Normalized Cut Method for Image Segmentation

- J. Shi and J. Malik, *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(8), 1997
- Divisive (aka splitting, partitioning) method
- Graph-theoretic criterion for measuring goodness of an image partition
- Hierarchical partitioning
  - dendrogram type representation of all regions

- Criterion for measuring a candidate partitioning: Affinity measure between elements **within** each region is **high**, and the affinity **between** elements across regions is **low**
- Affinity: element × element $\rightarrow \Re^{+}$ Examples of components of an affinity function: spatial position, intensity, color, texture, motion. Defines the similarity of a pair of data elements.

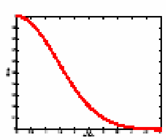
Affinity (Similarity) Measures

- Intensity

$$\mathrm{aff}\,(\mathbf{x},\mathbf{y}) = e^{-\left\| I(\mathbf{x}) - I(\mathbf{y}) \right\|^2 / 2\sigma_I^2}$$

- Distance

$$\mathrm{aff}(\mathbf{x},\mathbf{y}) = e^{-\left\| \mathbf{x} - \mathbf{y} \right\|^2 / 2\sigma_d^2}$$



- Color
- Texture
- Motion

Problem Formulation

- Given an undirected graph G = (*V*, *E*), where *V* is a set of nodes, one for each data element (e.g., pixel), and *E* is a set of edges with weights representing the affinity between connected nodes
- Find the image partition that maximizes the "association" within each region *and* minimizes the "disassociation" between regions
- Finding the optimal partition is NP-complete

- Let A, B partition G. Therefore, A ∪ B = V, and A ∩ B = ∅
- The **affinity** or **similarity** between A and B is defined as

$$cut(A,B) = \sum_{i\in A,\, j\in B} w_{ij}$$

= total weight of edges removed

- The **optimal bi-partition** of G is the one that minimizes *cut*
- *Cut* is biased towards small regions

- So, instead define the **normalized** similarity, called the *normalized-cut*(A,B), as

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

where $assoc(A,V) = \sum_{i\in A,\, k\in V} w_{ik}$

= total connection weight from nodes in A to all nodes in G

- *Ncut* measures the disimilarity between regions ("*disassociation*" measure)
- *Ncut* removes the bias based on region size (usually)

- Similarly, define the "**normalized association**:"

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

- *Nassoc* measures how similar, on average, nodes within the groups are to each other
- New goal: Find the bi-partition that minimizes *ncut*(A,B) **and** maximizes *nassoc*(A,B)
- But, it can be proved that *ncut*(A,B) = 2 – *nassoc*(A,B), so we can just **minimize *ncut***: *y* = arg min *ncut*

- Let **y** be a P = |V| dimensional vector where

$$y_i = \begin{cases} 1, & \text{if node } i \in A \\ -1, & \text{otherwise} \end{cases}$$

- Let $d(i) = \sum_j w_{ij}$

define the affinity of node *i* with all other nodes
- Let **D** = P x P diagonal matrix:

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ & & \dots & \\ 0 & 0 & \dots & d_P \end{bmatrix}$$

"degree matrix"

- Let **A** = P x P symmetric matrix:

"affinity matrix"

$$\mathbf{A} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1P} \\ w_{21} & w_{22} & \cdots & w_{2P} \\ & & \cdots & \\ w_{P1} & w_{P2} & \cdots & w_{PP} \end{bmatrix}$$

- It can be shown that

  y = arg $\min_x$ *ncut*(**x**)

$$= \arg\min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D}-\mathbf{A})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} \text{ subject to } \mathbf{y}^T\mathbf{D}\mathbf{1}=0$$

- Relaxing the constraint on **y** so as to allow it to have real values means that we can **approximate the solution** by solving an equation of the form: $(\mathbf{D} - \mathbf{A})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$

- The solution, **y**, is an eigenvector of (**D** – **A**)
- An eigenvector is a characteristic vector of a matrix and specifies a segmentation based on the values of its components; similar points will hopefully have similar eigenvector components.
- Theorem: If **M** is any real, symmetric matrix and **x** is orthogonal to the *j*-1 smallest eigenvectors $\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}$, then $\mathbf{x}^T\mathbf{M}\mathbf{x} / \mathbf{x}^T\mathbf{x}$ is minimized by the next smallest eigenvector $\mathbf{x}_j$ and its minimum value is the eigenvalue λ_j

---

- Smallest eigenvector is always 0

  because *A*=*V*, *B*={} means *ncut*(*A*,*B*)=0
- Second smallest eigenvector is the real-valued **y** that minimizes *ncut*
- Third smallest eigenvector is the real-valued **y** that optimally sub-partitions the first two regions
- Etc.
- Note: Converting from the real-valued **y** to a binary-valued **y** introduces errors that will propagate to each sub-partition

