u>Mean d/dy value</u>
Win. #1	4	10
Win. #2	18	7
⋮	⋮	⋮
Win. #9	20	20

⋮

statistics to
summarize patterns
in small windows



Texture Representation



	<u>Mean d/dx value</u>	<u>Mean d/dy value</u>
Win. #1	4	10
Win. #2	18	7
⋮	⋮	⋮
Win. #9	20	20

⋮

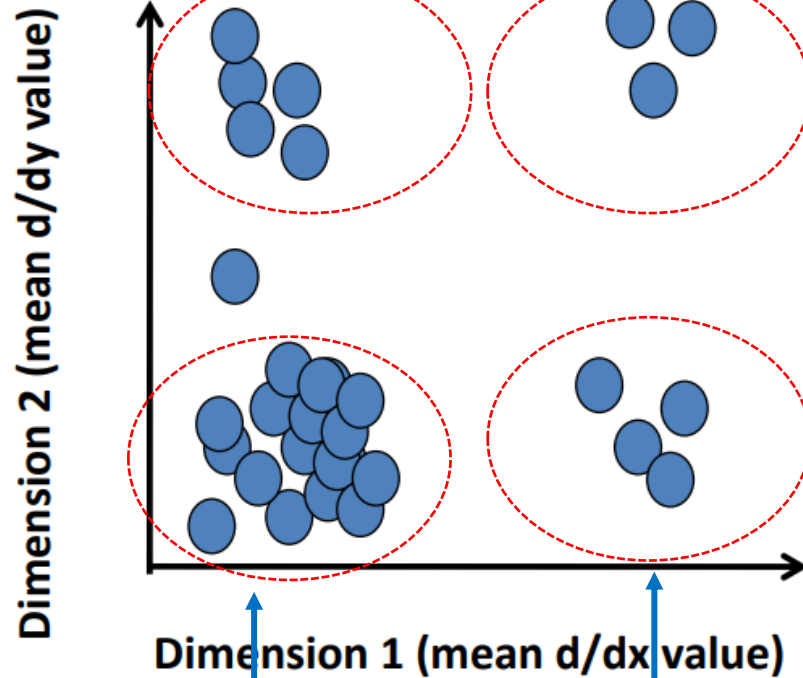
statistics to
summarize patterns
in small windows



Texture Representation

Windows with primarily horizontal edges

Both



Windows with small gradients in both directions

Windows with primarily vertical edges

	<u>Mean d/dx value</u>	<u>Mean d/dy value</u>
Win. #1	4	10
Win. #2	18	7
⋮	⋮	⋮
Win. #9	20	20

⋮

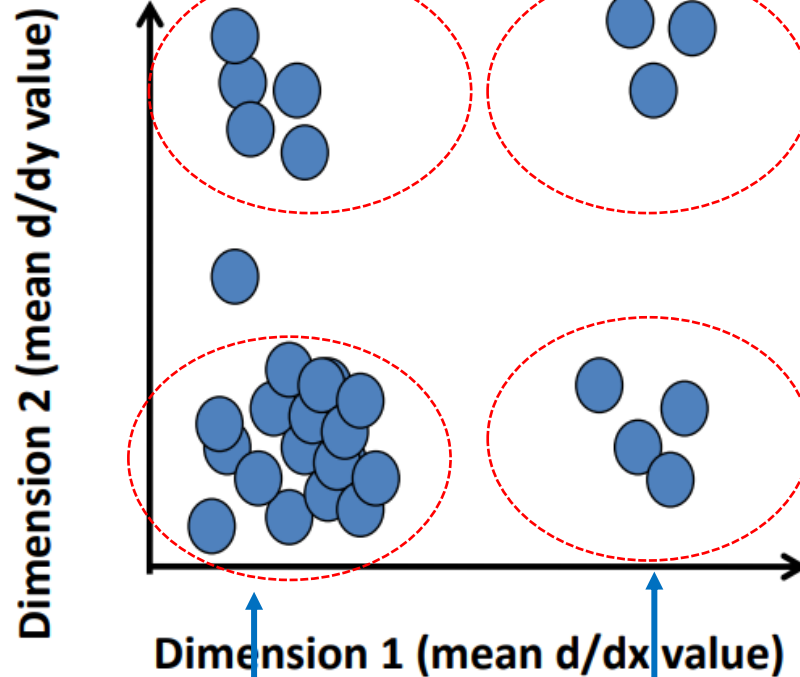
statistics to summarize patterns in small windows



Texture Representation

Windows with primarily horizontal edges

Both



Windows with small gradients in both directions

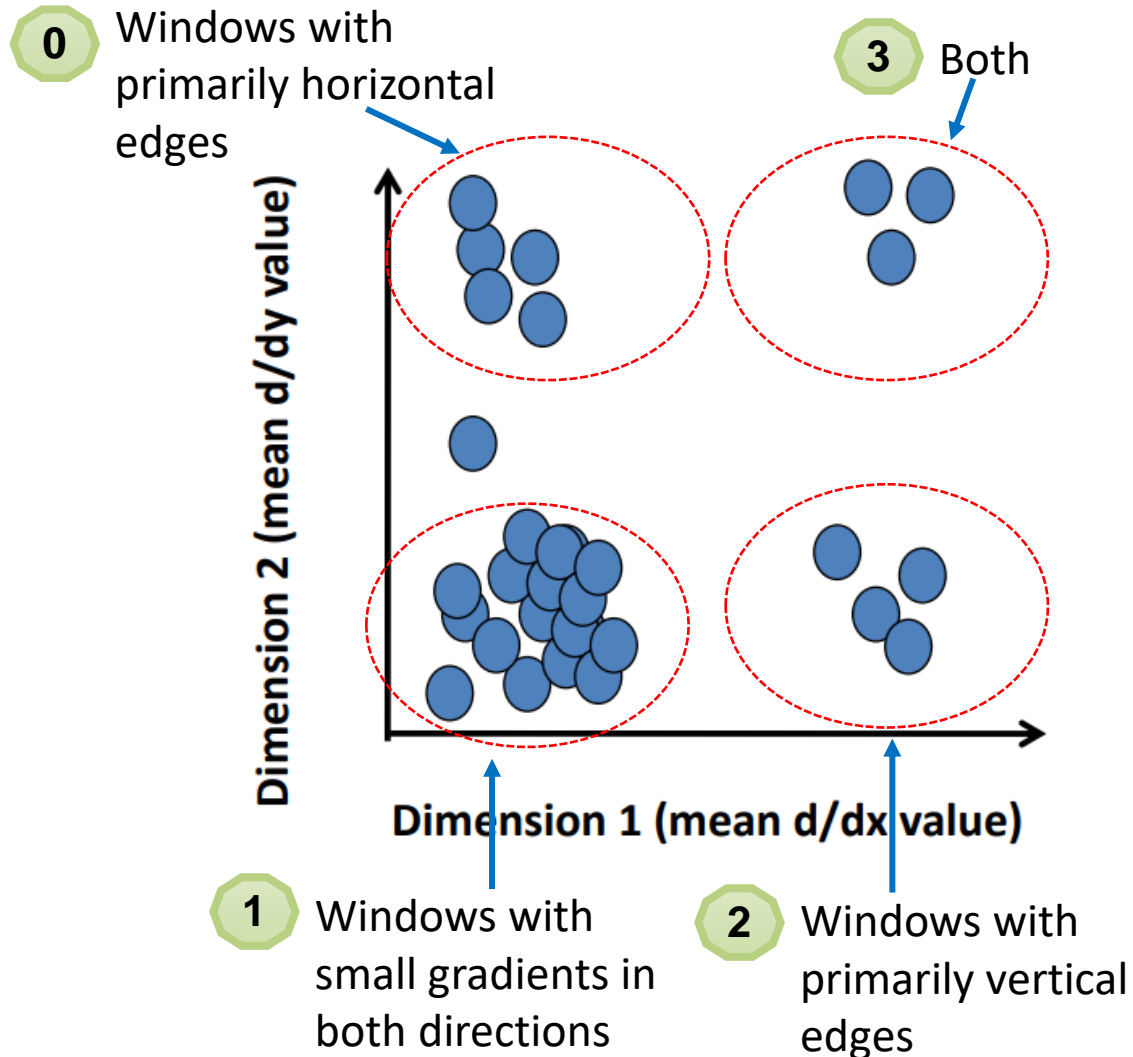
Windows with primarily vertical edges

There are 4 types of textures defined here

Think of a way how we can represent each image pixel with only these 4 texture features...



Texture Representation



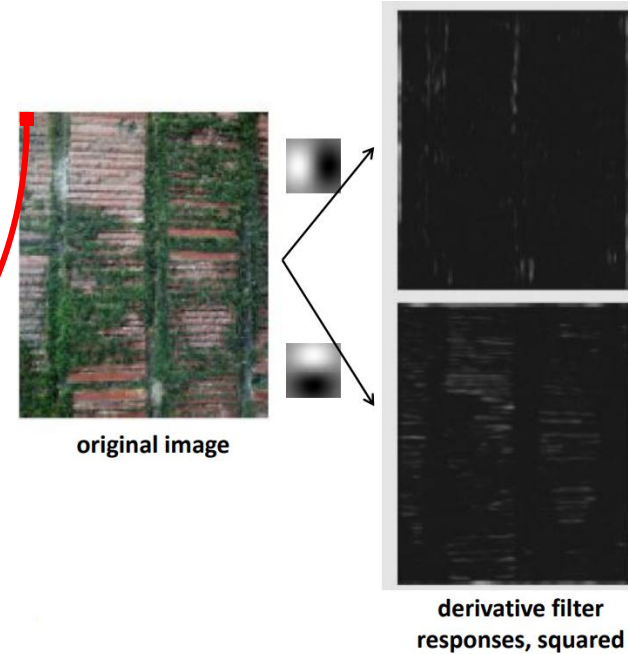
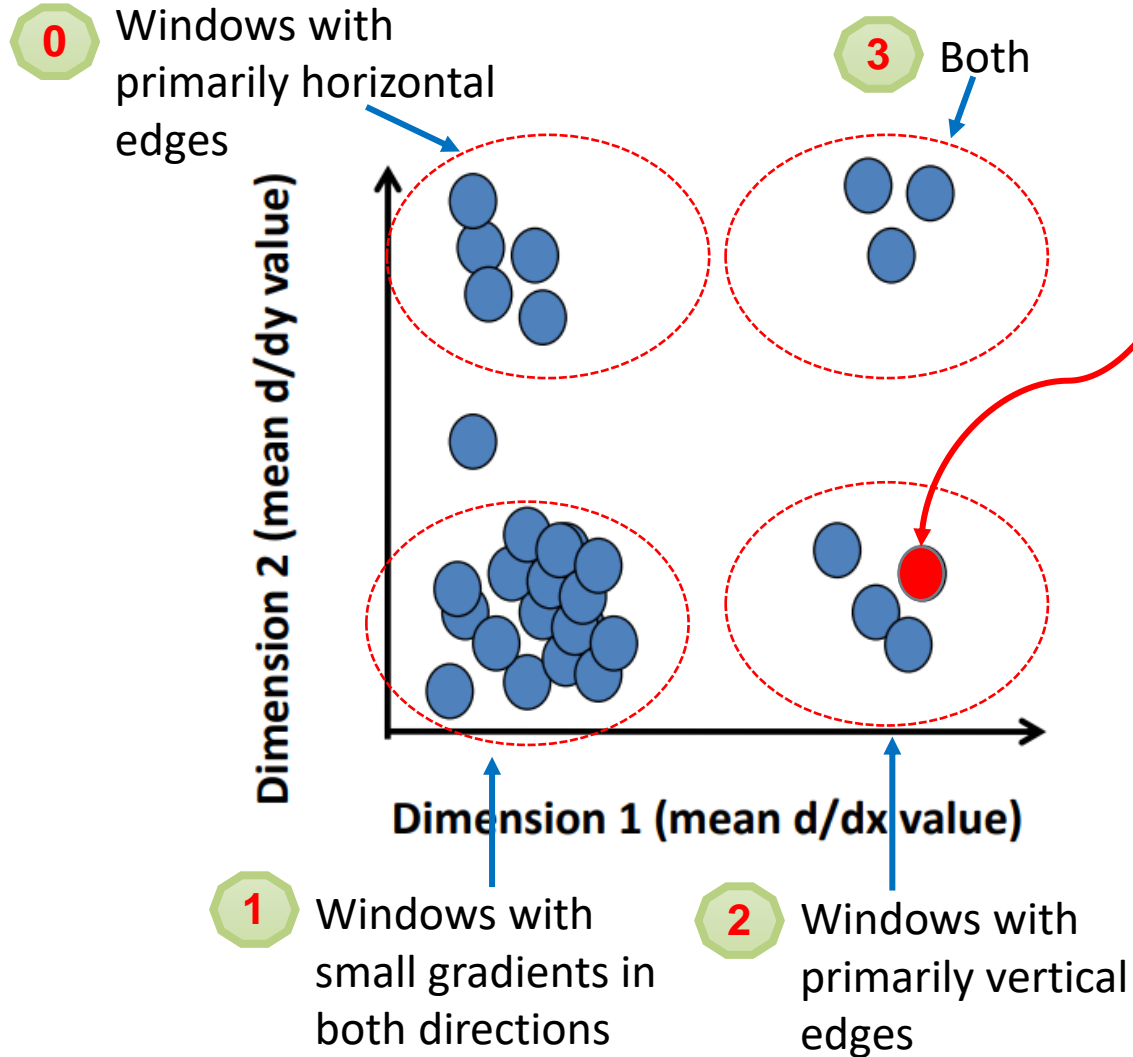
Method #1
Represent each pixel

Label each type of texture
(e.g. 0, 1, 2, 3)

Assign the feature values
(d/dx and d/dy in this case)
to the “nearest” group. Do
that for all pixels.



