Learning Spectral Graph Segmentation

AISTATS 2005

Timothée Cour
Jianbo Shi
Computer and Information Science Department
University of Pennsylvania

Nicolas Gogin
Computer Science
Ecole Polytechnique

Graph-based Image Segmentation

- Weighted graph G =(V,W)
- V = vertices (pixels i)
- Similarity between pixels i and j : $W_{ij} = W_{ji} \geq 0$

Segmentation = graph partition of pixels

Spectral Graph Segmentation

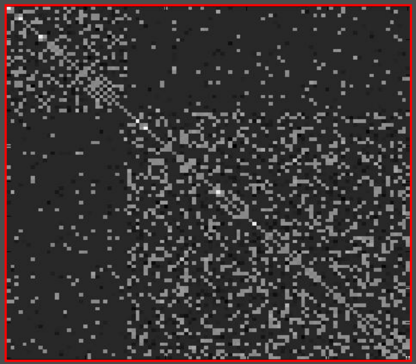


Image I → Graph Affinities

W=W(I,Θ)

Intensity
Color
Edges
Texture
...

Spectral Graph Segmentation

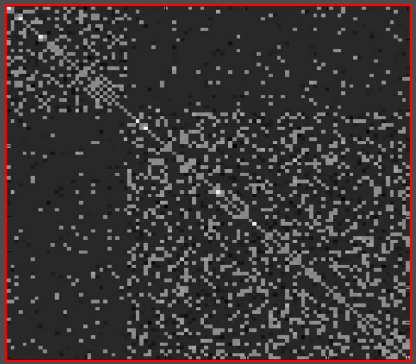


Image I → Graph Affinities

W=W(I,Θ)

Intensity
Color
Edges
Texture
...

$$NCut(A,B) = \frac{cut(A,B)}{Vol\ A \times Vol\ B}$$

Spectral Graph Segmentation

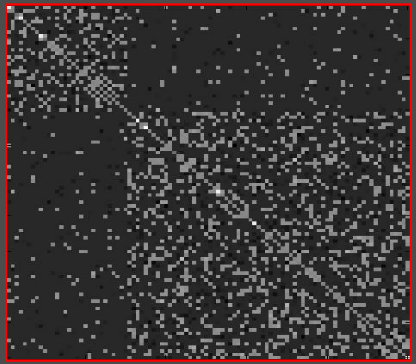


Image I → Graph Affinities $W=W(I,\Theta)$ → Eigenvector $X(W)$

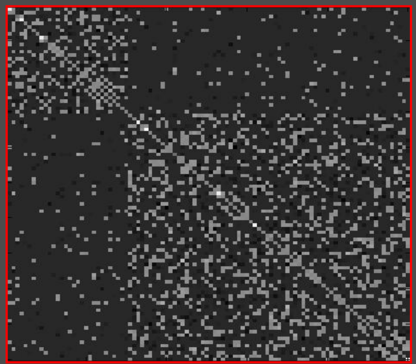
$$NCut(A,B) = \frac{cut(A,B)}{Vol\ A \times Vol\ B}$$

⇒

$$WX = \lambda DX$$

$$X_A(i) = \begin{cases} 1 & \text{if } i \in A \\ 0 & \text{if } i \notin A \end{cases}$$

Spectral Graph Segmentation



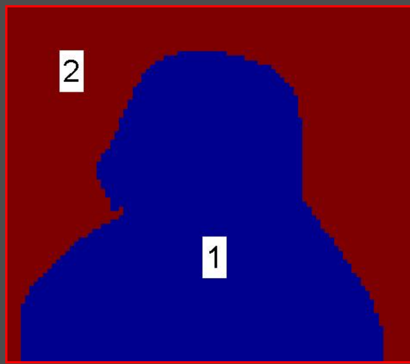


Image I → Graph Affinities W=W(I,Θ) → Eigenvector X(W) → Discretisation

$$NCut(A,B) = \frac{cut(A,B)}{Vol\ A \times Vol\ B}$$

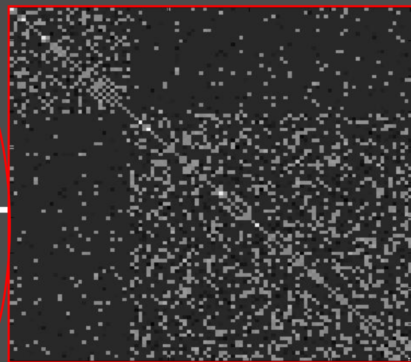
⇒

$$WX = \lambda DX$$

$$X_A(i) = \begin{cases} 1 \text{ if } i \in A \\ 0 \text{ if } i \notin A \end{cases}$$

Spectral Graph Segmentation

Intensity
+
Color
+
Edges
+
Texture
+
...



