Math 5365:Computer Literacy & Programming; *Mathematical Foundations of Data Science with Applications on the MAPLE and MATLAB Platforms*

Graph Partition/ Spectral Clustering

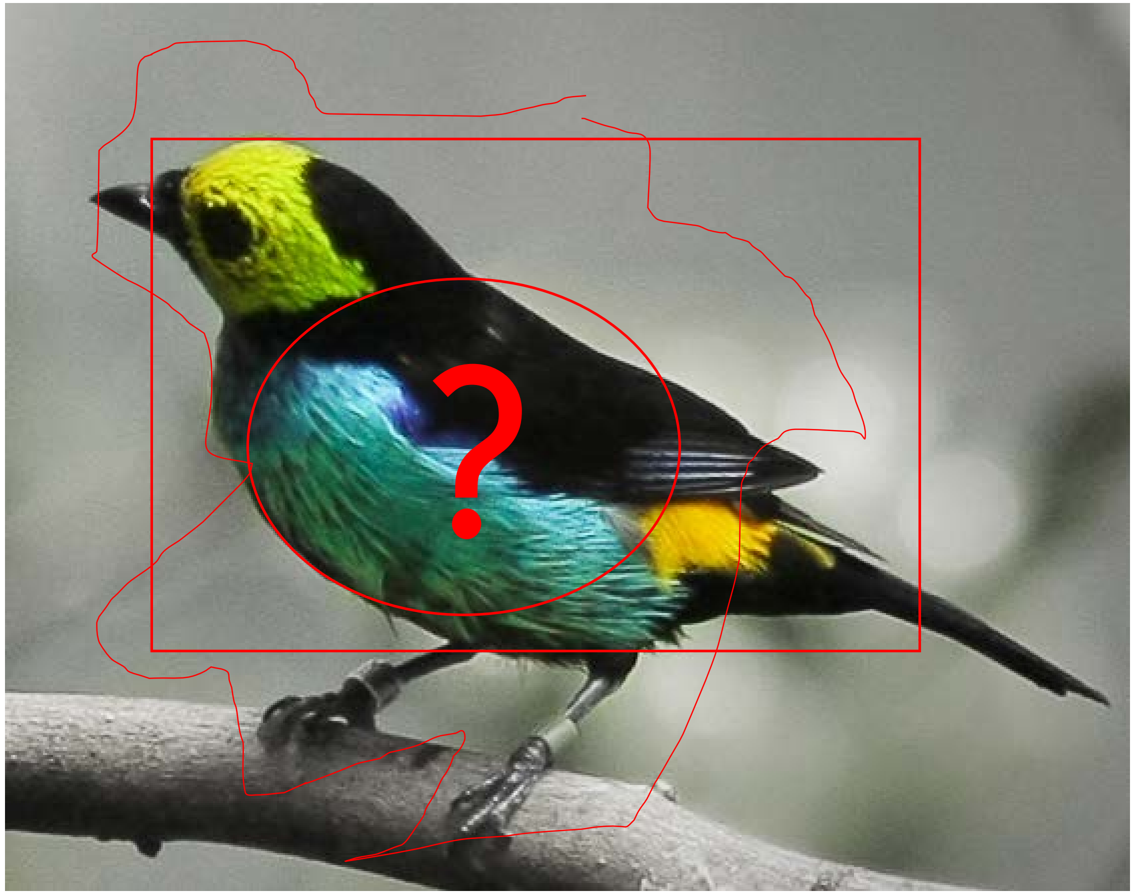
Graph Cut Background

Graph Cut Background

- First: select a region of interest

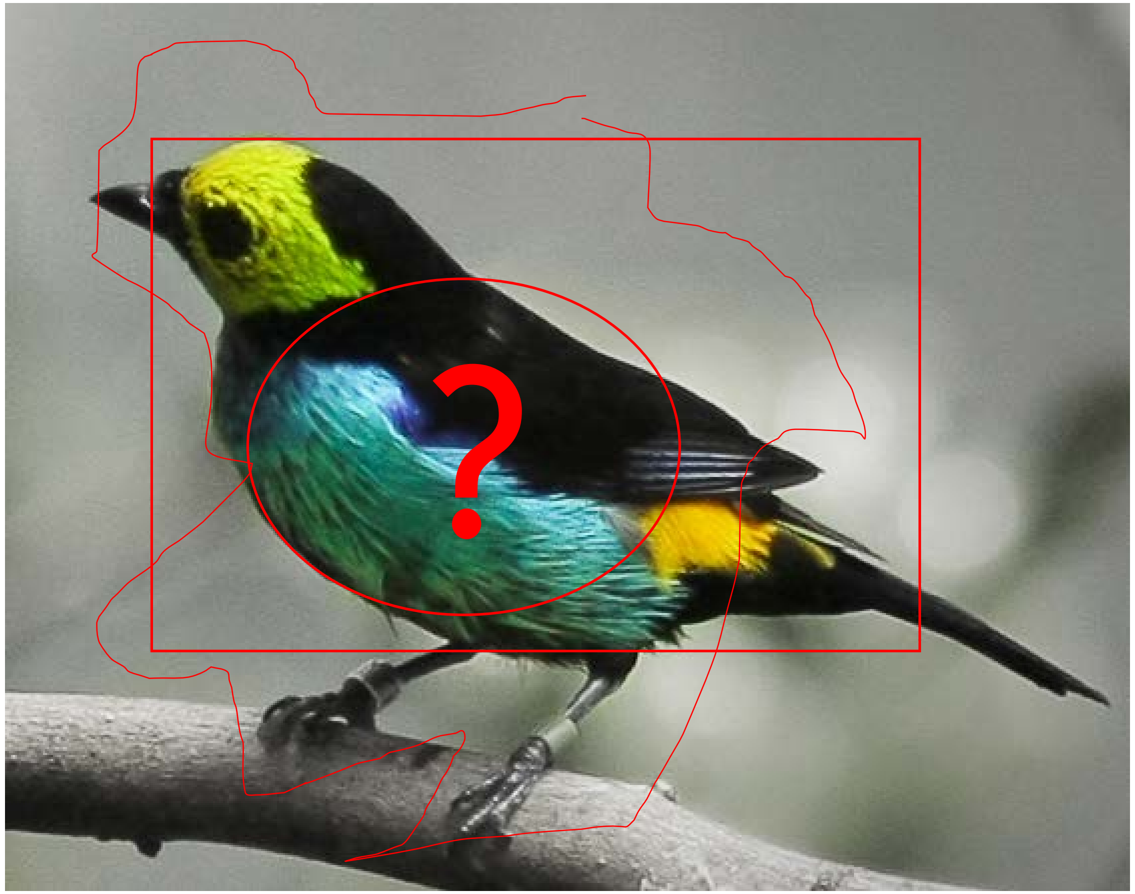
Graph Cut Background

- How to select the object automatically?



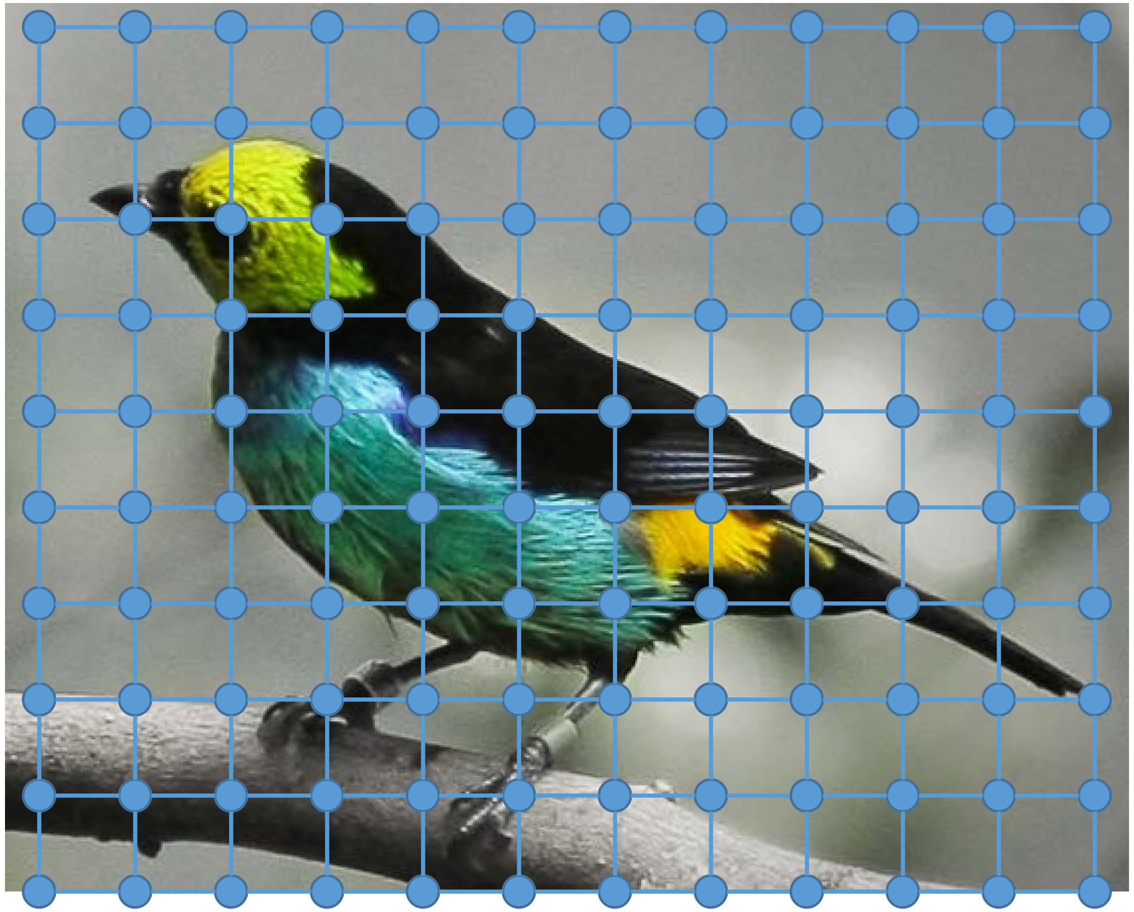
Graph Cut Background

- We care about two terms: graph and cuts



Graph Cut Background

- What are graphs?