Texture Representation



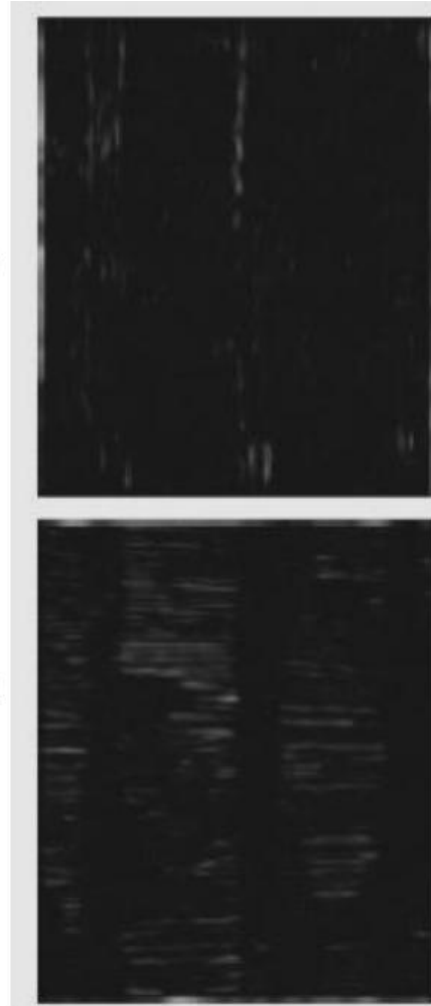
Example:
Pixel marked in red is located in group 2
⇒ Assigned with texture “value” of 2



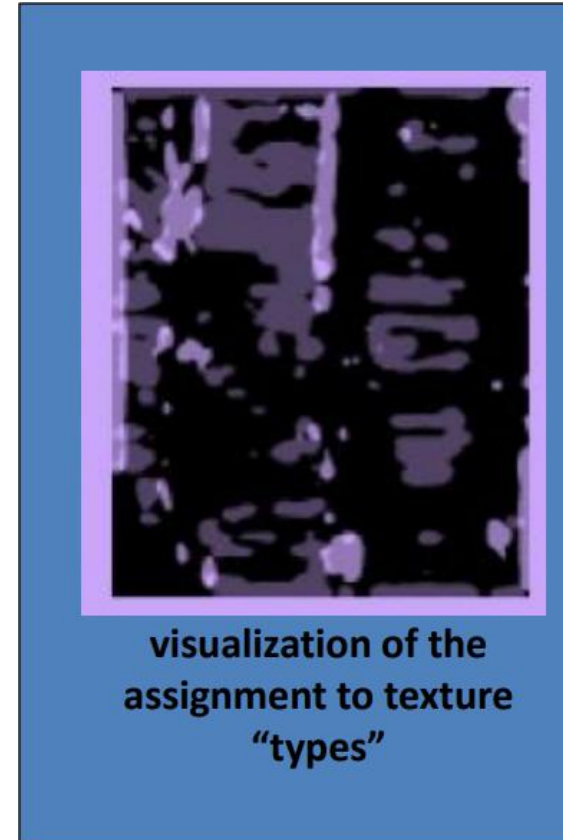
Texture Representation



original image



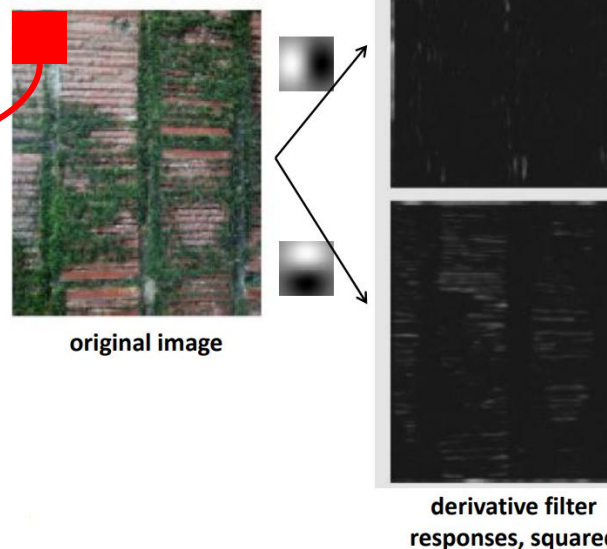
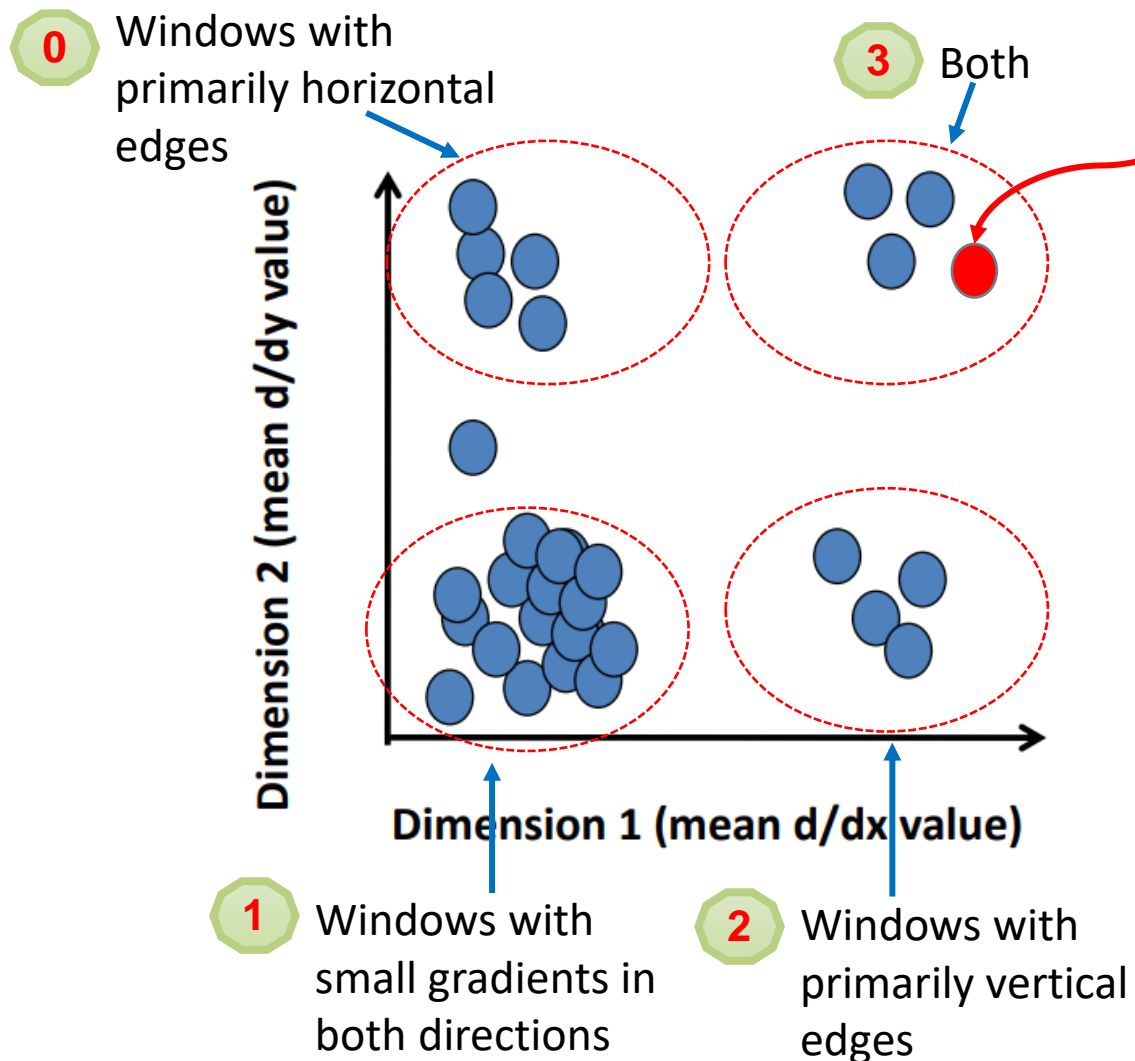
derivative filter
responses, squared



visualization of the
assignment to texture
"types"



Texture Representation



Method #2
Represent each window or “patch”

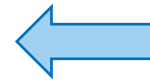
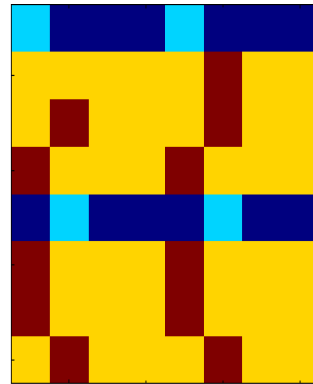
Example:
Mean gradients of the window marked in red is located in group 3
⇒ Assigned with texture “value” of 3



