# Content-Based Image Retrieval
# Readings: Chapter 8: 8.1-8.4

- Queries
- Commercial Systems
- Retrieval Features
- Indexing in the FIDS System
- Lead-in to Object Recognition

# Content-based Image Retrieval (CBIR)

Searching a large database for images that *match* a query:

- What kinds of databases?
- What kinds of queries?
- What constitutes a match?
- How do we make such searches efficient?

# Applications

- Art Collections
  e.g. Fine Arts Museum of San Francisco
- Medical Image Databases
  CT, MRI, Ultrasound, The Visible Human
- Scientific Databases
  e.g. Earth Sciences
- General Image Collections for Licensing
  Corbis, Getty Images
- The World Wide Web
  Google, Microsoft, etc

# What is a query?

- an image you already have
- a rough sketch you draw
- a symbolic description of what you want
  e.g. an image of a man and a woman on a beach

# Some Systems You Can Try

Corbis Stock Photography and Pictures

http://pro.corbis.com/

- Corbis sells high-quality images for use in advertising, marketing, illustrating, etc.
- Search is entirely by keywords.
- Human indexers look at each new image and enter keywords.
- A thesaurus constructed from user queries is used.

# Microsoft Bing

- http://www.bing.com/
- first use keywords, ~~then mouse over an image and click on show similar images~~

# Google Similar Image Search

//https://images.google.com/imghp?hl=en&gws_rd=ssl

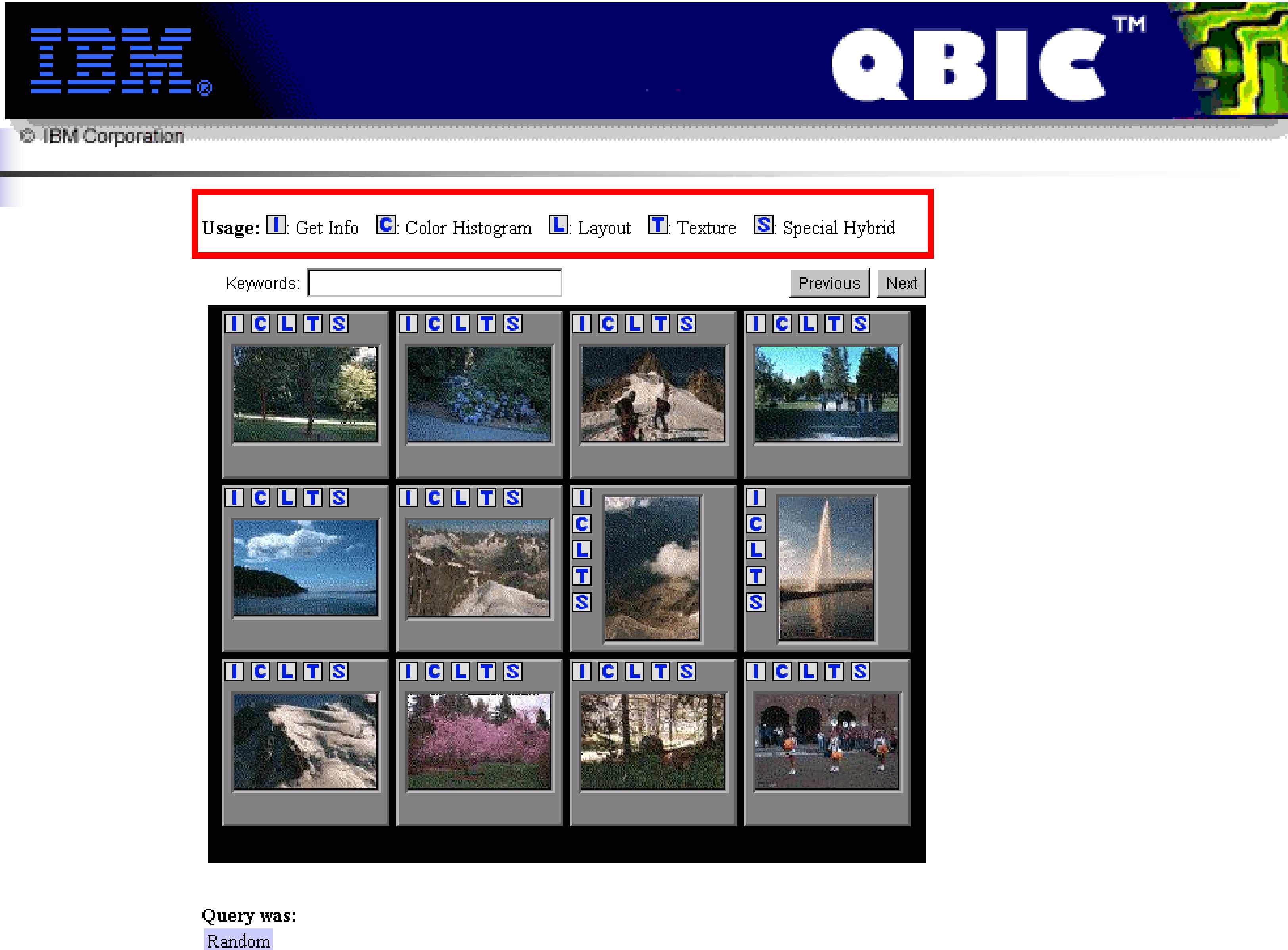
Drag in images or browse for them in the regular Google Image Tool.

# QBIC

IBM's QBIC (Query by Image Content

- The first commercial system.
- Uses or has-used color percentages, color layout, texture, shape, location, and keywords.

# Original QBIC system looked like this



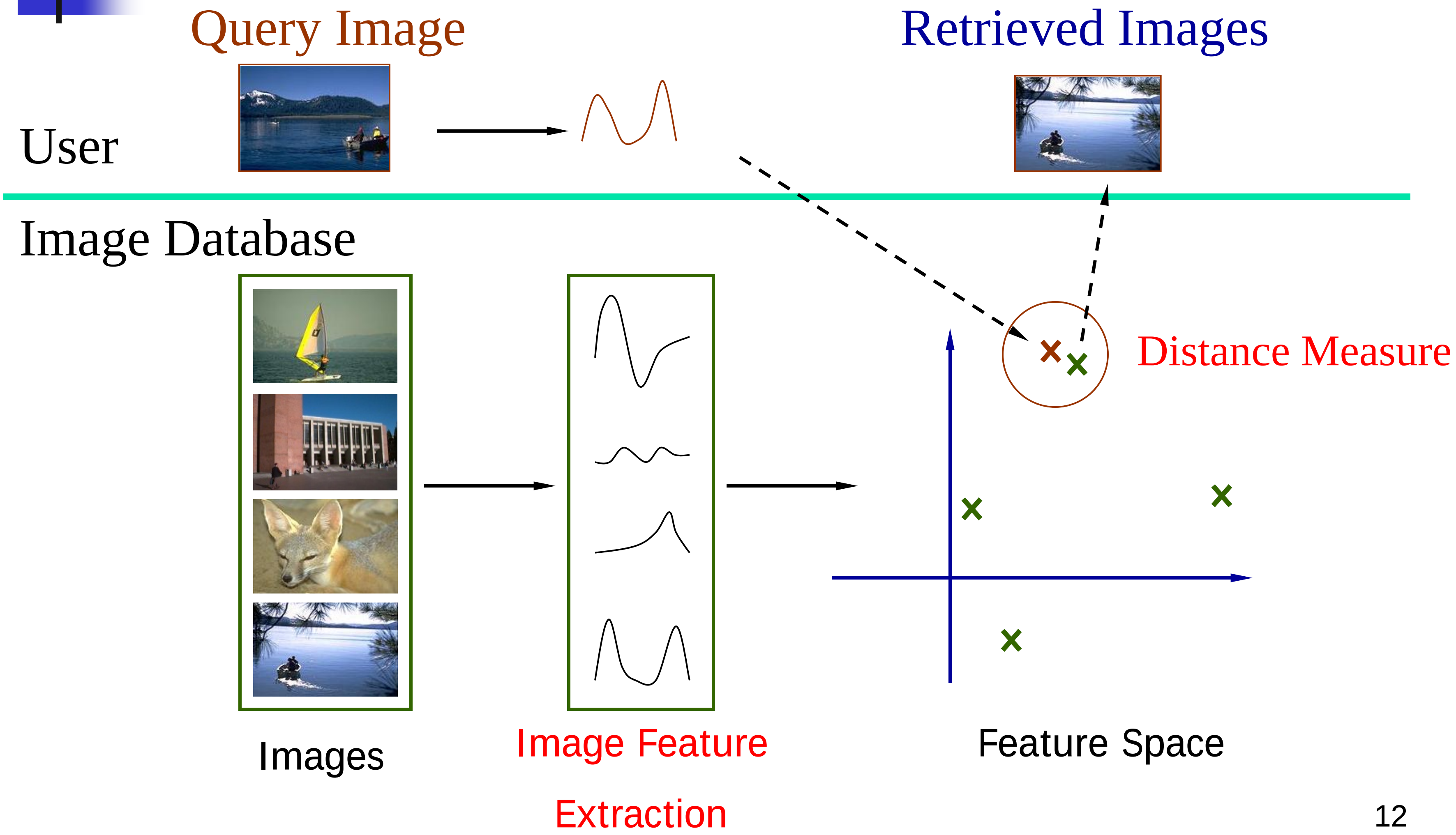
# Like

- Shopping search engine
- http://www.like.com/
- Google bought it.

