Segmentation / Classification

CS 600.361/600.461

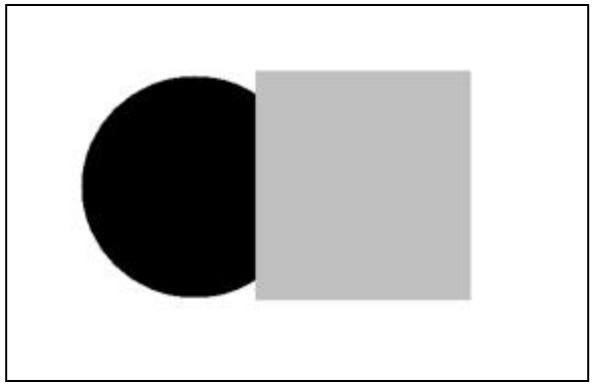
Instructor: Greg Hager
Slides by Nicolas Padoy

(Adapted from slides by Rick Szeliski and Kristen Grauman)

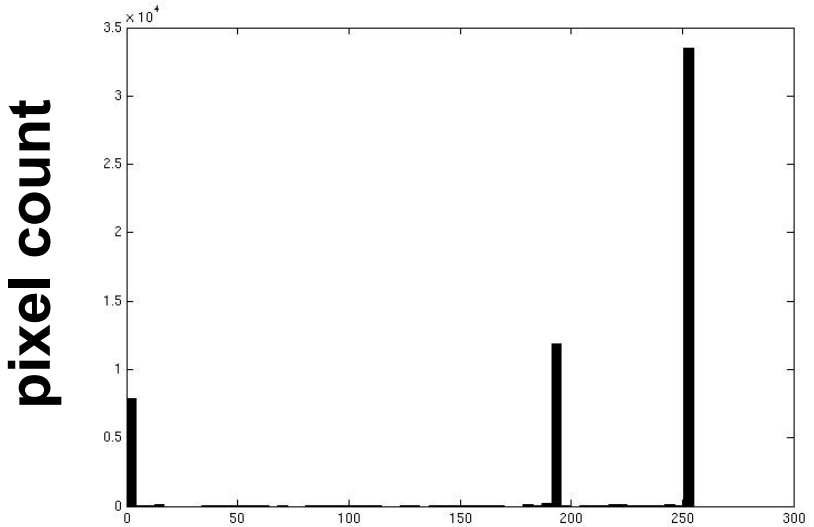
Outline

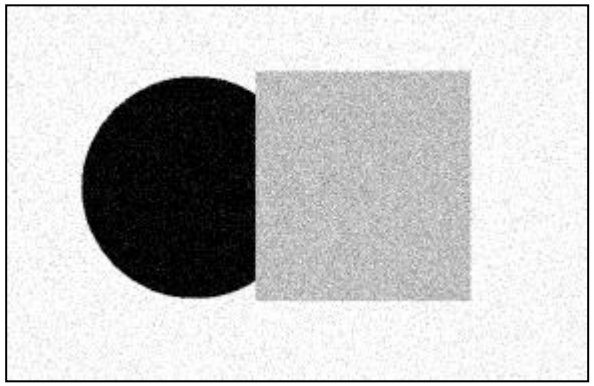
- Reminders
- More on bottom-up segmentation via clustering
  - Mixture of Gaussians
  - Mean-shift
  - Graph-based approaches
- Classification
  - K-nearest-neighbor

Clustering-based Approaches to Segmentation

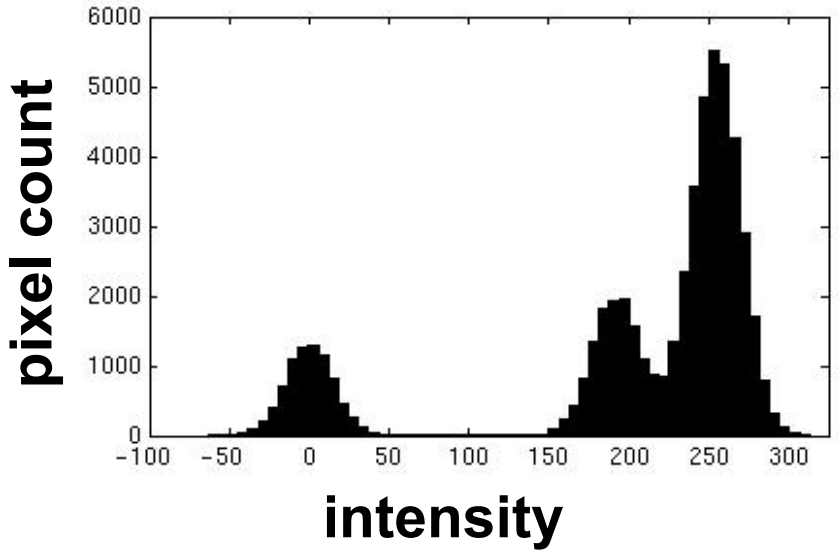


input image





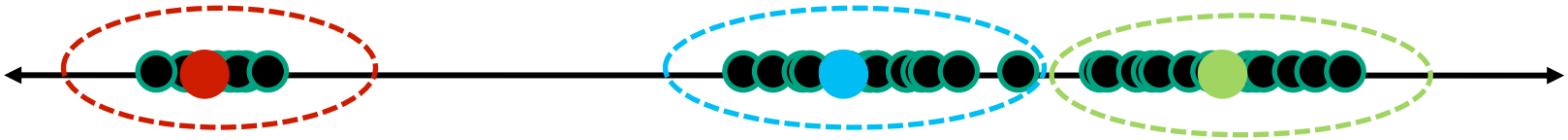
input image



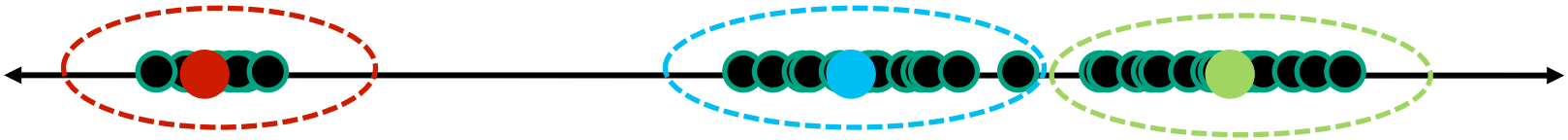
Clustering

With this objective, it is a "chicken and egg" problem:

- If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.



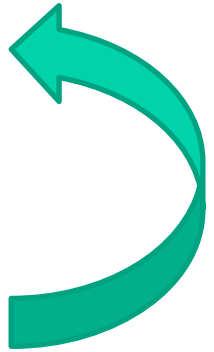
- If we knew the **group memberships**, we could get the centers by computing the mean per group.



K-means clustering

Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.

1. Randomly initialize the cluster centers, c_1, ..., c_K
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2



Properties

- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

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

Source: Steve Seitz

K-means: pros and cons

Pros

Simple, fast to compute

Converges to local minimum of within-cluster squared error

Cons/issues

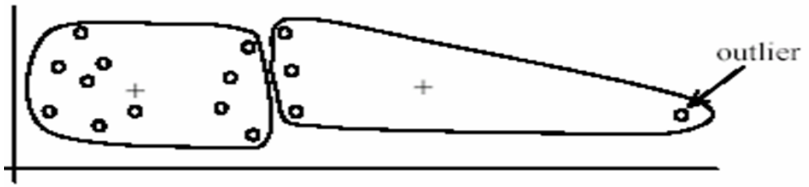
Setting k?

Sensitive to initial centers

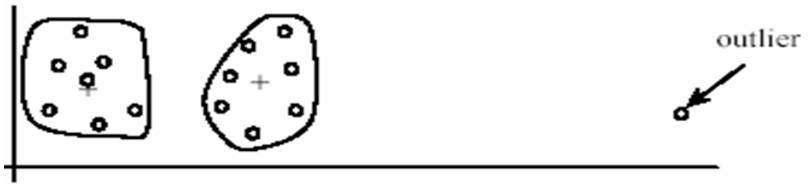
Sensitive to outliers

Detects spherical clusters

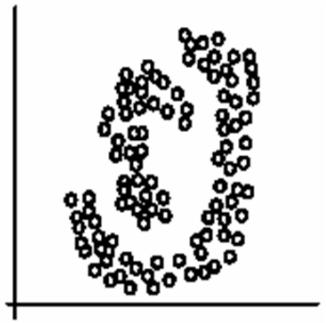
Assuming means can be computed



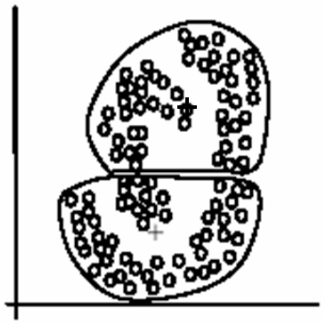
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters

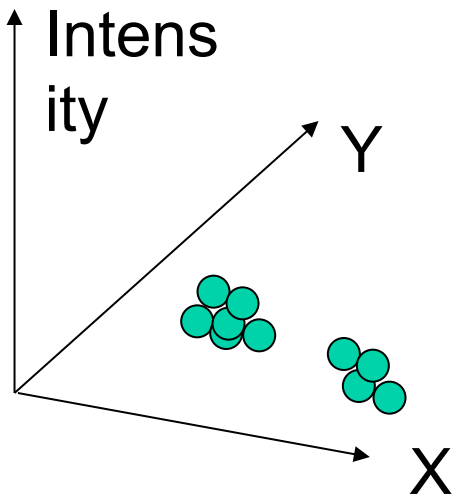


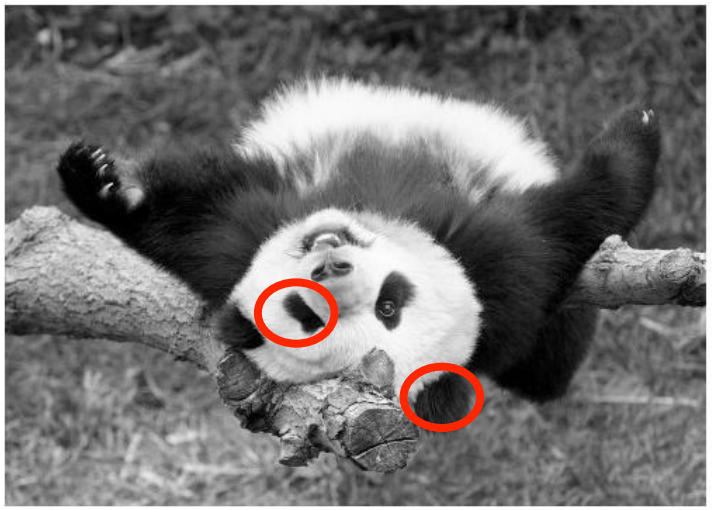
(B): *k*-means clusters

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity+position** similarity

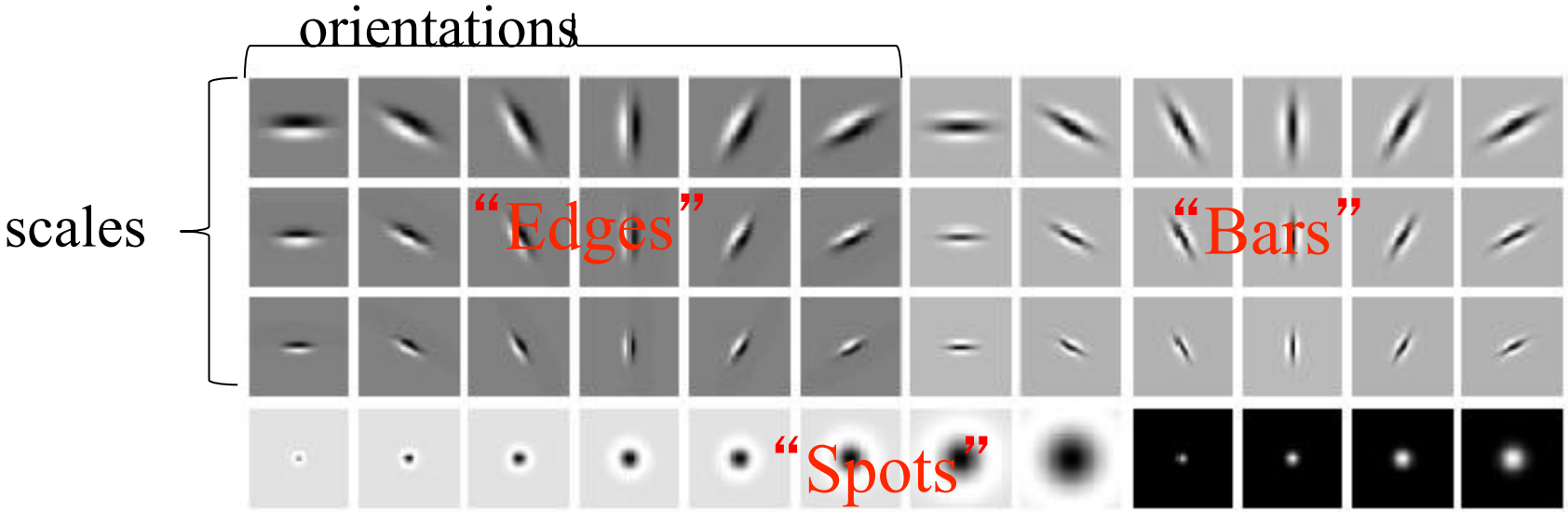




Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

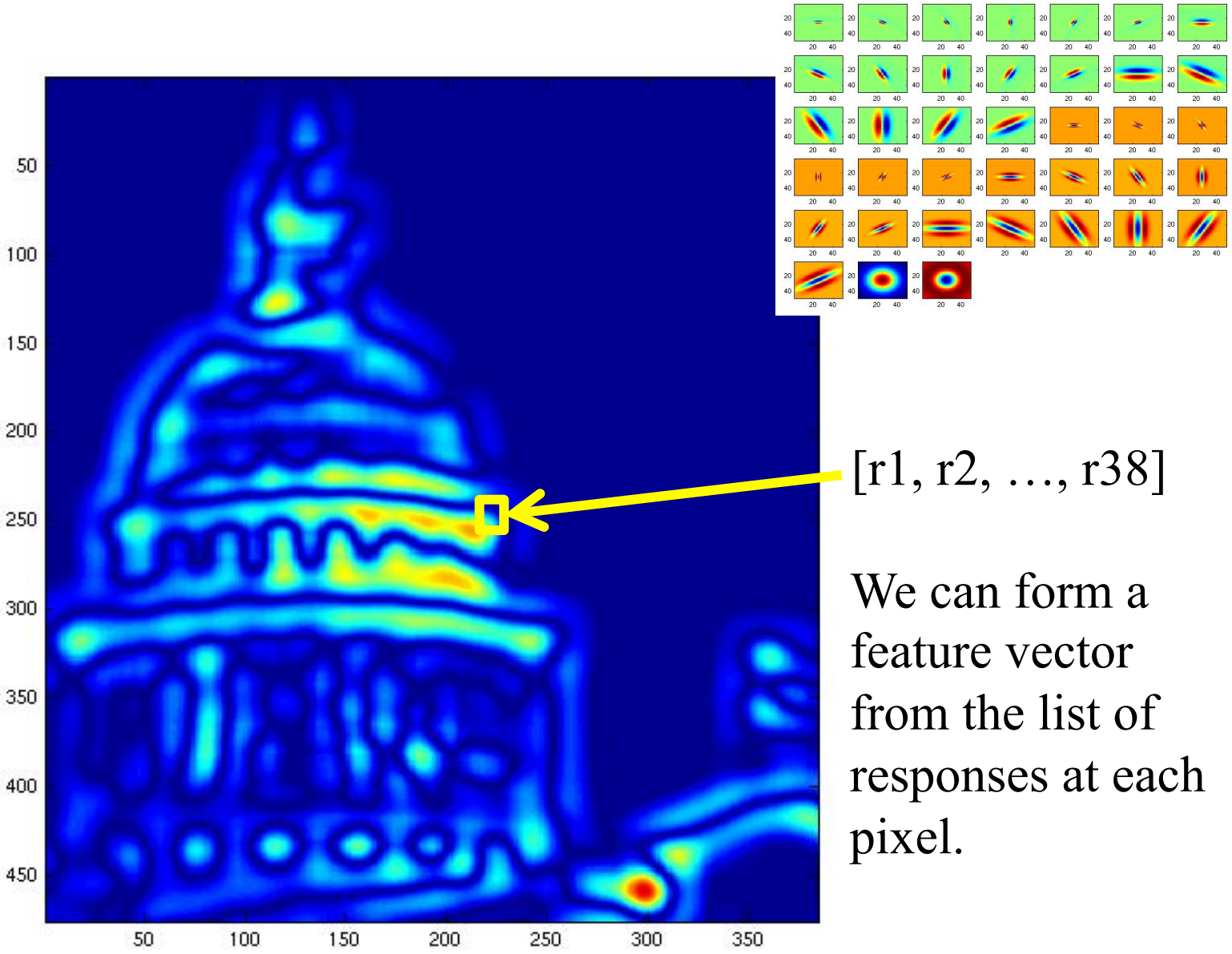
Image credit: D. Forsyth

Filter banks



What filters to put in the bank?

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



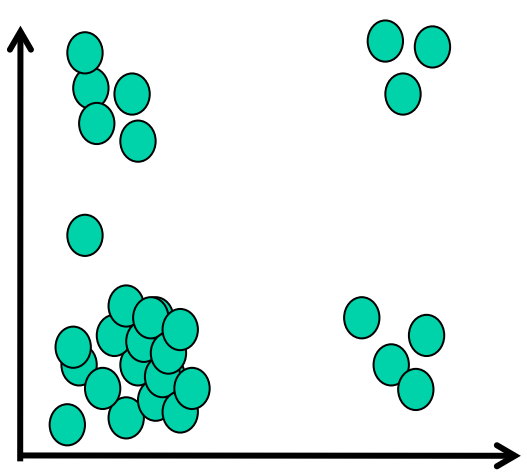
[r1, r2, …, r38]

We can form a feature vector from the list of responses at each pixel.

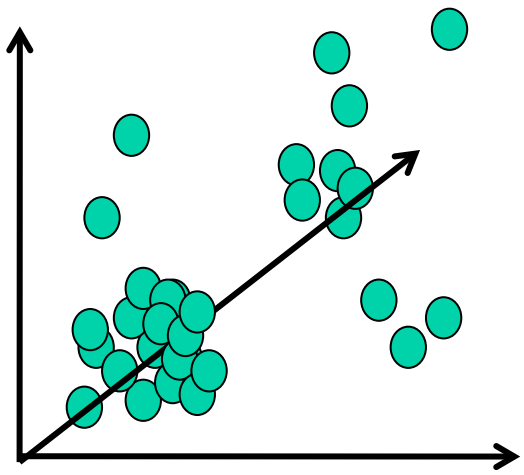
d-dimensional features

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

Euclidean distance (L_2)



2d



3d

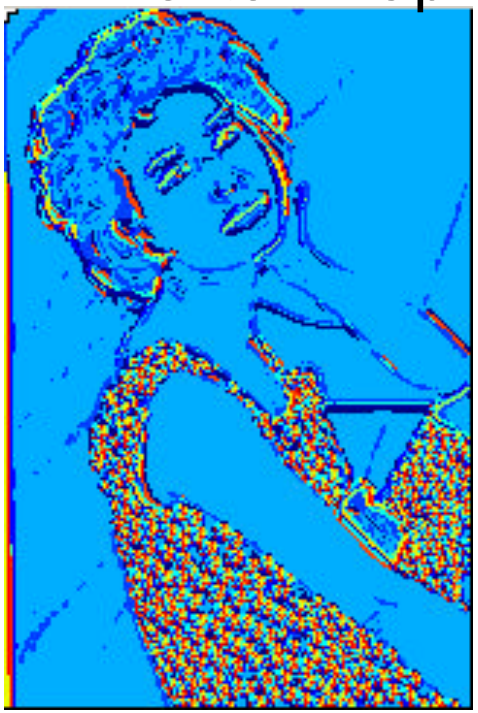
. . .

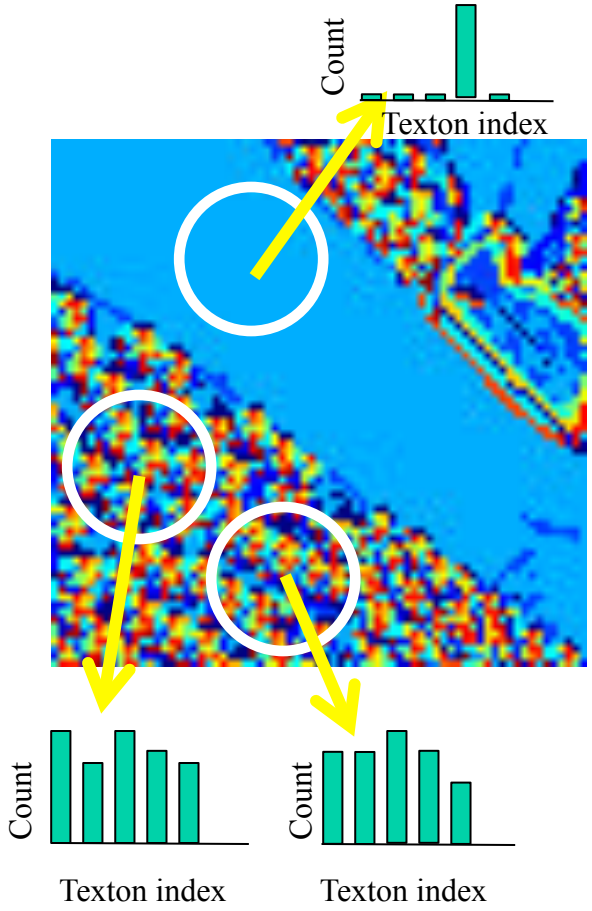
Segmentation with texture features

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

Image

Texton map





Malik, Belongie, Leung and Shi. IJCV 2001.