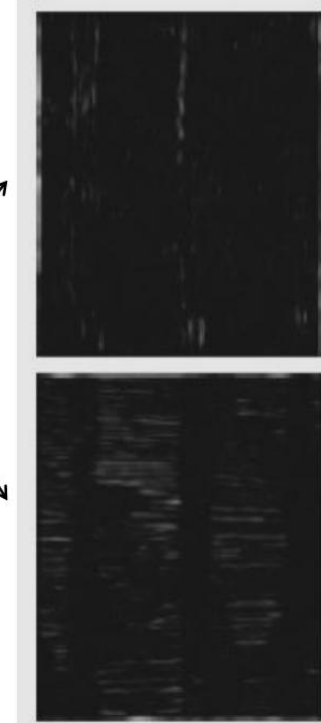
Texture Representation

Each window is now labelled with one of the four group labels

This is a better way to represent textures than using each pixel



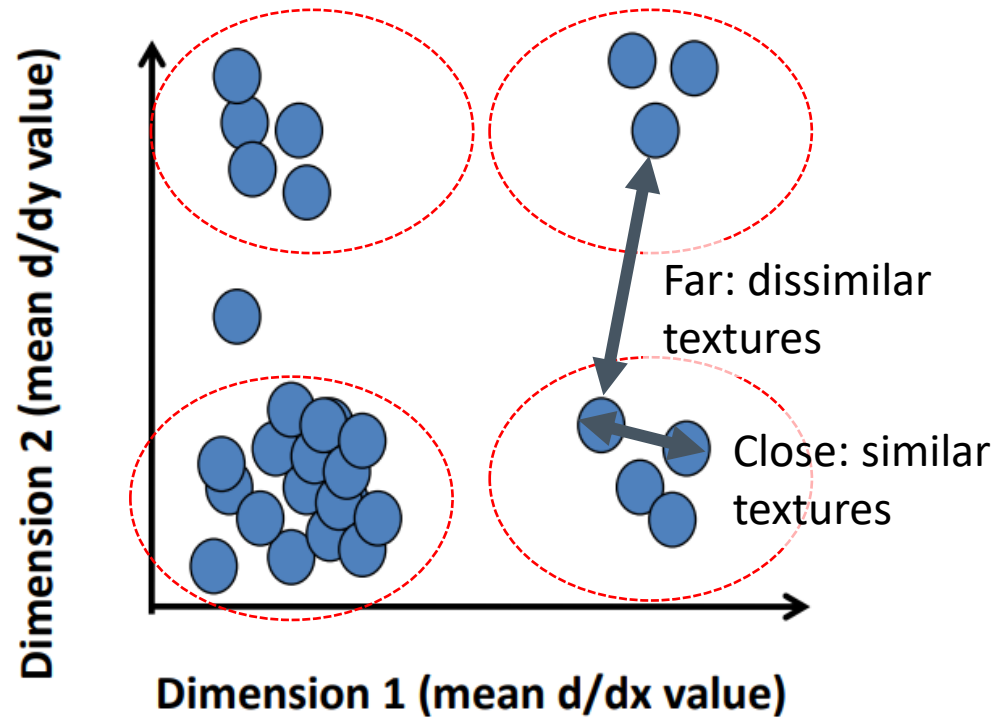
original image



derivative filter responses, squared



Texture Representation



	<u>Mean d/dx value</u>	<u>Mean d/dy value</u>
Win. #1	4	10
Win. #2	18	7
⋮	⋮	⋮
Win. #9	20	20

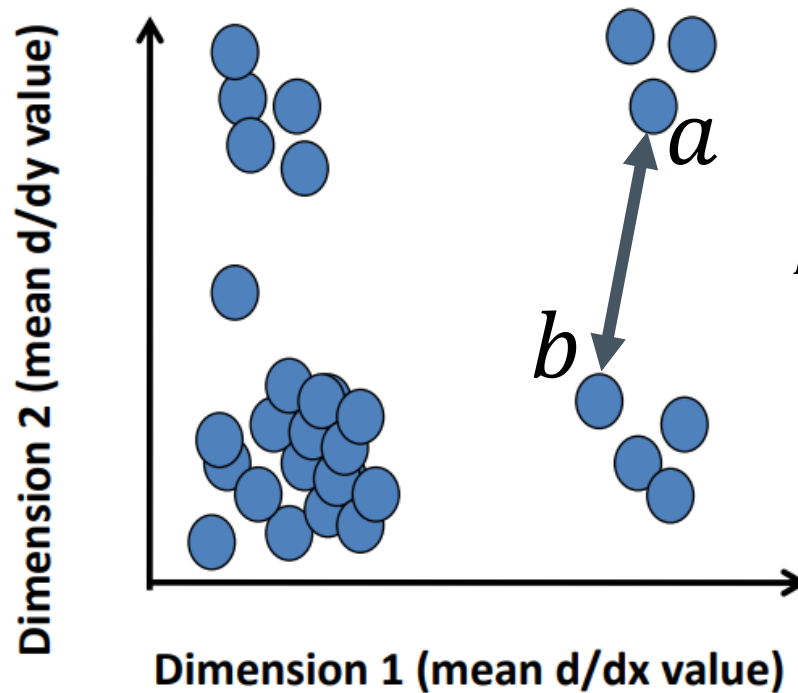
⋮

statistics to
summarize patterns
in small windows



Differentiating Textures

› How can we differentiate between texture windows?



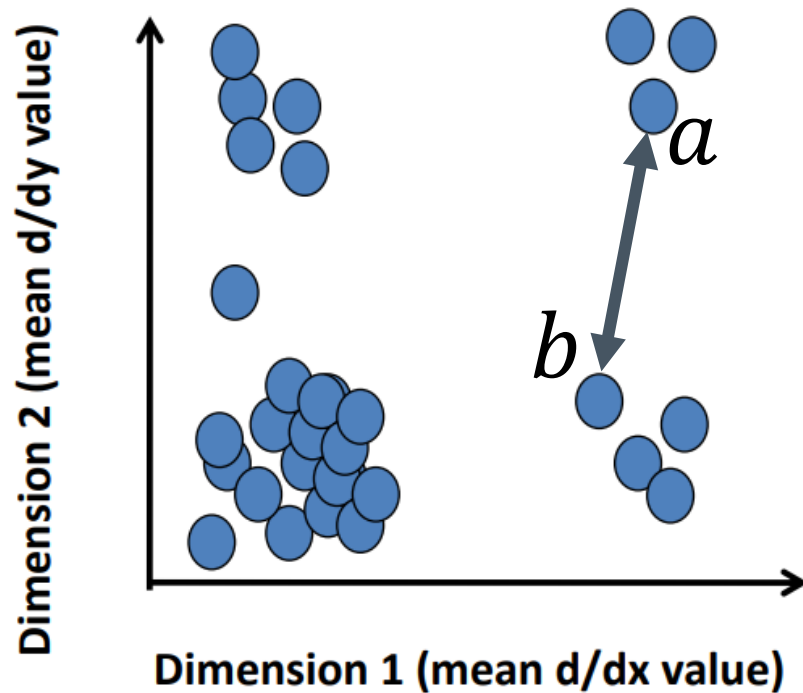
$$D(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Since there are only 2 dimensions, it is easy to determine the distance between the textures represented in two windows (a and b in example)

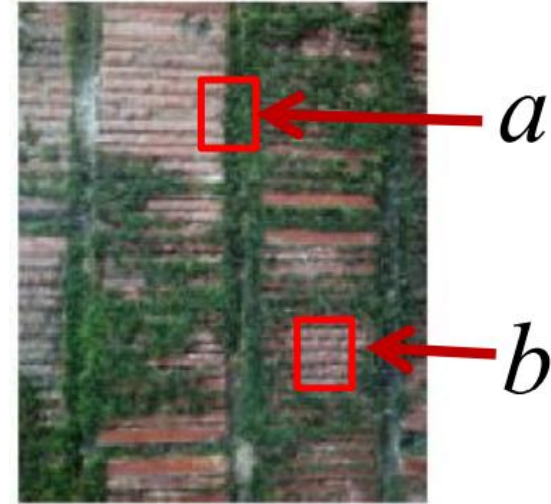


Differentiating Textures

› How can we differentiate between texture windows?



Distance reveals how dissimilar texture from window a is with texture from window b .

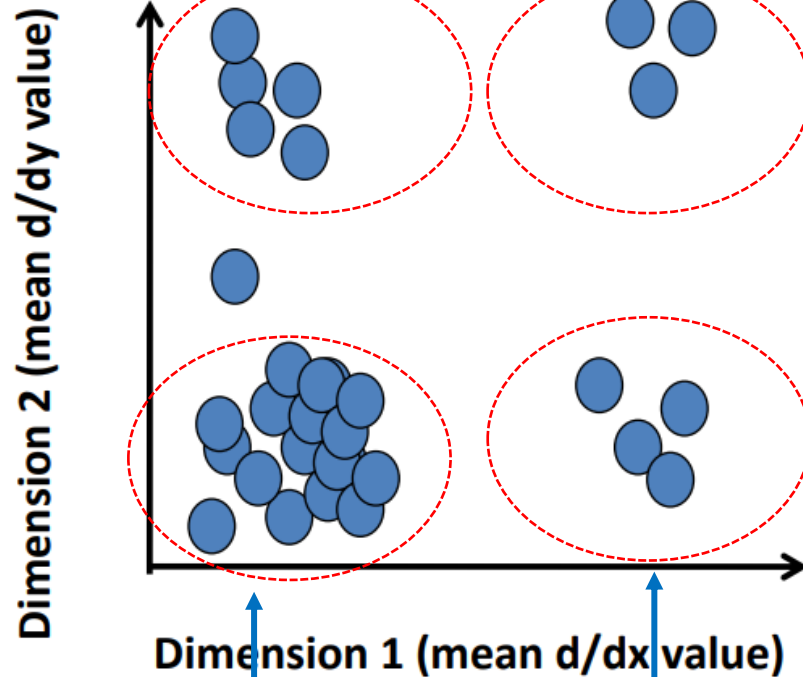




Issue #1: Texture information

Windows with primarily horizontal edges

Both



Are 2 values (dimension = 2) sufficient to capture the texture information?

Solution: Use more information, more!

Windows with small gradients in both directions

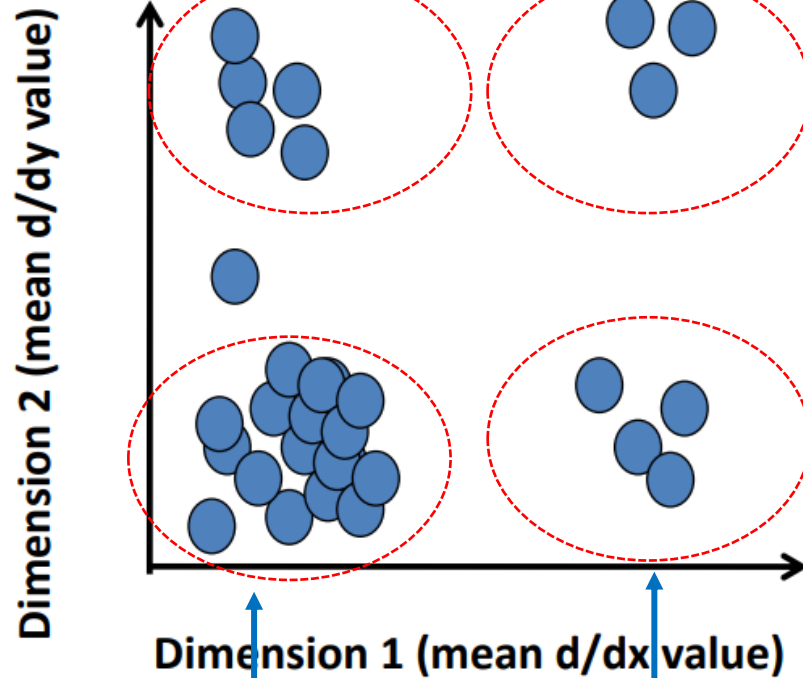
Windows with primarily vertical edges



Issue #2: Texture groupings

Windows with primarily horizontal edges

Both



Windows with small gradients in both directions

Windows with primarily vertical edges

How to know the “groups” of textures and then assign them?

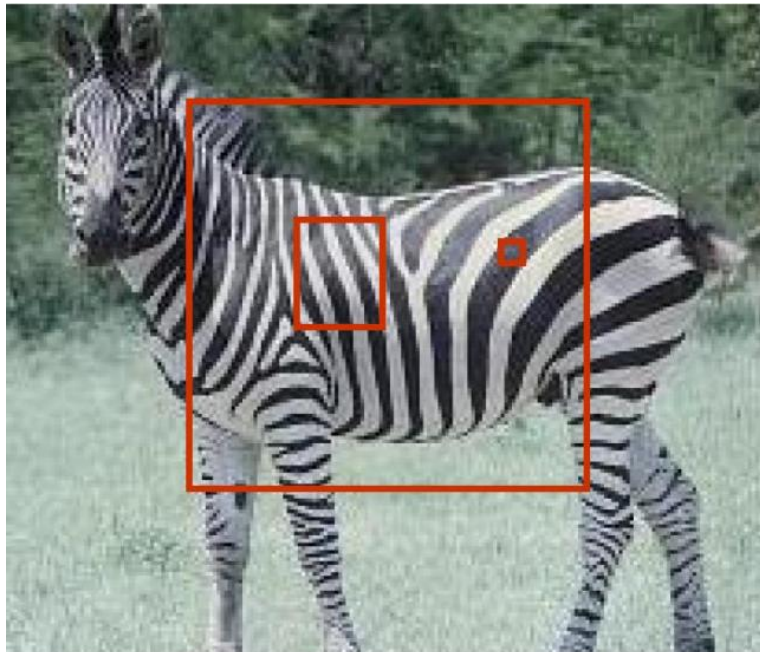
Solution: Clustering!