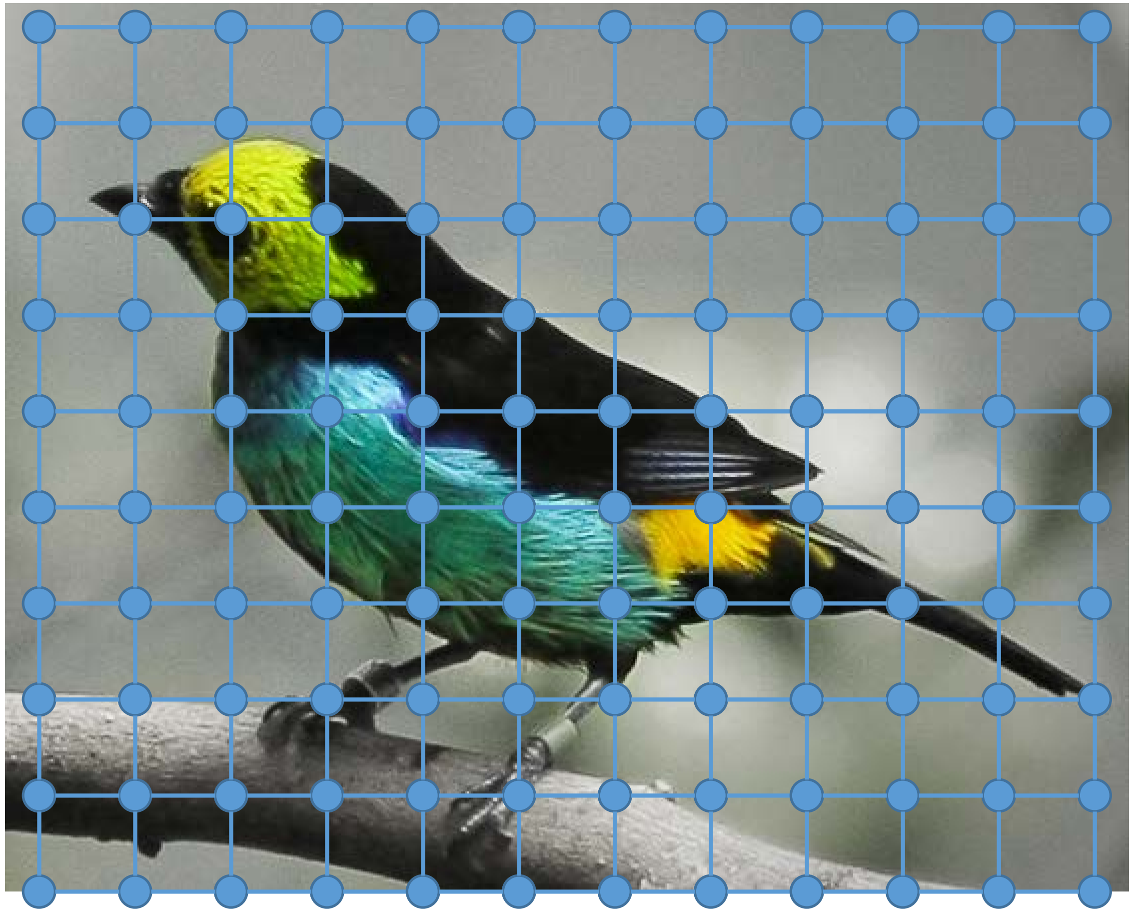


Nodes
- usually pixels
- sometimes samples

Edges
- weights associated ($W(i,j)$)
- E.g. RGB value difference

Graph Cut Background

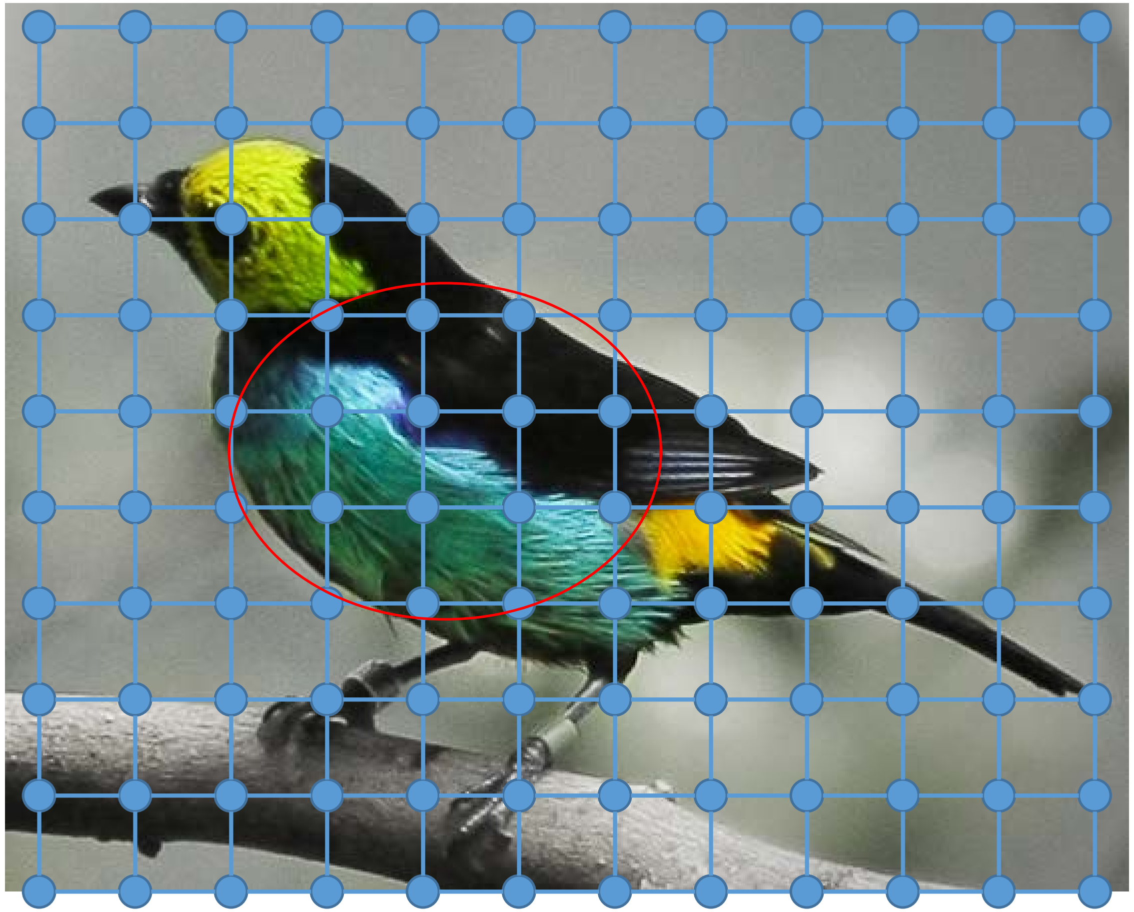
- What are cuts?



- Each “cut” -> points, $W(i,j)$
- Optimization problem
- $W(i,j) = |\mathrm{RGB}(i) - \mathrm{RGB}(j)|$

Graph Cut Background

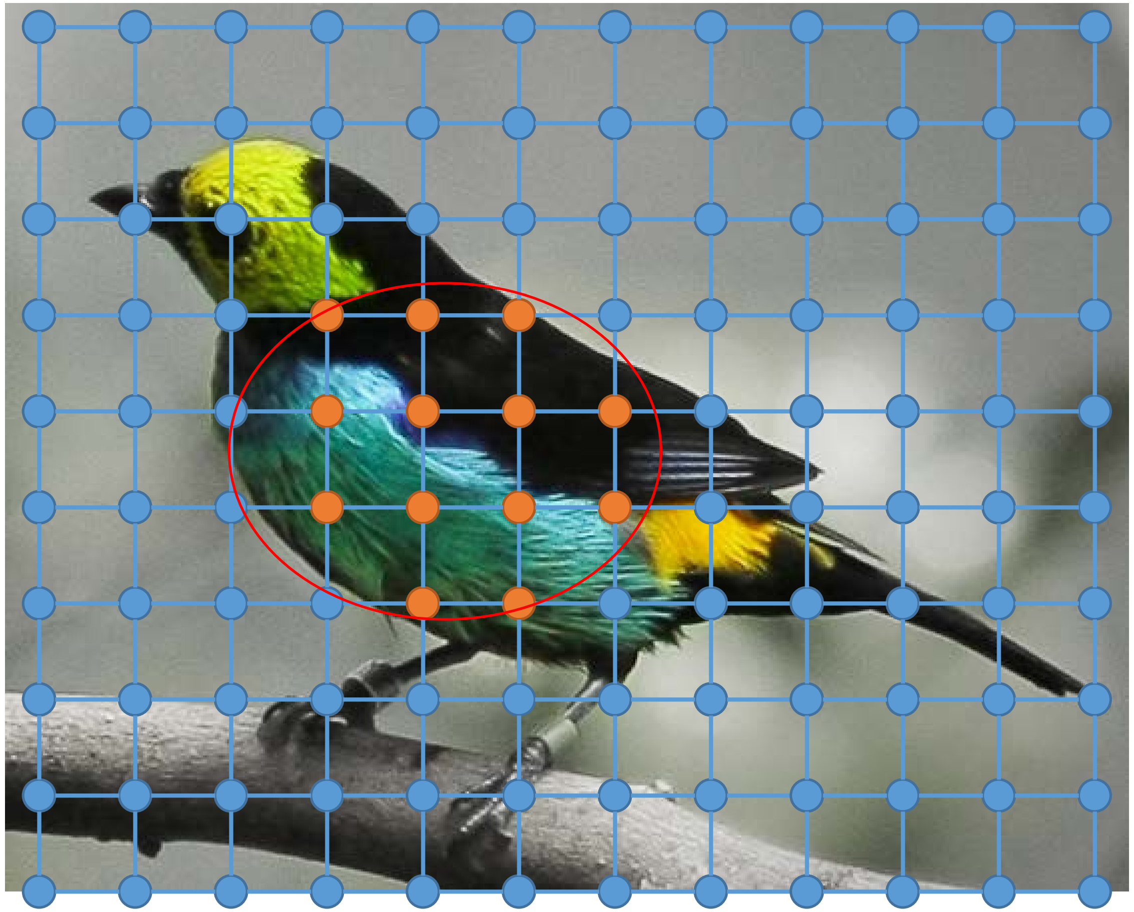
- Go back to our selected region



- Each “cut” -> points, $W(i,j)$
- Optimization problem
- $W(i,j) = |\mathrm{RGB}(i) - \mathrm{RGB}(j)|$

