Topic 15:

Image Segmentation

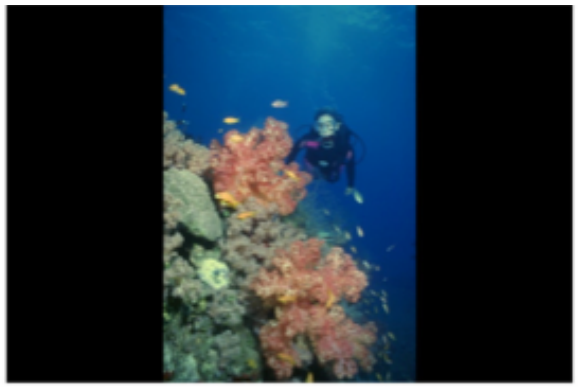
- intro to segmentation
- the affinity matrix
- spectral methods
- efficient graph-based methods
- density estimation methods
- a deeper look at affinity functions

what is a segment?

Plausible answers

- "You know it when you see it"
- "It depends on the task"
- "Segments are objects"

One of the key challenges has been to operationalize this definition!



human-centered approach

human-centered approach

People are remarkably consistent in their boundary annotations

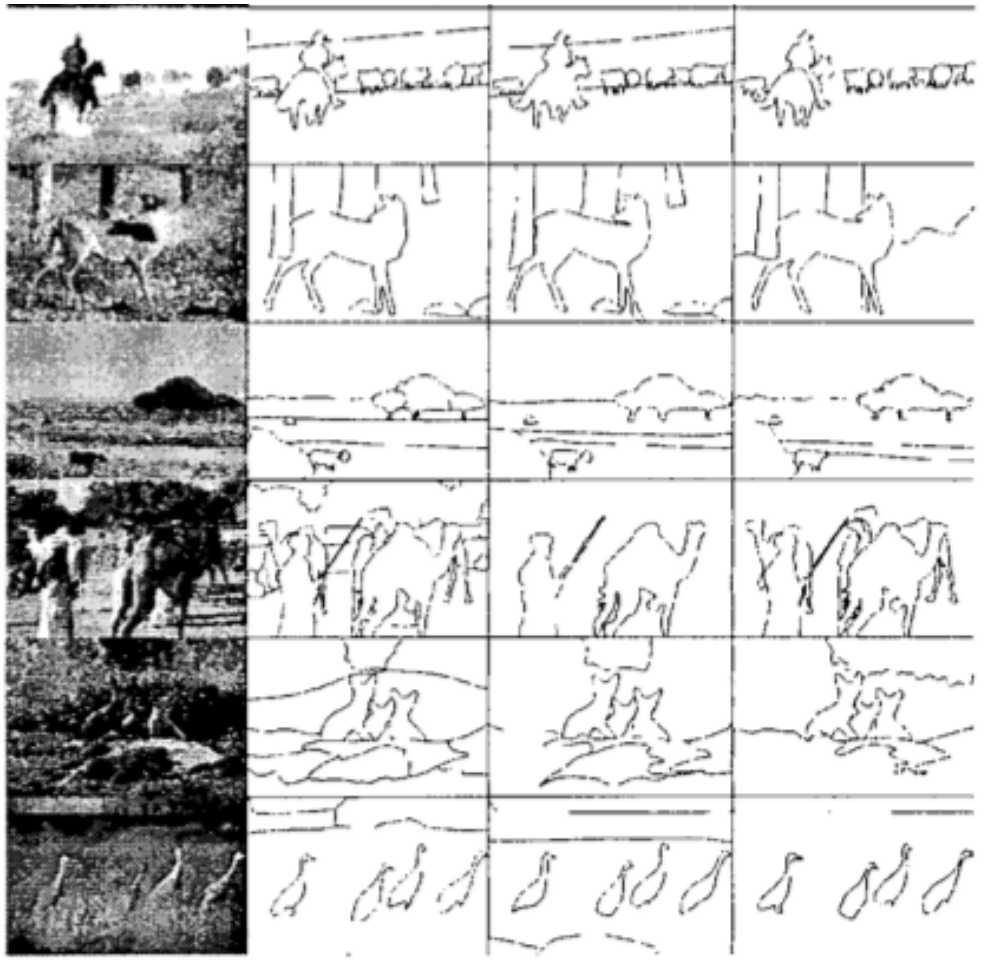


Figure 1: Sample of 10 images from the segmentation database. Each image has been segmented by 3 different people. A total of 10 people are represented in this data.

(Martin et al, ICCV 2001)

human-centered approach

People are remarkably consistent in their boundary annotations

... although they differ in the level of detail specified

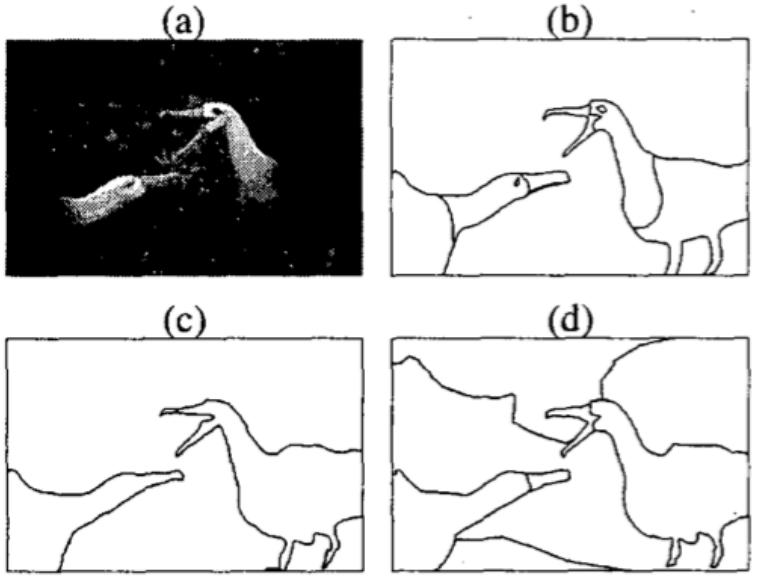


Figure 3: Motivation for making segmentation error measures tolerant to refinement. (a) shows the original image. (b)-(d) show three segmentations in our database by different subjects. (b) and (d) are both *simple refinements* of (c), while (b) and (d) illustrate *mutual refinement*.

↑ not one right answer but a hierarchy of them

(Martin et al, ICCV 2001)

task-centered approach

autonomous navigation

- pedestrians
- signs
- obstacles

medical imaging

- a tumor

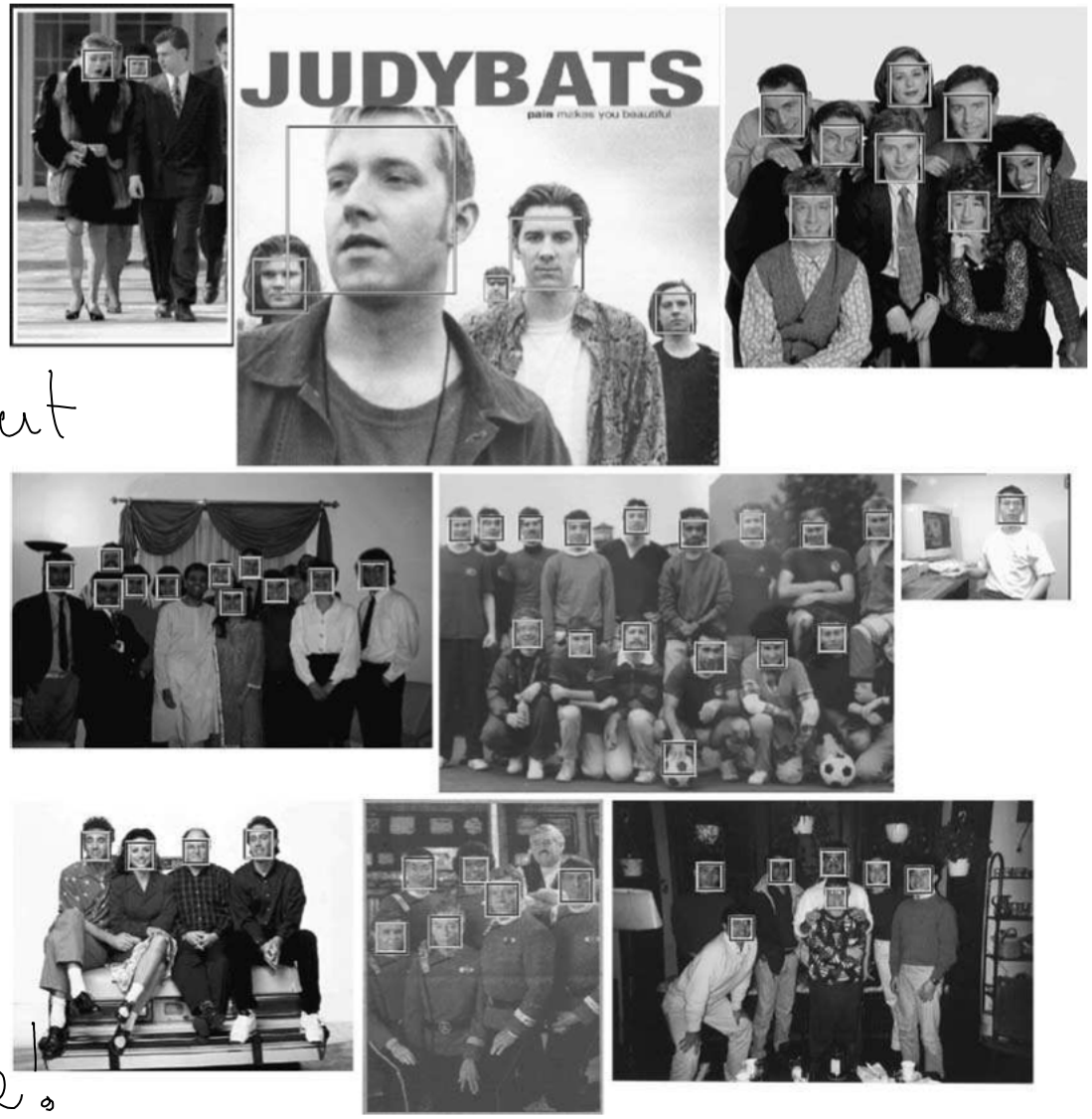
powerpoint

- objects/segments specified interactively

what comes first? recognition or segmentation?

what comes first? recognition or segmentation?

it is certainly possible to train object detectors that operate without explicit boundary detection



this issue is still a matter of considerable debate.

(Viola & Jones, IJCV 04)

many other quandaries here...

- bottom-up or top-down ?
  (prior to rec) (after/during rec)
- "hard" or "soft" segments ?
  (unique assigment of pixels to segments) ("fuzzy" assignment)
- supervised or unsupervised ?
  (relies on training data)
- flat or hierarchical ?
- interactive or automatic ?

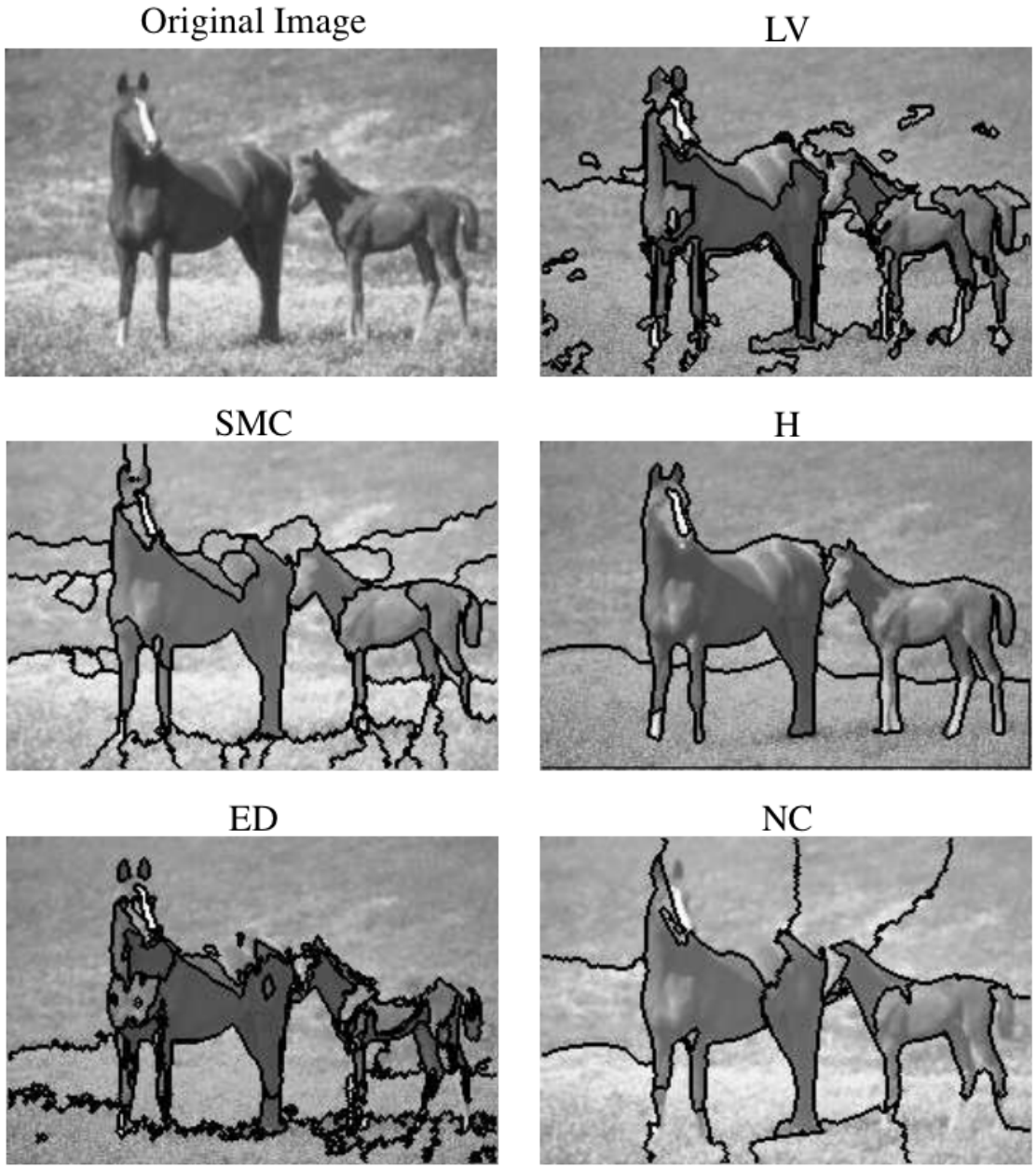
our focus

We consider **bottom-up image segmentation**. That is, we ignore (top-down) contributions from object recognition in the segmentation process.

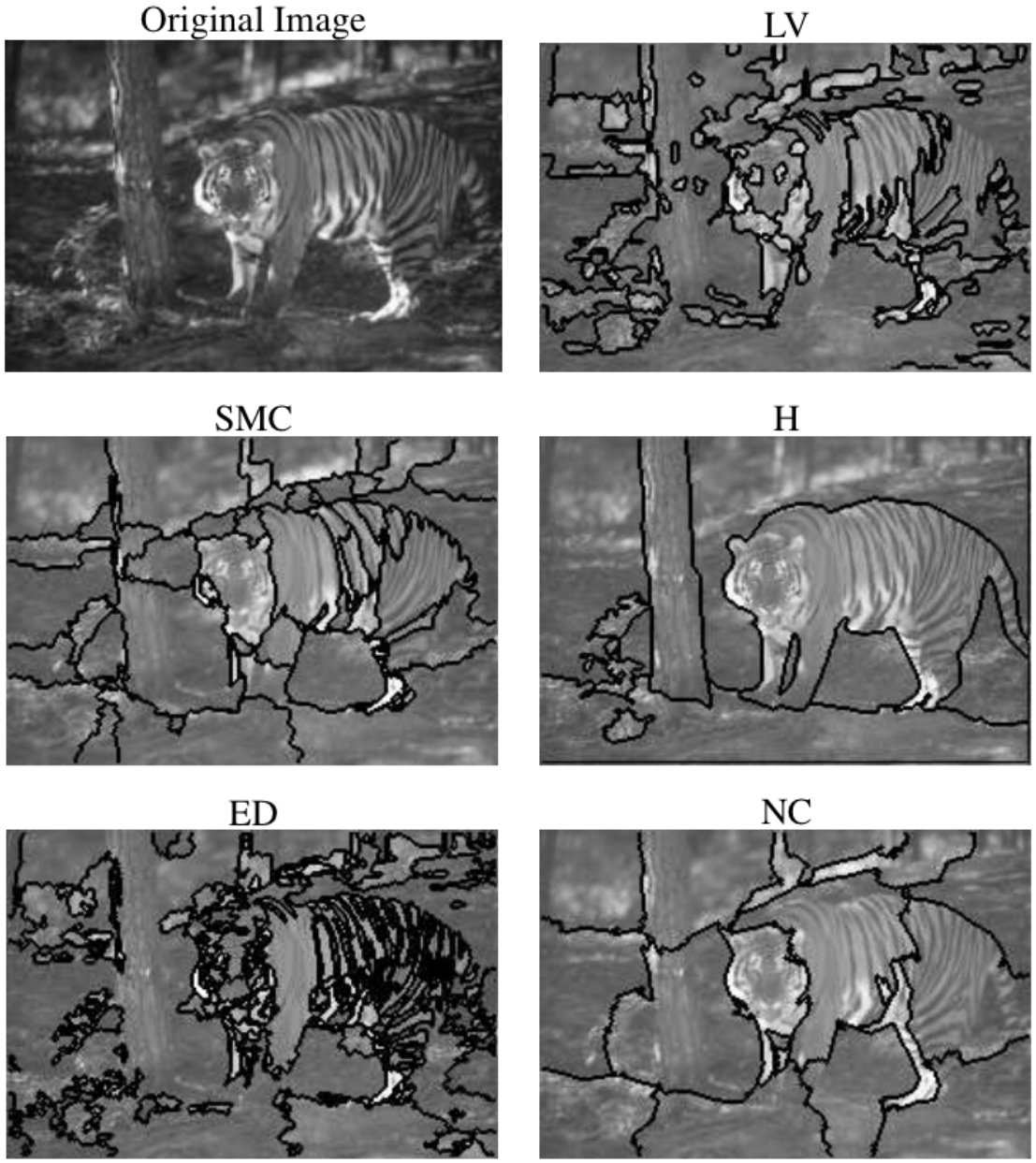
For input we primarily consider image brightness here, although similar techniques can be used with colour, motion, and/or stereo disparity information.

- the techniques we will cover should be viewed as core building blocks for designing more powerful algorithms
- we will discuss one such notable example later today

example segmentations: which one is best?



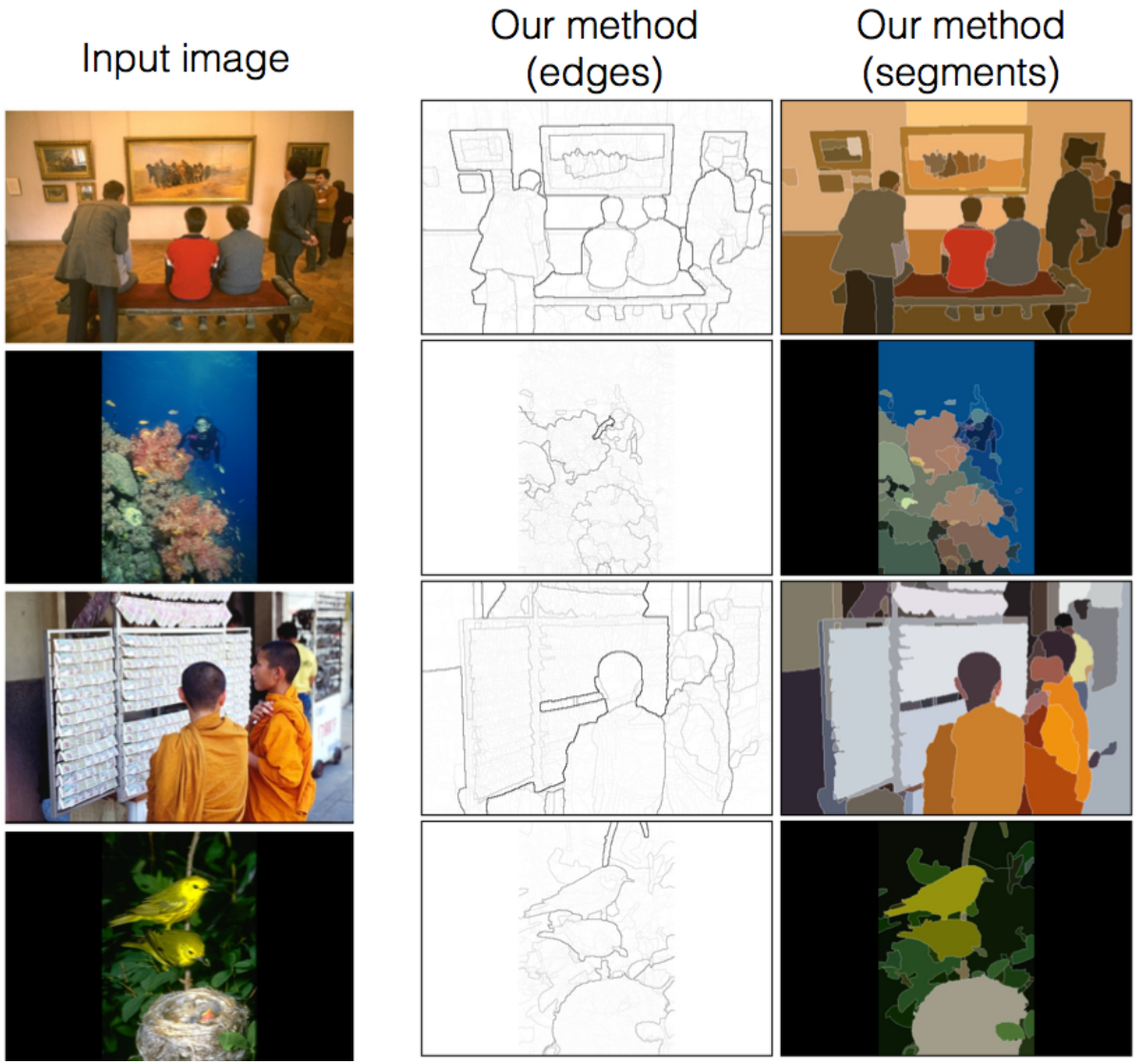
example segmentations: which one is best?



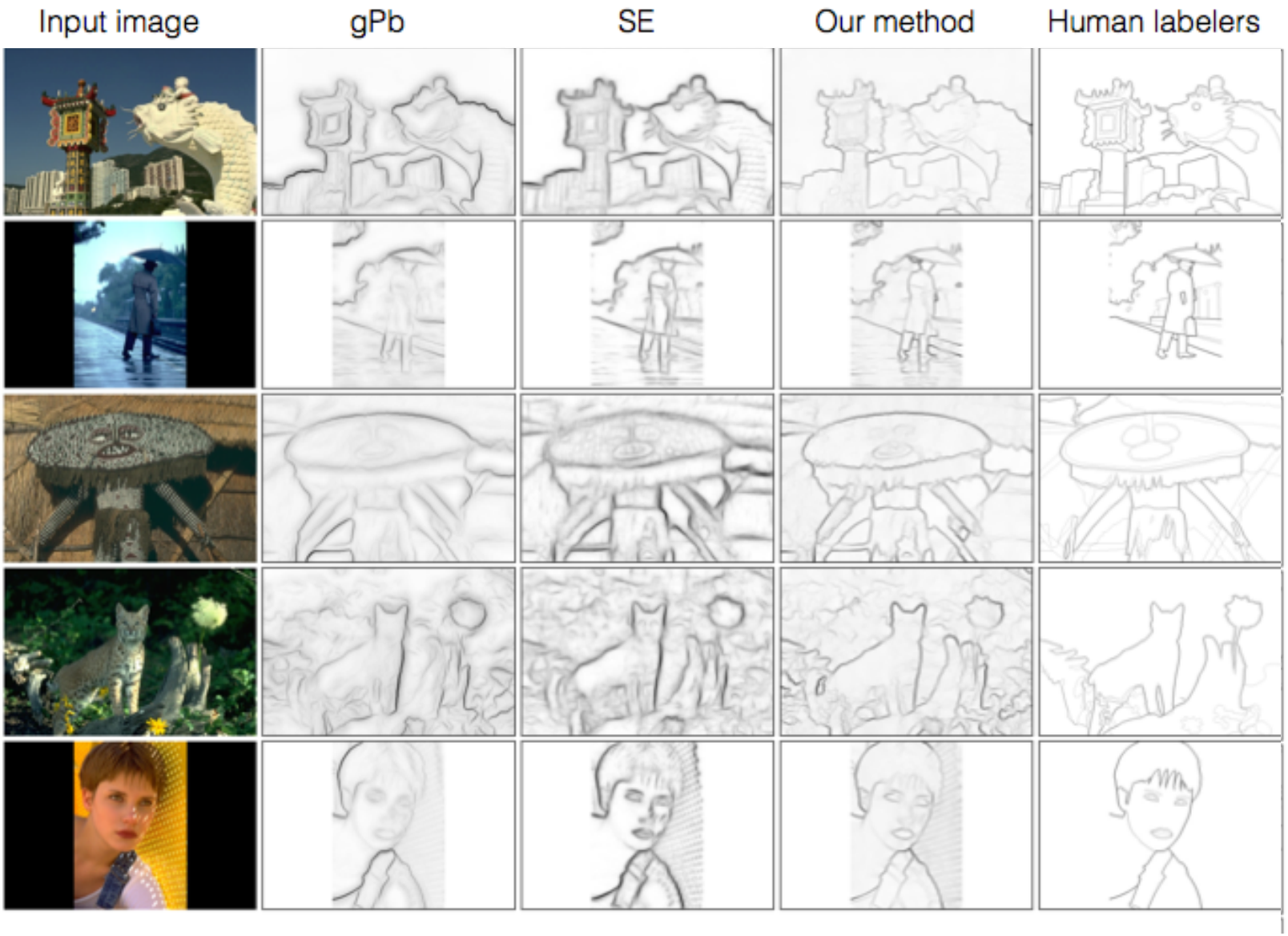
example segmentations: remarks

- The quality of the segmentation depends on the image. Smoothly shaded surfaces with clear gray-level steps between different surfaces are ideal for the above algorithms.
- For simple images (p. 2) it is plausible that machine segmentations like those above are useful for visual tasks, e.g., object recognition.
- For more complex images (pp. 5, 6), machine segmentations provide a less reliable indicator for surface boundaries, and their utility for subsequent processing is questionable.
- While many algorithms work well with simple images, they break down with clutter and camouflage. The assessment of segmentation algorithms therefore needs to be done on standardized datasets.
- Humans probably use object recognition in conjunction with segmentation, while the machine algorithms above do not.

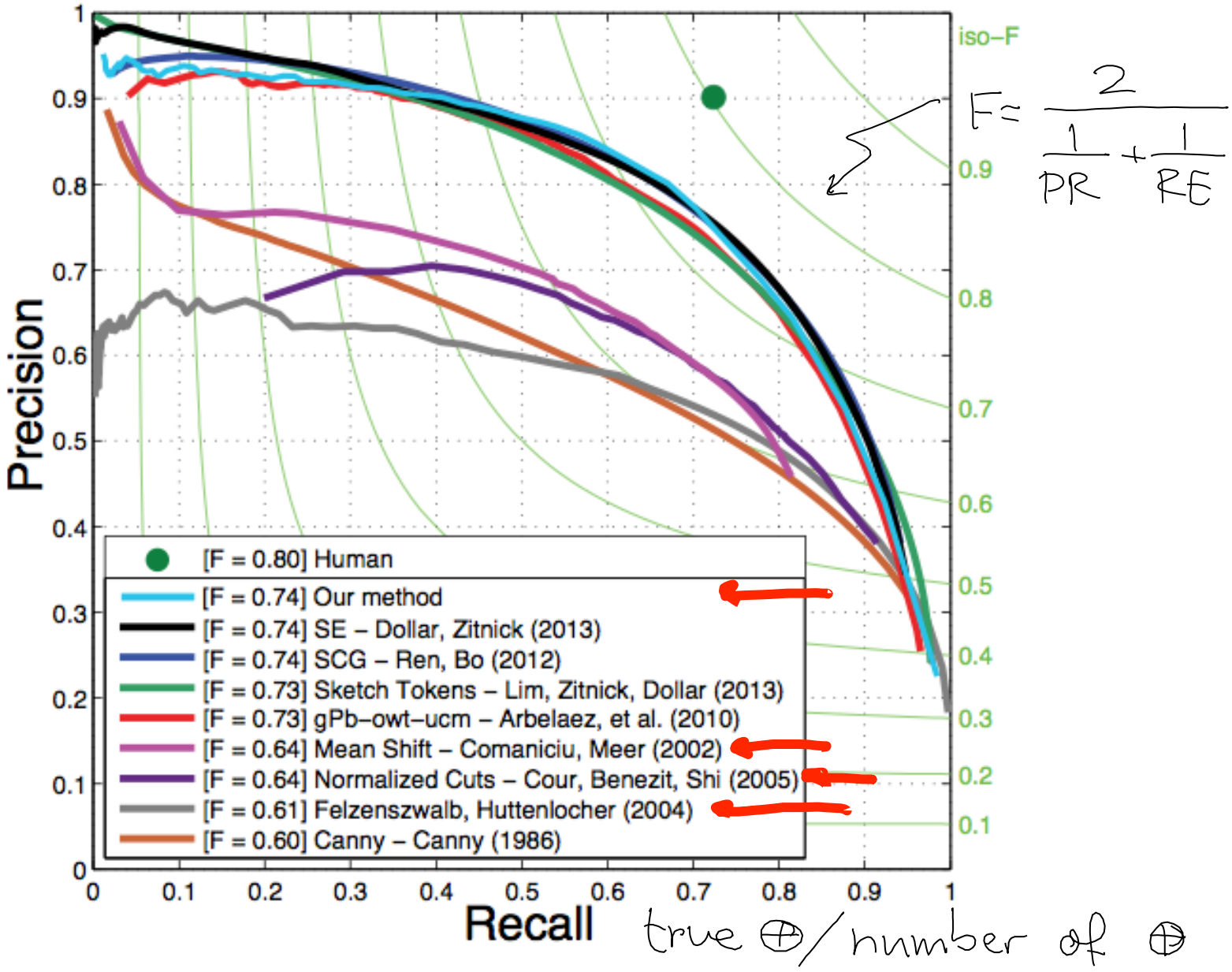
the Berkeley Segmentation Dataset (Martin et al)



evaluating on BSDS



quantitative evaluation on BSDS500



$F = \frac{2}{\frac{1}{PR} + \frac{1}{RE}}$

$\frac{\text{true } \oplus}{\text{classified } \oplus}$

true ⊕ / number of ⊕

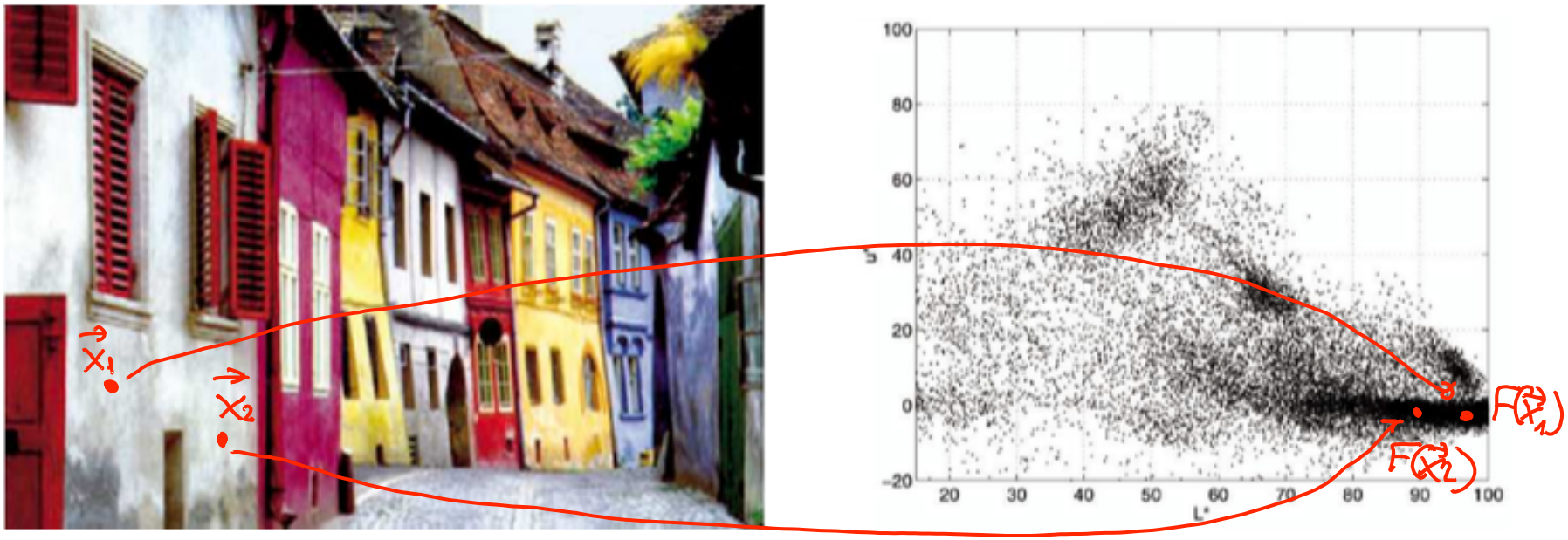
Topic 15:

Image Segmentation

- intro to segmentation
- the affinity matrix
- spectral methods
- efficient graph-based methods
- density estimation methods
- a deeper look at affinity functions

segmentation as feature-based clustering

From Comaniciu and Meer [5].



Image

Scatter plot (2D L*u*)

General approach:

1. Assign each pixel $\vec{x}_i$ to a "feature vector" $F(\vec{x}_i)$
2. Group pixels based on feature vector similarity

assigning feature vectors to pixels

Given an image $I(\vec{x})$, consider feature vectors $\vec{F}(\vec{x})$ of the form

$$\vec{F}(\vec{x}) = \begin{pmatrix} \vec{x} \\ I(\vec{x}) \\ \vec{L}(\vec{x}) \end{pmatrix}$$

image position

intensity filter responses

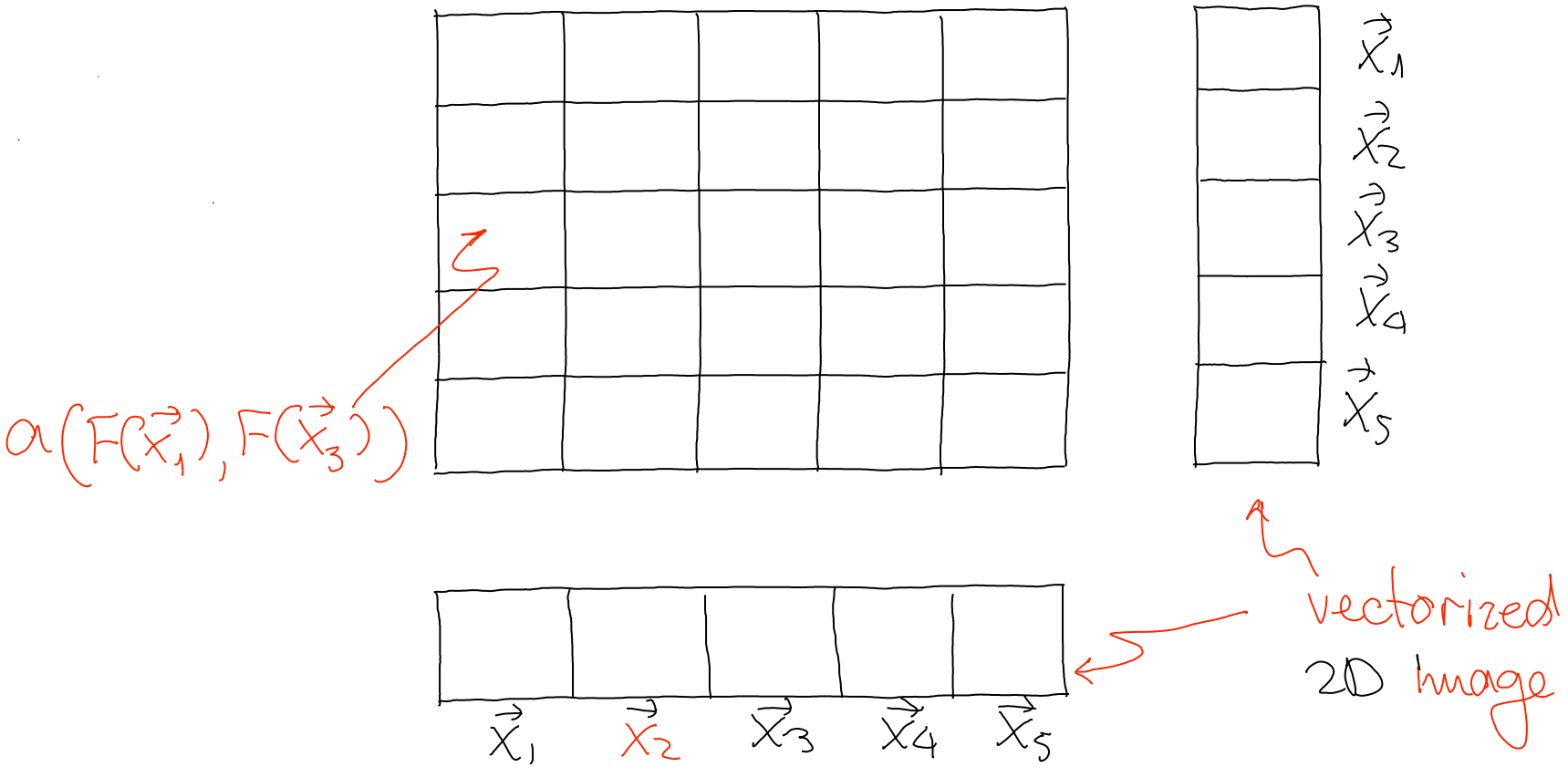
color filter responses

$\vec{L}(\vec{x})$ is a vector of local image features, perhaps bandpass filter responses. For colour images, $\vec{F}(\vec{x})$ would also include information about the colour at pixel $\vec{x}$.