Other Issues

› Window scale

- We are assuming we know the relevant window size to obtain these statistics



Possible to perform scale selection by looking for window scale where texture description is not changing

Resolving Issue #1

Are **TWO** types of filters sufficient for describing texture in an image?



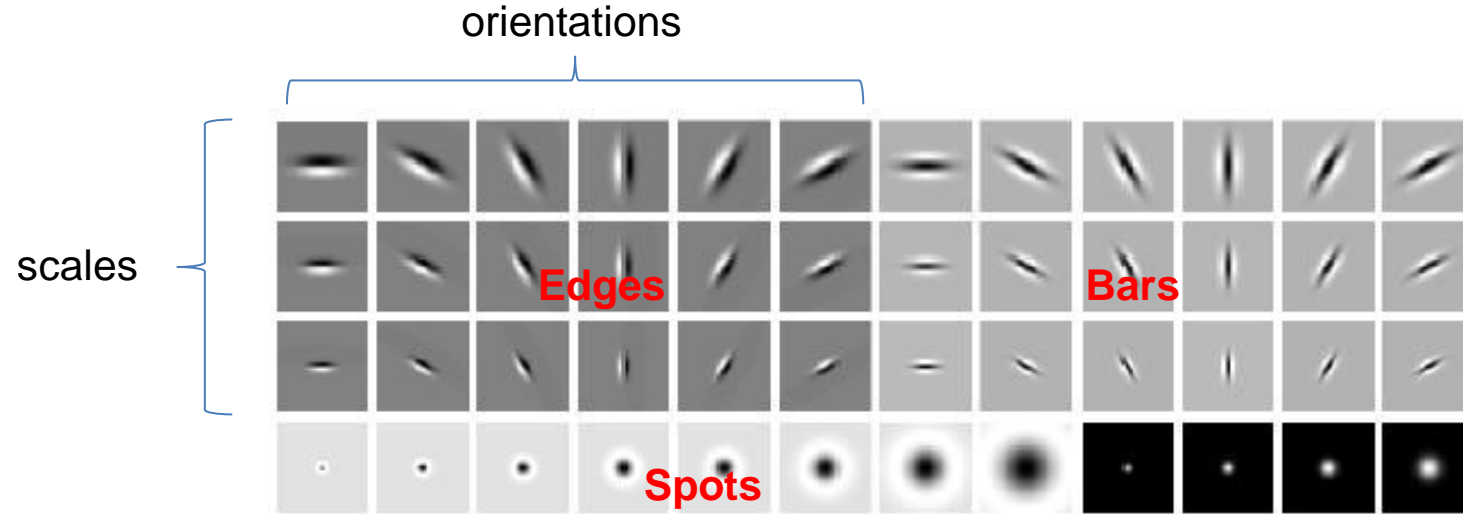


Filter bank

- › Our previous example used only 2 filters
 - x and y derivatives revealed some information about local structure
- › We can generalize to apply a **collection of multiple (d) filters** – also known as a “**filter bank**”
 - Feature vectors will be *d -dimensional*
 - Same way to calculate distance/dissimilarity



Filter bank



> What filters to put in the bank?

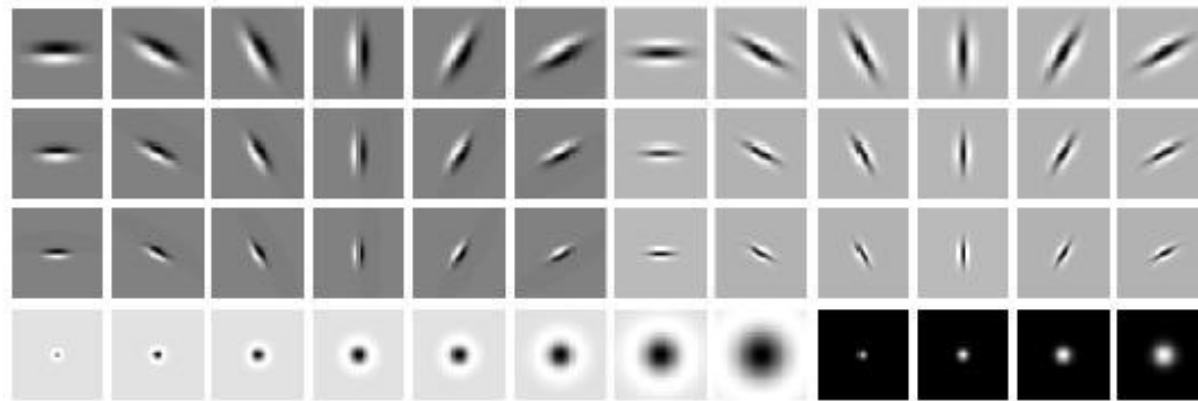
- Typically, we want a combination of scales and orientations, and different types of patterns

Matlab code: <http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html>

Python code: http://www.rgislib.org/rgislib_imagefilter.html#filter-banks



Leung-Malik (LM) filter bank



> 48 filters:

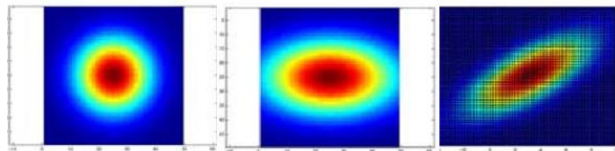
- 1st and 2nd derivative of Gaussians at 6 orientations and 3 scales (total 36)
- 8 LoG filters, 4 Gaussian filters (total 12)



Filter bank: Example



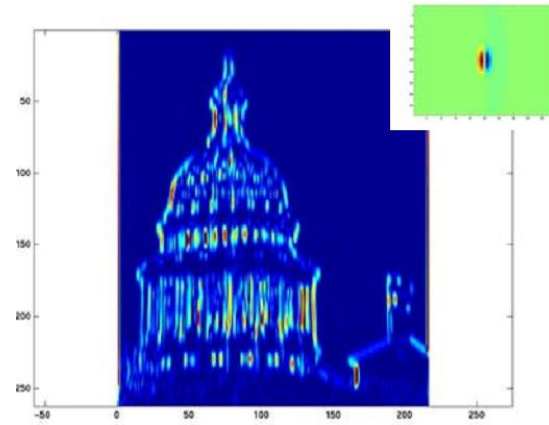
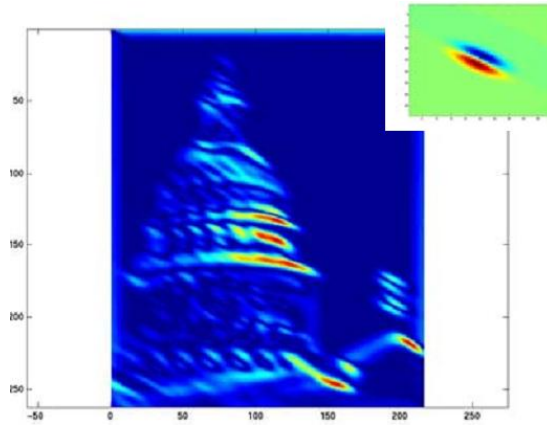
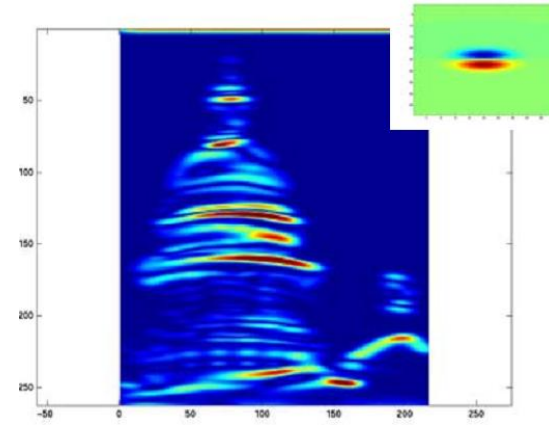
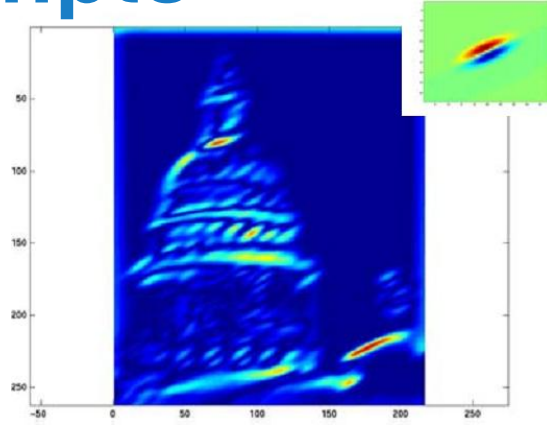
$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$



$$\Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix}$$

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

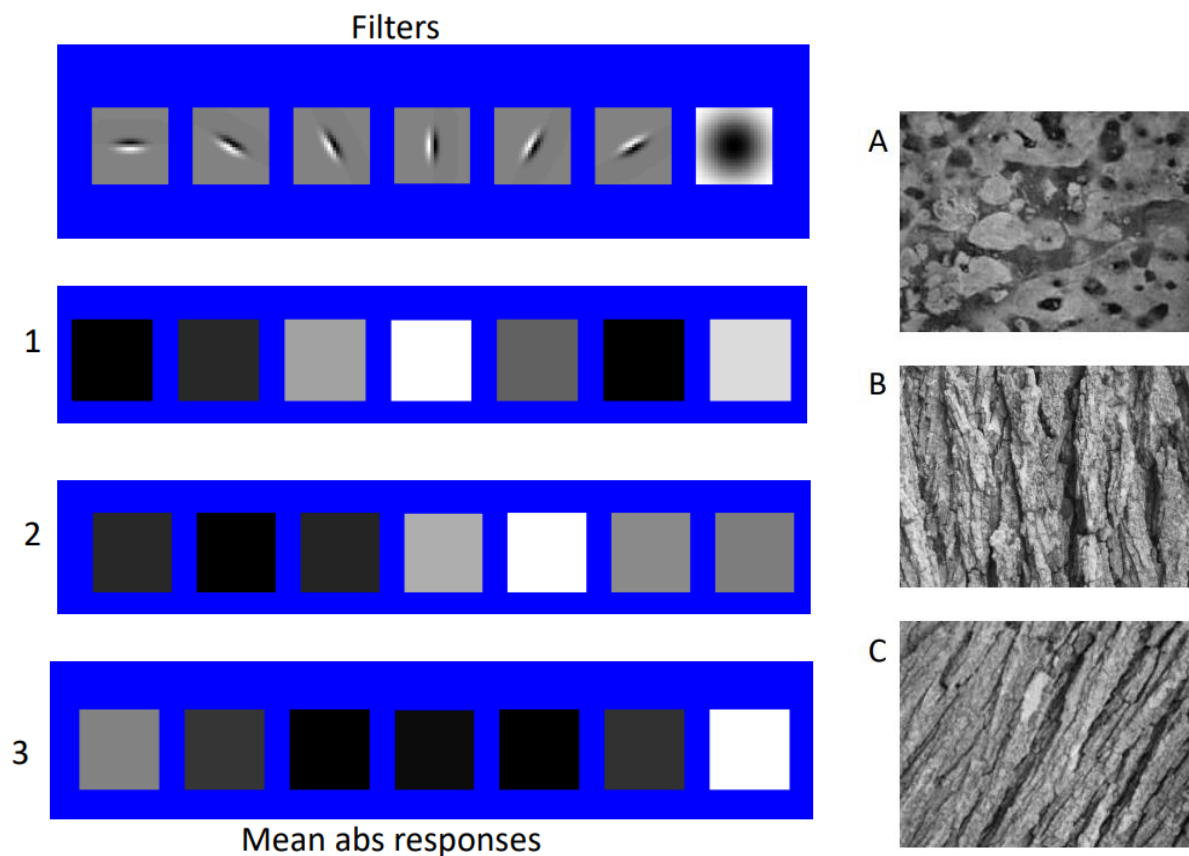
$$\Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$