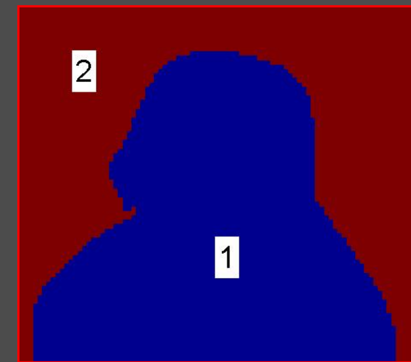


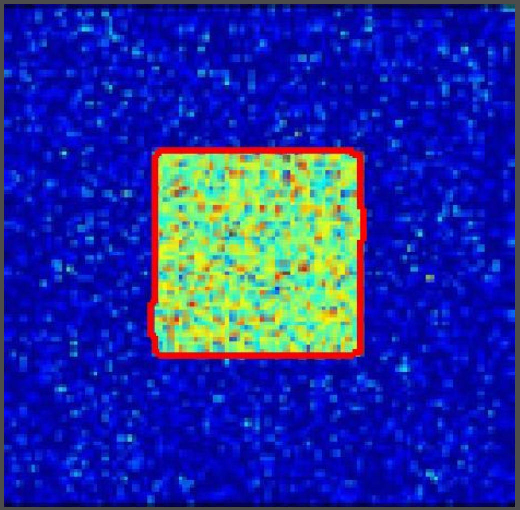
Image I → Graph Affinities $W=W(I,\Theta)$ → Eigenvector $X(W)$

Learn graph parameters Θ

Target segmentation

Reverse pipeline

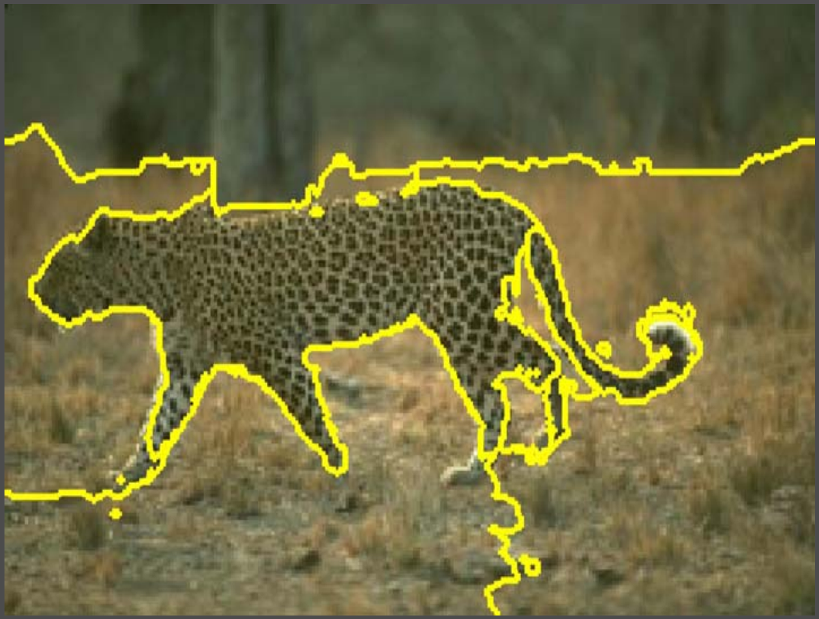
intensity cues



edges cues

color cues

[Shi & Malik, 97]