# Problem with Text-Based Search

- Retrieval for pigs for the color chapter of my book
- Small company (was called Ditto)
- Allows you to search for pictures from web pages

# Image Features / Distance Measures

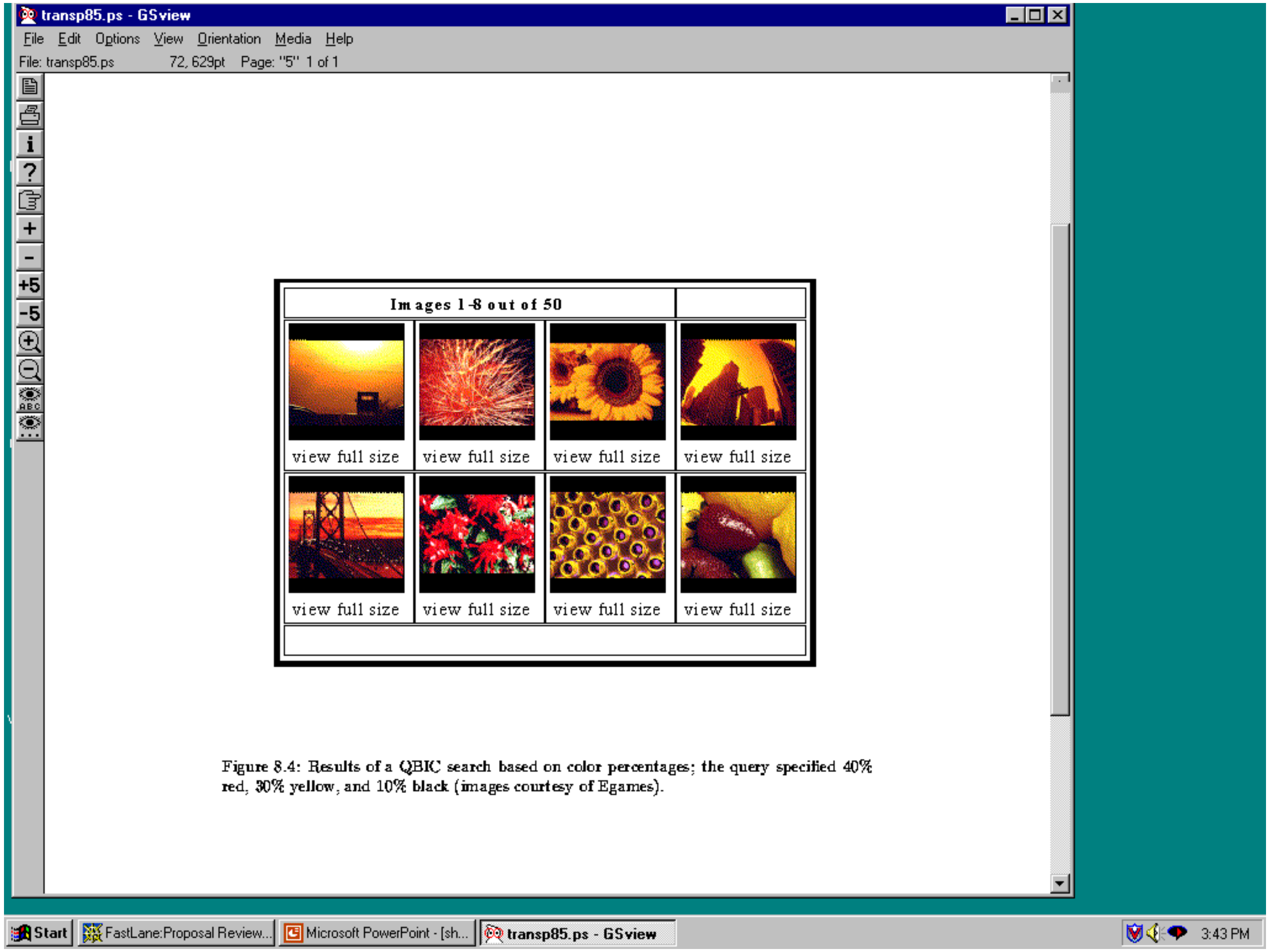


# Features

- Color (histograms, gridded layout, wavelets)
- Texture (Laws, Gabor filters, local binary pattern)
- Shape (first segment the image, then use statistical or structural shape similarity measures)
- Objects and their Relationships

This is the most powerful, but you have to be able to recognize the objects!

# Color Histograms



# QBIC's Histogram Similarity

The QBIC color histogram distance is:

$$d_{hist}(I,Q) = (h(I) - h(Q))^T \mathbf{A} \, (h(I) - h(Q))$$

- h(I) is a K-bin histogram of a database image
- h(Q) is a K-bin histogram of the query image
- A is a K x K similarity matrix

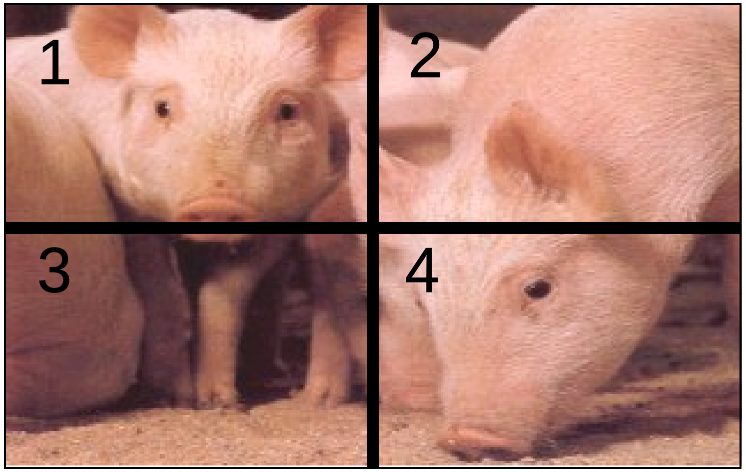
# Similarity Matrix

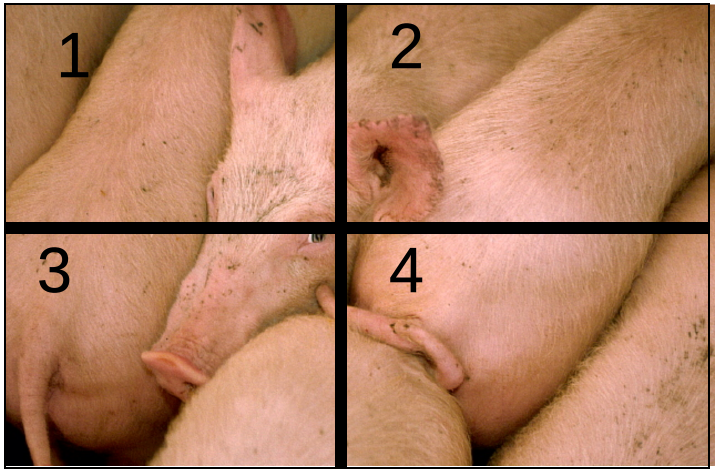
| | R | G | B | Y | C | V |
|---|---|---|---|---|---|---|
| R | 1 | 0 | 0 | .5 | 0 | .5 |
| G | 0 | 1 | 0 | .5 | .5 | 0 |
| B | 0 | 0 | 1 | | ? | |
| Y | | | | 1 | | |
| C | | ? | | | 1 | |
| V | | | | | | 1 |

How similar is blue to cyan?

# Gridded Color

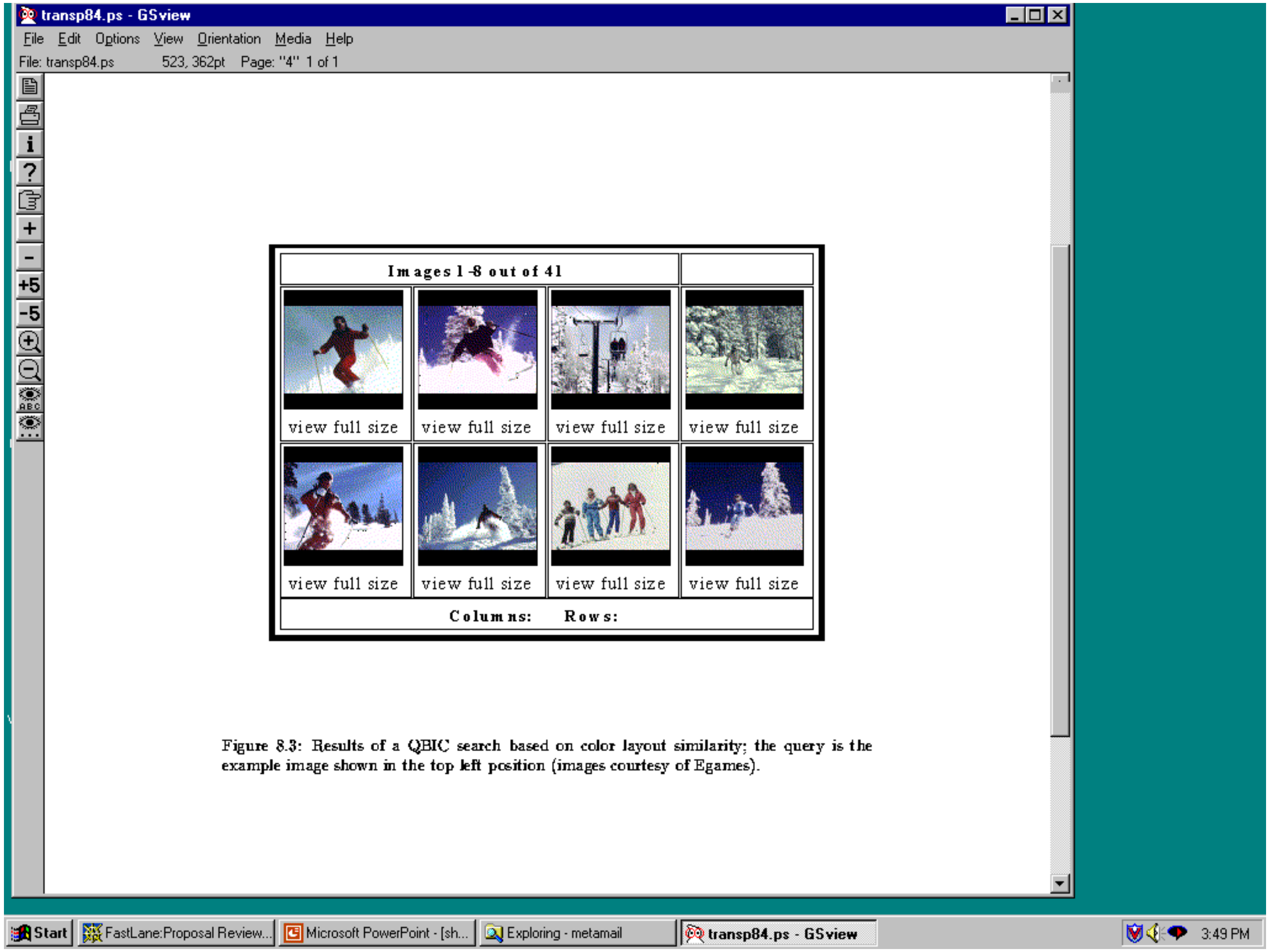
Gridded color distance is the sum of the color distances in each of the corresponding grid squares.