Filter bank responses

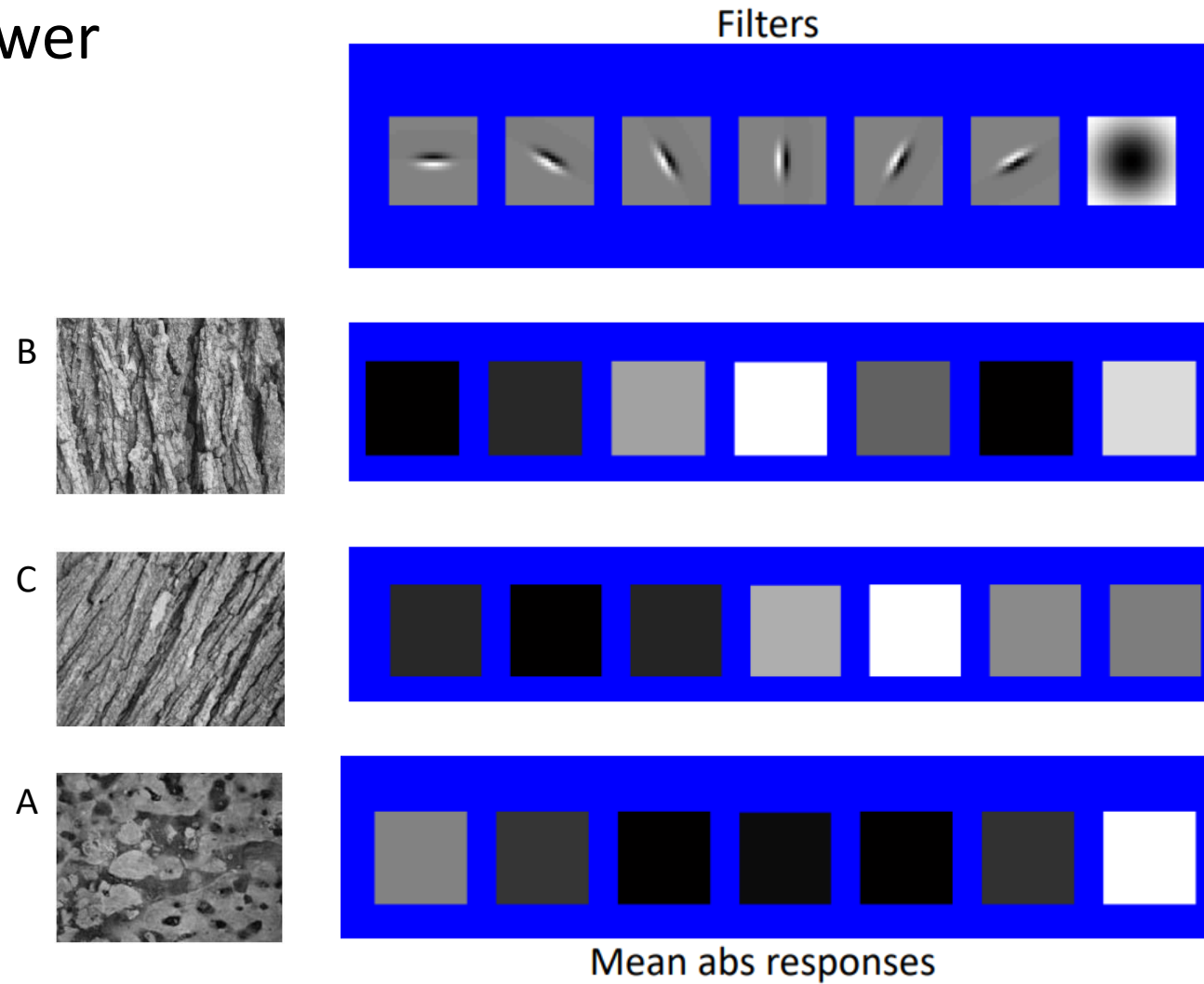
› Try: Match the texture to the response:





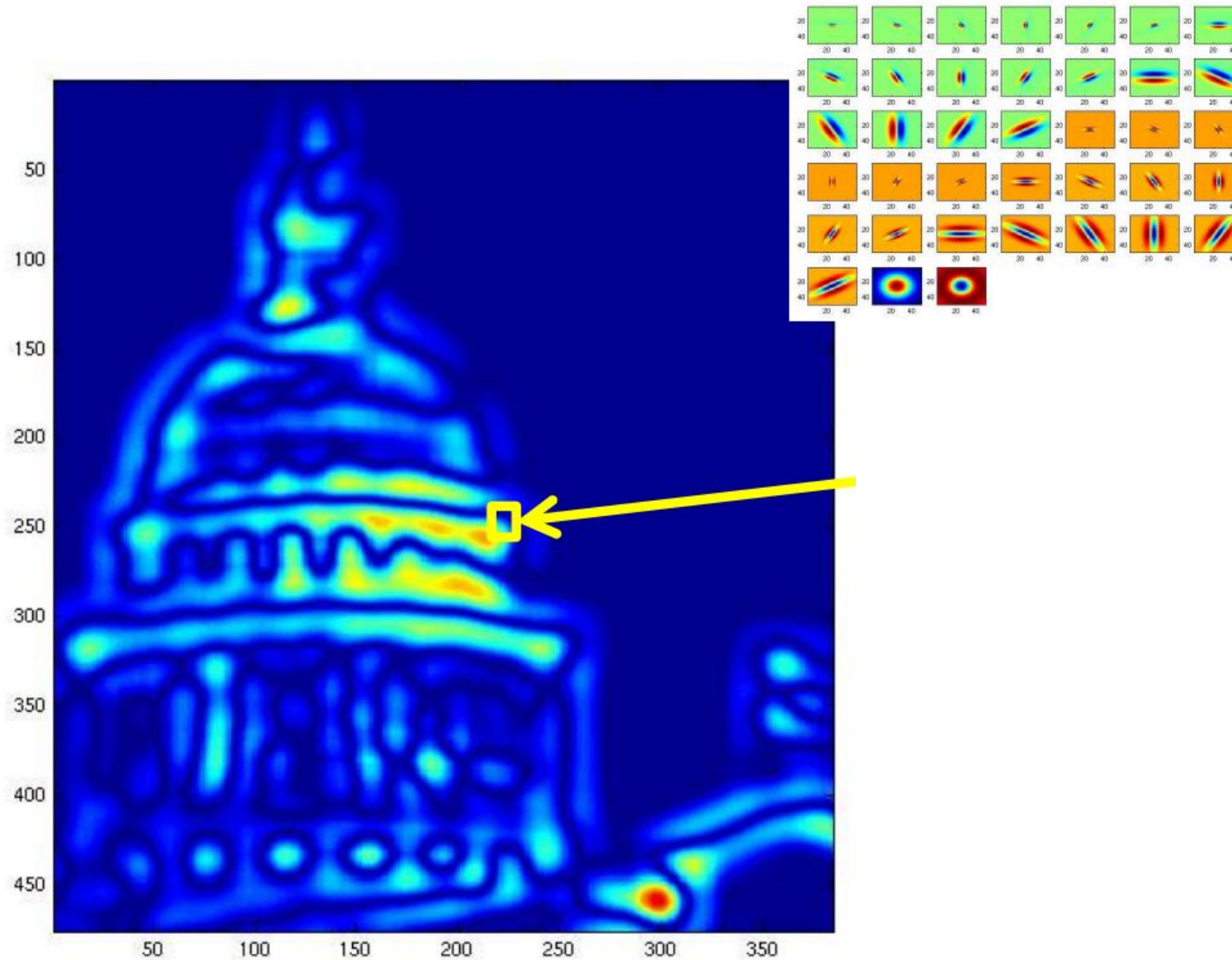
Filter bank responses

- Answer





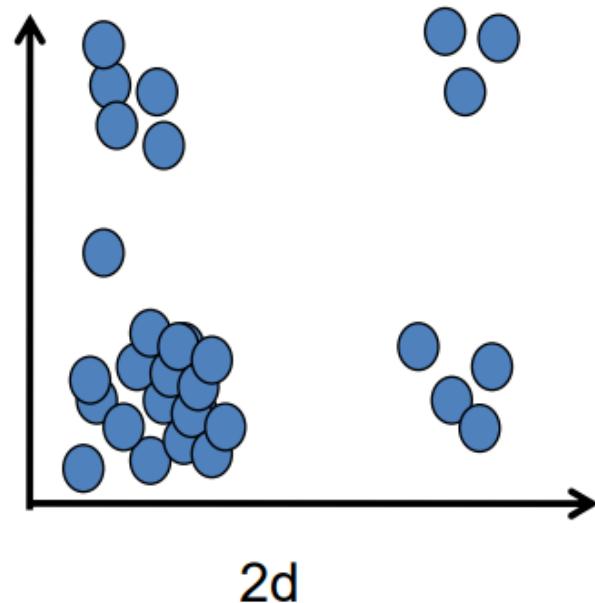
Filter bank responses \Rightarrow Feature





d -dimensional features

- › Previously: 2 filters \Rightarrow 2-dimensional feature vector
- › Now: 48 filters \Rightarrow 48-dimensional feature vectors
- › Distance can be measured with L2-distance or Euclidean distance



$$D(a, b) = \sqrt{\sum_{i=1}^d (a_i - b_i)^2}$$

Euclidean distance (L_2)

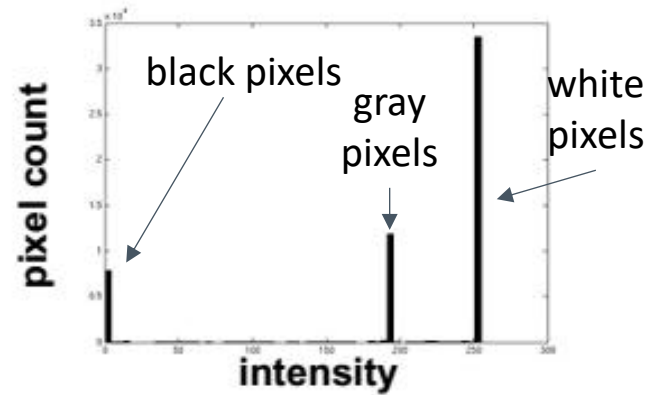
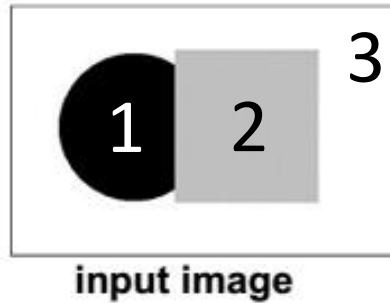
Resolving Issue #2

How to group these texture information into “groups”?





Motivation



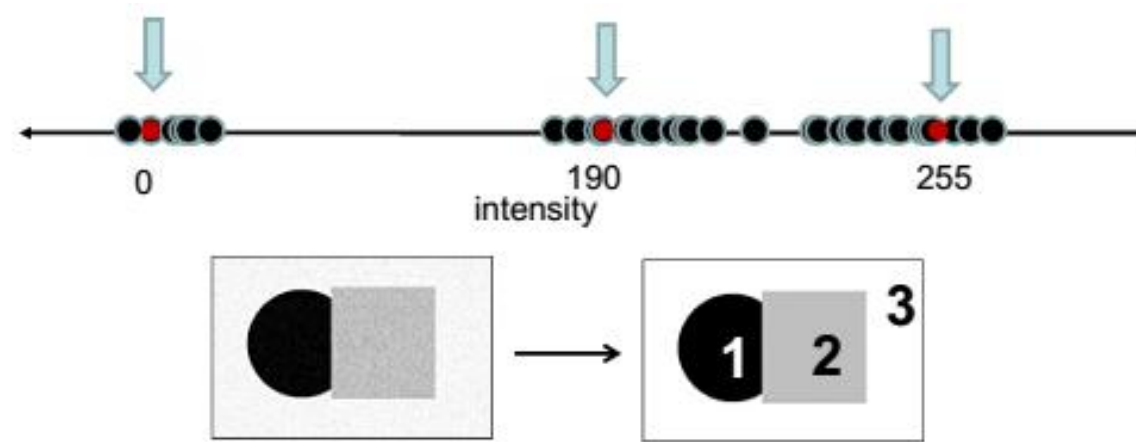
Threshold to 3 regions.
Easy.