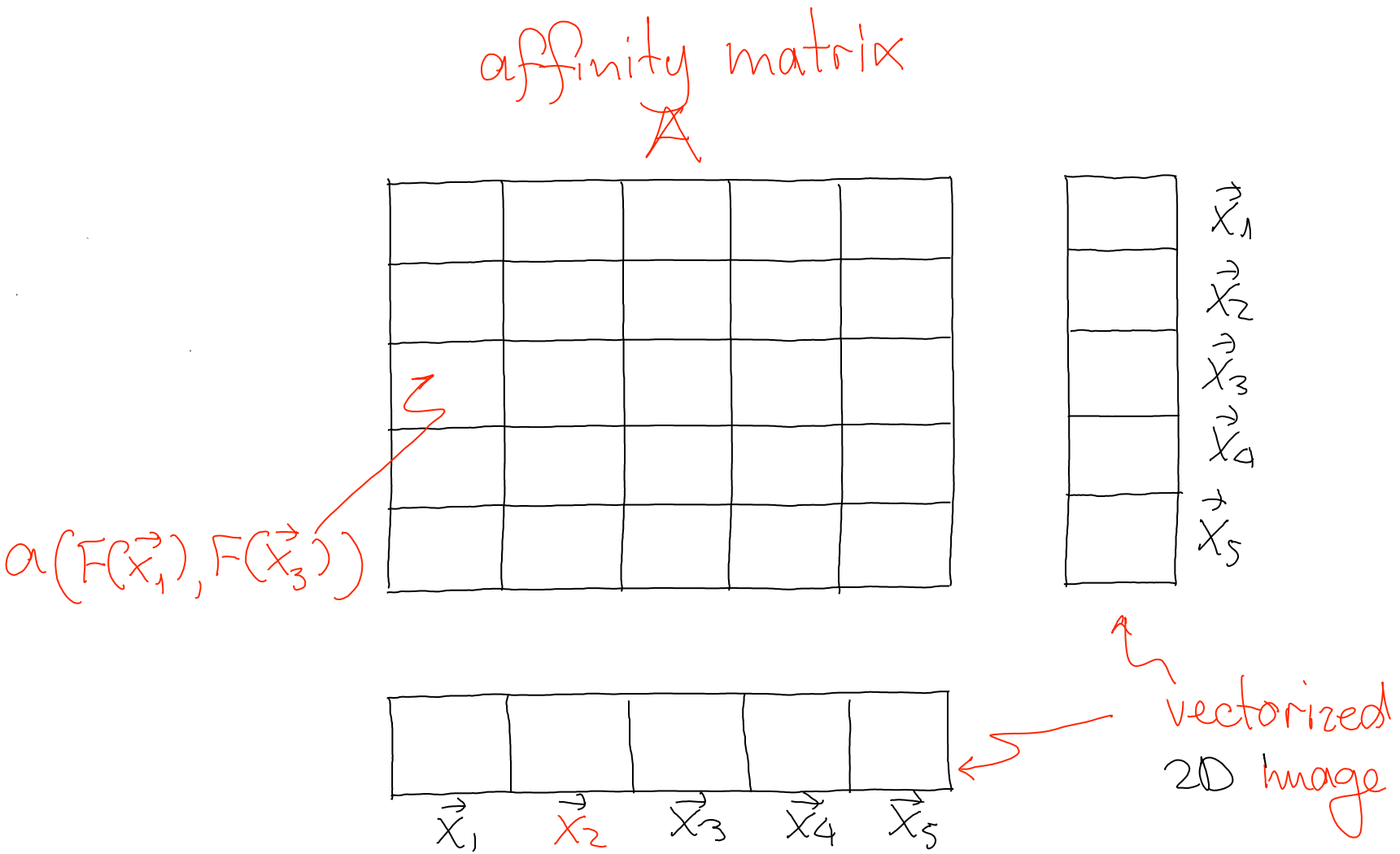
To segment the image we might seek a clustering of the feature vectors $\vec{F}(\vec{x})$ observed in that image. A compact region of the image having a distinct gray-level or colour will correspond to a region in the feature space with a relatively high density of sampled feature vectors.

affinity function between pairs of pixels

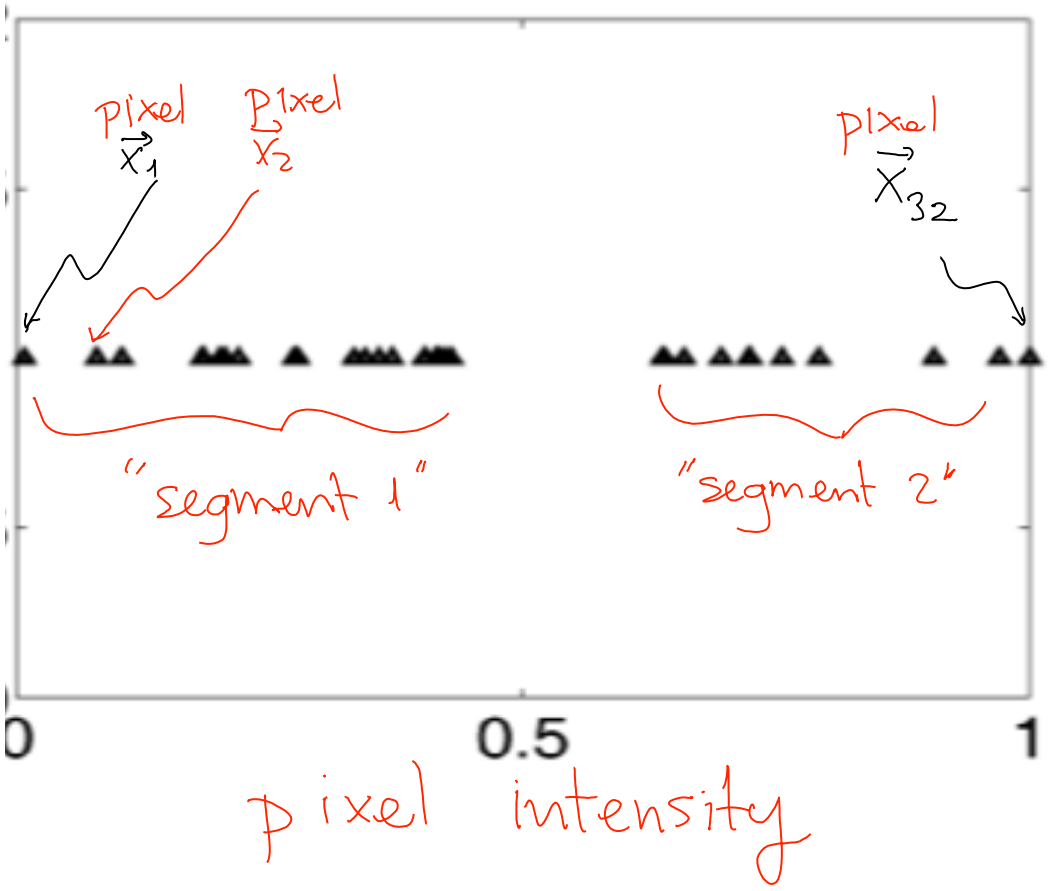
A function $a(\,) \geq 0$ that assigns a similarity score to a given pair of feature vectors

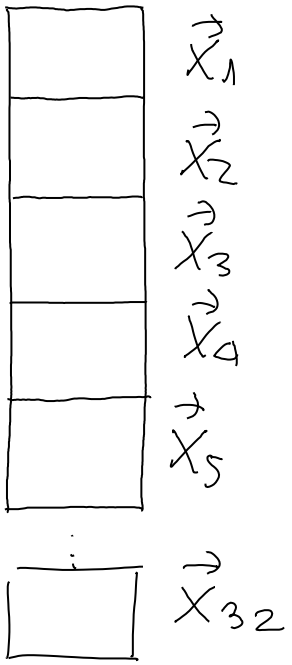


the affinity matrix A

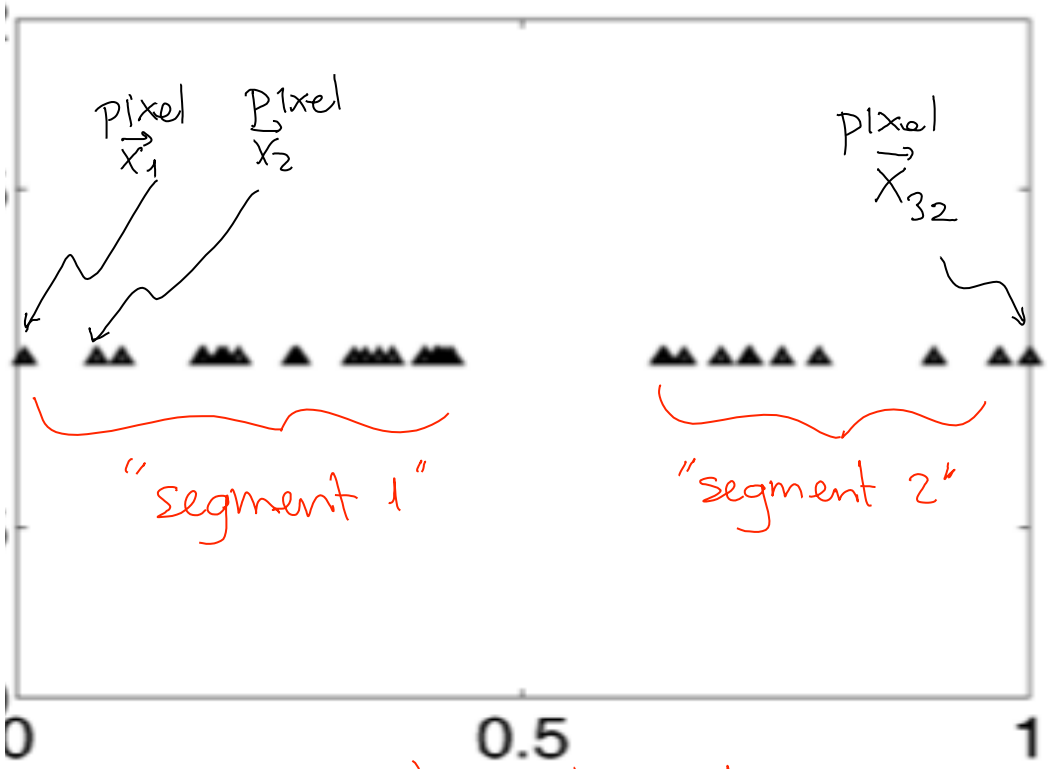


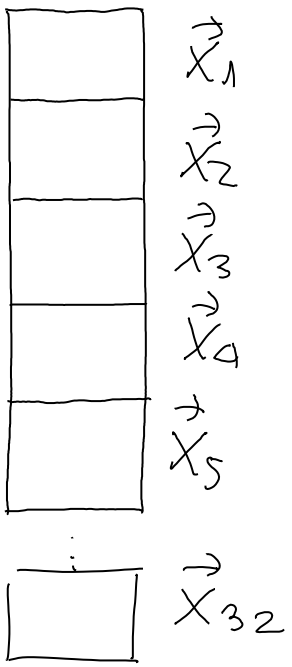
example: 1D image





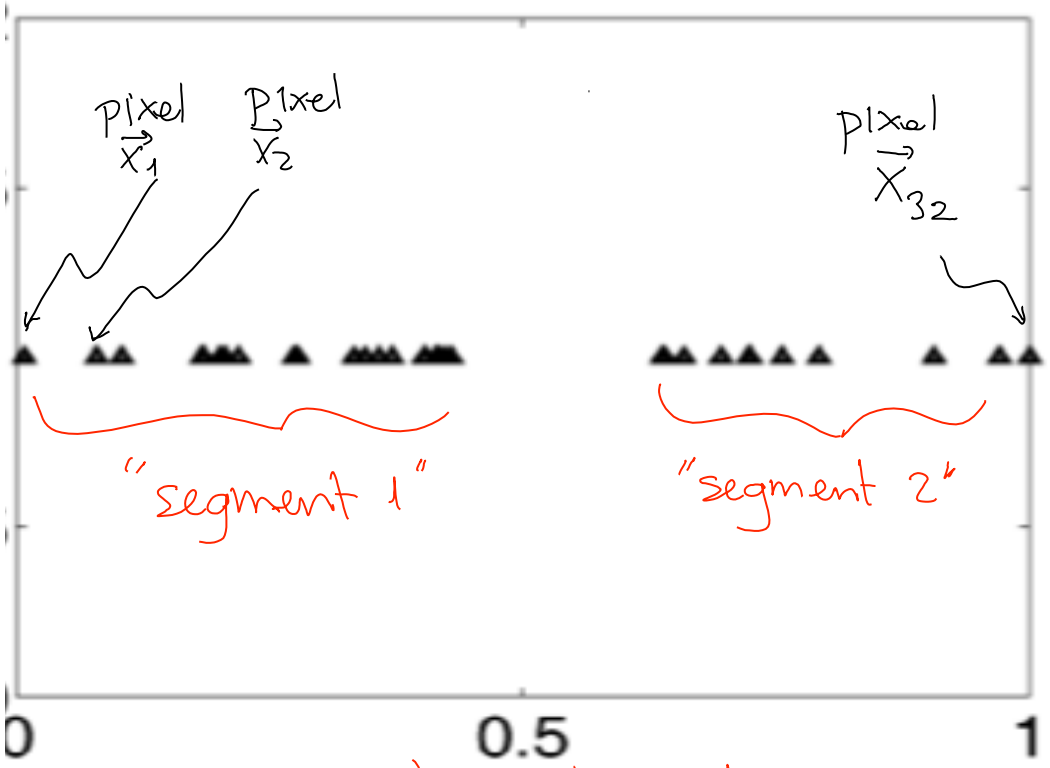
example: 1D image, intensity feature

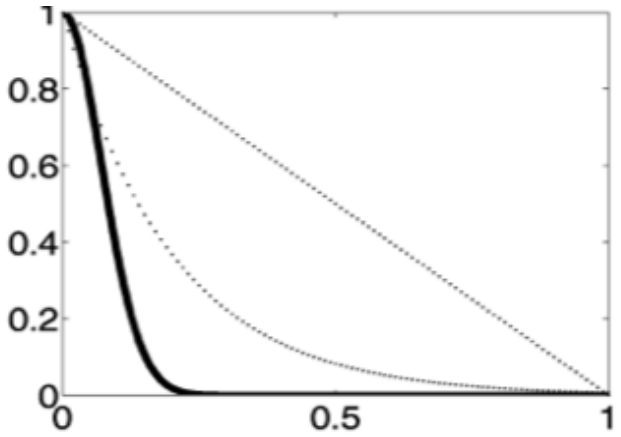




$$F(\vec{x}_i) = I(\vec{x}_i)$$

1D image, intensity feature, exponential affinity





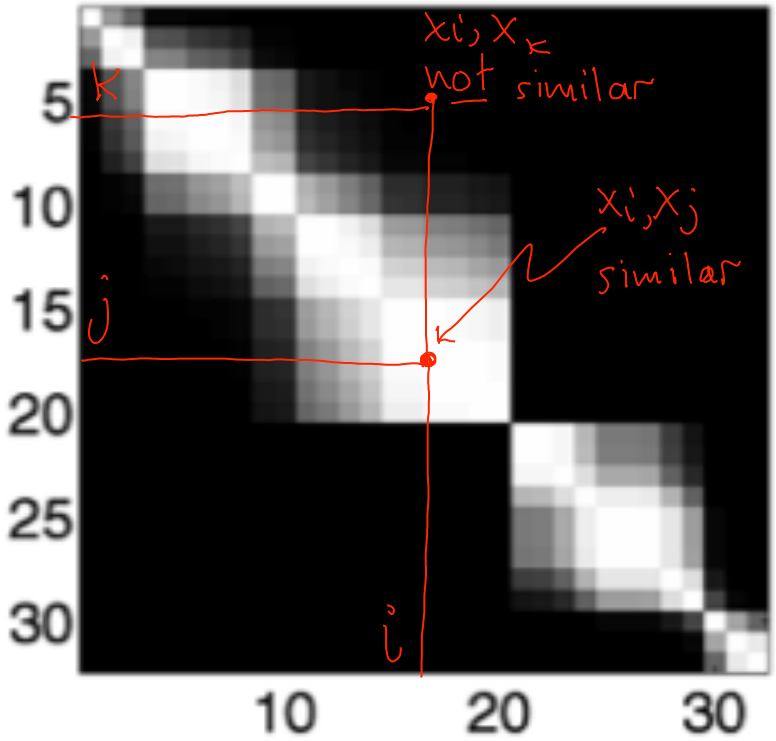
pixel intensity

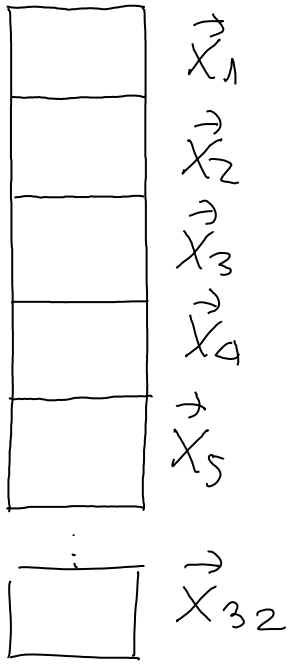
$F(\vec{x}_i) = I(\vec{x}_i)$

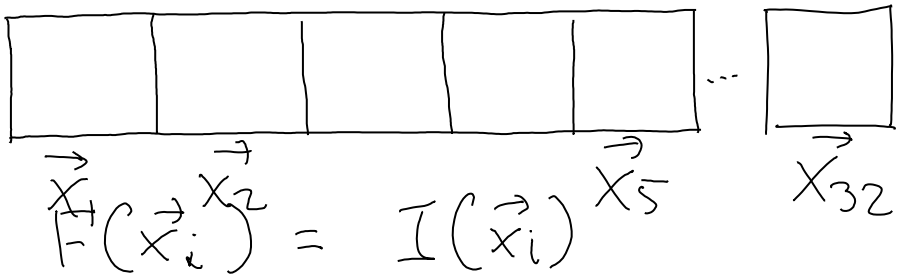
affinity function

$$a(F_i, F_j) = e^{-\left(\frac{F_i - F_j}{0.1}\right)^2}$$

the affinity matrix







Topic 15:

Image Segmentation

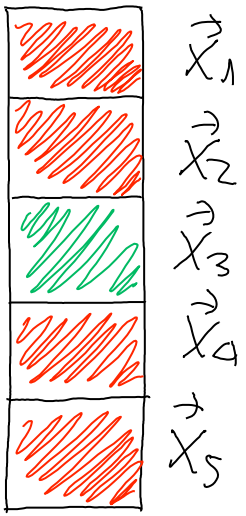
- intro to segmentation
- the affinity matrix
- spectral methods
  - normalized cuts for hard segmentation
  - spectral matting for computing alpha mattes
- efficient graph-based methods
- density estimation methods
- a deeper look at affinity functions

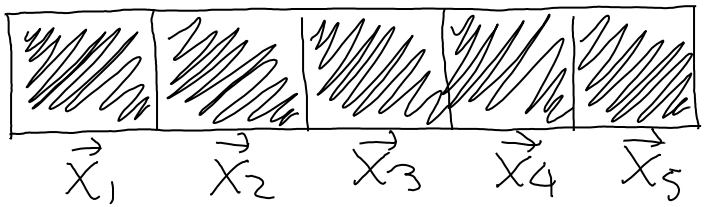
the affinity matrix A

Consider a noiseless image with two segments and a "perfect" affinity function

A

1	1	0	0	0
1	1	0	0	0
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1



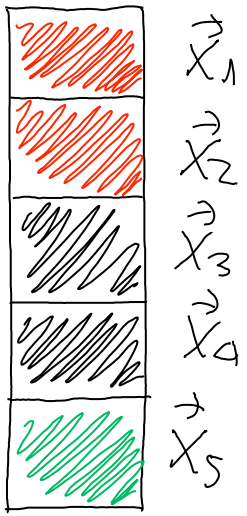


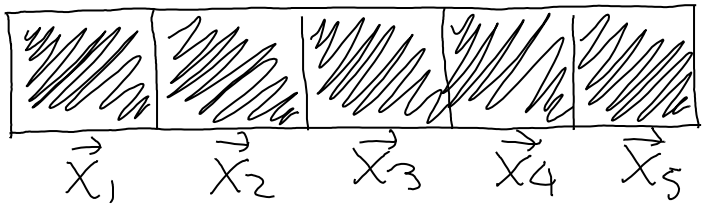
the ideal affinity matrix

$A_{ij} = 1$ if x_i, x_j belong to the same segment else 0

A

1	1	0	0	0
1	1	0	0	0
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1





the ideal segment vector

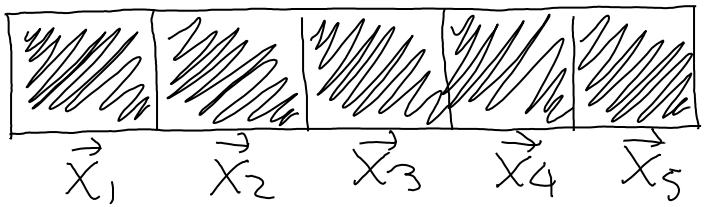
Let $y_i = 1$ if pixel $\vec{x}_i$ belongs to segment 1

A

1	1	0	0	0
1	1	0	0	0
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1

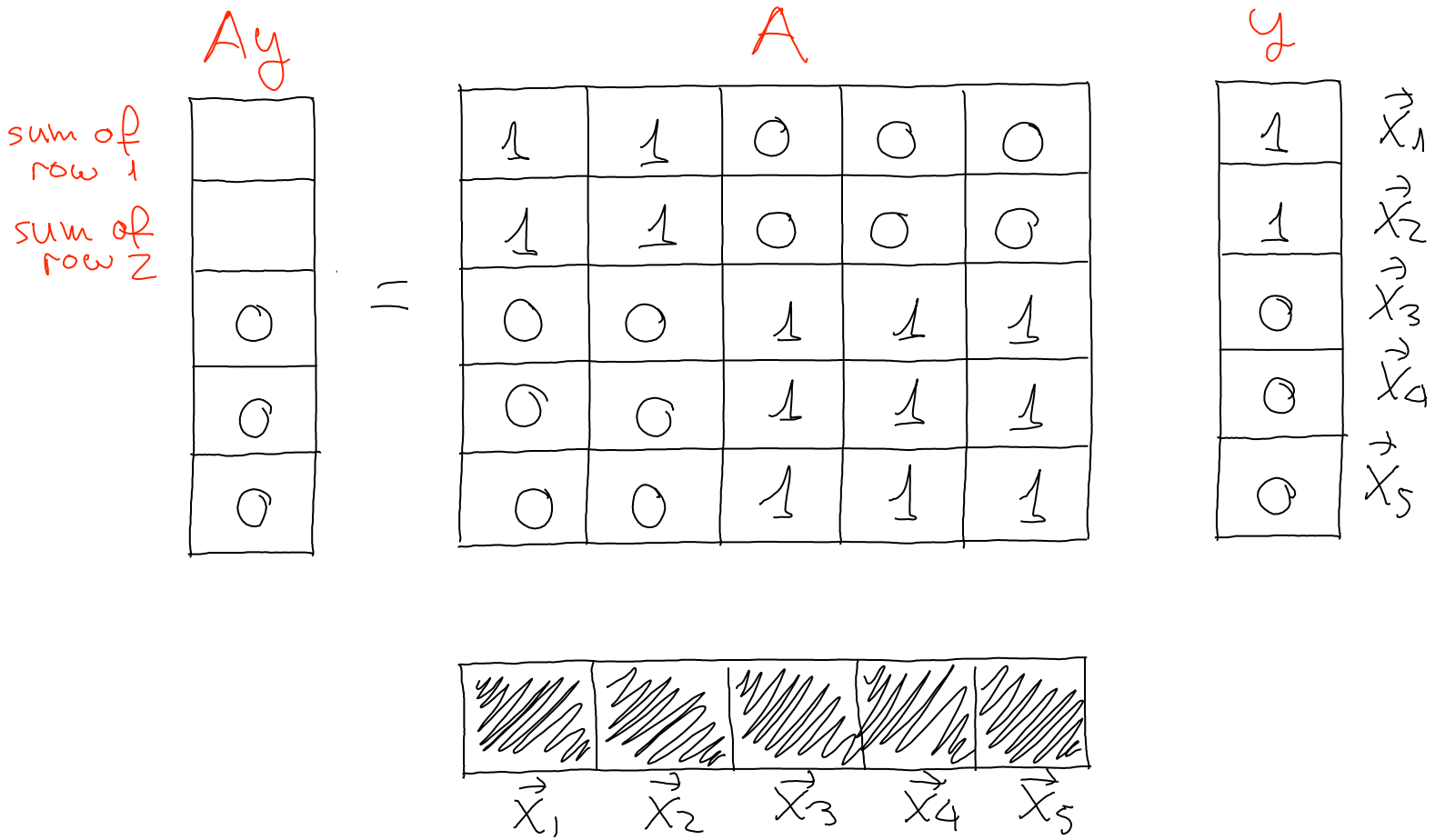
y

1	$\vec{x}_1$
1	$\vec{x}_2$
0	$\vec{x}_3$
0	$\vec{x}_4$
0	$\vec{x}_5$



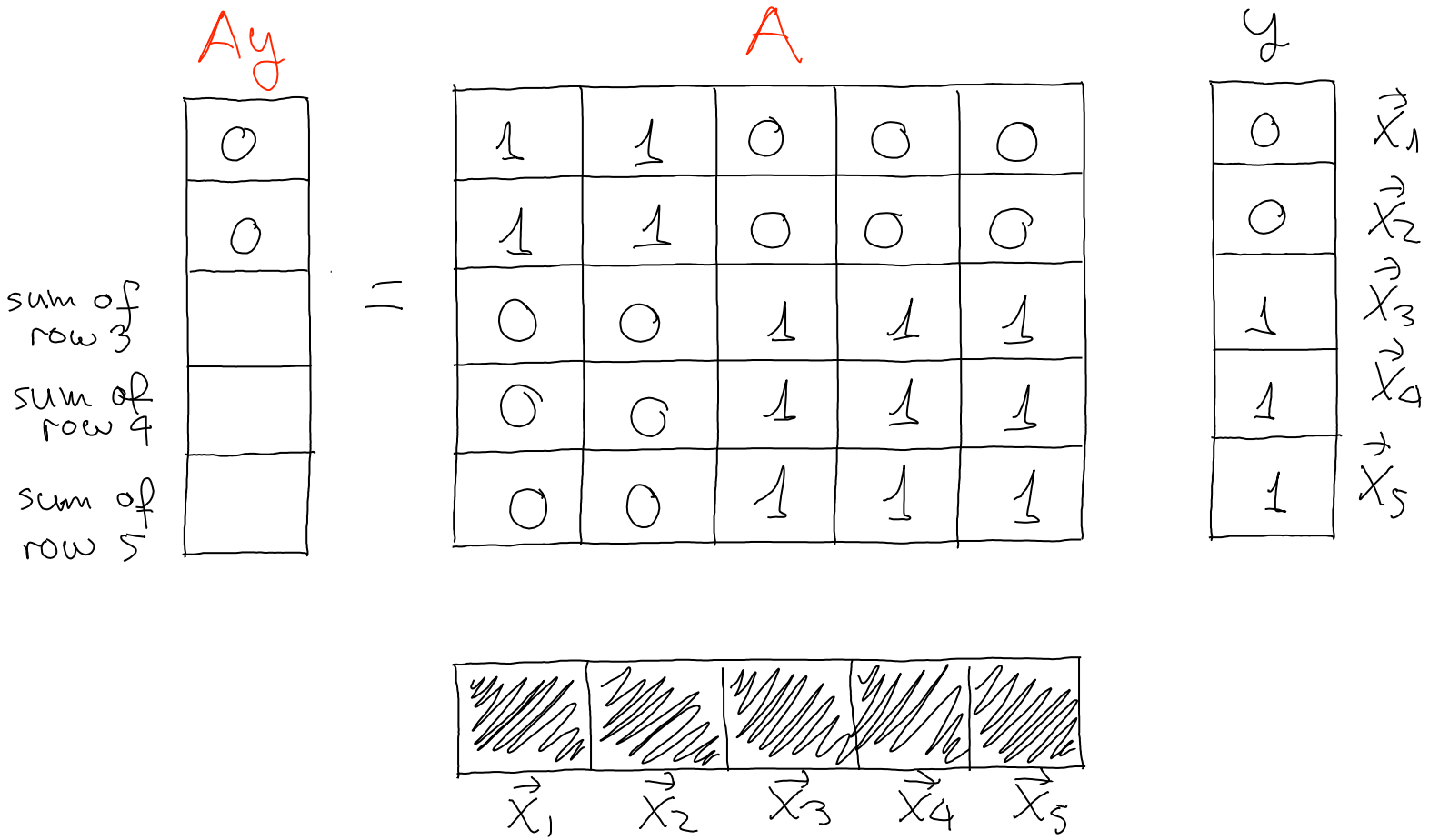
product of affinity matrix & segment vectors

Let $y_i = 1$ if pixel $\vec{x}_i$ belongs to segment 1



product of affinity matrix & segment vectors

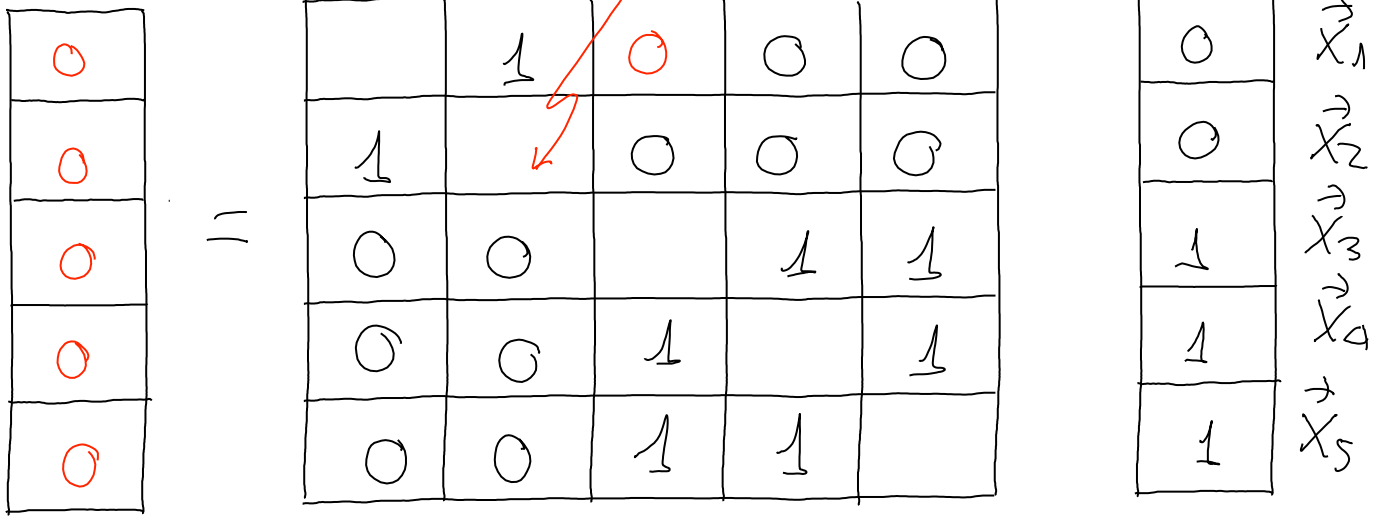
Let $y_i = 1$ if pixel $\vec{x}_i$ belongs to segment 2

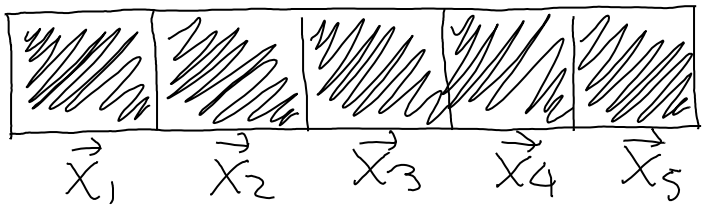


effect of subtracting row-sums from A's diagonal

Define a new matrix by subtracting A's row-sums from its diagonal elements

$A_{ii} - \sum_j A_{ij}$

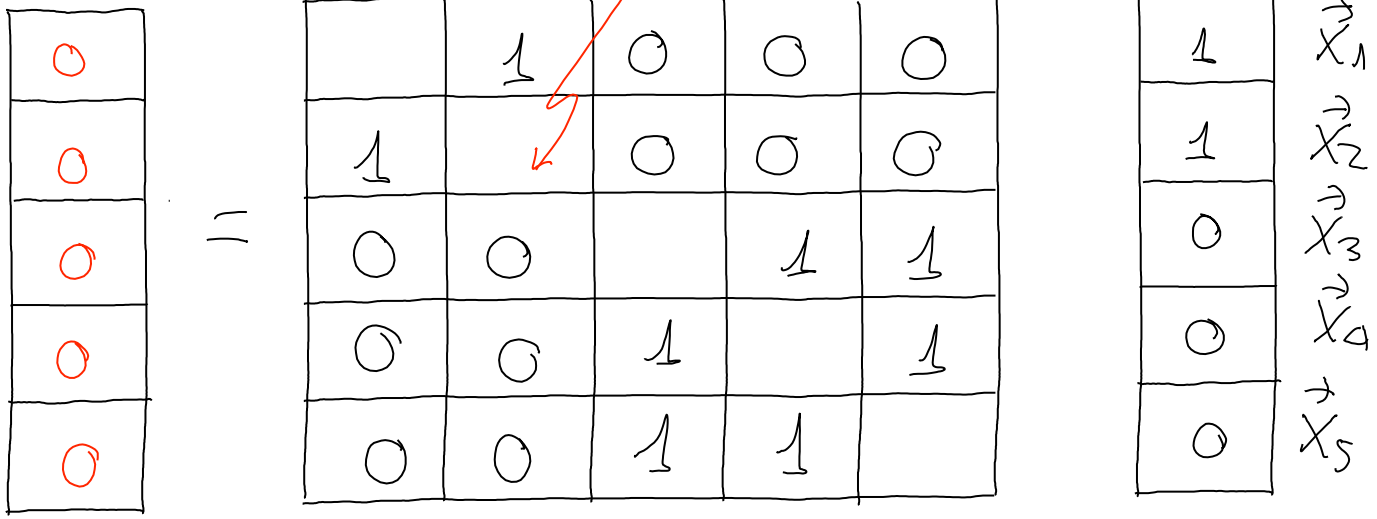


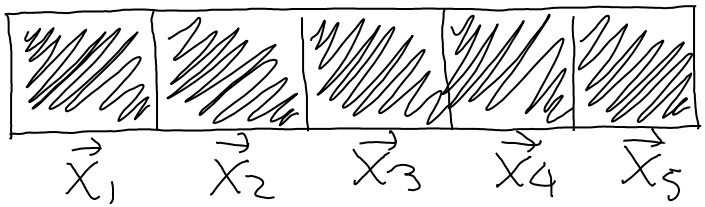


effect of subtracting row-sums from A's diagonal

Define a new matrix by subtracting A's row-sums from its diagonal elements

$A_{ii} - \sum_j A_{ij}$



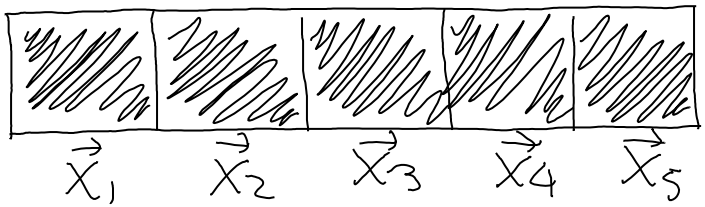


the Laplacian matrix L

Define a new matrix L by subtracting A's row-sums from its diagonal elements & negating

$\sum_j A_{ij} - A_{ii}$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} & -1 & 0 & 0 & 0 \\ -1 & & 0 & 0 & 0 \\ 0 & 0 & & -1 & -1 \\ 0 & 0 & -1 & & -1 \\ 0 & 0 & -1 & -1 & \end{bmatrix} \begin{matrix} y \\ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix} \begin{matrix} \\ \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vec{x}_4 \\ \vec{x}_5 \end{matrix}$$



relation between segments & L's eigenvectors

Segment vectors are eigenvectors of L with zero eigenvalue

$L = \mathrm{diag}(A \cdot 1) - A$

$0 \cdot Ly = 0 \cdot y$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} & -1 & 0 & 0 & 0 \\ -1 & & 0 & 0 & 0 \\ 0 & 0 & & -1 & -1 \\ 0 & 0 & -1 & & -1 \\ 0 & 0 & -1 & -1 & \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vec{x}_4 \\ \vec{x}_5 \end{matrix}$$

Vector $y = 1$ is always also an eigenvector of L (corresponds to 'trivial' segment that contains all pixels)

relation between segments & L's eigenvectors

Segment vectors are eigenvectors of L with <u>zero</u> eigenvalue

$0 \cdot Ly = L = \mathrm{diag}(A \cdot 1) - A \quad y$

$0 \cdot Ly$		$L = \mathrm{diag}(A \cdot 1) - A$				y	
0	=		-1	0	0	0	1 $\vec{x}_1$
0		-1		0	0	0	1 $\vec{x}_2$
0		0	0		-1	-1	0 $\vec{x}_3$
0		0	0	-1		-1	0 $\vec{x}_4$
0		0	0	-1	-1		0 $\vec{x}_5$

A, L are both symmetric positive definite $\Rightarrow$

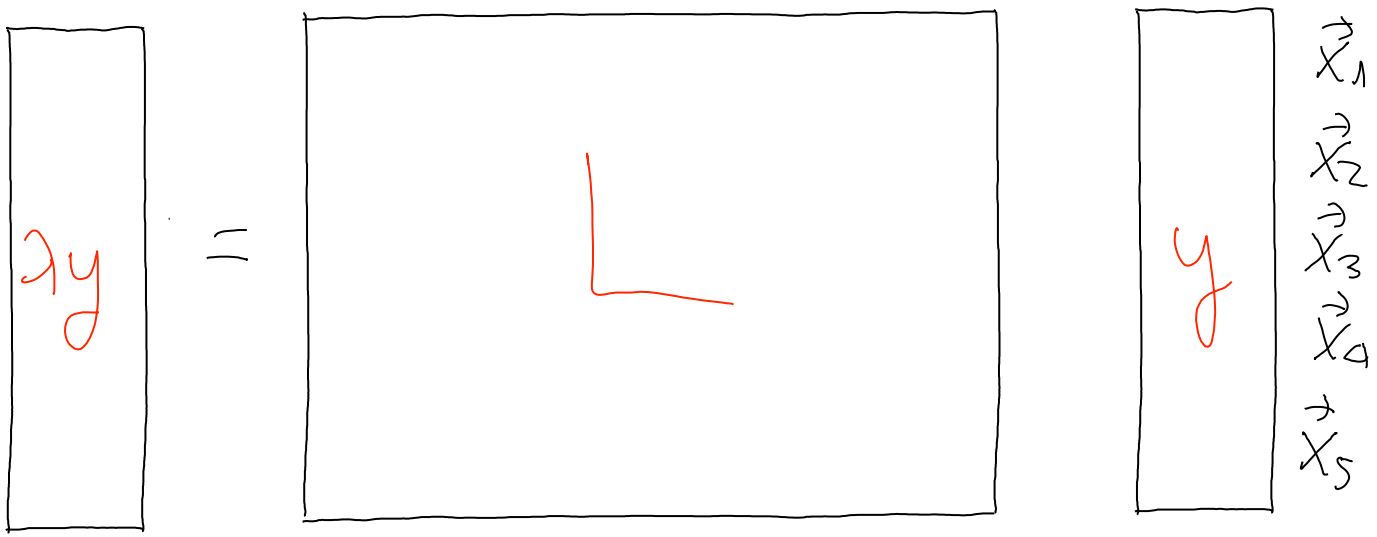
they both have orthogonal eigenvectors

$\Rightarrow$ can avoid trivial solution by seeking the 2nd-to-last eigenvector of L

general affinity & Laplacian matrices

Eigenvectors computable by solving an eigenvalue problem

$$Ly = \lambda y$$



Complications

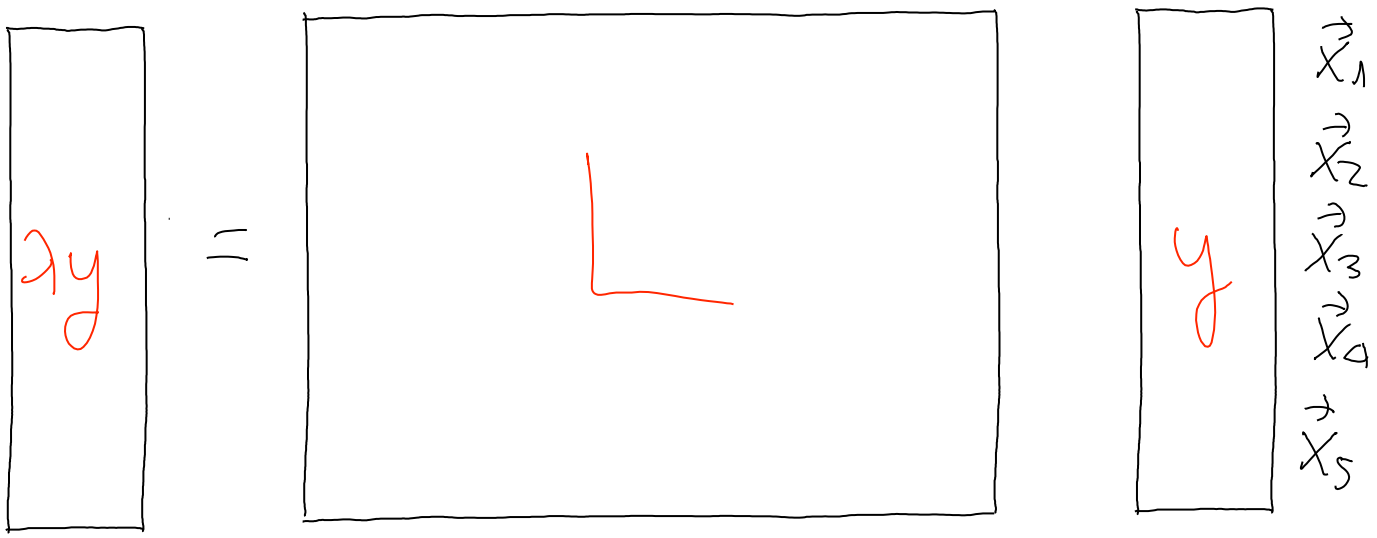
- y not constrained to be binary → must do thresholding
- if y_1, y_2 are eigenvectors with zero eigenvalue then so are Ry_1, Ry_2 for any rotation matrix R

must be computed!

