CSE 252C: Computer Vision III

Lecturer: Serge Belongie
Scribes: Andrew Rabinovich and Vincent Rabaud
Edited by: Catherine Wah

LECTURE 8
Image Segmentation

I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees. It is impossible to achieve "327" as such. And yet even though such droll calculation were possible and implied, say, for the house 120, the trees 90, the sky 117 – I should at least have this arrangement and division of the total, and not, say, 127 and 100 and 100; or 150 and 177.

Max Wertheimer (1923)

8.1. Grouping

Grouping is still an open question in computer vision. It has been studied for some time, and early insight into grouping in the HVS was offered by the psychologists of the Gestalt movement at the beginning of the 20th century.

[1]Department of Computer Science and Engineering, University of California, San Diego.

September 20, 2009

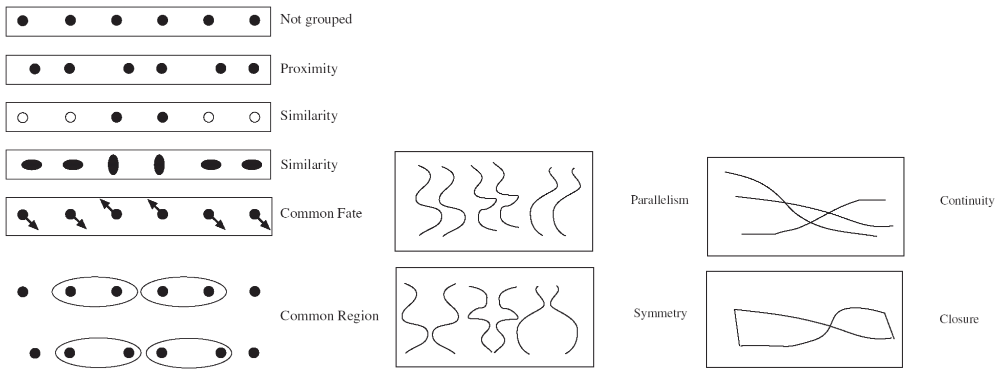

Figure 1. Gestalt's grouping factors.

This philosophical movement has put forward several *grouping factors* that describe how the human visual stimulus works, illustrated in Figure **??**. Most of the research activity in grouping today in the area of image segmentation is focused on the problem of operationalizing the so-called Gestalt factors of grouping:

- *Proximity*
- *Similarity*
- *Common Fate*
- *Common Region*
- *Parallelism*
- *Symmetry*
- *Continuity*
- *Closure*

The Gestalt school is known for the saying that "the whole is different than the sum of the parts." They used the term *gestalt qualität* to describe the set of internal relationships that makes the scene a whole. While the Gestalt school failed to provide any implementation details, they did make the point that we should exploit ecologically valid visual effects when attempting to organize a scene. Their ideas were left for several decades until implementations became possible on machines.

Remark. The anti-segmentation people say that people cannot come up with the same segmentation, *e.g.* BSE (Fowlkes et al., *CVPR*'03). In general, it is difficult to study segmentation in isolation, but having a database of human segmented images helps a lot. As Yair Weiss noted: "It is true that people cannot agree on what is a good segmentation, but people agree on what is a bad segmentation" (Figure **??**).

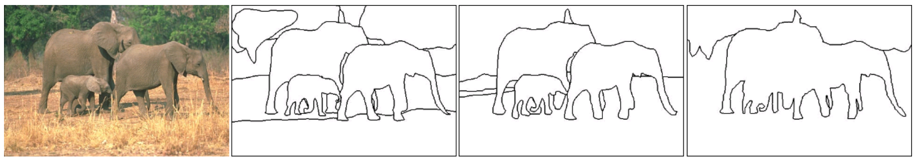

Figure 2. Several human segmentations of the same image (from Martin et al., *ICCV*'01).

8.2. Normalized-Cut

8.2.1. Introduction

Normalized-Cut (Shi & Malik, *CVPR*'97) is one means of operationalizing a Gestalt-inspired framework for image segmentation. We will focus on the "engine" and not as much on the cues; cue integration is difficult and as of yet unsolved.