Now, how to determine the “three” main intensities that define the groups seen in the image?

One way is to **cluster**.



Finding Clusters

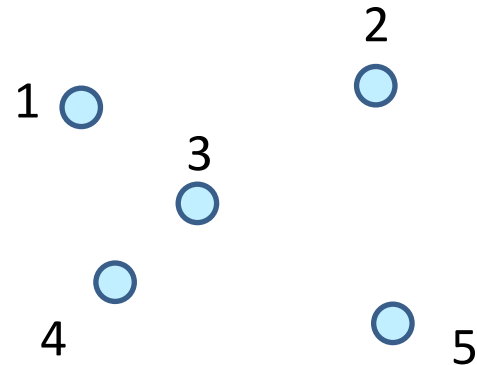
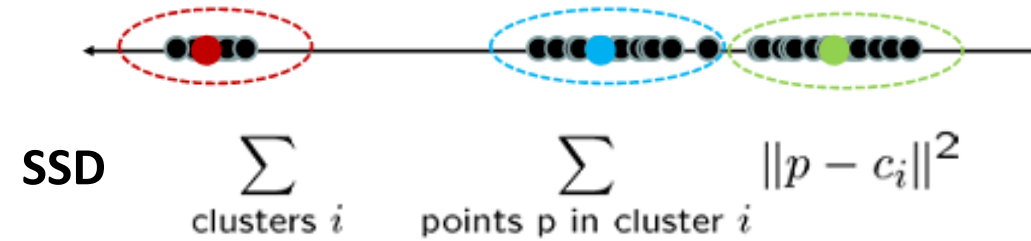


- › **Aim:** Choose **three “centres”** as the representative intensities, and label every pixel according to which of these centres it is nearest to
- › Best cluster centres are those that minimize sum of squared distances (SSD) between all points and their nearest cluster centre

$$\sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2$$



The intuition behind SSD



Look at this cluster of points.
Which of these points will be the **NEAREST** to all other points?

How would you find the answer?



How to solve this?

> “Chicken and egg” problem.



- **If we knew the cluster centres**, we could allocate points to groups by assigning each to its closest centre
- **If we knew the group memberships**, we could get the centres by computing the mean per group



K-means clustering

› **Goal:** Cluster to minimize variance in data given clusters

$$c^*, \delta^* = \arg \min_{c, \delta} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} (c_i - x_j)^2$$

Cluster center Data

Whether x_j is assigned to c_i



K-means clustering

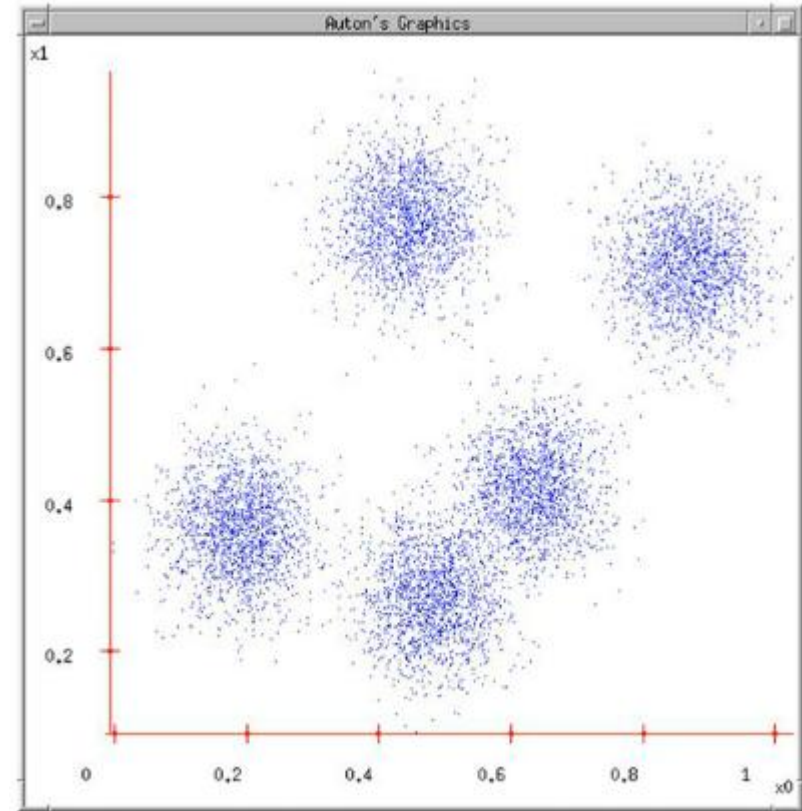
- › Basic idea: Randomly initialize the k cluster centres, and iterate between the two steps we just saw.
 1. Randomly initialize cluster centres c_1, \dots, c_k
 2. Given cluster centres, determine points in each cluster – for each point p , find closest c_i , put p into cluster I
 3. Given points in each cluster, solve for c_i – set c_i to the mean of points in cluster I
 4. If c_i have changed, repeat Step 2. Otherwise, terminate.
- › Properties:
 - Will always converge to *some* solution
 - Does not always find the global minimum of objective function

$$\sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2$$



K-means clustering

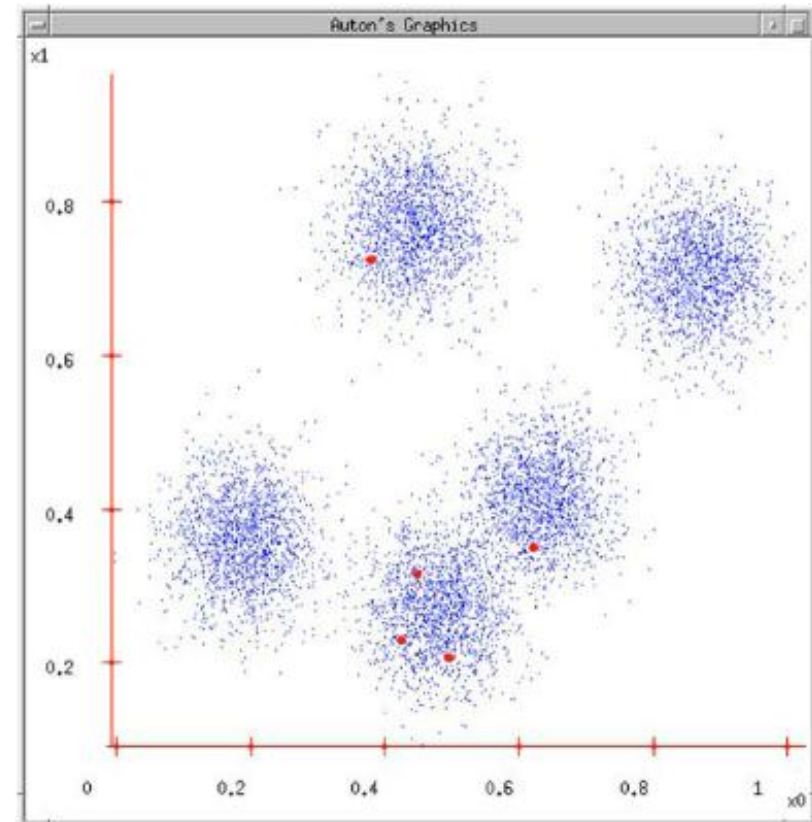
1. Determine beforehand how many clusters or value of k





K-means clustering

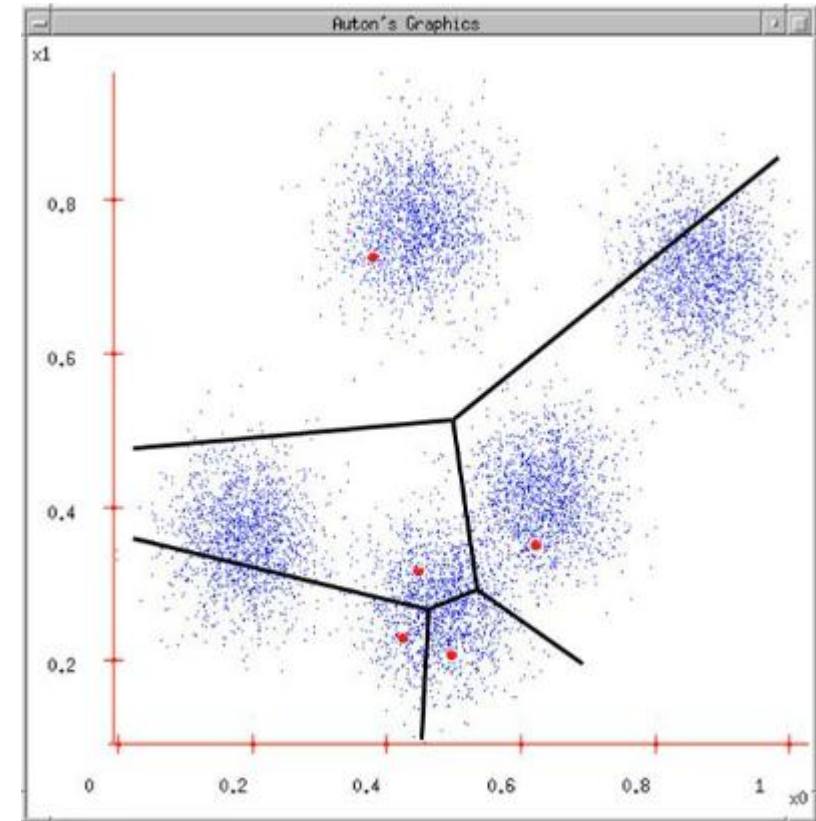
1. Determine beforehand how many clusters or value of k
2. Randomly guess k cluster centre locations





K-means clustering

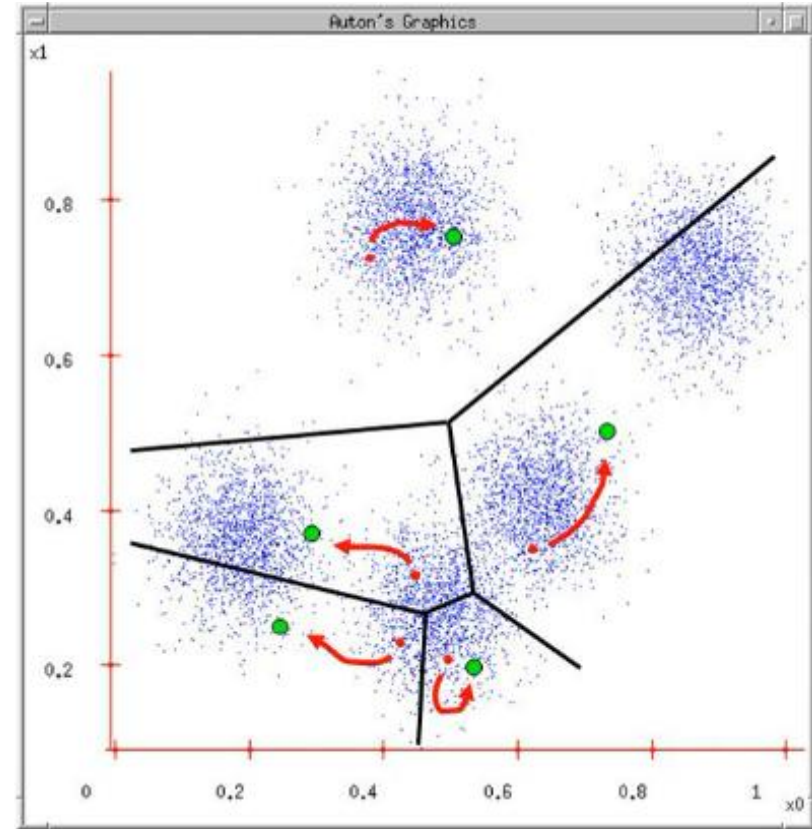
1. Determine beforehand how many clusters or value of k
2. Randomly guess k cluster centre locations
3. Each data point finds out which centre it is closest to (each centre “owns” a set of points)