Adapted from Lana Lazebnik

Mixtures of Gaussians

Probabilistic clustering

Basic questions

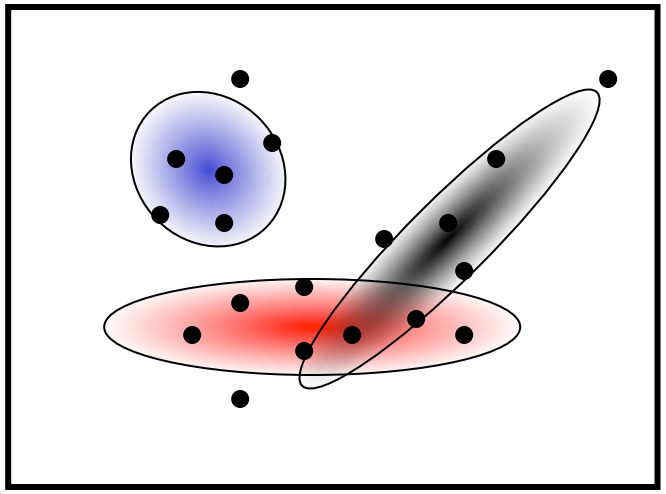
- what's the probability that a point **x** is in cluster m?
- what's the shape of each cluster?

K-means doesn't answer these questions

Basic idea

- instead of treating the data as a bunch of points, assume that they are all generated by sampling a continuous function
- This function is called a **generative model**
  - defined by a vector of parameters **θ**

Mixture of Gaussians



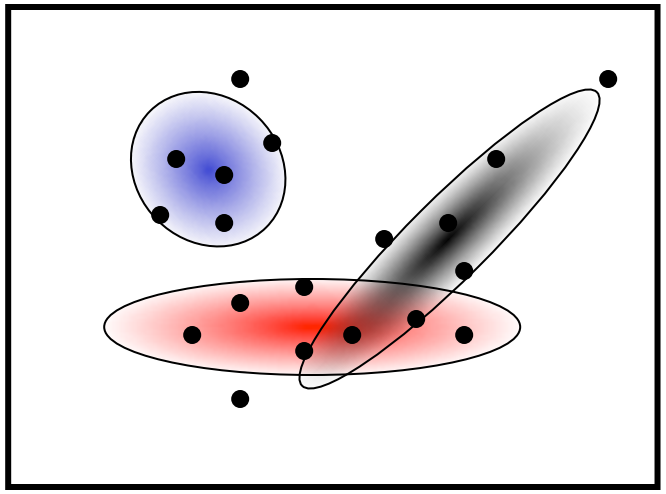
One generative model is a mixture of Gaussians (MOG)

- K Gaussian blobs with means $\boldsymbol{\mu_b}$ covariance matrices $\mathbf{V_b}$, dimension d
  - blob *b* defined by: $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
- blob *b* is selected with probability
- the likelihood of observing **x** is a weighted mixture of Gaussians

$$P(x|\theta) = \sum_{b=1}^{K} \alpha_b P(x|\theta_b)$$

- where $\theta = [\mu_1, \ldots, \mu_n, V_1, \ldots, V_n]$

Expectation maximization (EM)



Goal

- find blob parameters θ that maximize the likelihood function:

$$P(data|\theta) = \prod_x P(x|\theta)$$

Approach:

1. E step: given current guess of blobs, compute ownership of each point
2. M step: given ownership probabilities, update blobs to maximize likelihood function
3. repeat until convergence

EM details

E-step

- compute probability that point **x** is in blob b, given current guess of **θ**

$$P(b|x,\mu_b,V_b) = \frac{\alpha_b P(x|\mu_b,V_b)}{\sum_{i=1}^{K}\alpha_i P(x|\mu_i,V_i)}$$

M-step

- compute probability that blob b is selected

$$\alpha_b^{new} = \frac{1}{N}\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)$$

N data points

- mean of blob b

$$\mu_b^{new} = \frac{\sum_{i=1}^{N} x_i P(b|x_i,\mu_b,V_b)}{\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)}$$

- covariance of blob b

$$V_b^{new} = \frac{\sum_{i=1}^{N}(x_i-\mu_b^{new})(x_i-\mu_b^{new})^T P(b|x_i,\mu_b,V_b)}{\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)}$$

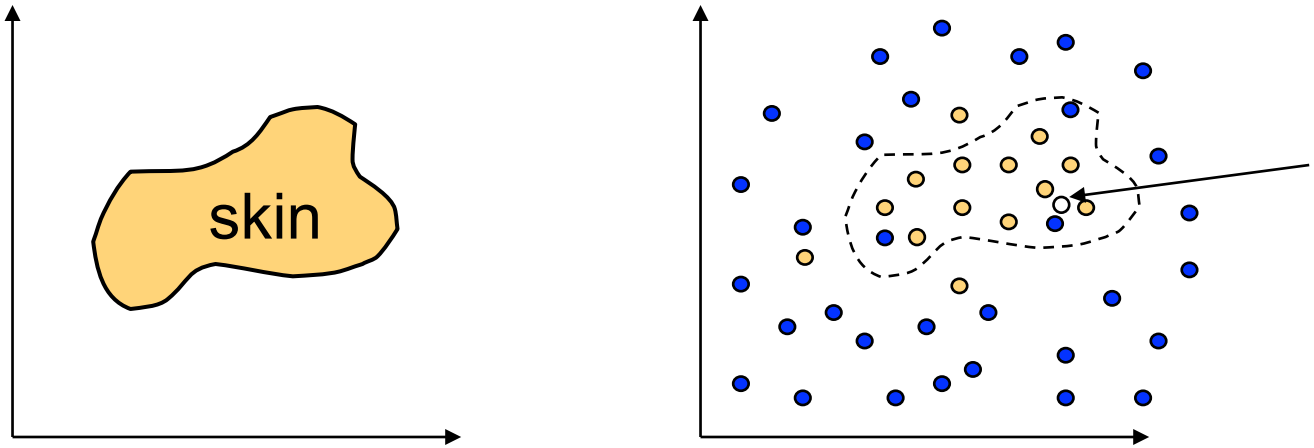
EM demo

http://lcn.epfl.ch/tutorial/english/gaussian/html/index.html

Applications of EM

Turns out this is useful for all sorts of problems

- any clustering problem
- any model estimation problem
- finding outliers
- segmentation problems
  - segmentation based on color
  - segmentation based on motion
  - foreground/background separation
- …
- Remember also the previous skin classification problem



Problems with EM

1. Local minima

   k-means is NP-hard even with k=2

2. Need to know number of clusters (k)

   (solution: see e.g. Bayesian information criteria (BIC))

3. Need to choose generative model

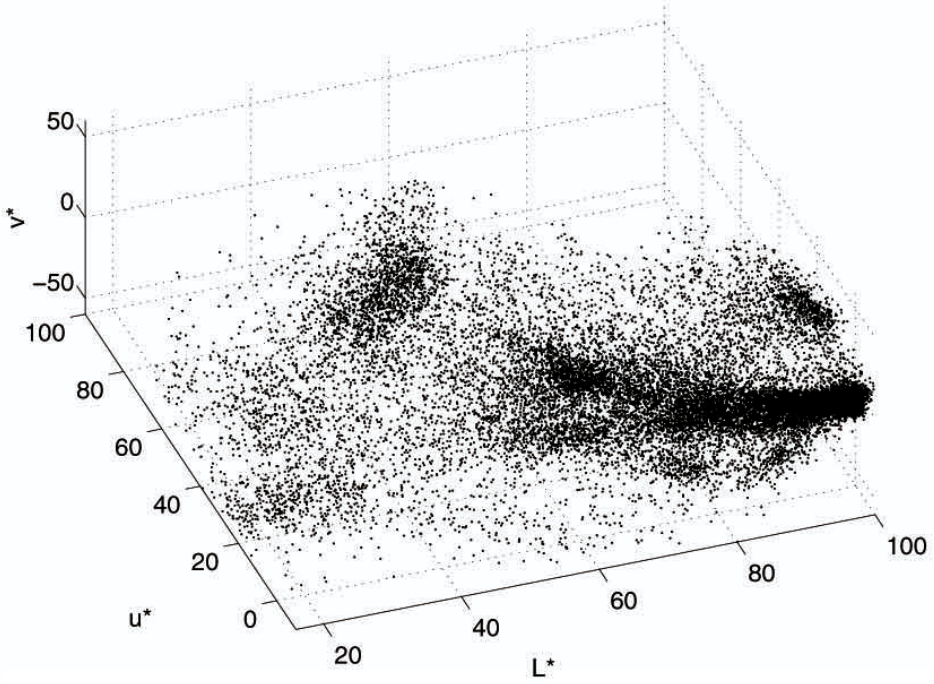
Mean-shift clustering

Mean shift algorithm

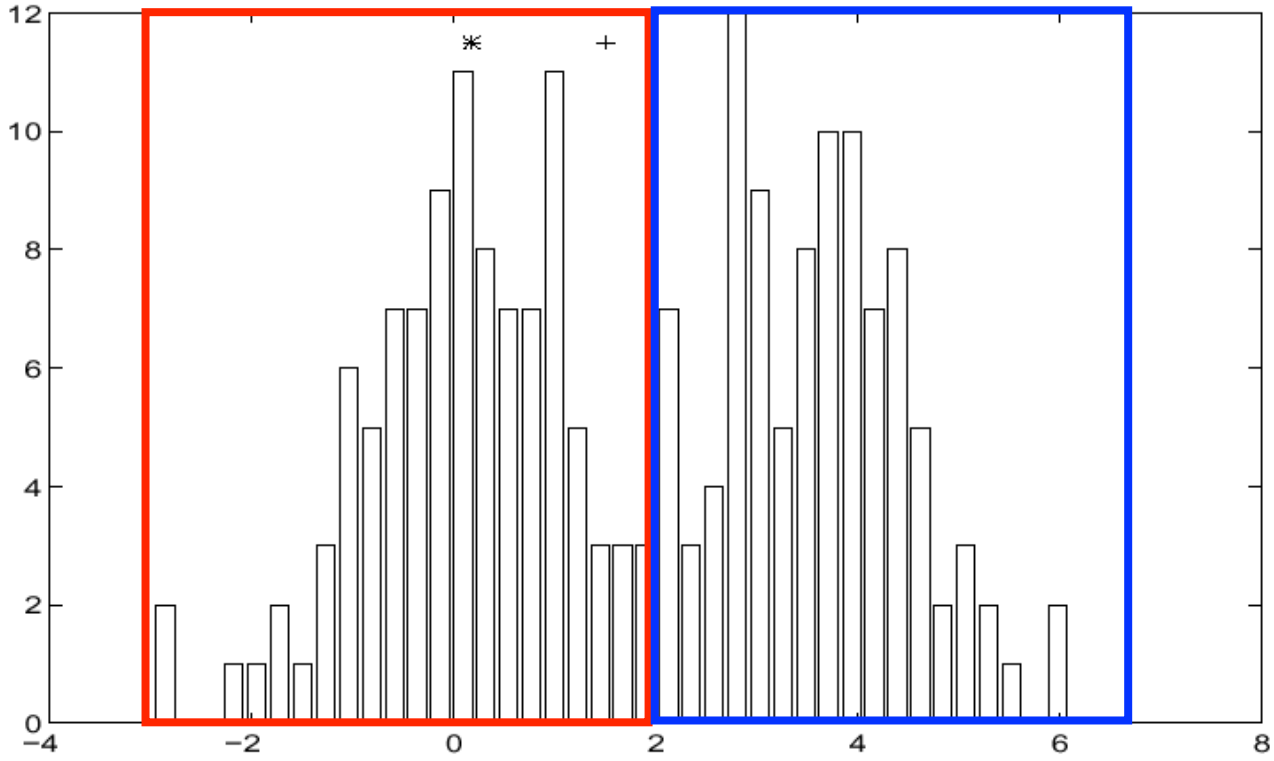
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

Feature space



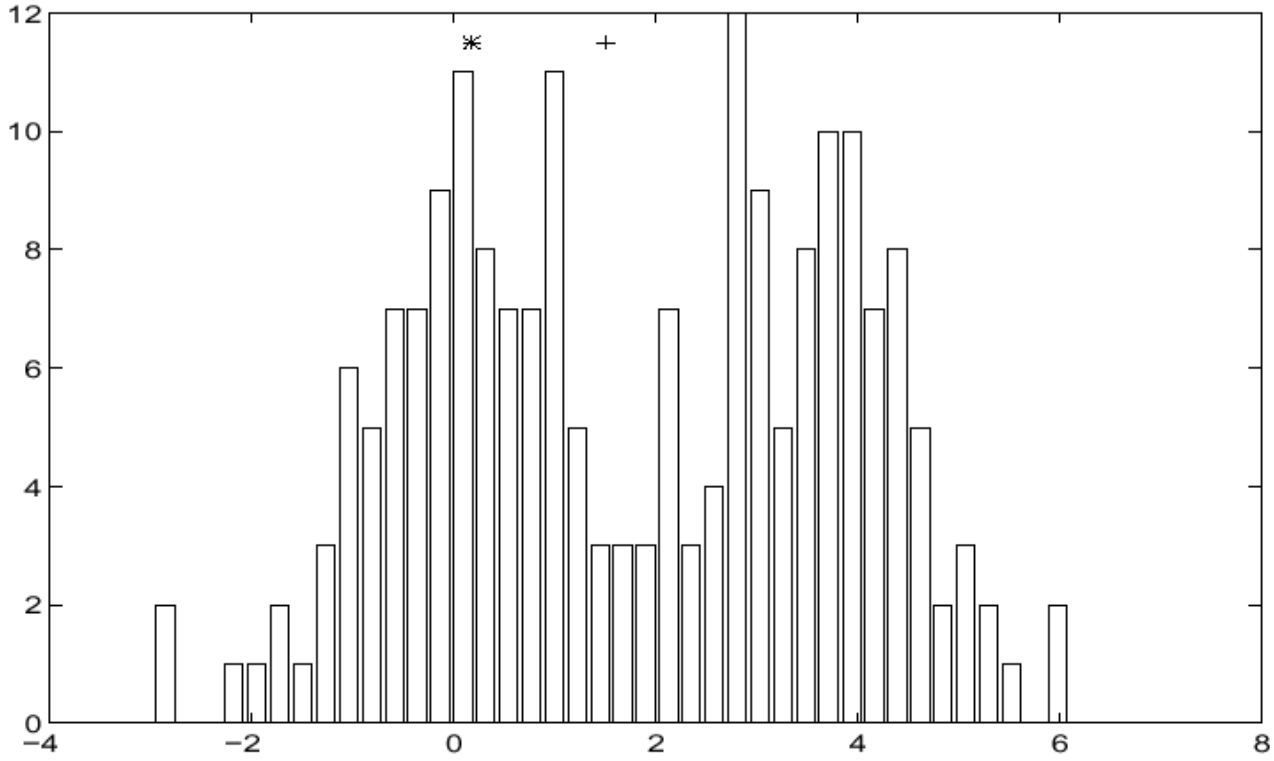
Finding Modes in a Histogram



How Many Modes Are There?

- Easy to see, hard to compute

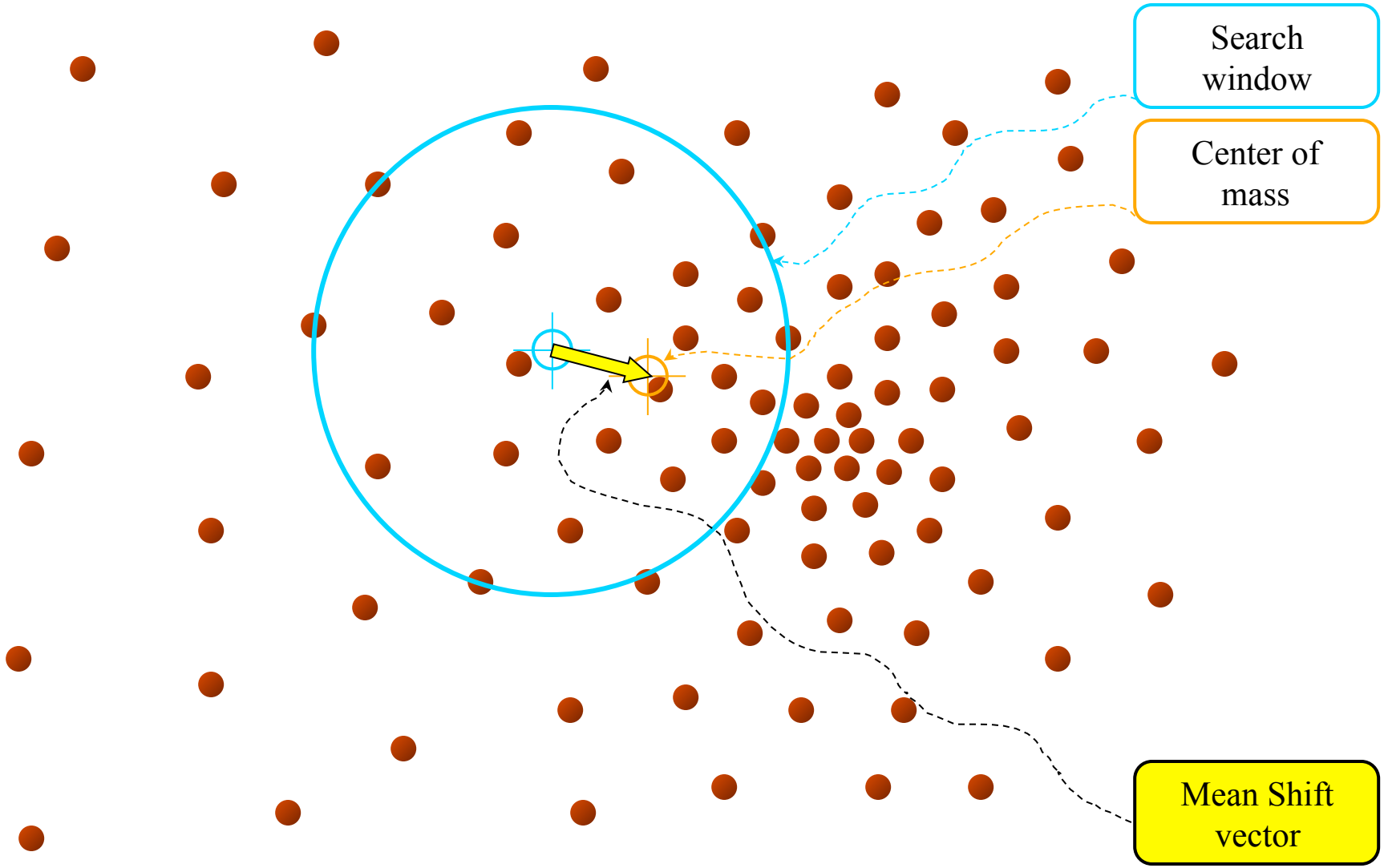
Mean Shift [Comaniciu & Meer]



Iterative Mode Search

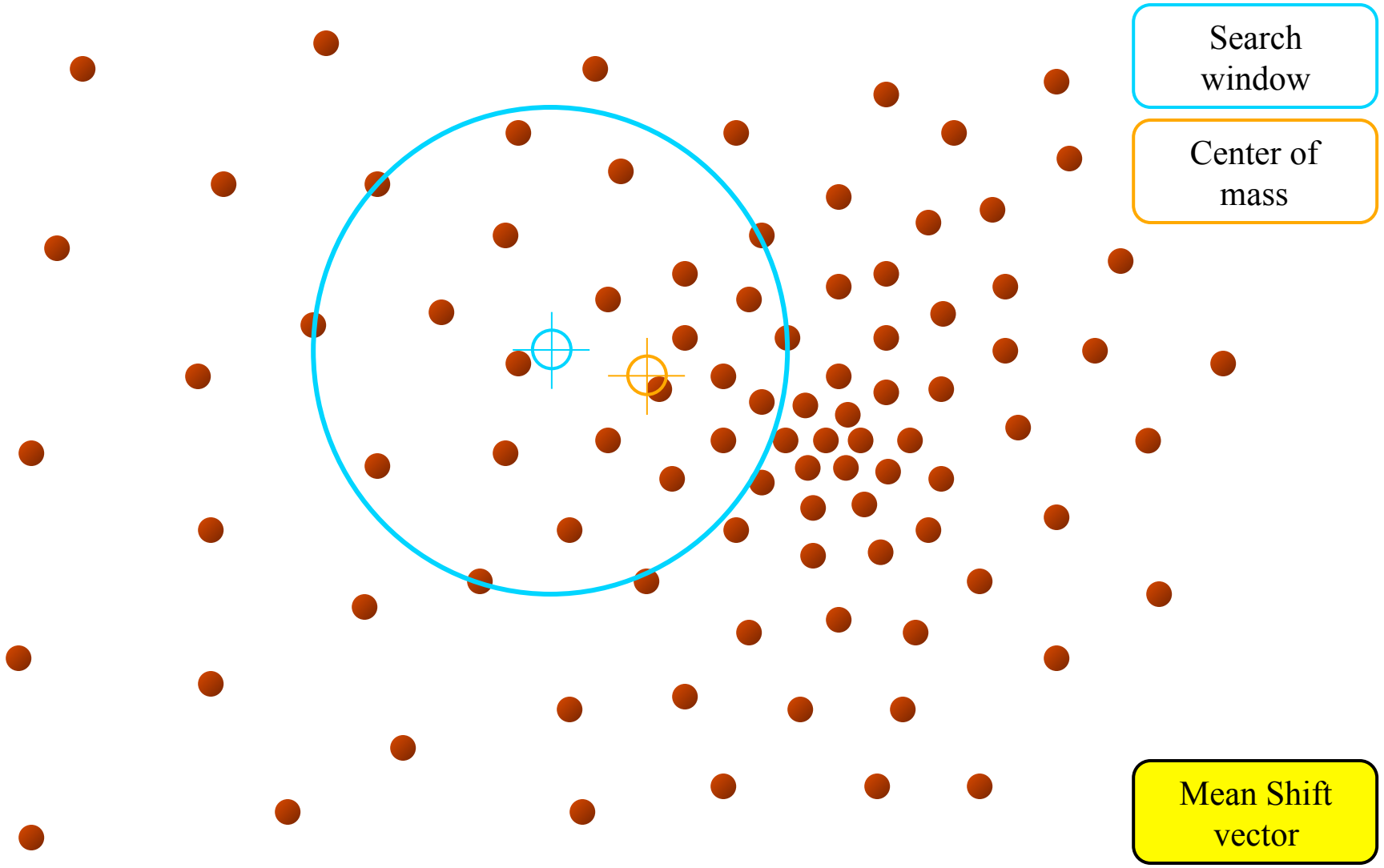
1. Initialize random seed, and window W
2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} xH(x)$
3. Translate the search window to the mean
4. Repeat Step 2 until convergence

