

# Object Recognition I

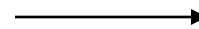
Linda Shapiro

EE/CSE 576

# Low- to High-Level

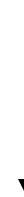


low-level



edge image

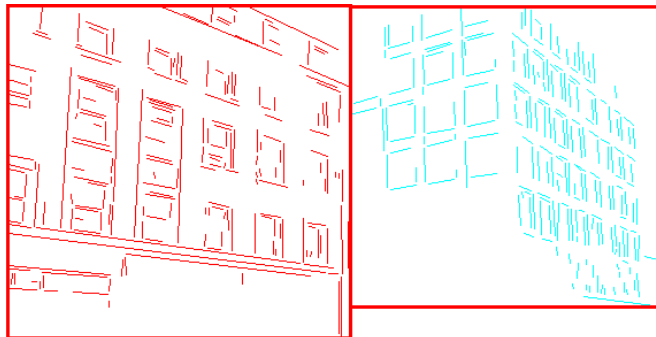
mid-level



consistent  
line clusters



high-level



## Building Recognition

# High-Level Computer Vision

- Detection of classes of objects (faces, motorbikes, trees, cheetahs) in images
- Recognition of specific objects such as George Bush or machine part #45732
- Classification of images or parts of images for medical or scientific applications
- Recognition of events in surveillance videos
- Measurement of distances for robotics

# High-level vision uses techniques from AI

- Graph-Matching: A\*, Constraint Satisfaction, Branch and Bound Search, Simulated Annealing
- Learning Methodologies: Decision Trees, Neural Nets, SVMs, EM Classifier
- Probabilistic Reasoning, Belief Propagation, Graphical Models

# Graph Matching for Object Recognition

- For each specific object, we have a geometric model.
- The geometric model leads to a symbolic model in terms of image features and their spatial relationships.
- An image is represented by all of its features and their spatial relationships.
- This leads to a graph matching problem.

# House Example

2D model

2D image

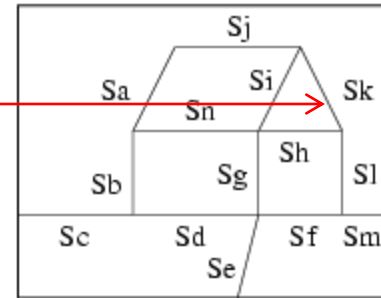
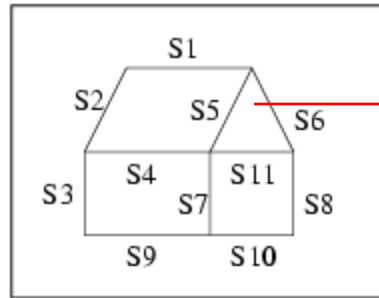


Image 1

**P**

Image 2

**L**

**RP and RL are connection relations.**

$$P = \{S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11\}.$$

$$L = \{Sa, Sb, Sc, Sd, Se, Sf, Sg, Sh, Si, Sj, Sk, Sl, Sm\}.$$

$$R_P = \{ (S1, S2), (S1, S5), (S1, S6), (S2, S3), (S2, S4), (S3, S4), (S3, S9), (S4, S5), (S4, S7), (S4, S11), (S5, S6), (S5, S7), (S5, S11), (S6, S8), (S6, S11), (S7, S9), (S7, S10), (S7, S11), (S8, S10), (S8, S11), (S9, S10) \}.$$

$$R_L = \{ (Sa, Sb), (Sa, Sj), (Sa, Sn), (Sb, Sc), (Sb, Sd), (Sb, Sn), (Sc, Sd), (Sd, Se), (Sd, Sf), (Sd, Sg), (Se, Sf), (Se, Sg), (Sf, Sg), (Sf, Sl), (Sf, Sm), (Sg, Sh), (Sg, Si), (Sg, Sn), (Sh, Si), (Sh, Sk), (Sh, Sl), (Sh, Sn), (Si, Sj), (Si, Sk), (Si, Sn), (Sj, Sk), (Sk, Sl), (Sl, Sm) \}.$$

$$f(S1) = S_j$$

$$f(S4) = S_n$$

$$f(S7) = S_g$$

$$f(S10) = S_f$$

$$f(S2) = S_a$$

$$f(S5) = S_i$$

$$f(S8) = S_l$$

$$f(S11) = S_h$$

$$f(S3) = S_b$$

$$f(S6) = S_k$$

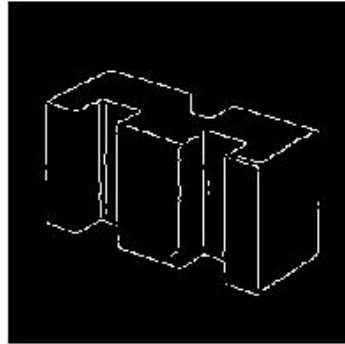
$$f(S9) = S_d$$

# But this is too simplistic

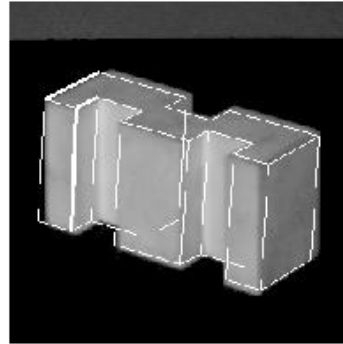
- The model specifies all the features of the object that may appear in the image.
- Some of them don't appear at all, due to occlusion or failures at low or mid level.
- Some of them are broken and not recognized.
- Some of them are distorted.
- Relationships don't all hold.

# TRIBORS: view class matching of polyhedral objects

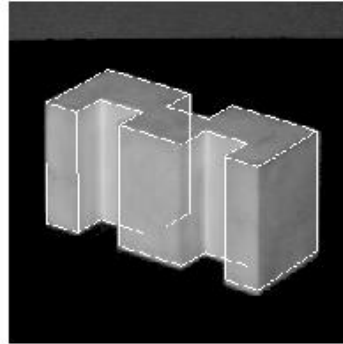
edges from image



model overlaid



improved location



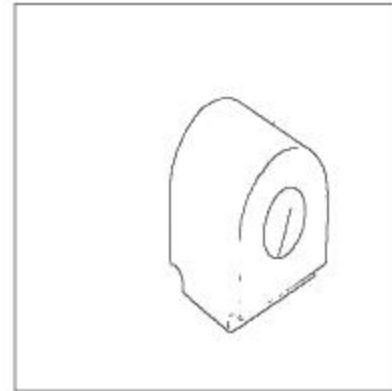
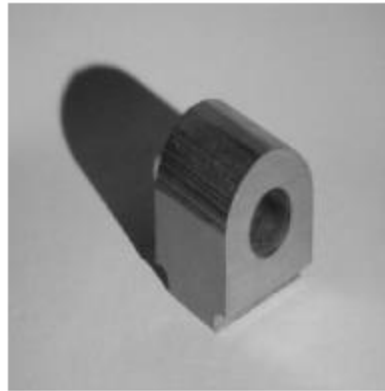
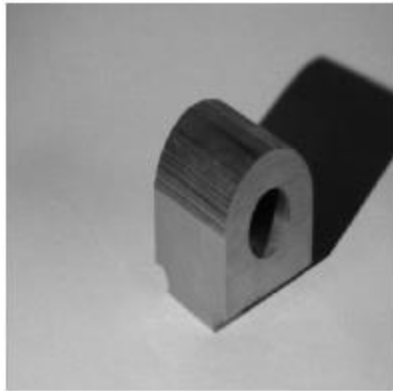
- A **view-class** is a typical 2D view of a 3D object.
- Each object had 4-5 view classes (hand selected).
- The representation of a view class for matching included:
  - **triplets of line segments** visible in that class
  - the **probability of detectability** of each triplet

The first version of this program used **iterative-deepening A\* search**.  
**STILL TOO MUCH OF A TOY PROBLEM.**



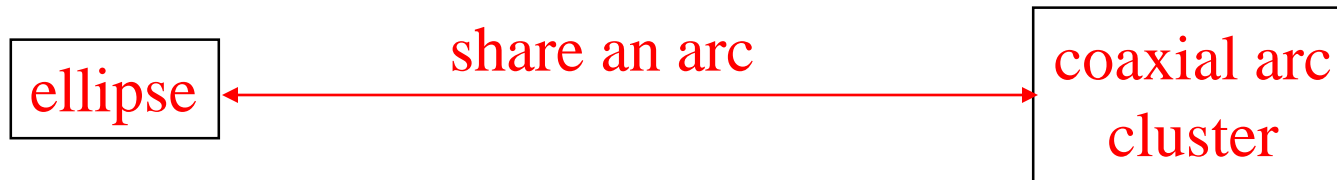
# RIO: Relational Indexing for Object Recognition

- RIO worked with more complex parts that could have
  - planar surfaces
  - cylindrical surfaces
  - threads

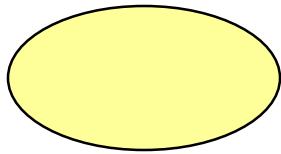


# Object Representation in RIO

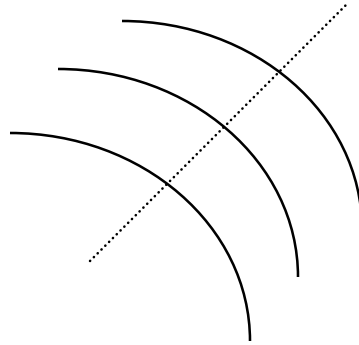
- 3D objects are represented by a **3D mesh** and set of **2D view classes**.
- Each **view class** is represented by an **attributed graph** whose nodes are features and whose attributed edges are relationships.
- For purposes of indexing, attributed graphs are stored as sets of **2-graphs**, graphs with 2 nodes and 2 relationships.