---

NCUT Segmentation Algorithm

1. Set up problem as G = (*V*,*E*) and define affinity matrix **A** and degree matrix **D**
2. Solve $(\mathbf{D} - \mathbf{A})\mathbf{x} = \lambda\mathbf{D}\mathbf{x}$ for the eigenvectors with the smallest eigenvalues
3. Let $\mathbf{x}_2$ = eigenvector with the 2nd smallest eigenvalue λ_2
4. Threshold $\mathbf{x}_2$ to obtain the binary-valued vector $\mathbf{x}'_2$ such that $ncut(\mathbf{x}'_2) \geq ncut(\mathbf{x}^t_2)$ for all possible thresholds *t*
5. For each of the two new regions, if *ncut* < threshold T, then recurse on the region

Comments on the Algorithm

- Recursively bi-partitions the graph instead of using the 3rd, 4th, etc. eigenvectors for robustness reasons (due to errors caused by the binarization of the real-valued eigenvectors)
- Solving standard eigenvalue problems takes $O(P^3)$ time
- Can speed up algorithm by exploiting the “locality” of affinity measures, which implies that **A** is sparse (non-zero values only near the diagonal) and (**D** – **A**) is sparse. This leads to a $O(P\sqrt{P})$ time algorithm

Example: 2D Point Set

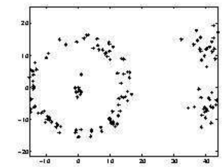
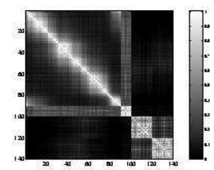


Figure 3: A point set in the plane.



Eigenvalues and Eigenvectors

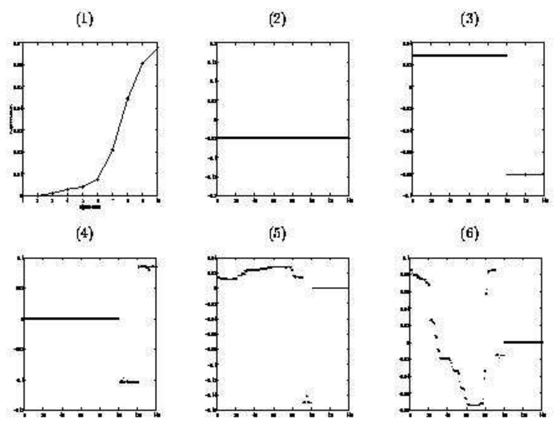


Figure 5: Subplot (1) plots the smallest 10 eigenvalues of the generalized eigenvalue system

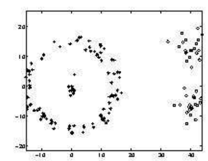


Figure 6: Partition of the point set using the eigenvector with the second smallest eigenvalue.

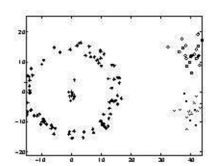


Figure 7: Sub-partitioning of the point sets using the eigenvectors with the third and fourth

image segmentation based on brightness and spatial features. Figure 8 shows an image that we would like to segment.

Example 2: A Grayscale Image

Figure 8: A gray level image of a baseball game.

Just as in the point set grouping case, we have the following steps for image segmentation:

1. Construct a weighted graph, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, by taking each pixel as a node, and connecting each pair of pixels by an edge. The weight on that edge should reflect the likelihood of the two pixels belong to one object. Using just the brightness value of the pixels and their spatial location, we can define the graph edge weight connecting two nodes i and j as:

$$w_{ij} = e^{\frac{-\|F_{(i)}-F_{(j)}\|_2^2}{\sigma_I}} * \begin{cases} e^{\frac{-\|X_{(i)}-X_{(j)}\|_2^2}{\sigma_X}} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

Figure 9 shows the weight matrix $\mathbf{W}$ associated with this weighted graph.

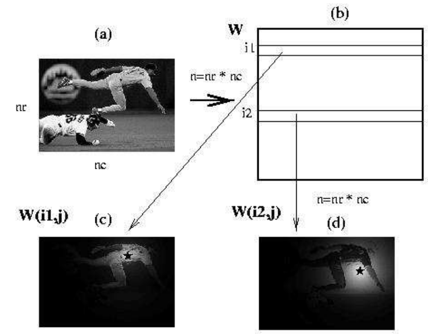


Figure 9: The similarity measure between each pair of pixels in (a) can be summarized in a $n \times n$ weight matrix $\mathbf{W}$, shown in (b), where n is the number of pixels in the image. Instead of displaying $\mathbf{W}$ itself, which is very large, two particular rows, i_1 and i_2 of $\mathbf{W}$ are shown in (c) and (d). Each of the rows, is the connection weights from a pixel to all other pixels in the image. The two rows, i_1 and i_2 are reshaped into the size of the image, and displayed. The brightness value in (c) and (d) reflects the connection weights. Note that $\mathbf{W}$ contains large number of zeros or near zeros, due to the spatial proximity factor.