Mean shift



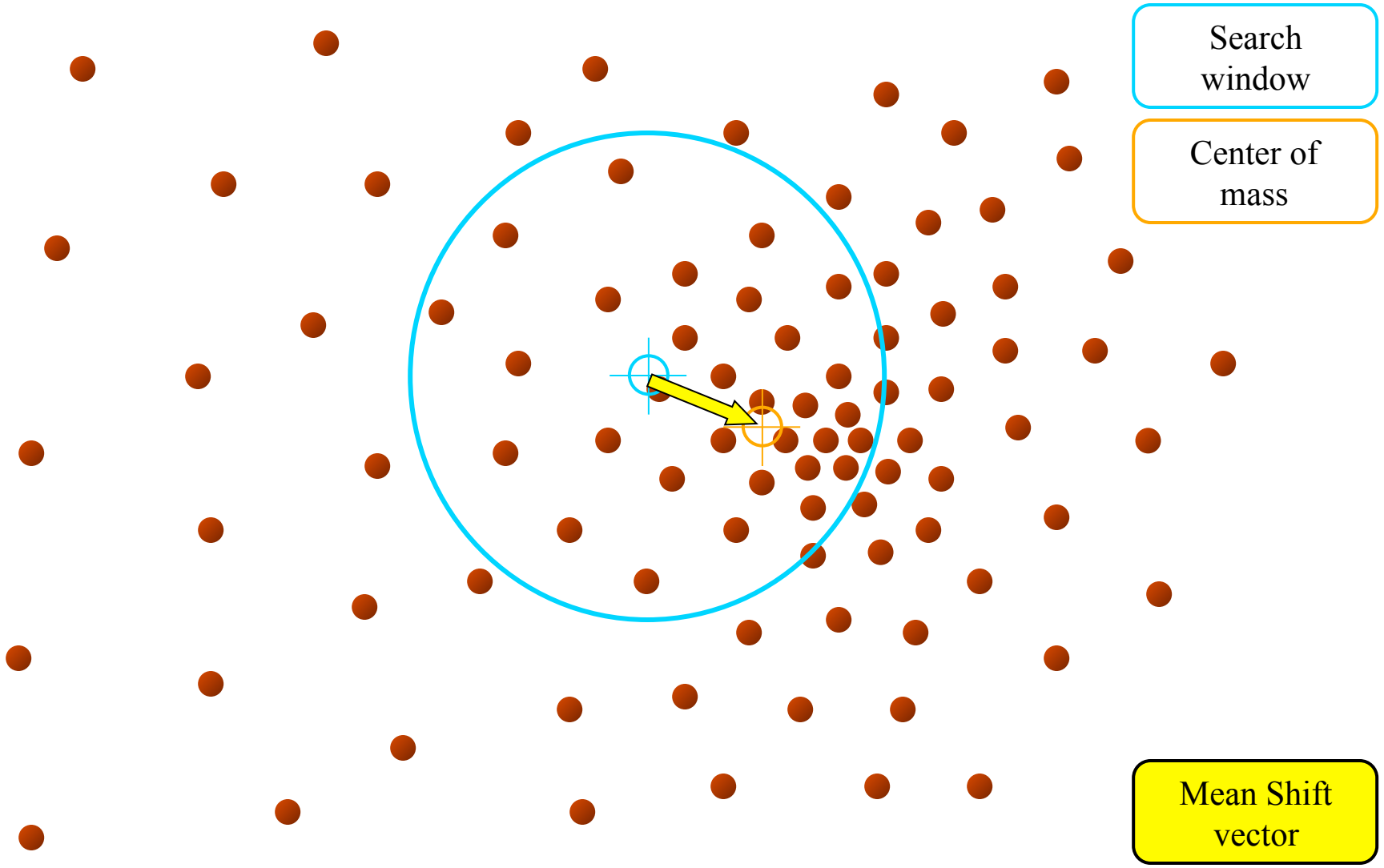
Slide by Y. Ukrainitz & B. Sarel

Mean shift



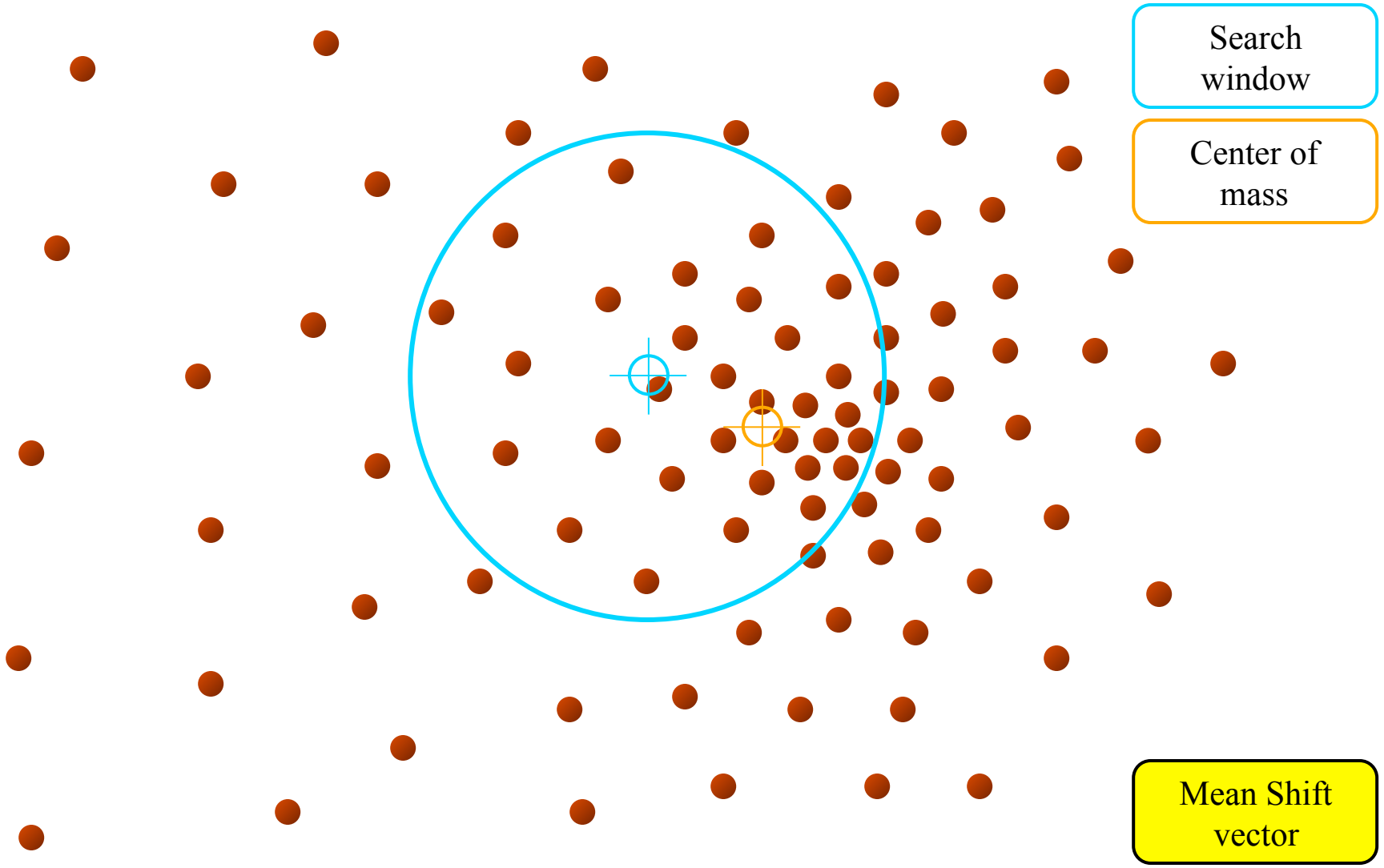
Slide by Y. Ukrainitz & B. Sarel

Mean shift



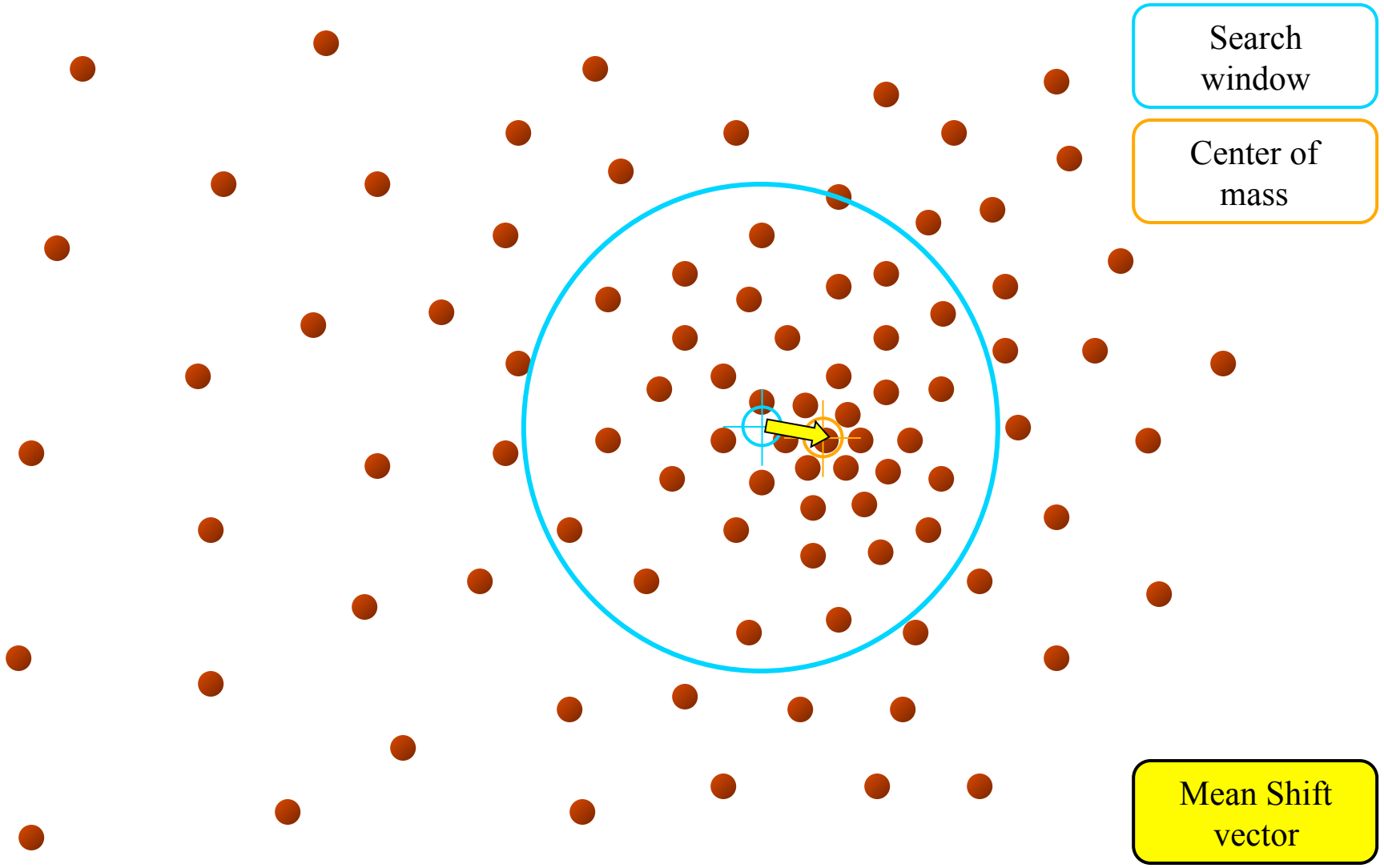
Slide by Y. Ukrainitz & B. Sarel

Mean shift



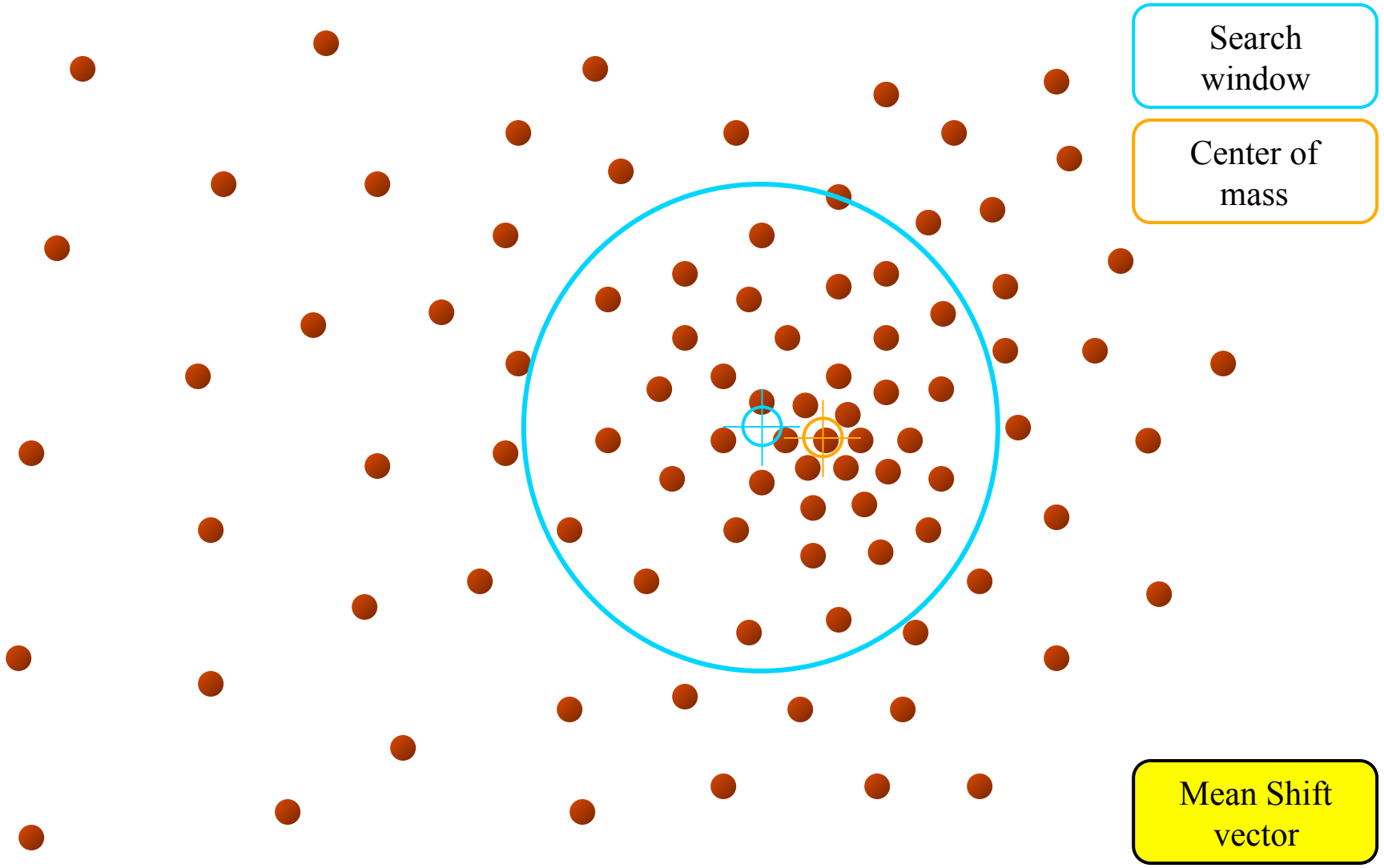
Slide by Y. Ukrainitz & B. Sarel

Mean shift



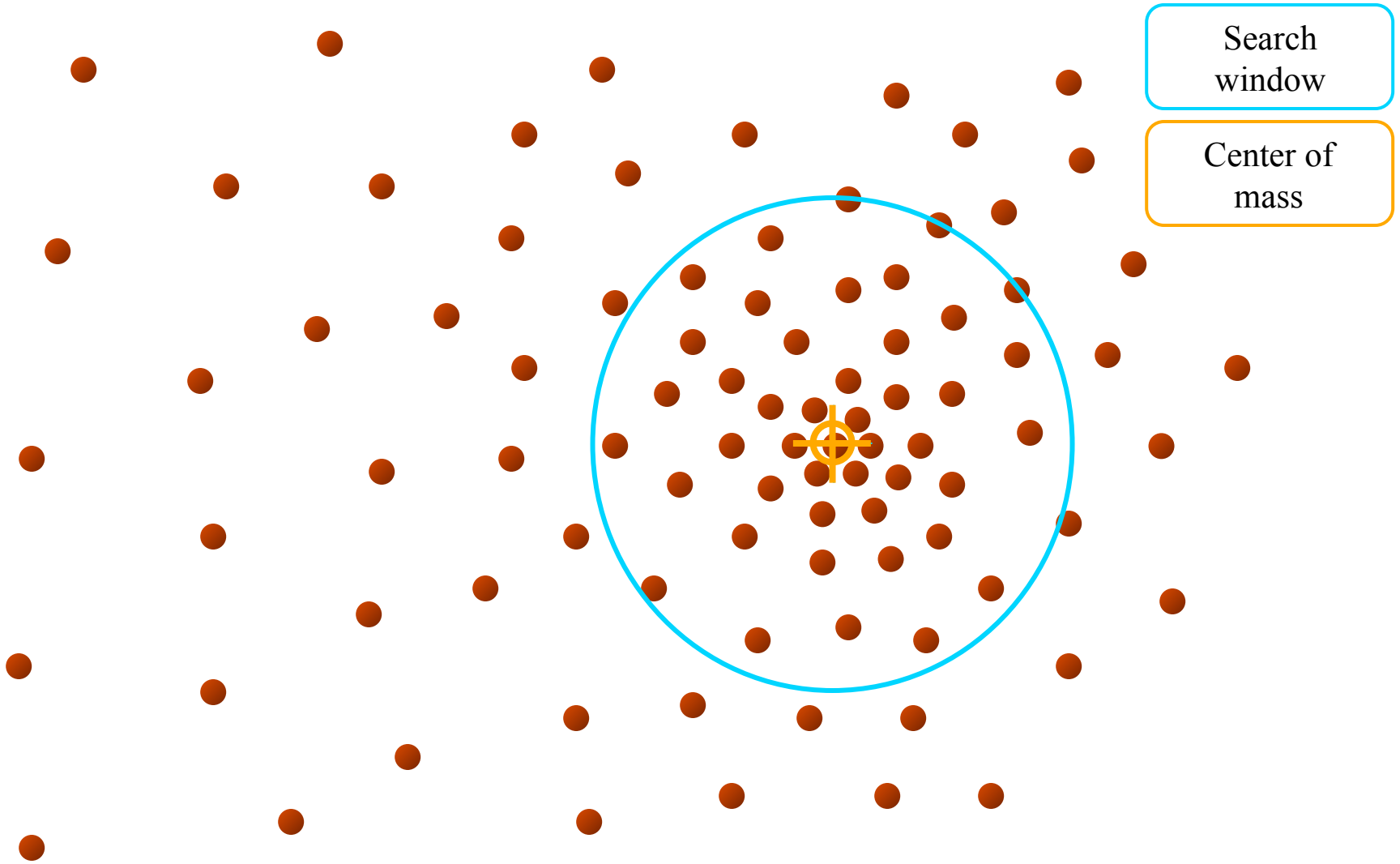
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

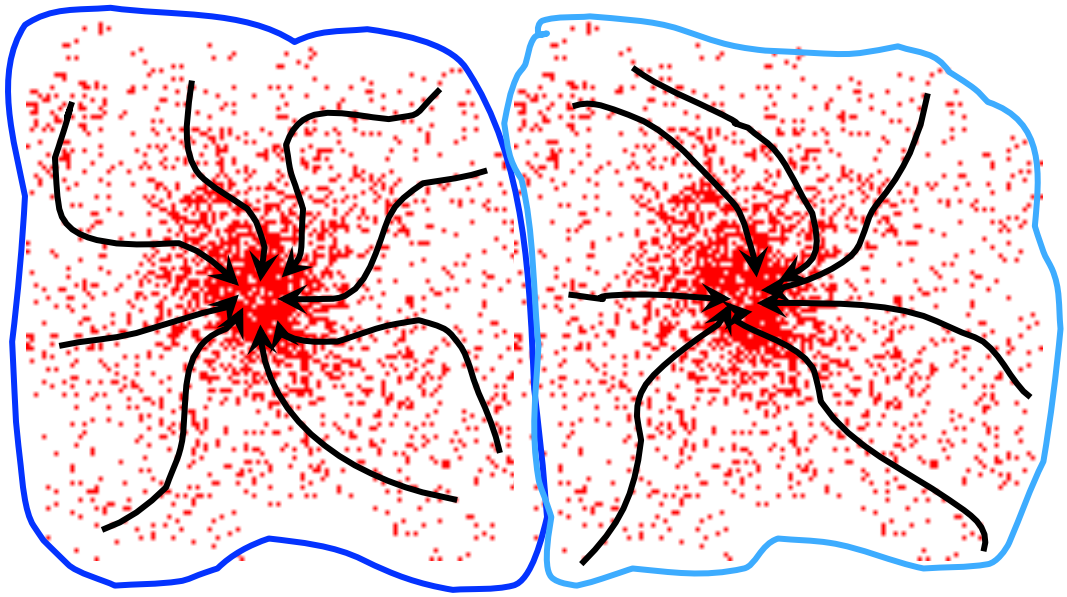
Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