# RIO Features



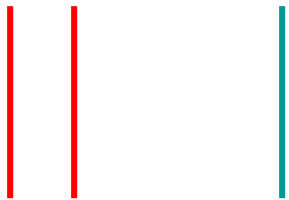
ellipses



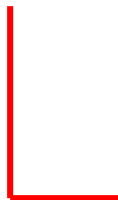
coaxials



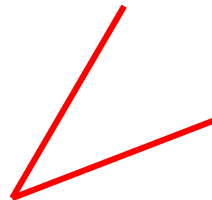
coaxials-multi



parallel lines  
close and far



L



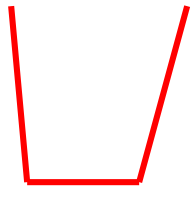
V



Y



Z



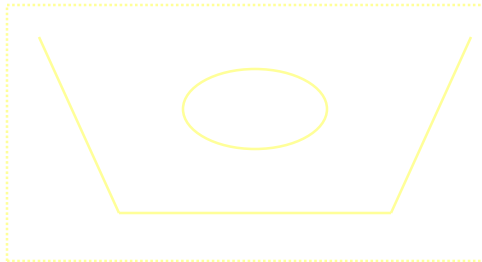
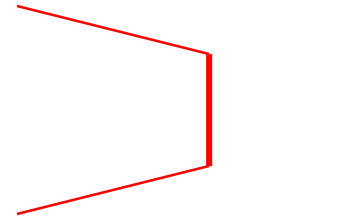
U

junctions

triples

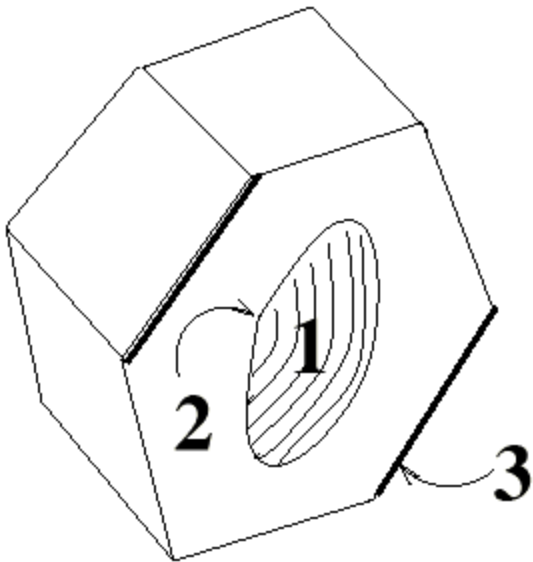
# RIO Relationships

- share one arc
- **share one line**
- share two lines
- coaxial
- close at extremal points
- bounding box encloses / enclosed by



# Hexnut Object

## MODEL-VIEW



### RELATIONS:

a: encloses

b: coaxial

### FEATURES:

1: coaxials-multi

2: ellipse

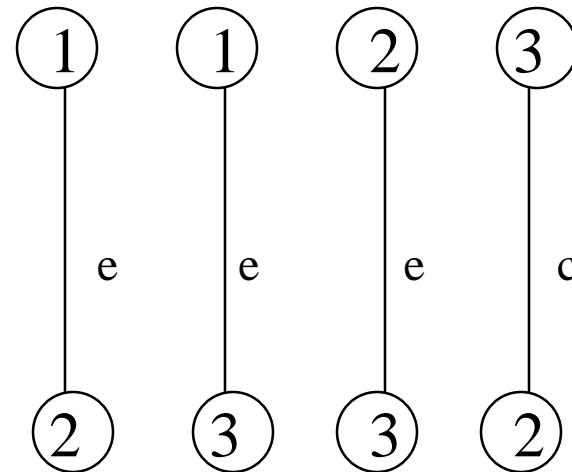
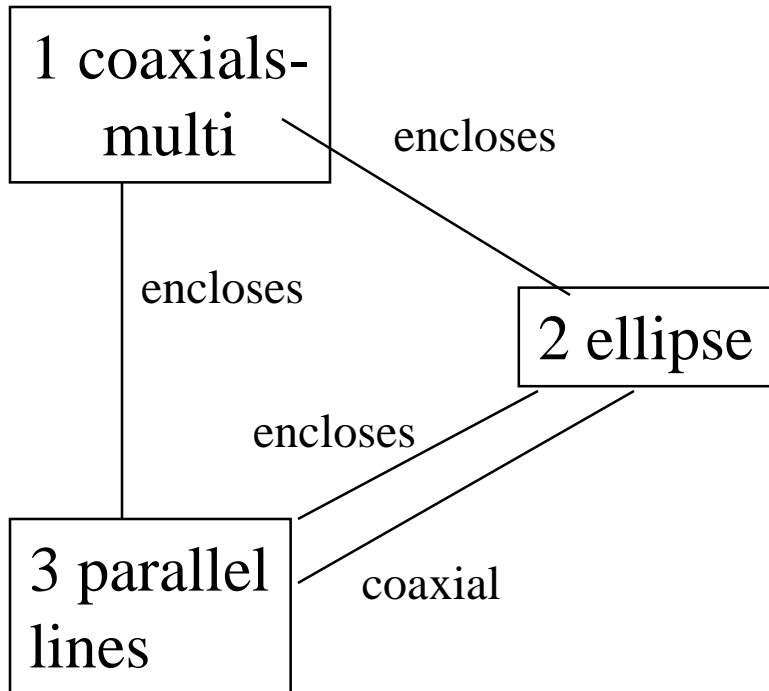
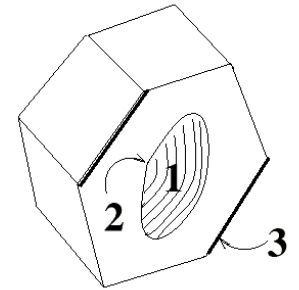
3: parallel lines

How are 1, 2, and 3 related?

What other features and relationships can you find?

# Graph and 2-Graph Representations

MODEL-VIEW



**RDF!**

# Relational Indexing for Recognition

## Preprocessing (off-line) Phase

for each model view  $M_i$  in the database

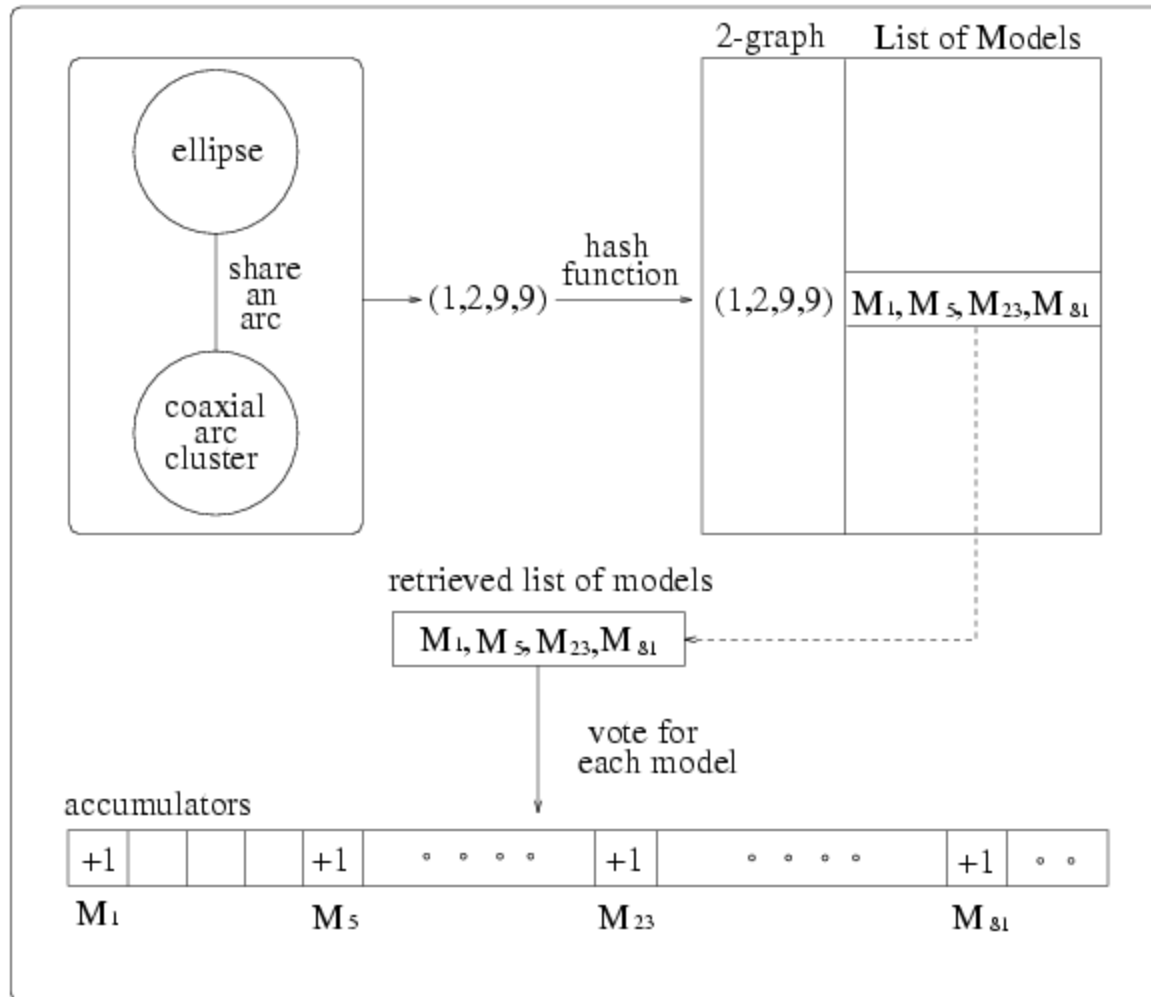
- **encode** each 2-graph of  $M_i$  to produce an index
- store  $M_i$  and associated information in the indexed bin of a hash table  $H$