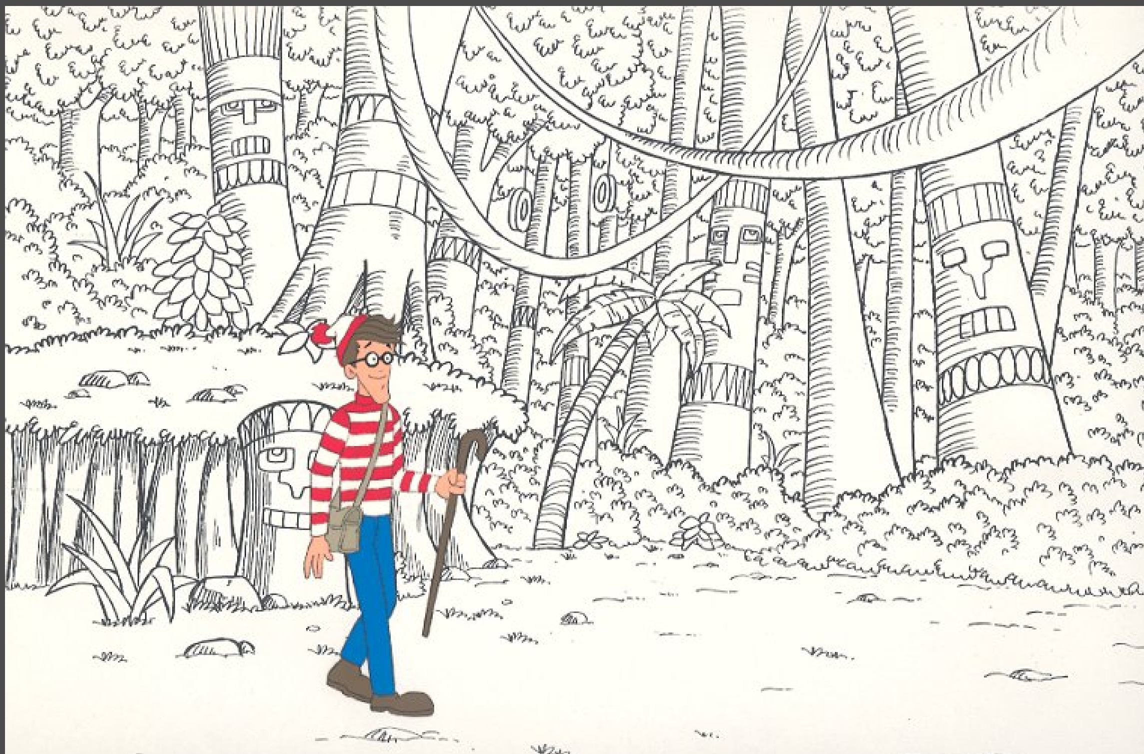


Multiscale cues

Texture cues

[Yu, 04; Fowlkes 04]

Where is Waldo ?



Do you use
- Edges cues ?
- Color cues ?
- Texture cues ?

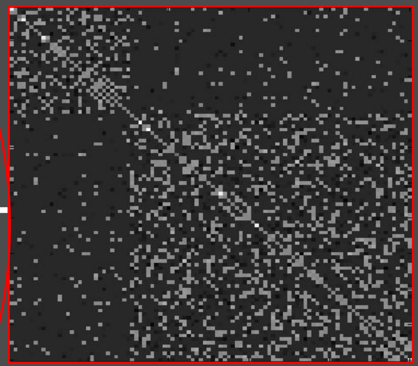
-That’s not enough, you need

Shape cues

High-level object priors

Spectral Graph Segmentation

Intensity
+
Color
+
Edges
+
Texture
+
...