Graph Cut Background

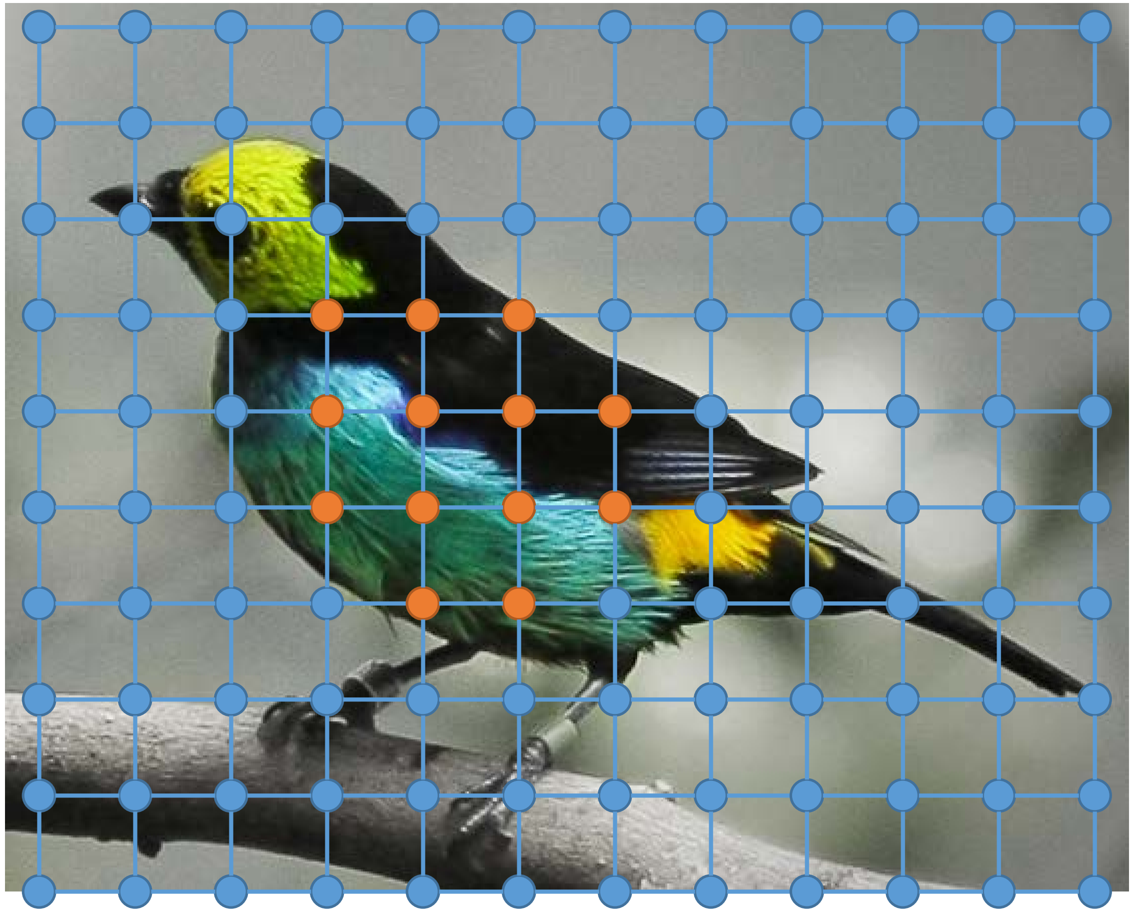
- Go back to our selected region



- Each “cut” -> points, $W(i,j)$
- Optimization problem
- $W(i,j) = |\mathrm{RGB}(i) - \mathrm{RGB}(j)|$

Graph Cut Background

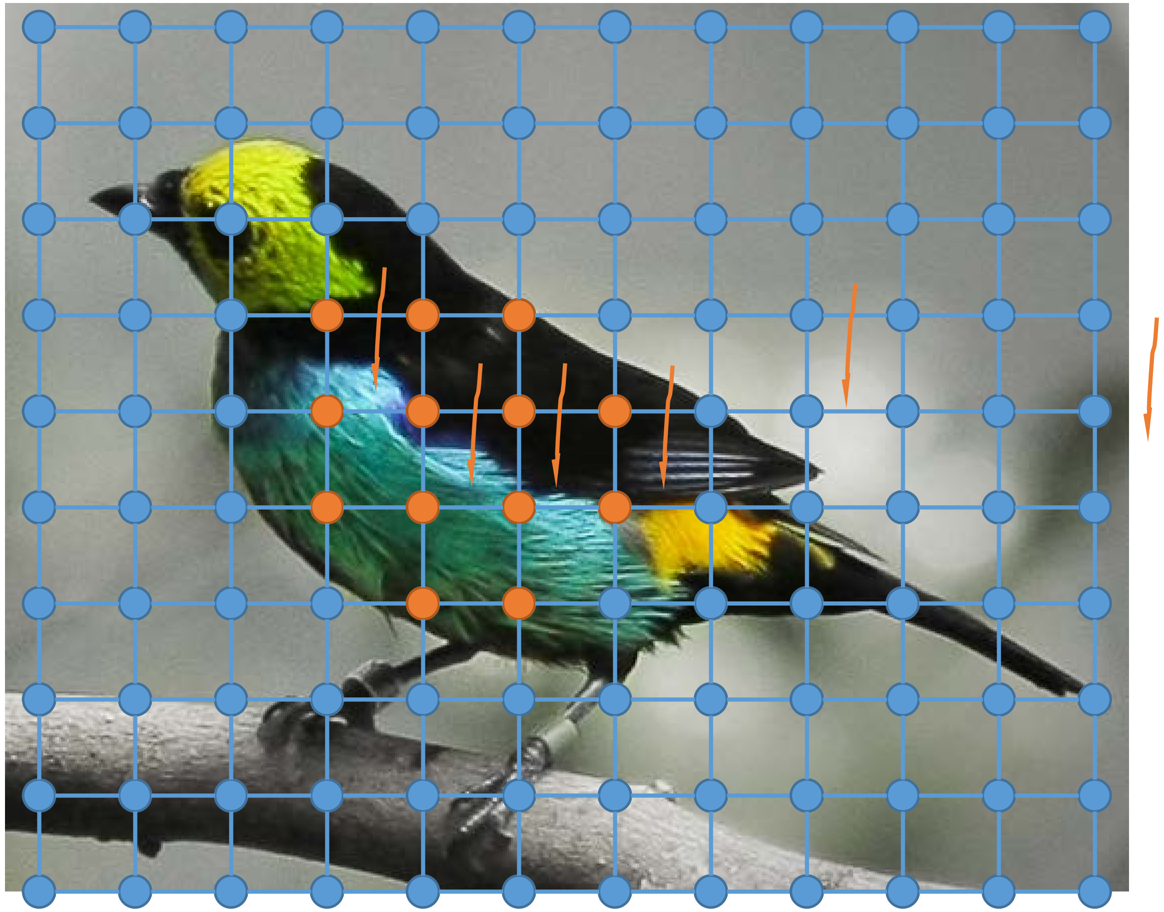
- We want highest sum of weights



- Each “cut” -> points, $W(i,j)$
- Optimization problem
- $W(i,j) = |\mathrm{RGB}(i) - \mathrm{RGB}(j)|$

Graph Cut Background

- We want highest sum of weights

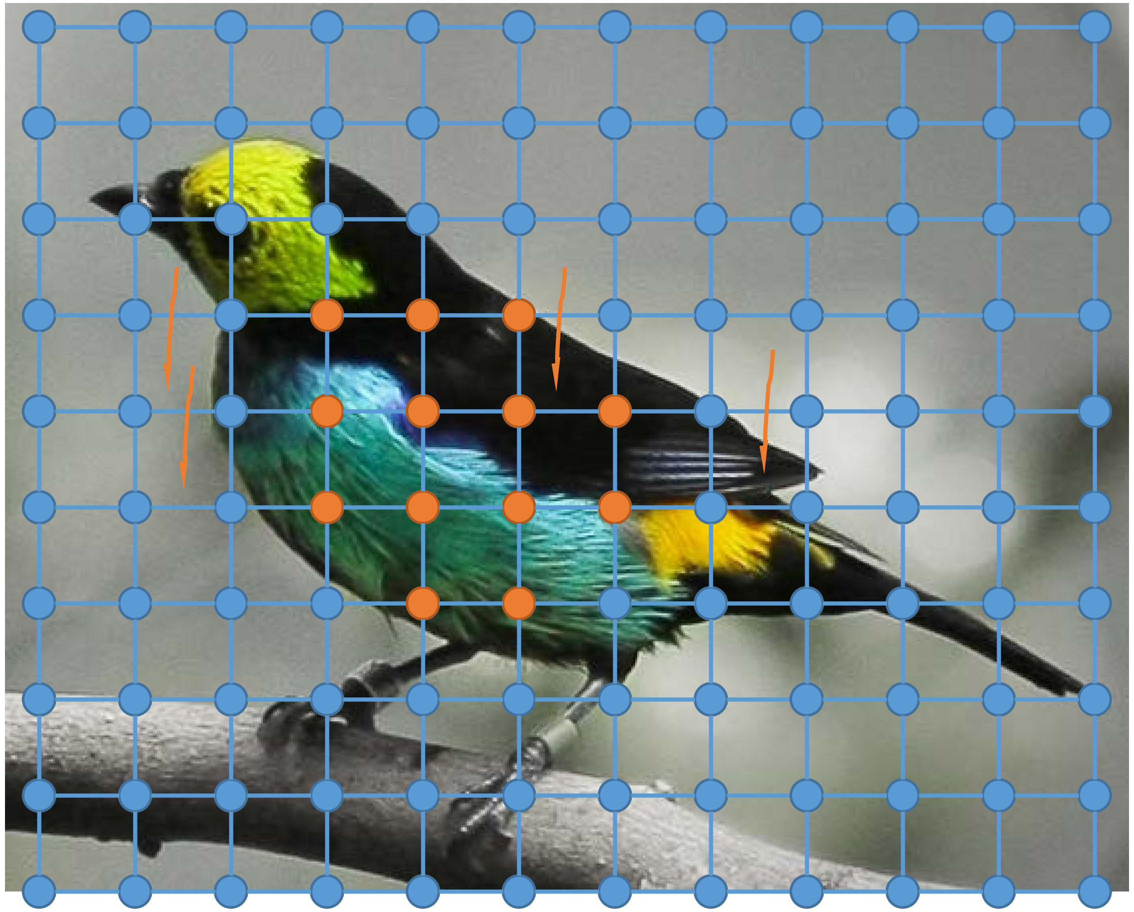


- Each “cut” -> edge weight $W(i,j)$
- Optimization problem $W(i,j) = |\mathrm{RGB}(i) - \mathrm{RGB}(j)|$