# Matching (on-line) phase

1. Construct a relational (2-graph) **description**  $D$  for the scene
2. For each **2-graph**  $G$  of  $D$ 
  - encode it, producing an index to access the hash table  $H$
  - cast a vote for each  $M_i$  in the associated bin
3. Select the  $M_i$ 's with high votes as possible hypotheses
4. Verify or disprove via **alignment**, using the 3D meshes

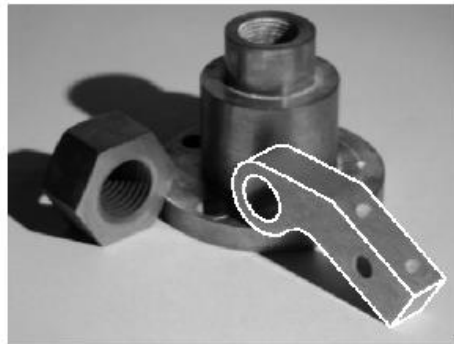
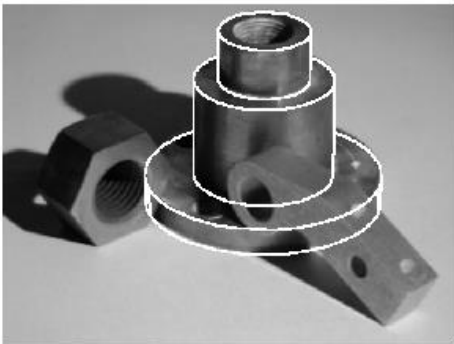
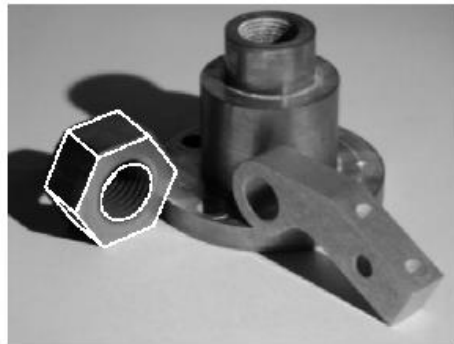
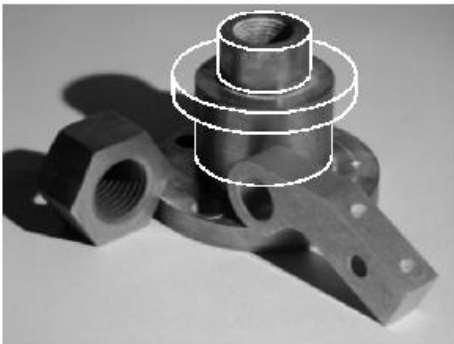


# The Voting Process



# RIO Verifications

incorrect  
hypothesis



1. The matched features of the hypothesized object are used to determine its **pose**.
2. The **3D mesh** of the object is used to project all its features onto the image.
3. A **verification procedure** checks how well the object features line up with edges on the image.

Use of classifiers is big in computer vision today.

- 2 Examples:
  - Rowley's Face Detection using neural nets
  - Yi's image classification using EM

# Object Detection: Rowley's Face Finder

1. convert to gray scale
2. normalize for lighting
3. histogram equalization
4. apply neural net(s)  
trained on 16K images



What data is fed to  
the classifier?

32 x 32 windows in  
a pyramid structure

# Object Class Recognition using Images of Abstract Regions

Yi Li, Jeff A. Bilmes, and Linda G. Shapiro  
Department of Computer Science and Engineering  
Department of Electrical Engineering  
University of Washington

# Problem Statement

**Given:** Some images and their corresponding descriptions



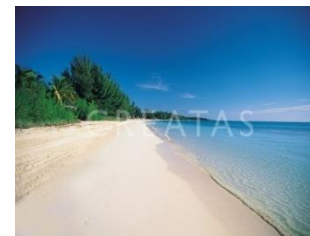
{trees, grass, cherry trees}



{cheetah, trunk}



{mountains, sky}



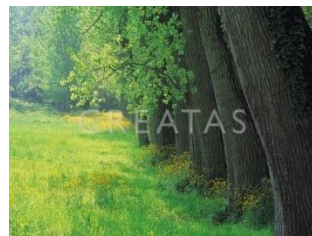
{beach, sky, trees, water}



**To solve:** What object classes are present in new images



?



?



?

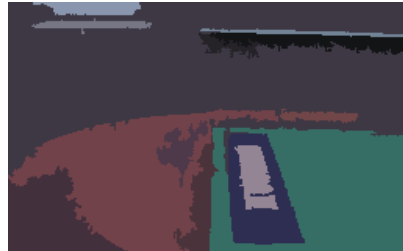


?

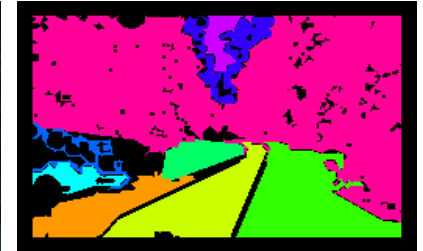


# Image Features for Object Recognition

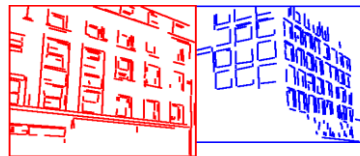
- Color



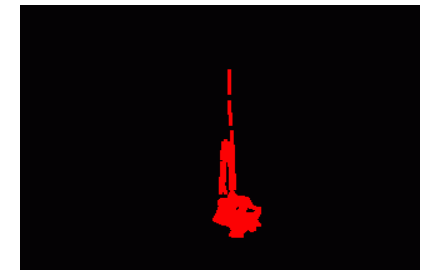
- Texture



- Structure



- Context



# Abstract Regions

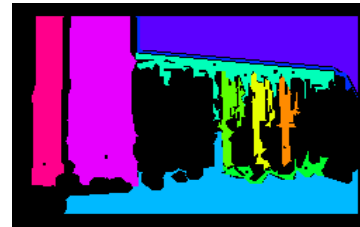
Original Images



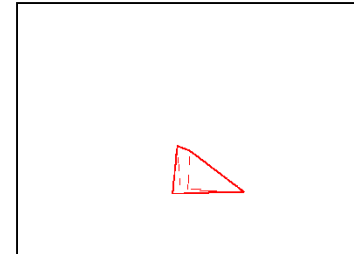
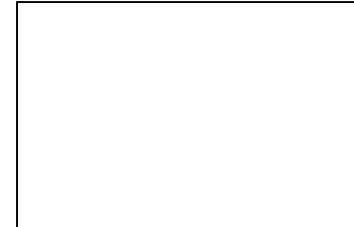
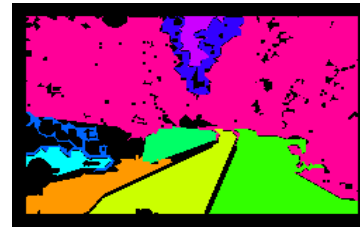
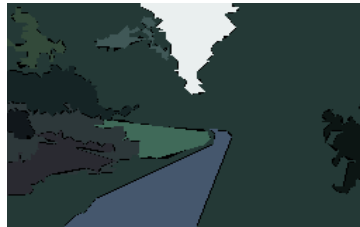
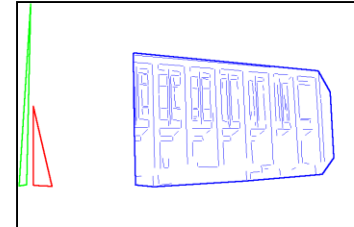
Color Regions