general affinity & Laplacian matrices

Eigenvectors computable by solving an eigenvalue problem

$$Ly = \lambda y$$



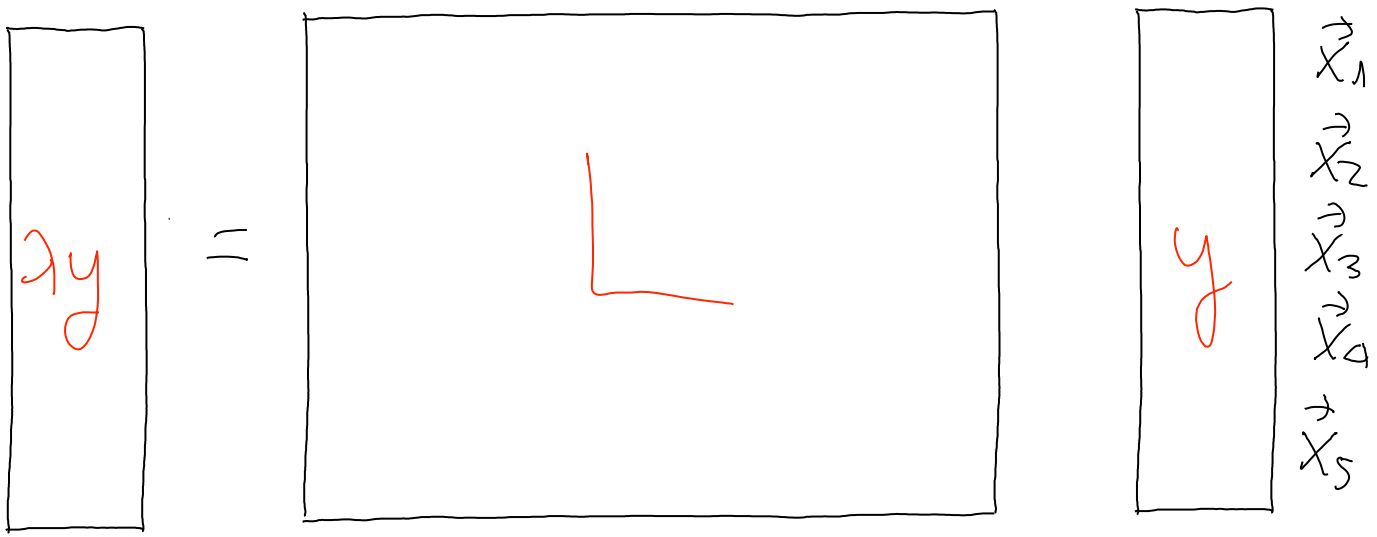
Complications:

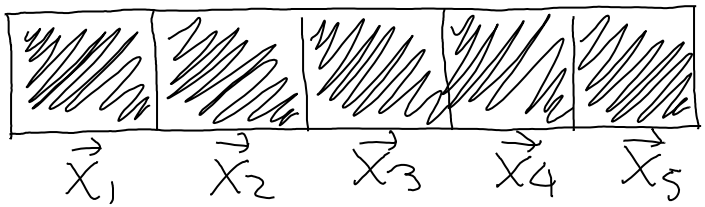
(most important) eigenvectors of L, as defined, do not give great segments ...

general affinity & Laplacian matrices

Eigenvectors computable by solving an eigenvalue problem

$$Ly = \lambda y$$

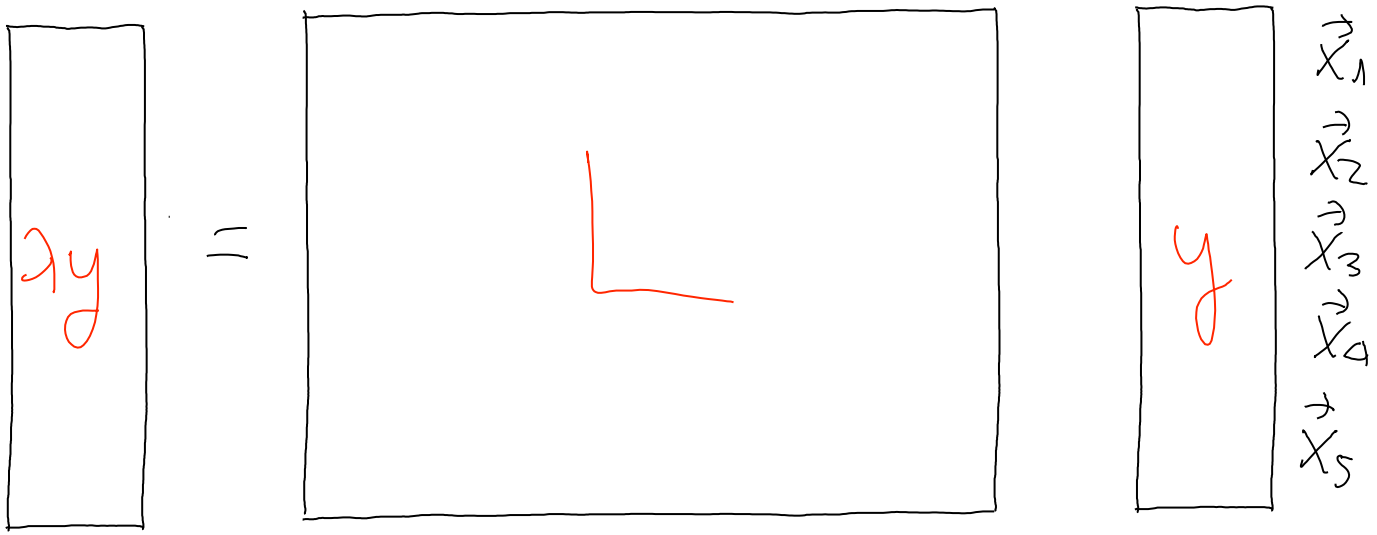




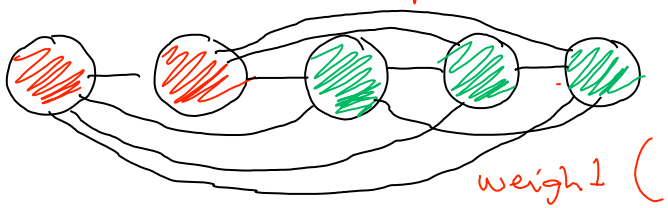
binary segmentation as a graph cut problem

The null vectors of L are best understood by considering an equivalent graph cut problem

$$L = \text{diag}(A \cdot \underline{1}) - A$$



weighted graph representation



$\text{weight}(x_i, x_j) = A_{ij}$

$\vec{x}_1$ $\vec{x}_2$ $\vec{x}_3$ $\vec{x}_4$ $\vec{x}_5$

binary segmentation as a graph cut problem

The null vectors of L are best understood by considering an equivalent graph cut problem

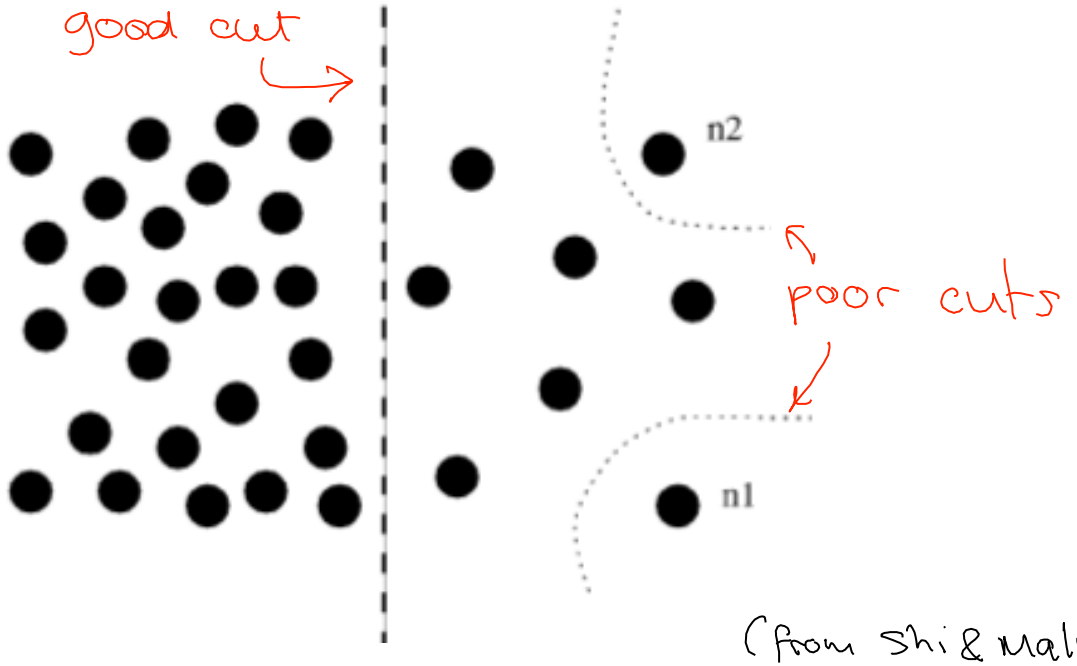
Goal: Partition (i.e. cut) the graph into two parts, each corresponding to a segment, by removing a subset of its edges

weighted graph representation

$weight(x_i, x_j) = A_{ij}$

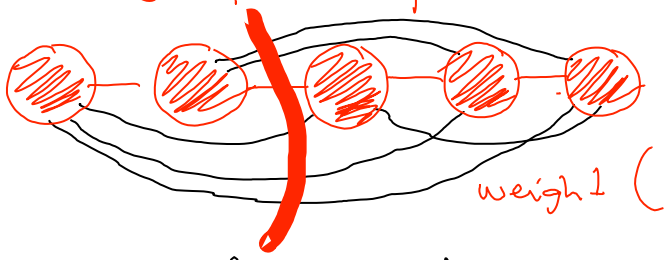
cut(F, G) = sum of weights of edges between F, G

2D example (closer = more similar)



(from Shi & Malik, PAMI 2000)

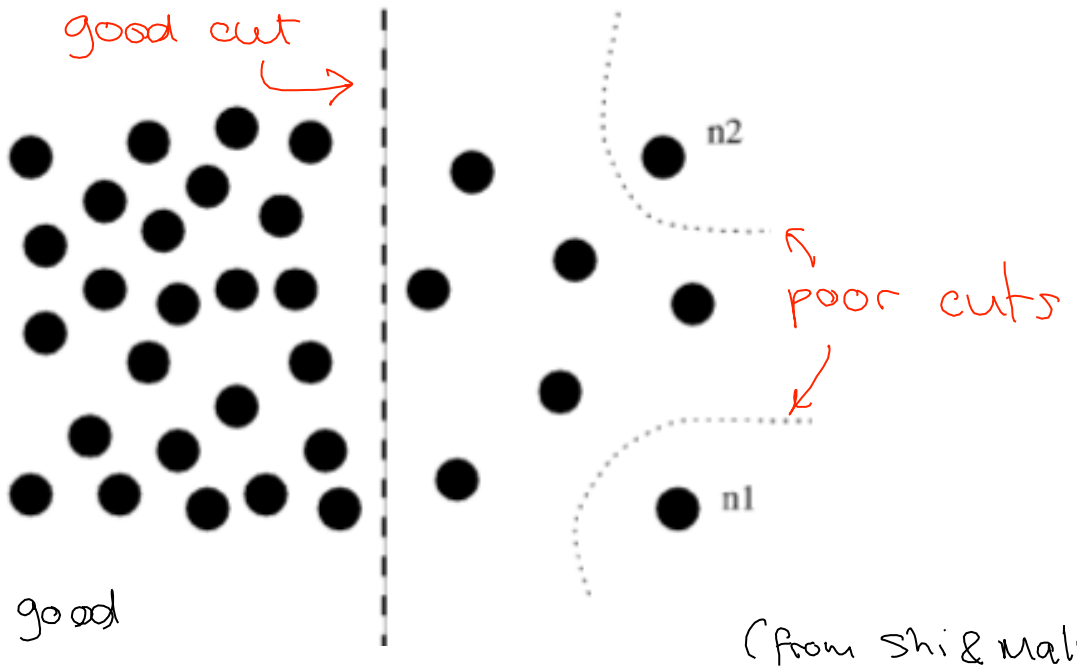
weighted graph representation



$weight(x_i, x_j) = A_{ij}$

$cut(F, G)$ = sum of weights of edges between F, G

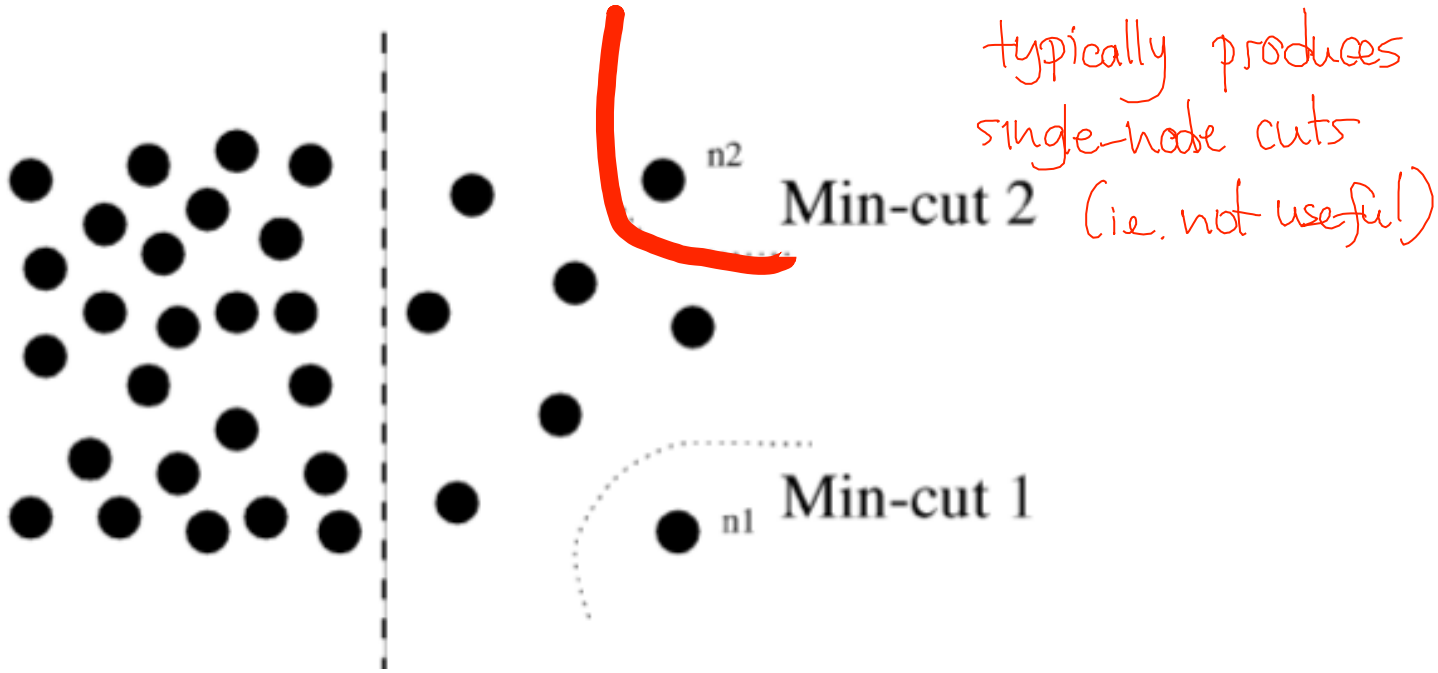
general problem statement



(from Shi & Malik, PAMI 2000)

Given a weighted graph (V, E) partition it into parts F and $G = V - F$ into order to minimize a function $L(F, G)$ of the edge weights

the minimum cut

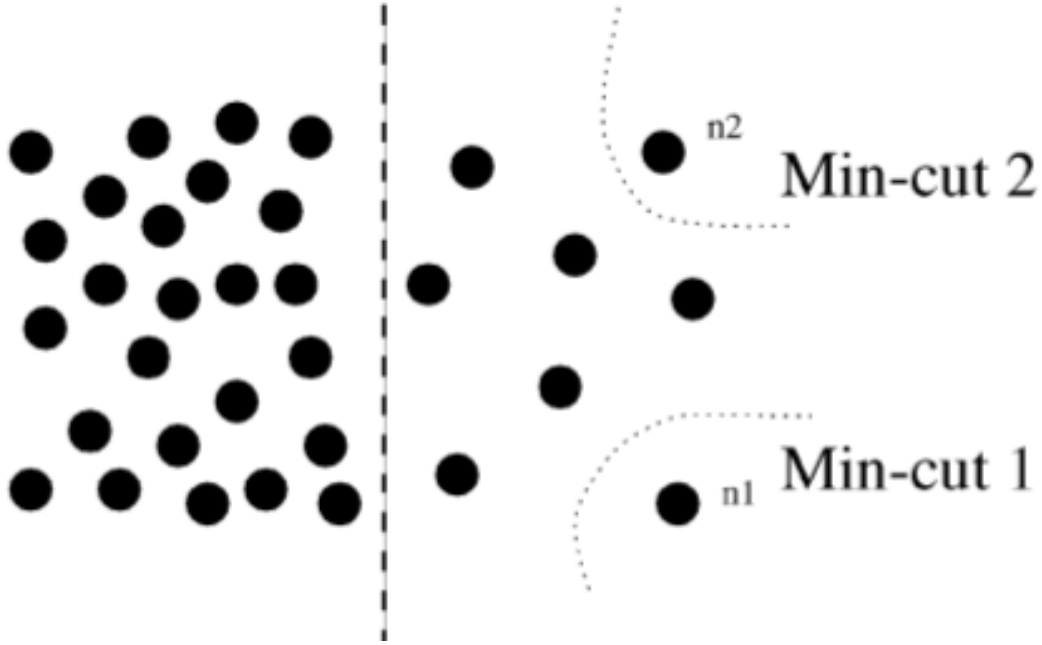


typically produces single-node cuts (ie. not useful)

compute $\arg\min_{F,G} L(F,G)$ with

$$L(F,G) = cut(F,G)$$

the minimum average cut



compute $\arg\min_{F,G} L(F,G)$ with

$$L(F,G) = \frac{cut(F,G)}{|F|} + \frac{cut(F,G)}{|G|}$$

$|F|$ ← #nodes in F

denominator biases solution toward larger sets

the minimum average cut

Observation (Shi & Malik, PAMI 2000)

> The second-lowest eigenvector of L is a real-valued solution to the minimum average cut problem

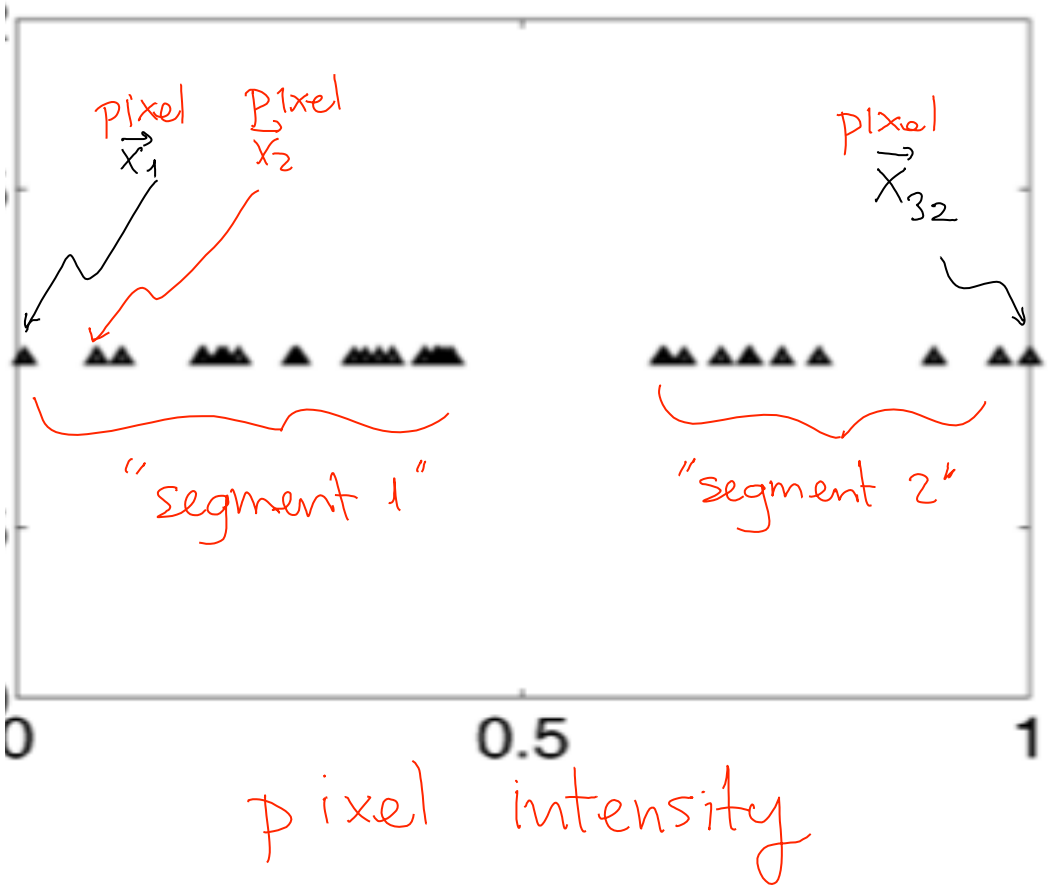
compute $\arg\min_{F,G} L(F,G)$ with

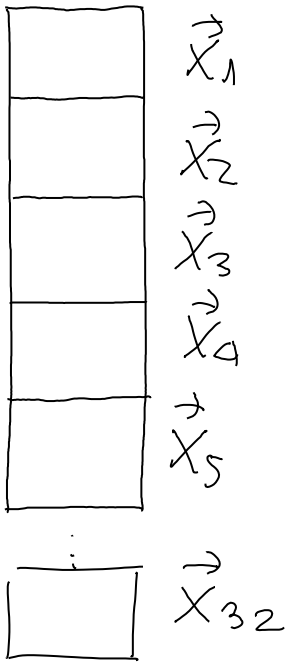
$$L(F,G) = \frac{cut(F,G)}{|F|} + \frac{cut(F,G)}{|G|}$$

$|F|$: # nodes in F

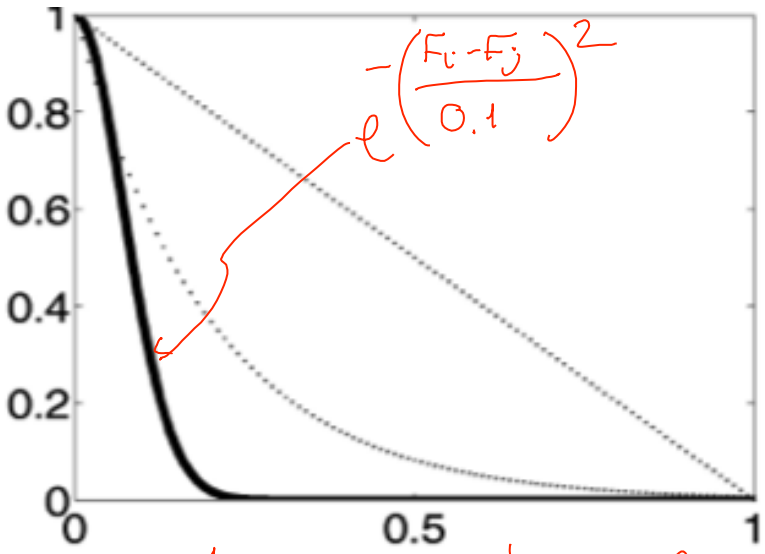
denominator biases solution toward larger sets

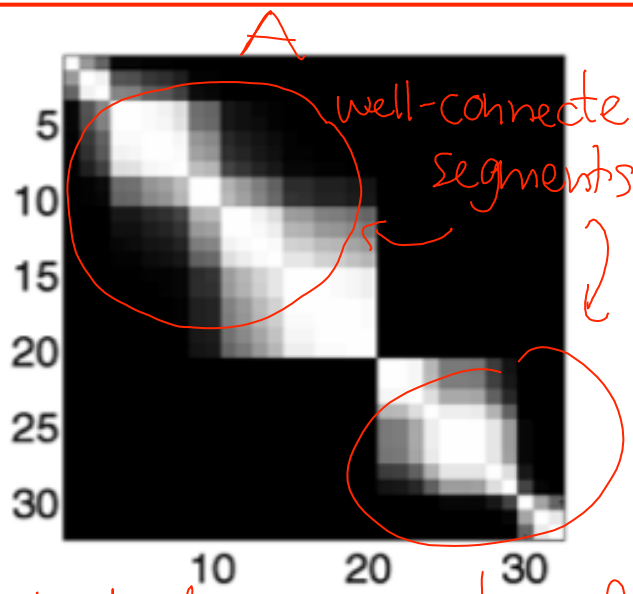
example: 1D image



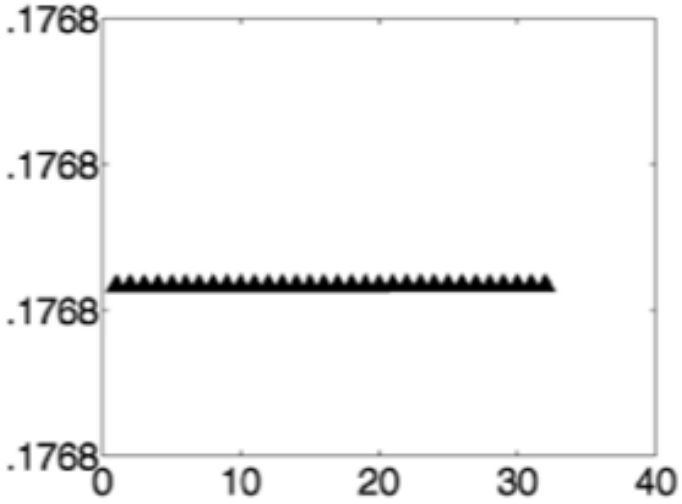


the minimum average cut

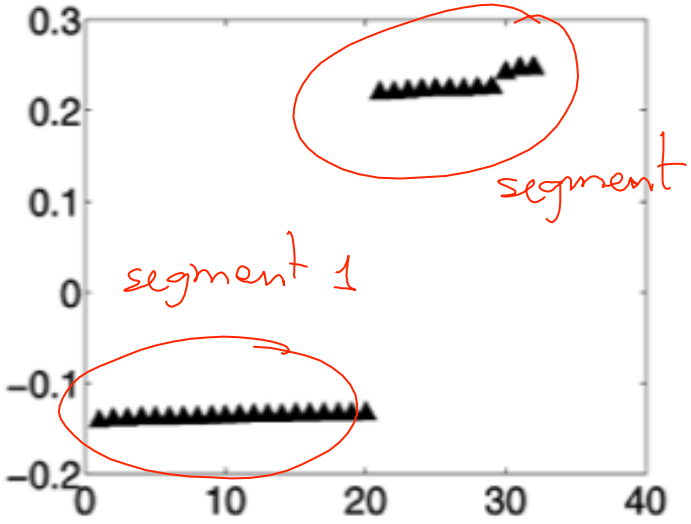




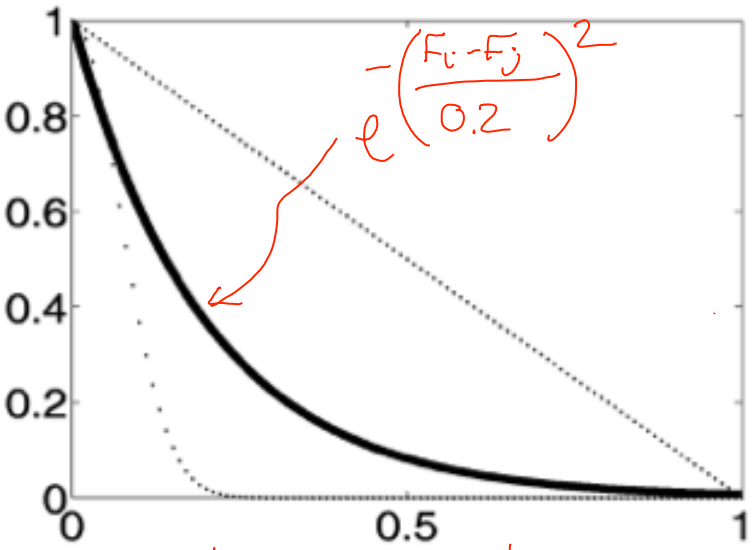
last eigenvector of L

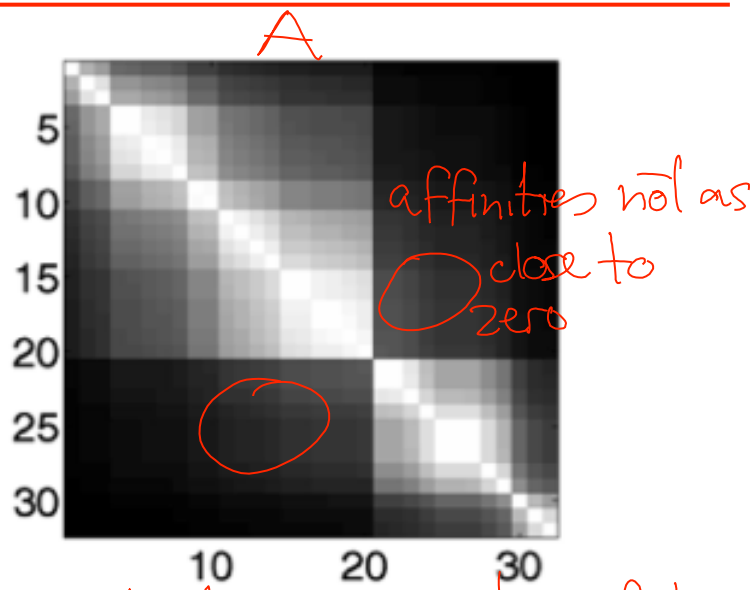


2nd-to-last eigenvector of L

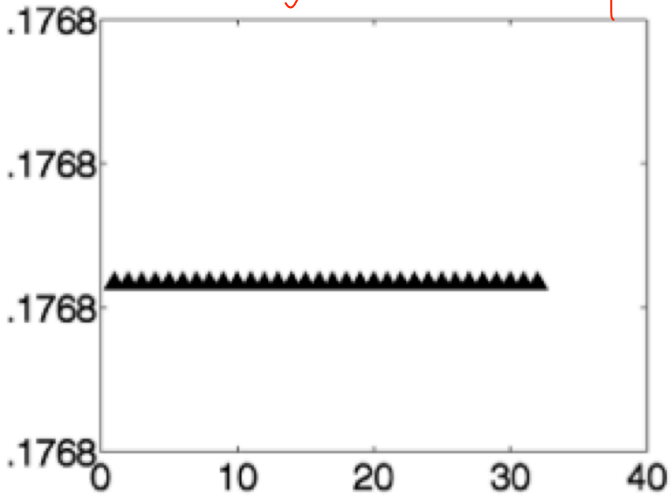


the minimum average cut

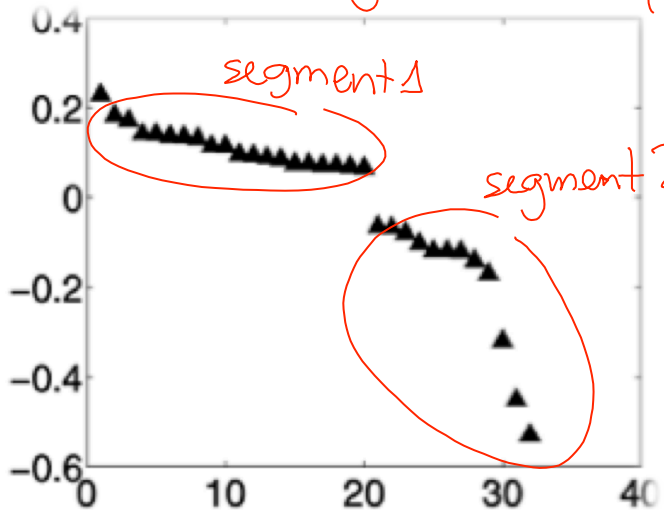




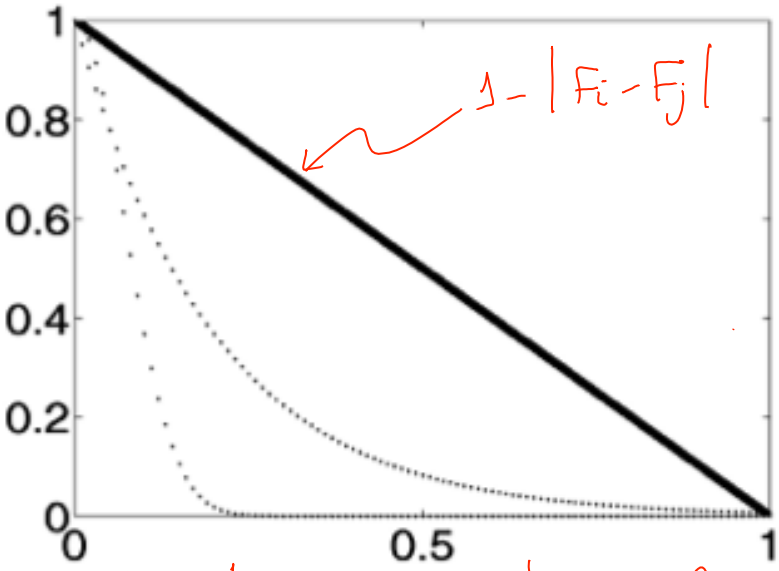
last eigenvector of L

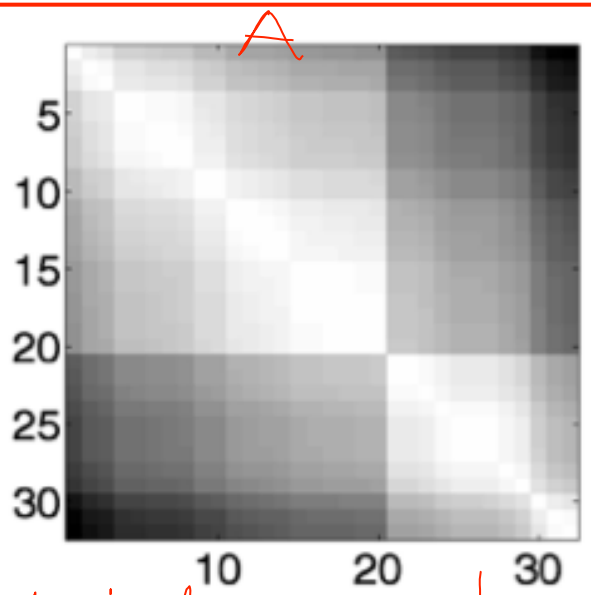


2nd-to-last eigenvector of L

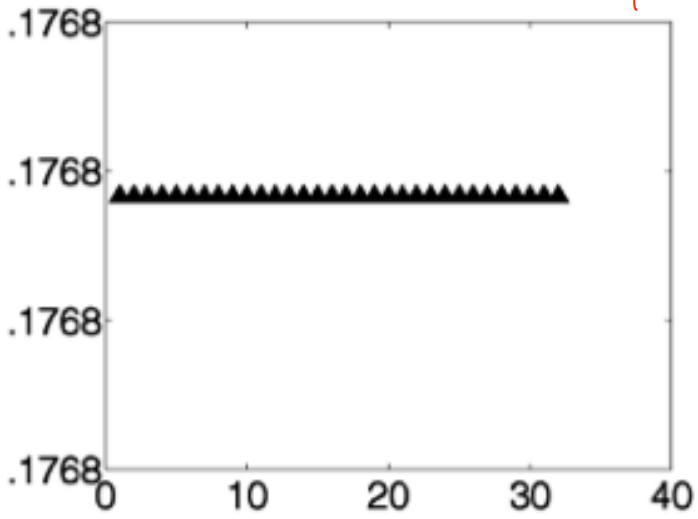


the minimum average cut

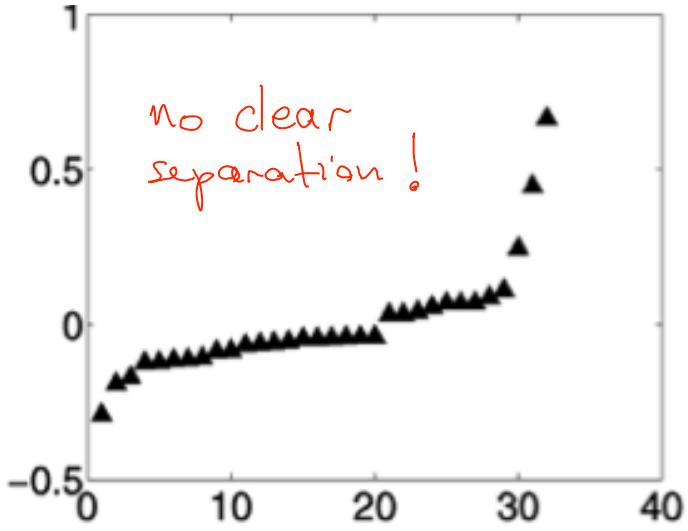




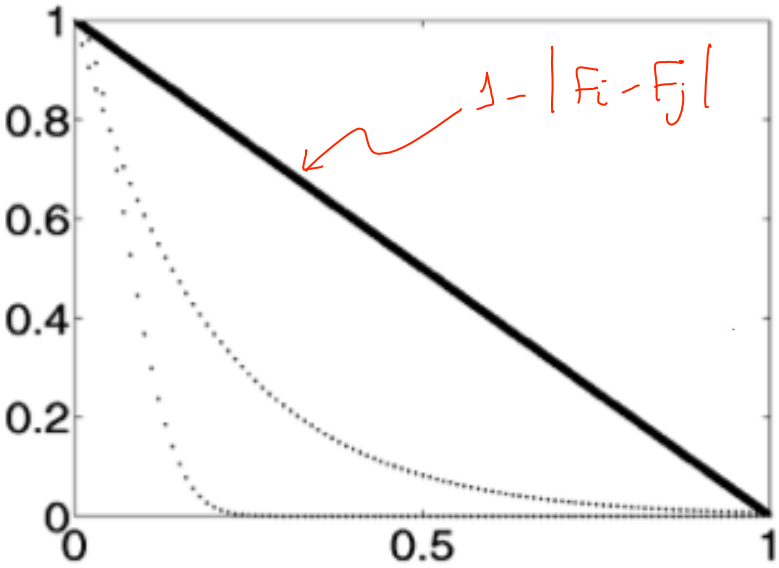
last eigenvector of L

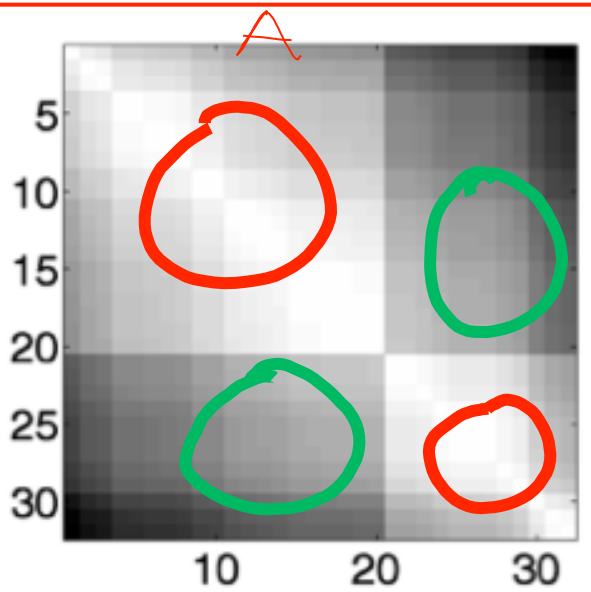


2nd-to-last eigenvector of L



the minimum average cut

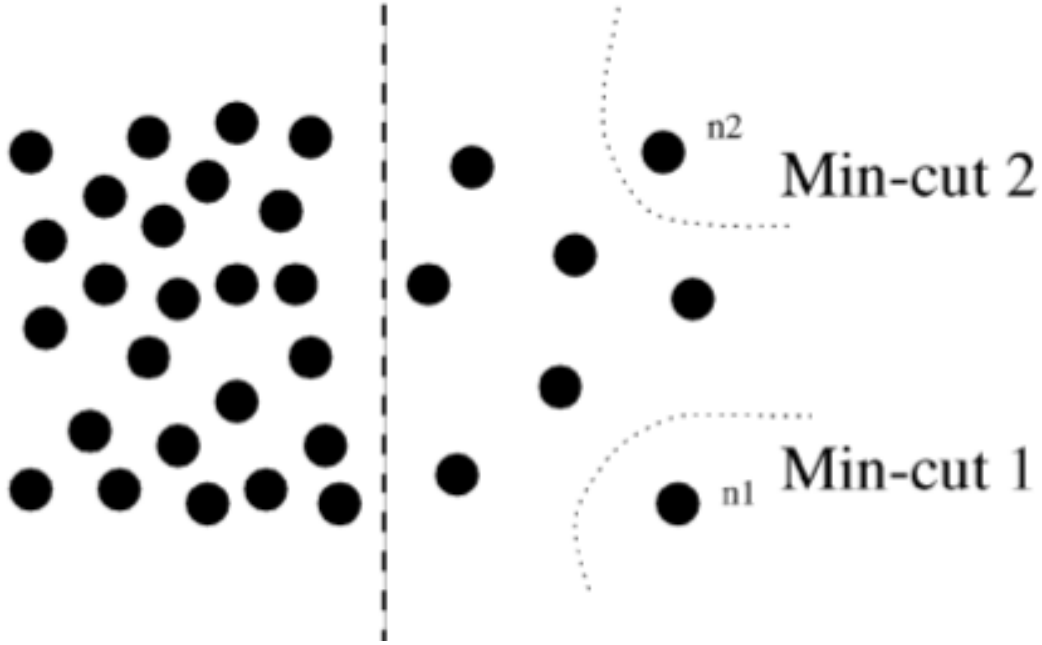




average cut measures the dis-similarity across segments but does not take into account pixel similarities within segments

$$L(F,G) = \frac{cut(F,G)}{|F|} + \frac{cut(F,G)}{|G|}$$

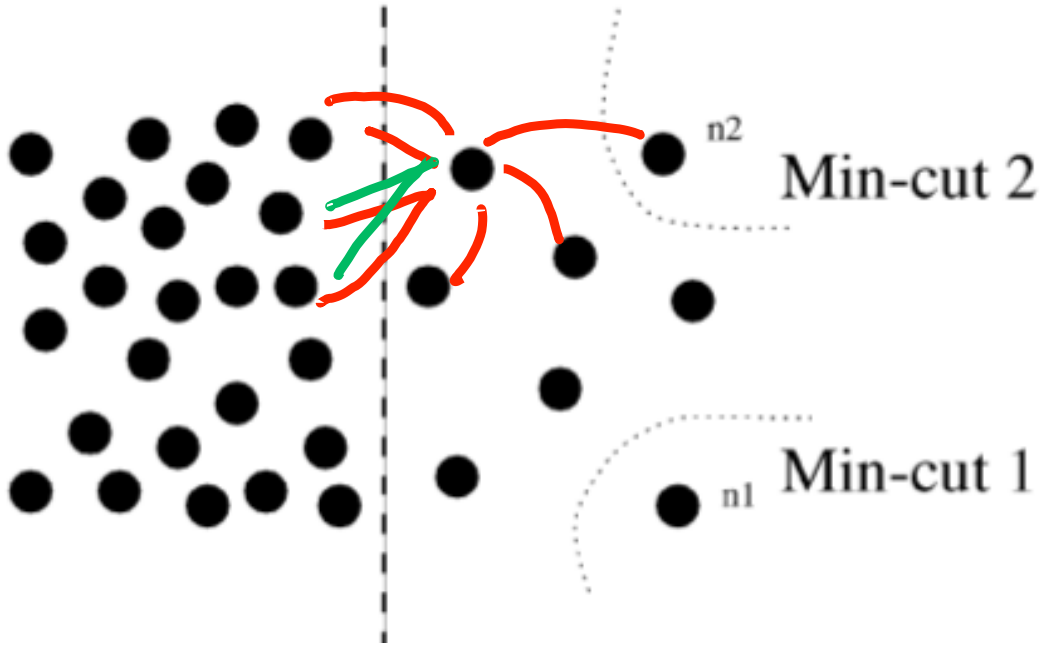
a better measure: the minimum normalized cut



compute $\arg\min_{F,G} L(F,G)$ with

$$L(F,G) = \frac{cut(F,G)}{\quad} + \frac{cut(F,G)}{\quad}$$

a better measure: the minimum normalized cut



compute $\underset{F,G}{\arg\min}\, L(F,G)$ with

$$L(F,G) = \frac{cut(F,G)}{\sum_{\substack{x_i \in F \\ x_j \in V}} A_{ij}} + \frac{cut(F,G)}{\sum_{\substack{x_i \in G \\ x_j \in V}} A_{ij}}$$

total connection from F to G

total connection from G to all nodes in graph

spectral approximation to the normalized cut

Observation (Shi & Malik, PAMI 2000)