These cuts give low weights $W(i,j) = |\mathrm{RGB}(i) - \mathrm{RGB}(j)|$ is low

Graph Cut Background

- We want highest sum of weights



- Each “cut” -> points, $W(i,j)$
- Optimization problem

$$W(i,j) = |\mathbf{RGB}(i) - \mathbf{RGB}(j)|$$

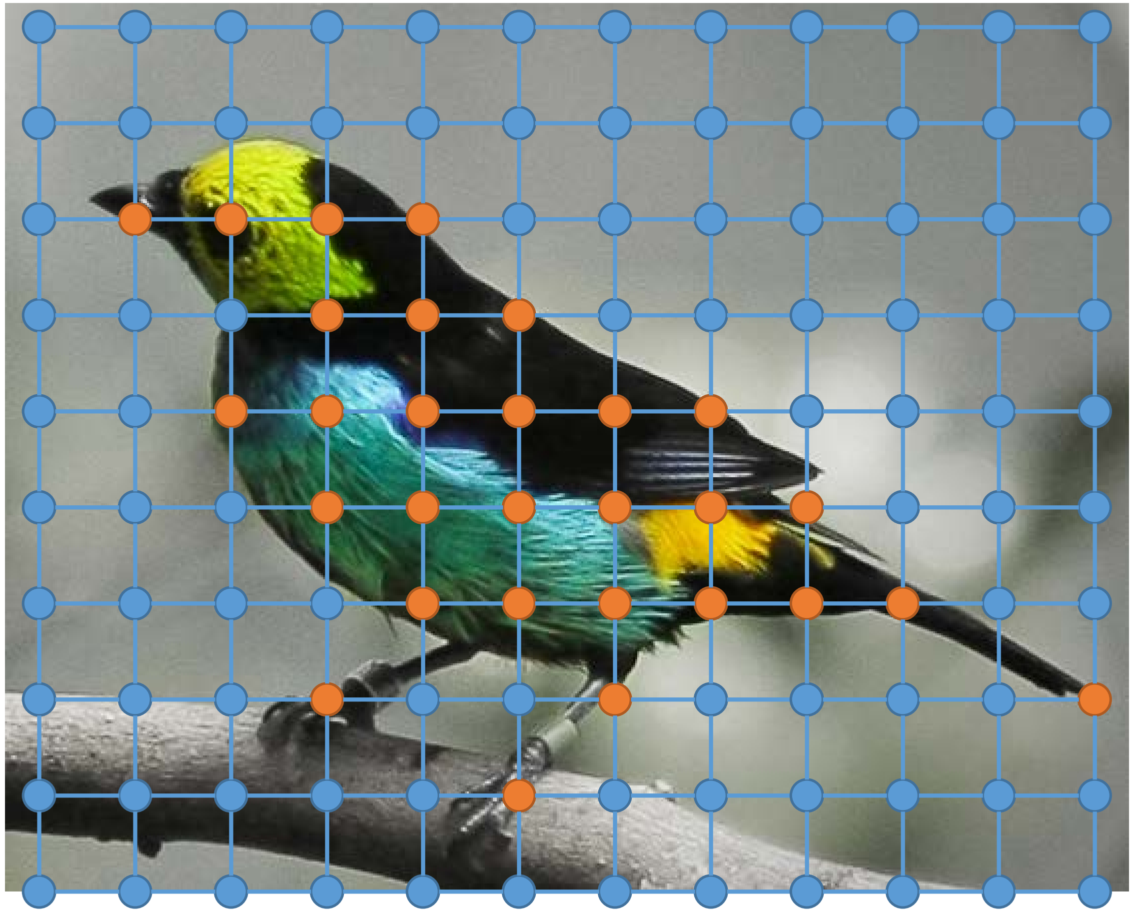
These cuts give high points

$$W(i,j) = |\mathbf{RGB}(i) - \mathbf{RGB}(j)|$$

is high

Graph Cut Background

- Optimization solver

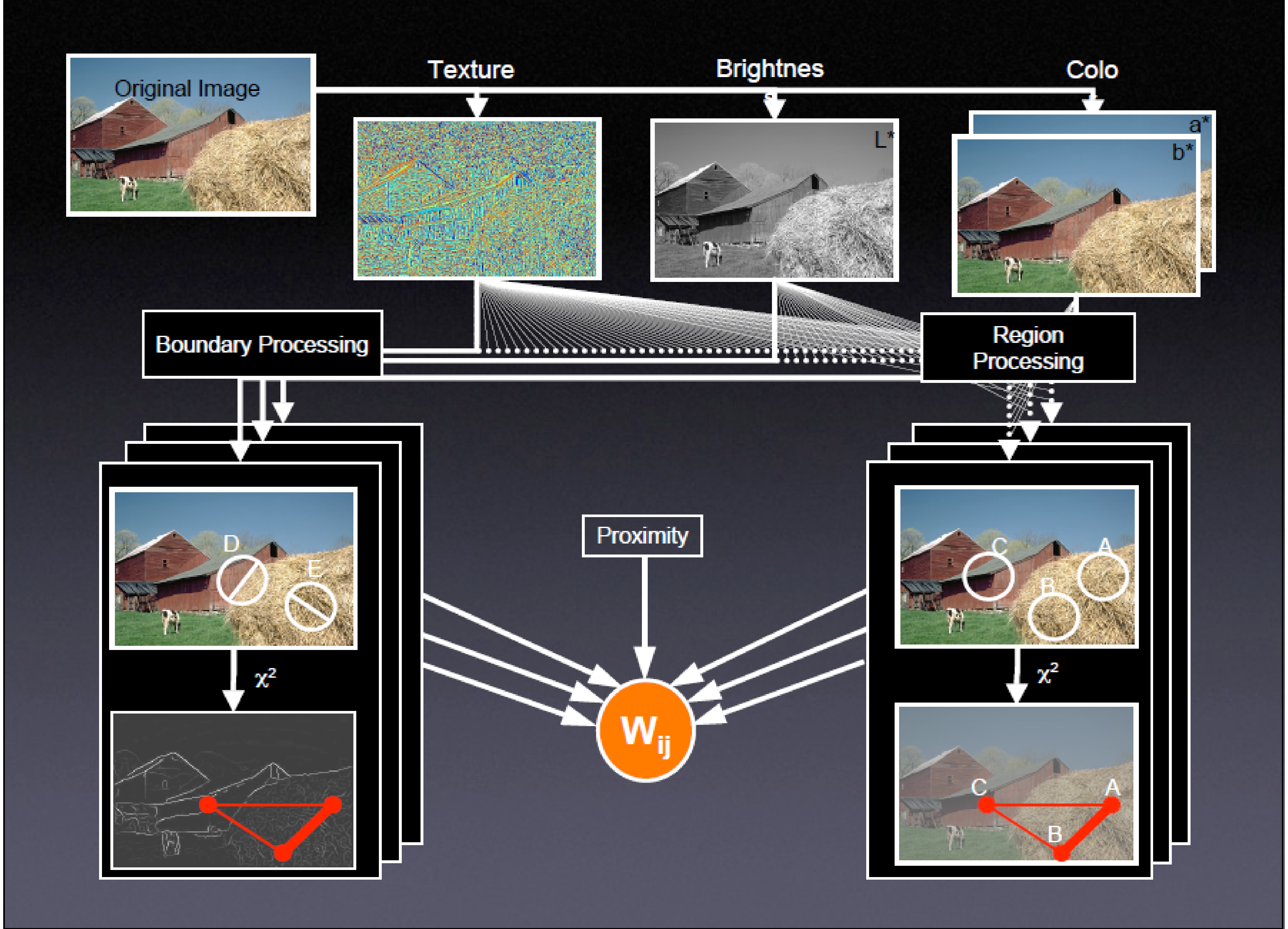


Solver Example Recursion:

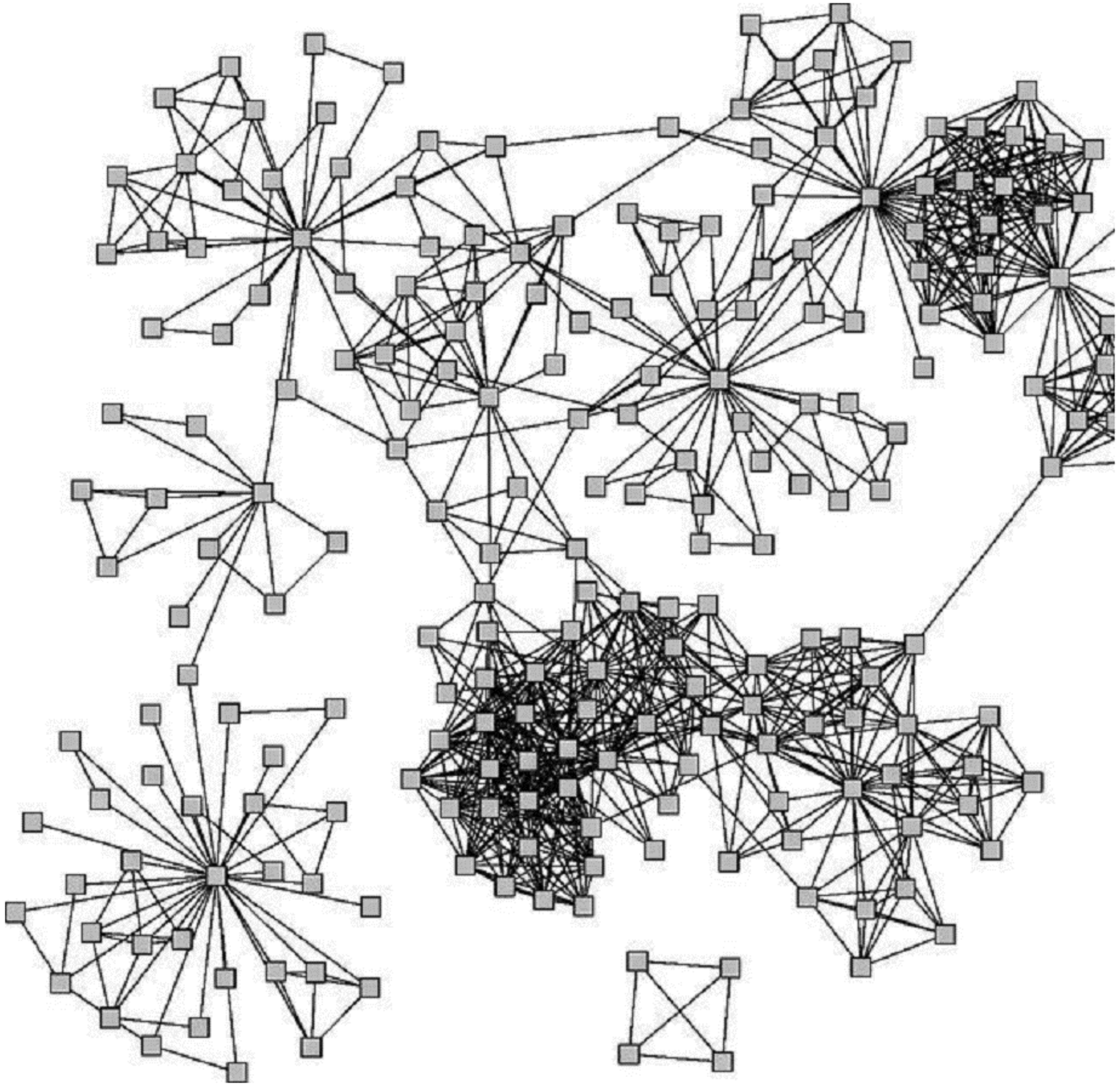
1. Grow
2. If $W(i,j)$ low
    1. Stop
    2. Continue