What color distance would you use for a pair of grid squares?

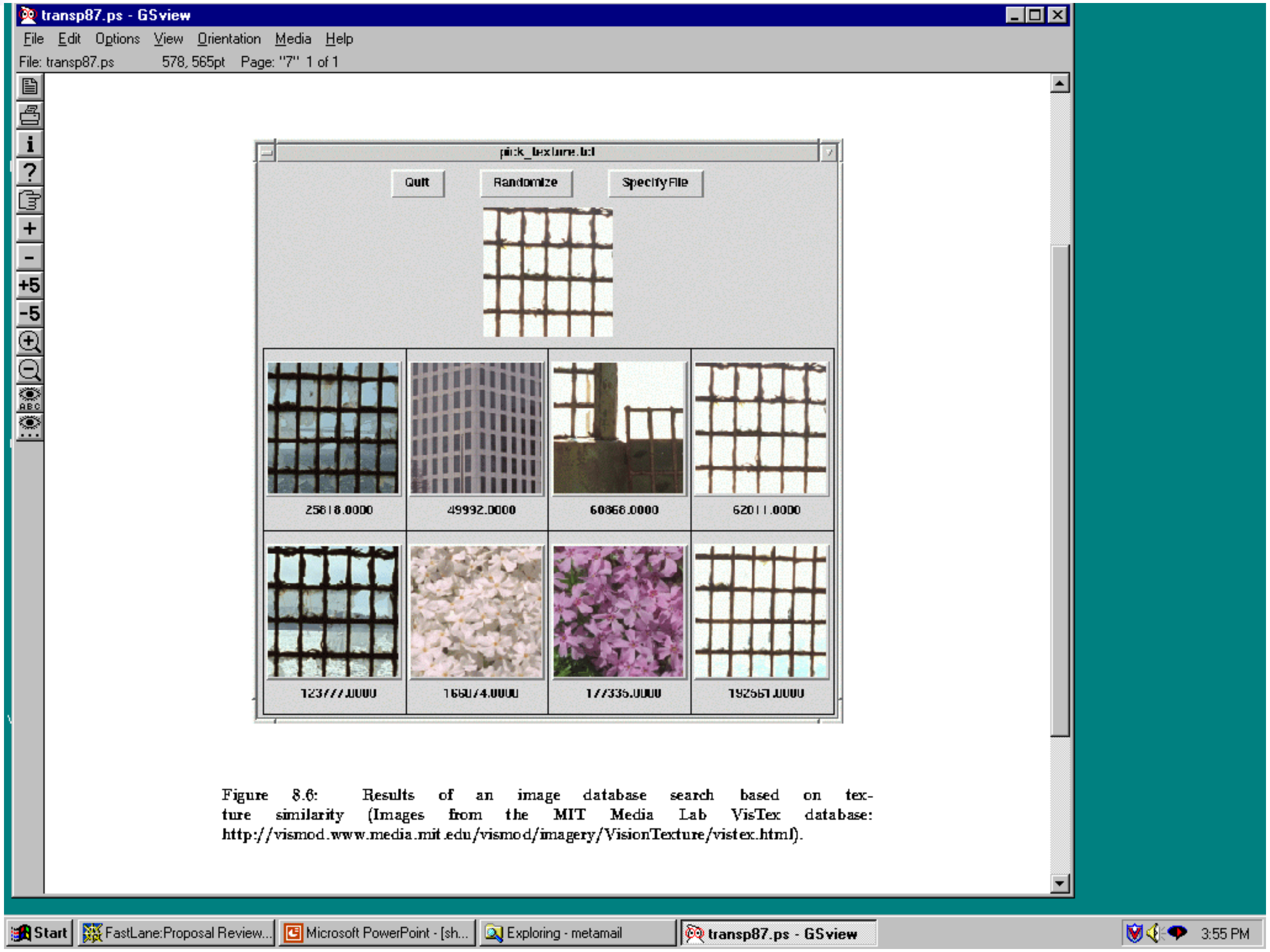
# Color Layout (IBM's Gridded Color)



# Texture Distances

- Pick and Click (user clicks on a pixel and system retrieves images that have in them a region with similar texture to the region surrounding it.
- Gridded (just like gridded color, but use texture).
- Histogram-based (e.g. compare the LBP histograms).

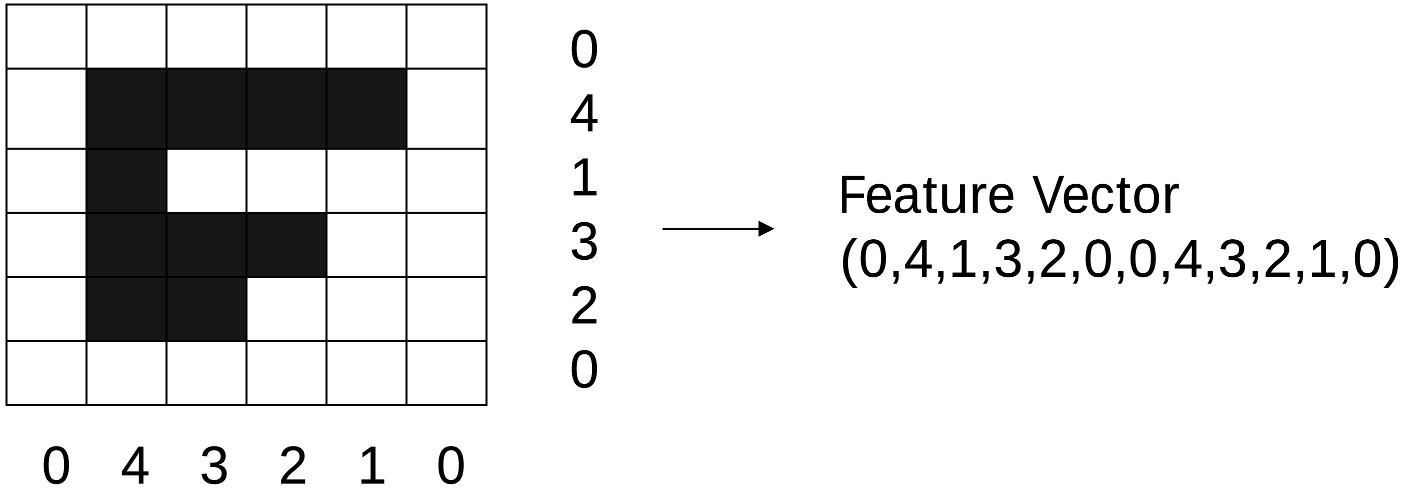
# Laws Texture



# Shape Distances

- Shape goes one step further than color and texture.
- It requires identification of regions to compare.
- There have been many shape similarity measures suggested for pattern recognition that can be used to construct shape distance measures.

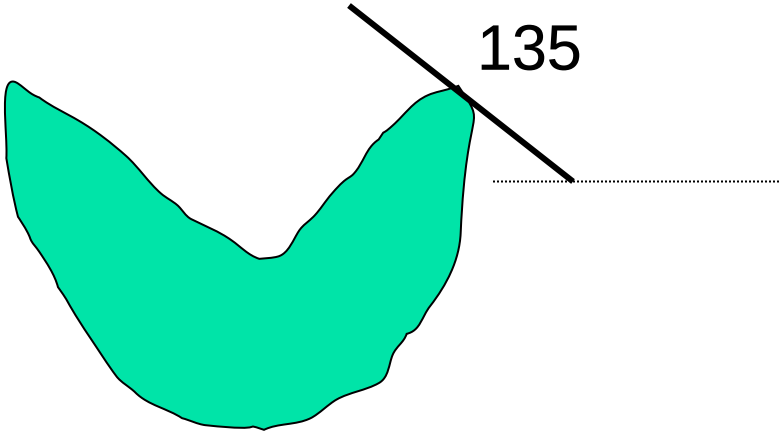
# Global Shape Properties: Projection Matching

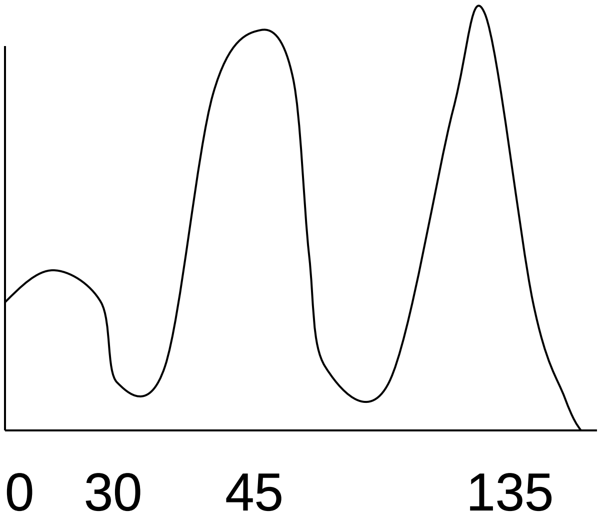



In projection matching, the horizontal and vertical projections form a histogram.

What are the weaknesses of this method? strengths?

# Global Shape Properties: Tangent-Angle Histograms




Is this feature invariant to starting point?
Is it invariant to size, translation, rotation?