K-means clustering

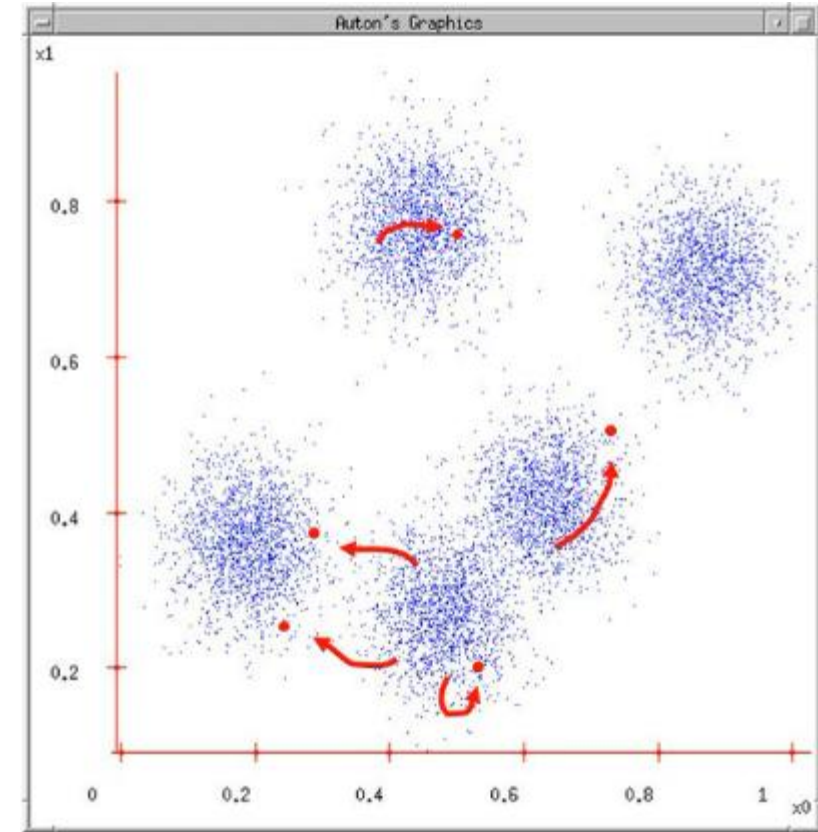
1. Determine beforehand how many clusters or value of k
2. Randomly guess k cluster centre locations
3. Each data point finds out which centre it is closest to (each centre “owns” a set of points)
4. Each centre finds the centroid of its own group





K-means clustering

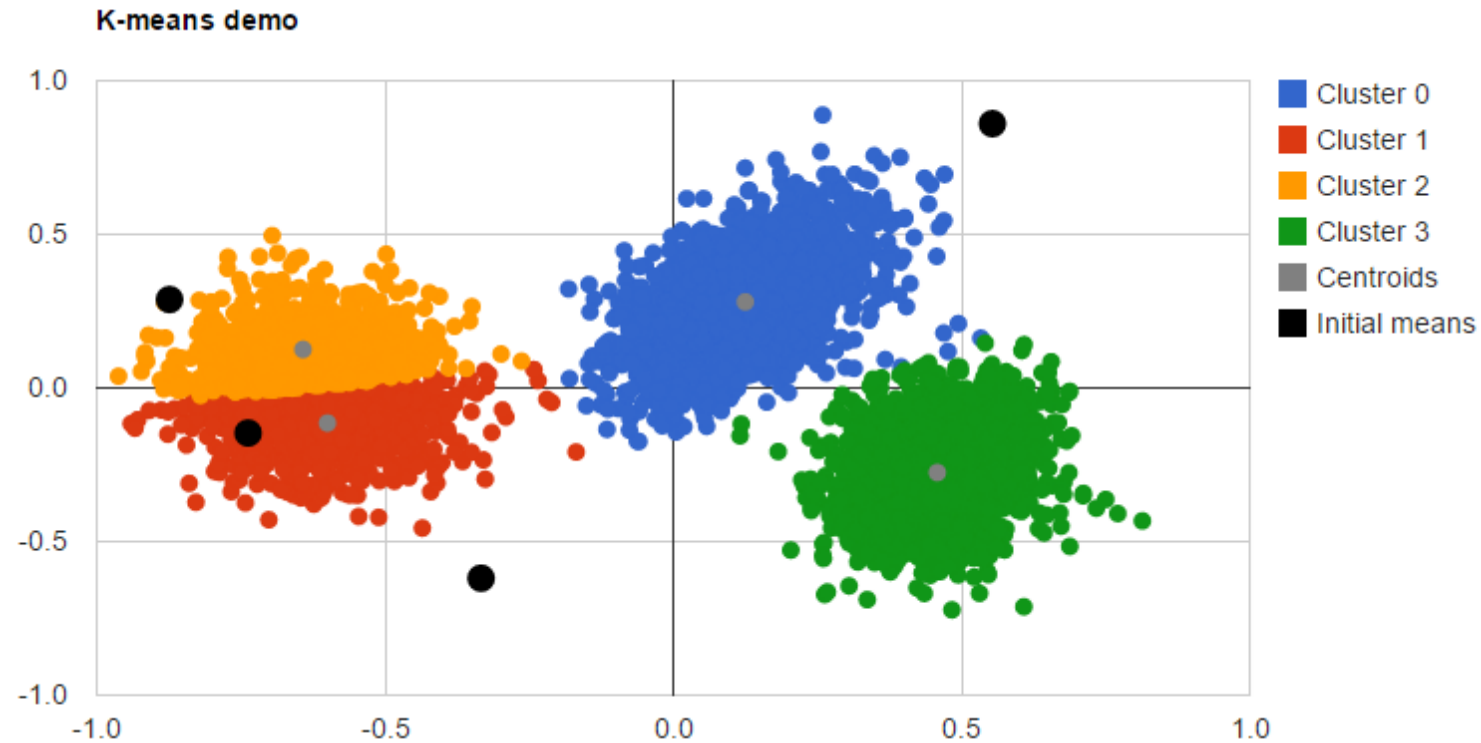
1. Determine beforehand how many clusters or value of k
2. Randomly guess k cluster centre locations
3. Each data point finds out which centre it is closest to (each centre “owns” a set of points)
4. Each centre finds the centroid of its own group
5. With the new centroid, repeat again the process from (3) until algorithm terminates





K-means clustering

› A nice demo: <http://syskall.com/kmeans.js/>





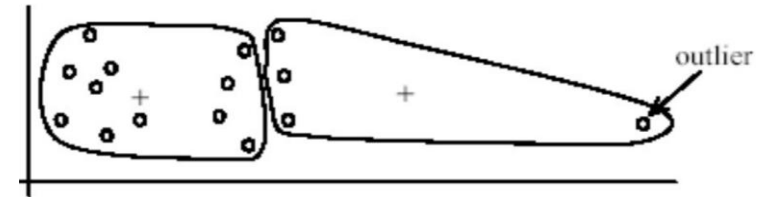
K-means: Pros and Cons

> Pros

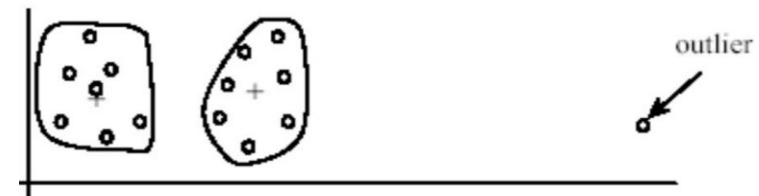
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

> Cons

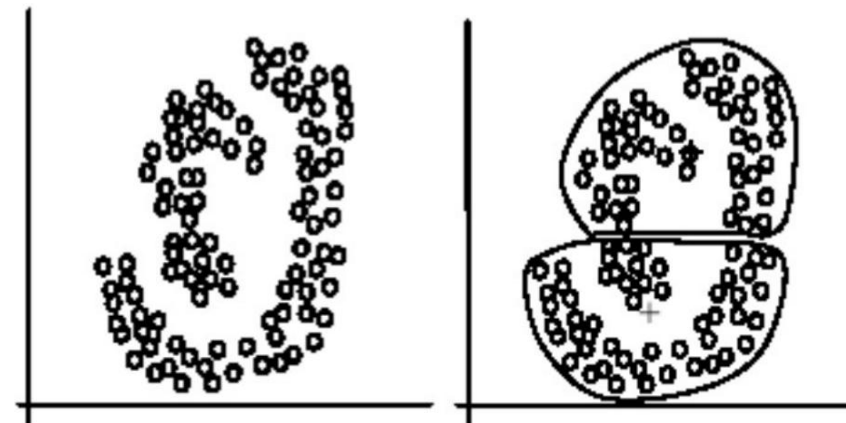
- Setting k ?
- Sensitive to initial centres
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters

(B): k -means clusters

Histograms for Texture Representation





Texture Representation Revisited

- › Textures are made up of **repeated local patterns**
 - **Find** these “patterns”
 - › Use filters that LOOK like patterns (spots, bars, raw patches, etc.)
 - › Consider magnitude of response of these patterns
 - **Describe** their **statistics** within each local window (or “neighbourhood”)
 - › Mean, standard deviation
 - › (and at a higher level..) **Histogram of feature occurrences**



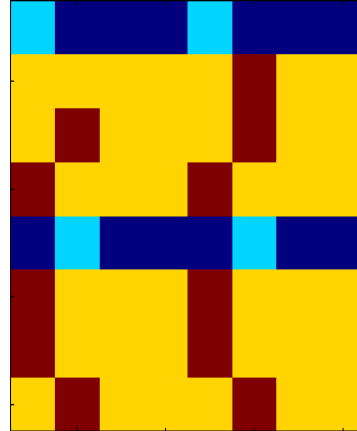
Example Revisited



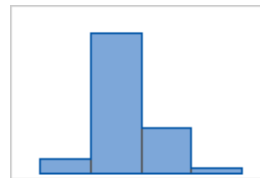
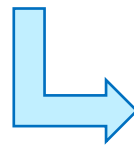
original image



visualization of the assignment to texture "types"



Instead of only assigning each pixel to texture types, build some **statistics** about it after that



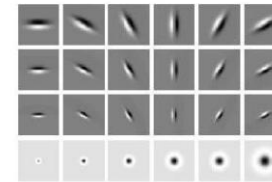
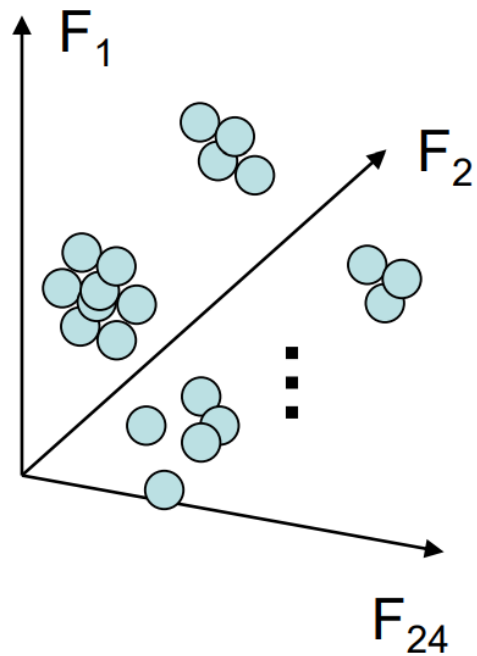
Histogram of feature occurrences

Based on the earlier e.g. a histogram of 4 bins is now the new "feature vector"
⇒ Describes the statistical distribution of textures in an image