Texture Regions



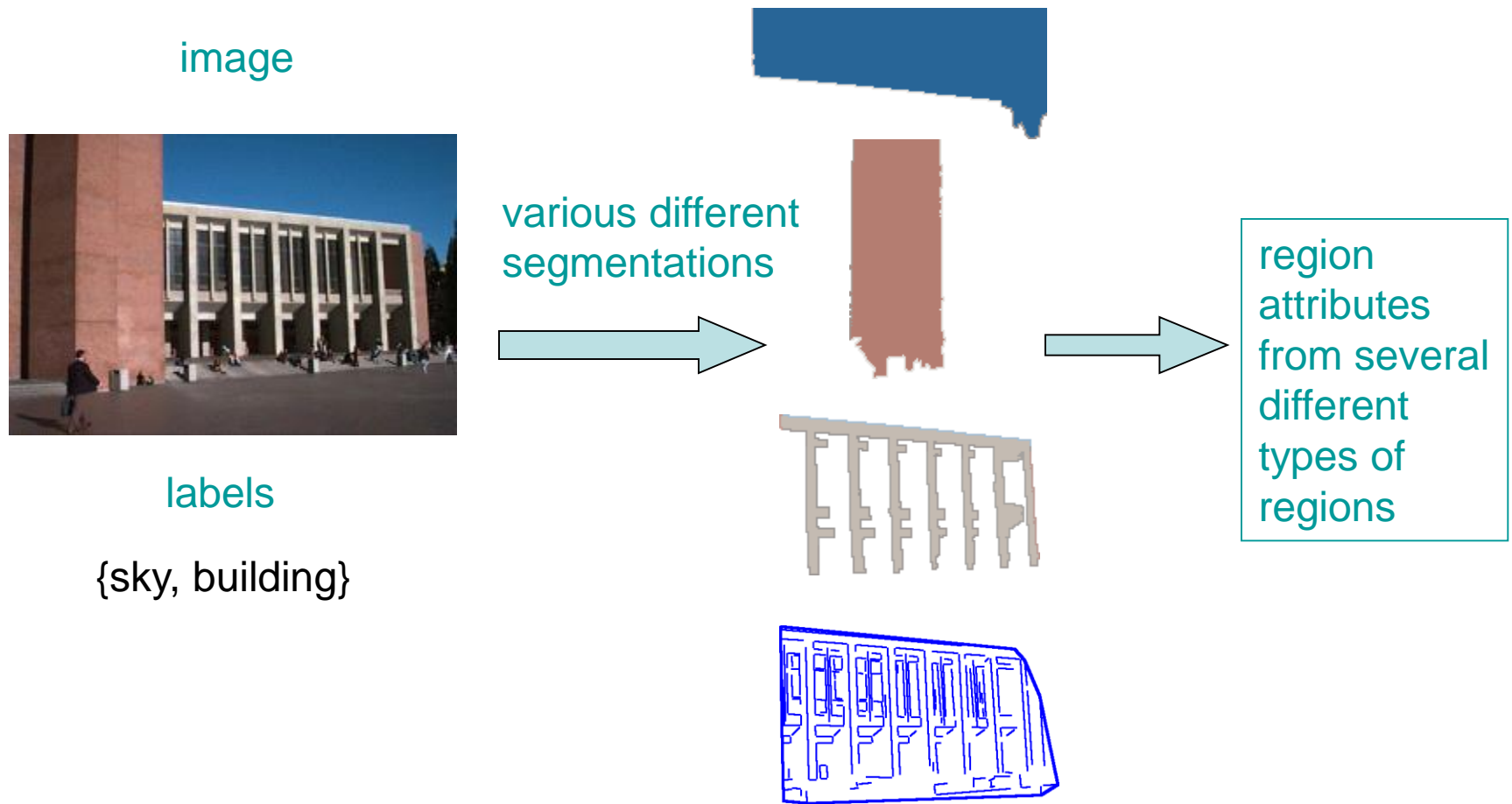
Line Clusters





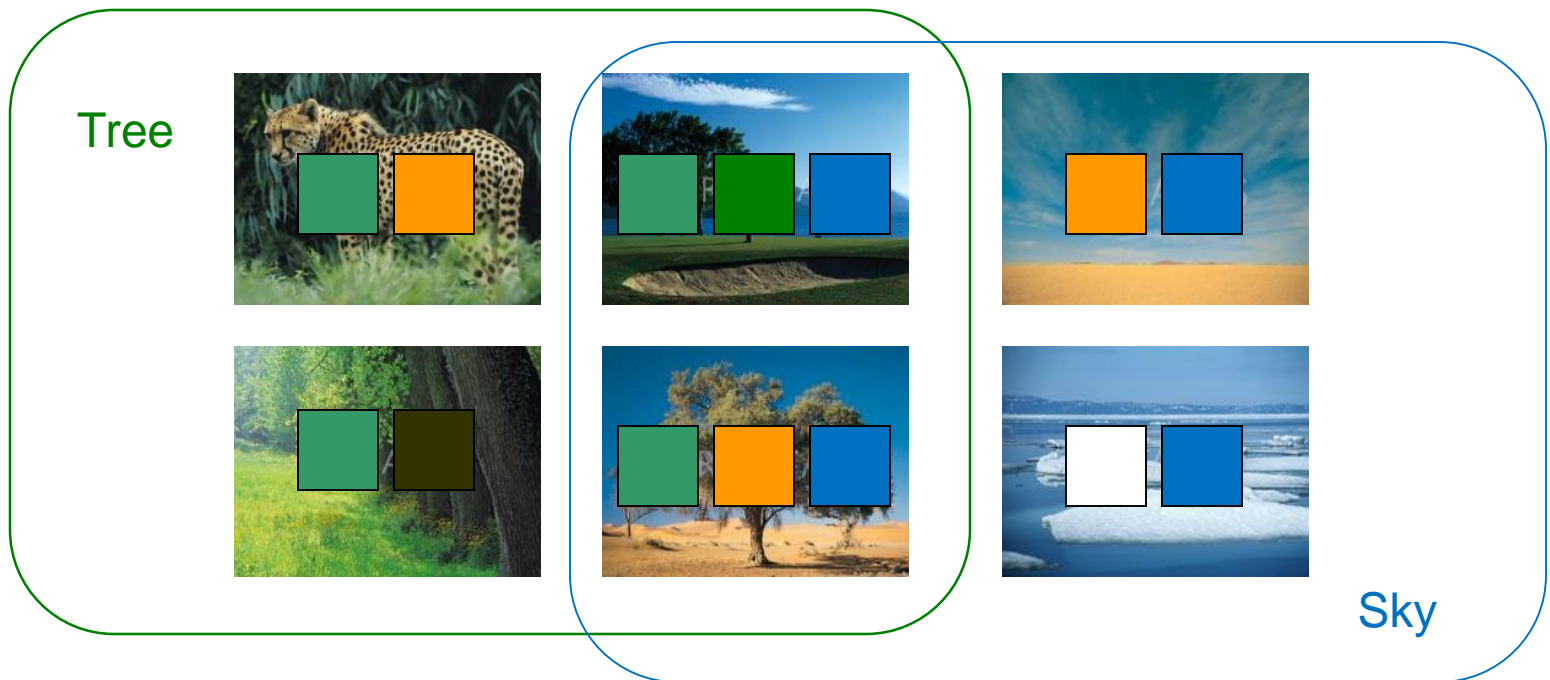
# Abstract Regions

Multiple segmentations whose regions are not labeled; a list of labels is provided for each training image.



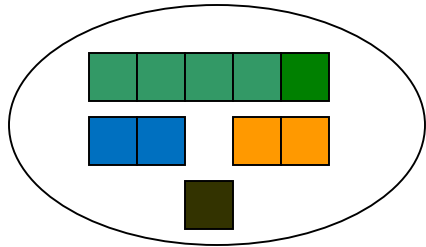
# Model Initial Estimation

- Estimate the initial model of an object using all the region features from all images that contain the object

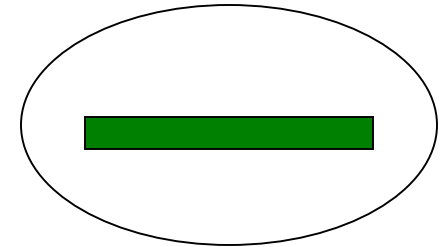


# EM Classifier: the Idea

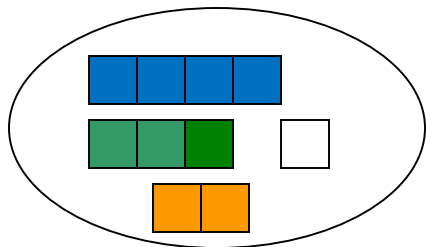
Initial Model for "trees"



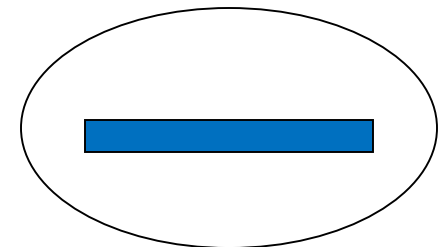
Final Model for "trees"



Initial Model for "sky"



Final Model for "sky"



EM

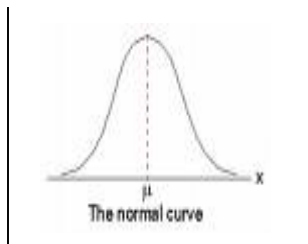


# EM Algorithm

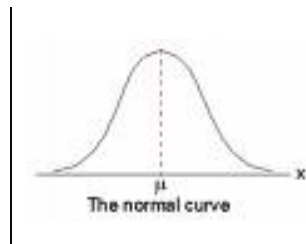
- Start with **K clusters**, each represented by a **probability distribution**
- Assuming a **Gaussian** or Normal distribution, each cluster is represented by its **mean and variance** (or covariance matrix) and has a weight.
- Go through the training data and soft-assign it to each cluster. Do this by **computing the probability that each training vector belongs to each cluster**.
- Using the results of the soft assignment, **recompute the parameters of each cluster**.
- Perform the last 2 steps iteratively.

# 1-D EM with Gaussian Distributions

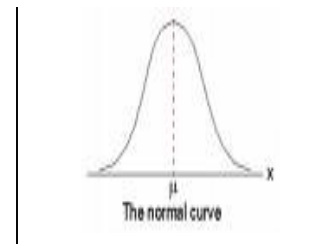
- Each cluster  $C_j$  is represented by a Gaussian distribution  $N(\mu_j, \sigma_j)$ .
- Initialization: For each cluster  $C_j$  initialize its mean  $\mu_j$ , variance  $\sigma_j$ , and weight  $\alpha_j$ .



$$N(\mu_1, \sigma_1)$$
$$\alpha_1 = P(C_1)$$



$$N(\mu_2, \sigma_2)$$
$$\alpha_2 = P(C_2)$$



$$N(\mu_3, \sigma_3)$$
$$\alpha_3 = P(C_3)$$

- With no other knowledge, use random means and variances and equal weights.

# Standard EM to EM Classifier

- That's the standard EM algorithm.
- For n-dimensional data, the variance becomes a co-variance matrix, which changes the formulas slightly.
- But **we used an EM variant to produce a classifier.**
- The next slide indicates the differences between what we used and the standard.

# EM Classifier

1. **Fixed Gaussian components** (one Gaussian per object class) and **fixed weights** corresponding to the frequencies of the corresponding objects in the training data.
2. **Customized initialization** uses only the training images that contain a particular object class to initialize its Gaussian.
3. **Controlled expectation step** ensures that a feature vector only contributes to the Gaussian components representing objects present in its training image.
4. **Extra background component** absorbs noise.