# Boundary Matching

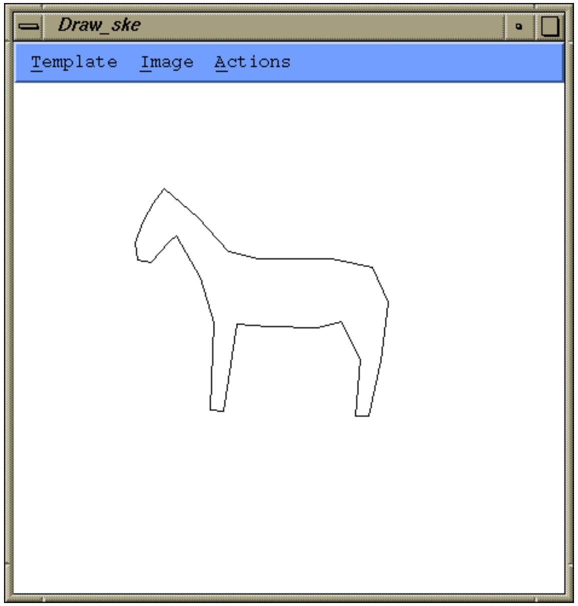
- Fourier Descriptors
- Sides and Angles
- Elastic Matching

The distance between query shape and image shape has two components:

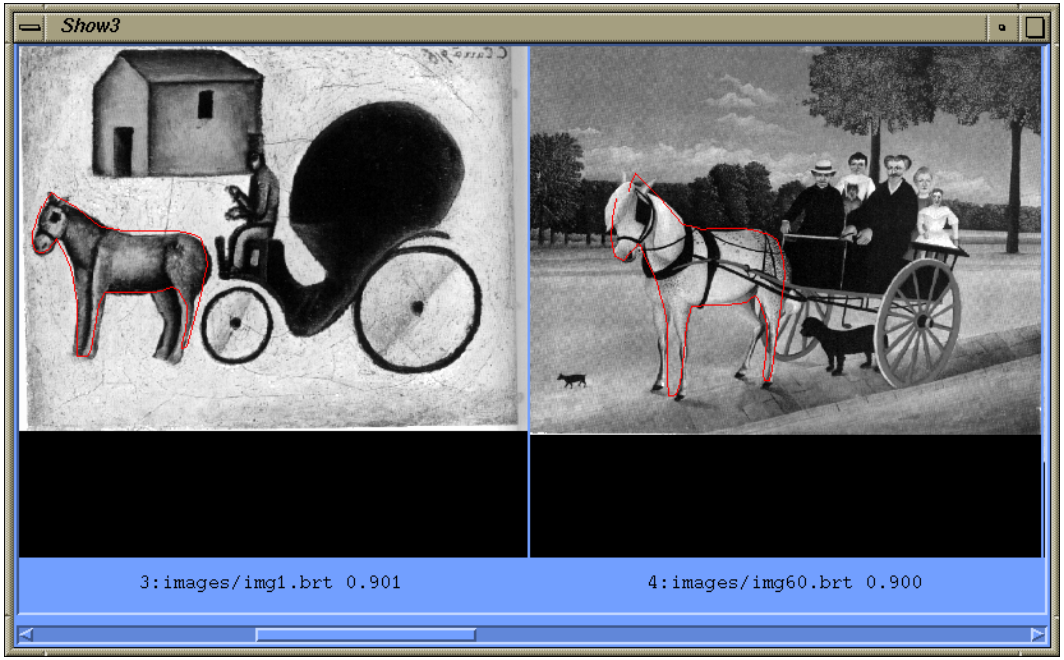
1. energy required to deform the query shape into one that best matches the image shape

2. a measure of how well the deformed query matches the image

# Del Bimbo Elastic Shape Matching




query




retrieved images

# Regions and Relationships

- Segment the image into regions
- Find their properties and interrelationships
- Construct a graph representation with nodes for regions and edges for spatial relationships
- Use graph matching to compare images

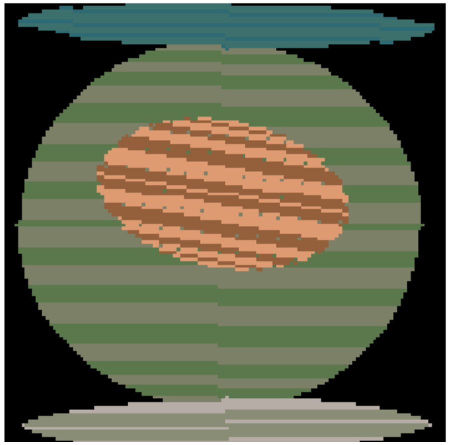
Like what?

# Blobworld (Carson et al, 1999)

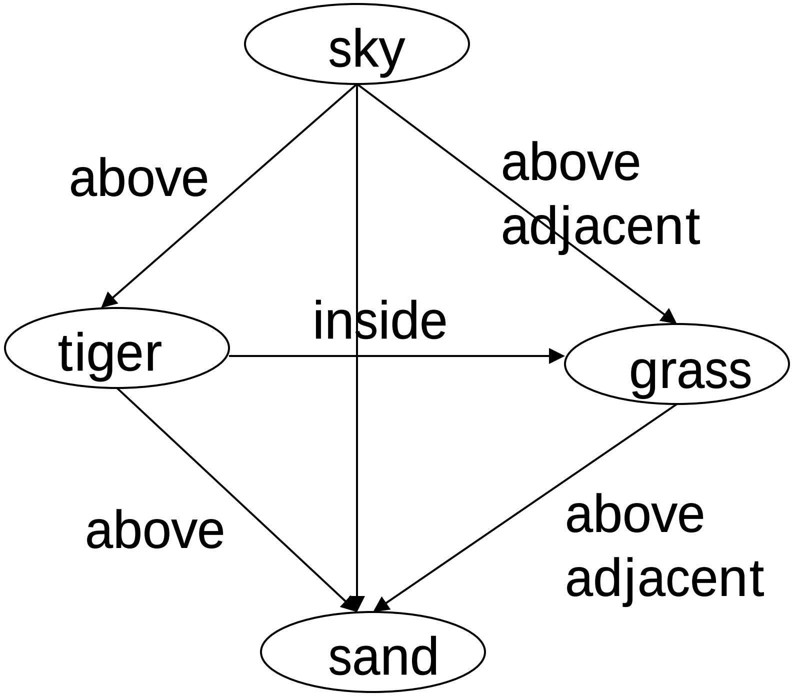
- Segmented the query (and all database images) using EM on color+texture
- Allowed users to select the most important region and what characteristics of it (color, texture, location)
- Asked users if the background was also important

# Tiger Image as a Graph (motivated by Blobworld)

image

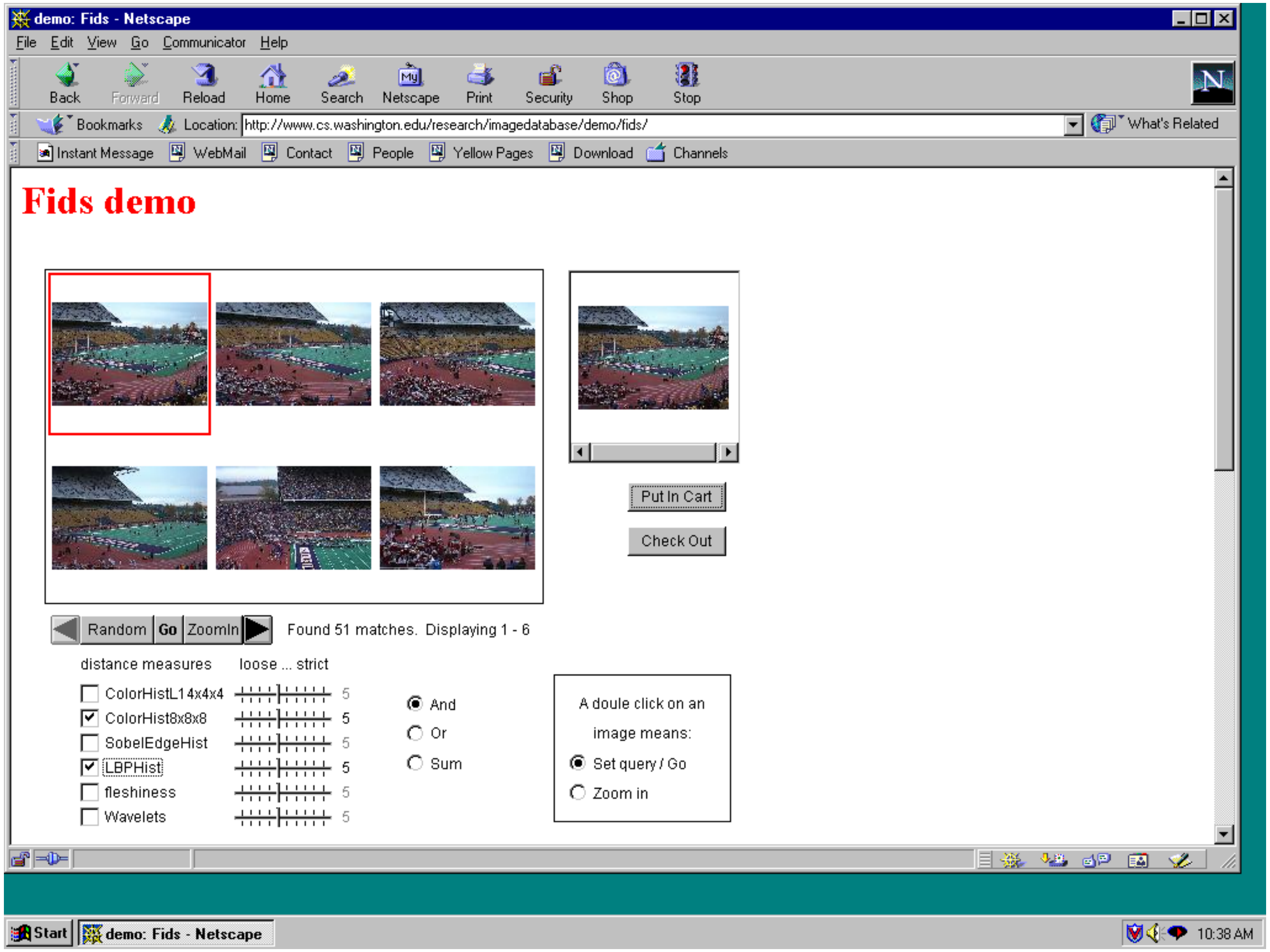


abstract regions



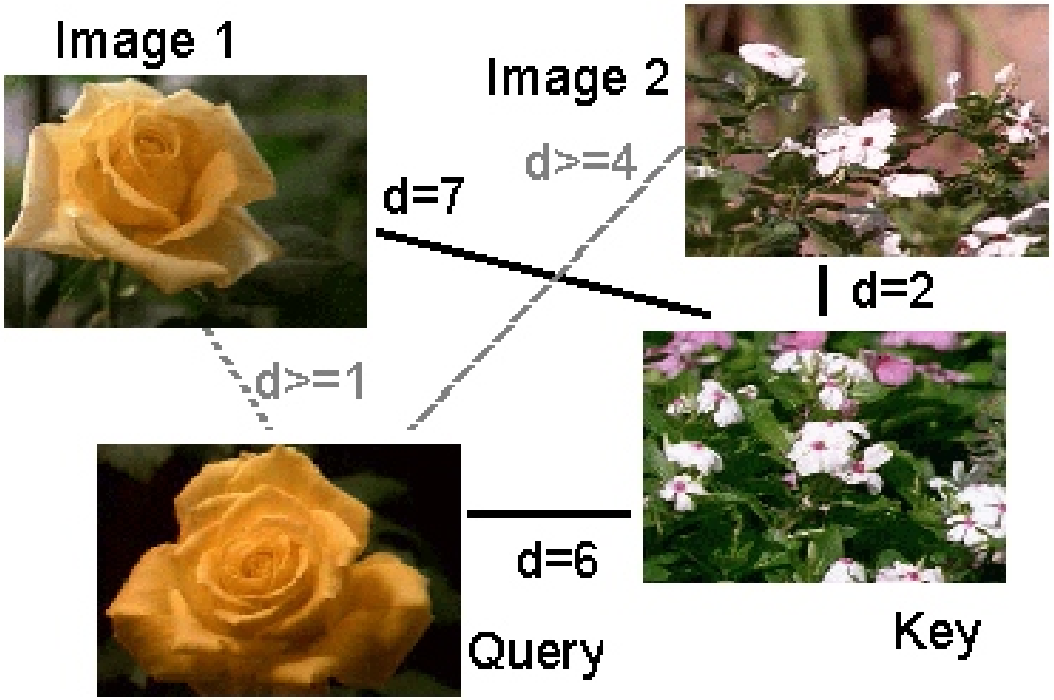
# Andy Berman's FIDS System

multiple distance measures
Boolean and linear combinations
efficient indexing using images as keys