Definition 8.1. In *Normalized-Cut*, the image is represented as a graph $G = (V, E)$ where the nodes are the pixels and the edge weights represent the "affinities" between pixels. Affinities (w_{ij}) are high (≈ 1) for very similar pixels, low (≈ 0) for very dissimilar pixels (Figure **??**); the similarity is based on *proximity* and *color*.

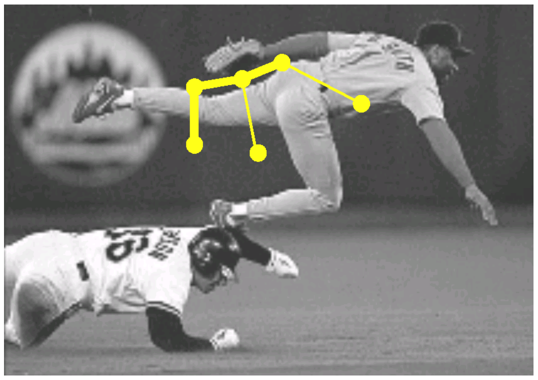

Figure 3. Illustration of the graph interpretation of an image. The thicker the edges, the greater the affinity between pixels (from Shi & Malik, *CVPR*'97).

8.2.2. Going from local to global

We store the affinities $w_{ij} \in [0, 1]$ in a matrix W – for example, $w_{ij} = e^{-(d_{ij}/.5)^2}$. This is the "local"; how do we find the "global"? One way is to recast the problem in terms of a random walk on the graph.

To make this work, we need to convert the affinities into probabilities, so that the total probability of hopping from one pixel to all the pixels it is connected to is equal to one. In order to achieve this, we divide each row of W by its sum, which forms a row-stochastic or "Markov" matrix, with nonnegative entries and rows summing to 1:

- Define the diagonal matrix D: $D_{ii} = \sum_j W_{ij}$
- Form the row-stochastic matrix $P = D^{-1}W$ ($0 \leq P_{ij} \leq 1$ and $\sum_j P_{ij} = 1$).

P represents the state transition matrix of a finite Markov chain. We will restrict our attention to "reversible" Markov chains, which is a kind of symmetry property on P, and we'll say more about them shortly.

8.2.3. Markov chains

To understand the "global" effects of the local grouping factors, we need to understand the long range or "steady-state" behavior of this Markov chain. Thus, we need to review some of the properties of Markov chains. The Markov property says that the next state of a system only depends on the present state and not the past history. Therefore, we need to know the state now, but not how it got there.

Let us consider the 2-state Markov chain in Figure **??**, where:

$$P = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}. \tag{8.2}$$

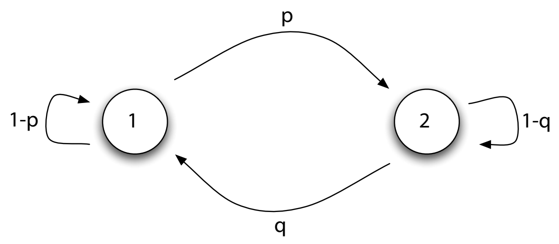

Figure 4. Simple Markov Chain

For example, state 1 means the phone is free, and state 2 means the phone is busy.

Suppose we have an initial probability distribution given by the row vector $\boldsymbol{\phi}_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$. Then to find the probability of finding it in each state at the next timestep, we "clock" the Markov chain once by computing $\boldsymbol{\phi}_0 P$, which is of course:

$$\boldsymbol{\phi}_1 = \boldsymbol{\phi}_0 P = \begin{bmatrix} 1-p & p \end{bmatrix}.$$

What about n timesteps later? Because of the Markov property, we can write:

$$\boldsymbol{\phi}_n = \boldsymbol{\phi}_{n-1} P = \boldsymbol{\phi}_0 P \cdots P = \boldsymbol{\phi}_0 P^n.$$

For example, assume the phone is free at time 0; we find the probability the phone is busy at time $n = 6$, where $p = \frac{1}{4}$, $q = \frac{1}{6}$:

$$P^6 = \begin{bmatrix} 3/4 & 1/4 \\ 1/6 & 5/6 \end{bmatrix}^6 = \begin{bmatrix} 0.424 & 0.576 \\ 0.384 & 0.616 \end{bmatrix} \text{ and } \boldsymbol{\phi}_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

then:

$$\boldsymbol{\phi}_0 P^6 = \begin{bmatrix} 0.424 & 0.576 \end{bmatrix},$$

where 0.576 is the probability the phone is busy at time 6.