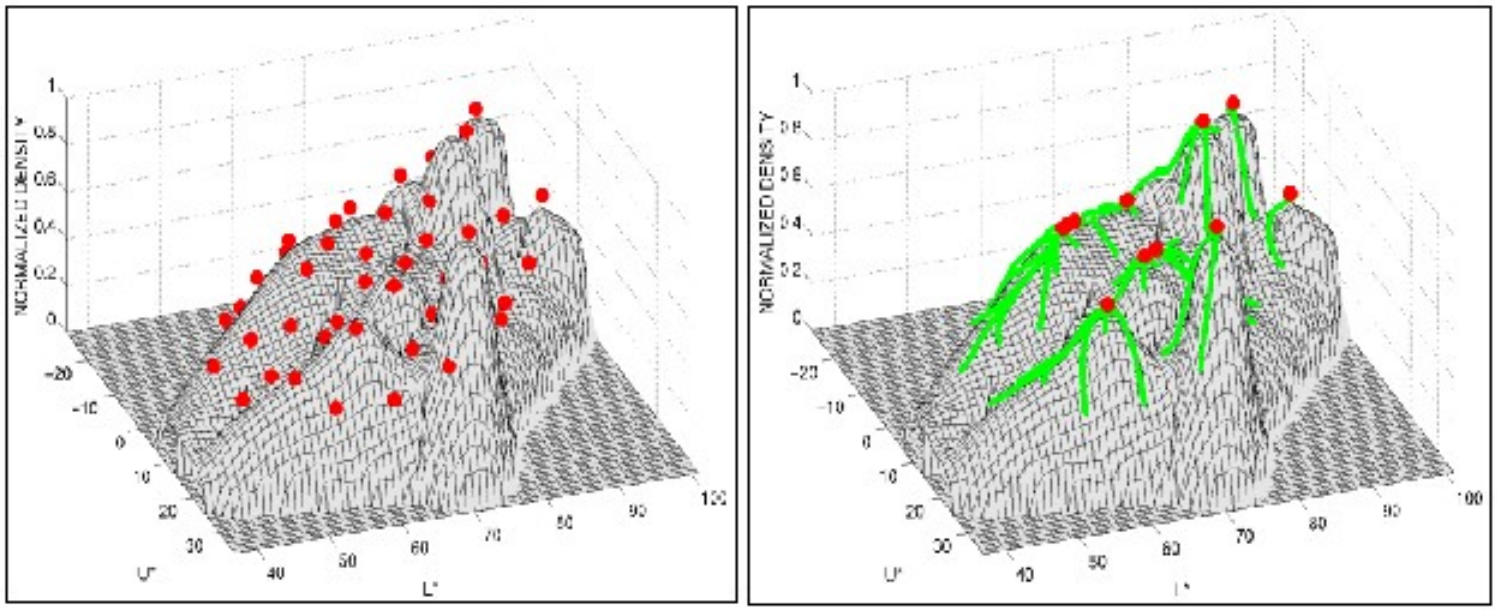


Slide by Y. Ukrainitz & B. Sarel

Mean-Shift

Approach

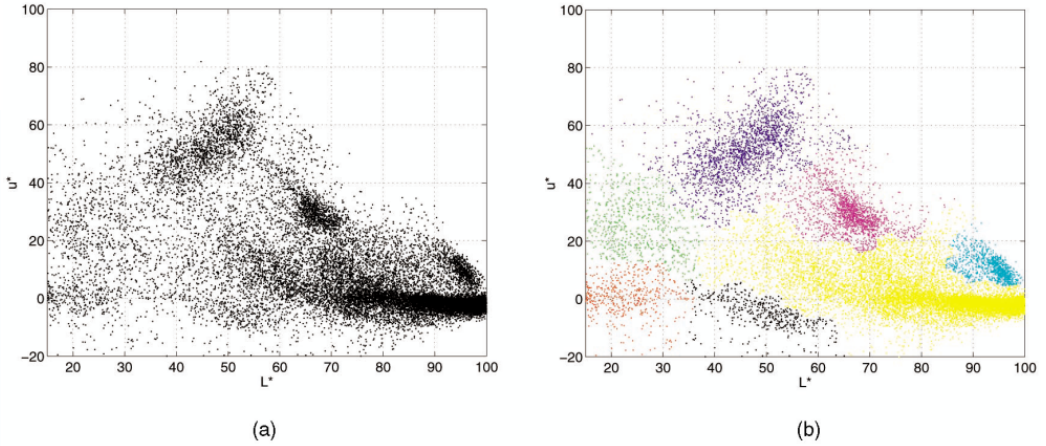
- Initialize a window around each point
- See where it shifts—this determines which segment it's in
- Multiple points will shift to the same segment

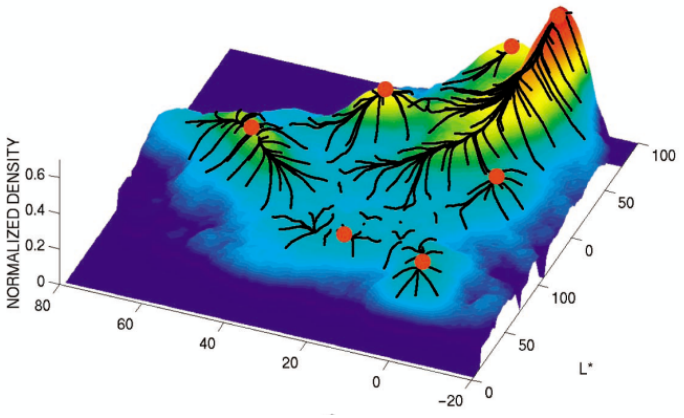


Mean shift trajectories

Mean shift clustering/segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode

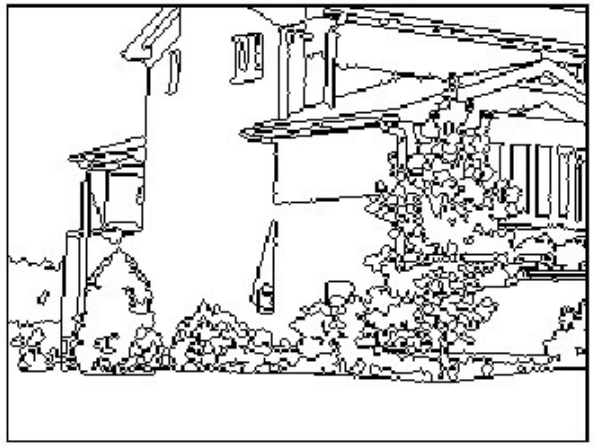




Mean-shift for image segmentation

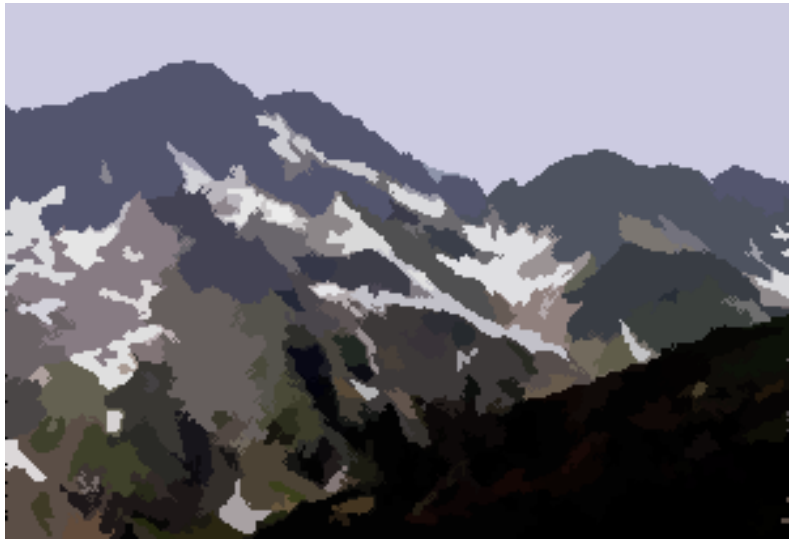
Useful to take into account spatial information

- instead of (R, G, B), run in (R, G, B, x, y) space
- D. Comaniciu, P. Meer, Mean shift analysis and applications, *7th International Conference on Computer Vision*, Kerkyra, Greece, September 1999, 1197-1203.
  - http://www.caip.rutgers.edu/riul/research/papers/pdf/spatmsft.pdf



More Examples: http://www.caip.rutgers.edu/~comanici/segm_images.html

Mean shift segmentation results



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean shift segmentation results

Mean shift

Pros:

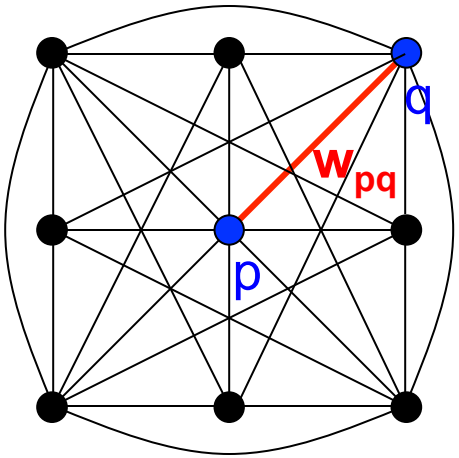
- Does not assume shape on clusters
- One parameter choice (window size)
- Generic technique
- Find multiple modes

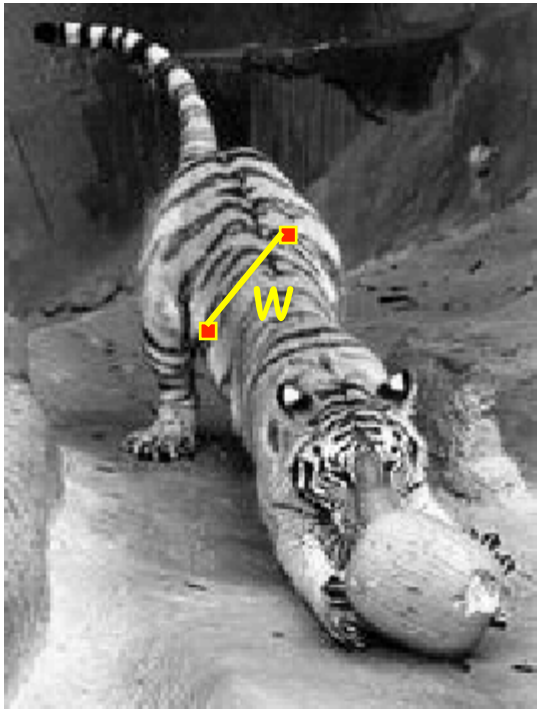
Cons:

- Selection of window size
- Does not scale well with dimension of feature space

Graph-based Segmentation

Images as graphs





Fully-connected graph

- node (vertex) for every pixel
- link between *every* pair of pixels, **p**,**q**
- affinity weight $\mathbf{w_{pq}}$ for each link (edge)
  - $\mathbf{w_{pq}}$ measures *similarity*
    - » similarity is *inversely proportional* to difference (in color and position…)

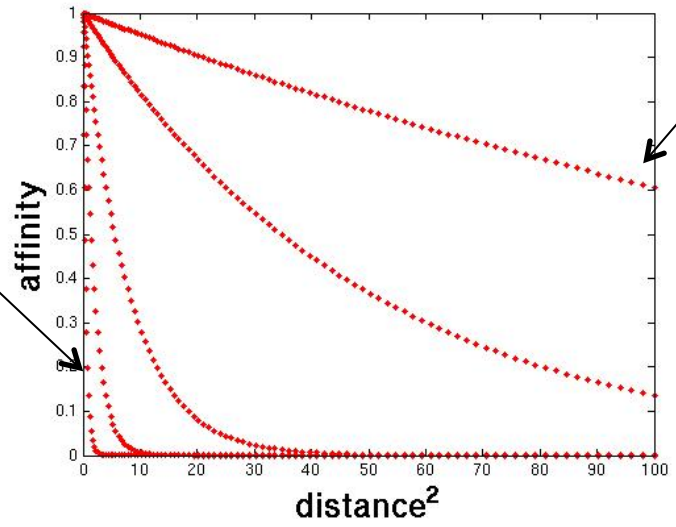
Source: Steve Seitz

Measuring affinity

One possibility:

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x-y\|^2\right)\right\}$$

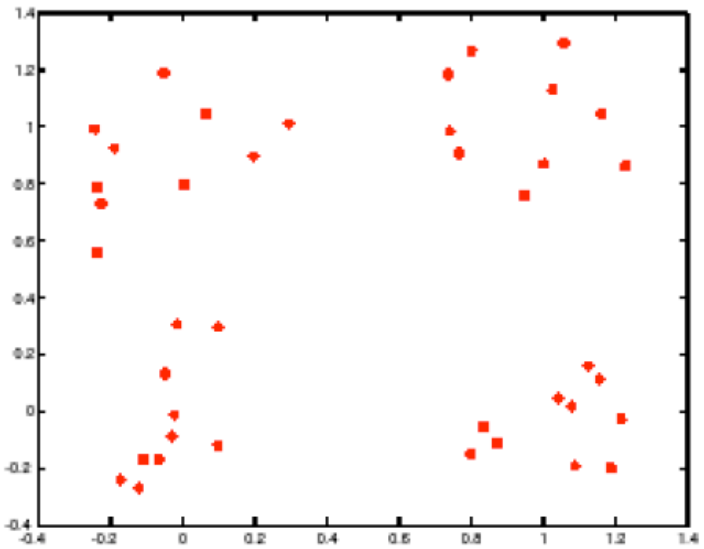
Small sigma: group only nearby points



Large sigma: group distant points

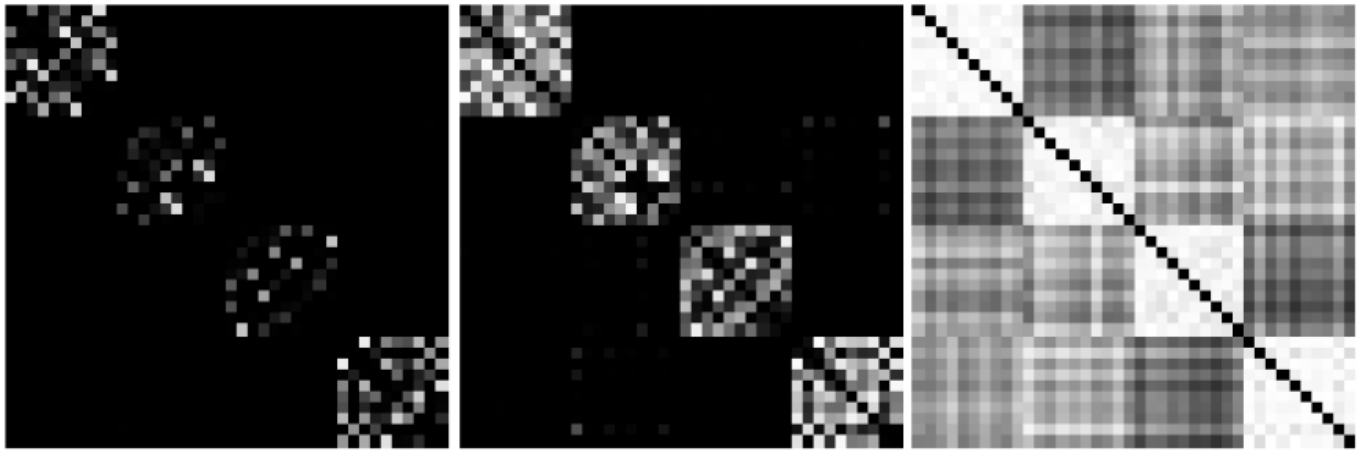
Measuring affinity

Data points



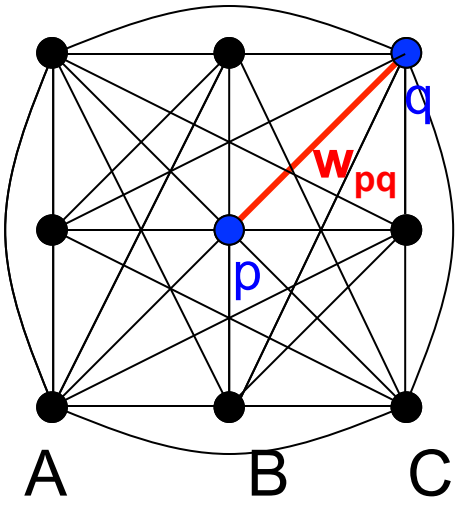
σ=.2

Affinity matrices



σ=.1 σ=.2 σ=1

Segmentation by Graph Cuts

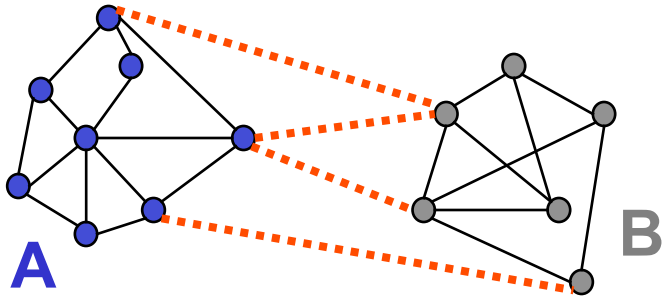


Break Graph into Segments

- Want to delete links that cross **between** segments
- Easiest to break links that have low similarity (low weight)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Source: Steve Seitz

Cuts in a graph: Min cut



Link Cut

- set of links whose removal makes a graph disconnected
- cost of a cut: $$cut(A,B) = \sum_{p \in A, q \in B} w_{p,q}$$

Find minimum cut

- gives you a segmentation
- fast algorithms exist for doing this

Source: Steve Seitz

Minimum cut

Problem with minimum cut:

Weight of cut proportional to number of edges in the cut; tends to produce small, isolated components.

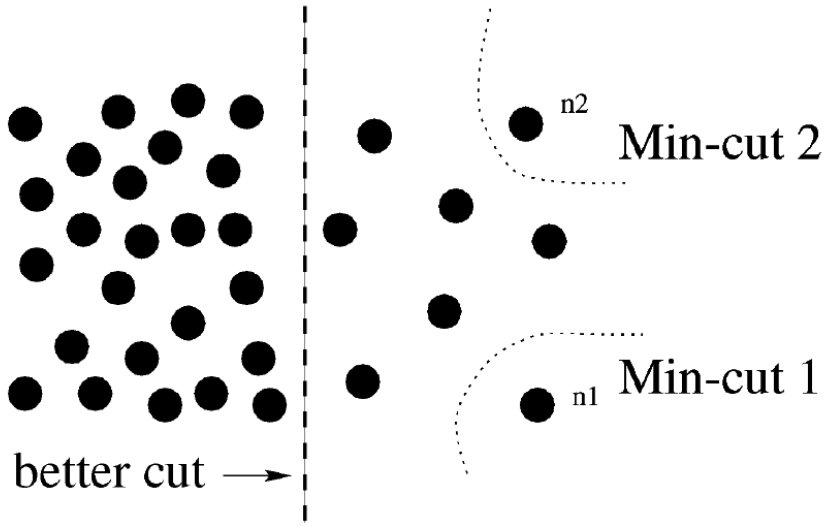
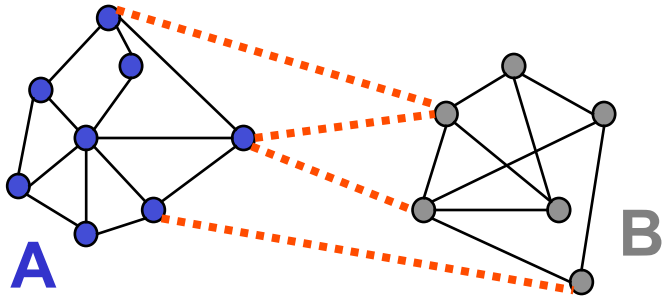


Fig. 1. A case where minimum cut gives a bad partition.

[Shi & Malik, 2000 PAMI]

Cuts in a graph: Normalized cut



Normalized Cut

- fix bias of Min Cut by **normalizing** for size of segments:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

assoc(A,V) = sum of weights of all edges that touch A

- Ncut value small when we get two clusters with many edges with high weights, and few edges of low weight between them
- Approximate solution for minimizing the Ncut value : generalized eigenvalue problem.

J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

Source: Steve Seitz

Example results

Results: Berkeley Segmentation Engine

http://www.cs.berkeley.edu/~fowlkes/BSE/

Normalized cuts: pros and cons

Pros:

Generic framework, flexible to choice of function that computes weights (“affinities”) between nodes

Does not require model of the data distribution

Cons:

Time complexity can be high

- Dense, highly connected graphs → many affinity computations
- Solving eigenvalue problem

Preference for balanced partitions

Modified Greedy Segmentation

1. Find minimum spanning tree (Felzenszwalb and Huttenlocher)
2. Create a sorted list of distance/neighbor pairs: (measure/edge pairs)

$$L = (d_1,e_1), (d_2,e_2), ... (d_N,e_N)$$

- Int(C) = $\max_e$ w(e) --- max dissimilarity
- Mint(C1, C2) = min (Int(C1) + k/|C1| , Int(C2) + k/|C2|)
- Diff(C1,C2) = min w(e), e joining C1 to C2

3. S_i is a segmentation; S_0 has each vertex by itself

   for i=1 to N

   a. If e_i joins a disjoint regions A and B in S_{i-1} and w(e_i) < Mint(A,B)

   Merge A and B in S_i

   else

   $S_i = S_{i-1}$

   Can show that Diff(C1,C2) > Mint(C1,C2) holds on resulting partition

   O(N log N) runtime

Summary

Segmentation to find object boundaries or mid-level regions, tokens.

Bottom-up segmentation via clustering

- General choices -- features, affinity functions, and clustering algorithms

Grouping also useful for quantization, can create new feature summaries

- Texton histograms for texture within local region

Example clustering methods

- K-means
- Mean shift
- Graph cut, normalized cuts

Approximate Minimization using Graph Cuts

Boykov, Veksler, Zabih, IEEE PAMI 23(11), pp 1222ff, 2001

Slides modified from those graciously provided by Ramin Zabih

Outline (Part 1)

Graph cuts for pixel labeling problems

- Problem definition and motivation
- Underlying graph algorithm (max flow)

Global and strong local minima

- Convex: exact global minimum
- Non-convex: expansion move algorithm

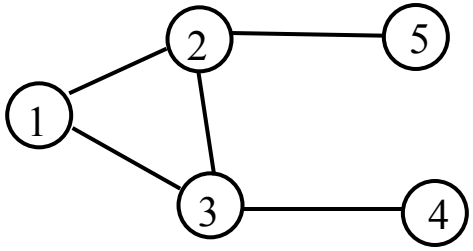
Theoretical and experimental properties

- How close do we get to the global minimum?
- What problems can graph cuts solve?

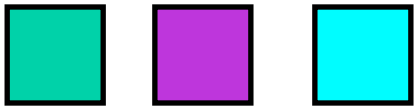
Pixel labeling problem

Given

$\mathcal{S} = \{1, \cdots, n\}$ $\mathcal{N} \subseteq \mathcal{S} \times \mathcal{S}$



$\mathcal{L} = \{l_1, \ldots, l_m\}$

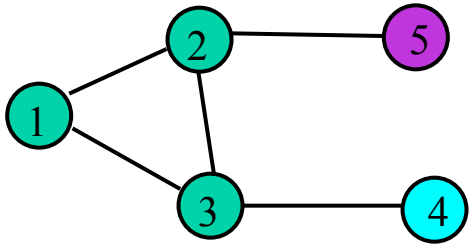


Assignment cost for giving a particular label to a particular node. Written as D.