Gaussian for  
trees

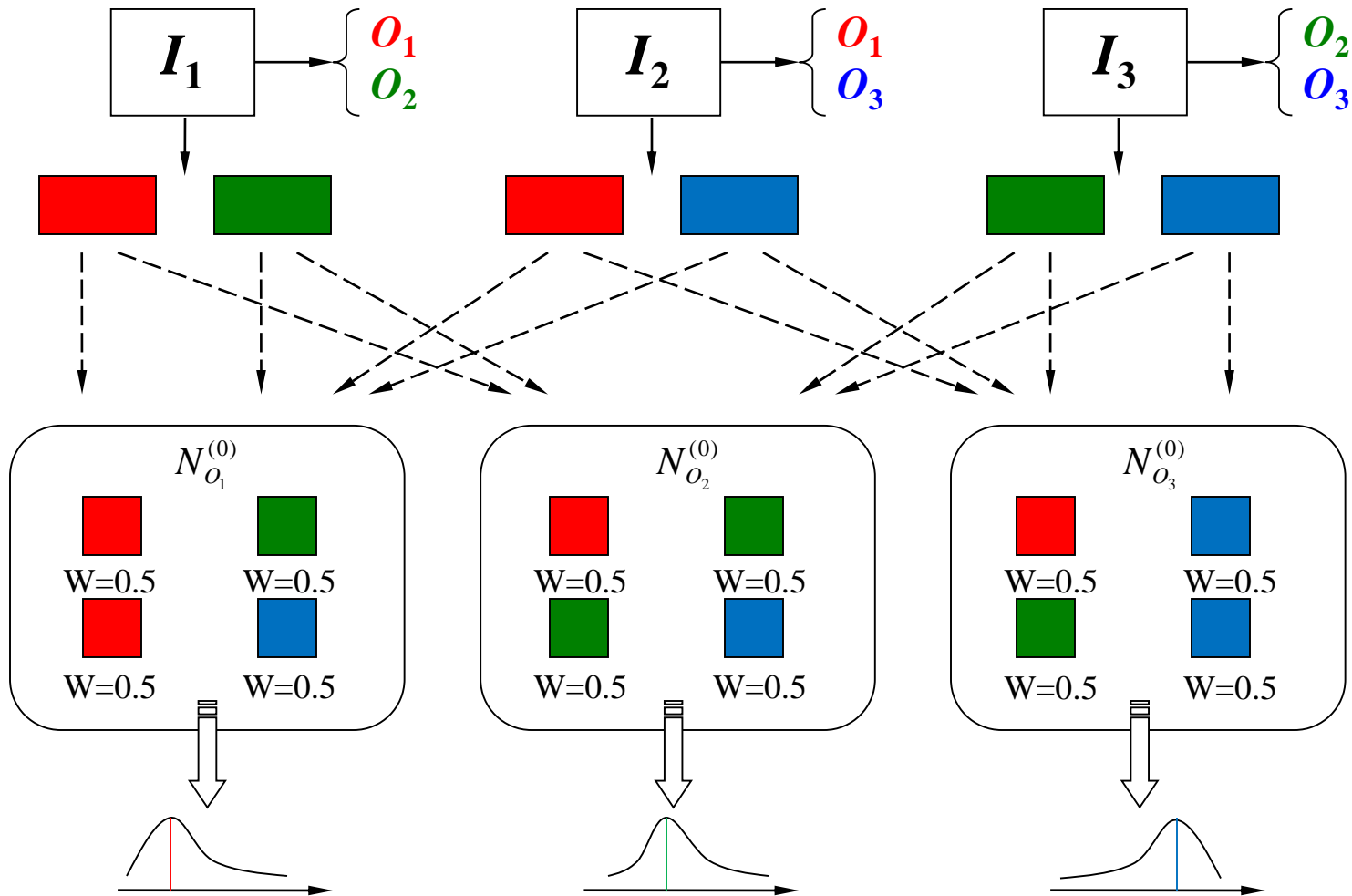
Gaussian for  
buildings

Gaussian for  
sky

Gaussian for  
background

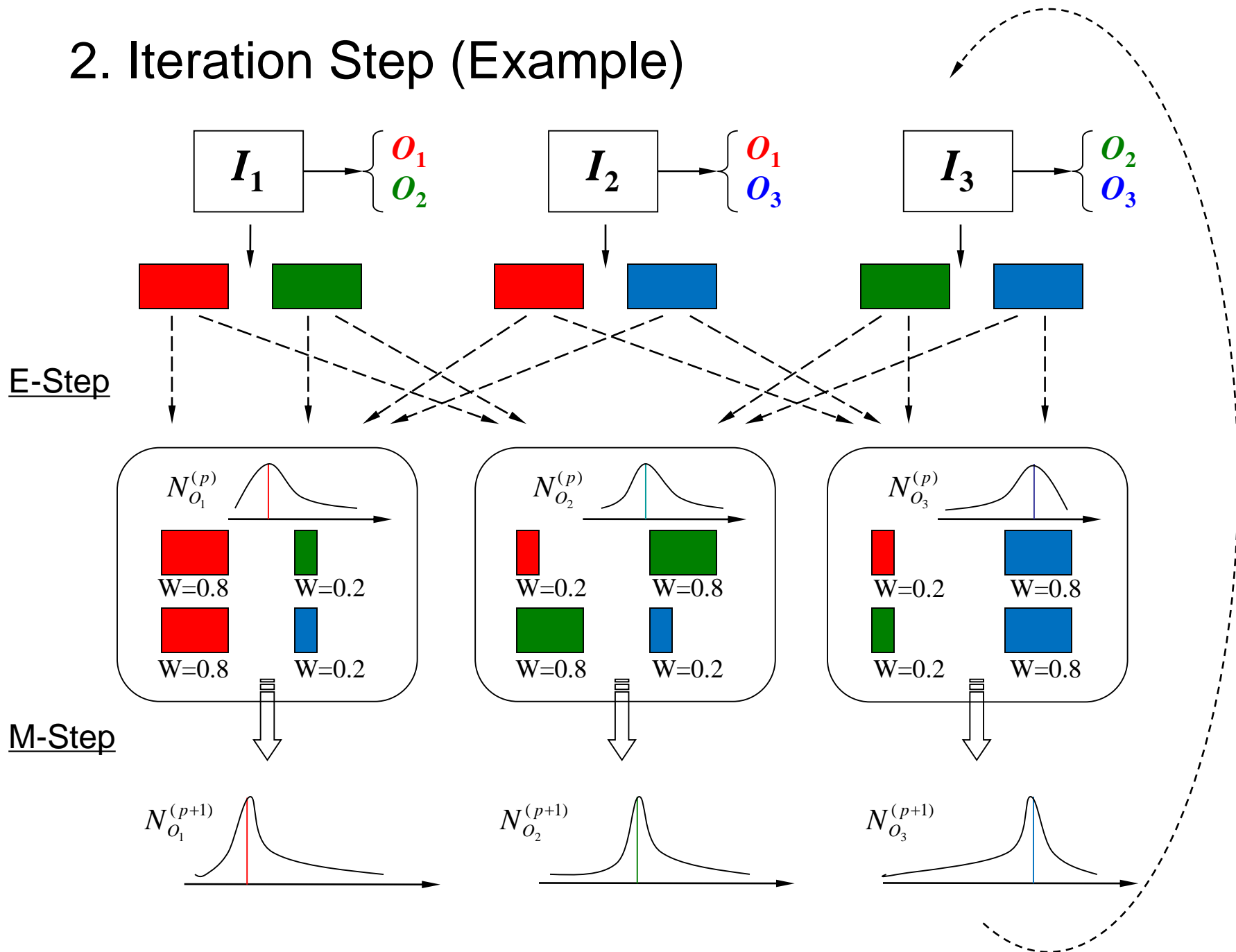
# 1. Initialization Step (Example)

Image & description

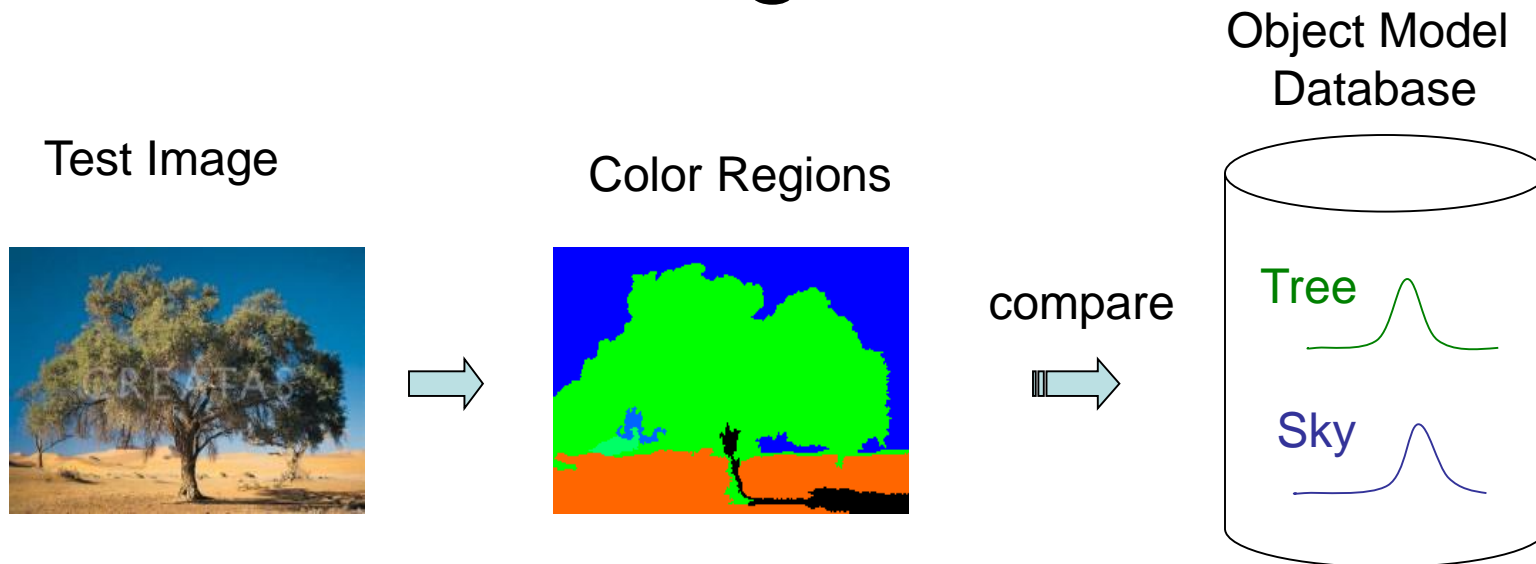




## 2. Iteration Step (Example)



# Recognition



How do you decide if a particular object is in an image?