To study the long term behavior, we consider the case where $n \to \infty$; for large n, P^n stabilizes at:

$$P^n = \begin{bmatrix} 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix}.$$

The *invariant probability distribution* is: $\boldsymbol{\pi} = \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$. If $\boldsymbol{\nu}$ is any probability vector, we have:

$$\lim_{n \to \infty} \boldsymbol{\nu} P^n = \boldsymbol{\pi}. \tag{8.3}$$

No matter what the initial state is, we end up $\boldsymbol{\pi}$. In particular, it is straightforward to show $\boldsymbol{\pi} = \boldsymbol{\pi} P$, which means that $\boldsymbol{\pi}$ is a left eigenvector of P with eigenvalue 1:

$$\begin{aligned} \text{if } \boldsymbol{\pi} &= \lim_{n \to \infty} \boldsymbol{\nu} P^n \text{ then} \\ \boldsymbol{\pi} &= \lim_{n \to \infty} \boldsymbol{\nu} P^{n+1} = (\lim_{n \to \infty} \boldsymbol{\nu} P^n) P = \boldsymbol{\pi} P \end{aligned}$$

It is also easy to check that the vector of all ones $\mathbf{1} = \begin{bmatrix} 1 & 1 \end{bmatrix}$ is a right eigenvector of P, with eigenvalues 1: $P\mathbf{1}^\top = \mathbf{1}^\top$. This just restates that all the rows of P sum to one.

One can prove that if P is a row stochastic matrix with $P_{ij} > 0$, then the Perron-Frobenius theorem says:

- 1 is a simple eigenvalue for P
- left eigenvectors can be chosen to have all positive entries (and hence can be made into a probability vector by multiplying by an appropriate constant)
- all other eigenvalues are < 1.

Finally, reversibility means that $\boldsymbol{\pi}_i P_{ij} = \boldsymbol{\pi}_j P_{ji}$. These "detailed balance equations" mean that we can run the system forwards or backwards in time, because in equilibrium the behavior is the same.

8.2.4. Power method

Consider the 2×2 example from before: we can diagonalize P as $P = Q\Lambda Q^{-1}$ where

$$Q = \begin{bmatrix} 1 & -p \\ 1 & q \end{bmatrix}, \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 1-p-q \end{bmatrix}, Q^{-1} = \frac{1}{p+q}\begin{bmatrix} q & p \\ -1 & 1 \end{bmatrix}$$

then

$$\begin{aligned} P^n &= \left(Q\Lambda Q^{-1}\right)^n \\ &= Q\Lambda^n Q^{-1} \\ &= Q\begin{bmatrix} 1 & 0 \\ 0 & (1-p-q)^n \end{bmatrix} Q^{-1} \simeq Q\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} Q^{-1} = \frac{1}{p+q}\begin{bmatrix} q & p \\ q & p \end{bmatrix} = \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\pi} \end{bmatrix}. \end{aligned}$$

In numerical linear algebra, this is known as the *power method* for finding eigenvectors and eigenvalues – note that when an eigenvalue is repeated, it doesn't work. We will assume the eigenvalue 1 is simple; if it's not, it means the graph contains two or more (trivially) disconnected components, each of which has the desired behavior with the simple eigenvalue of 1.

8.2.5. Segmentation

To connect this to the problem of segmentation, we need to consider the rate of convergence of the chain to its stationary distribution, known as the "time to equilibrium" or "mixing time." This rate of convergence is governed by the "gap" between the 1st and 2nd eigenvalues, since the larger the 2nd eigenvalue, the longer it takes for λ_2^n to get driven to zero. In the case of image segmentation, we are interested in the slowly mixing case, *i.e.* where λ_2 is close to λ_1, which means that there are sets of pixels that are hard to "leave."

We'd like to find these groups of pixels and segment them out. How do we do this? Shi & Malik (*CVPR*'97) proposed, in the two group case, to use the 2nd right eigenvector of P after thresholding it. (In the k-group case, one can either do this recursively or use multiple eigenvectors; we will revisit this situation later.) Recall that segmentation demands a hard decision; the 2nd eigenvector, which is necessarily orthogonal to $\mathbf{1}$, in general has a continuum of values, not just two. However, they observed that when clear partitions of the data are present, this second eigenvector is very nearly piecewise constant within each group. The nature of this approximation is studied in detail in spectral graph theory (*cf*. work by Fan Chung Graham).