Separation cost for assigning a particular pair of labels to neighboring nodes. Written as V.

Find

Labeling f = (f_1,…,f_n)



Such that the sum of the assignment costs and separation costs (the energy E) is small

Solving pixel labeling problems

- We want to minimize the energy E(f)

$$\arg\min_f \underbrace{\sum_{p\in\mathcal{S}} D_p(f_p)}_{\text{assignment costs}} + \underbrace{\sum_{p,q\in\mathcal{N}} V(f_p, f_q)}_{\text{separation costs}}$$

Classical problem in vision and beyond

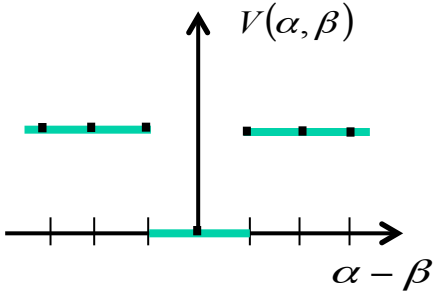
Bayesian justification

- Markov Random Fields (MRF's)

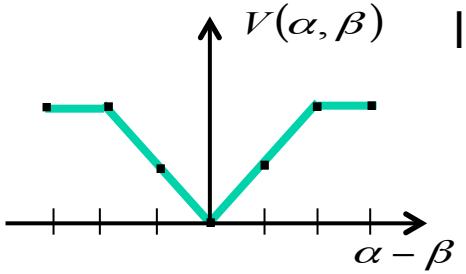
Choices of *V*

Robust

Potts model

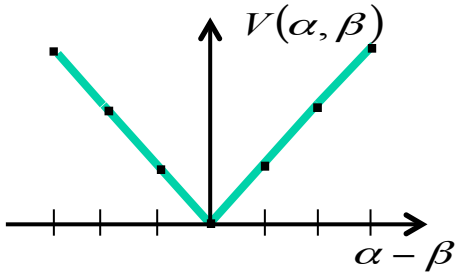


Truncated linear model

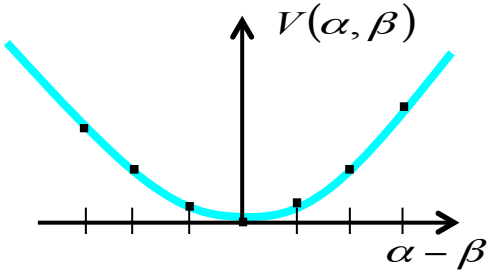


Not robust

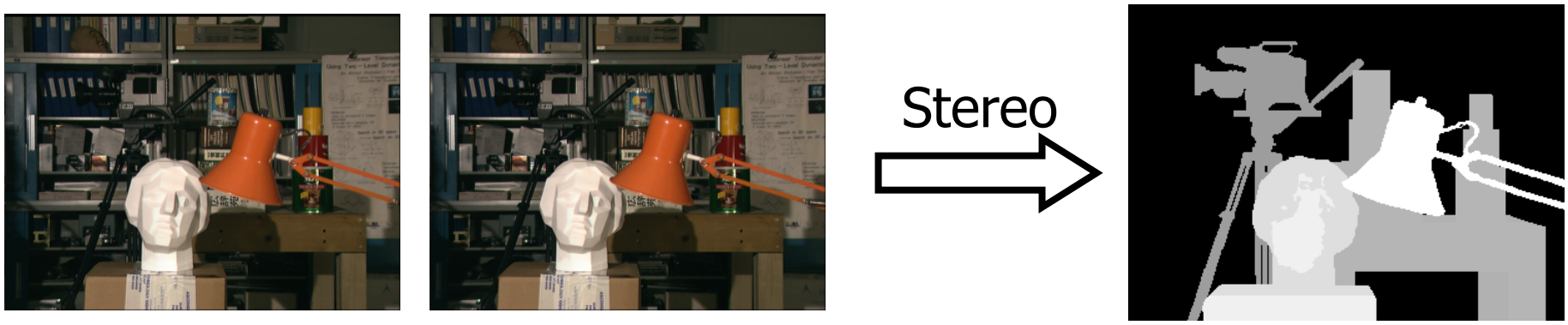
Linear model



Quadratic model



Pixel labeling for stereo



- Labels are shifts (hence depths)
- Assignment cost from intensity difference

$$D_p(\delta) = [I(p) - I'(p+\delta)]^2$$

- Neighboring pixels should be at similar depths
  - Except at the borders of objects!

How to minimize the energy?

Until late-90's, poor solutions

- Problem is NP-hard [K/BVZ PAMI '01]

In vision, we tend to focus on the deriving the "right" energy function

- Minimize via general-purpose methods
- Annealing, MCMC, CG, etc.

Computer scientists disagree

- General-purpose methods must be weak
- Nearby energy functions can be "easy"

Graph cuts

Reduce energy minimization problem to computing the min *s*-*t* cut on a graph

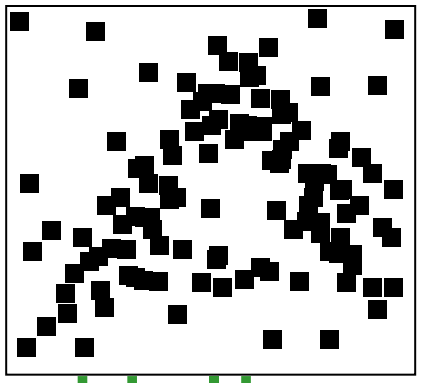
- Cuts are labelings, cut costs are energy
- Rapidly solvable by max flow

Running times are linear in the number of pixels and labels

- Asymptotically, low-order polynomial

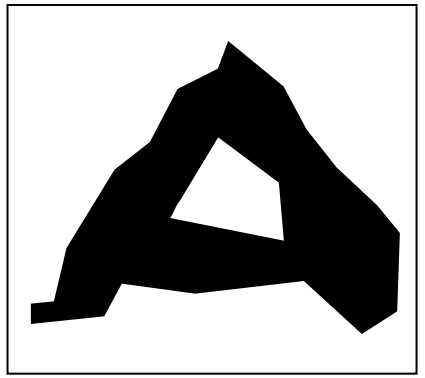
Binary image labeling problem

- Suppose we receive a noisy fax:
  - Some black pixels in the original image were flipped to white pixels, and some white pixels were flipped to black
  - We want to recover the original fax



original image

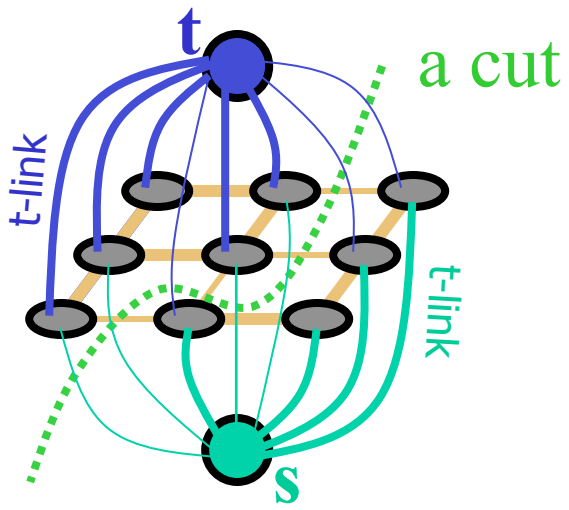
- Simple binary labeling problem
  - The sum of the assignment costs is the number of pixels that we think "flip"
  - The sum of the separation costs is the number of adjacent pixels that we think have different colors
  - Sometimes called "Ising" model



restored image

Solution via graph cuts

- Build the appropriate graph
  - Image pixels are nodes in the graph
    - Nearby pixels (nodes) connected by an edge, which we call an n-link
    - Terminal **s** is identified with label 0, and connected by edge we call a t-link with every image pixel
    - Terminal **t** is identified with label 1 and connected by t-link with every image pixel



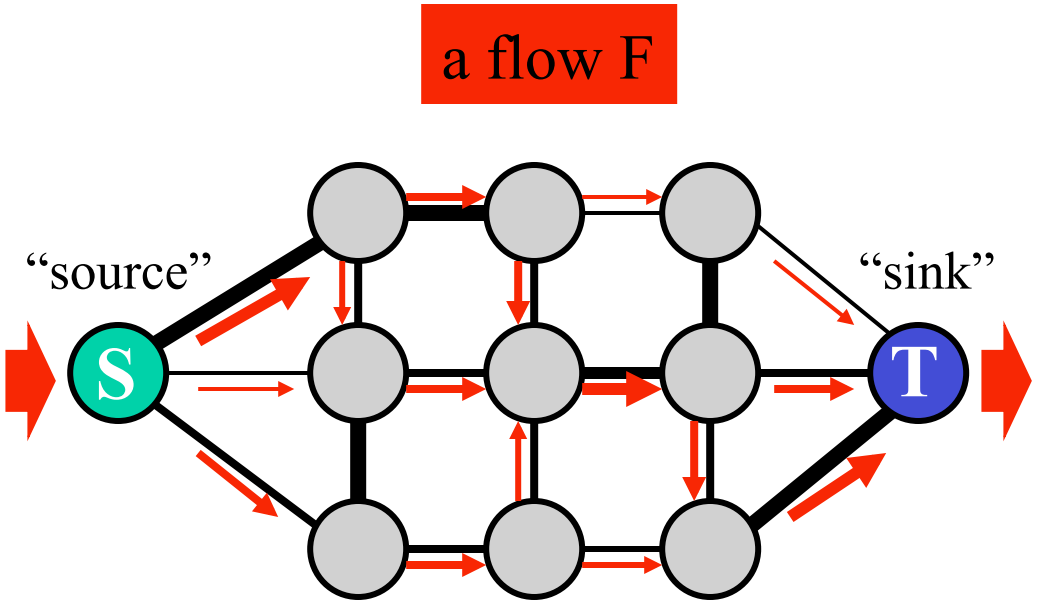
A cut separates **t** from **s**

Each pixel stays connected to either **t** or **s** (label 1 or 0)

Cuts correspond to labelings, and with right edge weights cost is same

Minimum cut gives the minimum energy labeling

Maximum flow problem

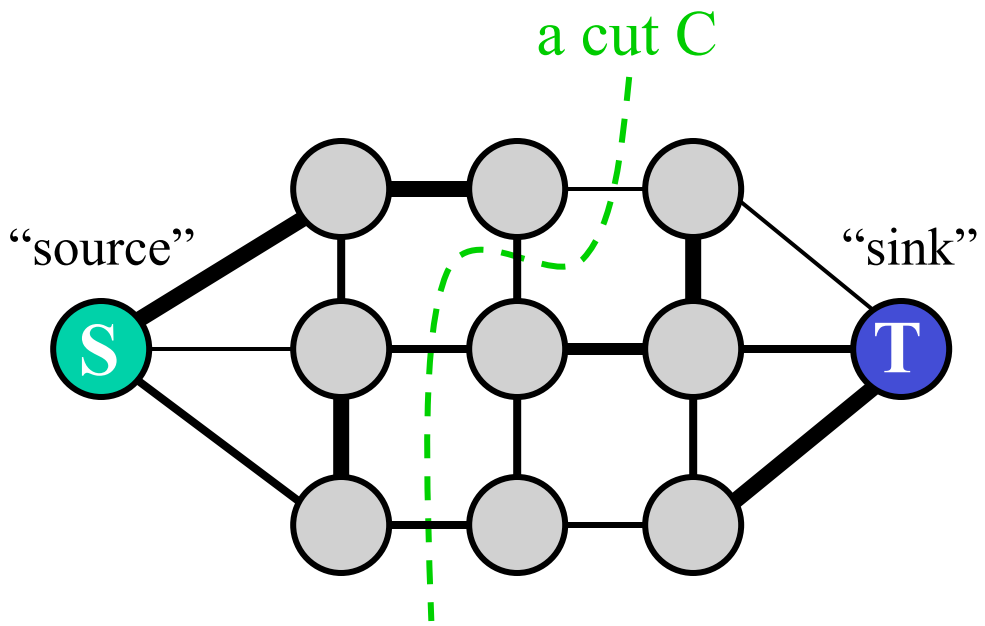


A graph with two terminals

Max flow problem:

- Each edge is a "pipe"
- Find the largest flow F of "water" that can be sent from the "source" to the "sink" along the pipes
- Source output = sink input = flow value
- Edge weights give the pipe's capacity

Minimum cut problem

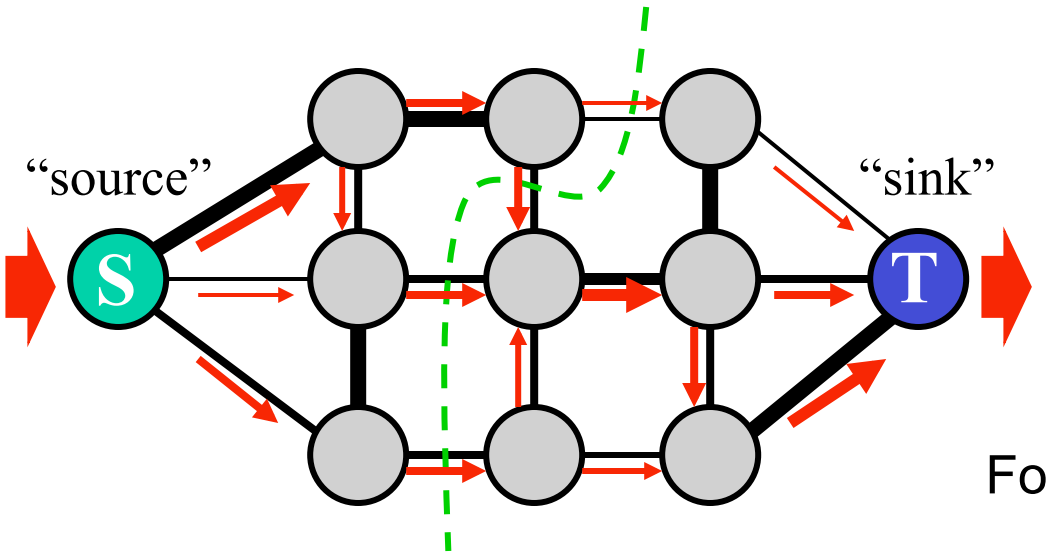


A graph with two terminals

Min cut problem:

- Find the cheapest way to cut the edges so that the “source” is separated from the “sink”
- Cut edges going from source side to sink side
- Edge weights now represent cutting “costs”

Max flow/Min cut theorem



A graph with two terminals

Max Flow = Min Cut:

- Proof sketch: value of a flow is value over any cut
- Maximum flow saturates the edges along the minimum cut
  - Ford and Fulkerson, 1962
  - Problem reduction!

Ford and Fulkerson gave first polynomial time algorithm for globally optimal solution

Fast algorithms for min cut

Max flow problem can be solved fast

- Many algorithms, such as augmenting paths
  - Find a path from S to T that does not go through any saturated edge
  - Push more flow through that path

Many graph problems are intractable

- Variants of min cut are NP-hard

Example: multiway cut problem

- More than 2 terminals
- Find lowest cost edges separating them all

What do graph cuts provide?

For less robust V, polynomial algorithm for global minimum!

- Discrete version of TV, but with non-convex D

For a particularly robust V, an approximation algorithm

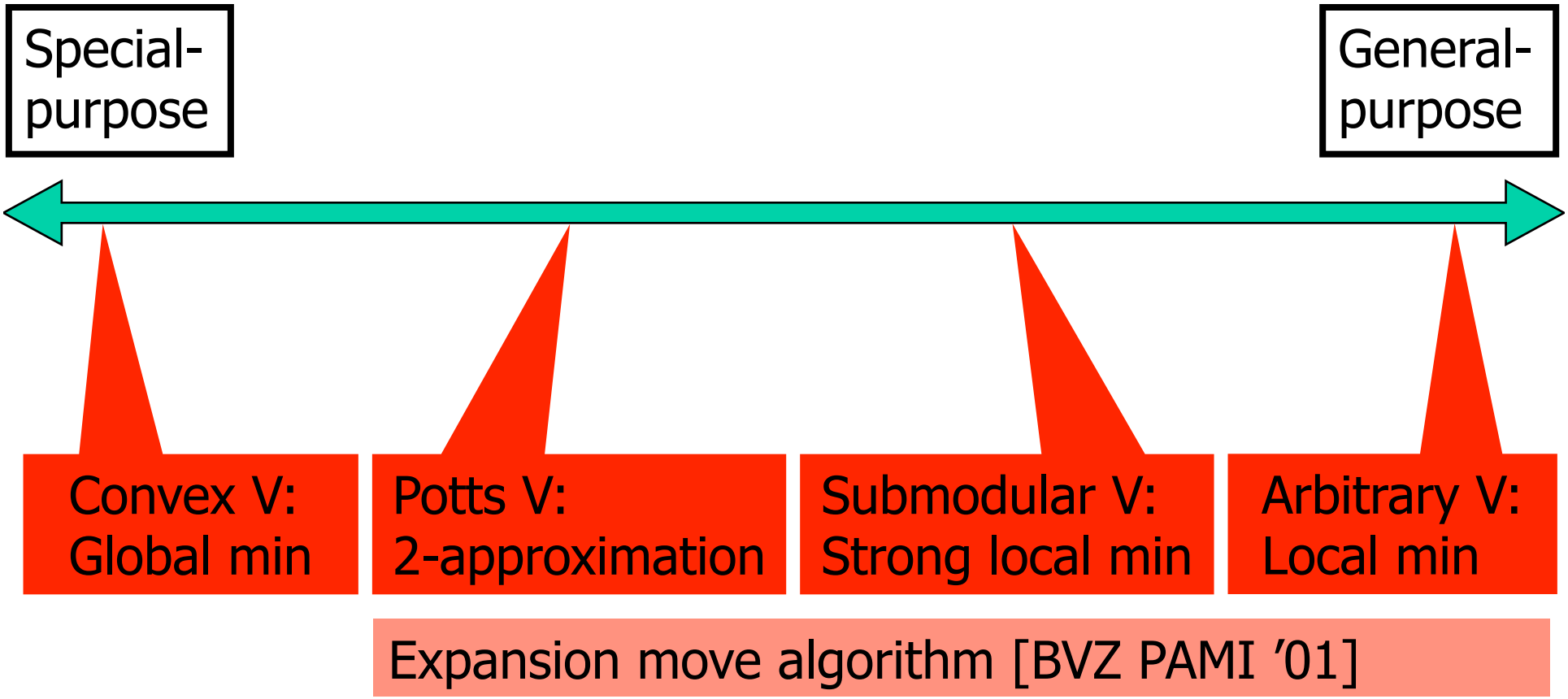
- Proof of NP hardness

For many choices of V, algorithms that find a "strong" local minimum

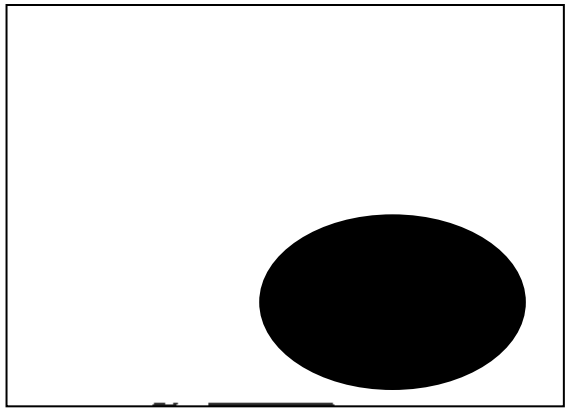
High quality experimental results

- Within 1% of the global minimum on a wide range of benchmarks

Spectrum of results



Expansion move energy

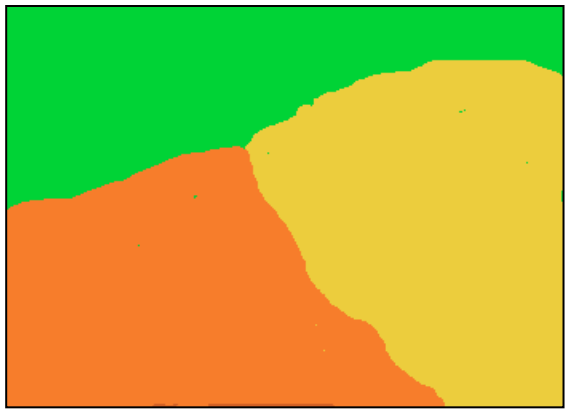


Goal: find the binary image with lowest energy

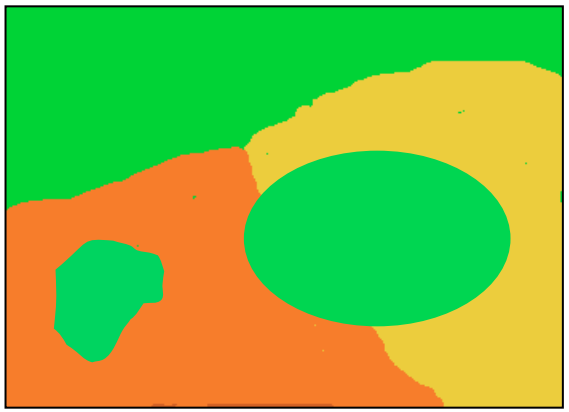
Binary image energy is a restricted version of original E