To calculate  $p(\text{tree} / \text{image})$

$$p(\text{tree} / \text{image}) = f \left( \begin{array}{l} p(\text{tree} / \text{blue}) \\ p(\text{tree} / \text{green}) \\ p(\text{tree} / \text{orange}) \\ p(\text{tree} / \text{black}) \end{array} \right)$$

$f$  is a function that combines probabilities from all the color regions in the image.

e.g. max or mean

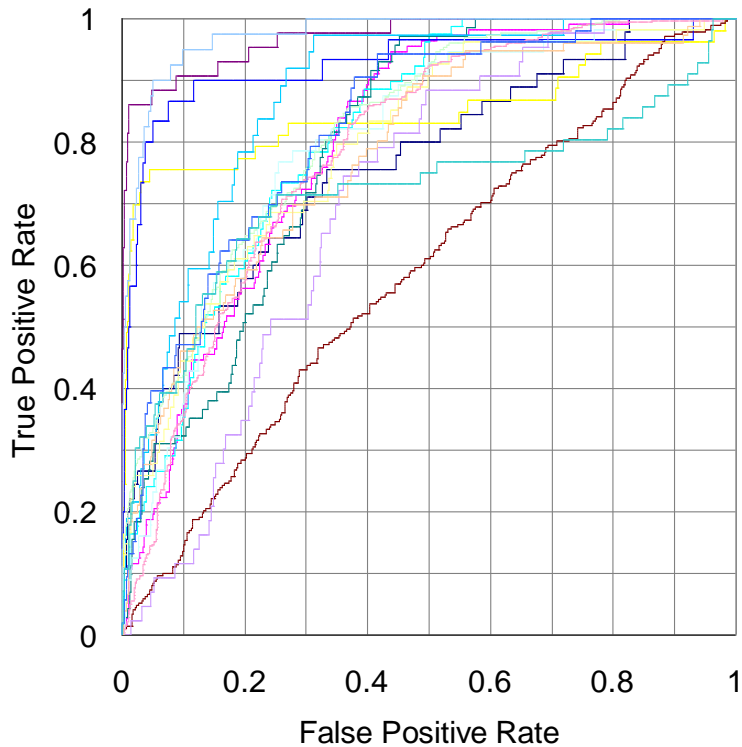
# Combining different types of abstract regions: First Try

- Treat the different types of regions **independently** and combine at the time of classification.
- $P(\text{object} | a_1, a_2, \dots, a_n) = P(\text{object} | a_1) \dots P(\text{object} | a_n)$
- Form **intersections** of the different types of regions, creating smaller regions that have both color and texture properties for classification.

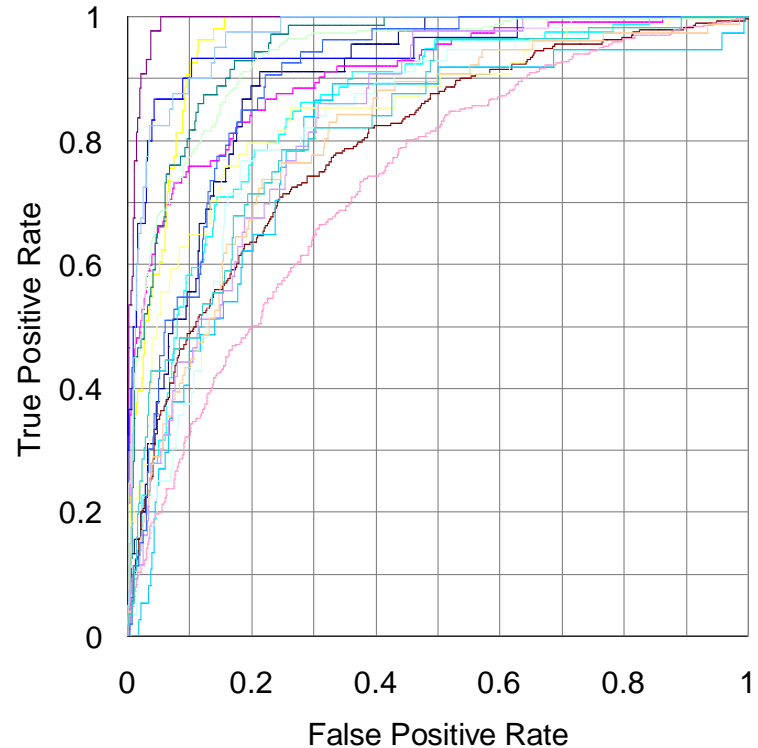
# Experiments (on 860 images)

- 18 keywords: mountains (30), orangutan (37), track (40), tree trunk (43), football field (43), beach (45), prairie grass (53), cherry tree (53), snow (54), zebra (56), polar bear (56), lion (71), water (76), chimpanzee (79), cheetah (112), sky (259), grass (272), tree (361).
- A set of cross-validation experiments (80% as training set and the other 20% as test set)
- The poorest results are on object classes “tree,” “grass,” and “water,” each of which has a high variance; a single Gaussian model is insufficient.

# ROC Charts: True Positive vs. False Positive



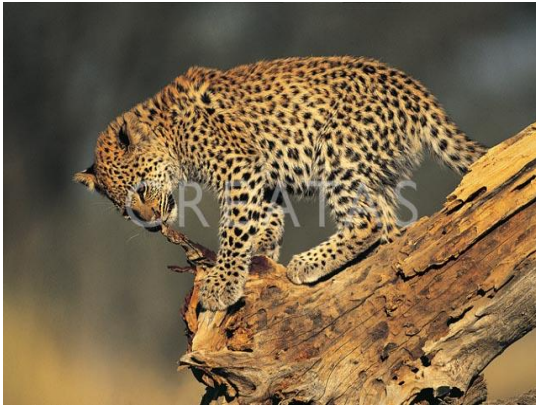
Independent Treatment of  
Color and Texture



Using Intersections of  
Color and Texture Regions

# Sample Retrieval Results

cheetah





# Sample Results (Cont.)

grass



# Sample Results (Cont.)

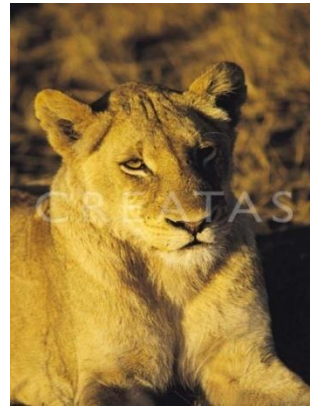
cherry tree





# Sample Results (Cont.)

lion



# Summary

- Designed a set of abstract region features: **color**, **texture**, **structure**, . . .
- Developed a new **semi-supervised EM-like algorithm** to recognize object classes in color photographic images of outdoor scenes; tested on 860 images.
- Compared **two different methods of combining** different types of abstract regions. The intersection method had a higher performance

# Weakness of the EM Classifier Approach

- It did not generalize well to multiple features
- It assumed that object classes could be modeled as Gaussians

# Second Approach

A Generative Discriminative Learning  
Algorithm for Image Classification

Yi Li, Linda Shapiro, Jeff Bilmes

ICCV 2005

# A Better Approach to Combining Different Feature Types

## Phase 1: JUST CLUSTERING in features space

- Treat each type of abstract region separately
- For abstract region type  $a$  and for object class  $o$ , use the EM algorithm to construct **clusters** that are **multivariate Gaussians** over the features for type  $a$  regions.

# Consider only abstract region type color (**c**) and object class object (**o**)

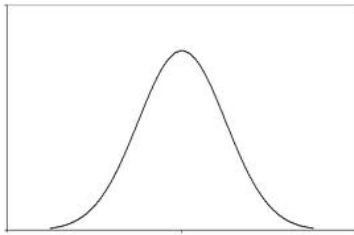
- At the end of Phase 1, we can compute the distribution of color feature vectors in an image containing object *o*.

$$P(X^c|o) = \sum_{m=1}^{M^c} w_m^c \cdot N(X^c; \mu_m^c, \Sigma_m^c)$$

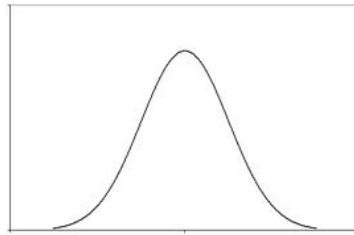
- $M^c$  is the number of components (clusters).
- The  $w$ 's are the weights ( $\alpha$ 's) of the components.
- The  $\mu$ 's and  $\Sigma$ 's are the parameters of the components.
- $N(X^c, \mu_m^c, \Sigma_m^c)$  specifies the probability that  $X^c$  belongs to a particular normal distribution.