For now, we will look at its empirical behavior. For the general case, how do we pick out the threshold? Shi & Malik (as well as Meila & Shi, 2001) address this problem by choosing the threshold version of $\boldsymbol{\nu}_2$ yielding the minimum "normalized cut" given a putative partition $V = A \cup B$, $A \cap B = \boldsymbol{\phi}$,

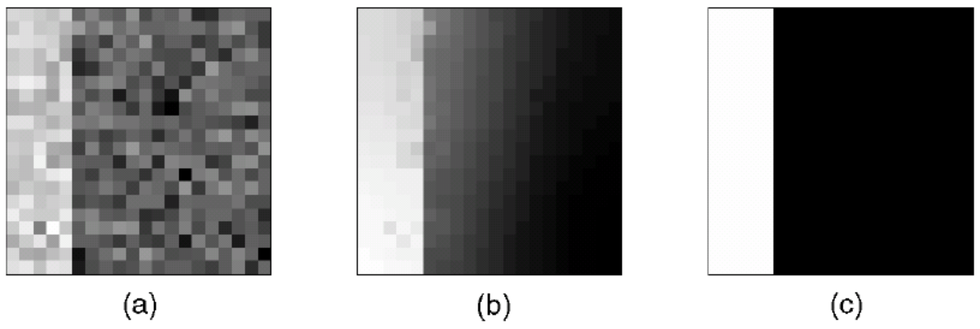

Figure 5. A result of normalized cut. (a) The original image (b) Updated values of the pixels taking affinities into account (c) The thresholded result.

$\text{NCut}(A, B) = P_{AB} + P_{BA}$, *i.e.* the probability of transitioning between A and B in one step (either $A \to B$ or $B \to A$), where $P_{AB} = \Pr[A \to B|A]$, is given by:

$$P_{AB} = \frac{\sum_{i \in A, j \in B} \boldsymbol{\pi}_i P_{ij}}{\sum_{i \in A} \boldsymbol{\pi}_i} = \frac{\sum_{i \in A, j \in B} W_{ij}}{\text{vol}(A)} = \frac{\text{cut}(A)}{\text{vol}(A)} \tag{8.4}$$

$$\Rightarrow \text{NCut}(A, B) = \text{cut}(A, B) \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right). \tag{8.5}$$

(Note $\text{vol}(A)$ is also given by $\text{vol}(A) = \sum_{i \in A, j \in V} W_{ij}$.)

Shi & Malik (with Papadimitriou) proved that exactly minimizing $\text{NCut}(A, B)$ over all choices of A and B is NP-complete, but they proposed using $\boldsymbol{\nu}_2$ as a starting point, then scanning over thresholds and picking the one with the lowest exact NCut value.

8.2.6. Spring-mass interpretation

Normalized-Cut can be studied from a different point of view by means of an analogy to a spring-mass system, which can offer some insight in cases of more than two groups. In the spring-mass setup, pixels are equivalent to point masses and affinities are represented by spring stiffnesses. To go from local to global, we pick up the resulting system and shake it, and the pixels in the different groups should "shake together."

8.2.7. k-group case

What are some ways of thresholding the eigenvectors to find the segments when there are more than two groups? As mentioned previously, recursive splitting based on the second eigenvector is one possibility. Another one is to take the first k eigenvectors and do clustering on them. Selecting k is not easy to do in general, but sometimes we can look for the gap. We can also do weighting on the eigenvectors; one option is "equipartition weighting," which says to weight the kth mode by $\frac{1}{w_k}$, but other alternatives have been proposed. Given these vectors, one can feed them to a standard clustering algorithm, *e.g.* k-means.

This reveals the "big secret" about eigenvector-based segmentation methods: they're not segmenting anything! You put in one segmentation problem and get out another. The benefit, of course, is that generally speaking, the second segmentation problem is easier. Thus, if you can get away with simple "central" clustering methods, go no further. In general, though, the eigenvectors obtained from NCut will yield a simpler clustering problem.