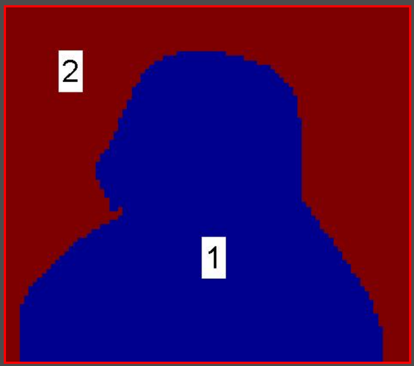
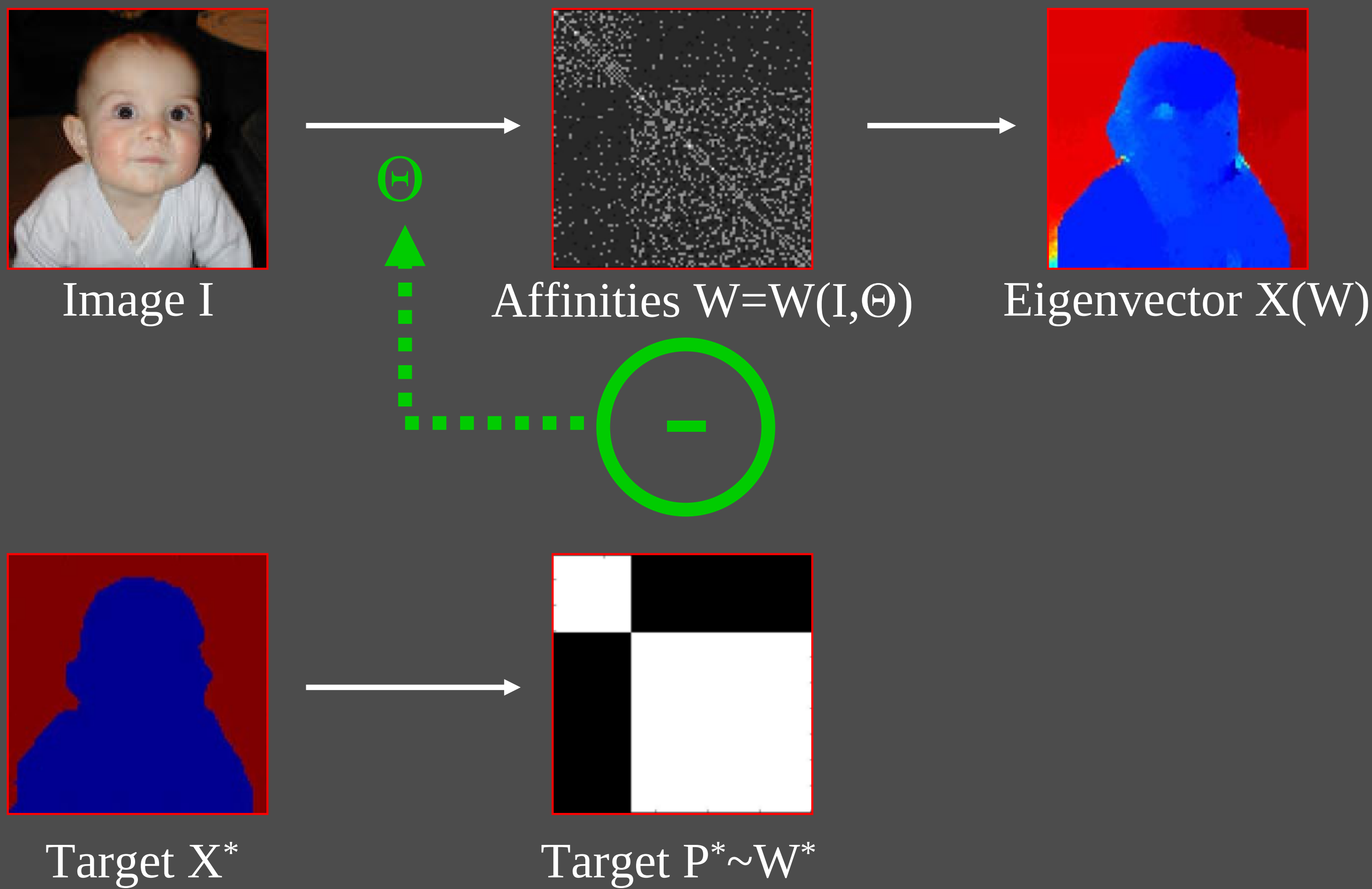