# Andy Berman's FIDS System:

Use of key images and the triangle inequality for efficient retrieval. d(I,Q) >= |d((I,K) – d(Q,K)|



## Andy Berman's FIDS System:

## Bare-Bones Triangle Inequality Algorithm

### Offline

1. Choose a small set of key images
2. Store distances from database images to keys

### Online (given query Q)

1. Compute the distance from Q to each key
2. Obtain lower bounds on distances to database images
3. Threshold or return all images in order of lower bounds

# Andy Berman's FIDS System:

## Flexible Image Database System: Example

An example from our system using a simple color measure.

# images in system: 37,748

threshold: 100 out of 1000

# images eliminated: 37,729

# Andy Berman's FIDS System:

## Bare-Bones Algorithm with Multiple Distance Measures

### Offline

1. Choose key images for each measure
2. Store distances from database images to keys for all measures

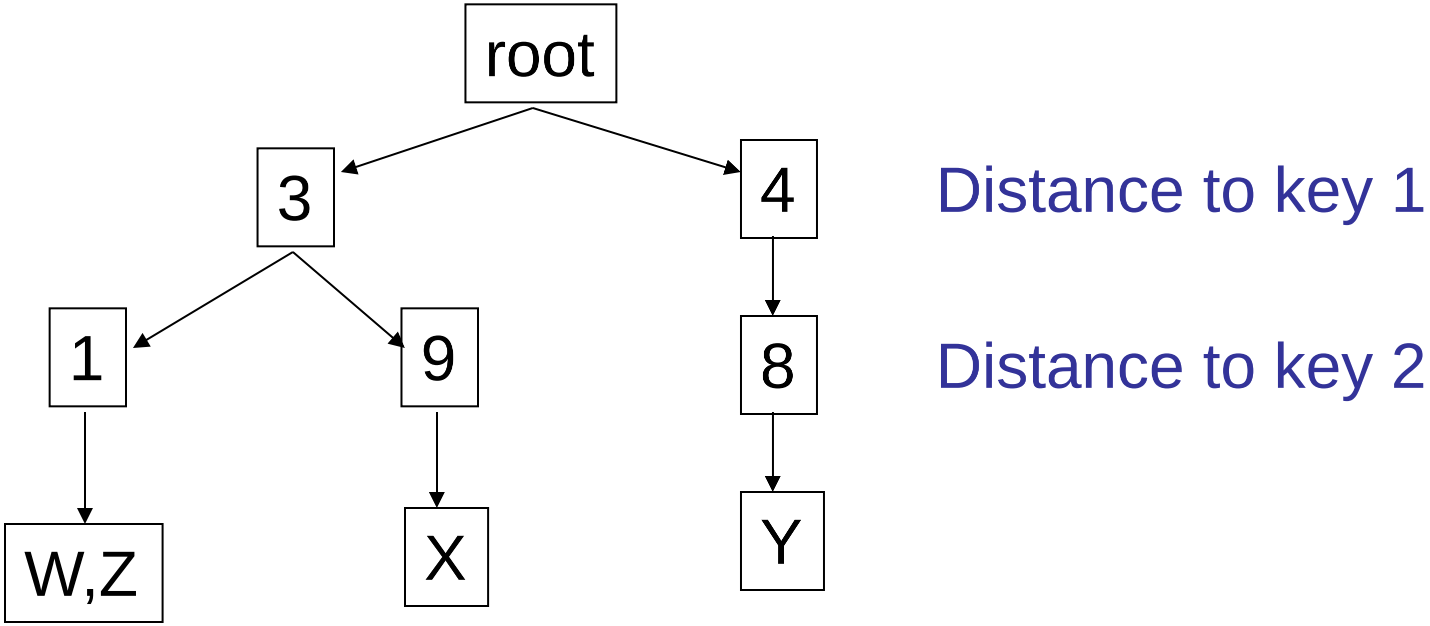
### Online (given query Q)

1. Calculate lower bounds for each measure
2. Combine to form lower bounds for composite measures
3. Continue as in single measure algorithm

# Andy Berman's FIDS System:

## Triangle Tries

A triangle trie is a tree structure that stores the distances from database images to each of the keys, one key per tree level.



## Andy Berman's FIDS System:

## Triangle Tries and Two-Stage Pruning

- First Stage: Use a short triangle trie.
- Second Stage: Bare-bones algorithm on the images returned from the triangle-trie stage.

The quality of the output is the same as with the bare-bones algorithm itself, but execution is faster.

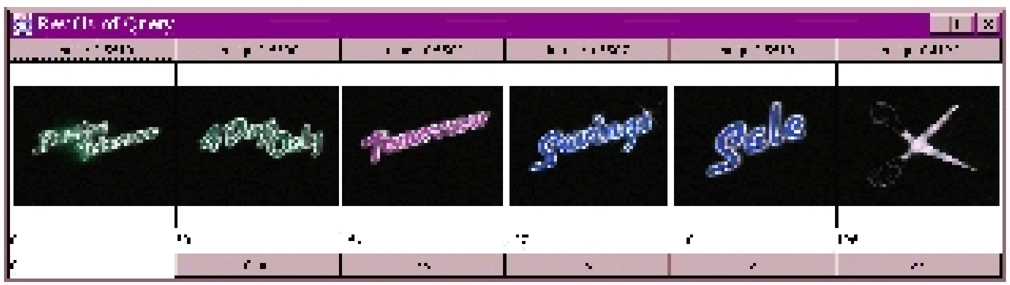
# Andy Berman's FIDS System:

## Flexible Image Database System: Example

# of images in system: 37,748
Depth of triangle trie: 6
# of images eliminated by trie: 30,300
# images eliminated by second-stage: 7429
19 images remaining, as before

# Andy Berman's FIDS System:

## Flexible Image Database System: Example



Example from our system using a combination color+texture measure

# images in system: 37,748

# images from color trie: 3,676

# images from texture trie: 497

# images in merged set: 3,785

# images eliminated: 33,963

Andy Berman's FIDS System:

## Performance on a Pentium Pro 200-mHz

Step 1. Extract features from query image. (.02s ≤ t ≤ .25s)

Step 2. Calculate distance from query to key images.
(1μs ≤ t ≤ .8ms)

Step 3. Calculate lower bound distances.
(t ≈ 4ms per 1000 images using 35 keys, which is about 250,000 images per second.)

Step 4. Return the images with smallest lower bound distances.

# Demo of FIDS

- http://www.cs.washington.edu/research/imagedatabase/demo/

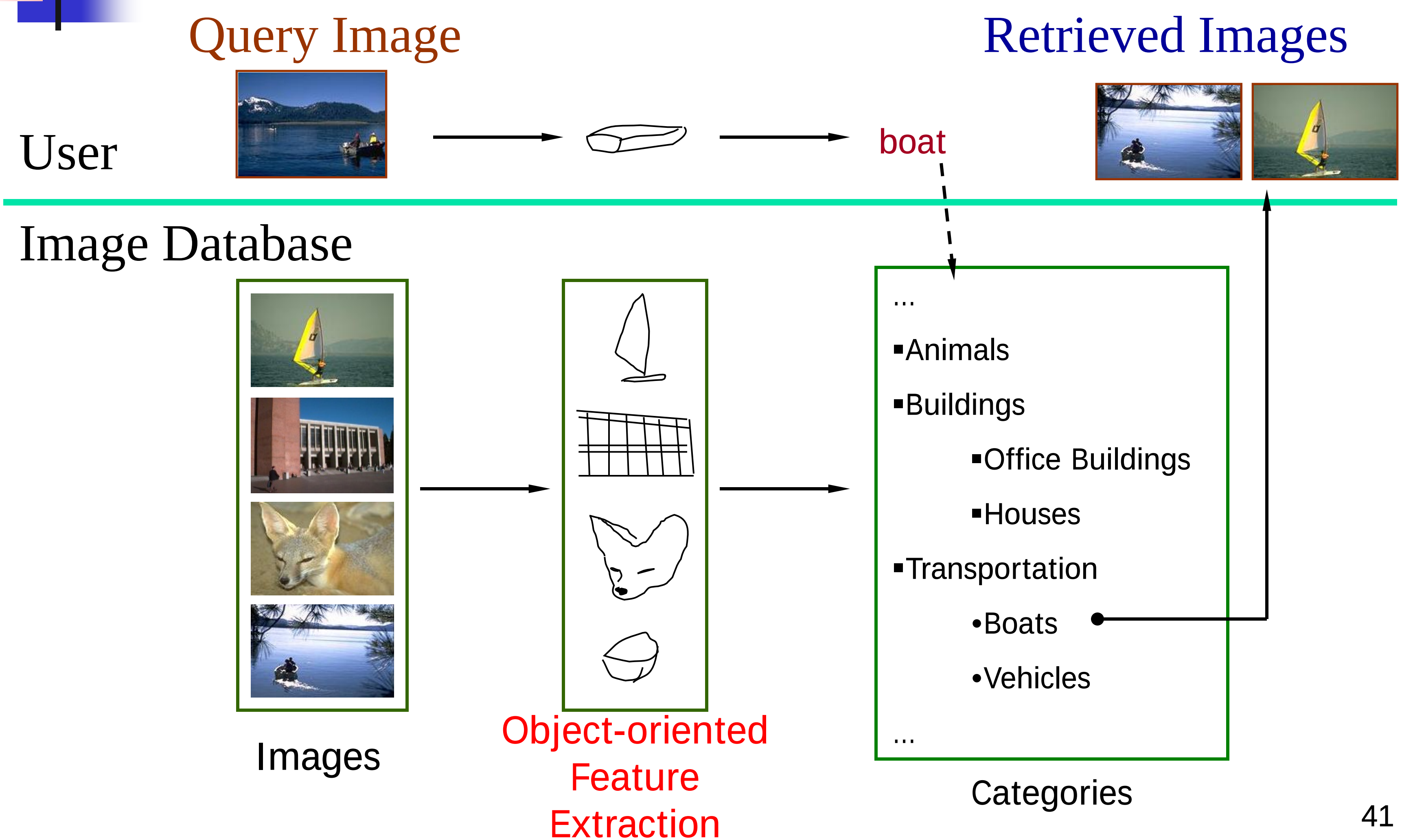
- Try this and the other demos on the same page.