Depends on f, alpha

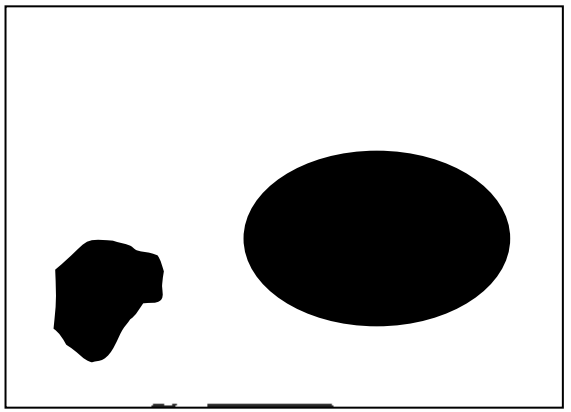
From Binary to N-ary Labeling



Input labeling

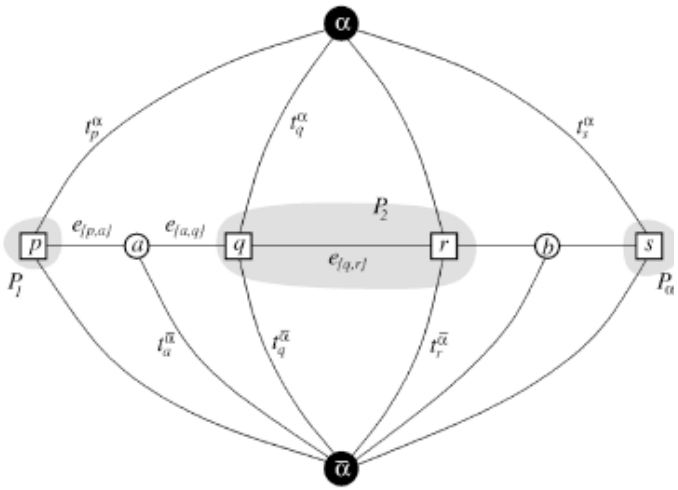


Expansion move

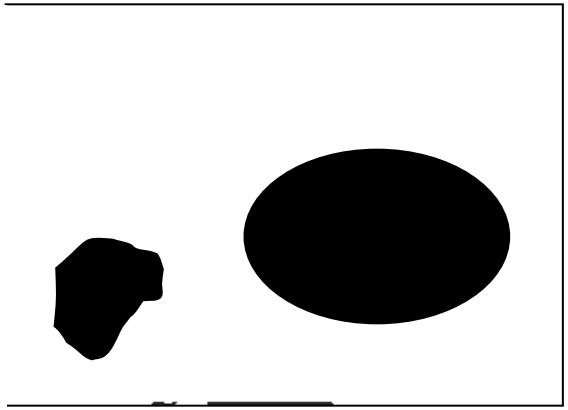


Binary image

Binary sub-problem

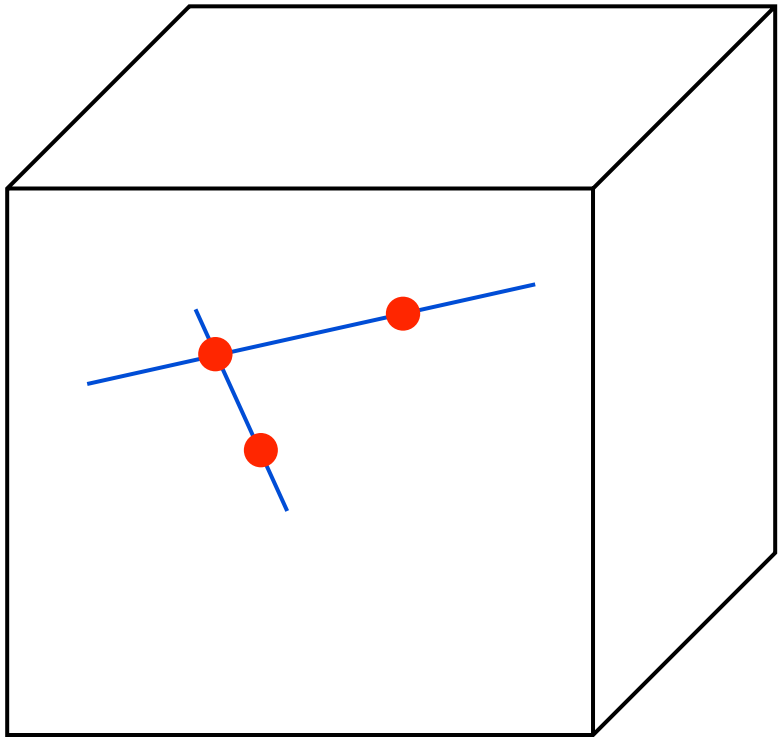


edge	weight	for
$t_p^{\bar{\alpha}}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\bar{\alpha}}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p \neq f_q$
$e_{\{a,q\}}$	$V(\alpha, f_q)$	
$t_a^{\bar{\alpha}}$	$V(f_p, f_q)$	
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p = f_q$



Binary image

Local improvement methods



- Subproblem: locally minimize restricted version of E
- Ultimately computes a minimum w.r.t. any line

Local improvement vs. Graph cuts

Continuous vs. discrete

- No floating point with graph cuts

Local min in line search vs. global min

Minimize over a line vs. hypersurface

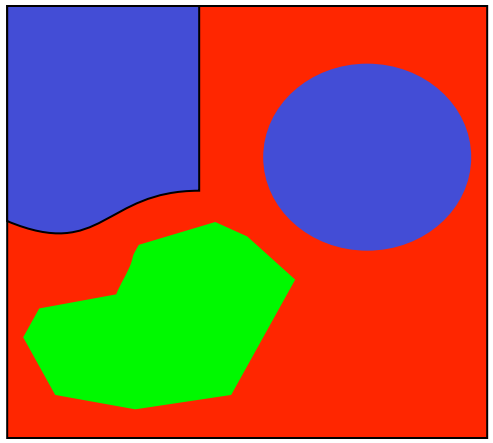
- Containing $O(2^n)$ candidates

Local minimum: weak vs. strong

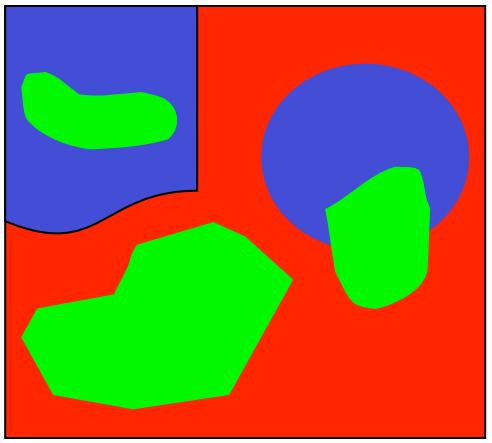
- Theoretical guarantees concerning distance from global minimum
  - 2-approximation for a common choice of E
- Within 1% of global min on benchmarks!

Expansion move algorithm

Input labeling f



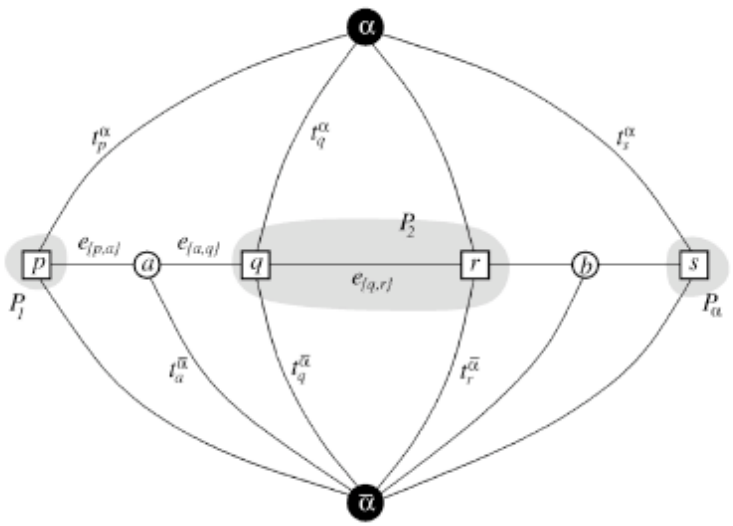
Green expansion move from f



Find green expansion move that most decreases E

- Move there, then find the best blue expansion move, etc
- Done when no alpha-expansion move decreases the energy, for any label alpha

Expansion move algorithm



T

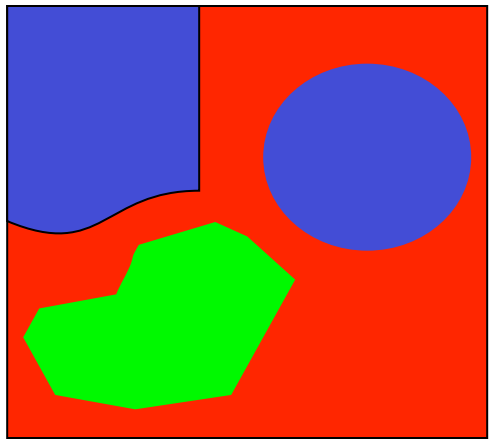
edge	weight	for
$t_p^{\bar{\alpha}}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\bar{\alpha}}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p \neq f_q$
$e_{\{a,q\}}$	$V(\alpha, f_q)$	
$t_a^{\bar{\alpha}}$	$V(f_p, f_q)$	
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p = f_q$

Find green expansion move that most decreases E

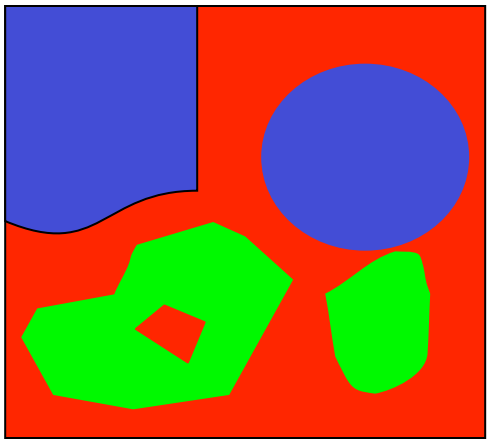
- Move there, then find the best blue expansion move, etc
- Done when no alpha-expansion move decreases the energy, for any label alpha

Swap algorithm

Input labeling f



Red/Green swap
move from f



Find a swap of pixel labels that most decreases energy

- Move there, then find the next best swap move, etc
- Done when no swap move decreases the energy, for any pair of labels

Swap algorithm

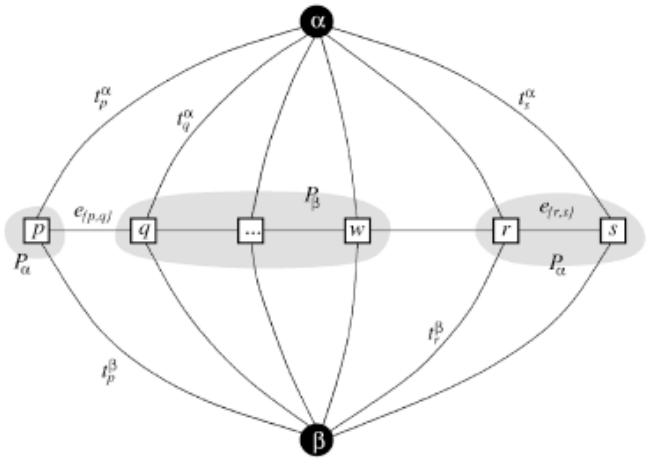


Fig. 4. An example of the graph $\mathcal{G}_{\alpha\beta}$ for a 1D image. The set of pixels in the image is $\mathcal{P}_{\alpha\beta} = \mathcal{P}_\alpha \cup \mathcal{P}_\beta$, where $\mathcal{P}_\alpha = \{p, r, s\}$ and $\mathcal{P}_\beta = \{q, \ldots, w\}$.

edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V(\alpha, \beta)$	$\substack{\{p,q\} \in \mathcal{N} \\ p,q \in \mathcal{P}_{\alpha\beta}}$

Find a swap of pixel labels that most decreases energy

- Move there, then find the next best swap move, etc
- Done when no swap move decreases the energy, for any pair of labels

Swap Optimization Alg.

1. Start with an arbitrary labeling f
2. Success = 0
3. For each label (resp. pair)
    1. Find f* = arg min E*(f') within one alpha-expansion (resp. alpha-beta swap) of f
    2. If (E(f*) < E(f)) set f = f* and success = 1
4. If success = 1, repeat from 2
5. Return f

Finding the Optimal Swap

Construct a subgraph just on the pair of labels

Compute special energies for the t nodes

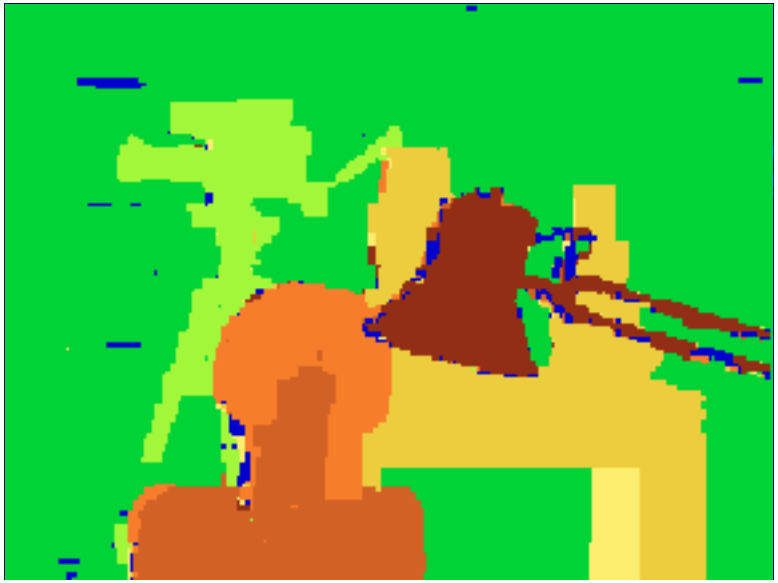
- t-alpha = data(alpha) + regularization over nodes on in graph
- t-beta = data(beta) + regularization over nodes not in graph
- Pixel connection = regularization V(alpha,beta)

Compute a cut

Assign new labels to pixels in subgraph

Sample results

Right answers



Dynamic programming
Graph cuts

Expansion moves in action

initial solution

-expansion

-expansion

-expansion

-expansion

-expansion

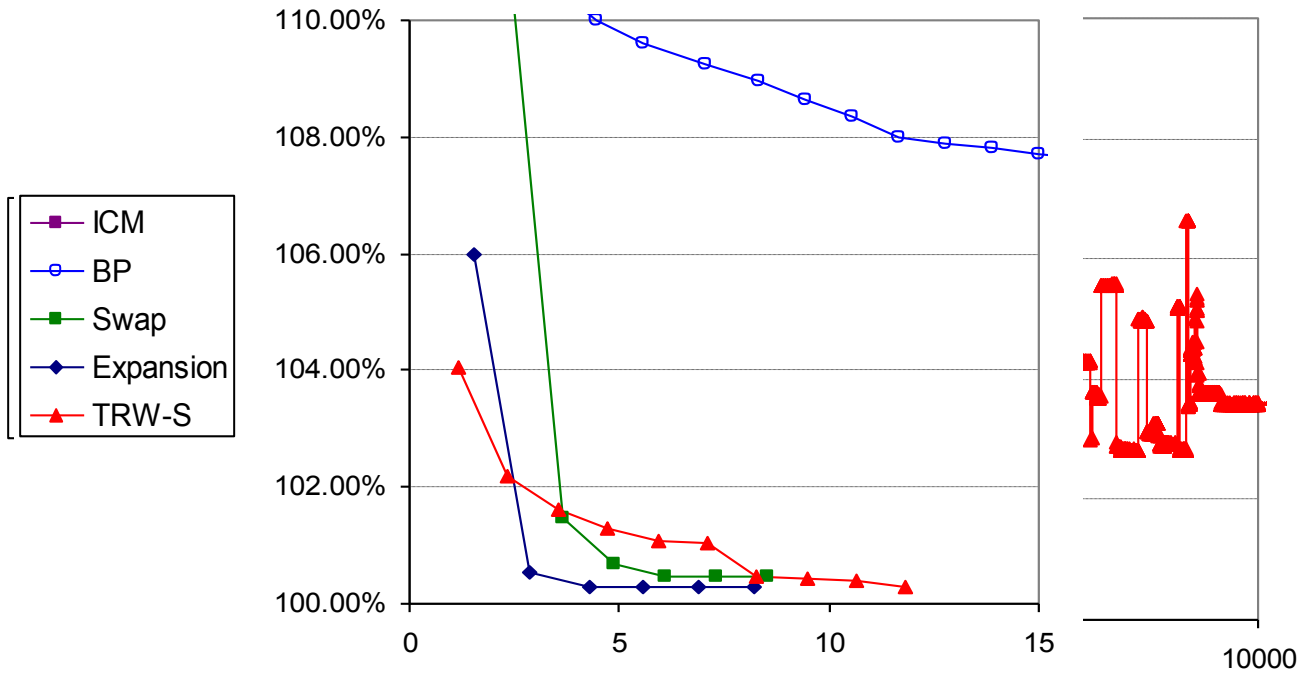
-expansion

-expansion

Must choose expansion that gives the largest decrease in energy:
binary energy minimization subproblem

Theoretical and experimental properties of the expansion move algorithm

Experimental performance



Easy problem (Boykov)

Hard problem (Kolmogorov)

Summary

Discrete optimization methods like graph cuts can be very powerful

High quality solutions for non-convex optimization problems in thousand of dimensions

Strong experimental results

Ties to many branches of applied math

Acknowledgements

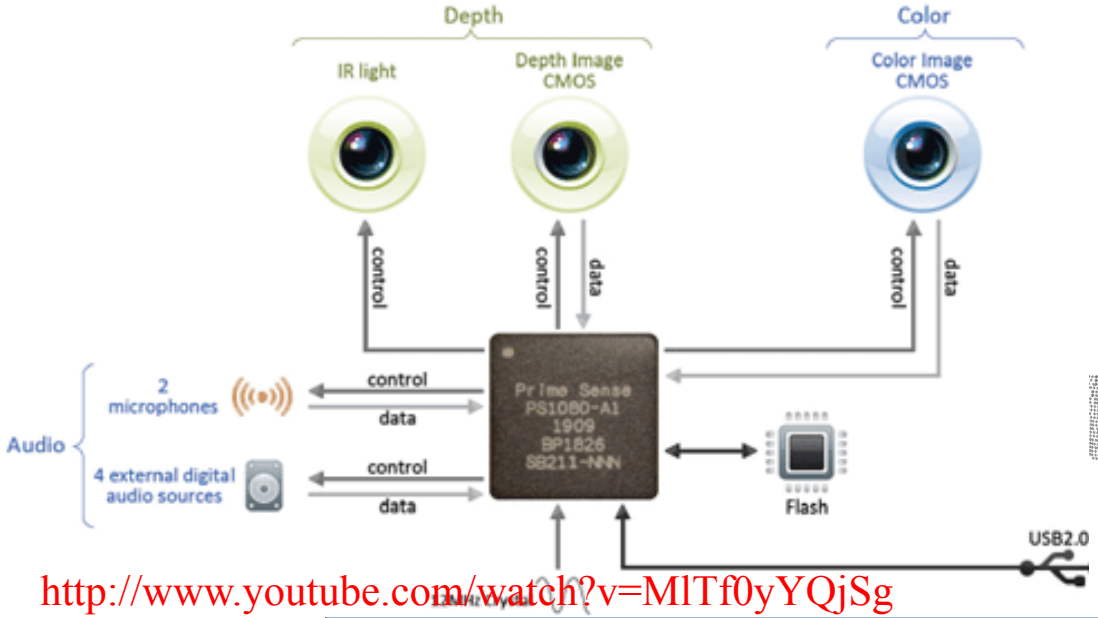
Major ideas

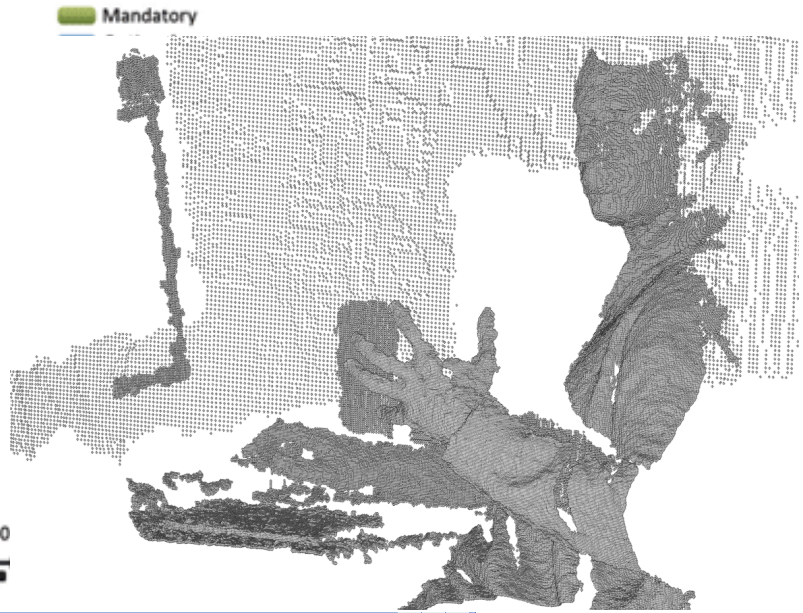
- Basic construction: Hammer ‘65
- Binary application: Greig, Porteus & Seheult ‘86
- Convex application: Ishikawa ‘03
- Expansion moves: Boykov, Veksler & Zabih ‘01
- Regularity: Kolmogorov & Zabih ‘04

Slides from:

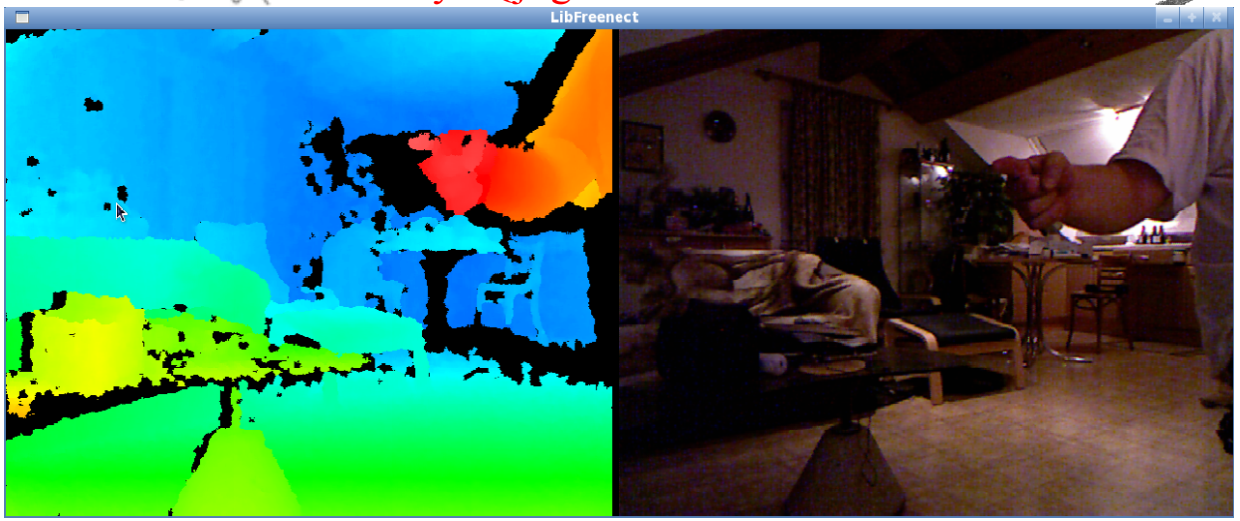
- Aseem Agarwala, Yuri Boykov, Vladimir Kolmogorov, Carsten Rother, Olga Veksler

Computer Vision Today?