Graph Cut Background

- Result : Isolation



Original Image
Texture
Brightnes
Colo
L*
a*
b*
Boundary Processing
Region Processing
D
E
C
A
B
χ2
Proximity
Wij



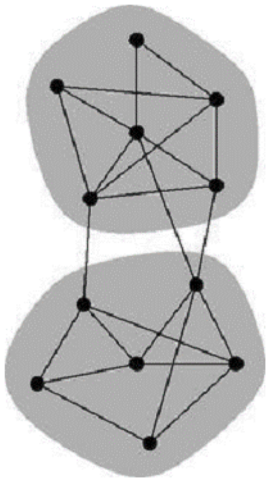
- Graph partitioning and community detection refer to the **division of the vertices** in a graph **based on the connection patterns** of the edges;

- The most common approach is to divide vertices in groups, such that **most edges connect members** from the same group;

Graph partitioning
(the number of groups is given)

Community detection
(the number of groups is unknown)

Graph bisection



- Partitioning in an arbitrarily number of parts can be realized by repeated graph bisection;
- The number of edges between the two groups is called ***cut size***;
- Can be thought as a generalized min cut problem

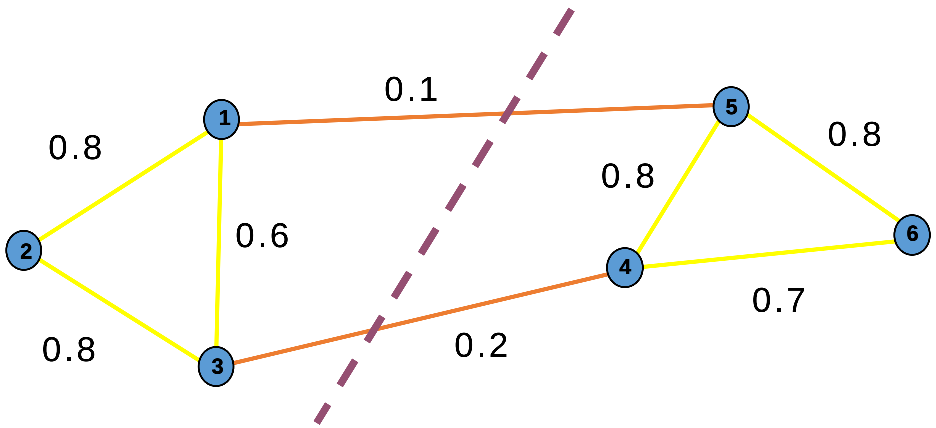
An optimal solution would require to examine all the possible partitions of the network in two groups of sizes n_1 and n_2 respectively. This is:

$$\frac{n!}{n_1!n_2!} = \frac{n!}{n_1!(n-n_1)!}$$

Clustering Objectives

- Traditional definition of a "good" clustering:
  1. Points assigned to same cluster should be highly similar.
  2. Points assigned to different clusters should be highly dissimilar.

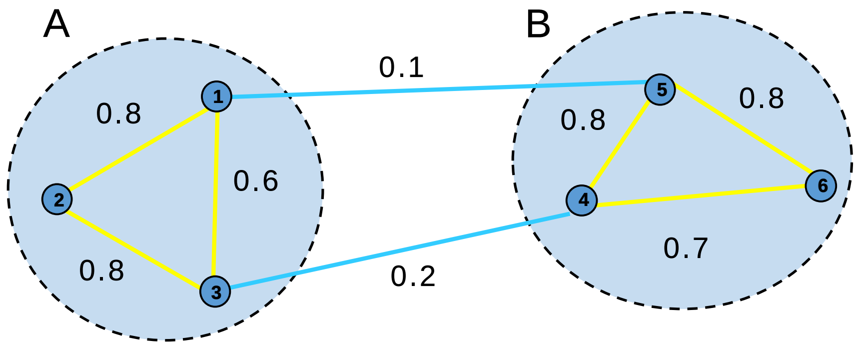
- Apply these objectives to the graph representation



Minimize weight of between-group connections

Graph Cuts

- Express partitioning objectives as a function of the "edge cut" of the partition.
- *Cut:* Set of **edges with only one vertex in a group**. We want to find the minimal cut between groups. The groups that has the minimal cut would be the partition



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

$cut(A,B) = 0.3$

Graph Cut Criteria (continued)

› **Criterion: Normalised-cut** (Shi & Malik,'97)

– Consider the connectivity between groups relative to the density of each group.

$$\min \; Ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

– Normalise the association between groups by *volume*.

- *Vol(A)*: The total weight of the edges originating from group *A*.

- Why use this criterion?
  - – Produces more balanced partitions.

Spectral Graph Theory

- Possible approach
  - Represent a similarity graph as a matrix
  - Apply knowledge from Linear Algebra…

- The *eigenvalues* and *eigenvectors* of a matrix provide global information about its structure.

$$\begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

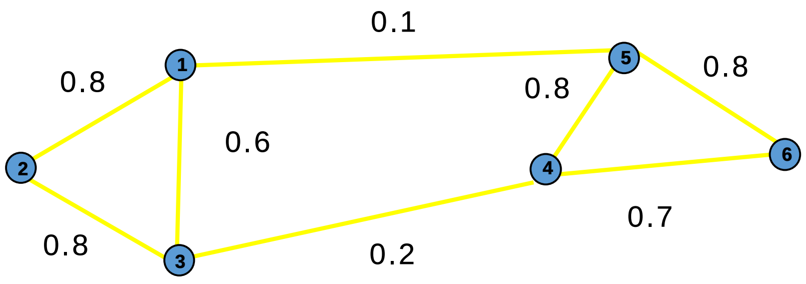
- *Spectral Graph Theory*
  - Analyse the "spectrum" of matrix representing a graph.
  - *Spectrum* : The eigenvectors of a graph, ordered by the magnitude(strength) of their corresponding eigenvalues.

$$\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$$

Matrix Representations

› **Adjacency (affinity) matrix (A)**

- $n \times n$ matrix
- $A = [w_{ij}]$: edge weight between vertex x_i and x_j



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

- **Important properties:**
  - Symmetric matrix

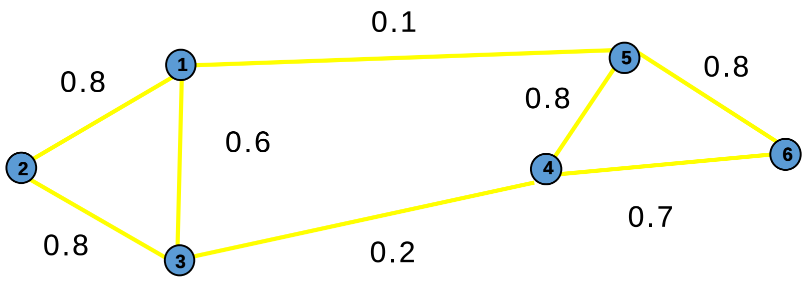
  $\Rightarrow$ Eigenvalues are real

  $\Rightarrow$ Eigenvector could span orthogonal base

Matrix Representations

› **Degree matrix ($\mathbf{D}$)**

- $n \times n$ diagonal matrix
- $D(i,i) = \sum_j w_{ij}$: total weight of edges incident to vertex x_i



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	0	0	0	0	0
x_2	0	1.6	0	0	0	0
x_3	0	0	1.6	0	0	0
x_4	0	0	0	1.7	0	0
x_5	0	0	0	0	1.7	0
x_6	0	0	0	0	0	1.5

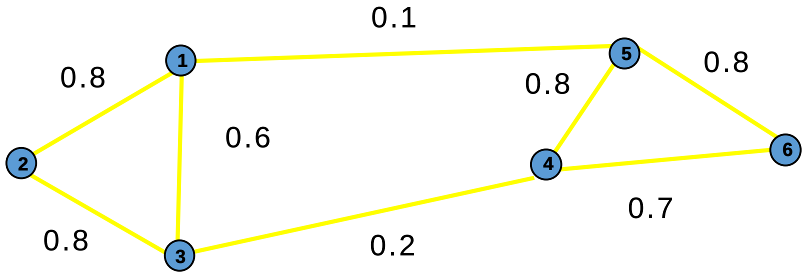
- **Important application:**
  - **Normalise adjacency matrix**

Matrix Representations

› **Laplacian matrix (L)**

– $n \times n$ symmetric matrix

$$L = D - A$$



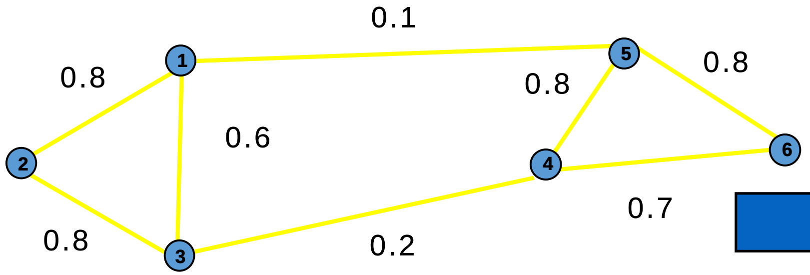
	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