# Weakness of Low-level Features

- Can't capture the high-level concepts



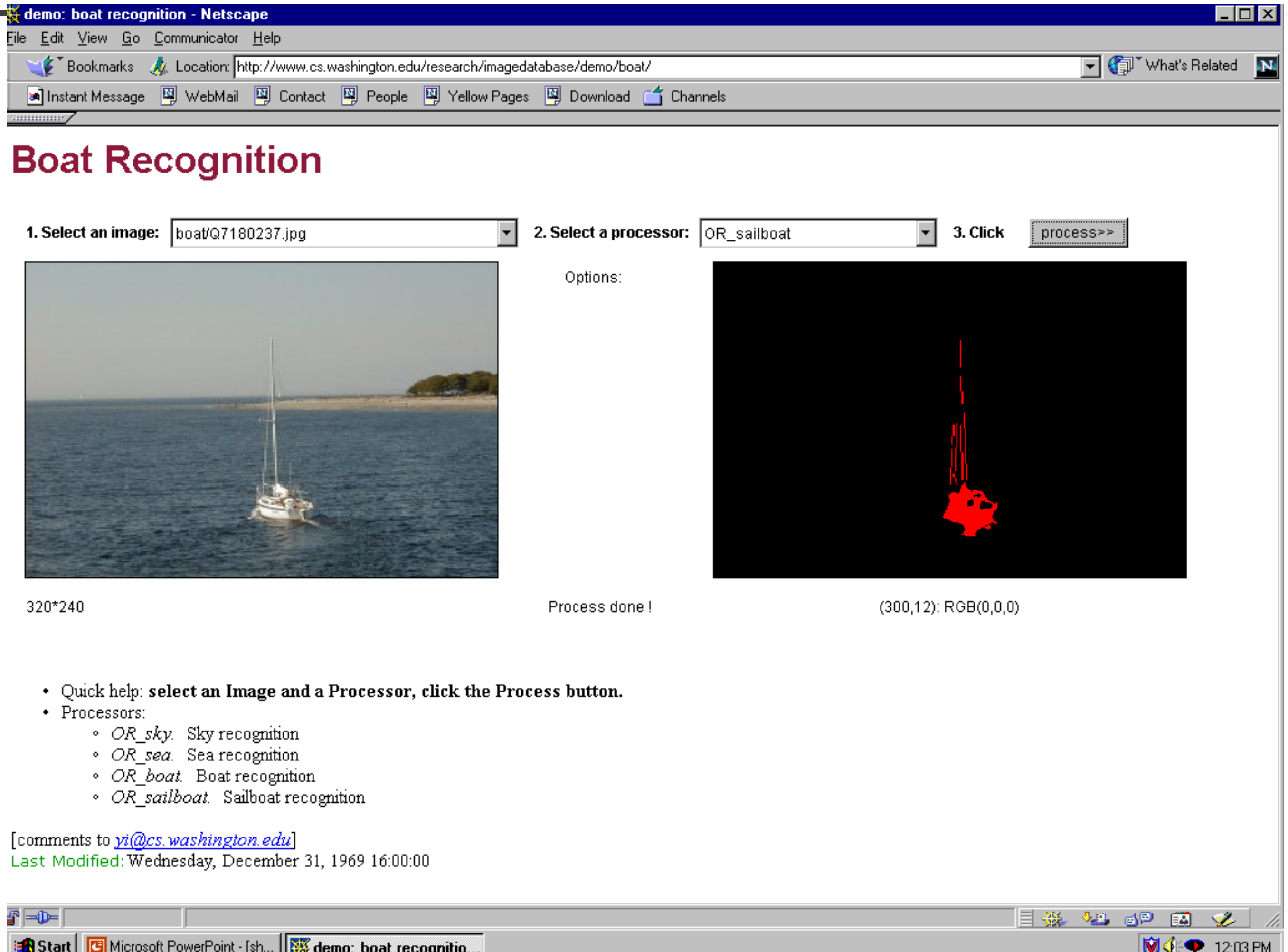
# Current Research Objective



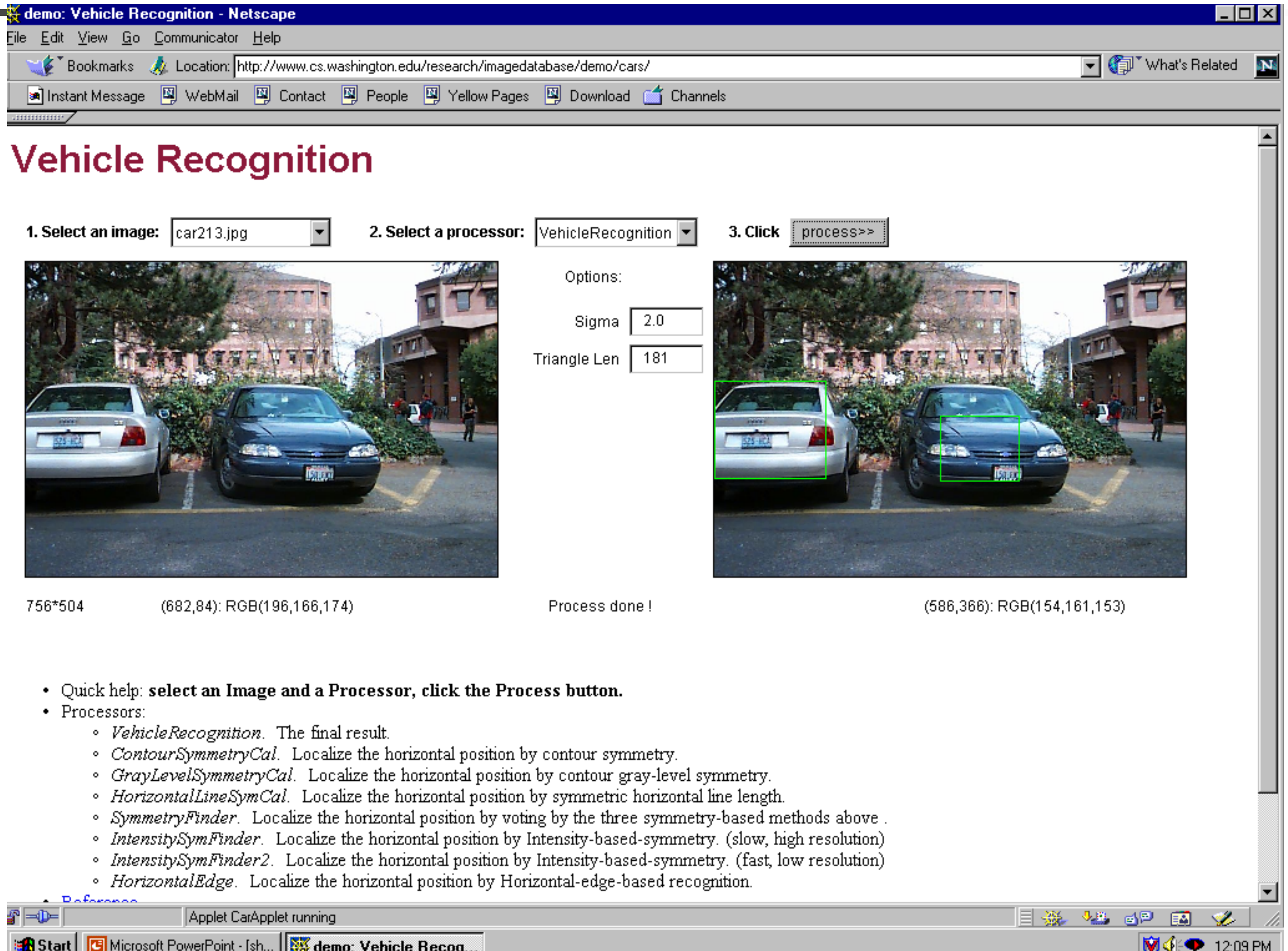
# Overall Approach

- Develop object recognizers for common objects
- Use these recognizers to design a new set of both low- and mid-level features
- Design a learning system that can use these features to recognize classes of objects

# Boat Recognition



# Vehicle Recognition



# Building Recognition

# Building Features: Consistent Line Clusters (CLC)

A **Consistent Line Cluster** is a set of lines that are homogeneous in terms of some line features.

- **Color-CLC**: The lines have the same color feature.
- **Orientation-CLC**: The lines are parallel to each other or converge to a common vanishing point.
- **Spatially-CLC**: The lines are in close proximity to each other.

# Color-CLC

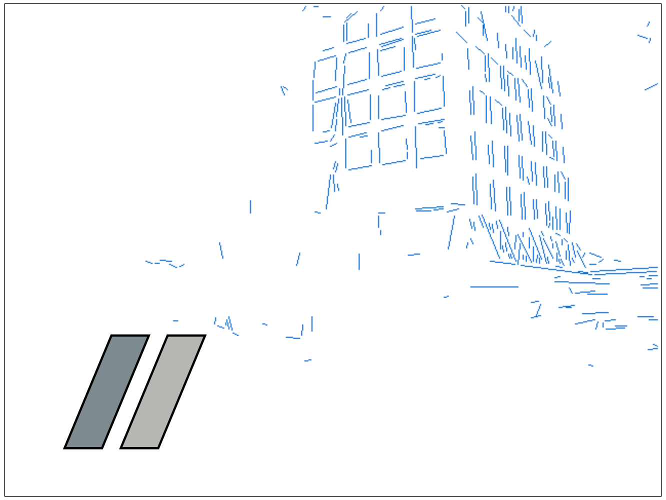
- Color feature of lines: color pair $(c_1,c_2)$
- Color pair space:

  RGB $(256^3*256^3)$ Too big!