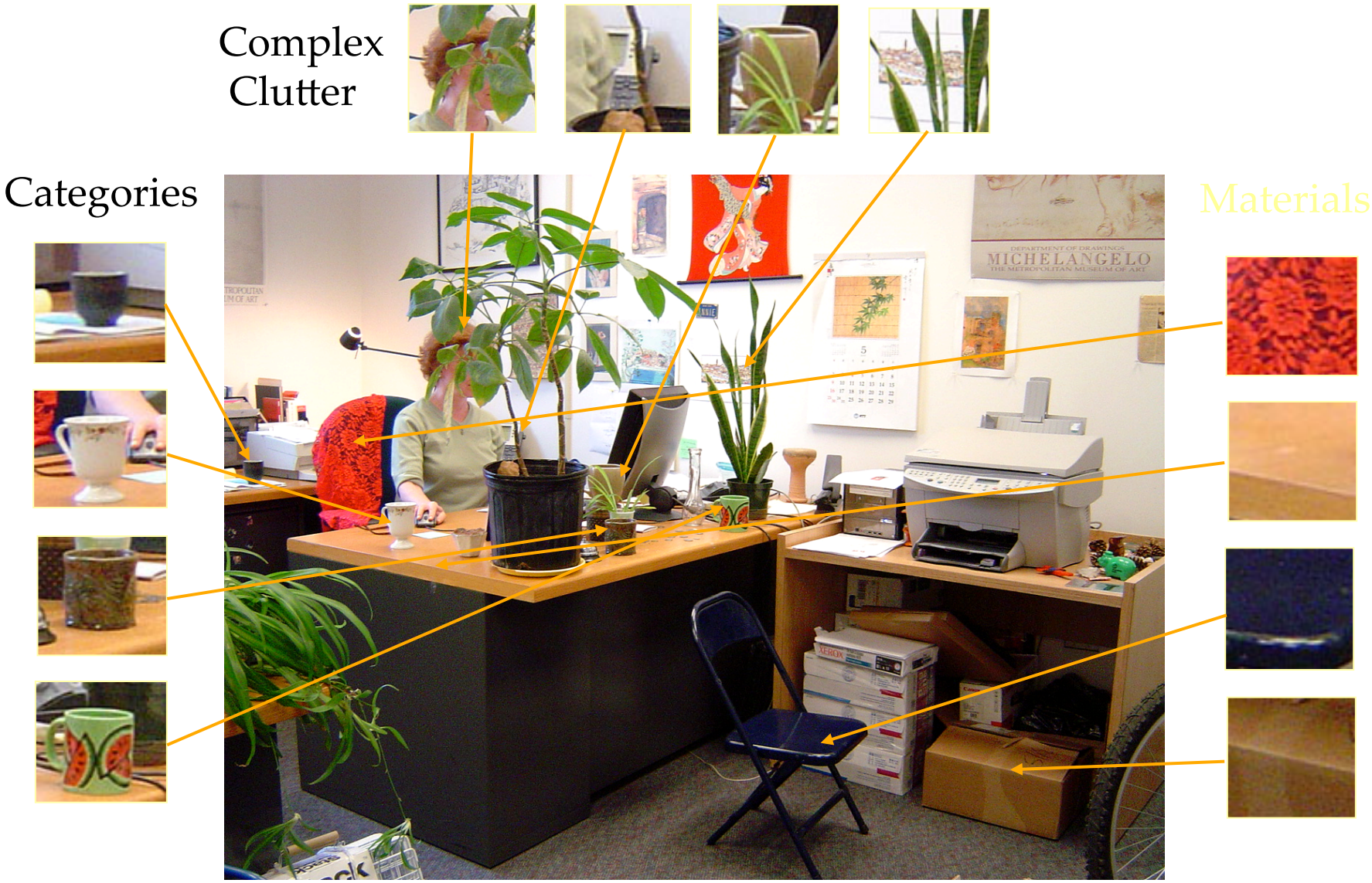




http://www.youtube.com/watch?v=MlTf0yYQjSg



Challenge: Understanding Complex Environments



Challenge: Highly Dynamic Environments

Recovering Geometry, Egomotion, Individual/Group Trajectories, and Activities

Cross-Cutting Challenges

"Semantic" representations

- Larger and larger data repositories, but little progress on making them "smart"

Large-scale verification of algorithms

- data repositories
- accepted evaluation methodologies

System integration

- almost no one has the resources to do it all and do it right

Facing the real world

- > 99% reliability
- manufacturable
- scalable

Conclusion

What you've learned this semester is the basis for a large and rich field

If you want to learn more, take the advanced vision seminar (offered by Prof. Vidal)

If you really want to learn more, join the CIRL lab or the Vision Lab (Prof. Vidal)

Final Project and Exam

Project due midnight Friday (5% late penalty/day thereafter)

You're free to arrange a time for demonstration – best times Monday or Tuesday after 3; you get 20 mins.

Examples of final reports have been sent around

- Try to include concrete test statistics
- Submit your code as well (does not need to be fully runnable)

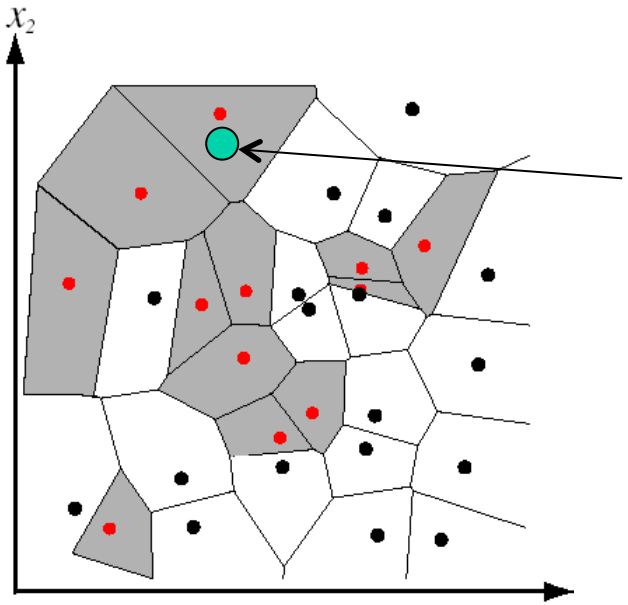
Final example will be administered through blackboard; look for an email early next week.

K-nearest neighbor

Nearest Neighbor classification

Assign label of nearest training data point to each test data point

Black = negative
Red = positive



from Duda *et al.*

Novel test example

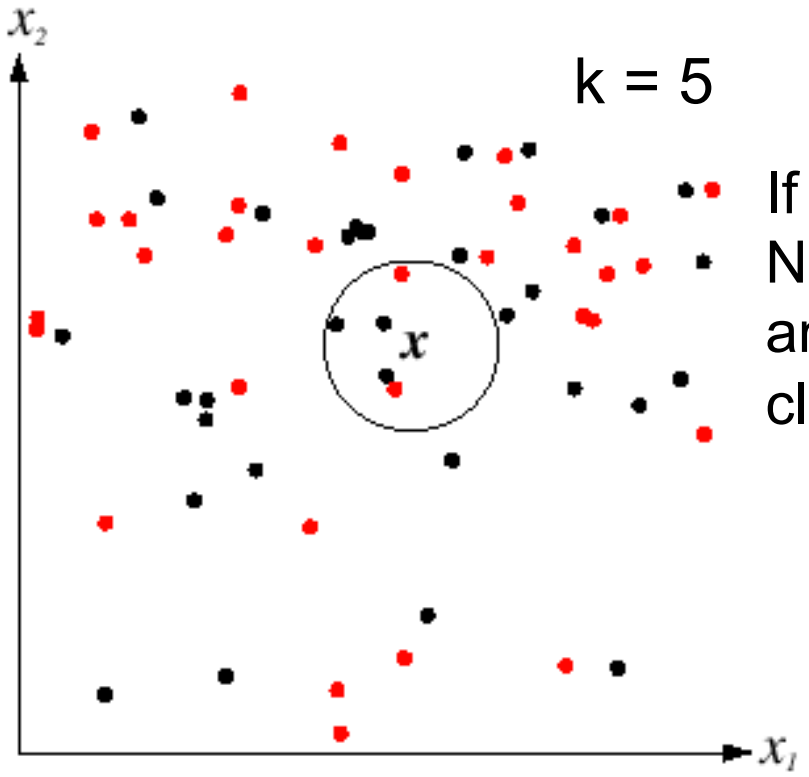
Closest to a positive example from the training set, so classify it as positive.

Voronoi partitioning of feature space
for 2-category 2D data

K-Nearest Neighbors classification

For a new point, find the k closest points from training data
Labels of the k points “vote” to classify

Black = negative
Red = positive



If query lands here, the 5 NN consist of 3 negatives and 2 positives, so we classify it as negative.

Source: D. Lowe

A nearest neighbor recognition example

Where in the World?

[Hays and Efros. **im2gps**: Estimating Geographic Information from a Single Image. CVPR 2008.]

Slides: James Hays

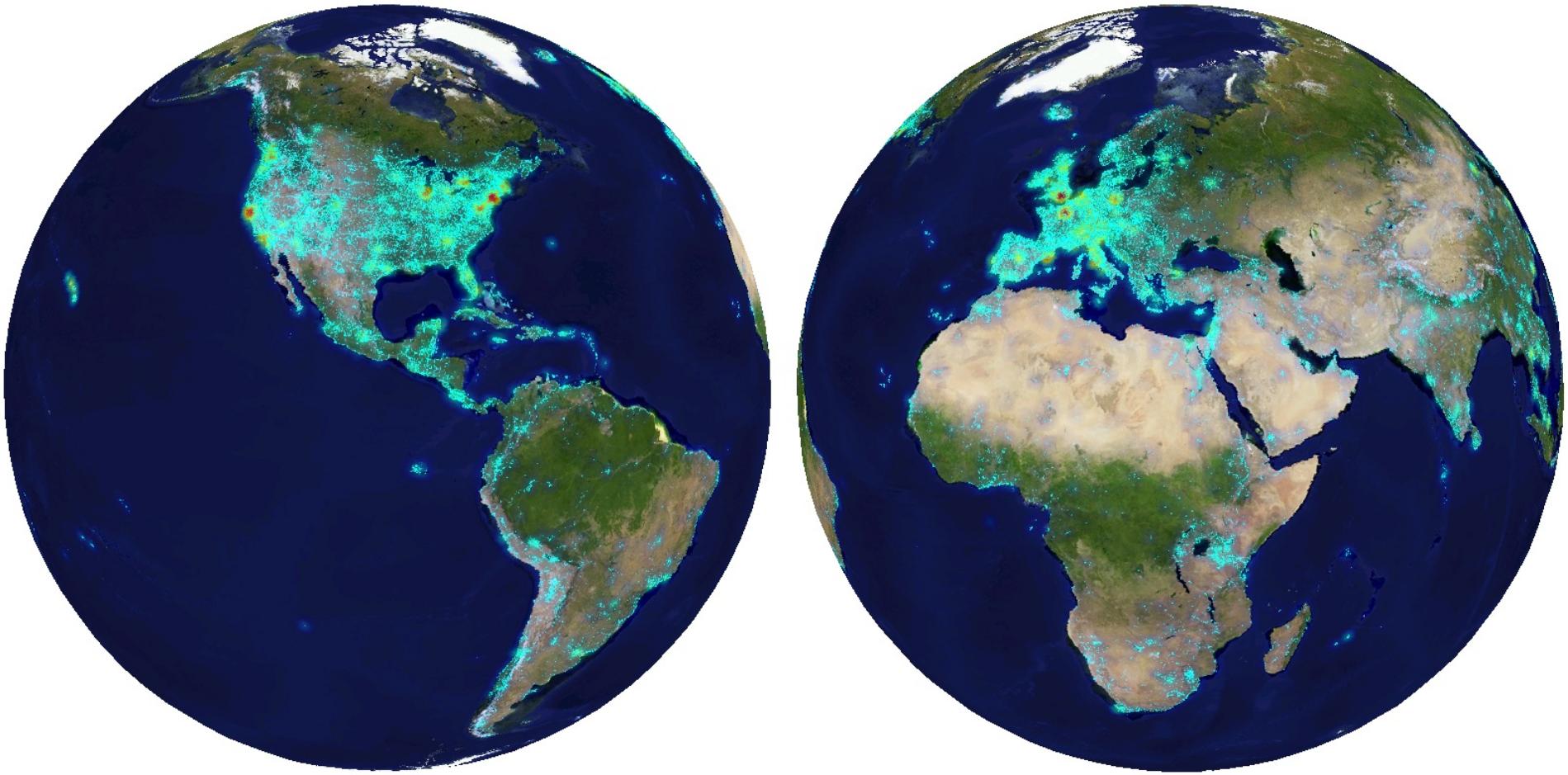
Where in the World?

Slides: James Hays

Where in the World?

Slides: James Hays

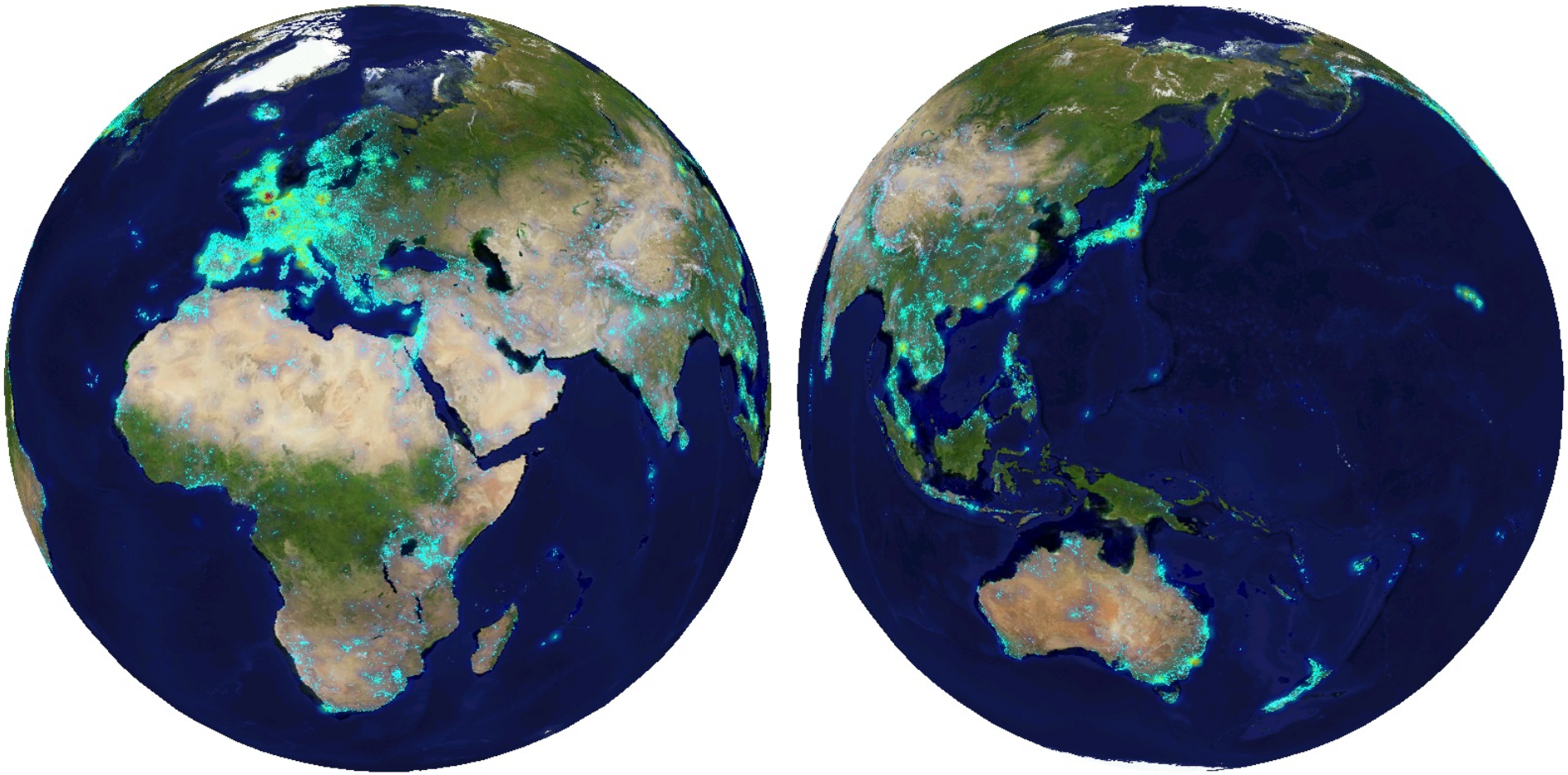
6+ million geotagged photos by 109,788 photographers



Annotated by Flickr users

Slides: James Hays

6+ million geotagged photos by 109,788 photographers



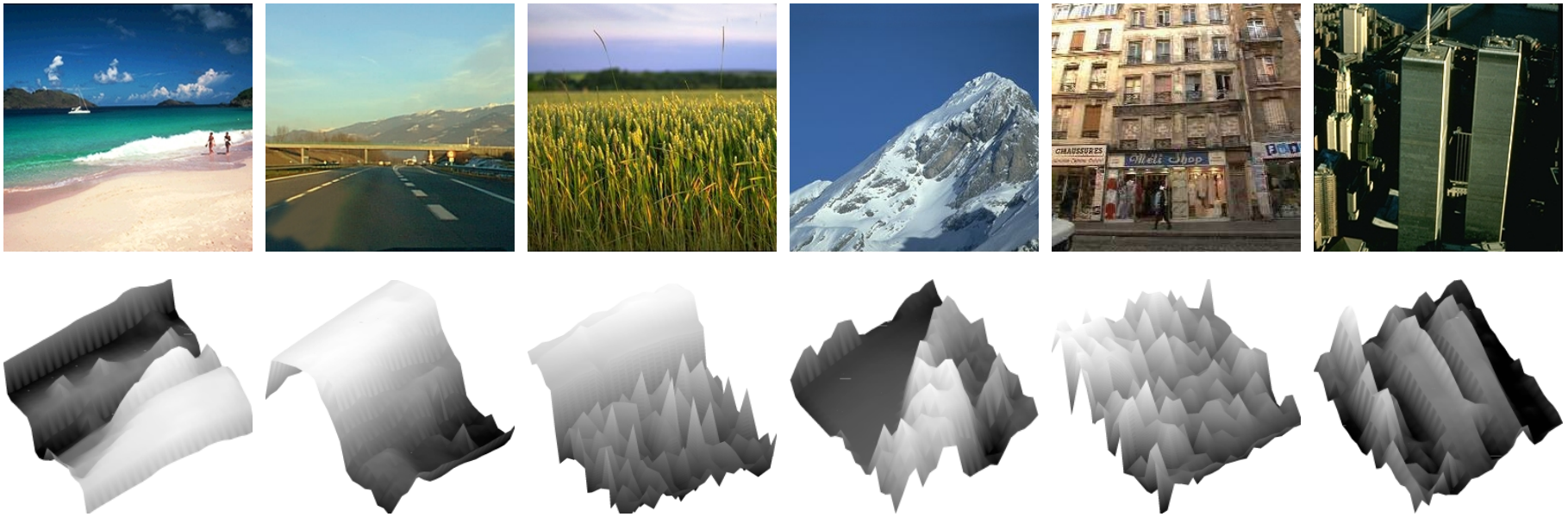
Annotated by Flickr users

Slides: James Hays

Which scene properties are relevant?

Spatial Envelope Theory of Scene Representation

Oliva & Torralba (2001)

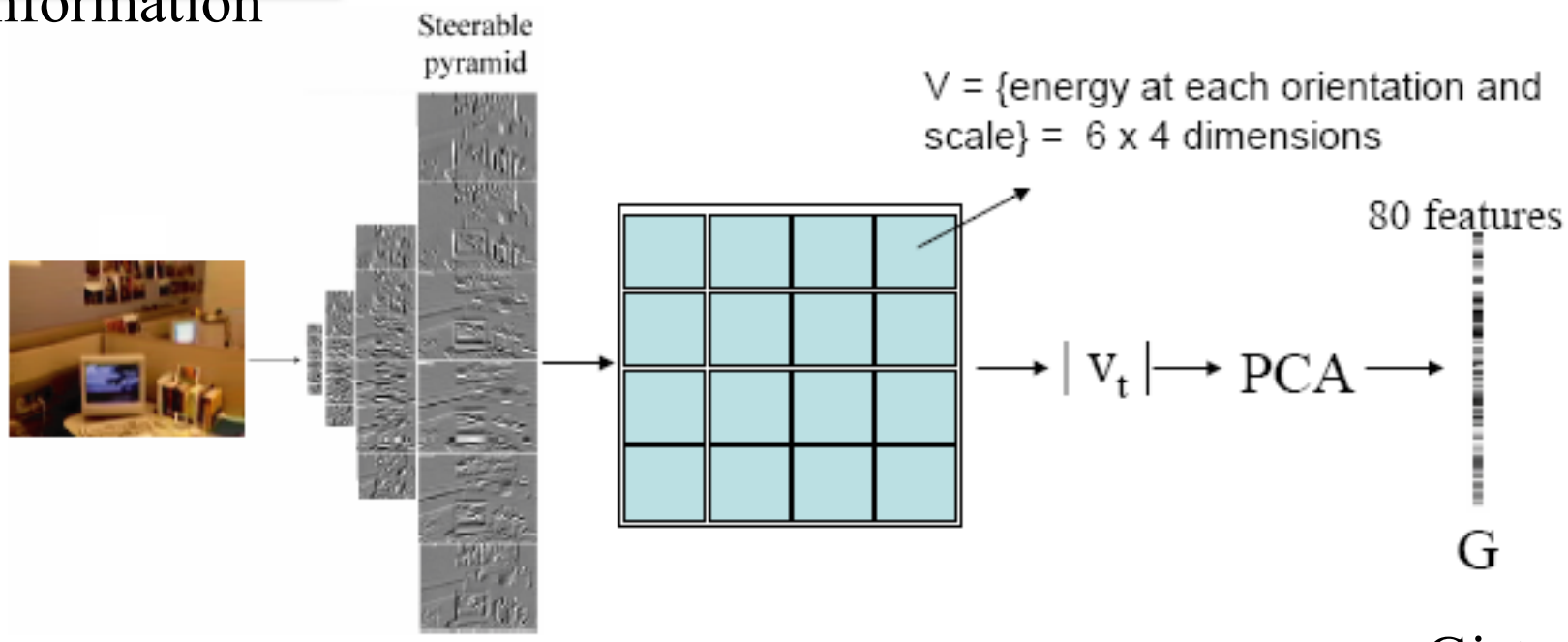


A scene is a single surface that can be represented by global (statistical) descriptors

Slide Credit: Aude Olivia

Global texture: capturing the "Gist" of the scene

Capture global image properties while keeping some spatial information



Gist descriptor

Oliva & Torralba IJCV 2001, Torralba et al. CVPR 2003

Which scene properties are relevant?