- **Important properties:**
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real and orthogonal
  - Eigenvalues and eigenvectors provide an insight into the connectivity of the graph…

Another option – normalized Laplacian

› **Laplacian matrix (L)**

– $n \times n$ symmetric matrix



$$D^{-0.5} \cdot (D - A) \cdot D^{-0.5}$$

1.00	-0.52	-0.39	0.00	-0.06	0.00
-0.52	1.00	-0.50	0.00	0.00	0.00
-0.39	-0.50	1.00	-0.12	0.00	0.00
0.00	0.00	-0.12	1.00	0.47-	0.44-
-0.06	0.00	0.00	-0.47	1.00	0.50-
0.00	0.00	0.00	0.44-	0.50-	1.00

- **Important properties:**
  – **Eigenvectors are real and normalized**

Find An Optimal Min-Cut

› Express a bi-partition *(A,B)* as a vector

$$p_i = \begin{cases} +1 \ \text{if } x_i \in \mathrm{A} \\ -1 \ \text{if } x_i \in \mathrm{B} \end{cases} = p^T L\, p$$

Laplacian matrix

- **The Laplacian is semi positive**
- **The *Rayleigh Theorem* shows:**
  - **The minimum value for** $\min Ncut(A,B)$ **is given by the <u>2nd smallest eigenvalue of the Laplacian *L*</u>.**
  - **The optimal solution for *p* is given by the eigenvector corresponding λ_2, referred as the *Fiedler Vector*.**

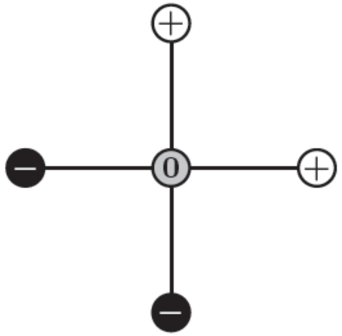
Bisection of a graph

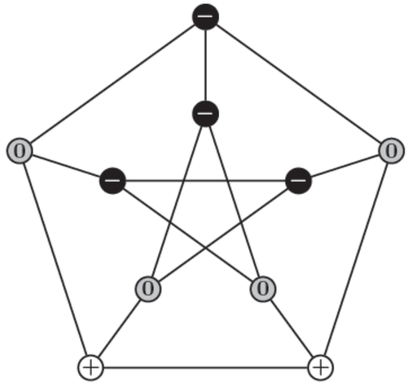
The Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is often used to partition a graph into two approximately equal size pieces with a small number of edges between the two pieces.

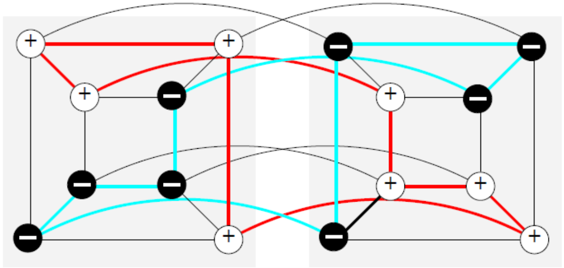
$$L\mathbf{v}_1 = 0, \mathbf{v}_1 = (1,1,1\ldots1)$$

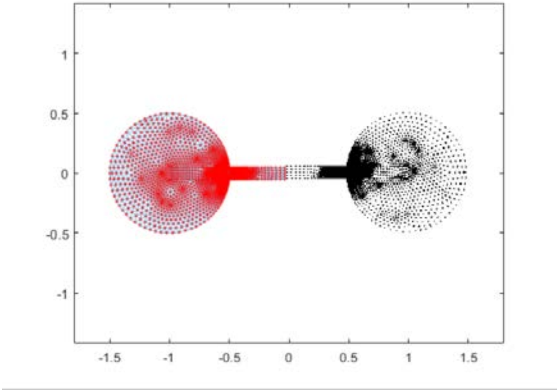
This means: $\mathbf{v}_1 \perp \mathbf{v}_2 \Rightarrow \mathbf{v}_2 = \left(\underbrace{+,+,\ldots+}_{\text{a "half"}},0,..0,\underbrace{-,-,\ldots-}_{\text{a "half"}} \right)$

$$\mathbf{v}_2 = \arg\max_{\mathbf{v}\perp\mathbf{v}_1} \left|A\mathbf{v}\right|^2, \text{ the second singular vector (Fiedler vector)}$$



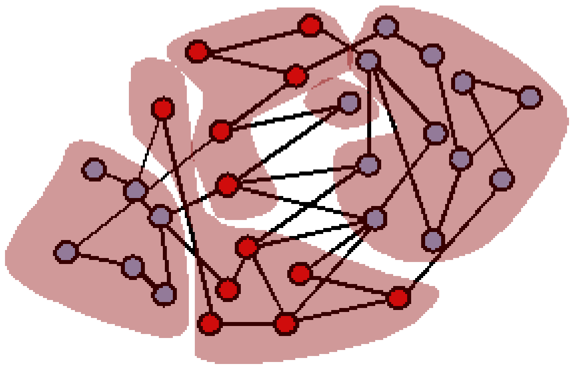


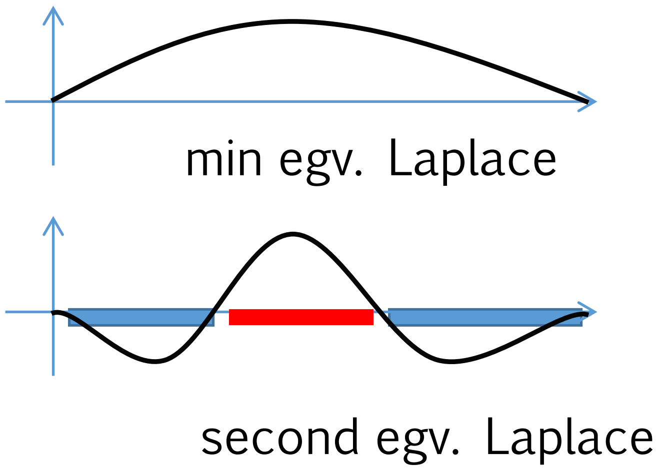




Courant's Nodal Domain Theorem

In graphs: a "nodal domain" is a maximal connected induced subgraph consisting entirely of "positive" and "negative" vertices.





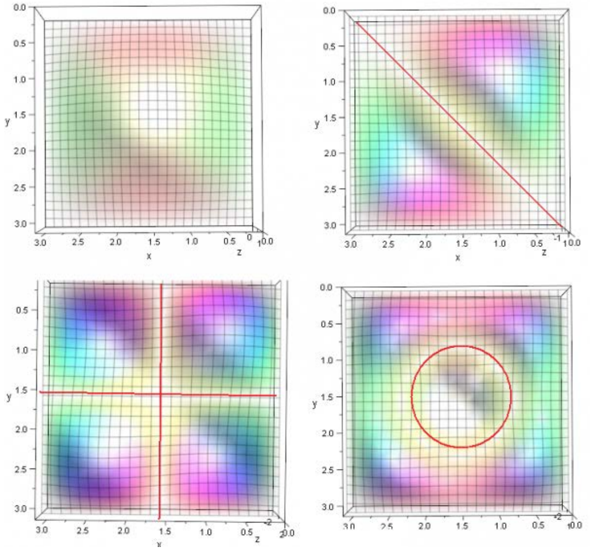
Courant’s Nodal Domain Theorem

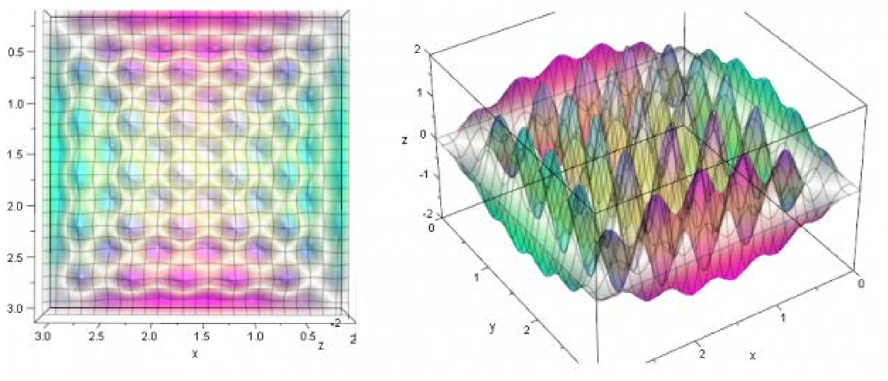
Courant’s Nodal Domain Theorem for elliptic operators on manifolds.

Given the self-adjoint second order differential equation

$$L[u] + \lambda \rho u = 0 \ (\rho > 0)$$

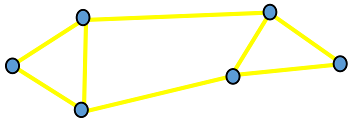
for a domain D *with arbitrary homogeneous boundary conditions; if its eigenfunctions are ordered according to increasing eigenvalues, then the nodes of the* n*-th eigenfunction divide the domain into no more than* n *subdomains.*





Spectral Bisection Algorithm

1. Pre-processing
   - Build Laplacian matrix L of the graph



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

2. **Decomposition**
   - Find eigenvalues X and eigenvectors Λ of the matrix L
   - Map vertices to corresponding components of λ_2