Image I → Graph Affinities $W = W(I, \Theta)$ → Eigenvector $X(W)$

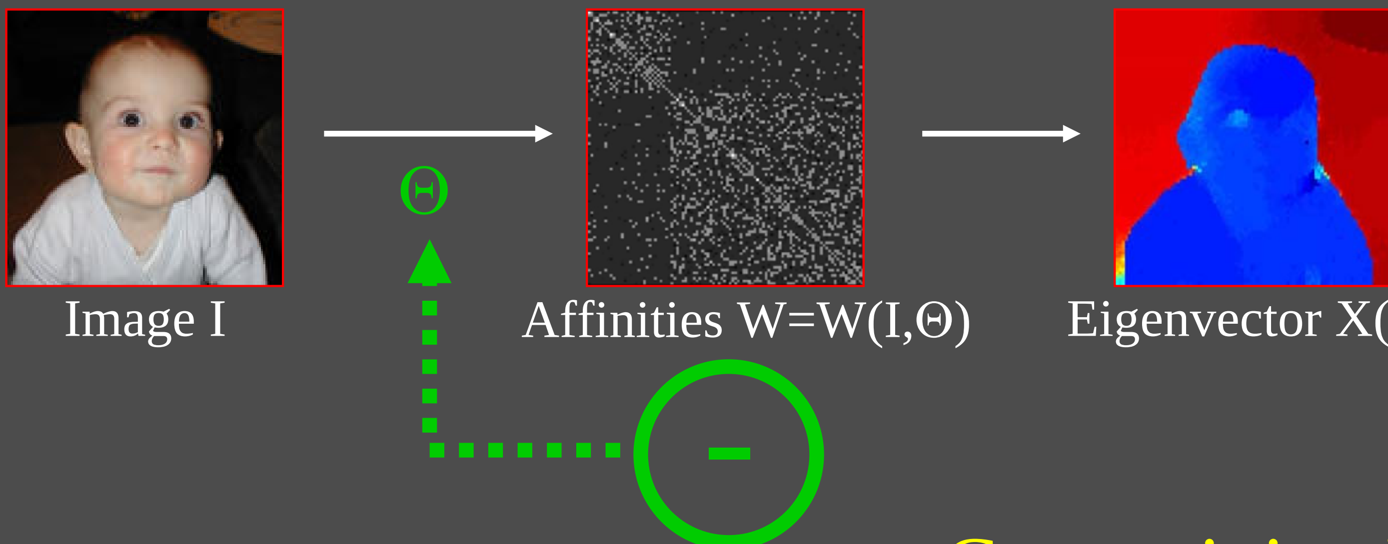
Learn graph parameters Θ

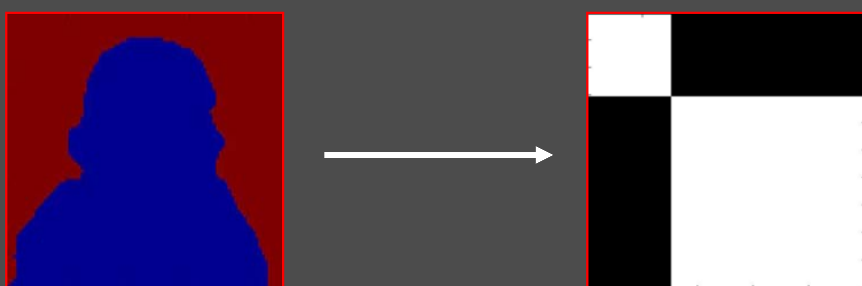
Target segmentation



Error criterion on affinity matrix W

[1] Meila and Shi (2001)
[2] Fowlkes, Martin, Malik (2004)





Constraining Wij is overkill:

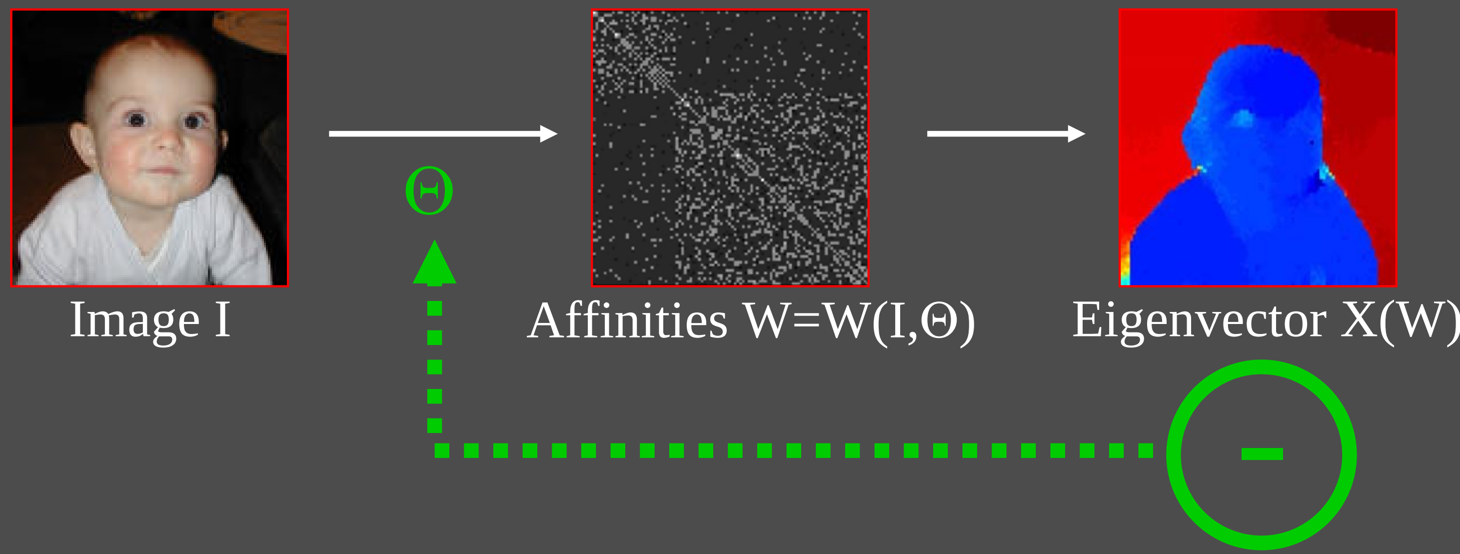
Many W have segmentation

W: $O(n^2)$ parameters

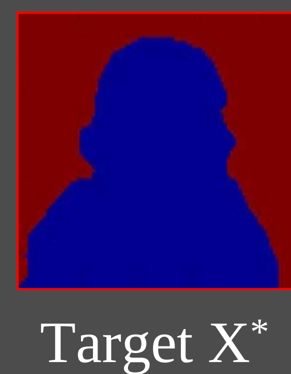
Segments : O(n) parameters

Error criterion on affinity matrix W

[1] Meila and Shi (2001)
[2] Fowlkes, Martin, Malik (2004)



Error criterion on partition vector X only!



Energy function for segmenting 1 image I

$$E_I(W) = \| X(W(I,\Theta)) - X^*(I) \|^2$$

Eigenvector X(W)

Target X^*

Eigenvector X(W)

Target X^*

Energy function for segmenting 1 image I

$$E_I(W) = \| X(W(I,\Theta)) - X^*(I) \|^2$$

$$\text{Min.} \quad E(\Theta) = \sum_{\text{images } I} \| X(W(I,\Theta)) - X^*(I) \|^2$$

Eigenvector X(W)

Target X*

Energy function for image I

$$E_I(W) = \| X(W(I,\Theta)) - X^*(I) \|^2$$

$$\text{Min.} \quad E(\Theta) = \sum_{\text{images } I} \| X(W(I,\Theta)) - X^*(I) \|^2$$

Can use gradient descent…

...but $X(W)$ is only implicitly defined, by

$$\left(W(I,\Theta) - \lambda D\right) X = 0,$$

$$X \neq f(W,\Theta)$$

Can Not backtrack easily

$$\text{Min.} \quad E(\Theta) = \sum_{\text{images } I} \| X(W(I,\Theta)) - X^*(I) \|^2$$

Gradient descent:

$$\Delta\Theta = -\eta \frac{\partial E}{\partial \Theta} = -\eta \frac{\partial E}{\partial X} \frac{\partial X}{\partial W} \frac{\partial W}{\partial \Theta}$$

$$\text{Min.} \quad E(\Theta) = \sum_{\text{images } I} \| X(W(I,\Theta)) - X^*(I) \|^2$$

Gradient descent:

X(W) is implicit

$$\Delta\Theta = -\eta \frac{\partial E}{\partial \Theta} = -\eta \frac{\partial E}{\partial X} \frac{\partial X}{\partial W} \frac{\partial W}{\partial \Theta}$$

E(X) is quadratic

depends on W(Θ)

$$\Delta\Theta = -\eta \; \frac{\partial E}{\partial X} \; \frac{\partial X}{\partial W} \; \frac{\partial W}{\partial \Theta}$$

Theorem: Derivative of NCut eigenvectors

The map $W \rightarrow (X, \lambda)$ is C^∞ over Ω and we can express the derivatives over any C^1 path $W(t)$ as:

$$\frac{d\lambda(W(t))}{dt} = \frac{X^T\left(W' - \lambda D'\right)X}{X^T D X}$$

$$\frac{dX(W(t))}{dt} = -\left(W - \lambda D\right)^{\dagger}\left(W' - \lambda D' - \frac{d\lambda}{dt} D\right)X$$

$$\Delta\Theta = -\eta \ \frac{\partial E}{\partial X} \ \frac{\partial X}{\partial W} \ \frac{\partial W}{\partial \Theta}$$

Theorem : Derivative of NCut eigenvectors

The map $W \rightarrow (X, \lambda)$ is C^{∞} over Ω and we can express the derivatives over any C^1 path $W(t)$ as :

$$\frac{d\lambda(W(t))}{dt} = \frac{X^T \left(W' - \lambda D'\right) X}{X^T D X}$$

$$\frac{dX(W(t))}{dt} = -\left(W - \lambda D\right)^{\dagger} \left(W' - \lambda D' - \frac{d\lambda}{dt} D\right) X$$

Feasible set in the space of graph weight matrices : $W \in \Omega$ iff

1) W is $n \times n$ symmetric matrix
2) $W\mathbf{1} > 0$
3) λ_2 is single with $WX_2 = \lambda_2 D X_2$

$$\Delta\Theta = -\eta \; \frac{\partial E}{\partial X} \; \frac{\partial X}{\partial W} \; \frac{\partial W}{\partial \Theta}$$