A real valued solution to the minimization problem below is given by the 2nd-to-last generalized eigenvector of the system

$$L y = \lambda \, \mathrm{diag}(A \cdot 1) y$$

compute $\arg\min_{F,G} L(F,G)$ with

$$L(F,G) = \frac{\mathrm{cut}(F,G)}{\sum_{\substack{x_i \in F \\ x_j \in V}} A_{ij}} + \frac{\mathrm{cut}(F,G)}{\sum_{\substack{x_i \in G \\ x_j \in V}} A_{ij}}$$

spectral approximation to the normalized cut

Observation (Shi & Malik, PAMI 2000)

A real valued solution to the minimization problem below is given by the 2nd-to-last generalized eigenvector of the system

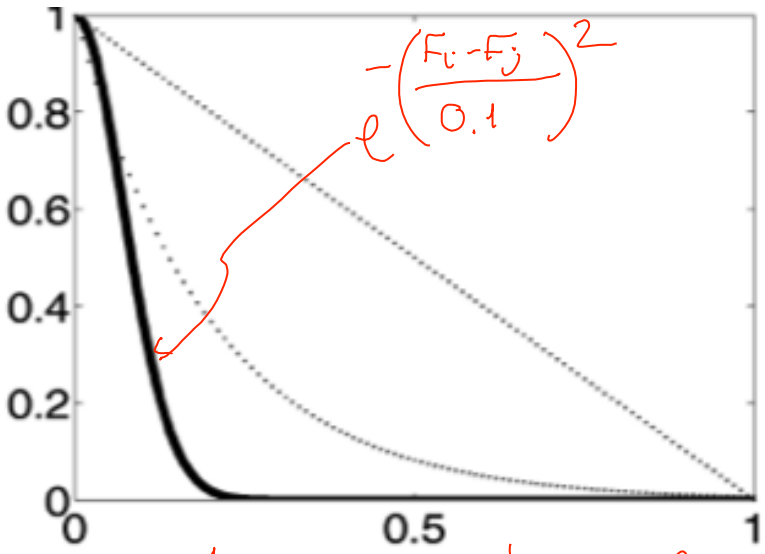
$$(*) \qquad Ly = \lambda \overbrace{\text{diag}(A \cdot 1)}^{D} y$$

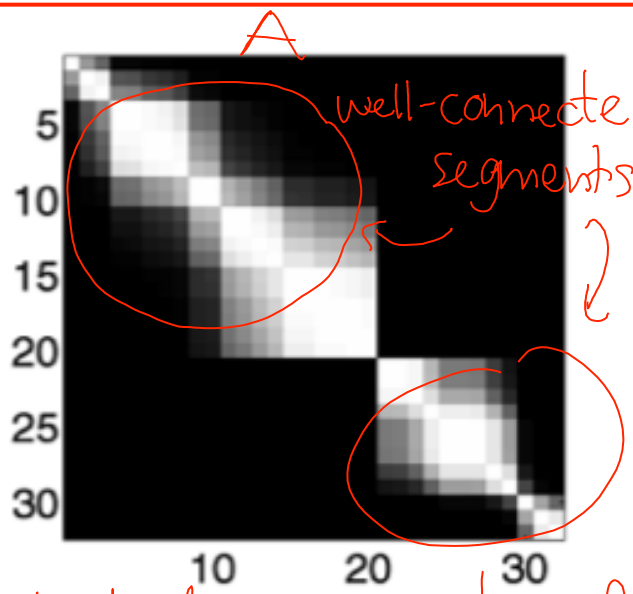
(*) equivalent to the problem

$$y = \arg\min_{y^T D 1 = 0} \frac{y^T L y}{y^T D y}$$

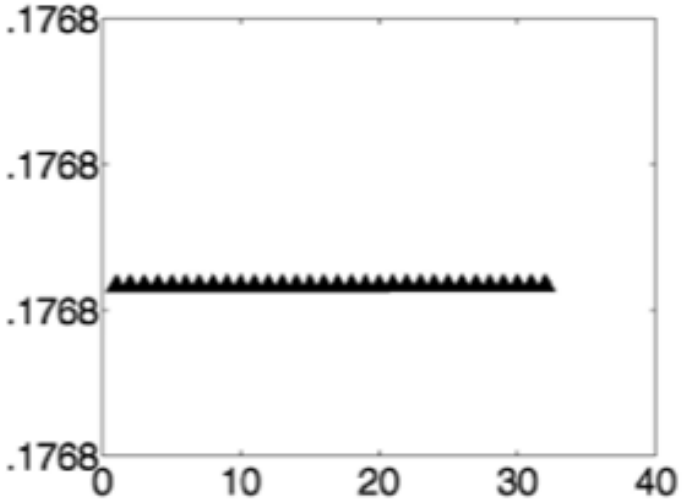
also known as the Raleigh quotient (see notes)

back to our 1D example: the minimum average cut

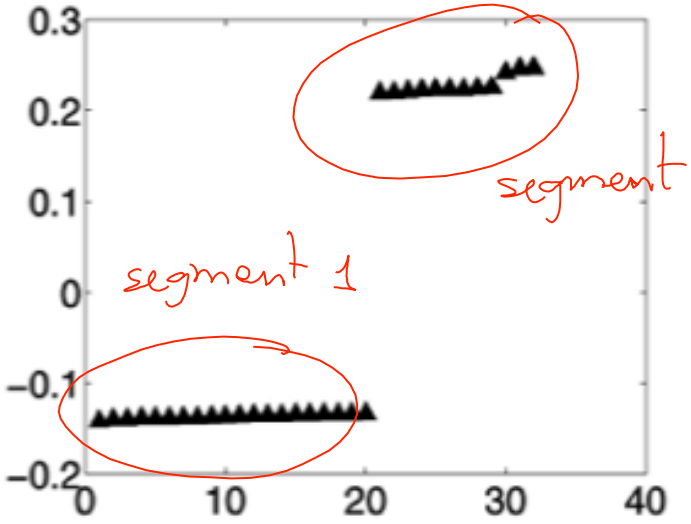




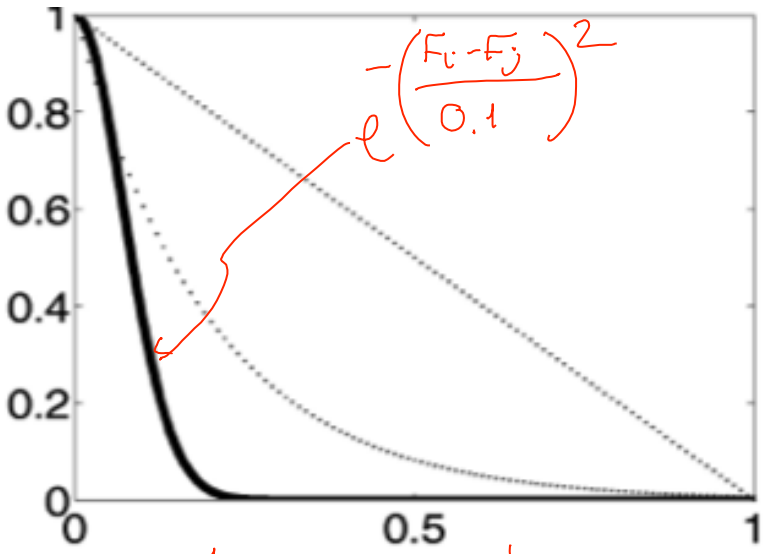
last eigenvector of L

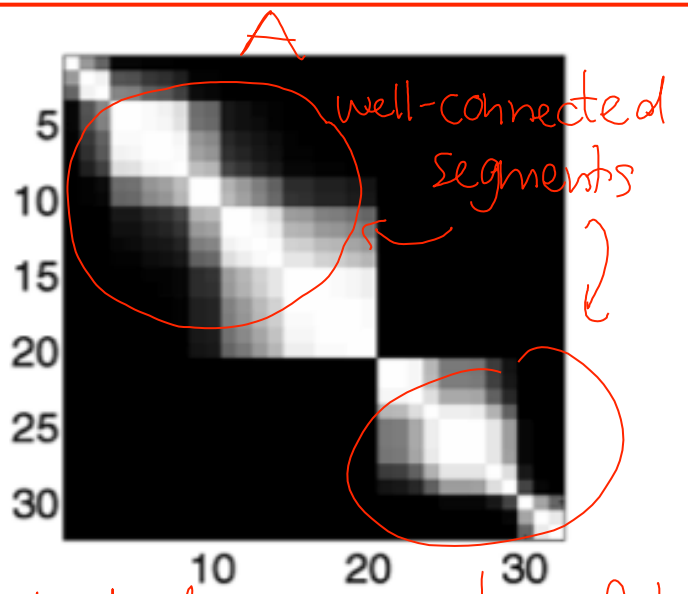


2nd-to-last eigenvector of L



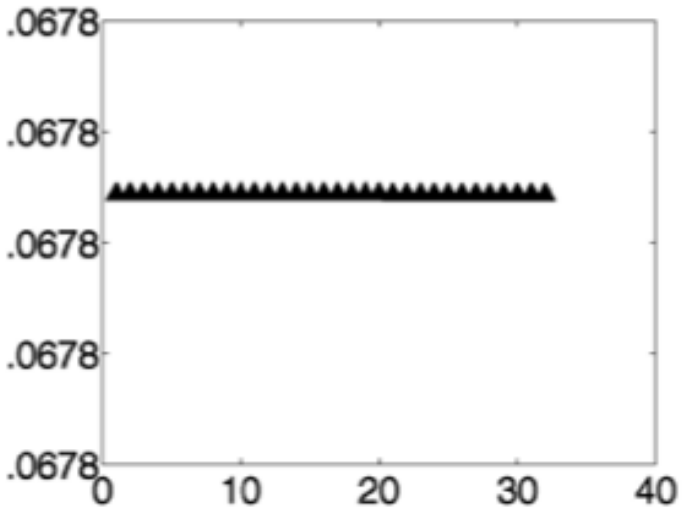
the minimum normalized cut (Ncut)

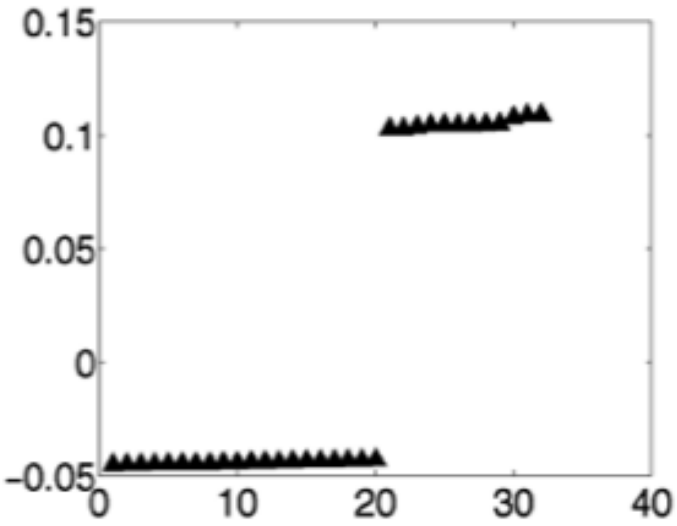




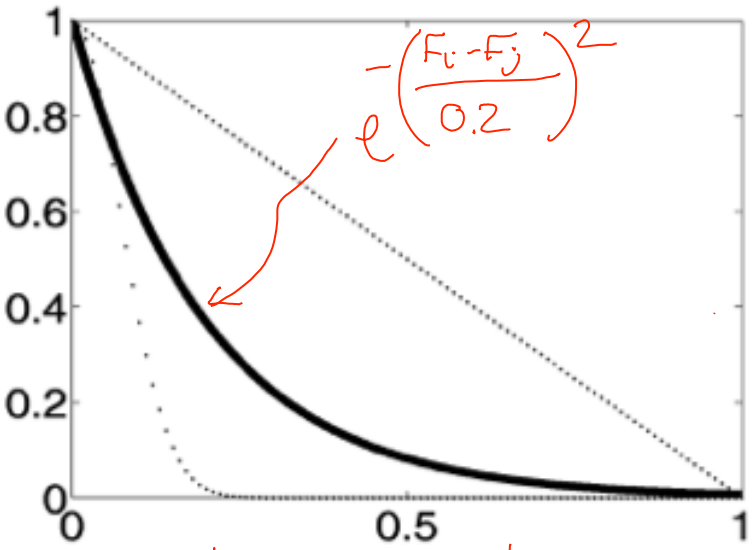
last eigenvector

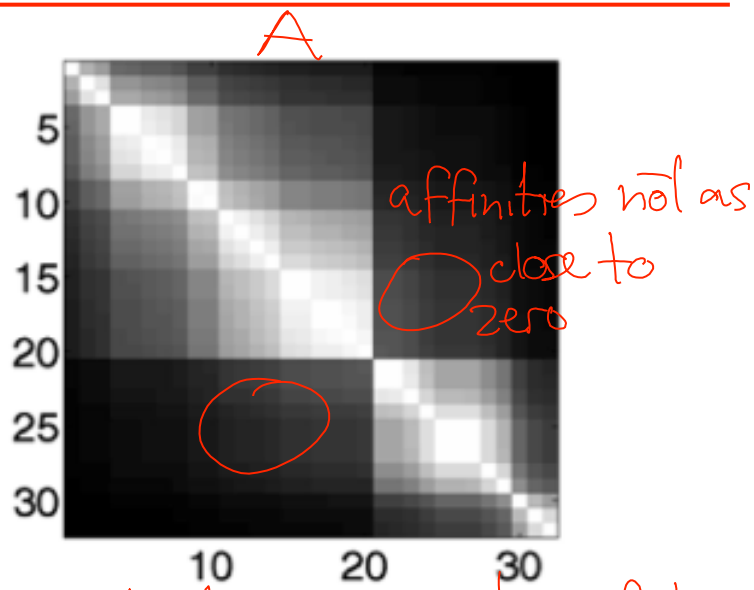
2nd-to-last eigenvector of L



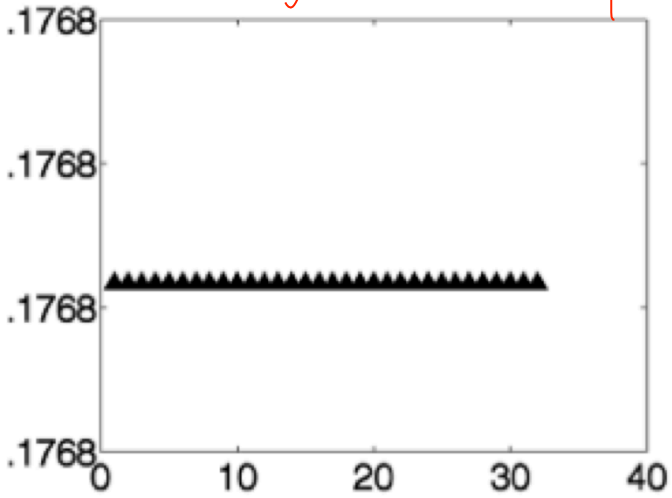


the minimum average cut

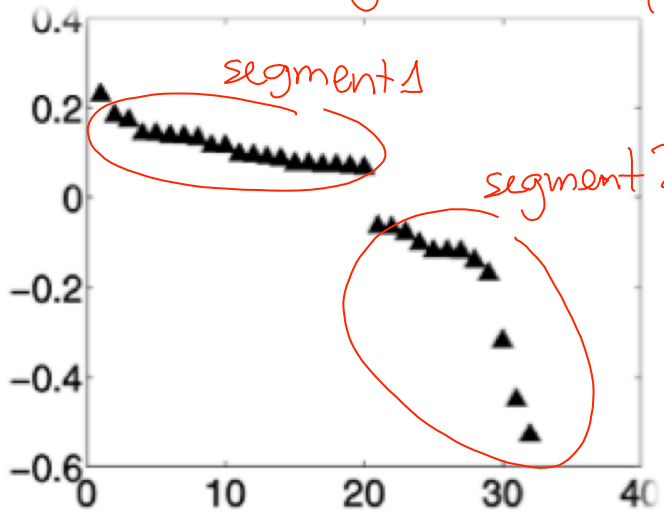




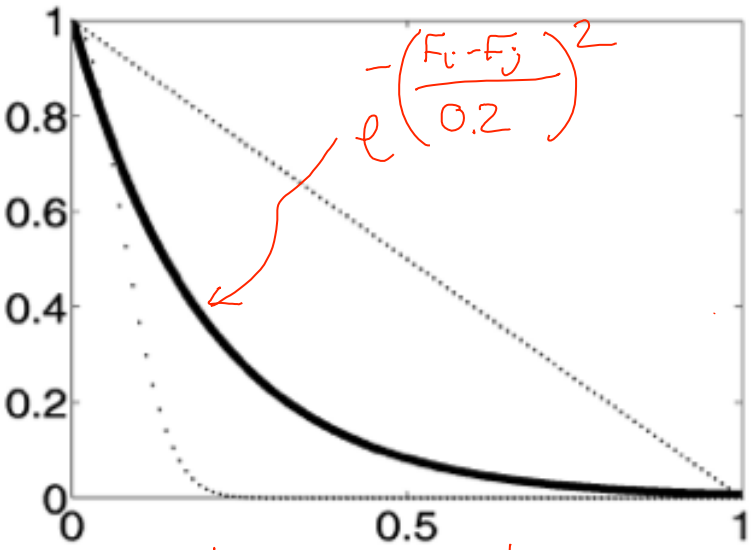
last eigenvector of L



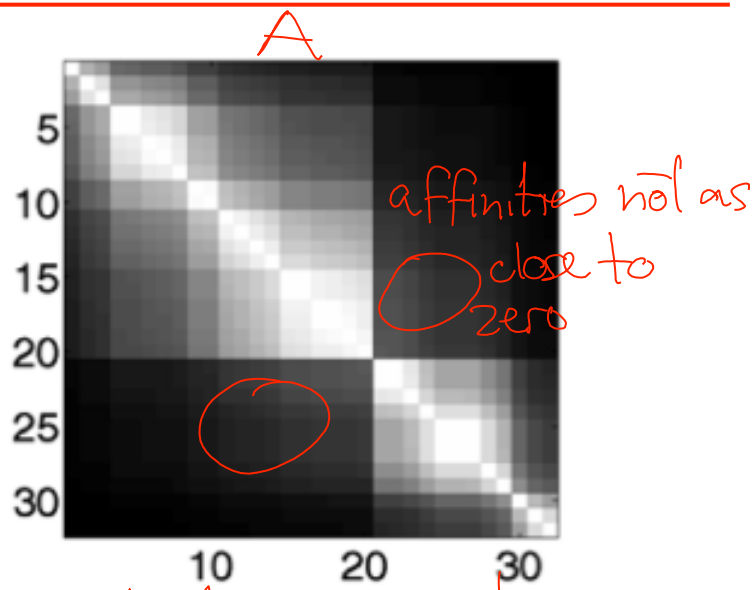
2nd-to-last eigenvector of L



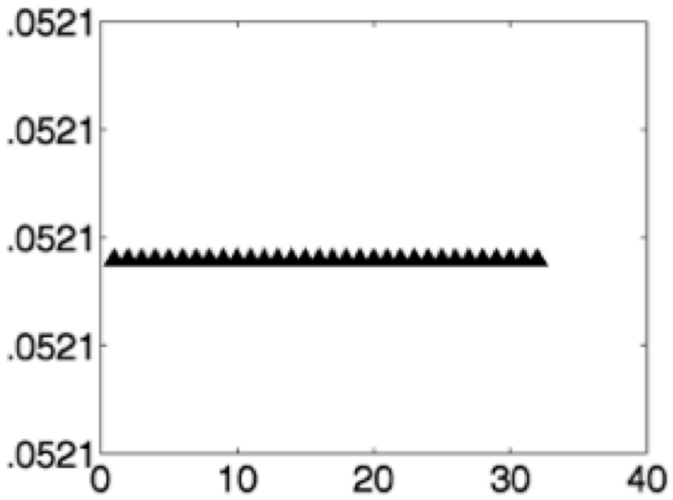
the minimum normalized cut (Ncut)

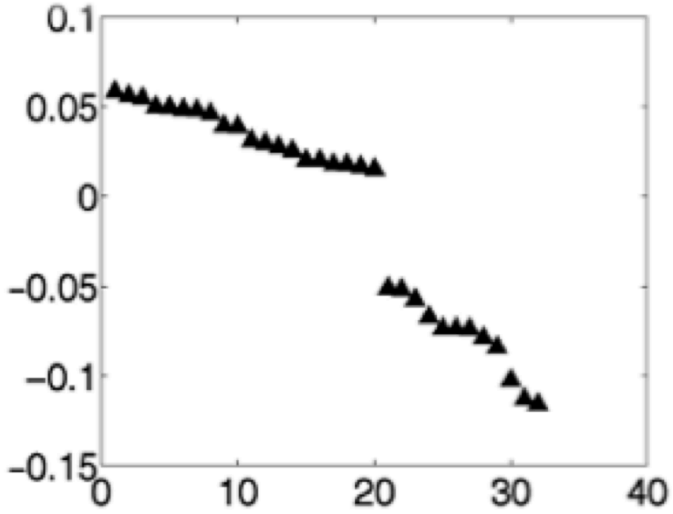


last eigenvector

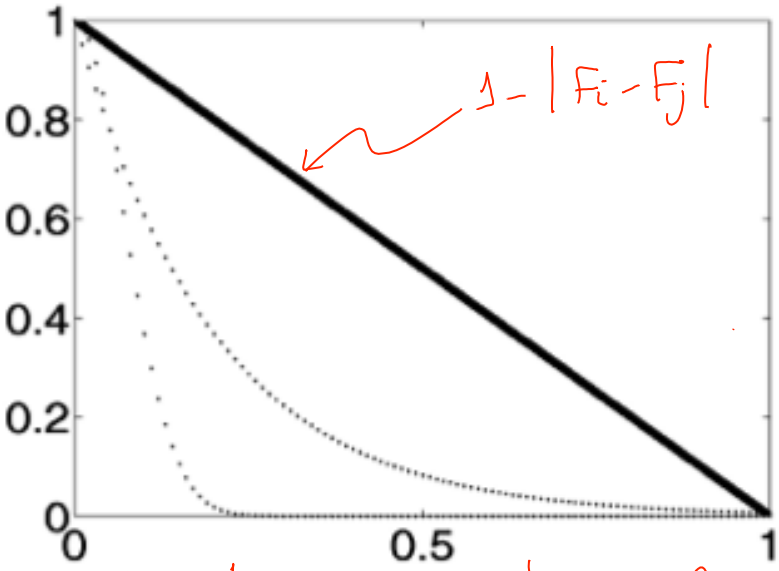


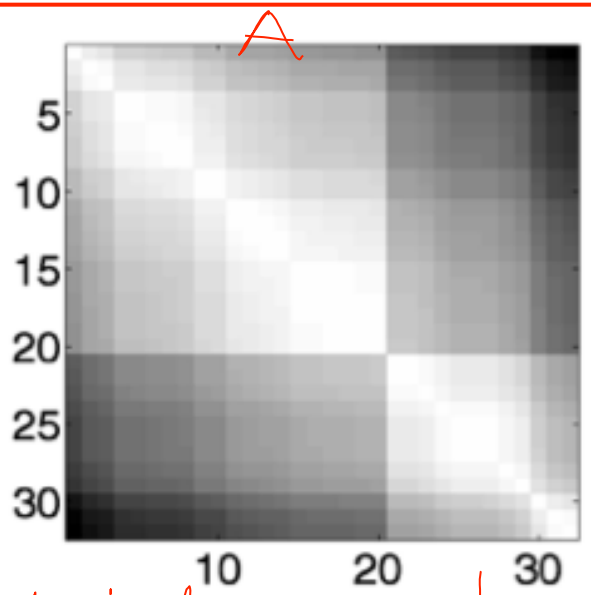
2nd-to-last eigenvector



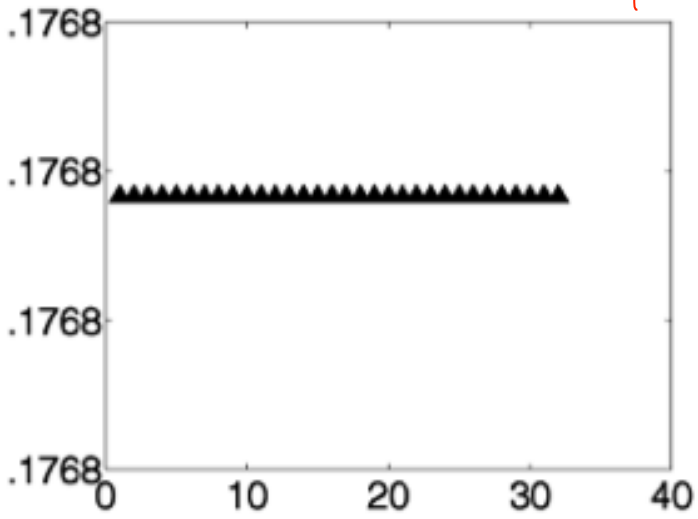


the minimum average cut

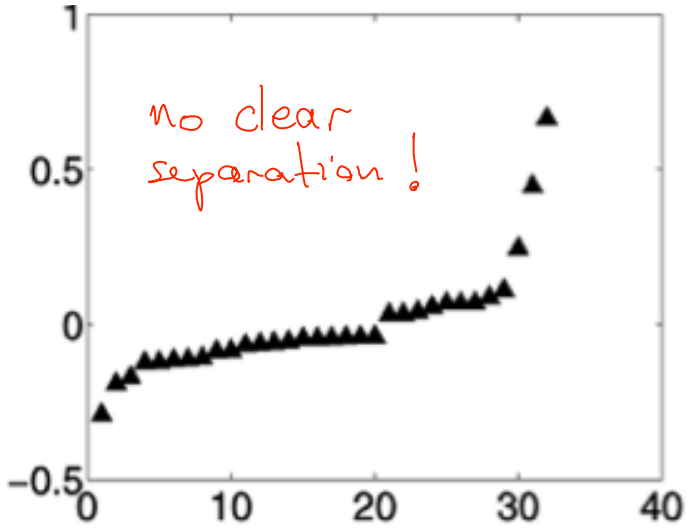




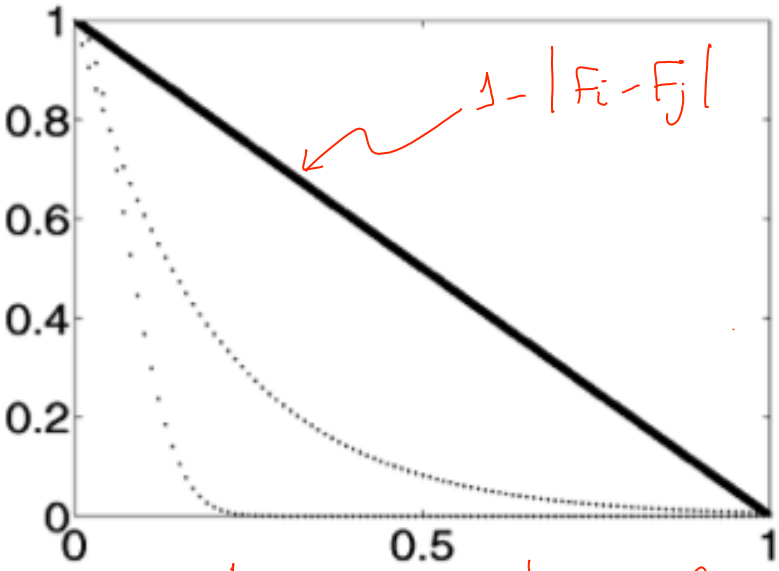
last eigenvector of L

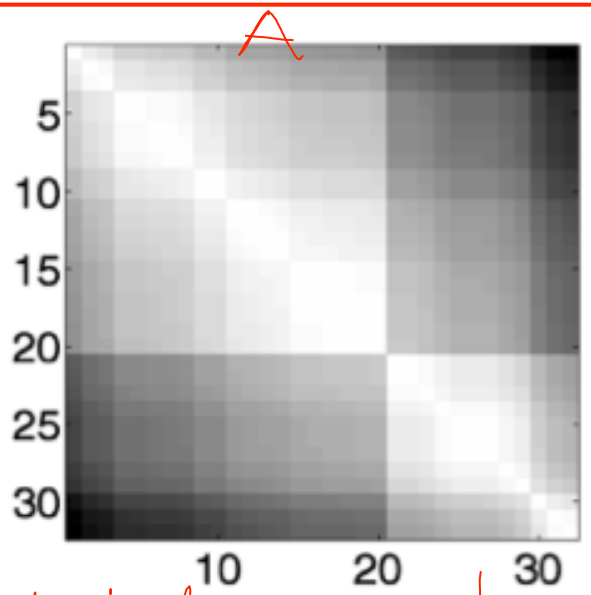


2nd-to-last eigenvector of L



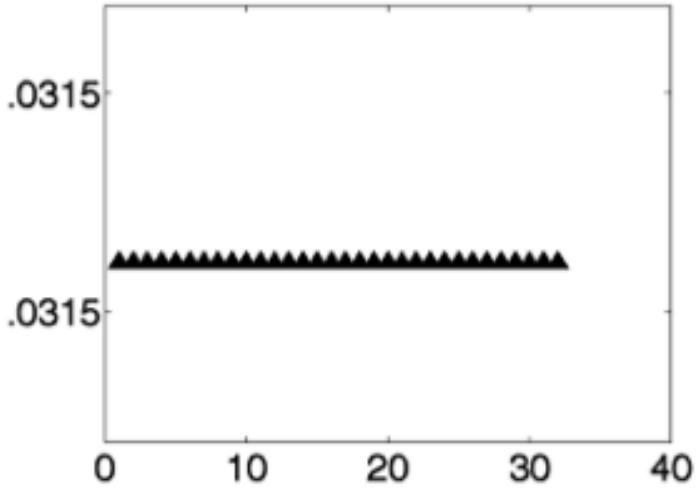
the minimum normalized cut (Ncut)

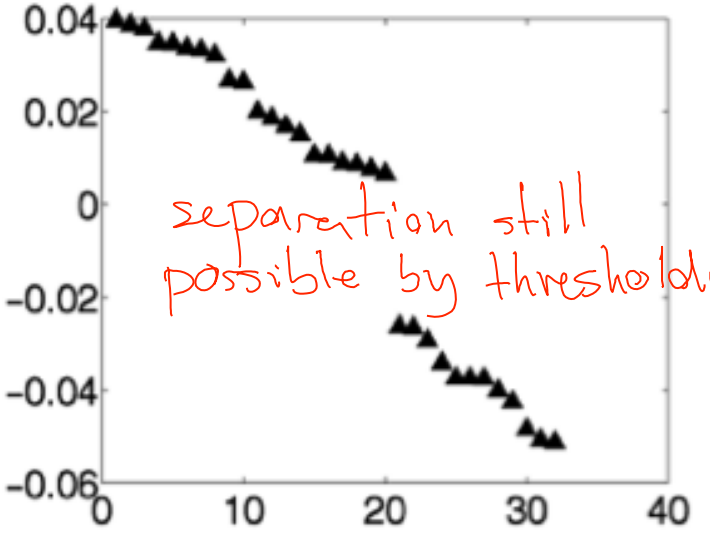




last eigenvector of L

2nd-to-last eigenvector of L





separation still possible by thresholding!

Ncut as a discrete optimization problem

Shi and Malik [12] prove that (15) is equivalent to the discrete optimization problem, for labelling $\vec{y} = (y_1, ..., y_N)^T$ with $N = |X|$,

$$\arg\min_{\vec{y}} \frac{\vec{y}^T(D-A)\vec{y}}{\vec{y}^T D\vec{y}}, \quad \text{subject to } y_i \in \{1, -b\} \text{ and } \vec{d}^T\vec{y} = 0 . \quad (16)$$

Here,

- A is the $N \times N$ symmetric matrix of affinities, $a(\vec{x}_i, \vec{x}_j)$, arranged (say) according to the raster ordering of the pixels;
- $\vec{d} = A\vec{1}$, where $\vec{1}$ is the N-vector comprising N ones;
- D is the diagonal matrix with $D_{i,i} = d_i$;
- $b > 0$.

compute $\arg\min_{F,G} L(F,G)$ with

$$L(F,G) = \frac{cut(F,G)}{\sum_{x_i \in F, x_j \in V} A_{ij}} + \frac{cut(F,G)}{\sum_{x_i \in G, x_j \in V} A_{ij}}$$

spectral approximation to the normalized cut

Shi and Malik [12] prove that (15) is equivalent to the discrete optimization problem, for labelling $\vec{y} = (y_1, ..., y_N)^T$ with $N = |X|$,

$$\arg\min_{\vec{y}} \frac{\vec{y}^T(D-A)\vec{y}}{\vec{y}^T D\vec{y}}, \quad \text{subject to } y_i \in \{1, -b\} \text{ and } \vec{d}^T\vec{y} = 0\,. \quad (16)$$

Here,

- A is the $N \times N$ symmetric matrix of affinities, $a(\vec{x}_i, \vec{x}_j)$, arranged (say) according to the raster ordering of the pixels;
- $\vec{d} = A\vec{1}$, where $\vec{1}$ is the N-vector comprising N ones;
- D is the diagonal matrix with $D_{i,i} = d_i$;
- $b > 0$.

Given a solution $\vec{y}$ of (16), the corresponding solution for partition F of (15) is then obtained by setting $F = \{\vec{x}_i \mid y_i > 0\}$. And, vice versa, given F we set $y_i = 1$ for each $\vec{x}_i \in F$, and set the other elements of $\vec{y}$ to $-b$, where $b > 0$ is chosen such that $\vec{d}^T\vec{y} = 0$.

spectral approximation to the normalized cut

The relaxed version of (16), ignoring the binary constraint $y_i \in \{1, -b\}$, and taking $\vec{y}$ to be real-valued, is a standard problem in linear algebra known as the Rayleigh Quotient.

- The solution $\vec{y}$ is the generalized eigenvector of the Laplacian matrix, $D - A$, with the second smallest eigenvalue μ:

$$(D - A)\,\vec{y} \;=\; \mu\, D\,\vec{y}\,. \tag{20}$$

To obtain a discrete (approximate) solution to (16), Shi and Malik threshold $\vec{y}$. They consider several thresholds τ, and choose the one with the lowest value of the Ncut objective function (14). Regions, F and $G = V - F$, are recursively partitioned using the same method, until a user-specified number of segments are found. (see pp.5-6.)

spectral approximation to the normalized cut

To obtain a discrete (approximate) solution to (16), Shi and Malik threshold $\vec{y}$. They consider several thresholds τ, and choose the one with the lowest value of the Ncut objective function (14). Regions, F and $G = V - F$, are recursively partitioned using the same method, until a user-specified number of segments are found. (see pp.5-6.)