$\Lambda =$

0.0
0.4
2.2
2.3
2.5
3.0

$X =$

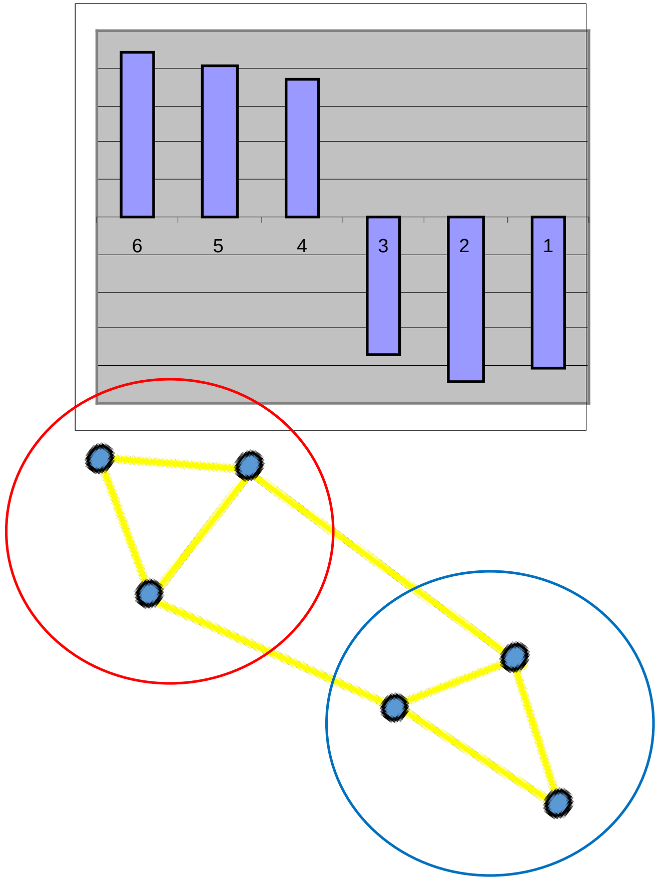
0.4	0.2	0.1	0.4	-0.2	-0.9
0.4	0.2	0.1	-0.	0.4	0.3
0.4	0.2	-0.2	0.0	-0.2	0.6
0.4	-0.4	0.9	0.2	-0.4	-0.6
0.4	-0.7	-0.4	-0.8	-0.6	-0.2
0.4	-0.7	-0.2	0.5	0.8	0.9

x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

Spectral Bi-partitioning Algorithm

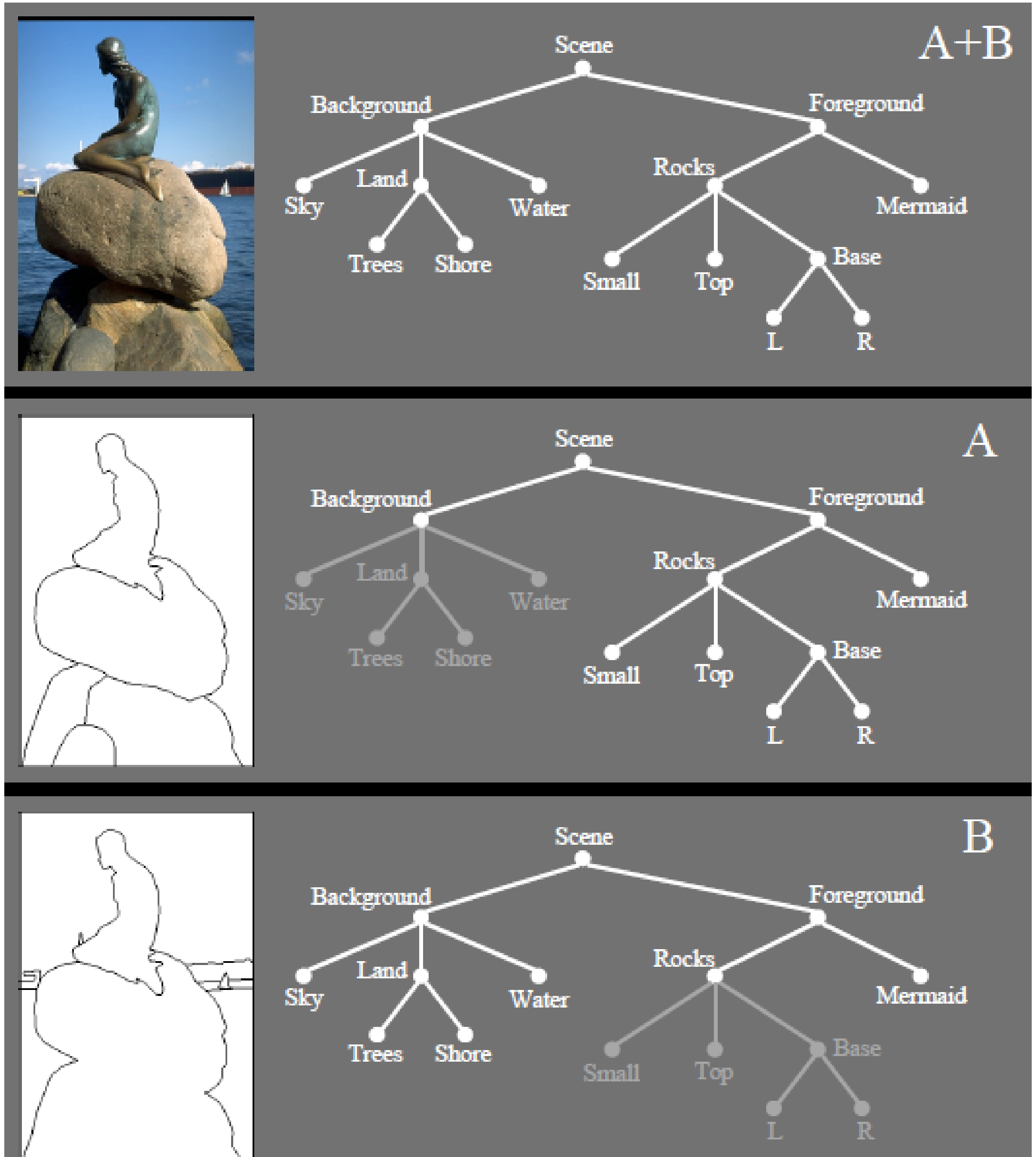
The matrix which represents the eigenvector of the laplacian the eigenvector matched to the corresponded eigenvalues with increasing order

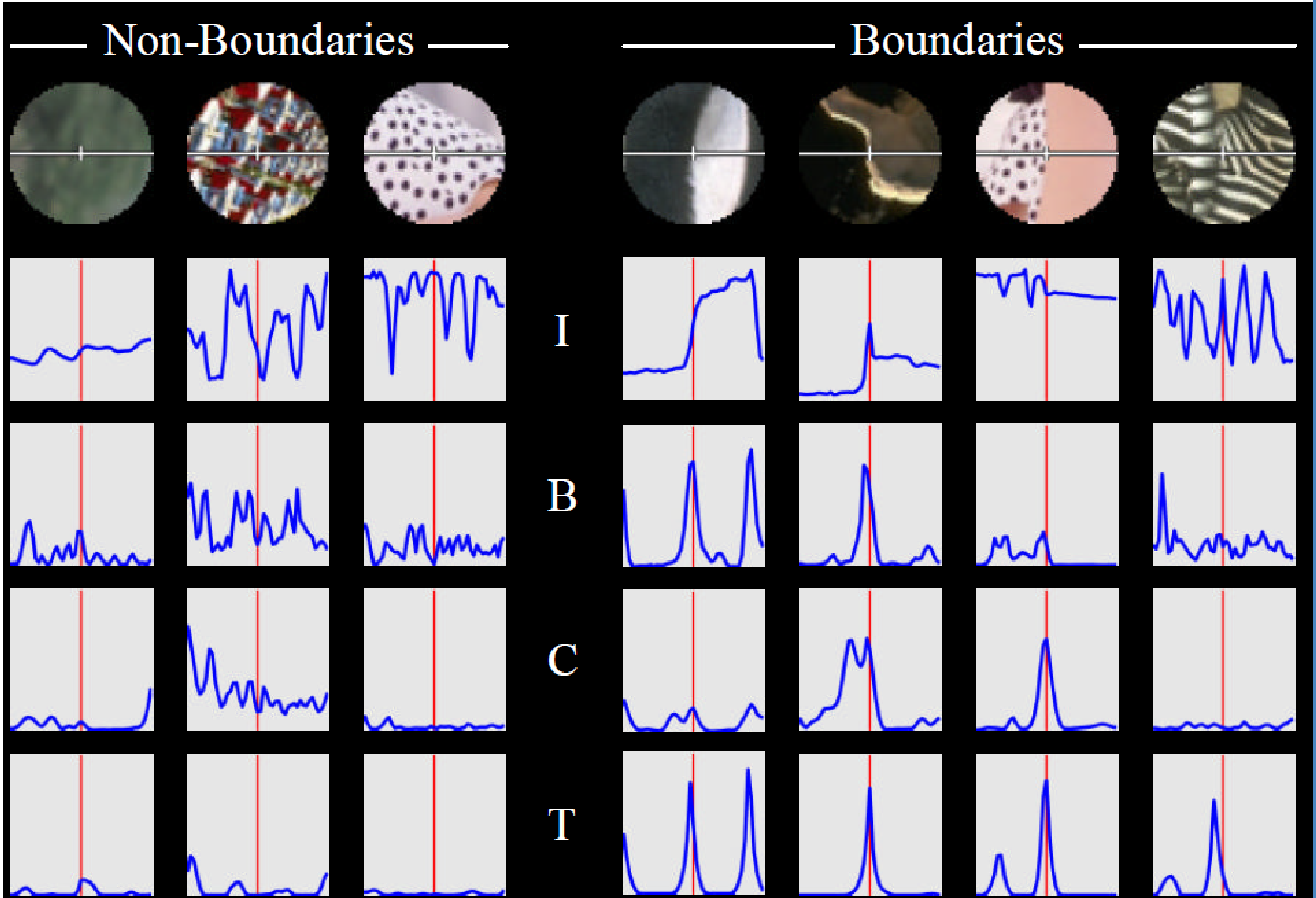
0.41	-0.41	0.65-	0.31-	0.38-	0.11
0.41	-0.44	0.01	0.30	0.71	0.22
0.41	-0.37	0.64	0.04	0.39-	0.37-
0.41	0.37	0.34	0.45-	0.00	0.61
0.41	0.41	0.17-	0.30-	0.35	0.65-
0.41	0.45	0.18-	0.72	0.29-	0.09