  Dominant colors (20*20)
- Finding the color pairs:

  One line → Several color pairs
- Constructing Color-CLC: use clustering

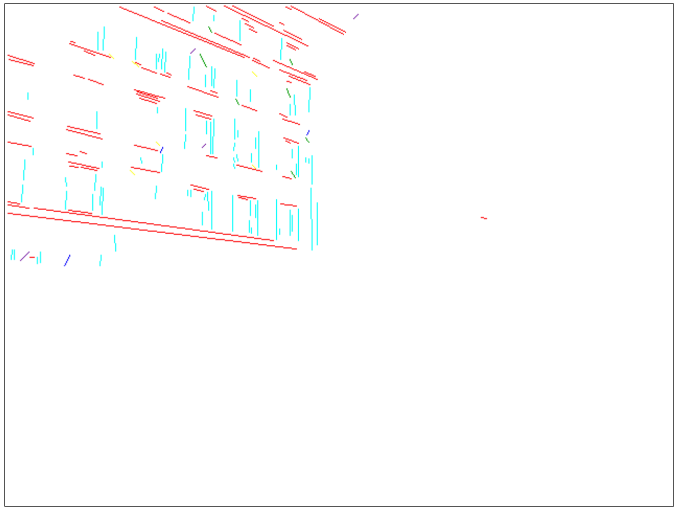
# Color-CLC

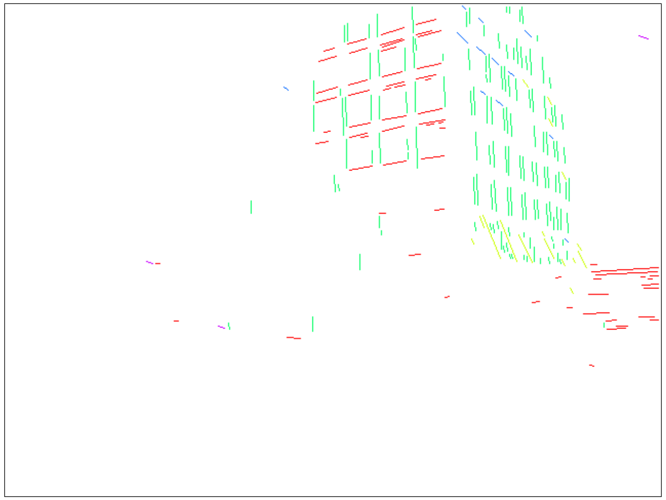


# Orientation-CLC

- The lines in an Orientation-CLC are parallel to each other in the 3D world
- The parallel lines of an object in a 2D image can be:
  - Parallel in 2D
  - Converging to a vanishing point (perspective)

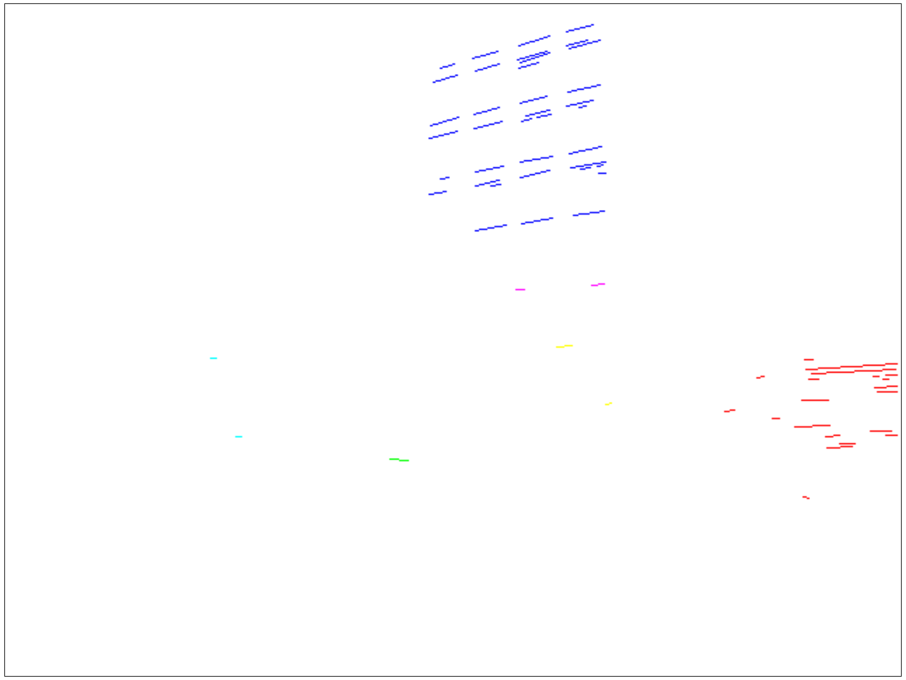
# Orientation-CLC





# Spatially-CLC

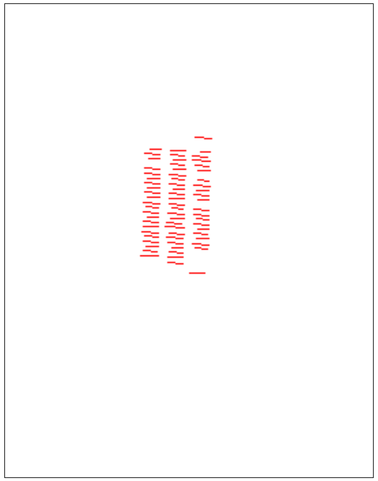
- Vertical position clustering
- Horizontal position clustering

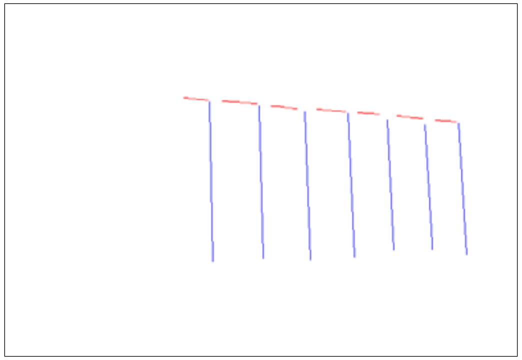


# Building Recognition by CLC

Two types of buildings → Two criteria

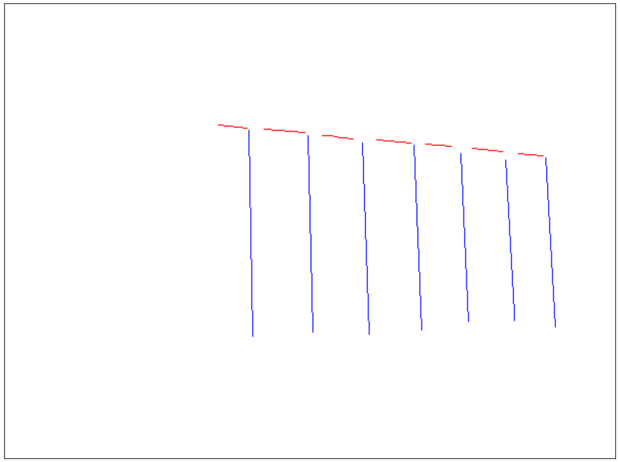
- Inter-relationship criterion
- Intra-relationship criterion





# Inter-relationship criterion

$$(N_{c1} > T_{i1} \text{ or } N_{c2} > T_{i1}) \text{ and } (N_{c1} + N_{c2}) > T_{i2}$$