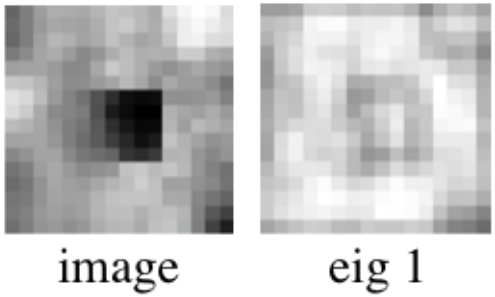


image eig 1

eig 2 eig 3

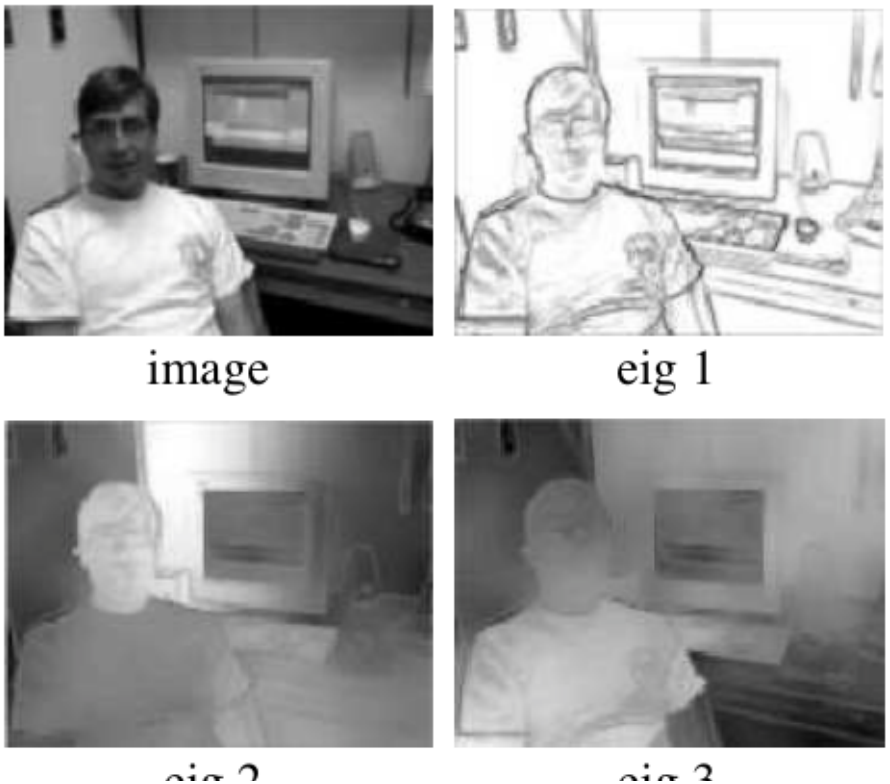


image eig 1

eig 2 eig 3

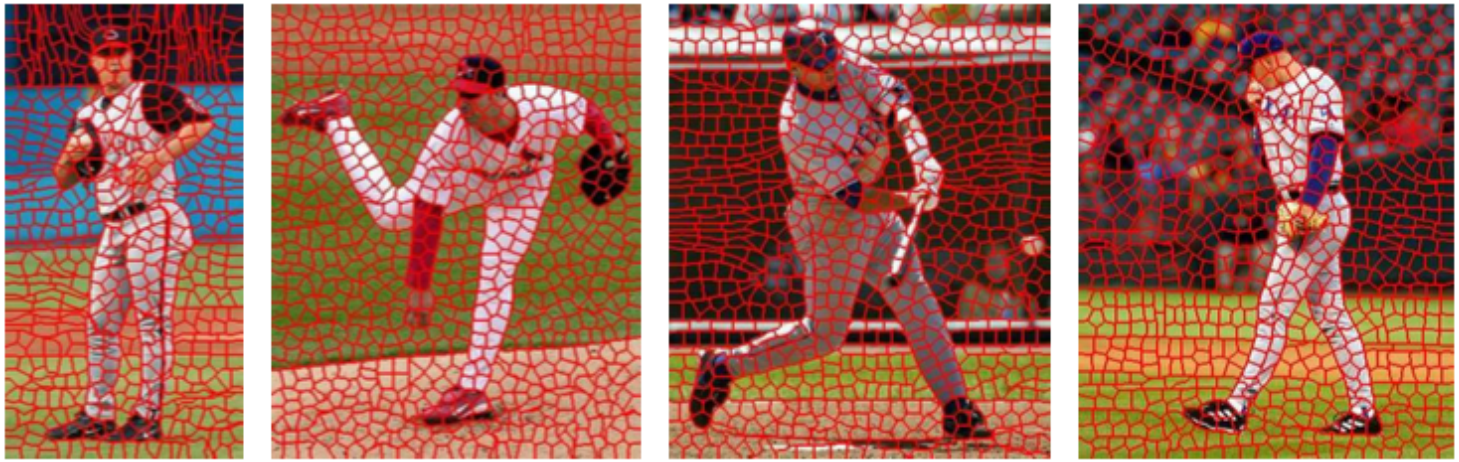
remarks on normalized cuts

The step of thresholding the second largest eigenvector to provide a partitioning proposal is a key limitation. In practice, the approximation only appears to be consistently reliable when there is just one clear way to partition the data. Yu and Shi [13] attempt to alleviate this problem by extracting K segments from the subspace spanned by the K eigenvectors of $I - B$ with the smallest eigenvalues.

Ncuts can be very slow for large images since the matrix becomes prohibitively large. Efficient approximations to the eigenvector problem have also been proposed.

remarks on normalized cuts (cont)

A common use of Ncuts is for the computation of significantly over-segmented images (called superpixels), to propose atomic (indivisible) regions as a basis for subsequent processing.



For further information, see the reading list in the CVPR 2004, graph-based segmentation tutorial [11].

Topic 15:

Image Segmentation

- intro to segmentation
- the affinity matrix
- spectral methods
  - normalized cuts for hard segmentation
  - spectral matting for computing alpha mattes
- efficient graph-based methods
- density estimation methods
- a deeper look at affinity functions

extension to soft segmentation: alpha matting

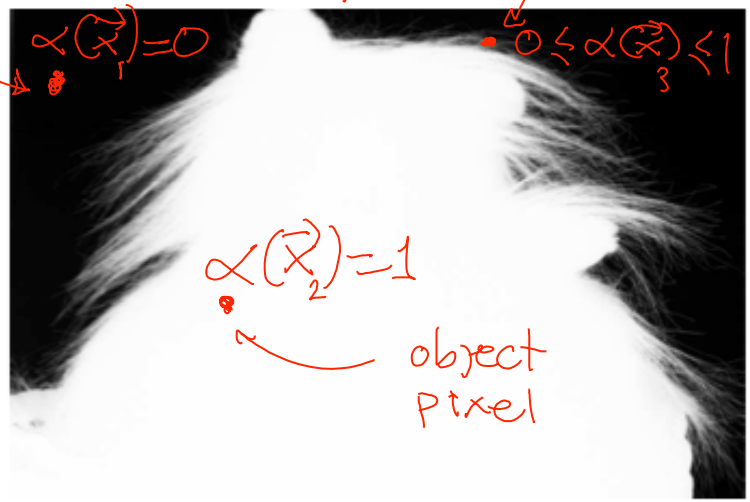
Input

Given an input photo, compute a transparency map (aka alpha matte) that enables object compositing

background pixel

semi-transparent pixel

Ground truth



Alpha Matte

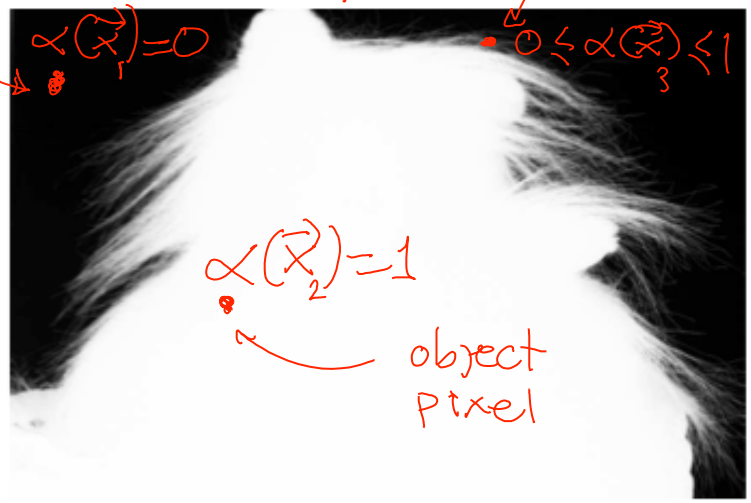
Composite

extension to soft segmentation: alpha matting

Input

Given an input photo, compute a transparency map (aka alpha matte) that enables object compositing

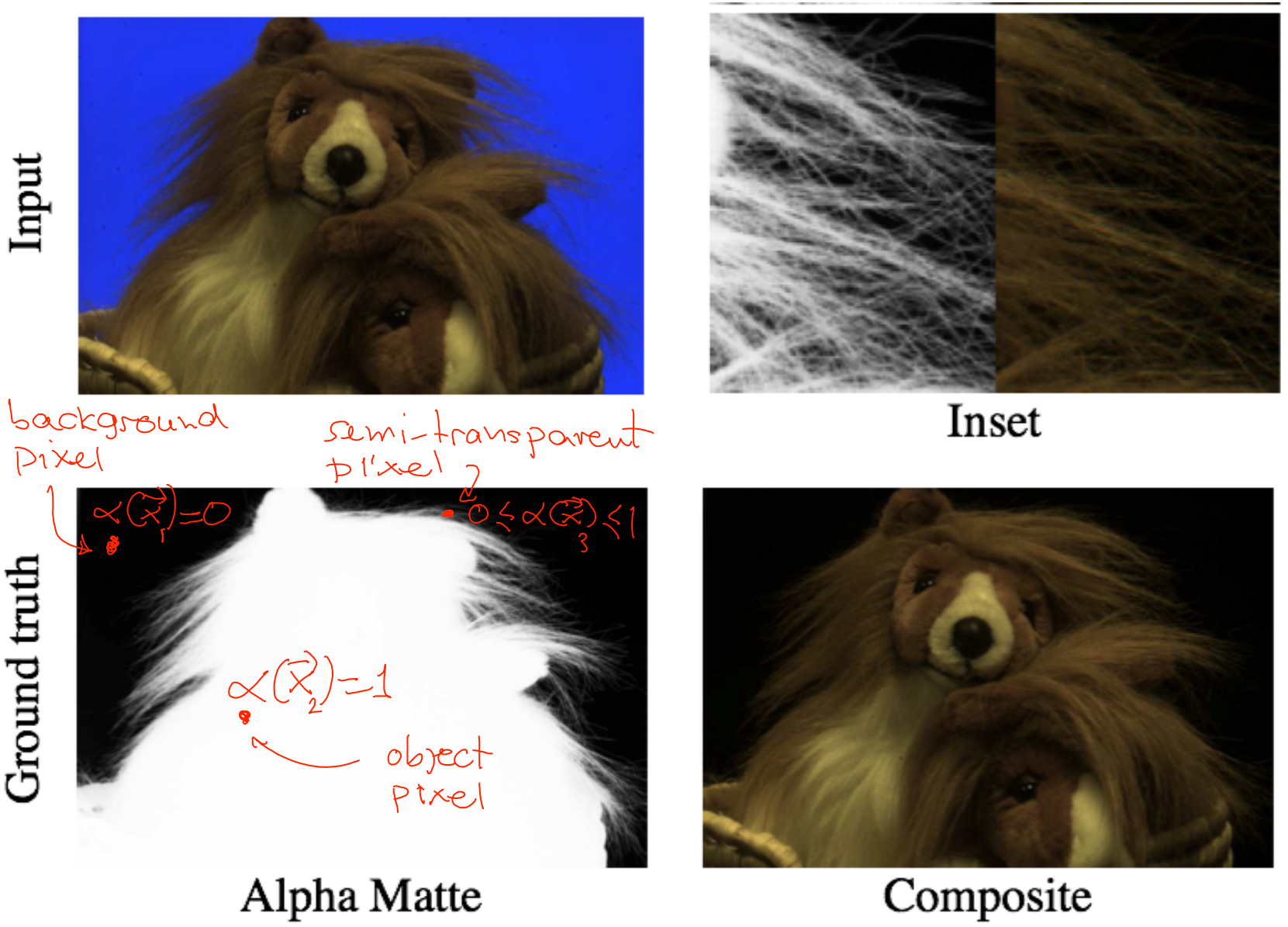
Ground truth



Alpha Matte

Composite

extension to soft segmentation: alpha matting



extension to soft segmentation: alpha matting

(Chuang et al, CVPR2000)

Accurate matting used to require considerable user interaction to get accurate results

extension to soft segmentation: alpha matting

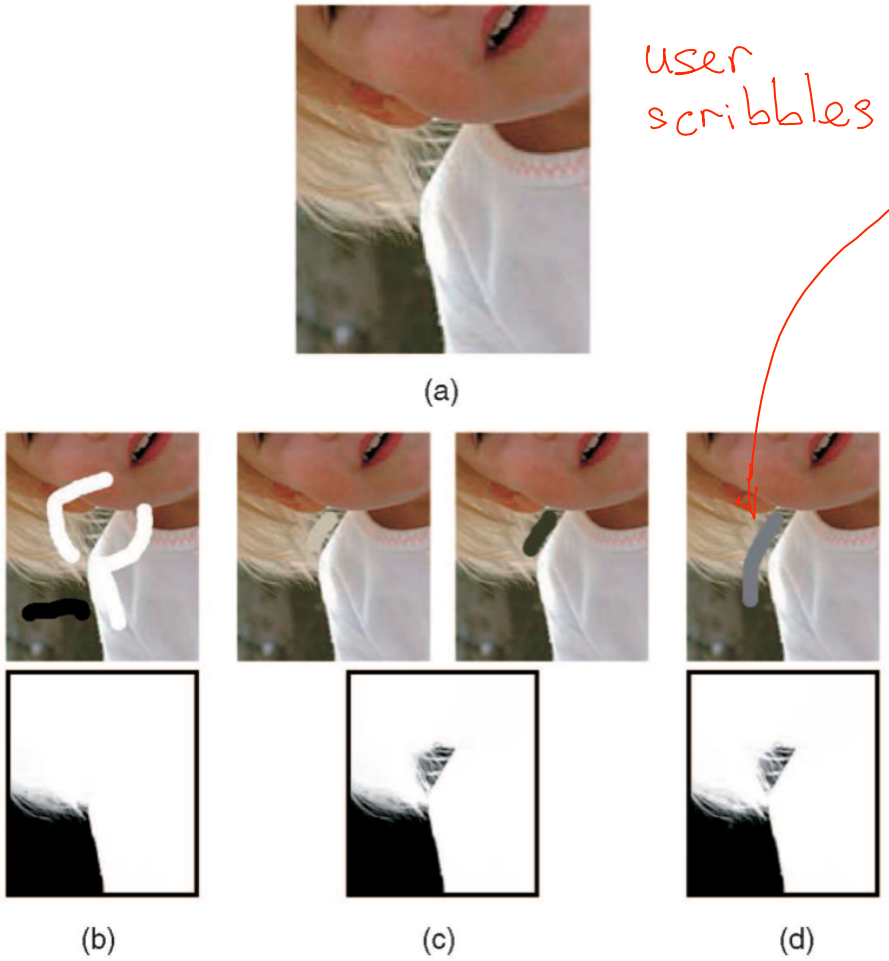
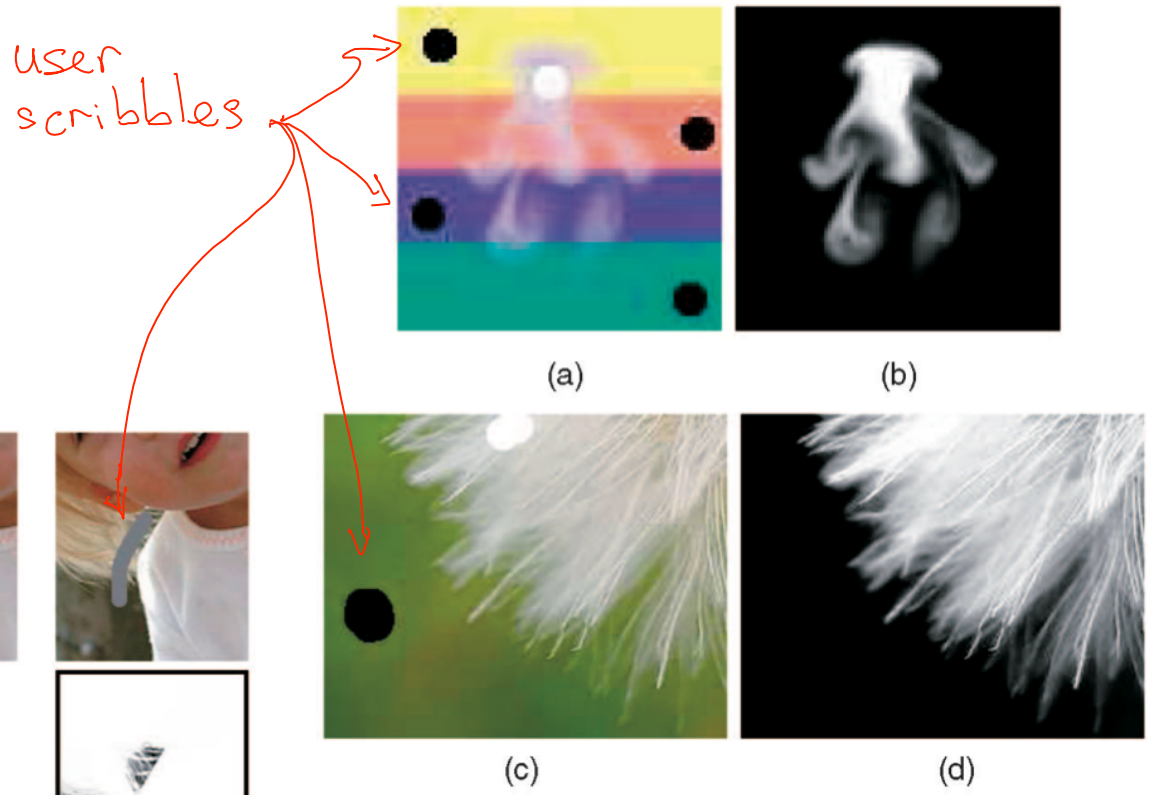


Fig. 5. Using additional scribbling brushes. (a) Input image. (b) Simple background (black) and foreground (white) scribbles. (c) Scribbling foreground and background colors explicitly. (d) Marking wider neighborhoods.



g. 4. Matting examples. (a) and (c) Input images with sparse nstraints. (b) and (d) Extracted mattes.

(Levin et al, PAMI2008)

spectral matting

Observation (Levin, Rav-Acha, Lischinski, PAMI 2008)

Claim 1. *Let $\alpha^1, \ldots, \alpha^K$ be the actual decomposition of the image I into k matting components. The vectors $\alpha^1, \ldots, \alpha^K$ lie in the nullspace of the matting Laplacian L (given by (6) with $\varepsilon = 0$) if every local image window w satisfies one of the following conditions:*

1. *A single component α^k is active within w.*
2. *Two components α^{k_1}, α^{k_2} are active within w and the colors of the corresponding layers F^{k_1}, F^{k_2} within w lie on two different lines in RGB space.*
3. *Three components α^{k_1}, α^{k_2} and α^{k_3} are active within w, each layer F^{k_1}, F^{k_2}, F^{k_3} has a constant color within w, and the three colors are linearly independent.*

matting Laplacian $\rightarrow L\alpha^k = \lambda\alpha^k$ for $\lambda = 0$

visualizing the matting components α^k

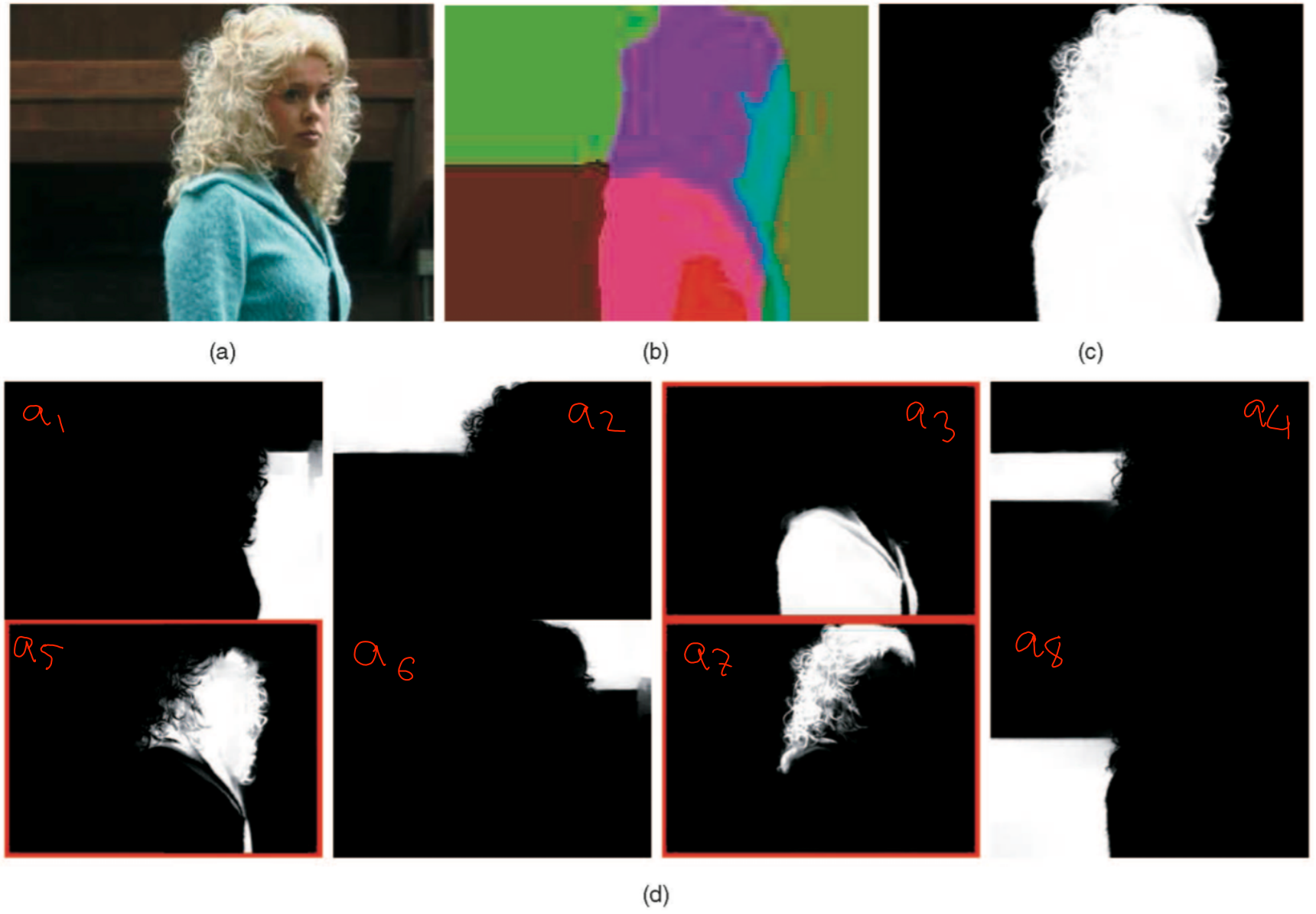


Fig. 1. Spectral segmentation and spectral matting. (a) Input image. (b) Hard segmentation. (c) Alpha matte. (d) Matting components computed by

the Laplacian eigenvectors y^k

$$Ly^k = \lambda y^k \quad \text{for} \quad \lambda = 0$$

Both $a^1, \ldots, a^k$ and $y^1, \ldots, y^k$ are in the nullspace of L. They are related by a rotation R that must be recovered

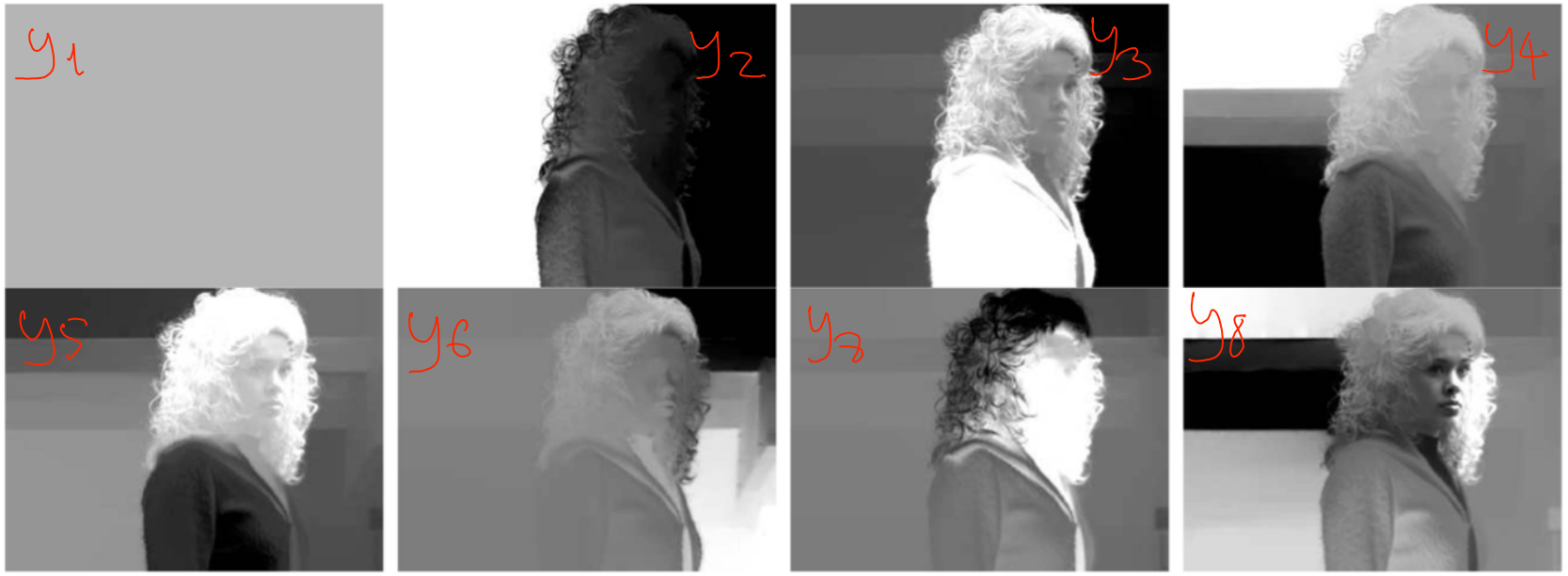


Fig. 2. The smallest eigenvectors of the matting Laplacian for the image in Fig. 1a. Linear combinations of these eigenvectors produced the matting components shown in Fig. 1d.

definition of the Matting Laplacian

$$L y^k = \lambda y^k \quad \text{for} \quad \lambda = 0$$

$L = \sum_q A_q$, each of which contains the affinities among pixels inside a local window w_q:

$$A_q(i,j) = \begin{cases} \delta_{ij} - \frac{1}{|w_q|} \\ \left(1 + (I_i - \mu_q)^T \left(\Sigma_q + \frac{\varepsilon}{|w_q|} I_{3\times 3}\right)^{-1} (I_j - \mu_q)\right) & (i,j) \in w_q \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Here, δ_{ij} is the Kronecker delta, μ_q is the 3×1 mean color vector in the window w_q around pixel q, Σ_q is a 3×3 covariance matrix in the same window, $|w_q|$ is the number of pixels in the window, and $I_{3\times 3}$ is the 3×3 identity matrix.

algorithm for computing alpha mattes

Given an image & desired number C of matting components

1. Form the matting Laplacian L
2. Compute the bottom K eigenvectors $y^1, y^2, \ldots, y^K$ of L
3. Compute the C×N matrix E that yields the "best" matting components

incorporates user input

see PAMI paper for details

$$[a^1\ a^2 \cdots a^C] = E\,[y^1\ y^2 \cdots y^K]$$

example using scribbles for guidance

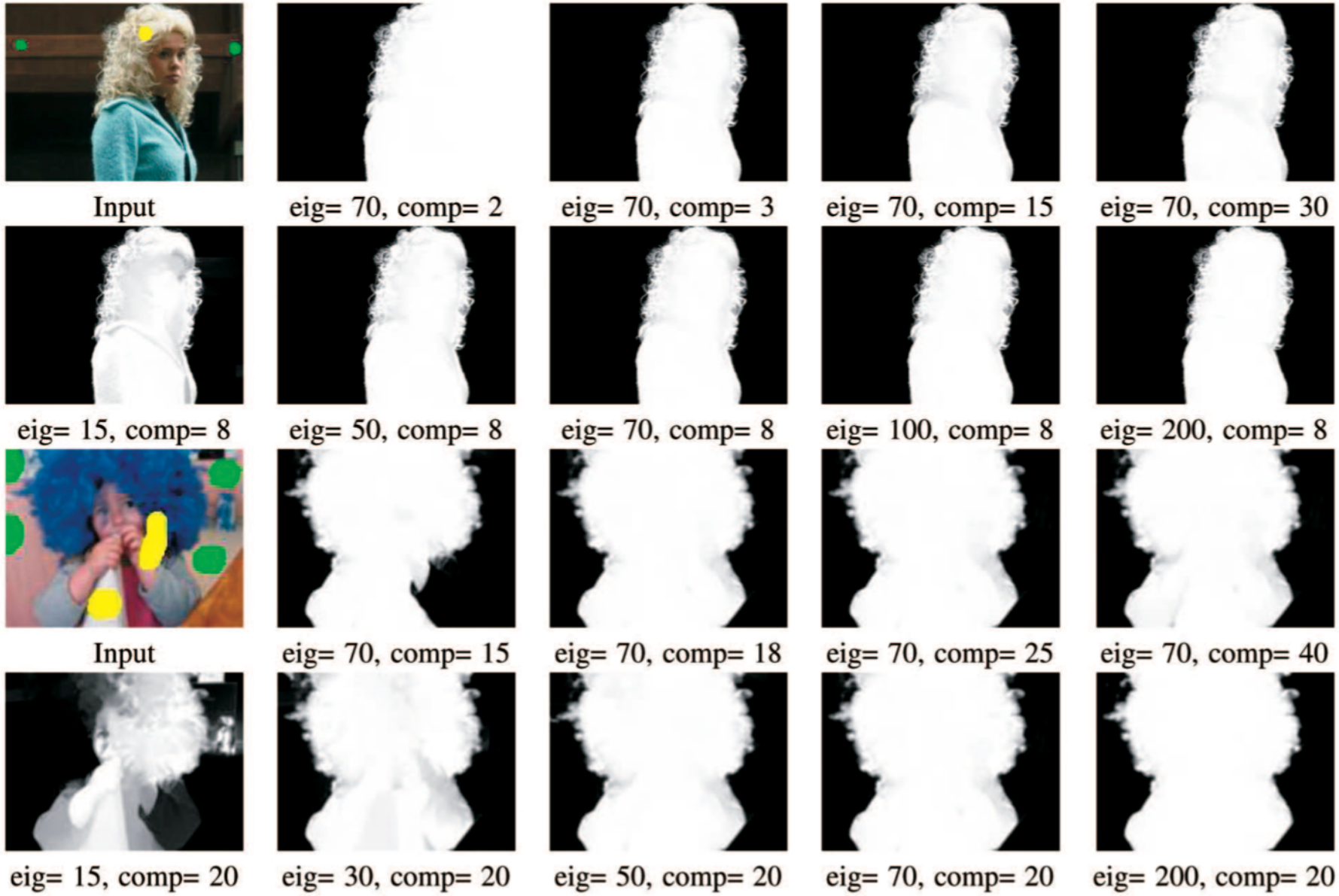


Fig. 12. The effect of changing the number of eigenvectors and the number of matting components on the resulting matte

Topic 15:

Image Segmentation

- intro to segmentation
- the affinity matrix
- spectral methods
  - normalized cuts for hard segmentation
  - spectral matting for computing alpha mattes
- efficient graph-based methods
- density estimation methods
- a deeper look at affinity functions

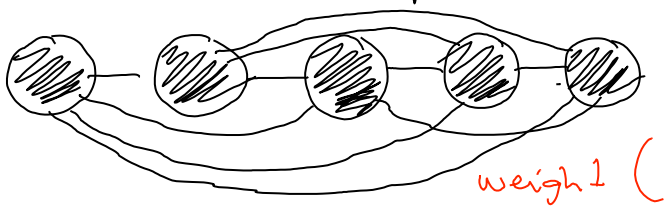
using efficient graph algorithms

eigenvalue computations are roughly $O(n^3)$
but Shi & Malik report $O(n^{3/2})$ in experiments
due to sparsity of A

↑ still very significant for large images

$\Rightarrow$ consider much simpler algorithms that operate in greedy fashion

weighted graph representation



$\text{weight}(x_i, x_j) = A_{ij}$

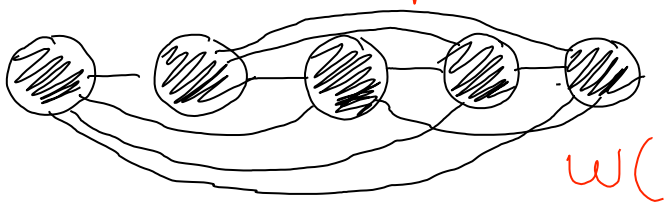
connected components (not robust)

A simple approach is to delete all edges between dissimilar pixels (i.e., with weights $w(\vec{x}_i, \vec{x}_j) > \tau$), and then seek connected components (CCs) in the remaining graph.

An efficient way to do CC clustering, with a variable τ, is to first build a minimal spanning tree (MST) of the graph. **Kruskal's Algorithm** is a greedy approach that is guaranteed to give an optimal MST:

- Begin with the completely disconnected graph.
- Add edges one at a time in increasing order of their weights, so long as adding an edge does not introduce cycles in the sub-graph.

weighted graph representation



notice change in definition of edge weights

$$w(x_i, x_j) = 1 - A_{ij}$$

connected components (not robust)

A simple approach is to delete all edges between dissimilar pixels (i.e., with weights $w(\vec{x}_i, \vec{x}_j) > \tau$), and then seek connected components (CCs) in the remaining graph.

An efficient way to do CC clustering, with a variable τ, is to first build a minimal spanning tree (MST) of the graph. **Kruskal's Algorithm** is a greedy approach that is guaranteed to give an optimal MST:

- Begin with the completely disconnected graph.
- Add edges one at a time in increasing order of their weights, so long as adding an edge does not introduce cycles in the sub-graph.

weighted graph representation

connected components (not robust)

A simple approach is to delete all edges between dissimilar pixels (i.e., with weights $w(\vec{x}_i, \vec{x}_j) > \tau$), and then seek connected components (CCs) in the remaining graph.

An efficient way to do CC clustering, with a variable τ, is to first build a minimal spanning tree (MST) of the graph. **Kruskal's Algorithm** is a greedy approach that is guaranteed to give an optimal MST:

- Begin with the completely disconnected graph.
- Add edges one at a time in increasing order of their weights, so long as adding an edge does not introduce cycles in the sub-graph.

weighted graph representation

connected components (not robust)

A simple approach is to delete all edges between dissimilar pixels (i.e., with weights $w(\vec{x}_i, \vec{x}_j) > \tau$), and then seek connected components (CCs) in the remaining graph.

An efficient way to do CC clustering, with a variable τ, is to first build a minimal spanning tree (MST) of the graph. **Kruskal's Algorithm** is a greedy approach that is guaranteed to give an optimal MST:

- Begin with the completely disconnected graph.
- Add edges one at a time in increasing order of their weights, so long as adding an edge does not introduce cycles in the sub-graph.

weighted graph representation

connected components (not robust)

A simple approach is to delete all edges between dissimilar pixels (i.e., with weights $w(\vec{x}_i, \vec{x}_j) > \tau$), and then seek connected components (CCs) in the remaining graph.

An efficient way to do CC clustering, with a variable τ, is to first build a minimal spanning tree (MST) of the graph. **Kruskal's Algorithm** is a greedy approach that is guaranteed to give an optimal MST:

- Begin with the completely disconnected graph.
- Add edges one at a time in increasing order of their weights, so long as adding an edge does not introduce cycles in the sub-graph.

$O(n \log n)$

weighted graph representation

connected components (not robust)

The CCs of the decimated graph (with edges having $w(\vec{x}_i, \vec{x}_j) > \tau$ removed) are then efficiently computed by deleting these same edges from the MST. The trees in the resulting forest provide the desired CCs.

Note that a single edge with $w(\vec{x}_i, \vec{x}_j) \leq \tau$ would be sufficient to cause two desired regions to be merged. Therefore CCs are not robust to stray links (aka "leaks") between regions. The consequence is that there is often no suitable value of τ that gives a useful segmentation.

weighted graph representation

local variation method

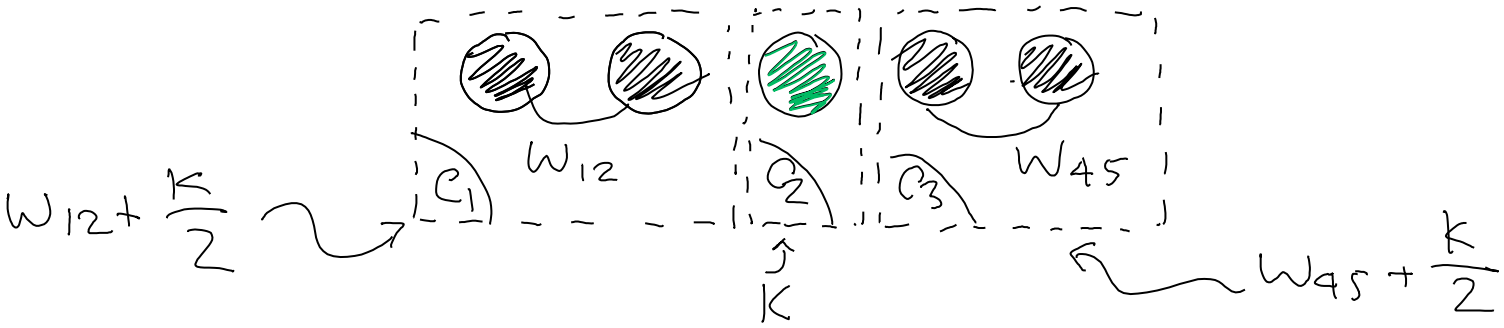
Felzenszwalb and Huttenlocher [8] introduce a simple but effective modification of Kruskal's algorithm. As in Kruskal's algorithm,

- it begins with the completely disconnected graph,
- edges are added one at a time in increasing order of their weights,
- it maintains a forest of MSTs for its current components.

During processing, each MST C_i is associated with a threshold

$$T(C_i) = w(C_i) + \frac{k}{|C_i|}, \tag{11}$$

where $w(C_i)$ is the maximum weight in the spanning tree C_i (i.e. the **local variation** of C_i). Also $k > 0$ is a constant, and $|C_i|$ is the number of pixels in C_i.



local variation method

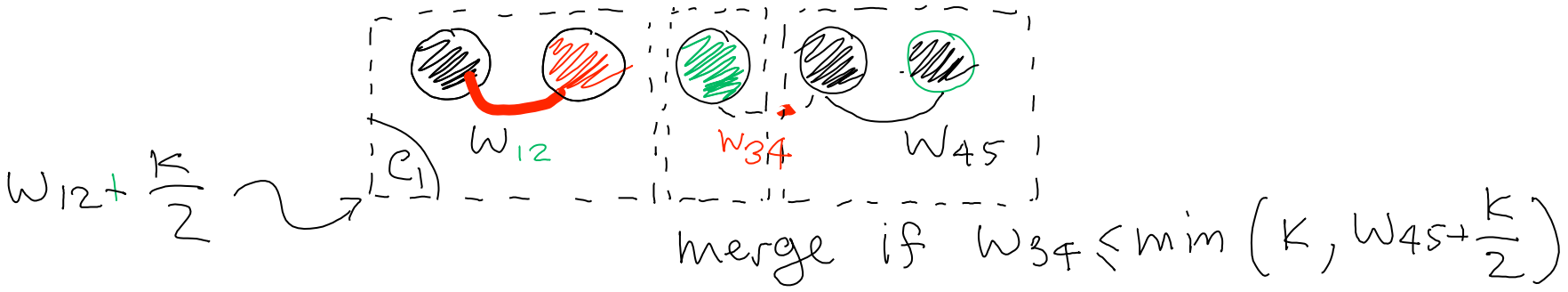
Suppose the edge $(\vec{x}_k, \vec{x}_l)$ is to be processed next, and its two endpoints are in two separate MSTs C_i and C_j. Then these MSTs are merged by adding the edge $(\vec{x}_k, \vec{x}_l)$ only if

$$w(\vec{x}_k, \vec{x}_l) \leq \min(T(C_i), T(C_j)) . \tag{12}$$

During processing, each MST C_i is associated with a threshold

$$T(C_i) = w(C_i) + \frac{k}{|C_i|} , \tag{11}$$

where $w(C_i)$ is the maximum weight in the spanning tree C_i (i.e. the **local variation** of C_i). Also $k > 0$ is a constant, and $|C_i|$ is the number of pixels in C_i.



local variation method

Suppose the edge $(\vec{x}_k, \vec{x}_l)$ is to be processed next, and its two endpoints are in two separate MSTs C_i and C_j. Then these MSTs are merged by adding the edge $(\vec{x}_k, \vec{x}_l)$ only if

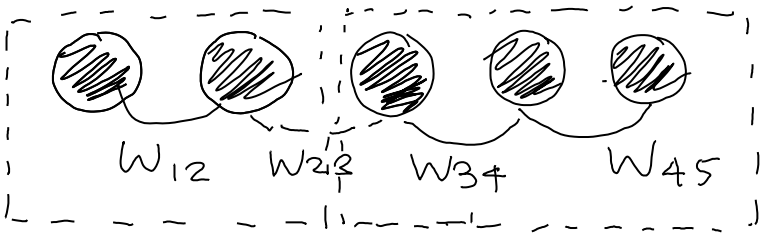
$$w(\vec{x}_k, \vec{x}_l) \leq \min(T(C_i), T(C_j)) . \quad (12)$$

During processing, each MST C_i is associated with a threshold

$$T(C_i) = w(C_i) + \frac{k}{|C_i|} , \quad (11)$$

where $w(C_i)$ is the maximum weight in the spanning tree C_i (i.e. the **local variation** of C_i). Also $k > 0$ is a constant, and $|C_i|$ is the number of pixels in C_i.

"slack" decreases as segment grows



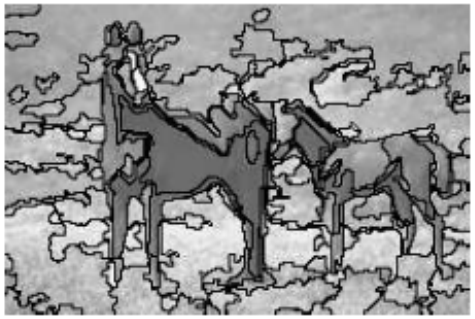
merge if $w_{23} \leq \min\left(w_{12} + \frac{k}{2}, \max(w_{34}, w_{45}) + \frac{k}{3}\right)$

examples of local variation segmentation

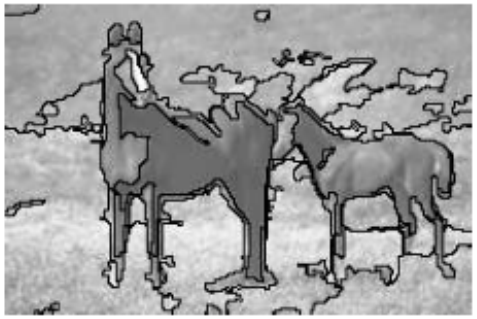
Sorting the edges according to weight causes the algorithm to grow relatively homogeneous regions first.