Theorem : Derivative of NCut eigenvectors

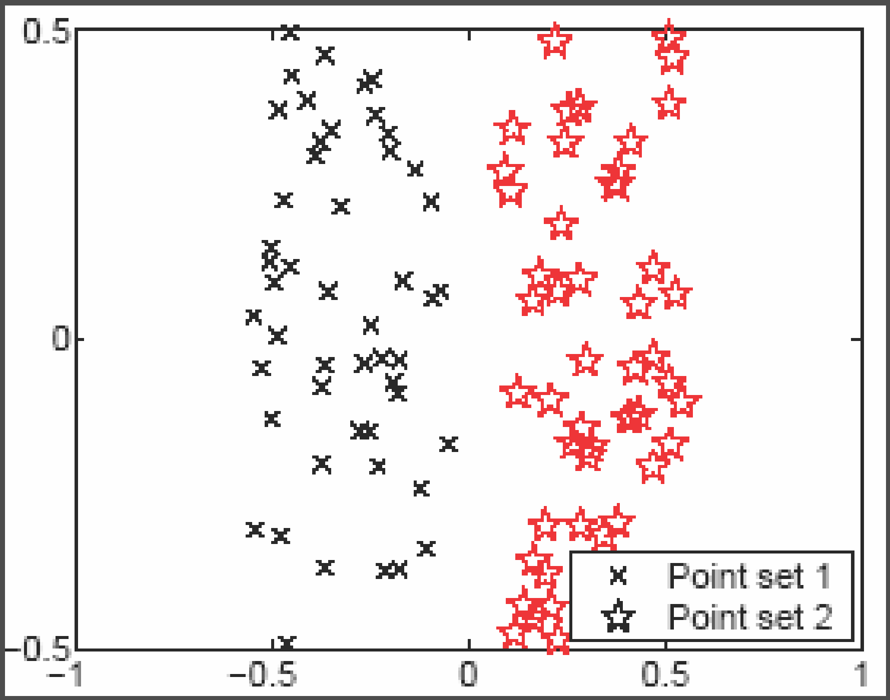
The map W $\rightarrow$ (X,λ) is C^∞ over Ω and we can express the derivatives over any C^1 path $W(t)$ as :

$$\frac{d\lambda(W(t))}{dt} = \frac{X^T\left(W' - \lambda D'\right)X}{X^T D X}$$

$$\frac{dX(W(t))}{dt} = -\left(W - \lambda D\right)^\dagger \left(W' - \lambda D' - \frac{d\lambda}{dt} D\right) X$$

[Meila, Shortreed, Xu, ’05]

Task 1: Learning 2D point segmentation

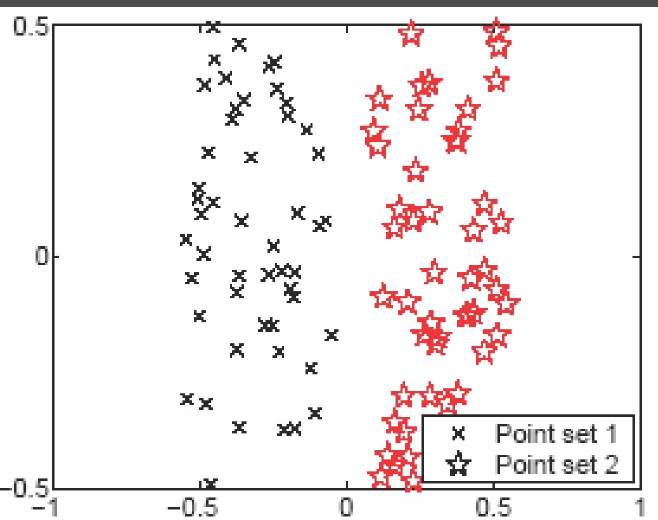


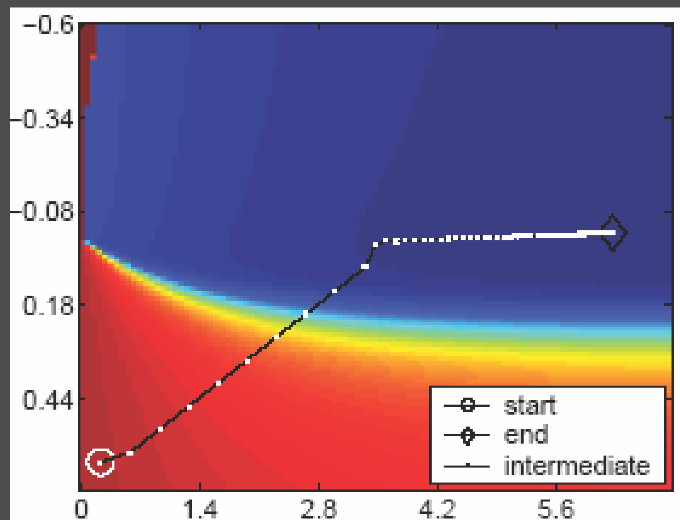
100 points at (x_i, y_i)