Gist scene descriptor

Color Histograms - CIE L*A*B* 4x14x14 histograms

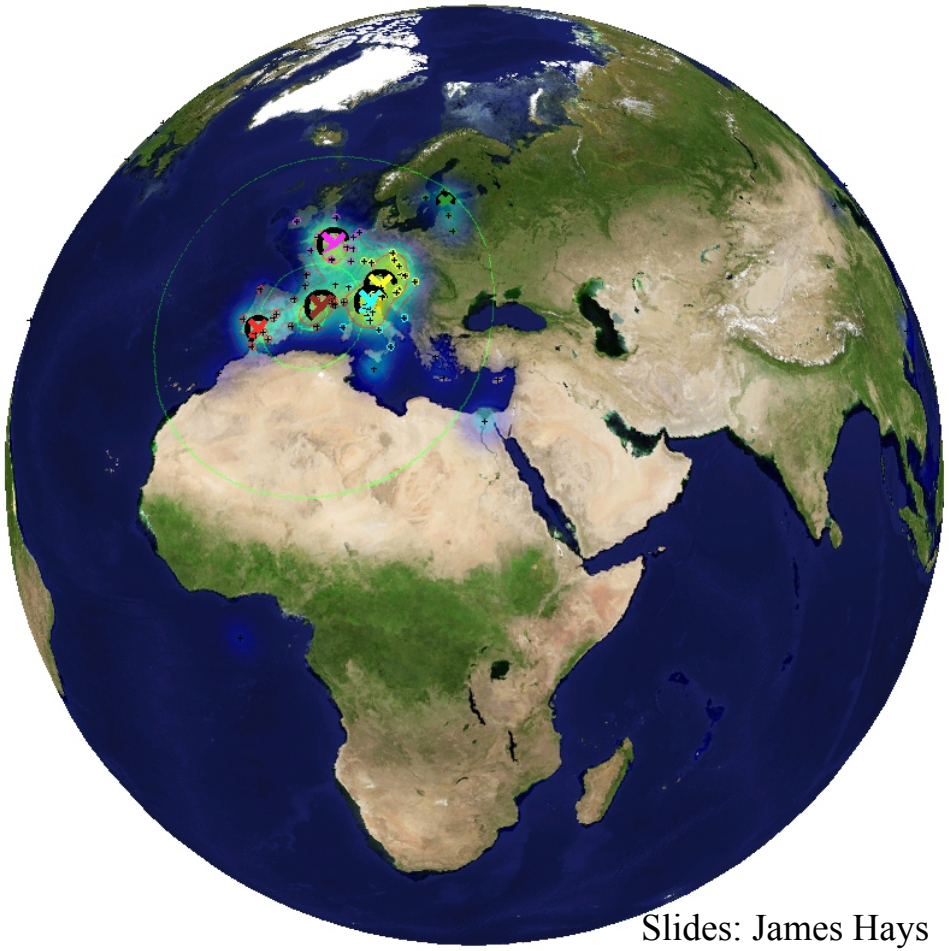
Texton Histograms – 512 entry, filter bank based

Line Features – Histograms of straight line stats

Scene Matches

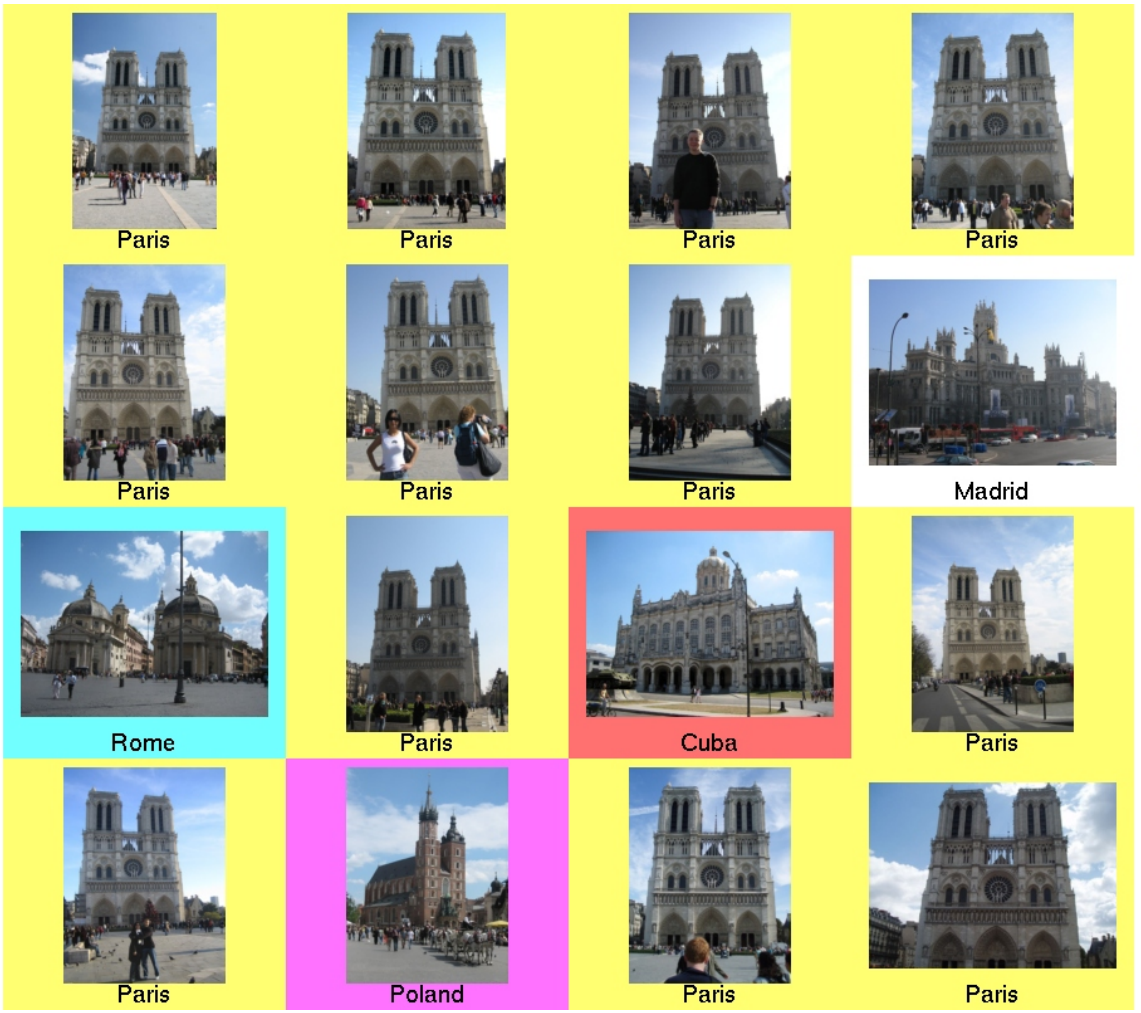
[Hays and Efros. **im2gps**: Estimating Geographic Information from a Single Image. CVPR 2008.]

Slides: James Hays



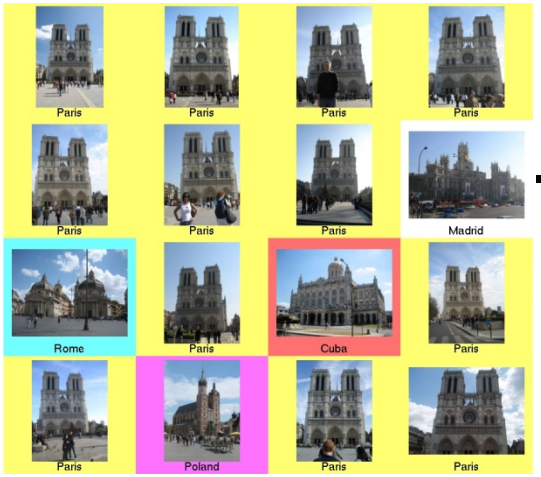
Slides: James Hays

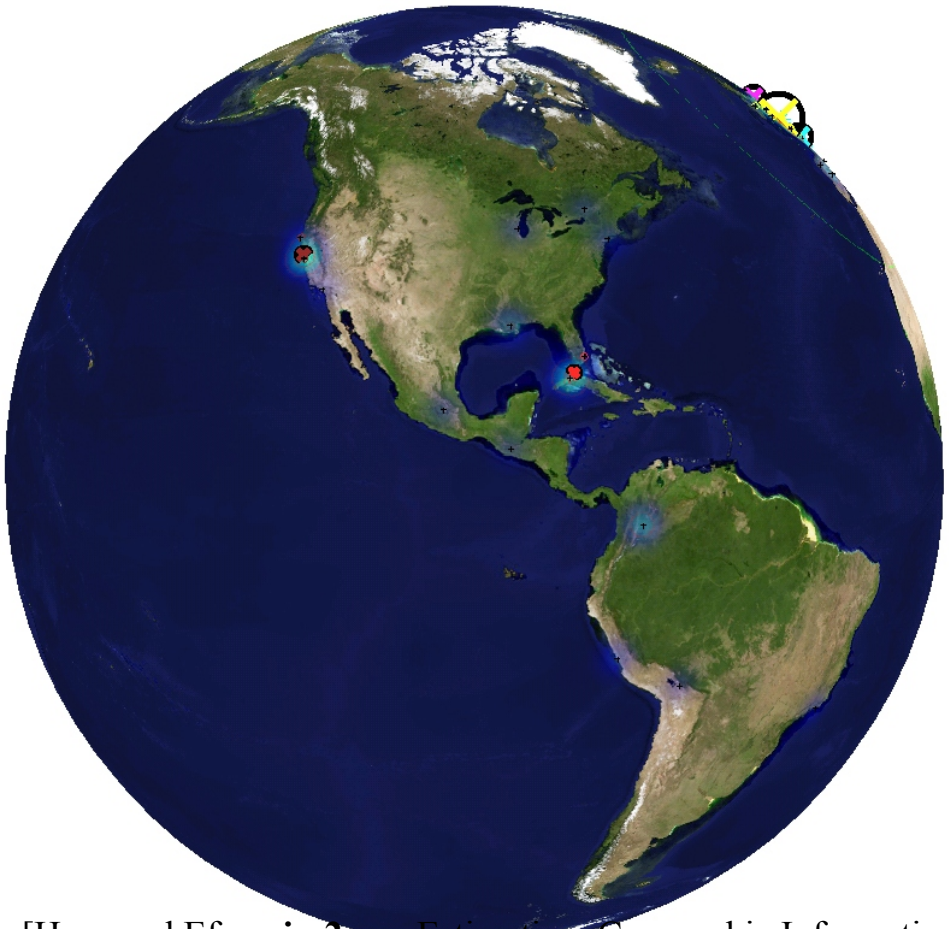
Scene Matches

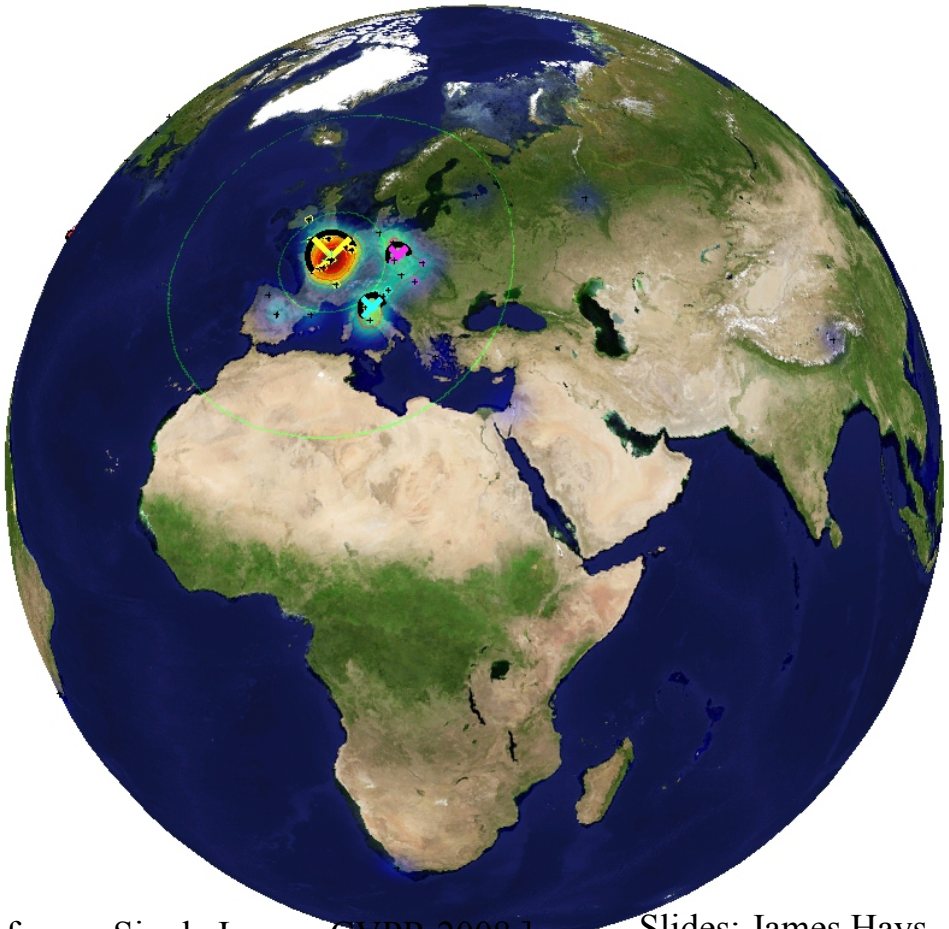


[Hays and Efros. **im2gps**: Estimating Geographic Information from a Single Image. CVPR 2008.]

Slides: James Hays







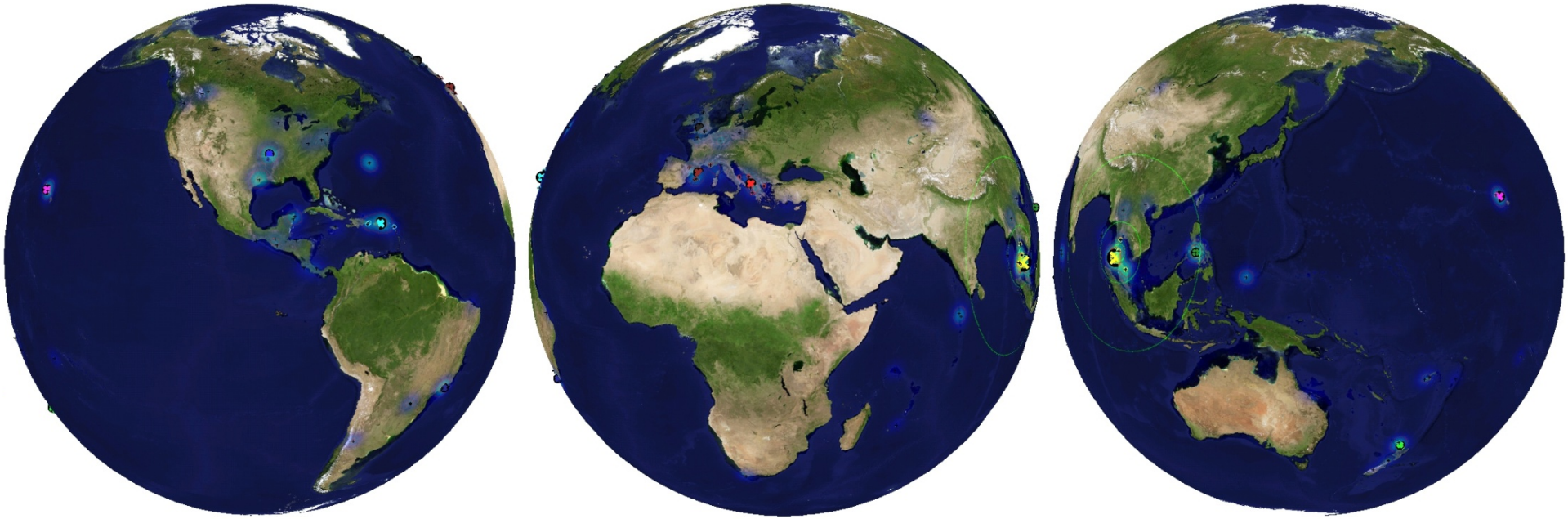
[Hays and Efros. **im2gps**: Estimating Geographic Information from a Single Image. CVPR 2008.]

Slides: James Hays

Scene Matches

[Hays and Efros. **im2gps**: Estimating Geographic Information from a Single Image. CVPR 2008.]

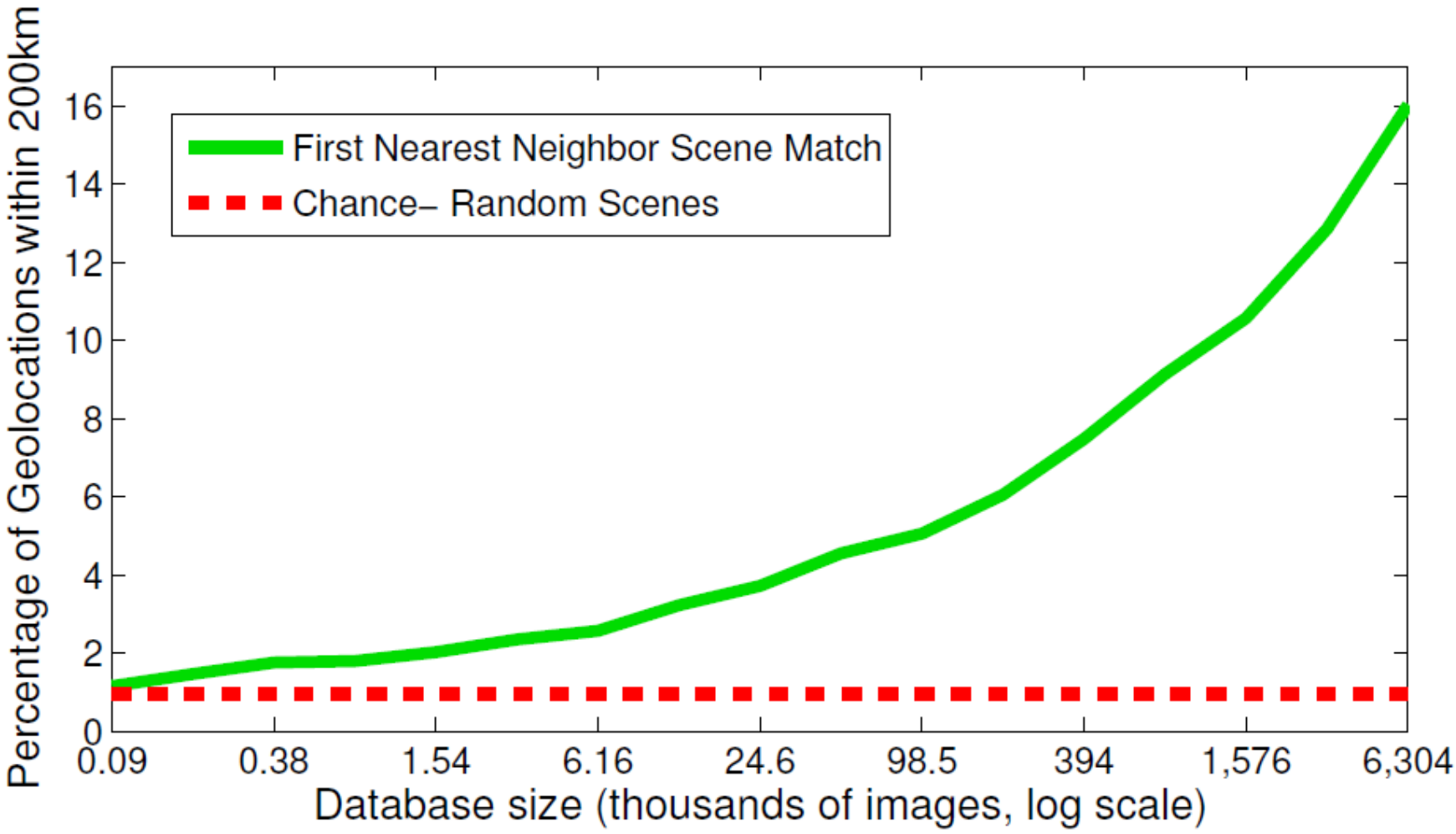
Slides: James Hays



[Hays and Efros. **im2gps**: Estimating Geographic Information from a Single Image. CVPR 2008.]

Slides: James Hays

The Importance of Data



[Hays and Efros. **im2gps**: Estimating Geographic Information from a Single Image. CVPR 2008.]

Slides: James Hays

Nearest neighbors: pros and cons

Pros:

- Simple to implement
- Flexible to feature / distance choices
- Naturally handles multi-class cases
- Can do well in practice with enough representative data

Cons:

- Large search problem to find nearest neighbors
- Storage of data
- Must know we have a meaningful distance function

Conclusion

- Today:
  - Segmentation
    - Mixtures of Gaussians
    - Mean-shift
    - Normalized cuts
  - Classification
    - K-nearest-neighbor

- Thursday
  - Support vector machines