# Color Components for Class o

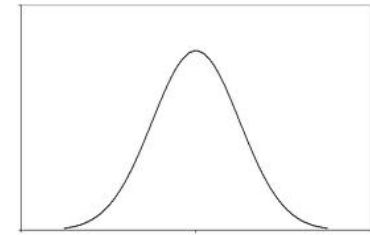
$$P(X^c|o) = \sum_{m=1}^{M^c} w_m^c \cdot N(X^c; \mu_m^c, \Sigma_m^c)$$



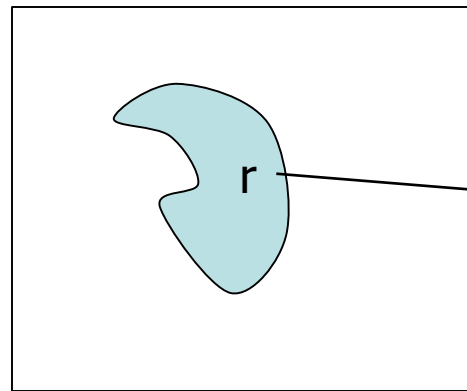
component 1  
 $\mu_1, \Sigma_1, w_1$



component 2  
 $\mu_2, \Sigma_2, w_2$



component  $M^c$   
 $\mu_M, \Sigma_M, w_M$



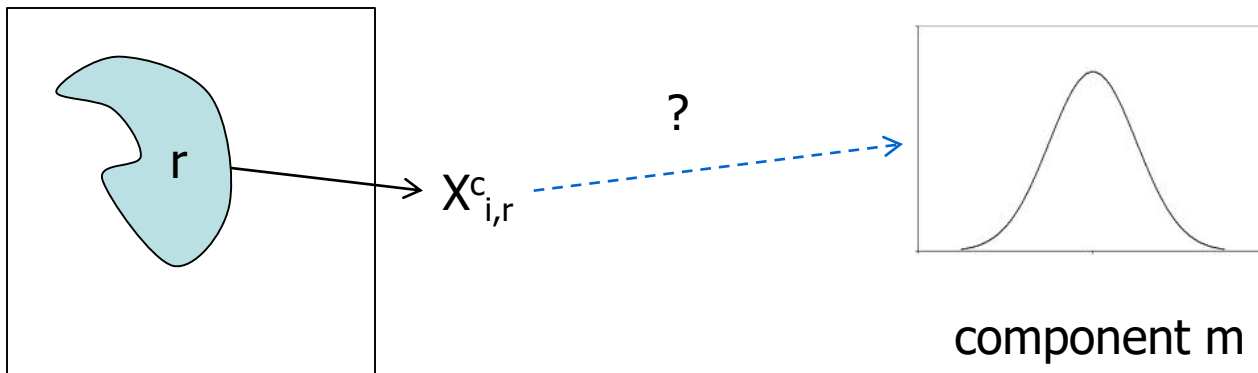
color feature vector  
 $X^c$  for region  $r$

Now we can determine which components are likely to be present in an image.

- The probability that the feature vector  $X$  from color region  $r$  of image  $I_i$  comes from component  $m$  is given by

$$P(X_{i,r}^c, m^c) = w_m^c \cdot N(X_{i,r}^c, \mu_m^c, \Sigma_m^c)$$

$$f_{\mathbf{x}}(x_1, \dots, x_k) = \frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$



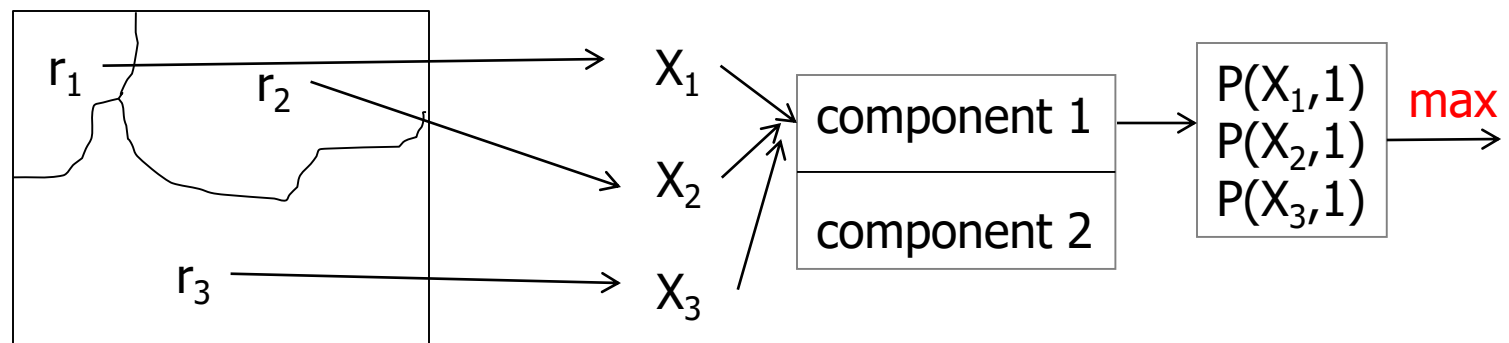


And determine the probability that the whole image is related to component  $m$  as a function of the feature vectors of all its regions.

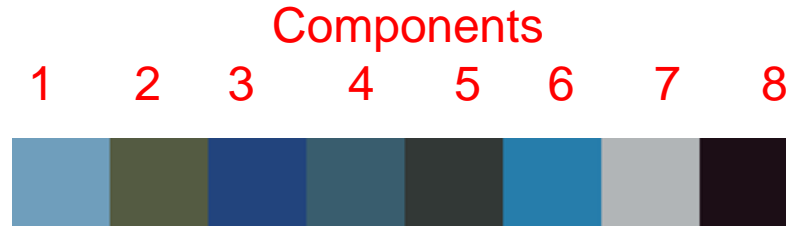
- Then the probability that image  $I_i$  has a region that comes from component  $m$  is

$$P(I_i, m^c) = f(\{P(X_{i,r}^c, m^c) | r = 1, 2, \dots\})$$

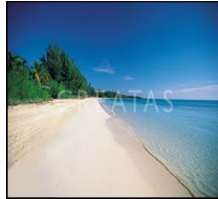
- where  $f$  is an aggregate function such as **mean** or **max**



# Aggregate Scores for Color

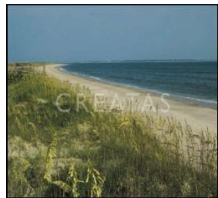


beach



.93	.16	.94	.24	.10	.99	.32	.00
-----	-----	-----	-----	-----	-----	-----	-----

beach



.66	.80	.00	.72	.19	.01	.22	.02
-----	-----	-----	-----	-----	-----	-----	-----

not  
beach



.43	.03	.00	.00	.00	.00	.15	.00
-----	-----	-----	-----	-----	-----	-----	-----

We now use **positive** and **negative** training images, calculate for each the probabilities of regions of each component, and form a **training matrix**.

$$\begin{array}{l} I_1^+ \\ I_2^+ \\ \vdots \\ I_1^- \\ I_2^- \\ \vdots \end{array} \left[ \begin{array}{cccc} P(I_1^+, 1^c) & P(I_1^+, 2^c) & \cdots & P(I_1^+, M^c) \\ P(I_2^+, 1^c) & P(I_2^+, 2^c) & \cdots & P(I_2^+, M^c) \\ \vdots & \vdots & & \\ P(I_1^-, 1^c) & P(I_1^-, 2^c) & \cdots & P(I_1^-, M^c) \\ P(I_2^-, 1^c) & P(I_2^-, 2^c) & \cdots & P(I_2^-, M^c) \\ \vdots & \vdots & & \end{array} \right]$$

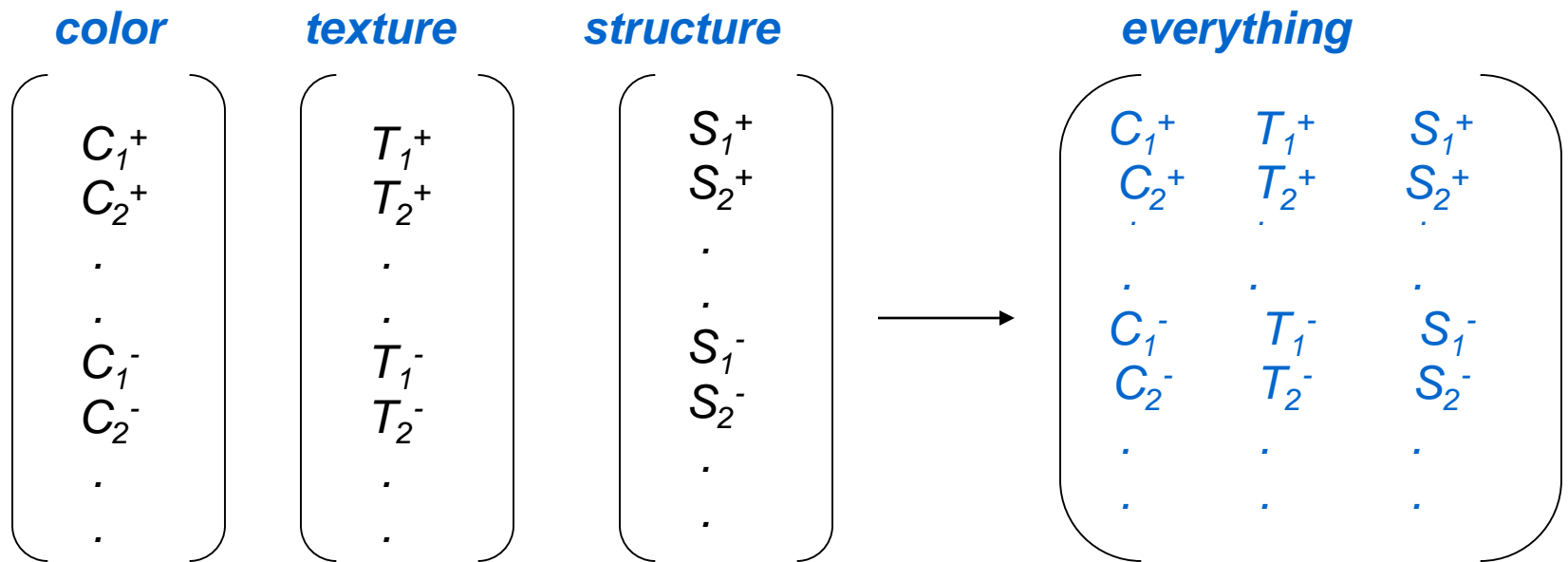
# Phase 2 Learning

- Let  $C_i$  be row  $i$  of the training matrix.
- Each such row is a feature vector for the color features of regions of image  $I_i$  that relates them to the Phase 1 components.
- Now we can use a second-stage classifier to learn  $P(o/I_i)$  for each object class  $o$  and image  $I_i$ .