Parameter k in (11) roughly controls the size of the regions in the segmentation. Larger k yields a looser constraint (12), and more merging.

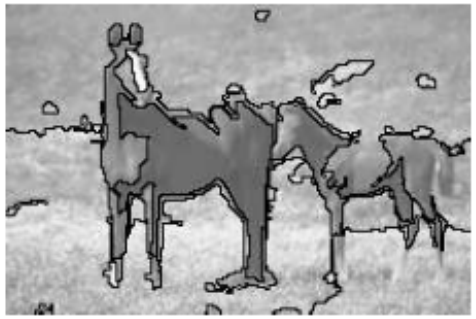
k = 50



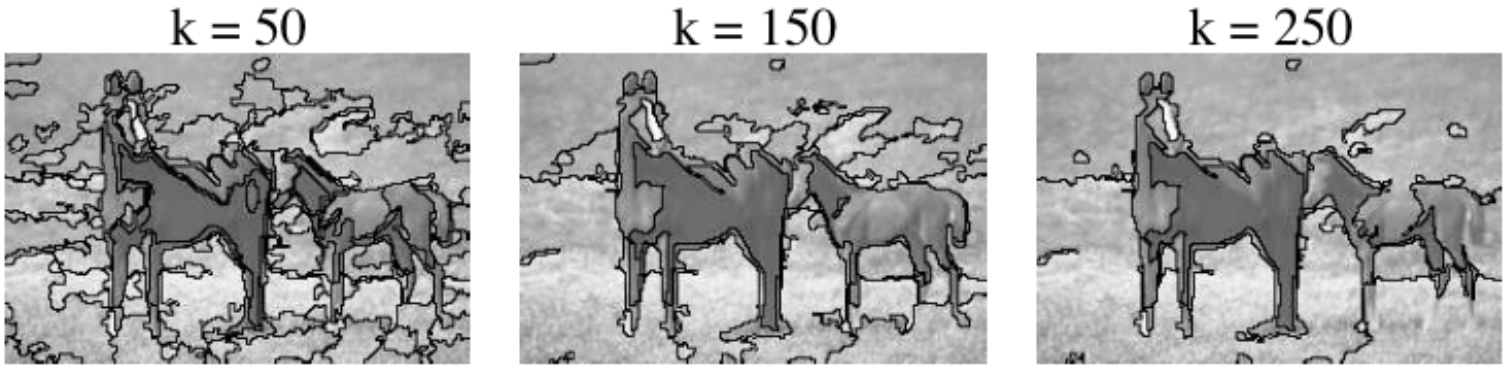
k = 150



k = 250



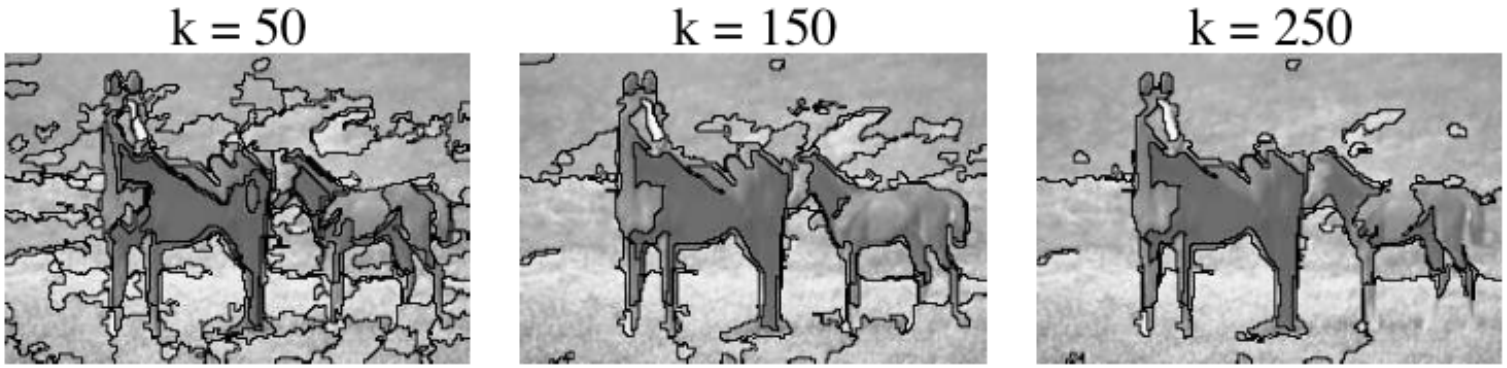
examples of local variation segmentation



Merging is sensitive to the local variation within regions. Due to the increasingly tight bound (11), a large homogeneous region C_i is only merged using edges with weights at most fractionally larger than $w(C_i)$, the largest affinity in the MST C_i. The bound is looser for small regions, encouraging their growth, thereby avoiding many tiny regions.

The approach has a tendency to produce narrow regions along 'true' segment boundaries (see examples above).

examples of local variation segmentation



Merging is sensitive to the local variation within regions. Due to the increasingly tight bound (11), a large homogeneous region C_i is only merged using edges with weights at most fractionally larger than $w(C_i)$, the largest affinity in the MST C_i. The bound is looser for small regions, encouraging their growth, thereby avoiding many tiny regions.

The weak link: But two very different regions can be merged if there is even one edge with a small weight that joins them.

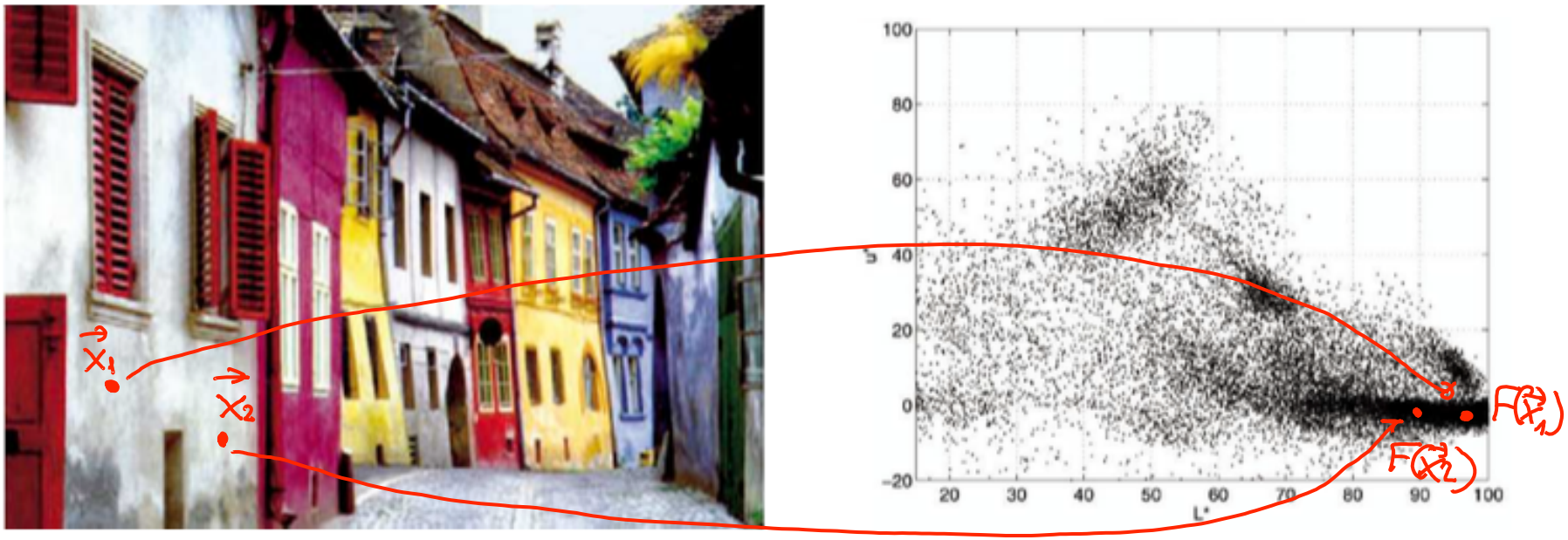
Topic 15:

Image Segmentation

- intro to segmentation
- the affinity matrix
- spectral methods
  - normalized cuts for hard segmentation
  - spectral matting for computing alpha mattes
- efficient graph-based methods
- density estimation methods
- a deeper look at affinity functions

probabilistic view of the segmentation problem

From Comaniciu and Meer [5].



Image

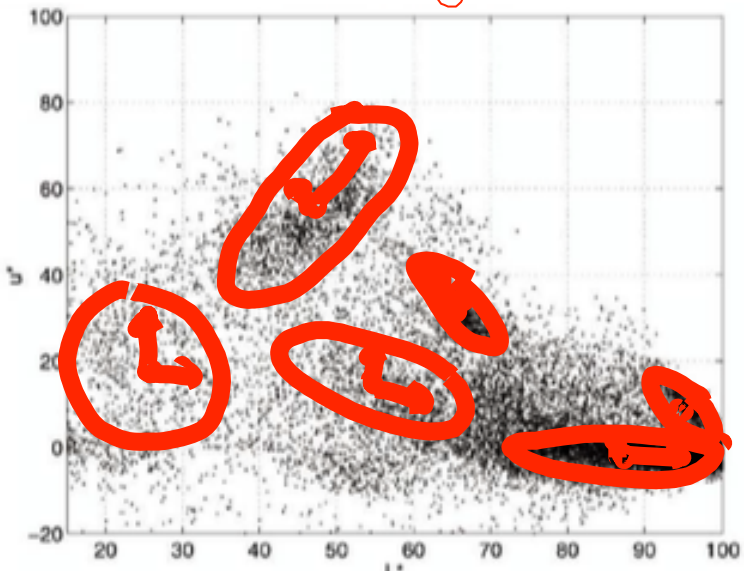
Scatter plot (2D L*u*)

The feature vectors of pixels in a segment follow a "hidden" segment-specific distribution

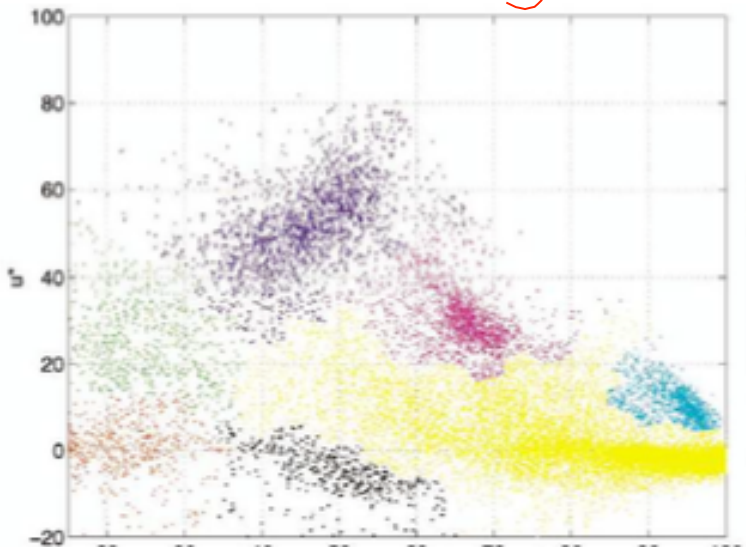
Our task is to identify feature vectors arising from the same underlying distribution

parametric vs. non-parametric methods

mixture-of-gaussians



mean-shift segmentation



- parametric: assume a specific distribution for the feature vectors & estimate its parameters

- non-parametric: distribution defined implicitly using the feature vectors themselves

mixture-of-gaussians model

A natural approach is then to model the observed feature vector distribution using a mixture of Gaussians (MoG) model M,

$$p(\vec{F}|M) = \sum_{k=1}^{K} \pi_k \, G(\vec{F} \mid \vec{m}_k, \Sigma_k).$$

Here, $\pi_k \geq 0$ are the mixing coefficients, with $\sum_{k=1}^{K} \pi_k = 1$. Further, $\vec{m}_k$ and Σ_k are the means and covariances of the component Gaussians.

- For a given K, the parameters of the MoG model $\{(\pi_k, \vec{m}_k, \Sigma_k)\}_{k=1}^{K}$ can be fit to the features $\{\vec{F}(\vec{x})\}_{\vec{x}\in X}$ using maximum-likelihood (X denotes the set of all pixels).

- Penalized likelihood (aka minimum description length (MDL)) can be used to select the number of components, K.

maximum ownership labelling

The segment label $c(\vec{x}) = k$ for a pixel $\vec{x}$ is the k which maximizes the ownership of $\vec{F}(\vec{x})$ in the MoG model M. That is,

$$c(\vec{x}) = \arg\max_k \left[\frac{\pi_k G(\vec{F}(\vec{x}) \mid \vec{m}_k, \Sigma_k)}{p(\vec{F}(\vec{x}) \mid M)} \right].$$

estimated using EM

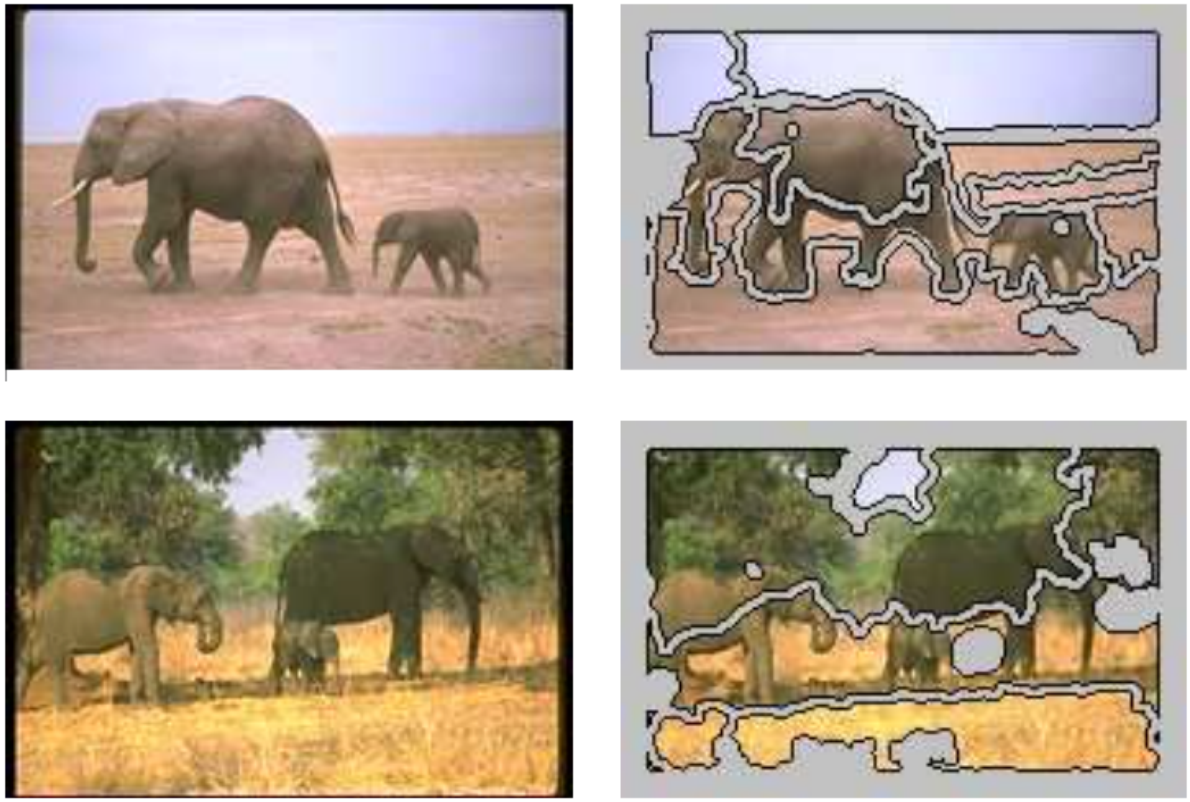
Above right: result for $K = 3$. The max-ownership image was post-processed using connected components. Small regions were discarded (gray). Right image: average colour for each remaining component. The width of the segment boundaries is due to the use of a spatial texture feature.

From Blobworld [3].

assumptions come home to roost…

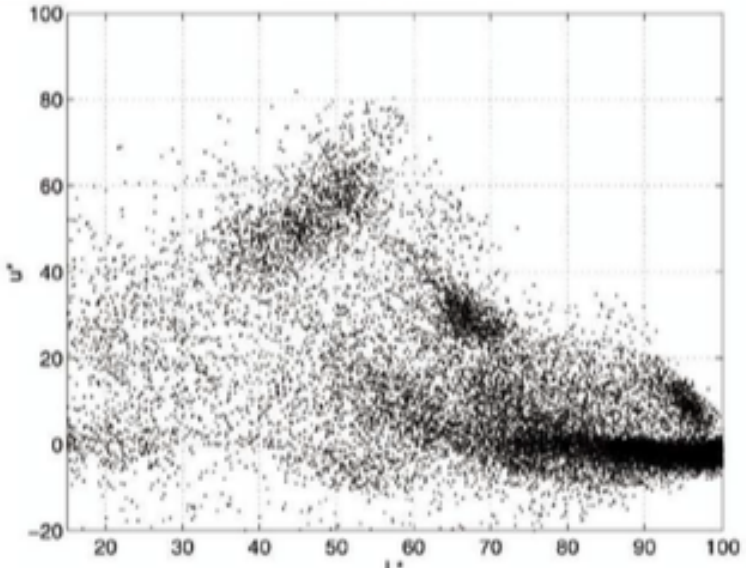
The quality of the resulting segmentation depends on the degree to which the given image matches the (implicit) assumptions we began with (and which led to the ML formulation); that is,

1. Different segments form compact, well-separated clusters in $\vec{F}$.
2. Gaussian components in M correspond to salient regions.

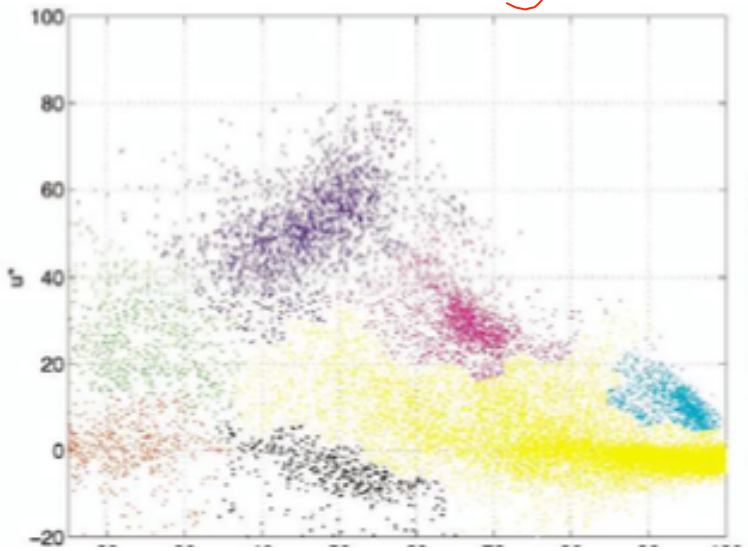


From Blobworld [3].

parametric vs. non-parametric methods



mean-shift segmentation



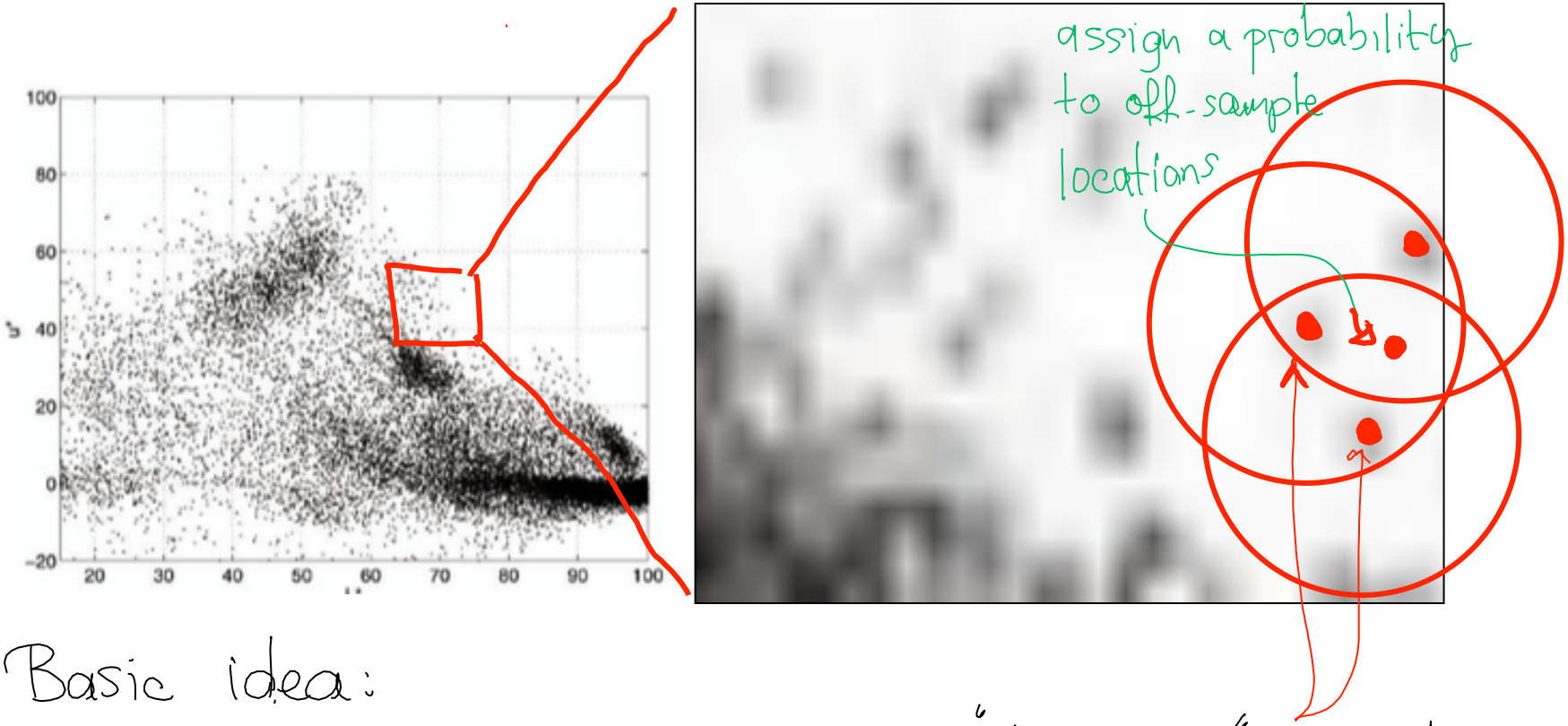
not a particularly good case for MoG modeling!

Topic 15:

Image Segmentation

- intro to segmentation
- the affinity matrix
- spectral methods
- efficient graph-based methods
- density estimation methods
  - parametric: mixture-of-Gaussians model
  - non-parametric: mean-shift segmentation

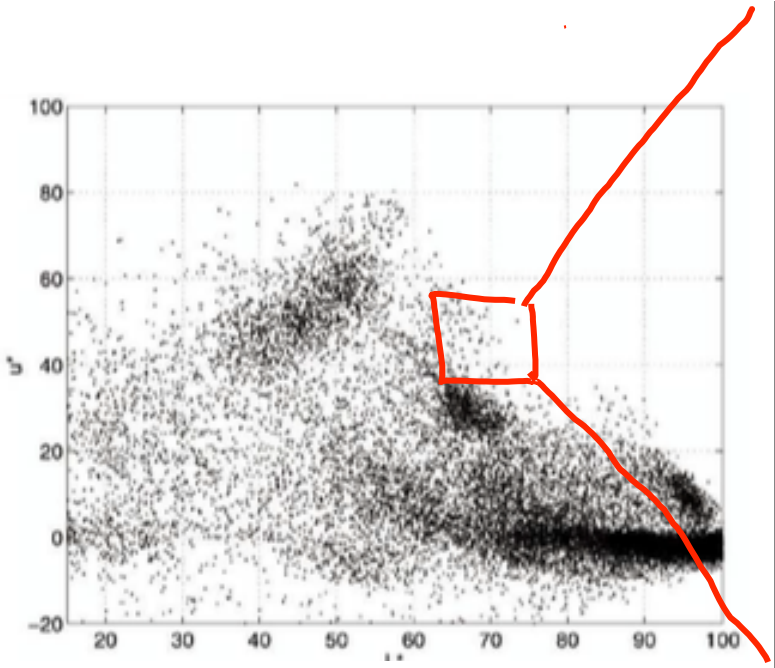
non-parametric density estimation

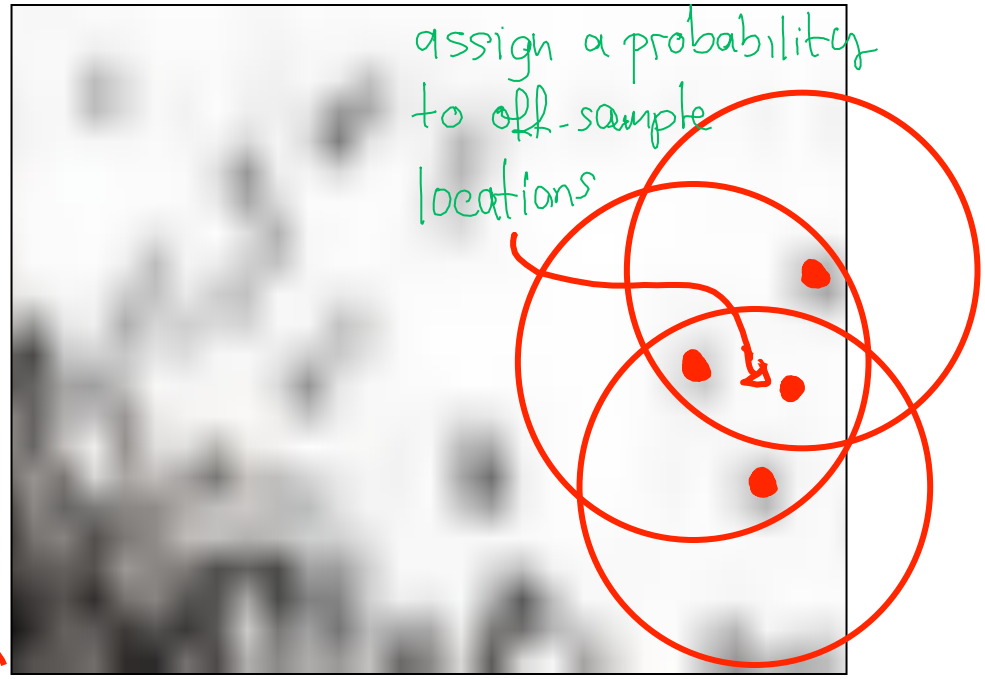


Basic idea:

- each feature vector is a "traning" sample from the underlyng austributiau
- define the distribution by interpolating between nearby samples

the non-parametric kernel density estimate





Desiderata:

- probability decreases as one moves away from a training sample
- integrates to 1 over the entire space
- is differentiable (for optimization purposes)

the non-parametric kernel density estimate

$$p_K(\vec{r}) = \frac{1}{|X|} \sum_{j=1}^{|X|} K(\vec{r} - \vec{r}_j)$$

* pixels in image

kernel / interpolation function

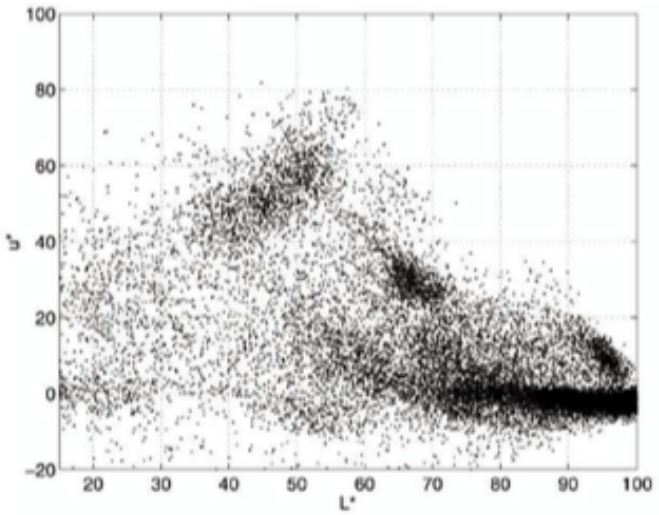
normalization constant so that $\int p_K(F)\,dF = 1$

Example kernels

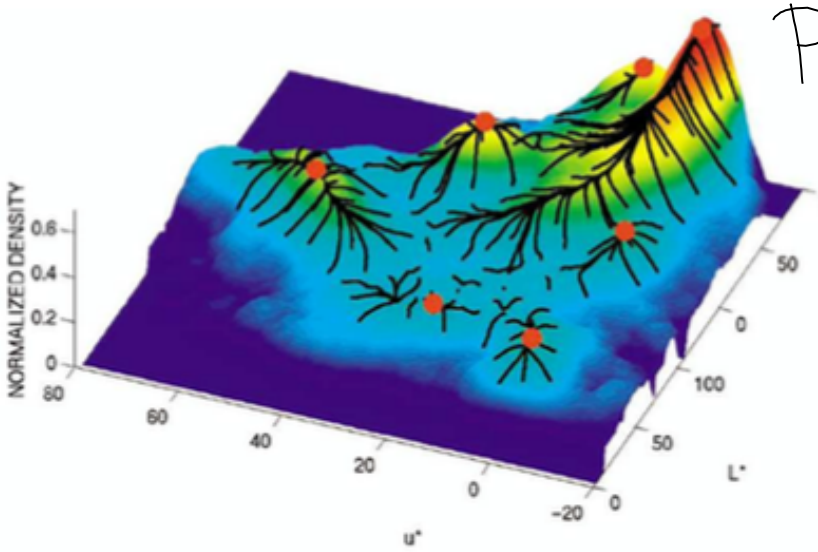
- Gaussian: $c\,e^{-(F-F_j)^T \Sigma^{-1} (F-F_j)}$ (inverse covariance matrix)
- Epanechnikov: $c \max\left(0, 1 - (F-F_j)^T \Sigma^{-1} (F-F_j)\right)$

the non-parametric density estimate

Image



Scatter plot (2D L*u*)



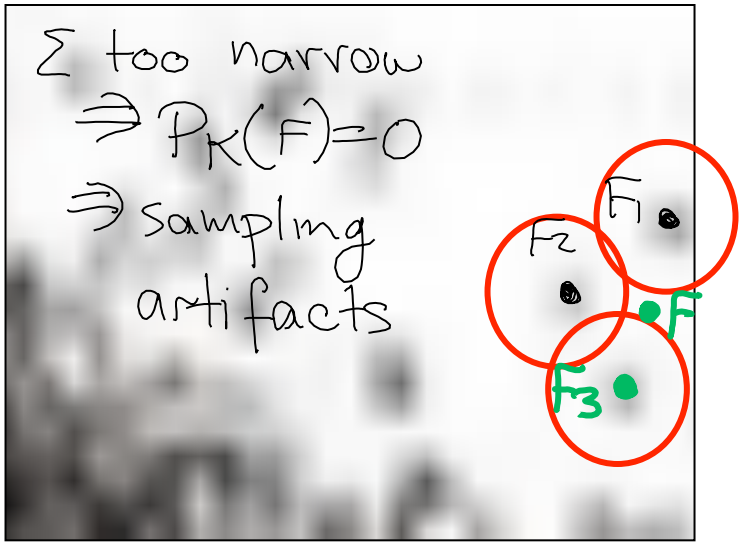
$P_K(F)$ computed with

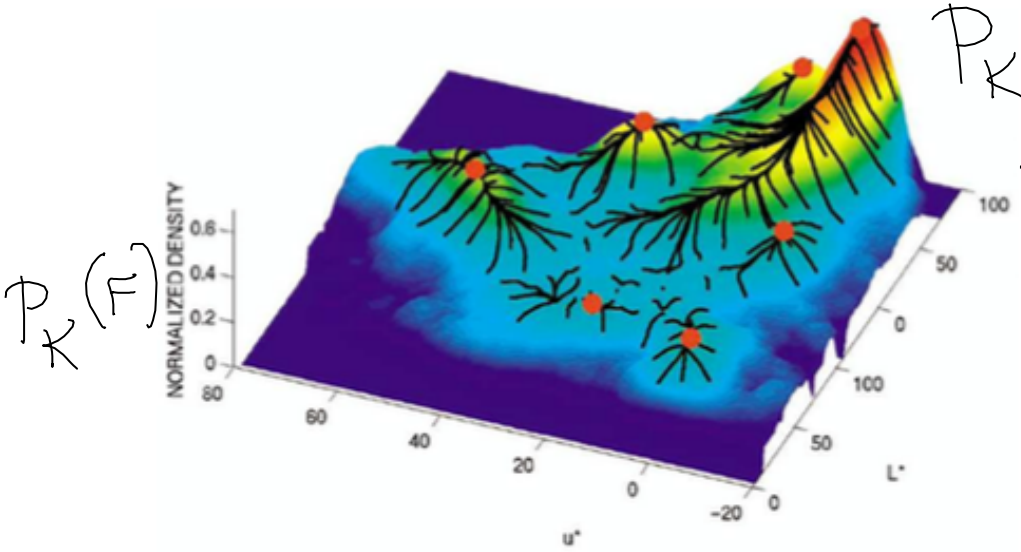
$K(F-F_j) =$ controls smoothness

$c \max(0, 1-(F-F_j)^T \Sigma^{-1} (F-F_j))$

the non-parametric density estimate

Image





$P_K(F)$ computed with

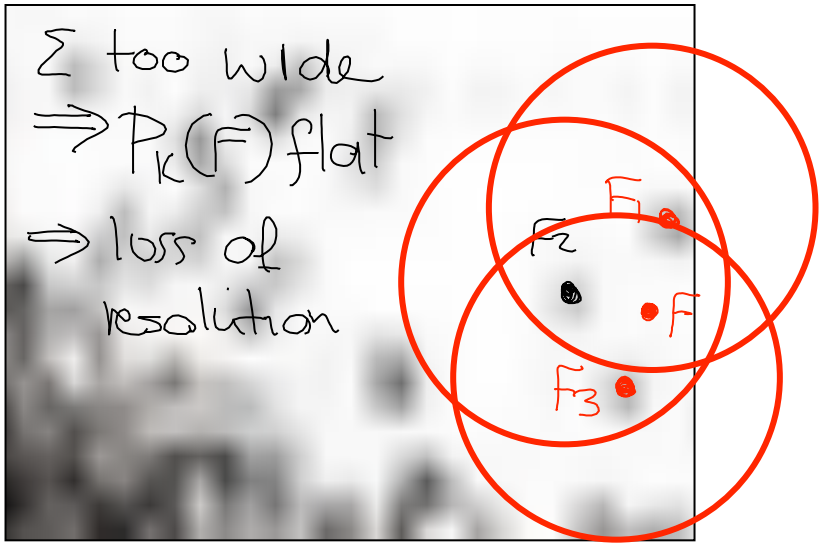
$K(F-F_j) =$ controls smoothness

$c \max(0,$

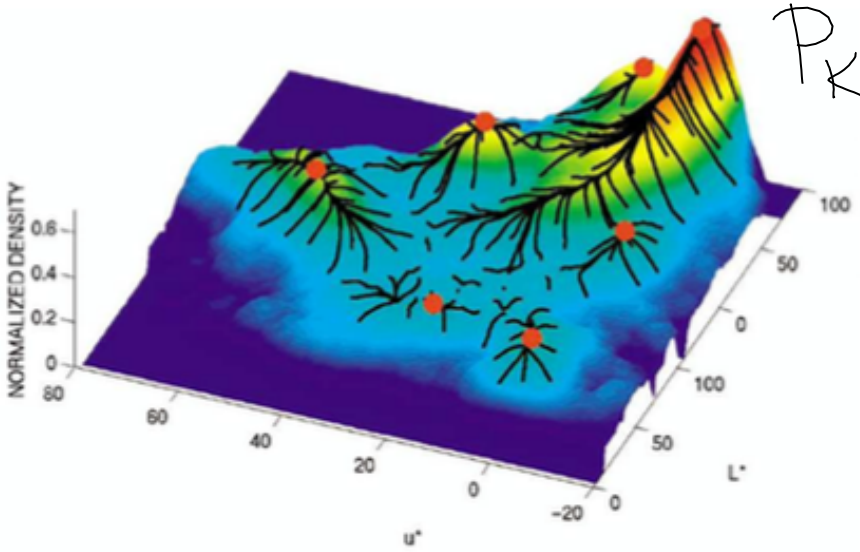
$1-(F-F_j)^T \Sigma^{-1} (F-F_j))$

the non-parametric density estimate

Image



$P_K(F)$



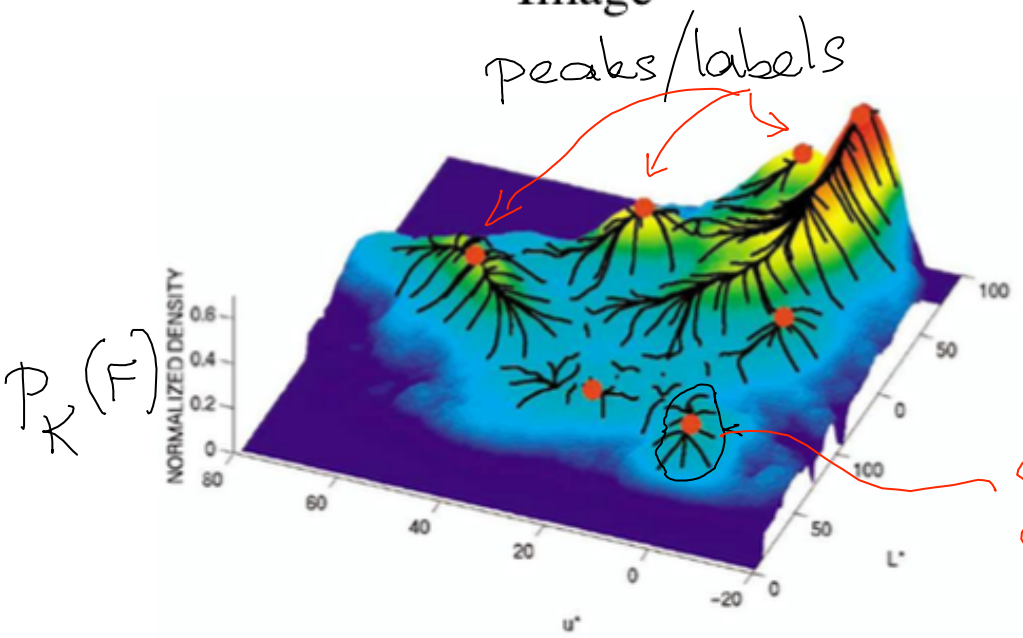
$P_K(F)$ computed with

$K(F-F_j) =$ $c \max(0,$ $1-(F-F_j)^T \Sigma^{-1} (F-F_j))$

controls smoothness

mean-shift segmentation

Image

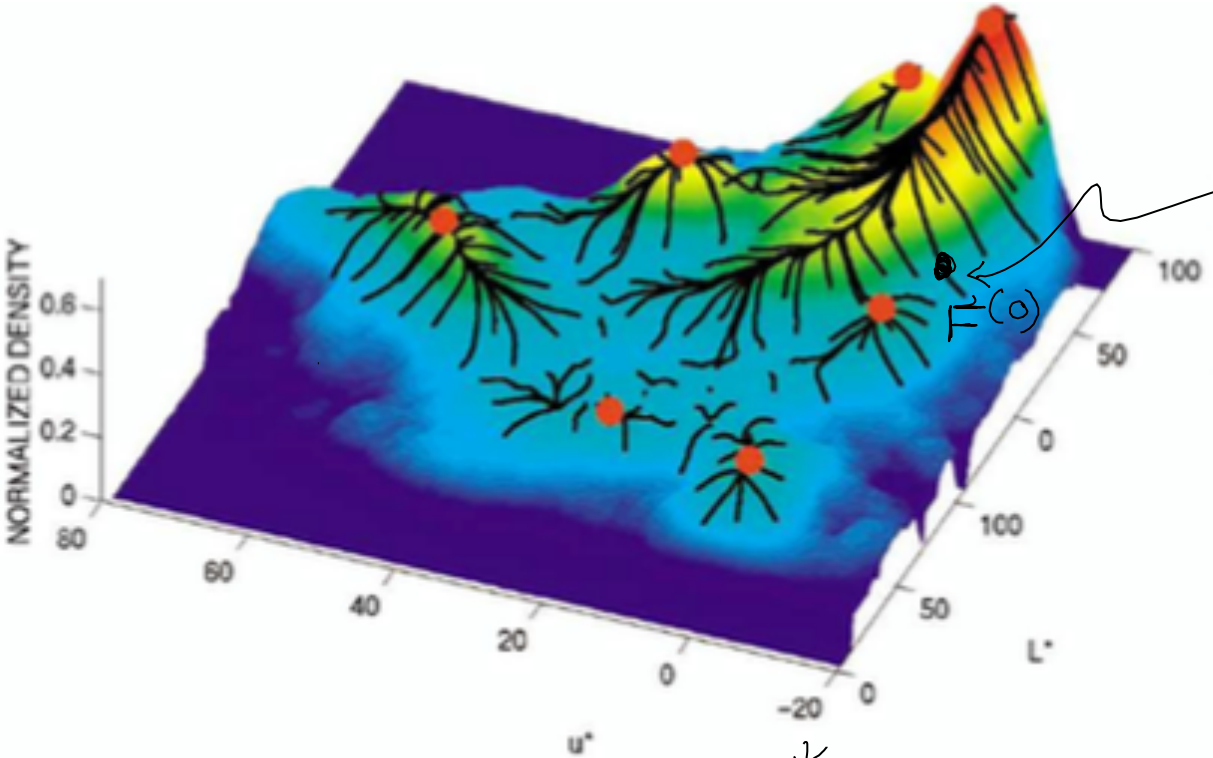


Main idea:

- peaks of $P_K(F)$ = segmentation labels
- feature vectors get the label of the peak in their neighborhood

all these feature vectors get the same label

finding peaks using the mean-shift procedure



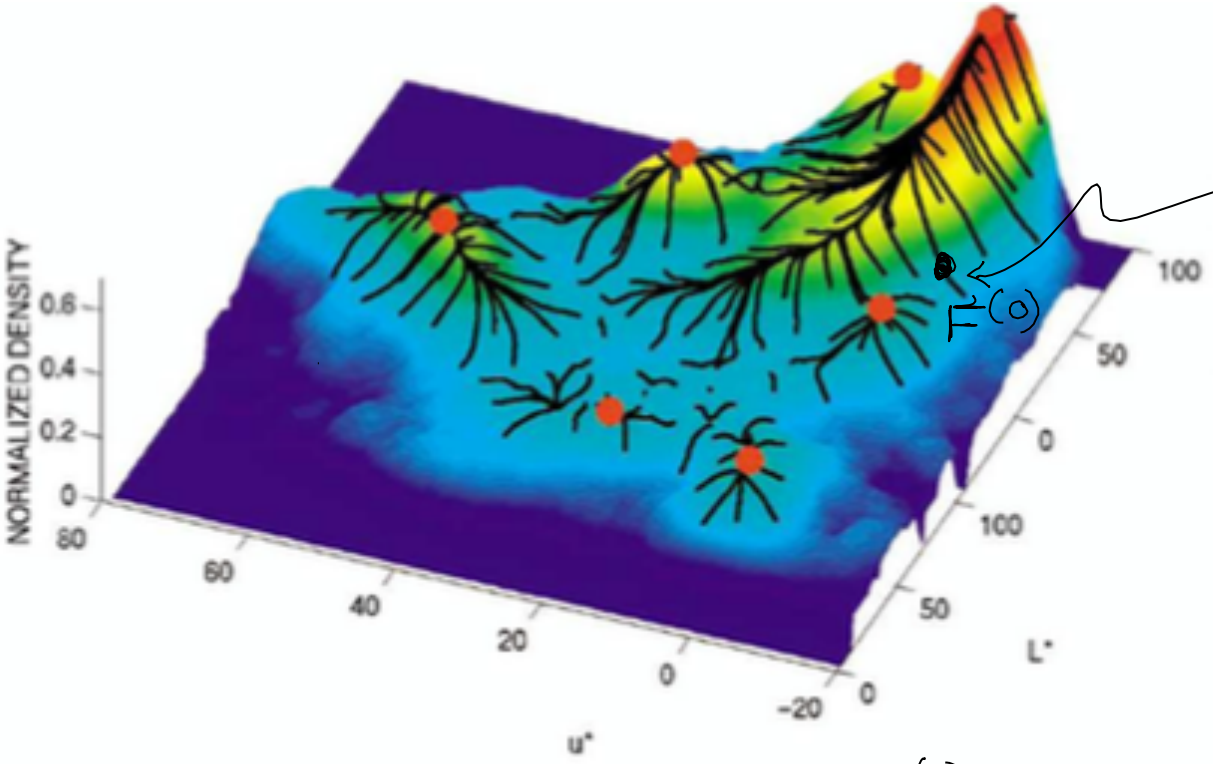
Given some initial vector $F^{(0)}$ we seek the local maximum $P_K(F^*)$ near $F^{(0)}$

Observations: if $F^* = \underset{F \text{ near } F^{(0)}}{\text{argmax}} P_K(F)$ then

(1) $$F^* = \frac{\sum_j w(F_j - F^*) F^j}{\sum_j w(F_j - F^*)}$$

(F^* is a weighted sum of all the F^j's)

finding peaks using the mean-shift procedure



Given some mitial vector $F^{(0)}$ we seek the local maximum $P_K(F^*)$ near $F^{(0)}$

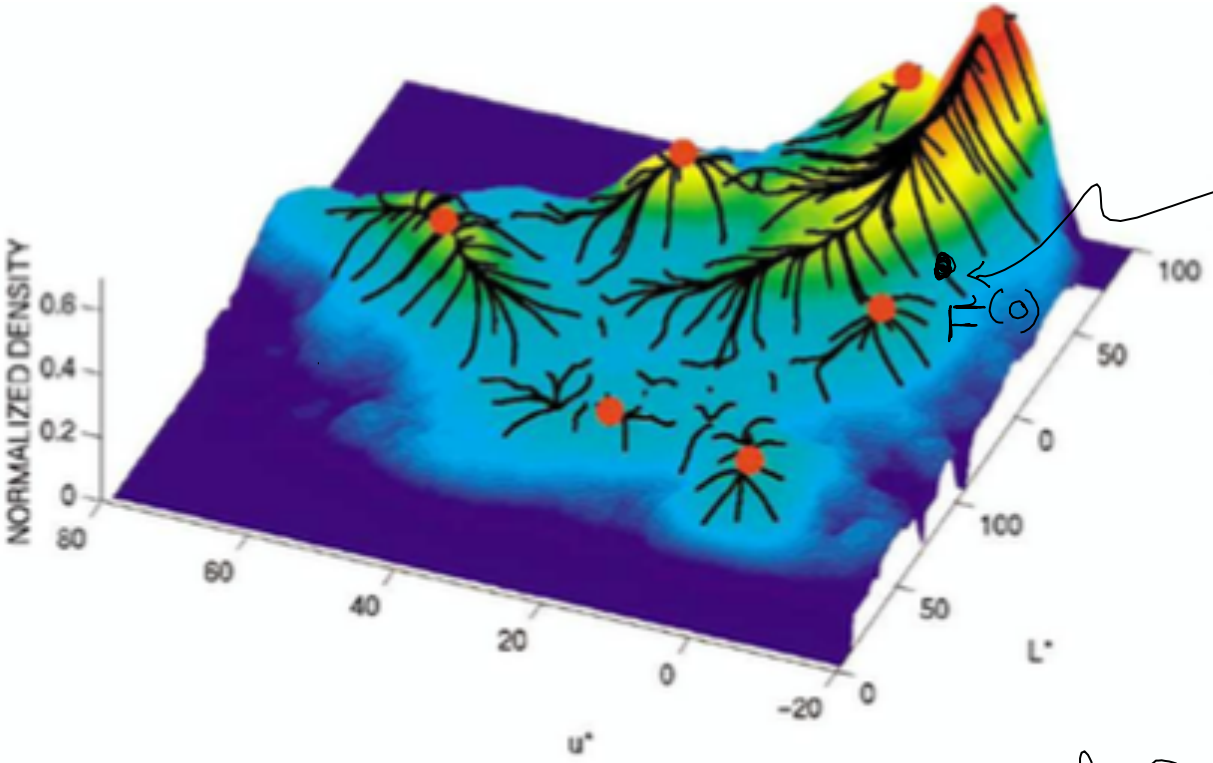
Observations

$$w(\vec{q}) = -\frac{dK}{ds}\left((F^j - F^*)^T \Sigma^{-1}(F^j - F^*)\right)$$

(1) $$F^* = \frac{\sum_j w(F_j - F^*) F^j}{\sum_j w(F_j - F^*)}$$

(derived from $\frac{\partial P_K}{\partial F}(F^*) = 0$)

finding peaks using the mean-shift procedure



Given some initial vector $F^{(0)}$ we seek the local maximum $P_K(F^*)$ near $F^{(0)}$

Observations:

weighted mean of feature points with weights centered on previous guess $F^{(t)}$

(2) $$F^{(t+1)} = \frac{\sum_j w(F_j - F^{(t)}) F^j}{\sum_j w(F_j - F^{(t)})}$$ converges to F^*

derivation of mean-shift procedure

(very similar to derivation of IRLS for robust estimation)

Let $\vec{F}_j \equiv \vec{F}(\vec{x}_j)$ be the feature vector at pixel $\vec{x}_j$. We define a kernel density estimate for the distribution of the image features $\vec{F}$ as follows:

$$p(\vec{F}) = \frac{1}{|X|} \sum_{j=1}^{|X|} k(s_j(\vec{F})) \,, \tag{5}$$

where $|X|$ is the number of pixels in the image, k be a non-negative kernel function, and, given a positive definite matrix C,

$$s_j(\vec{F}) \equiv (\vec{F} - \vec{F}_j)^T C^{-1} (\vec{F} - \vec{F}_j) \tag{6}$$

measures the scaled deviation between $\vec{F}$ and $\vec{F}_j$. We further assume that the kernel integrates to one in order for (5) to be a valid density function. And let $w(s) = \frac{dk}{ds}(s) \equiv k'(s)$ be the kernel derivative.

The goal of the mean-shift iterations is to find a trajectory to local maxima of $p(\vec{F})$, beginning at an arbitrary point in the feature space. A necessary condition for a local maxima of the kernel density estimate is that the gradient be zero. That is, for $\vec{F}^*$ to be a critical point it must satisfy

$$\frac{\partial p_k}{\partial \vec{F}}(\vec{F}^*) = \vec{0} \,. \tag{7}$$

derivation of mean-shift procedure

(very similar to derivation of IRLS for robust estimation)

Thus, to find critical points let's look at the form of the derivative of the feature distribution:

$$\begin{aligned}
\frac{\partial p_k}{\partial \vec{F}}(\vec{F}) &= \frac{1}{|X|} \sum_j \frac{\partial k}{\partial \vec{F}}(s_j(\vec{F})) \\
&= \frac{1}{|X|} \sum_j \frac{\partial k}{\partial s}\Big|_{s=s_j} \frac{\partial s_j}{\partial \vec{F}} \\
&= \frac{1}{|X|} \sum_j w(s_j(\vec{F})) \left[2C^{-1}\vec{F} - 2C^{-1}\vec{F}_j\right] \qquad (8)
\end{aligned}$$

With some algebraic simplification, we find that that the critical points satisfy

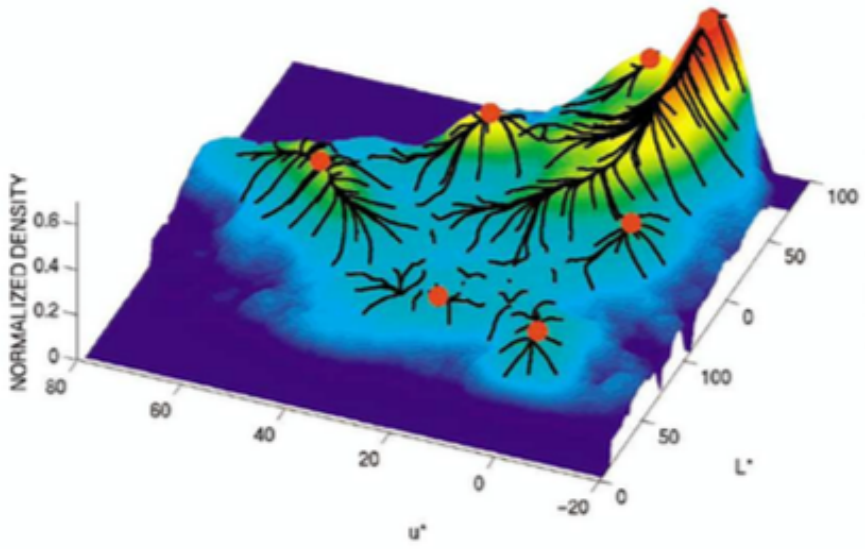
$$\left[\sum_j w(s_j(\vec{F}^*))\right] \vec{F}^* = \sum_j w(s_j(\vec{F}^*))\, \vec{F}_j \,. \qquad (9)$$

To find a critical point, given an initial guess, $\vec{F}^{(t)}$, we compute the weights using $\vec{F}^{(t)}$. Then, holding the weights fixed we solve for the next point on the trajectory toward the critical point; i.e.,

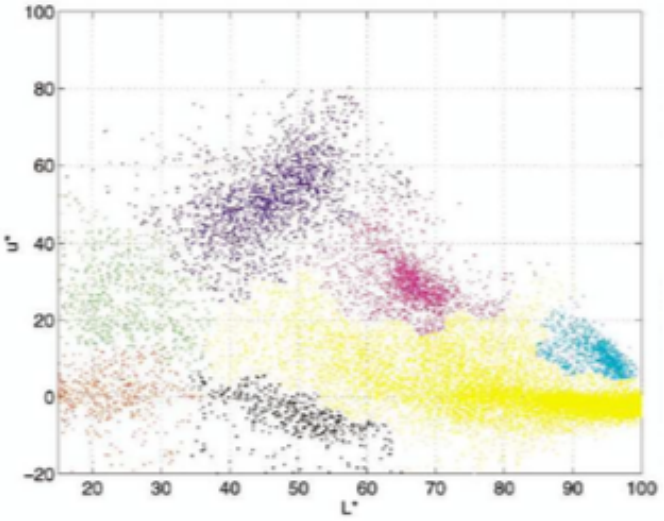
$$\left[\sum_j w(s_j(\vec{F}^{(t)}))\right] \vec{F}^{(t+1)} = \sum_j w(s_j(\vec{F}^{(t)}))\, \vec{F}_j, \qquad (10)$$

watersheds of mean-shift

The label at pixel $\vec{x}_0$ is defined to be the mode to which mean shift iterations (4) converge when started at $\vec{F}^{(0)} = \vec{F}(\vec{x}_0)$. That is, the *segments* are the **domains of convergence** (aka watersheds) of the mean-shift iterations (right plot).



Mean Shift Trajectories



Mean Shift Clusters

properties of mean-shift procedure

1. **Convergence:** Mean-shift iterations converge to a stationary point of $p_K(\vec{F})$ (see [5]).
2. **Anti-edge Detection:** The mean-shift iterations are repelled from local maxima of the norm of the gradient (wrt $\vec{x}$) of $\vec{F}^{\,T}(\vec{x})\Sigma^{-1}\vec{F}(\vec{x})$. This occurs, for example, at strong edges in the image $I(\vec{x})$.
3. **Fragmentation of Constant Gradient Regions:** The density $p_K(\vec{F}$ is constant (up to discretization artifacts) where the gradient of $\vec{F}^{\,T}(\vec{x})\Sigma^{-1}\vec{F}(\vec{x})$ is constant. Where $\vec{\nabla} I(\vec{x})$ is constant, points of $p_K(\vec{F})$ are stationary and mean-shift iterations stall (see left fig). Postprocessing is used to select salient local maxima (see [5]).

mean-shift segmentation algorithm

Feature vectors

$$F = \begin{pmatrix} \vec{x} \\ \vec{c} \end{pmatrix}$$

$\vec{x}$: pixel position (spatial domain)

$\vec{c}$: pixel color (range domain)

Kernel

Epanechnikov kernel

$$K\left(\begin{bmatrix} \vec{x} \\ \vec{c} \end{bmatrix}\right) = \frac{A}{h_s^2 h_r^3} k\left(\left\|\frac{x}{h_s}\right\|^2\right) k\left(\left\|\frac{\vec{c}}{h_r}\right\|^2\right)$$

h_s: spatial bandwidth

h_r: range bandwidth

mean-shift segmentation algorithm

1. Run mean-shift procedure for the image and store all information about the SD convergence point of each pixel
2. Group together all pixels whose convergence points are closer than h_s in the spatial domain and h_r in the range domain
3. (optional) eliminate segments with fewer than M pixels.

mean-shift segmentation results

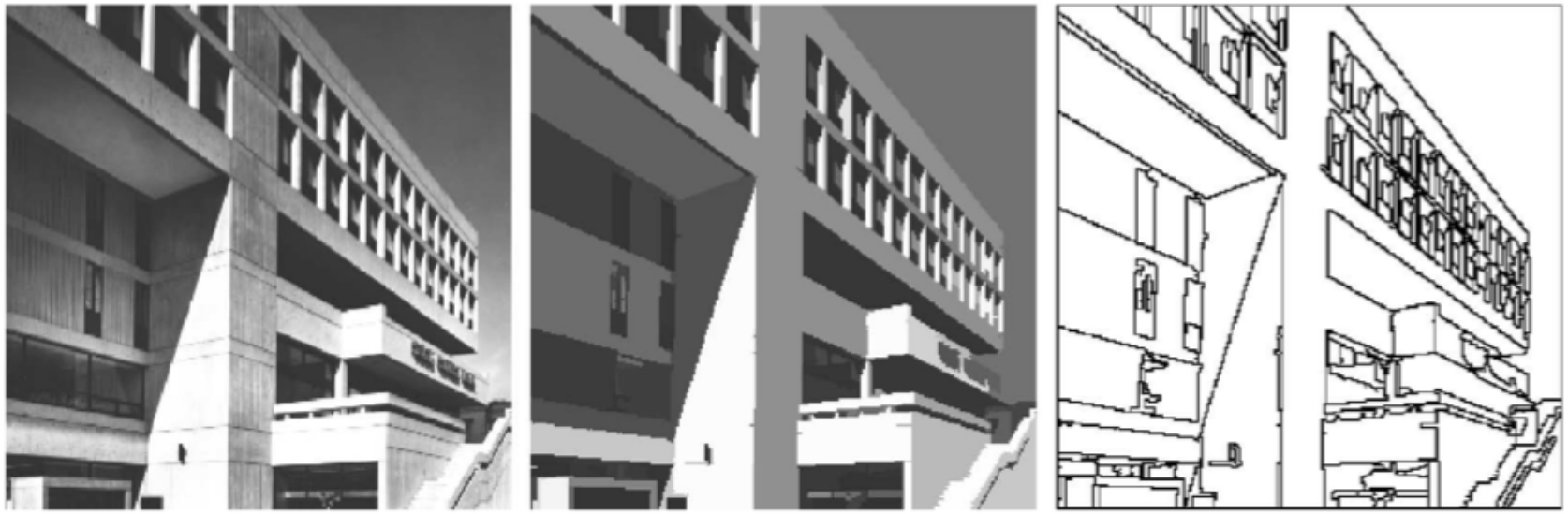


Fig. 6. *MIT* image. (a) Original. (b) Segmented $(h_s, h_r, M) = (8, 7, 20)$. (c) Region boundaries.

(from Meer & Comaniciu, PAMI 2002)

mean-shift segmentation results

Fig. 7. *Room* image. (a) Original. (b) Region boundaries delineated with $(h_s, h_r, M) = (8, 5, 20)$, drawn over the input.

(from Meer & Comaniciu, PAMI 2002)

mean-shift segmentation results

Fig. 8. *Lake* image. (a) Original. (b) Segmented with $(h_s, h_r, M) = (16, 7, 40)$.

(from Meer & Comaniciu, PAMI 2002)

mean-shift segmentation results

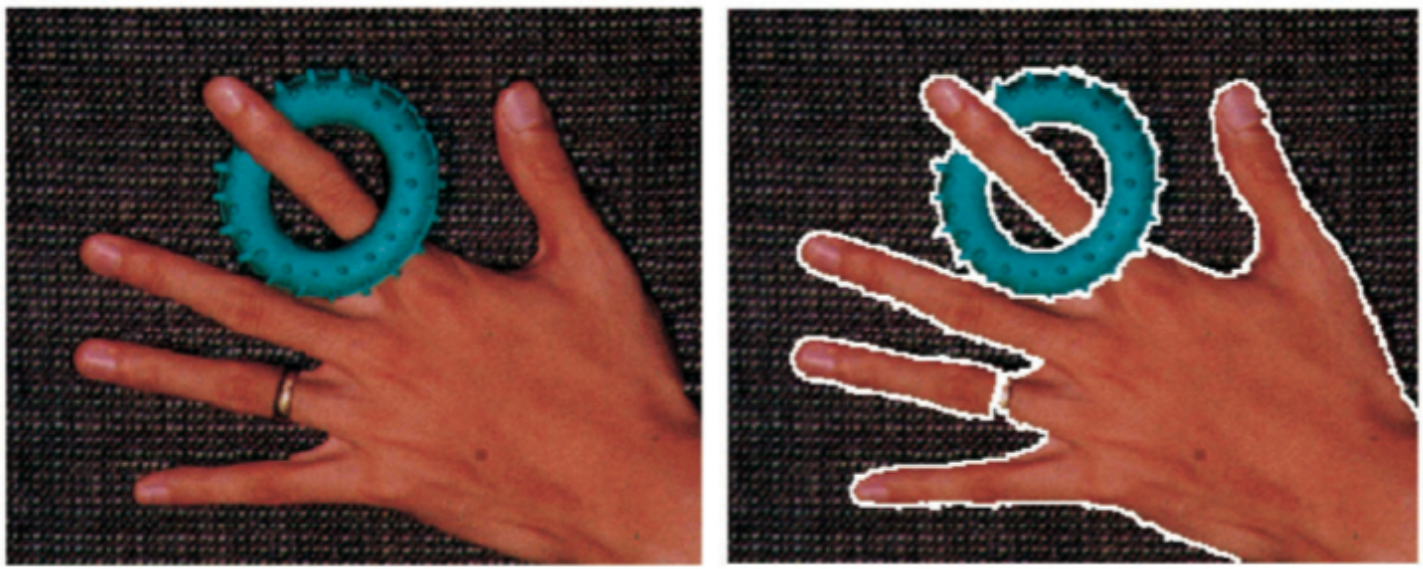


Fig. 9. *Hand* image. (a) Original. (b) Region boundaries delineated with $(h_s, h_r, M) = (16, 19, 40)$ drawn over the input

(from Meer & Comaniciu, PAMI 2002)

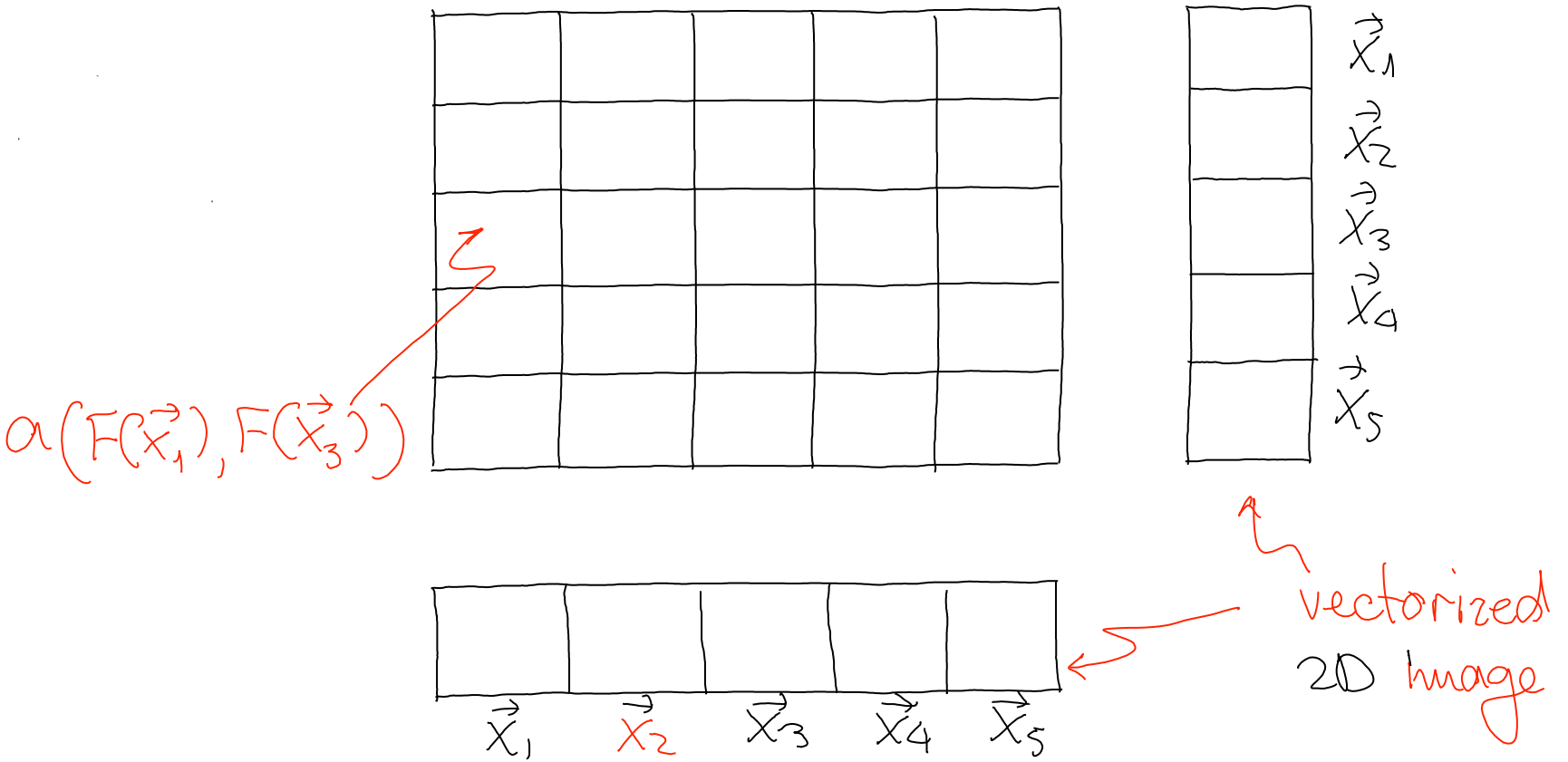
Topic 15:

Image Segmentation

- intro to segmentation
- the affinity matrix
- spectral methods
  - normalized cuts for hard segmentation
  - spectral matting for computing alpha mattes
- efficient graph-based methods
- density estimation methods
- a deeper look at affinity functions

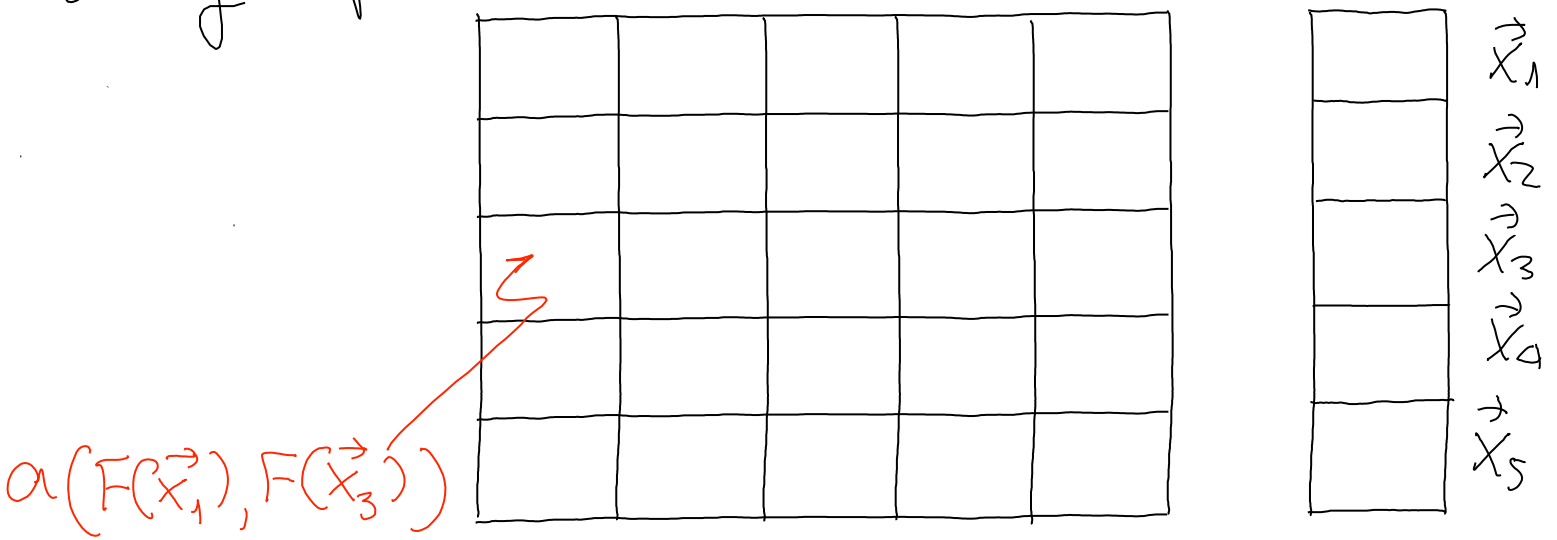
affinity function between pairs of pixels

So far we have taken a somewhat naive view of what an affinity function ought to be



affinity function examples

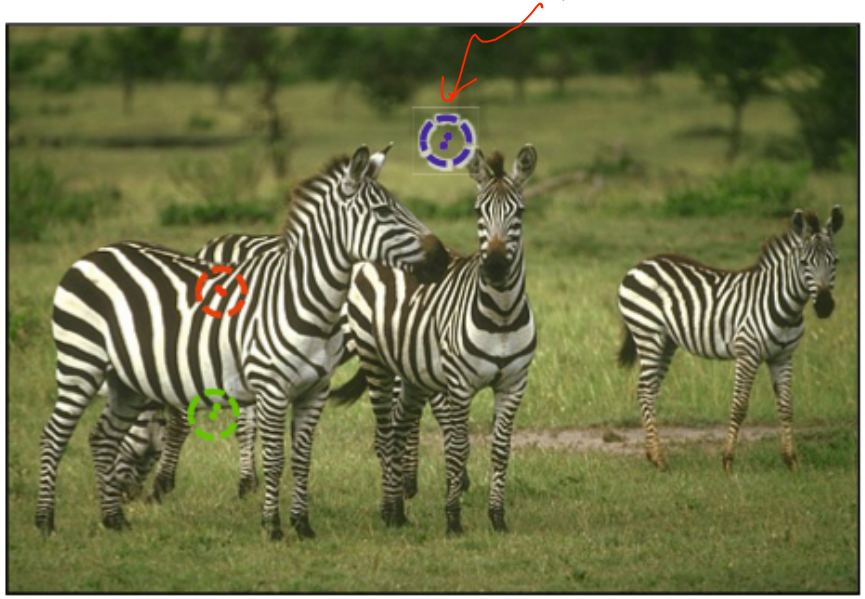
the examples below attempt to capture the fact that features that occur together should be grouped together

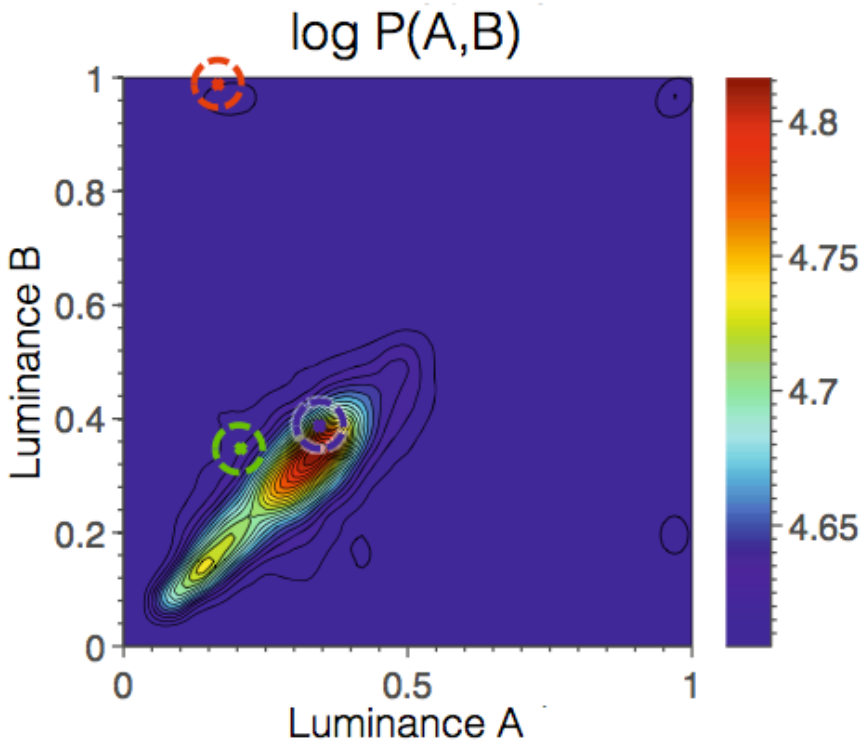


$$e^{\frac{-\|F_i - F_j\|^2}{\sigma^2}}, \quad e^{-(F_i - F_j)^T \Sigma^{-1} (F_i - F_j)}, \quad 1 - \|F_i - F_j\|^2, \ldots$$

probability of occurrence of features A and B

on same object, high P(A,B)





$$P(A,B) = \frac{1}{Z} \sum w(d)\, p(A,B;d)$$

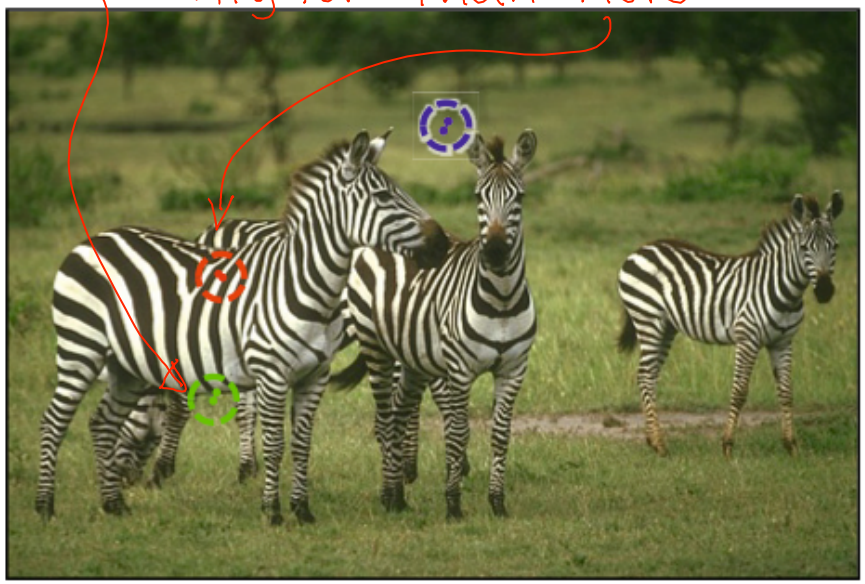
normalization constant

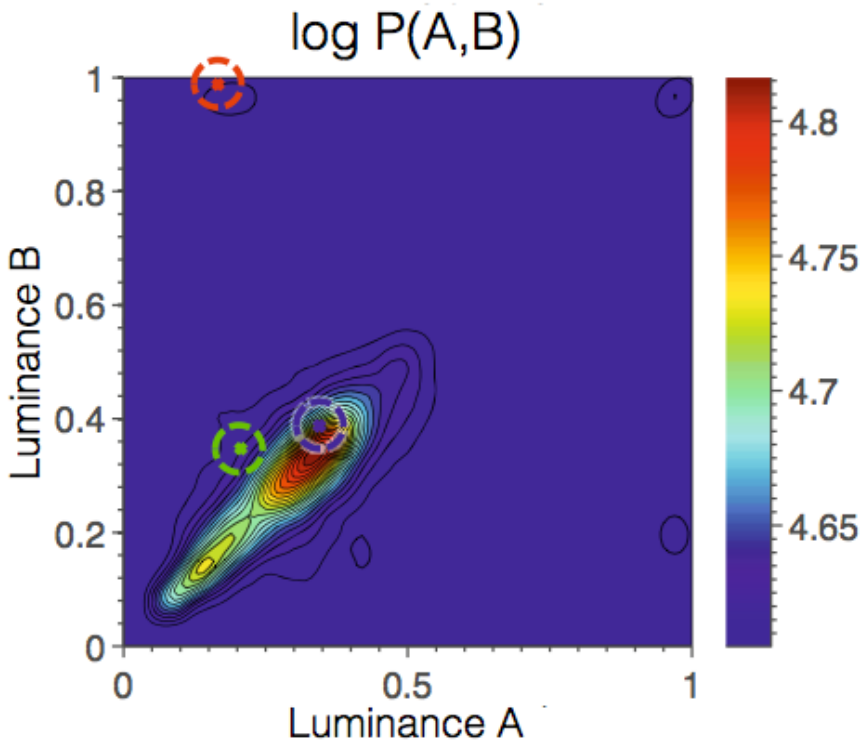
weight that diminishes with distance

probability of A, B occurring d pixels apart

probability of occurrence of features A and B

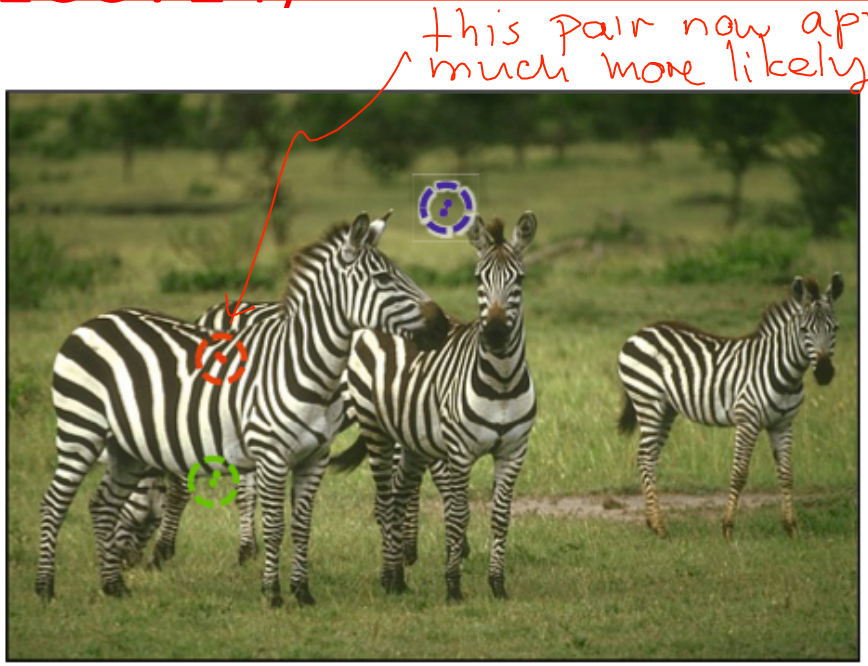
not on same object but P(A,B) higher than here

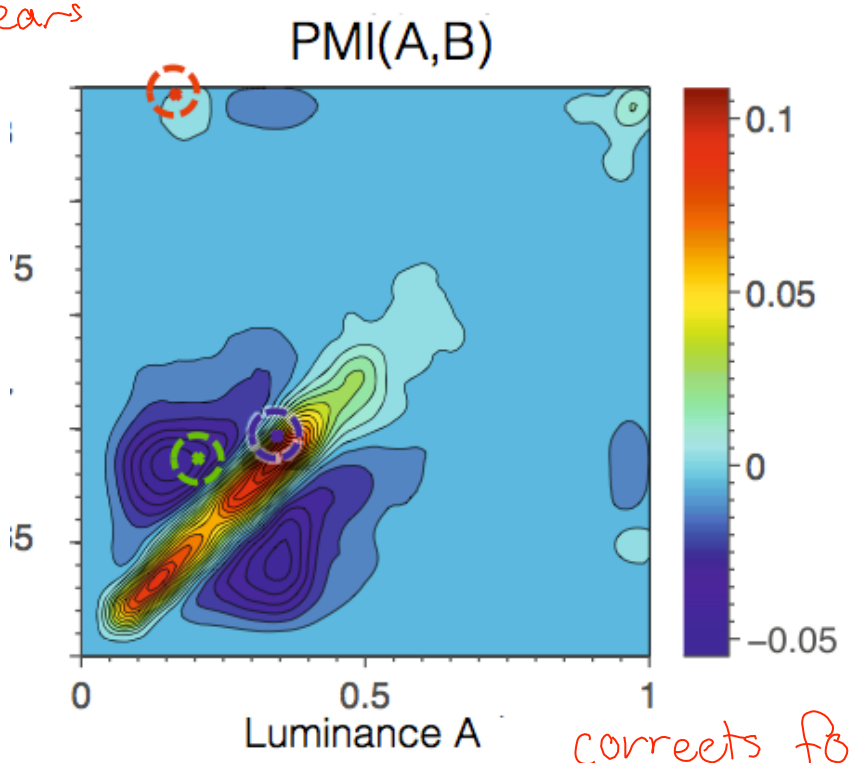




pixels that occur frequently in an image (eg. green) are much more likely to be next to any other color $\Rightarrow$ P(green, ·) inflated even across object boundaries

pointwise mutual information (Isola et al, ECCV14)