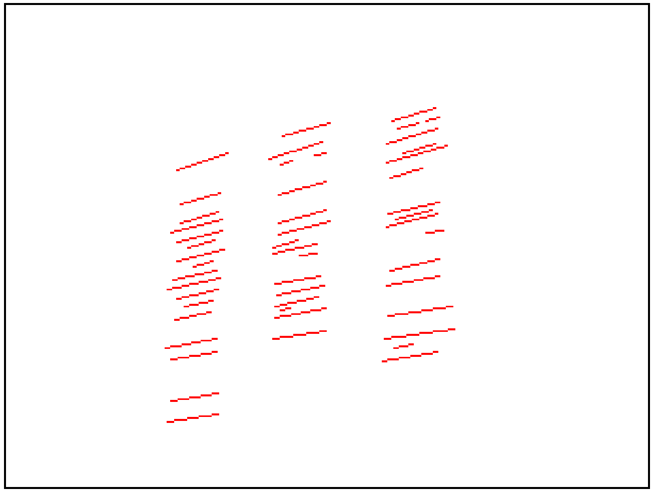


$N_{c1}$ = number of intersecting lines in cluster 1

$N_{c2}$ = number of intersecting lines in cluster 2

# Intra-relationship criterion

$$|S_o| > T_{j1} \text{ or } w(S_o) > T_{j2}$$



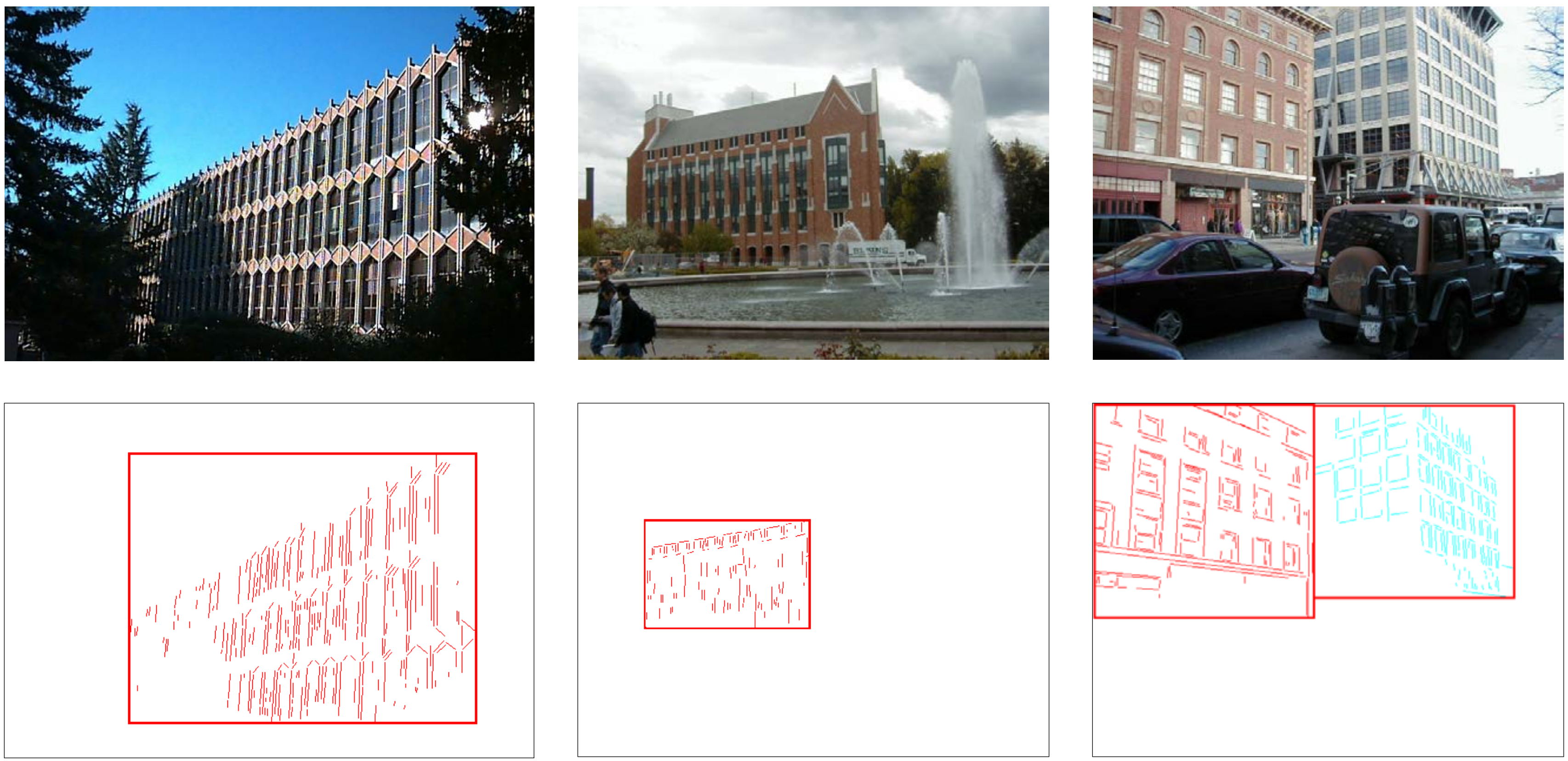
$S_0$ = set of heavily overlapping lines in a cluster

# Experimental Evaluation

- Object Recognition
  - 97 well-patterned buildings (bp): 97/97
  - 44 not well-patterned buildings (bnp): 42/44
  - 16 not patterned non-buildings (nbnp): 15/16 (one false positive)
  - 25 patterned non-buildings (nbp): 0/25
- CBIR

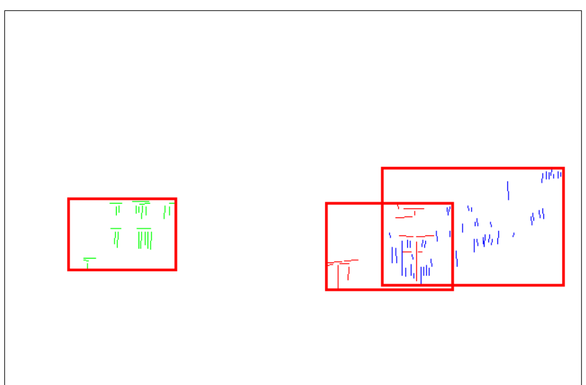
# Experimental Evaluation

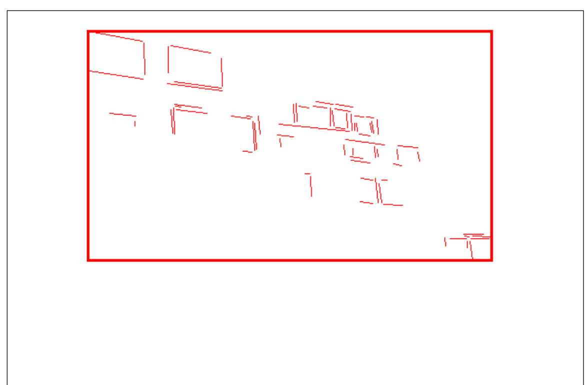
## Well-Patterned Buildings

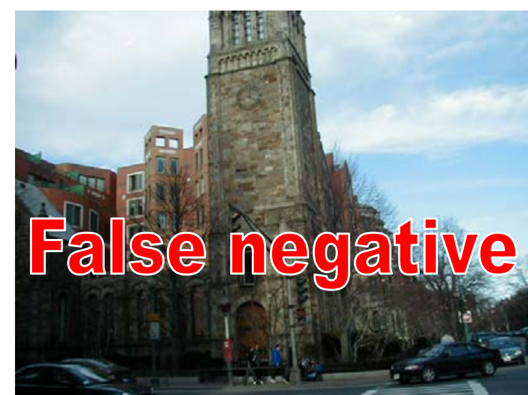


# Experimental Evaluation

## Non-Well-Patterned Buildings

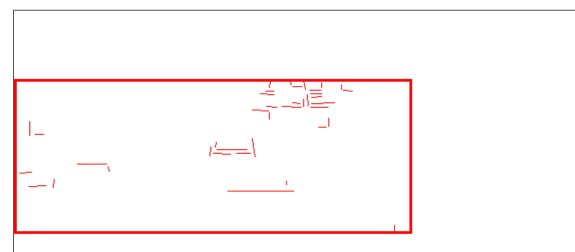







# Experimental Evaluation

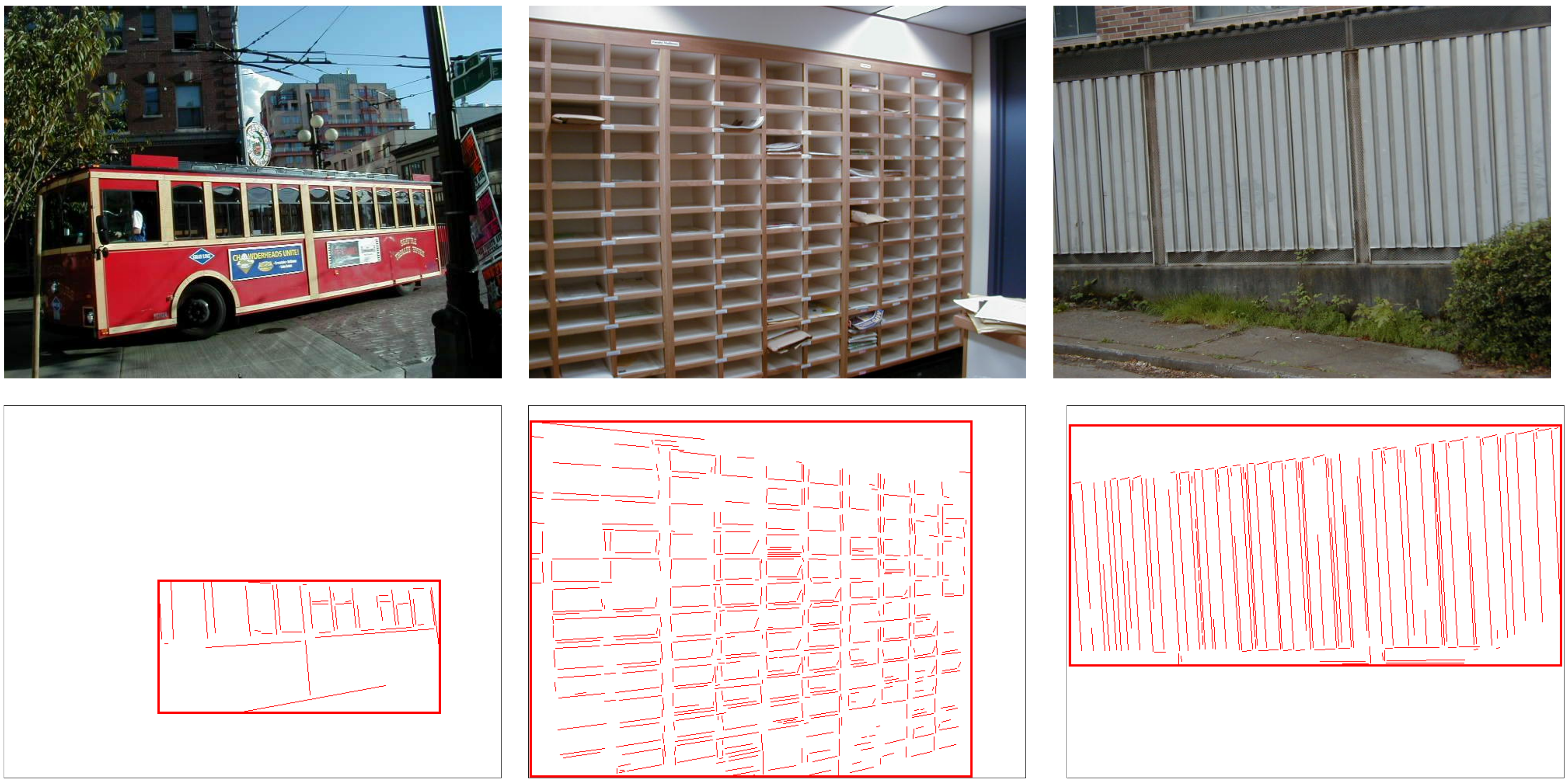
## Non-Well-Patterned Non-Buildings



**False positive**

# Experimental Evaluation

## Well-Patterned Non-Buildings (false positives)



# Experimental Evaluation (CBIR)

| | Total Positive Classification (#) | Total Negative Classification (#) | False positive (#) | False negative (#) | Accuracy (%) |
|---|---|---|---|---|---|
| Arborgreens | 0 | 47 | 0 | 0 | 100 |
| Campusinfall | 27 | 21 | 0 | 5 | 89.6 |
| Cannonbeach | 30 | 18 | 0 | 6 | 87.5 |
| Yellowstone | 4 | 44 | 4 | 0 | 91.7 |

# Experimental Evaluation (CBIR)

## False positives from Yellowstone