# Multiple Feature Case

- We calculate separate Gaussian mixture models for each different features type:
- Color:  $C_i$
- Texture:  $T_i$
- Structure:  $S_i$
- and any more features we have (motion).

Now we concatenate the matrix rows from the different region types to obtain a **multi-feature-type training matrix** and train a neural net classifier to classify images.



# ICPR04 Data Set with General Labels

	EM-variant with single Gaussian per object	EM-variant extension to mixture models	Gen/Dis with Classical EM clustering	Gen/Dis with EM-variant extension
<i>African animal</i>	71.8%	85.7%	89.2%	90.5%
<i>arctic</i>	80.0%	79.8%	90.0%	85.1%
<i>beach</i>	88.0%	90.8%	89.6%	91.1%
<i>grass</i>	76.9%	69.6%	75.4%	77.8%
<i>mountain</i>	94.0%	96.6%	97.5%	93.5%
<i>primate</i>	74.7%	86.9%	91.1%	90.9%
<i>sky</i>	91.9%	84.9%	93.0%	93.1%
<i>stadium</i>	95.2%	98.9%	99.9%	100.0%
<i>tree</i>	70.7%	79.0%	87.4%	88.2%
<i>water</i>	82.9%	82.3%	83.1%	82.4%
<b>MEAN</b>	<b>82.6%</b>	<b>85.4%</b>	<b>89.6%</b>	<b>89.3%</b>

# Comparison to ALIP: the Benchmark Image Set

- Test database used in SIMPLicity paper and ALIP paper.
- 10 classes (*African people, beach, buildings, buses, dinosaurs, elephants, flowers, food, horses, mountains*). 100 images each.

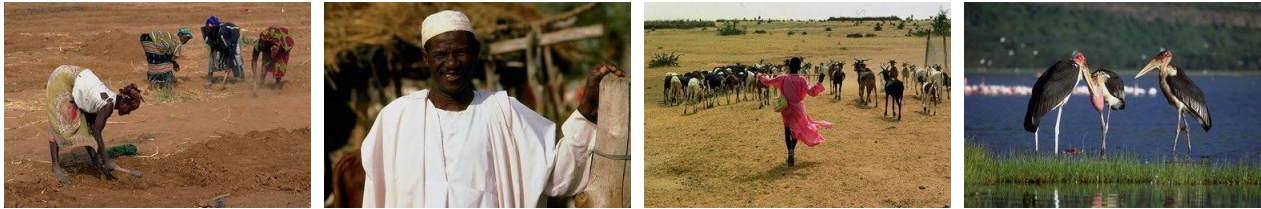


# Comparison to ALIP: the Benchmark Image Set

	ALIP	cs	ts	st	ts+st	cs+st	cs+ts	cs+ts+st
<i>African</i>	52	69	23	26	35	79	72	74
<i>beach</i>	32	44	38	39	51	48	59	64
<i>buildings</i>	64	43	40	41	67	70	70	78
<i>buses</i>	46	60	72	92	86	85	84	95
<i>dinosaurs</i>	100	88	70	37	86	89	94	93
<i>elephants</i>	40	53	8	27	38	64	64	69
<i>flowers</i>	90	85	52	33	78	87	86	91
<i>food</i>	68	63	49	41	66	77	84	85
<i>horses</i>	60	94	41	50	64	92	93	89
<i>mountains</i>	84	43	33	26	43	63	55	65
<b>MEAN</b>	<b>63.6</b>	<b>64.2</b>	<b>42.6</b>	<b>41.2</b>	<b>61.4</b>	<b>75.4</b>	<b>76.1</b>	<b>80.3</b>

# Comparison to ALIP: the 60K Image Set

## 0. Africa, people, landscape, animal



## 1. autumn, tree, landscape, lake



## 2. Bhutan, Asia, people, landscape, church



# Comparison to ALIP: the 60K Image Set

3. California, sea, beach, ocean, flower



4. Canada, sea, boat, house, flower, ocean



5. Canada, west, mountain, landscape, cloud, snow, lake



# Comparison to ALIP: the 60K Image Set

Number of top-ranked categories required	1	2	3	4	5
ALIP	11.88	17.06	20.76	23.24	26.05
Gen/Dis	11.56	17.65	21.99	25.06	27.75

The table shows the percentage of test images whose true categories were included in the top-ranked categories.

# Groundtruth Data Set

- UW Ground truth database (1224 images)
- 31 elementary object categories: *river* (30), *beach* (31), *bridge* (33), *track* (35), *pole* (38), *football field* (41), *frozen lake* (42), *lantern* (42), *husky stadium* (44), *hill* (49), *cherry tree* (54), *car* (60), *boat* (67), *stone* (70), *ground* (81), *flower* (85), *lake* (86), *sidewalk* (88), *street* (96), *snow* (98), *cloud* (119), *rock* (122), *house* (175), *bush* (178), *mountain* (231), *water* (290), *building* (316), *grass* (322), *people* (344), *tree* (589), *sky* (659)
- 20 high-level concepts: *Asian city*, *Australia*, *Barcelona*, *campus*, *Cannon Beach*, *Columbia Gorge*, *European city*, *Geneva*, *Green Lake*, *Greenland*, *Indonesia*, *indoor*, *Iran*, *Italy*, *Japan*, *park*, *San Juans*, *spring flowers*, *Swiss mountains*, and *Yellowstone*.





*beach, sky, tree, water*



*people, street, tree*



*building, grass, people,  
sidewalk, sky, tree*



*building, bush, sky,  
tree, water*



*flower, house, people,  
pole, sidewalk, sky*



*flower, grass, house,  
pole, sky, street, tree*



*building, flower, sky,  
tree, water*



*boat, rock, sky,  
tree, water*



*building, car, people, tree*



*car, people, sky*



*boat, house, water*



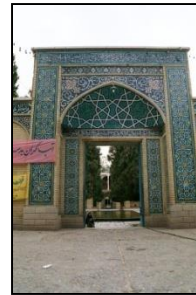
*building*

# Groundtruth Data Set: ROC Scores

<i>street</i>	60.4	<i>tree</i>	80.8	<i>stone</i>	87.1	<i>columbia gorge</i>	94.5
<i>people</i>	68.0	<i>bush</i>	81.0	<i>hill</i>	87.4	<i>green lake</i>	94.9
<i>rock</i>	73.5	<i>flower</i>	81.1	<i>mountain</i>	88.3	<i>italy</i>	95.1
<i>sky</i>	74.1	<i>iran</i>	82.2	<i>beach</i>	89.0	<i>swiss moutains</i>	95.7
<i>ground</i>	74.3	<i>bridge</i>	82.7	<i>snow</i>	92.0	<i>sanjuans</i>	96.5
<i>river</i>	74.7	<i>car</i>	82.9	<i>lake</i>	92.8	<i>cherry tree</i>	96.9
<i>grass</i>	74.9	<i>pole</i>	83.3	<i>frozen lake</i>	92.8	<i>indoor</i>	97.0
<i>building</i>	75.4	<i>yellowstone</i>	83.7	<i>japan</i>	92.9	<i>greenland</i>	98.7
<i>cloud</i>	75.4	<i>water</i>	83.9	<i>campus</i>	92.9	<i>cannon beach</i>	99.2
<i>boat</i>	76.8	<i>indonesia</i>	84.3	<i>barcelona</i>	92.9	<i>track</i>	99.6
<i>lantern</i>	78.1	<i>sidewalk</i>	85.7	<i>geneva</i>	93.3	<i>football field</i>	99.8
<i>australia</i>	79.7	<i>asian city</i>	86.7	<i>park</i>	94.0	<i>husky stadium</i>	100.0
<i>house</i>	80.1	<i>european city</i>	87.0	<i>spring flowers</i>	94.4		

# Groundtruth Data Set: Top Results

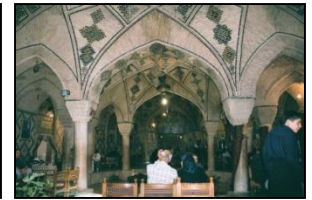
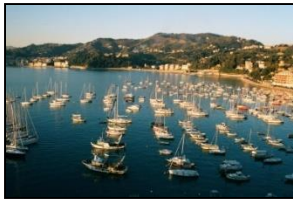
*Asian city*



*Cannon beach*



*Italy*



*park*





# Groundtruth Data Set: Top Results

*sky*



*spring flowers*



*tree*



*water*



# Groundtruth Data Set: Annotation Samples



**tree**(97.3), **bush**(91.6),  
**spring flowers**(90.3),  
**flower**(84.4),  
park(84.3),  
**sidewalk**(67.5),  
**grass**(52.5), **pole**(34.1)



**sky**(99.8),  
**Columbia gorge**(98.8),  
lantern(94.2), **street**(89.2),  
house(85.8), bridge(80.8),  
car(80.5), hill(78.3),  
boat(73.1), pole(72.3),  
**water**(64.3), mountain(63.8),  
**building**(9.5)



sky(95.1), **Iran**(89.3),  
house(88.6),  
**building**(80.1),  
boat(71.7), bridge(67.0),  
**water**(13.5), **tree**(7.7)



**Italy**(99.9), grass(98.5),  
**sky**(93.8), rock(88.8),  
**boat**(80.1), **water**(77.1),  
Iran(64.2), stone(63.9),  
bridge(59.6), **European**(56.3),  
sidewalk(51.1), **house**(5.3)

# Comments

- The generative/discriminative approach, using EM clustering to produce feature vectors, followed by a neural net classifier, was much more powerful.
- It is strongly related to the bag-of-words approach.
- Instead of histograms of words, it is using vectors of responses to Gaussians as feature vectors.