Texture-based grouping

- › Grouping pixels based on **texture** similarity



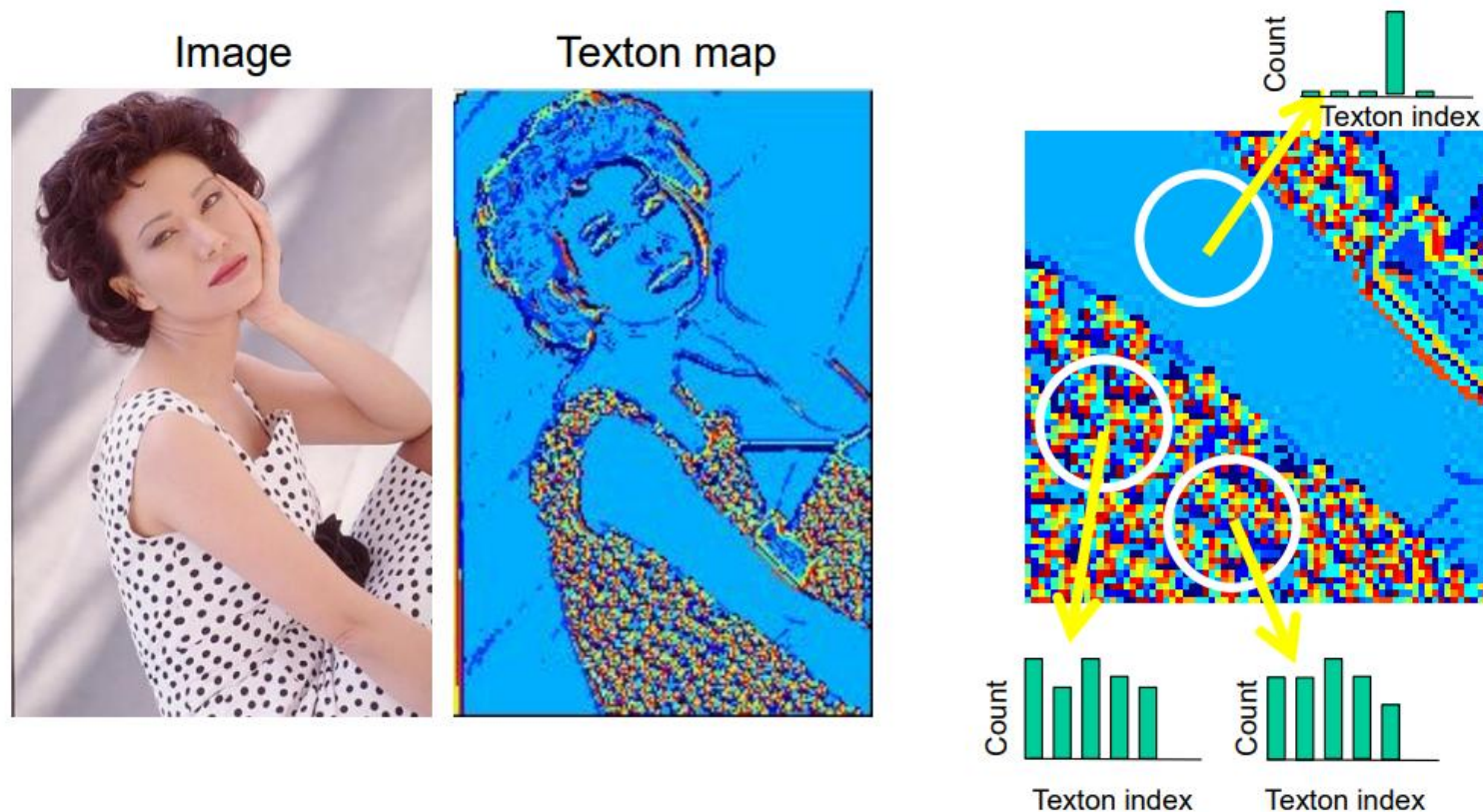
Filter bank
of 24 filters

- › Feature space: Filter bank responses (e.g. 24-D)



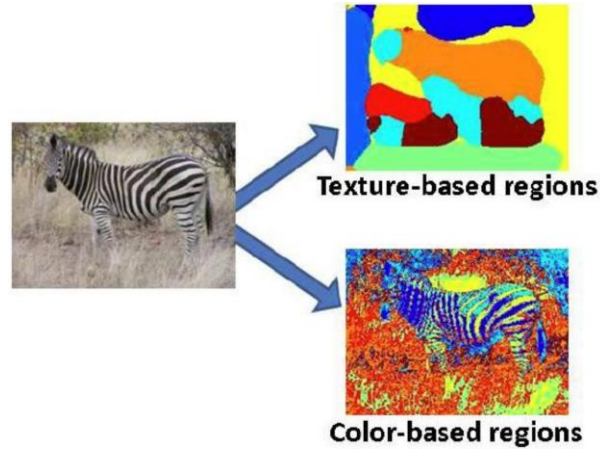
“Textons”

- › Find “textons” by clustering vectors of filter bank outputs
- › Describe texture in a window based on texton histogram





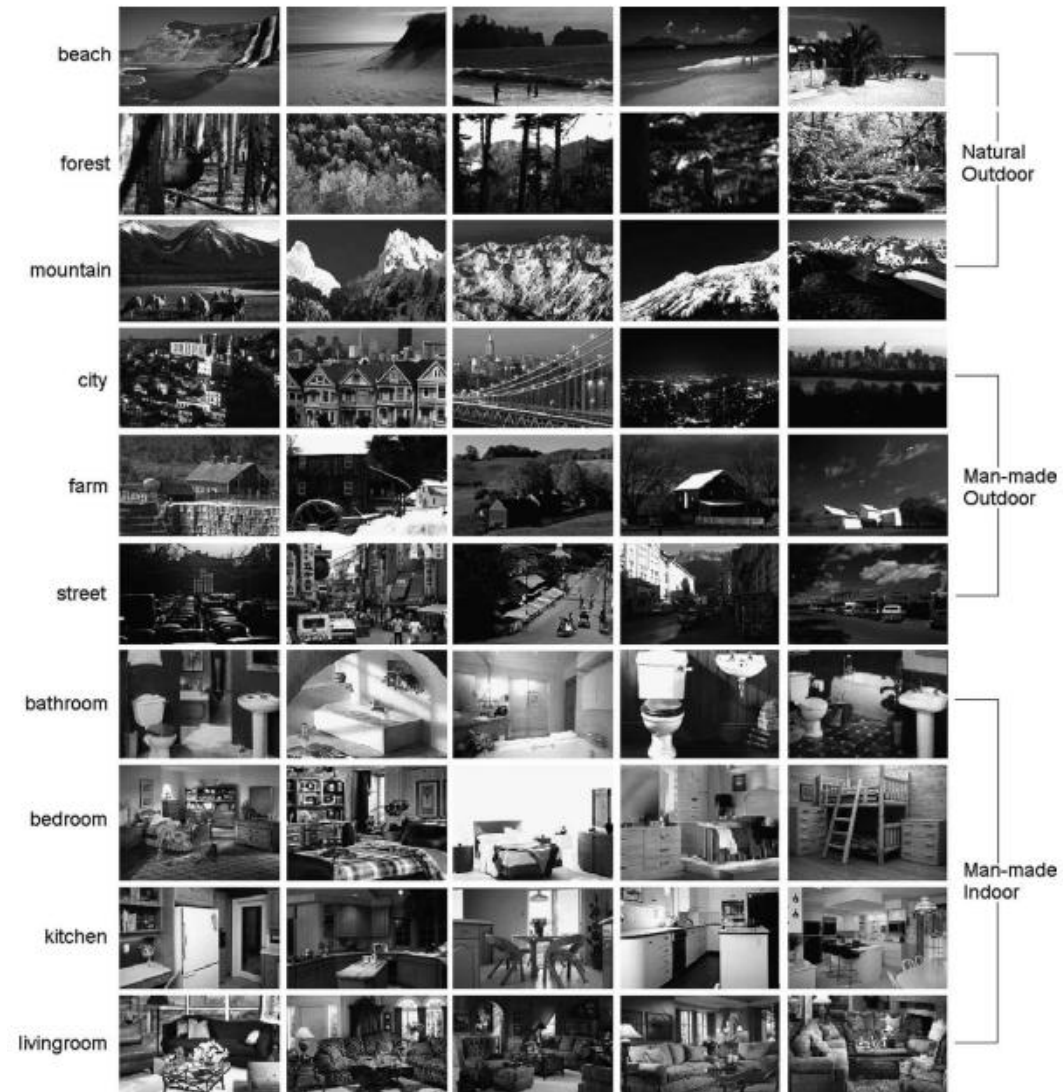
Color vs. Texture distribution



- › These matched images look very similar in terms of their color distributions.
- › Can texture distributions help distinguish them better?

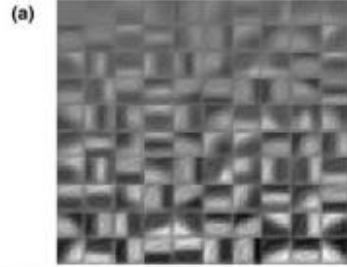


Application: Scene classification





Application: Scene classification



100 types of textures were found (by clustering) \Rightarrow called “**universal textons**”

Assign each pixel to nearest texton
Build histogram of textons

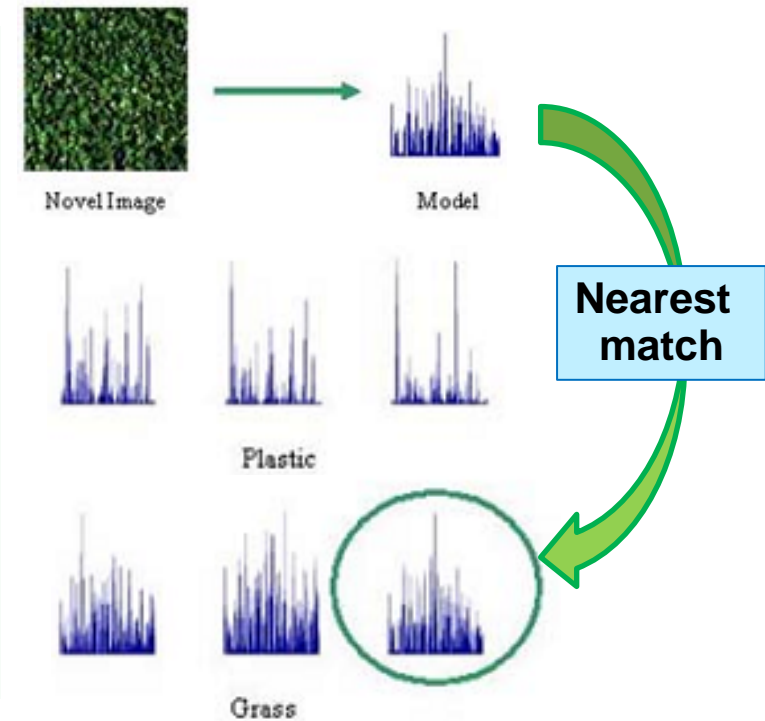
Classification by **chi square**, χ^2
measure to match texton histogram
against stored examples. Take the
nearest match.

$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{k=1}^K \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

Arguably better than
Euclidean distance.

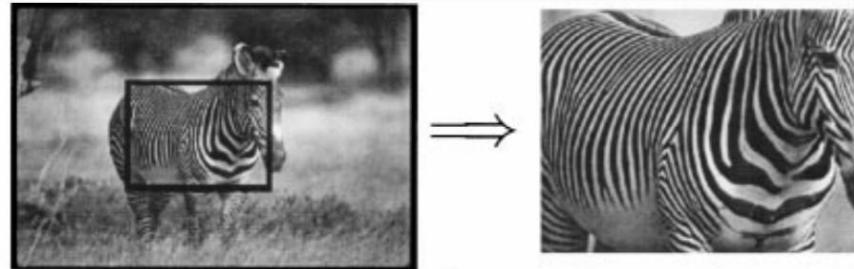


Application: Material classification





Application: Image Retrieval



(a)

Texture features
for image retrieval



[Introduction to
Earth Mover's distance](#)

Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99-121, November 2000,

SUMMARY

- › Textures
 - › Defining textures and representing them
- › Texture Representation
 - › Simple features (mean, std. dev.) from filters
 - › Filter bank – a series of filters
 - › Histogram of texture feature occurrences
- › Applications
 - › Scene classification
 - › Texture Matching
 - › Image Retrieval
- › Next
 - › Region Segmentation
- › Recommended Reading
 - › [Forsyth & Ponce] Chapter 10 (10.1 in particular)