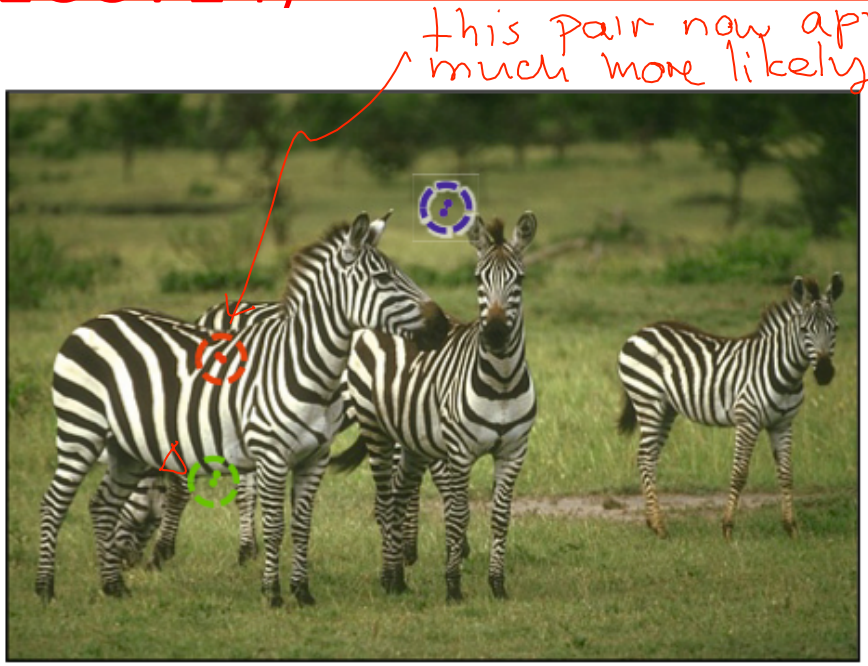


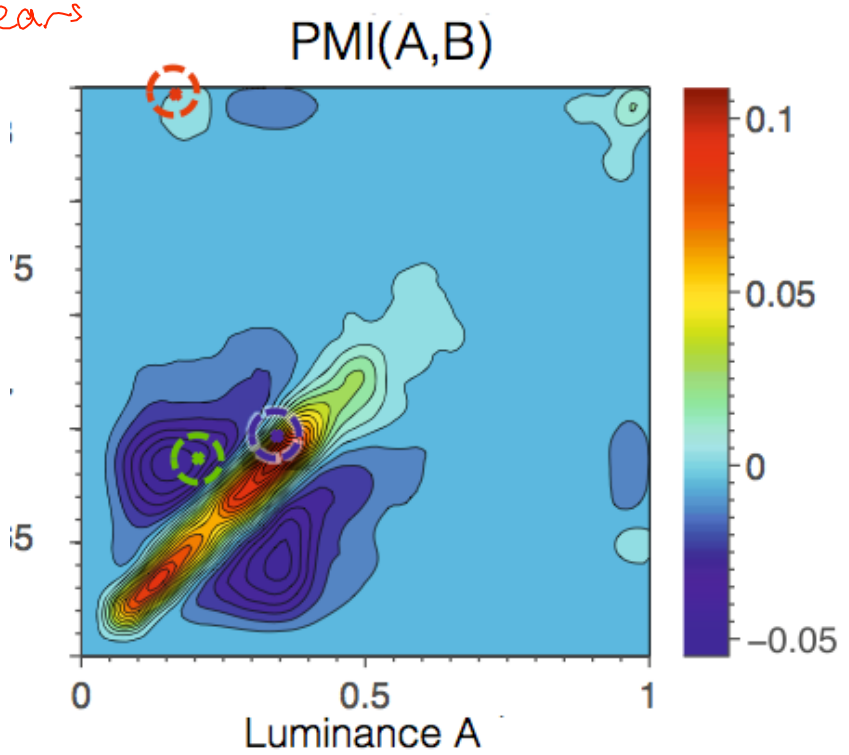
$$\left(\text{for } p \in [1,2]\right) \quad PMI_p(A,B) = \log \frac{P(A,B)^p}{P(A)\,P(B)}$$

corrects for baseline rarity of a feature

$$P(A) = \sum_B P(A,B) \qquad P(B) = \sum_A P(A,B)$$

pointwise mutual information (Isola et al, ECCV14)





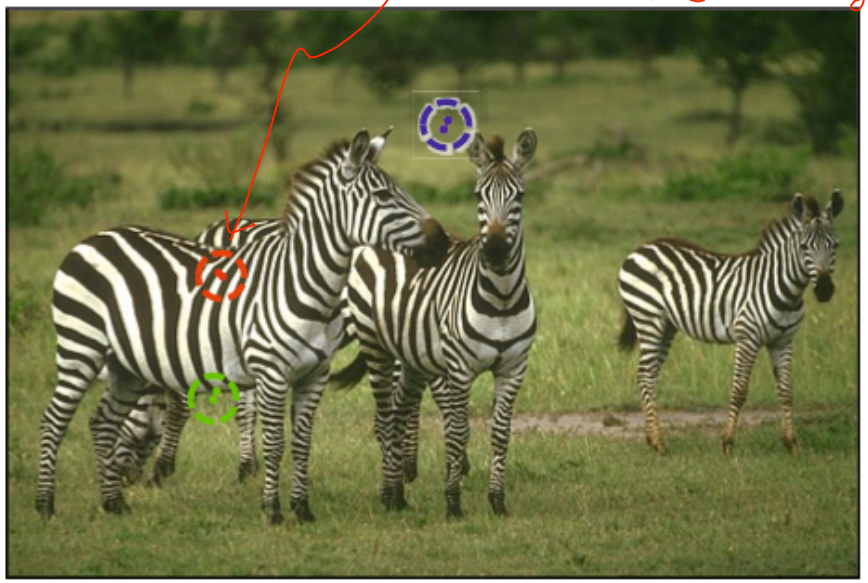
$$PMI_e(A,B) = \log \frac{P(A,B)^e}{P(A)P(B)}$$

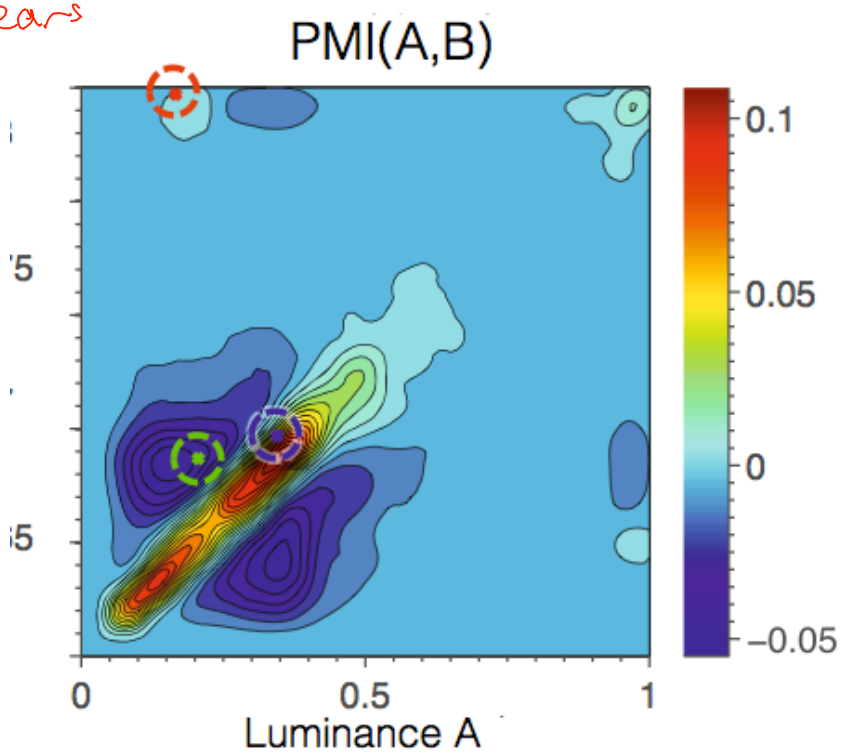
$$PMI_1(A,B) = \frac{P(A|B)}{P(A)}$$

measures how much more likely it is for A to occur next to B than to other feats

pointwise mutual information (Isola et al, ECCV14)

this pair now appears much more likely



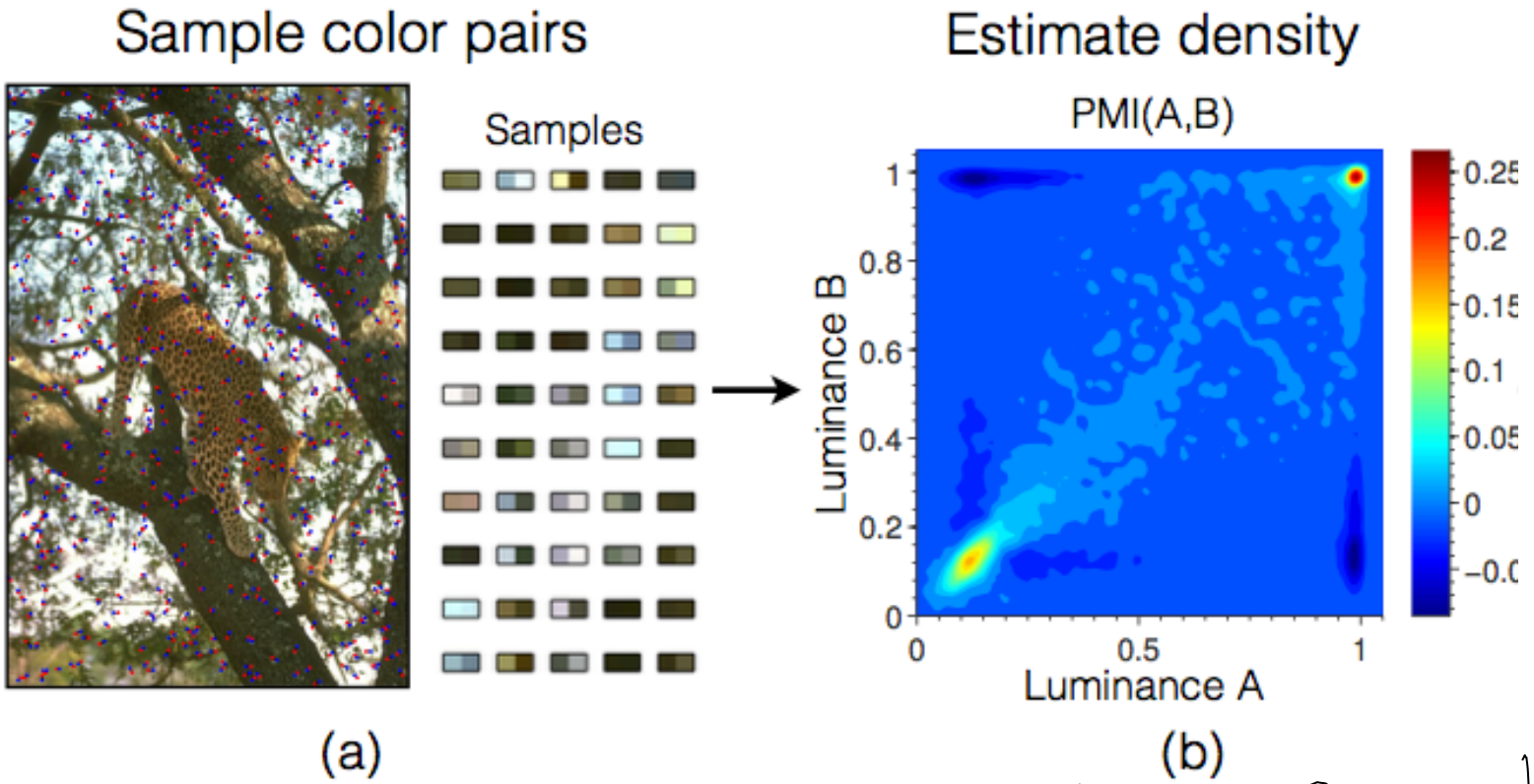


$$PMI_e(A,B) = \log \frac{P(A,B)^e}{P(A)P(B)}$$

observing A implies B is nearby and vice versa

$$PMI_2(A,B) = P(A|B)P(B|A)$$

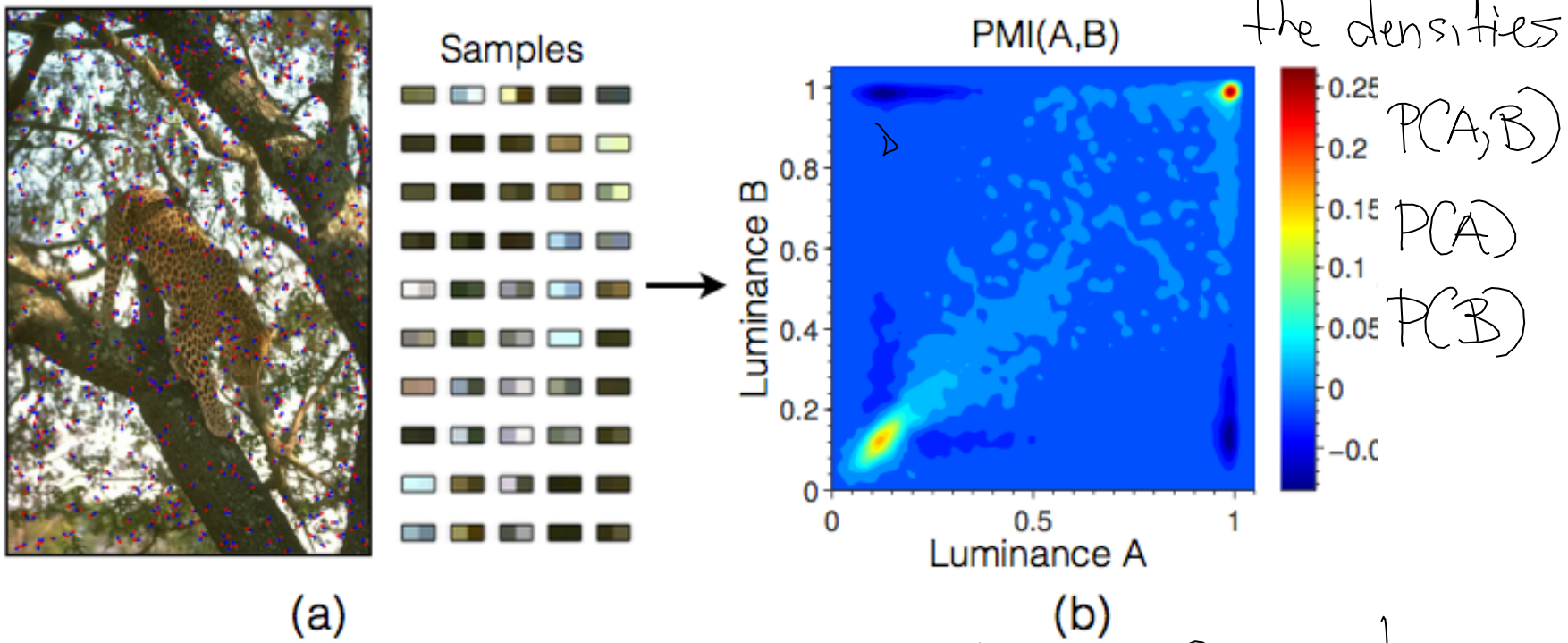
estimating P(A,B) by kernel density estimation



(1.) Uniformly draw 10000 positions for 1st pixel in pair, to get A

(2) Draw a 2nd sample at distance d with monotonically decreasing likelihood w(d).

estimating P(A,B) by kernel density estimation



(3) Epanechnikov kernels used to estimate the densities

P(A,B)

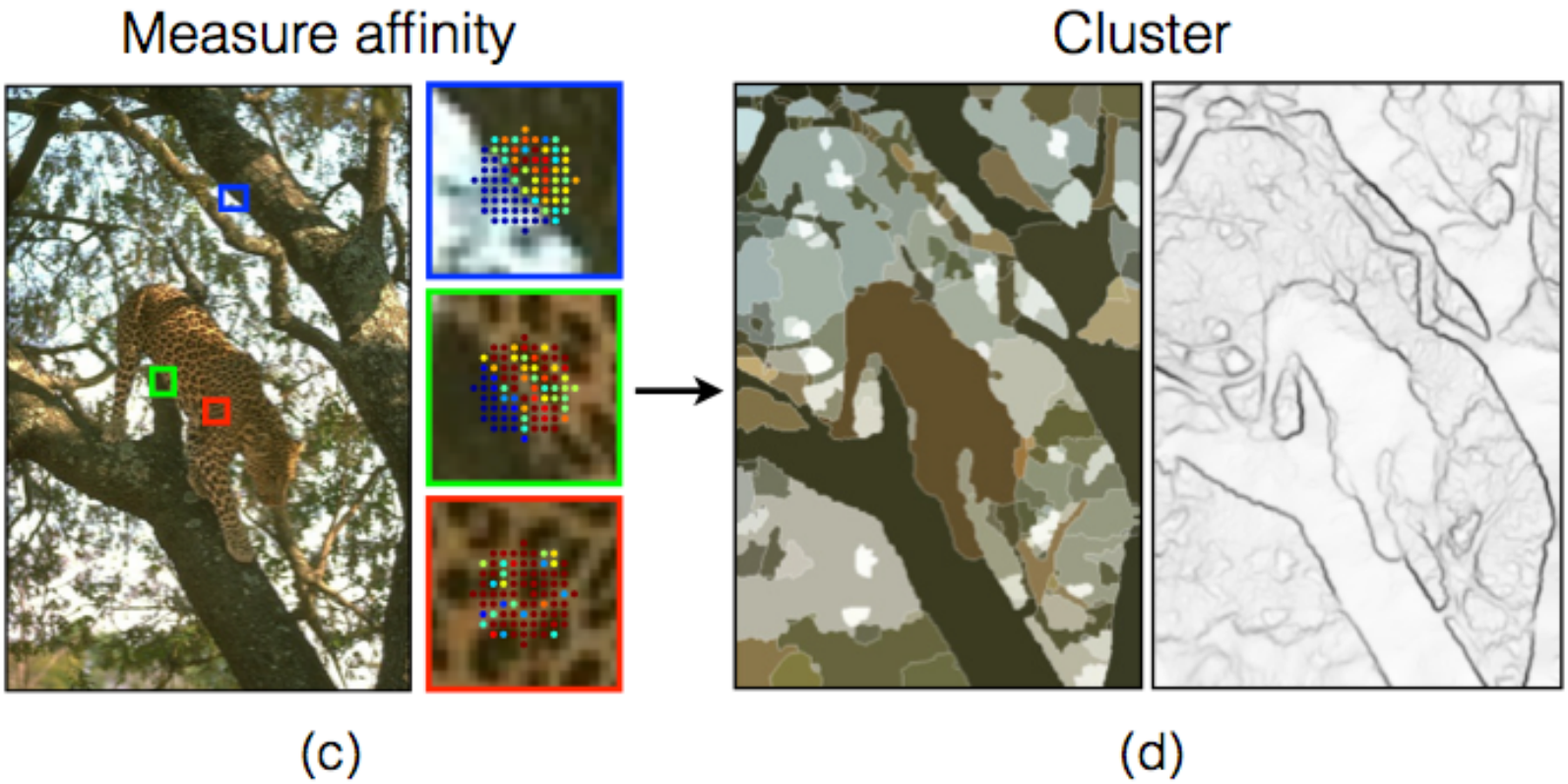
P(A)

P(B)

(1) Uniformly draw 10000 positions for 1st pixel in pair, to get A

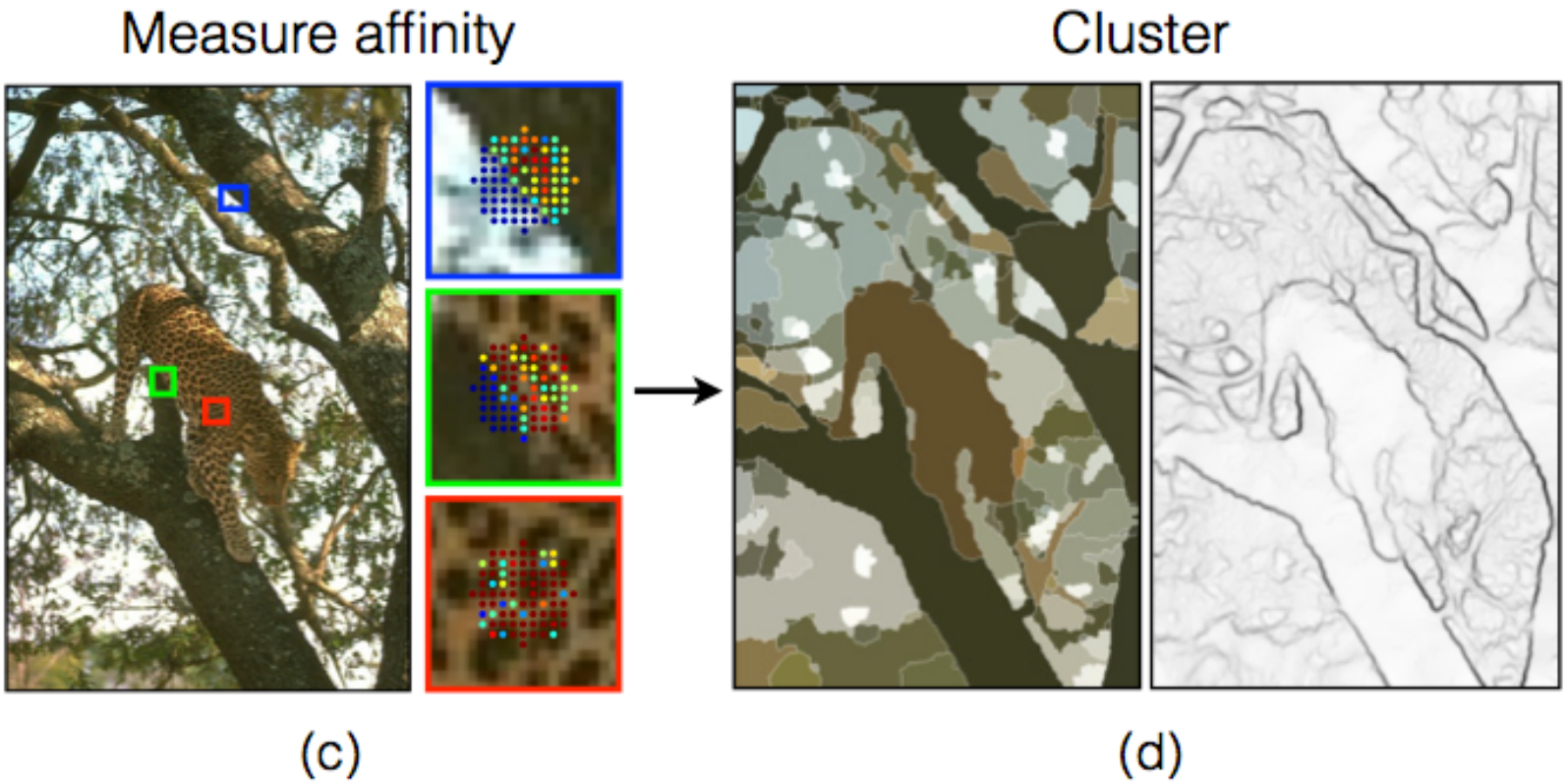
(2) Draw a 2nd sample at distance d with monotonically decreasing likelihood w(d).

from affinities to segments & boundaries (step 1)



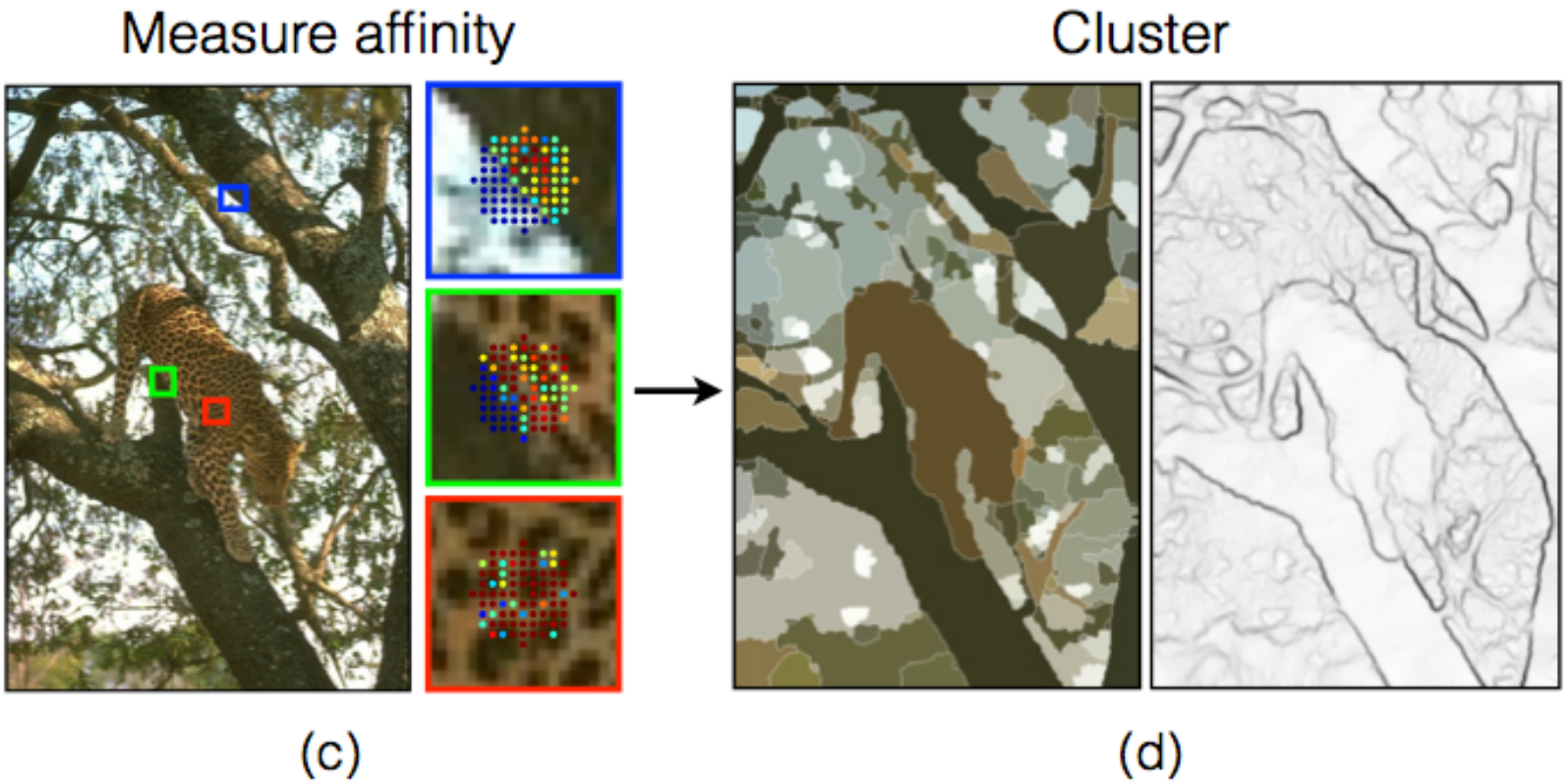
Compute the 100 lowest generalized eigenvectors from the normalized cuts procedure

from affinities to segments & boundaries (step 2)



- Apply a derivative filter along 8 directions to each of these eigenvectors
- Choose highest response at each pixel

from affinities to segments & boundaries (step3)

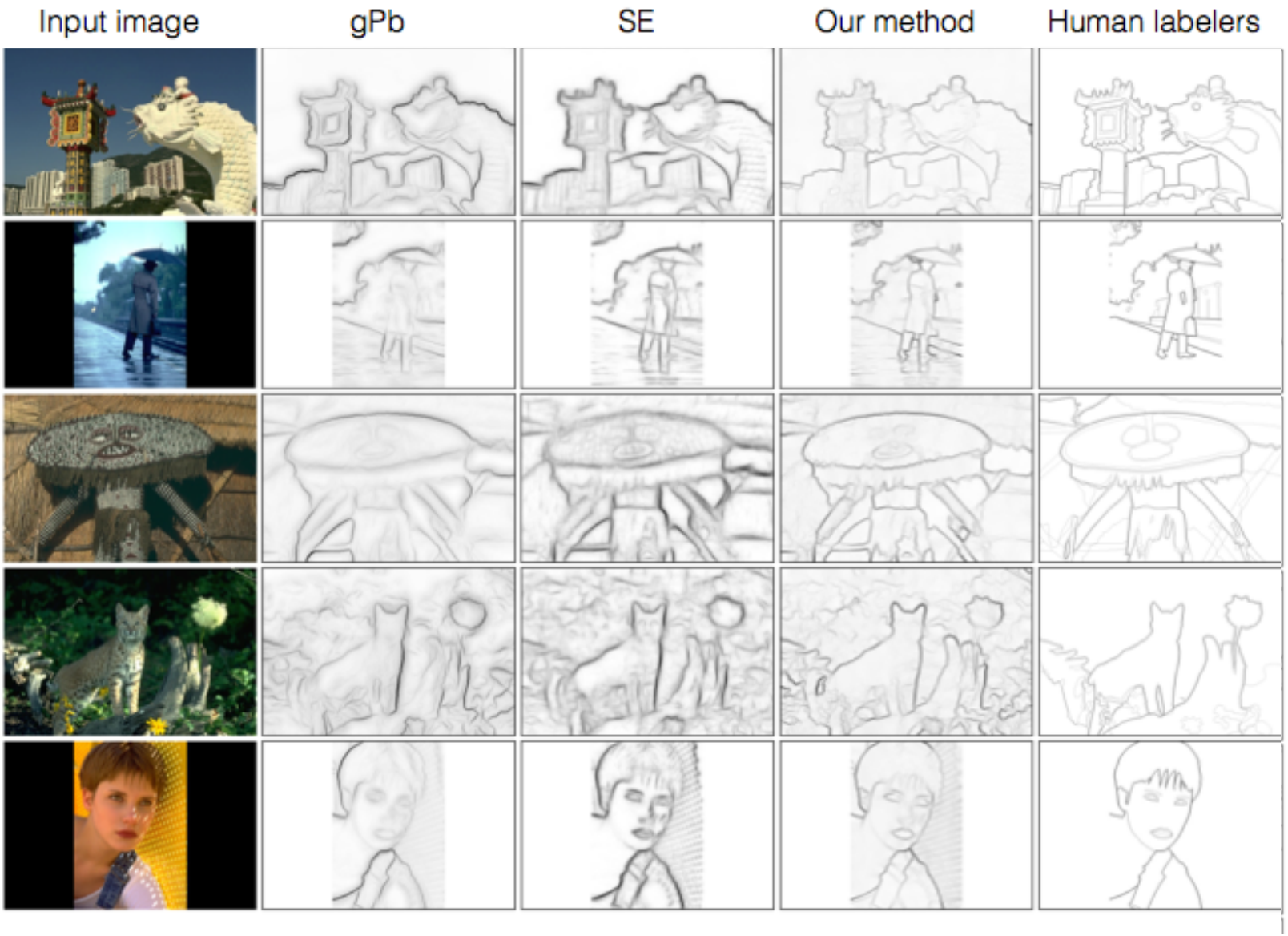


Postprocessing: (see Arbeláez, PAMI 2011)

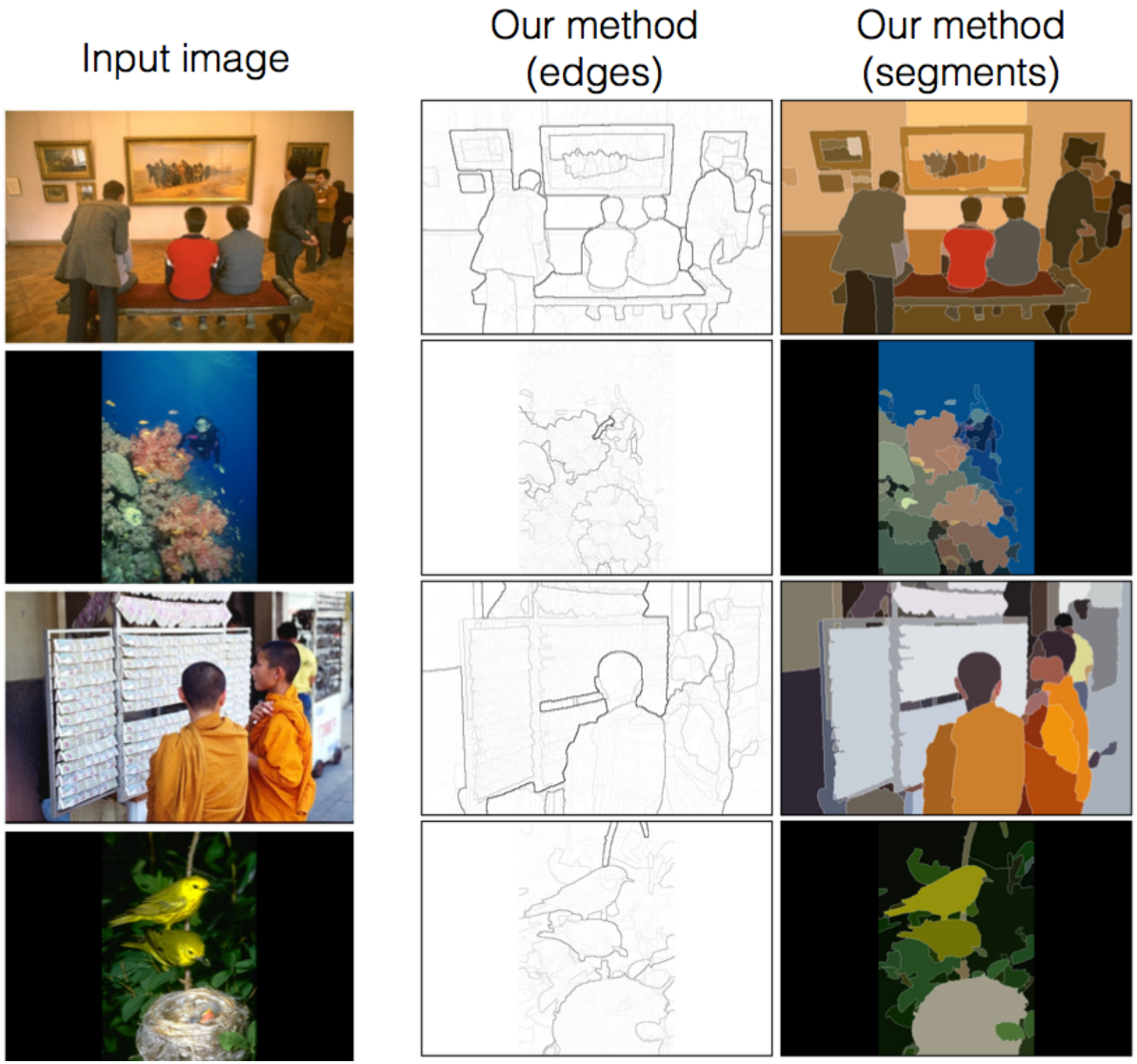
- Compute Oriented Watershed Transform
- Compute Ultrametric Contour Map

examples

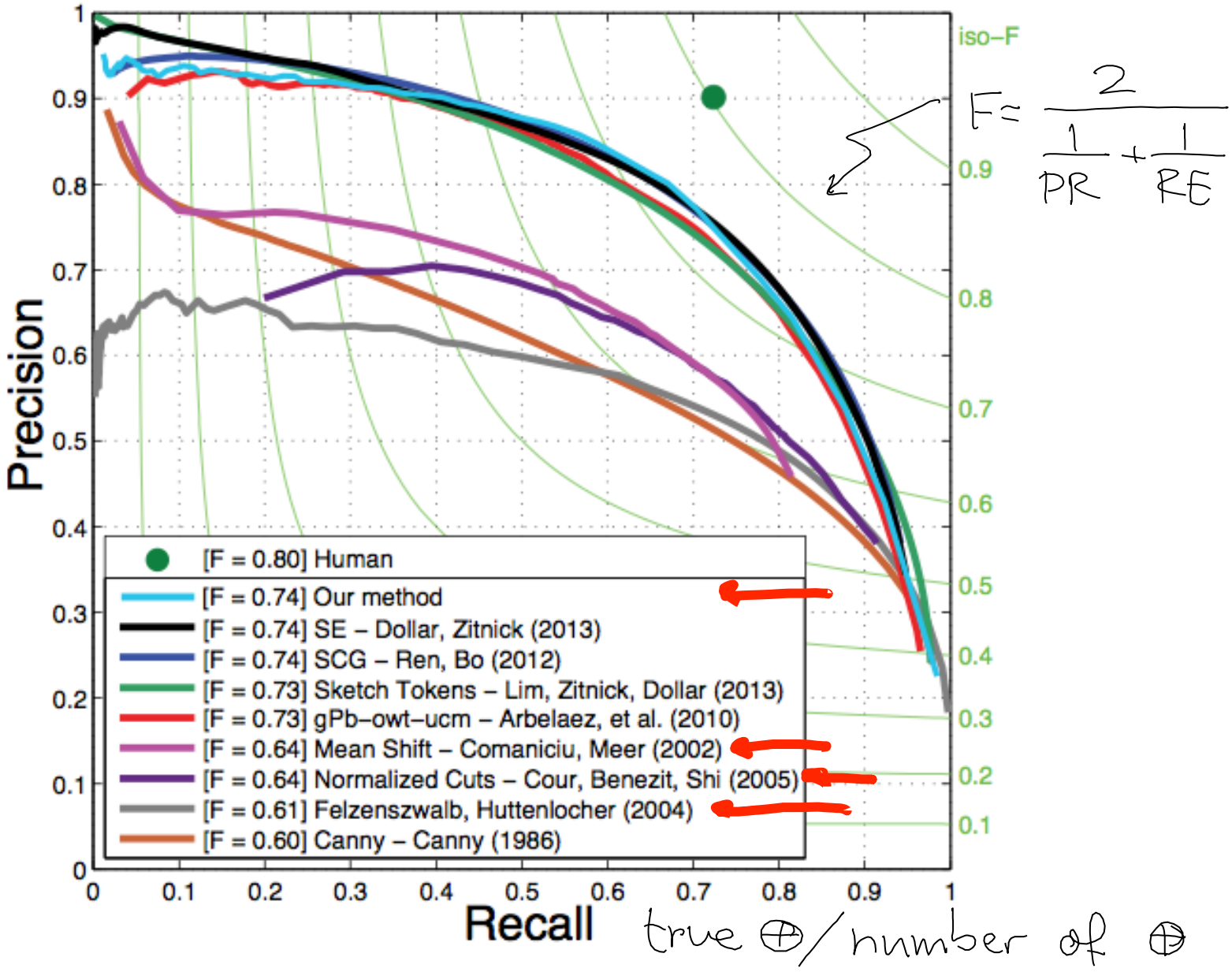
examples



examples



quantitative evaluation on BSDS500



$F = \frac{2}{\frac{1}{PR} + \frac{1}{RE}}$

$\frac{\text{true } \oplus}{\text{classified } \oplus}$

true ⊕ / number of ⊕

key references

- D. Martin, C. Fowlkes, D. Tal, J. Malik, A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics, Proc. ICCV 2001
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
- J. Shi, J. Malik, Normalized Cuts and Image Segmentation, IEEE PAMI, v.22, n.8, 2000
- A. Levin, D. Lischinski, Y. Weiss, A Closed-Form Solution to Natural Image Matting, IEEE PAMI, v.30, 2008
- A. Levin, A. Rav-Acha, D. Lischinski, Spectral Matting, IEEE PAMI, v. 30, 2008
- P. Felzenszwalb, D. Huttenlocher, Efficient Graph-Based Image Segmentation, IJCV, v.59, 2004
- D. Comaniciu, P. Meer, Mean Shift: A Robust Approach Toward Feature Space Analysis, IEEE PAMI, v.24, 2002
- P. Isola, D. Zoran, D. Krishnan, E. Adelson, Crisp Boundary Detection Using Pointwise Mutual Information, Proc. ECCV 2014, http://web.mit.edu/phillipi/pmi-boundaries/