Eigenvalues and Eigenvectors

Putting everything together, each of the matrix-vector computations cost $O(n)$ operations with a small constant factor. The number m depends on many factors[11]. In our experiments on image segmentation, we observed that m is typically less than $O(n^{\frac{1}{2}})$.

Figure 12 shows the smallest eigenvectors computed for the generalized eigensystem with the weight matrix defined above.

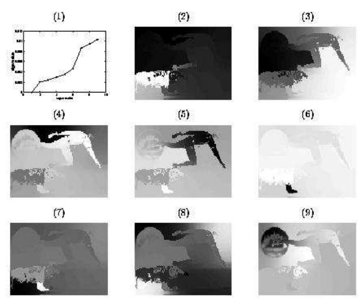


Figure 12: Subplot (1) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplot (2) - (9) shows the eigenvectors corresponding the 2nd smallest to the 9th smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

Discretizing an Eigenvector

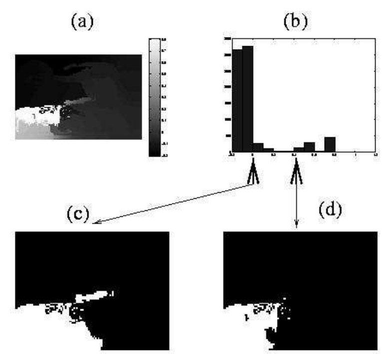


Figure 13: The eigenvector in (a) is a close approximation to a discrete partitioning indicator vector. Its histogram, shown in (b), indicates that the values in the eigenvector cluster around two extreme values. (c) and (d) shows the partitioning results with different splitting points indicated by the arrows in (b). The partition with the best normalized cut value is chosen.

Partitioning stops when histogram is not bimodal

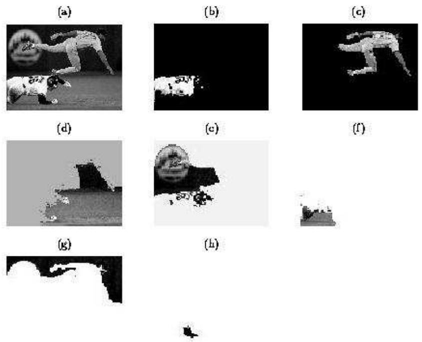


Figure 14: (a) shows the original image of size 80×100. Image intensity is normalized to lie within 0 and 1. Subplot (b) - (h) shows the components of the partition with *Ncut* value less than 0.04. Parameter setting: $\sigma_I = 0.1$, $\sigma_X = 10.0$, $r = 10$.

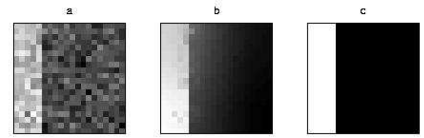


Figure 16: A synthetic image showing a noisy "step" image. Intensity varies from 0 to 1, and Gaussian noise with $\sigma = 0.2$ is added. Subplot (b) shows the eigenvector with the second smallest eigenvalue, and subplot (c) shows the resulting partition.

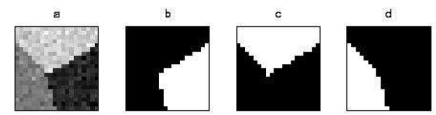


Figure 17: (a) A synthetic image showing three image patches forming a junction. Image intensity varies from 0 to 1, and Gaussian noise with $\sigma = 0.1$ is added. (b)-(d) shows the top three components of the partition.

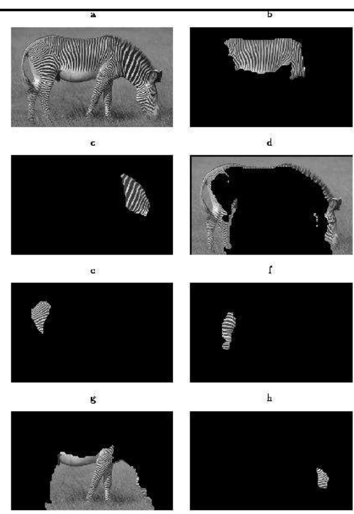


Figure 21: (a) shows an image of a zebra. The remaining images show the major com-

Some Example Results