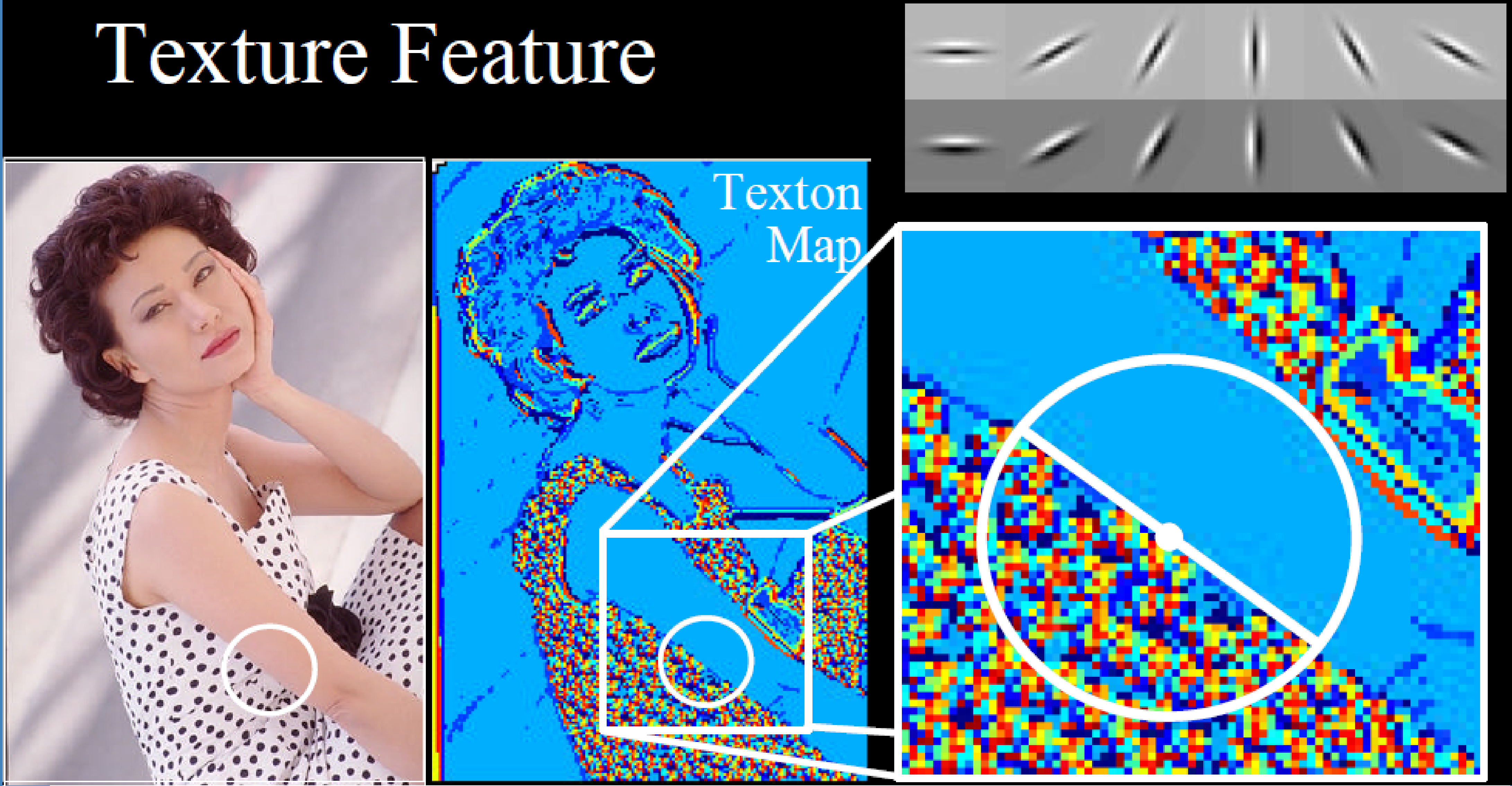
Recursive bi-partitioning (Hagen et al.,'91)

- Partition using only one eigenvector at a time
- Use procedure recursively
- Example: Image Segmentation
  - Uses 2nd (smallest) eigenvector to define optimal cut
  - Recursively generates two clusters with each cut

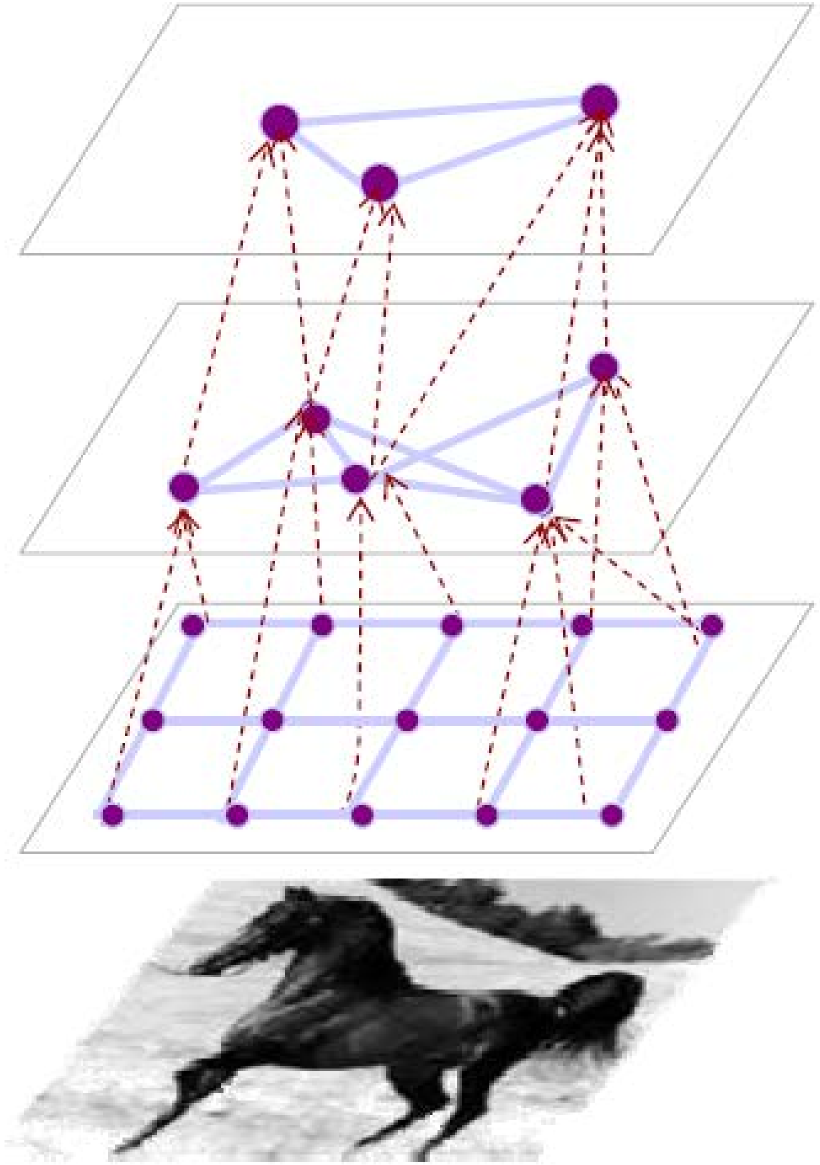

A+B
Scene
Background
Foreground
Sky
Land
Water
Trees
Shore
Rocks
Mermaid
Small
Top
Base
L
R
A
Scene
Background
Foreground
Sky
Land
Water
Trees
Shore
Rocks
Mermaid
Small
Top
Base
L
R
B
Scene
Background
Foreground
Sky
Land
Water
Trees
Shore
Rocks
Mermaid
Small
Top
Base
L
R


Non-Boundaries
Boundaries
I
B
C
T

Texture Feature



Hierarchy of nodal domains



Spectral Clustering of graphs (by SVD)

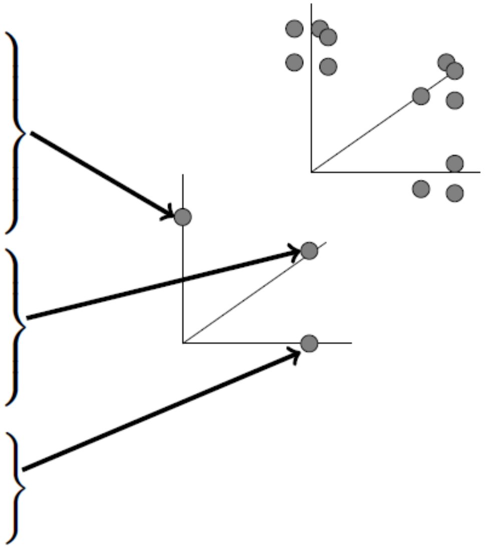
If one is clustering the rows of a matrix where the matrix elements are real numbers, then one selects a value for k and computes the top k-singular vectors.

The rows of the matrix are projected onto the space spanned by the top k-singular vectors and k-means clustering is used to find k clusters in this lower dimensional space.

$$\begin{pmatrix}
1&1&1&1&1&0&0&0&0&0&0&0\\
1&1&1&1&1&0&0&0&0&0&0&0\\
1&1&1&1&1&0&0&0&0&0&0&0\\
1&1&1&1&1&0&0&0&0&0&0&0\\
1&1&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&1&1&1&0&0&0\\
0&0&0&0&0&1&1&1&1&0&0&0\\
0&0&0&0&0&1&1&1&1&0&0&0\\
0&0&0&0&0&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&1&1&1
\end{pmatrix}
\quad
\begin{pmatrix}
1&0&0\\
1&0&0\\
1&0&0\\
1&0&0\\
1&0&0\\
0&1&0\\
0&1&0\\
0&1&0\\
0&1&0\\
0&0&1\\
0&0&1\\
0&0&1
\end{pmatrix}$$

adjacency matrix

singular vectors



Spectral Clustering of Data (by SVD)

$$\mathbf{X} = [\mathbf{f}_1, \ldots \mathbf{f}_n] \in R^{n \times N} \text{(a data matrix)}, \quad \mathbf{m} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{f}_k \text{ (the centroid)},$$

$$\mathbf{F} = [\mathbf{f}_1 - \mathbf{m}, \ldots \mathbf{f}_n - \mathbf{m}] \text{(variances)},$$

$$\mathbf{Cov} = \frac{1}{N-1}\mathbf{F}\mathbf{F}^T, \quad \mathbf{Cov}\,\mathbf{u}_k = \sigma_k \mathbf{u}_k, \quad \underbrace{\sigma_1 \geq \sigma_2 \geq}_{\text{only } n-1 \text{ eigenvalues} \neq 0!} \quad \ldots \sigma_N \mapsto \mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_{n-1}]$$

$$\mathbf{U} \text{ forms the orthonormal basis in } R^{n\text{-}1}: \quad (\mathbf{f}_k - \mathbf{m}) \mapsto \mathbf{g}_k = \mathbf{U}(\mathbf{f}_k - \mathbf{m}) \in R^{n\text{-}1}$$

$$\Rightarrow \mathbf{f}_k = \mathbf{U}^T \mathbf{g}_k + \mathbf{m}$$

$$\mathbf{U}^T\mathbf{U}, \text{ the projecton onto the subspace } [\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_{n-1}];$$

$$(1 - \mathbf{U}^T\mathbf{U}), \text{ the projecton onto the orthogonal subspace}$$

$$\text{The } (n-1)\text{- dimensional representation vectors}: \varphi_k = \mathbf{U}^T\mathbf{U}\mathbf{f}_k + (1 - \mathbf{U}^T\mathbf{U})\mathbf{m}$$