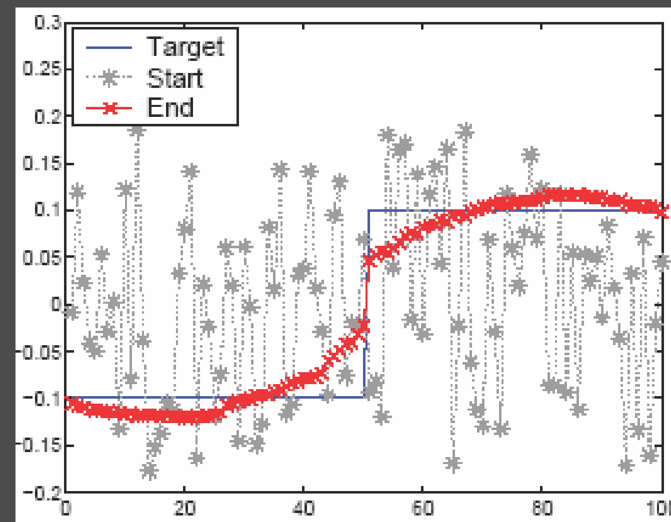
$$W_{ij} = e^{-\sigma_x |x_i - x_j|^2 - \sigma_y |y_i - y_j|^2}$$

$$\text{parameters} : \Theta = (\sigma_x, \sigma_y)$$

$$\text{Learn} \quad W_{ij} = \exp\left(-\sigma_x (x_i - x_j)^2 - \sigma_y (y_i - y_j)^2\right)$$





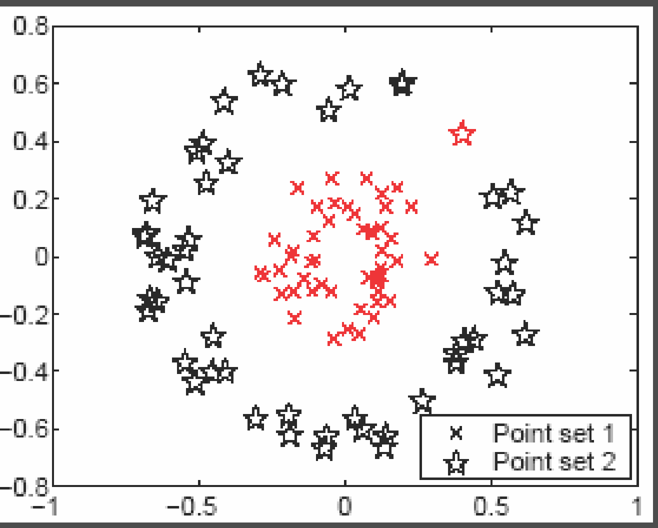


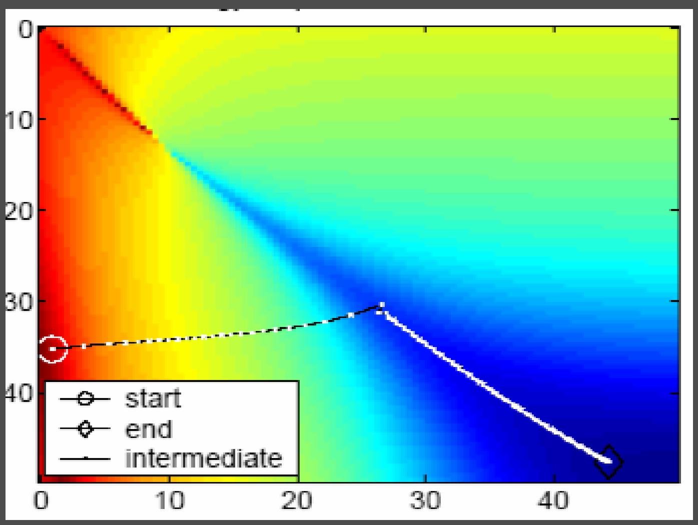
Ground truth segmentation X^* and X(W) learned

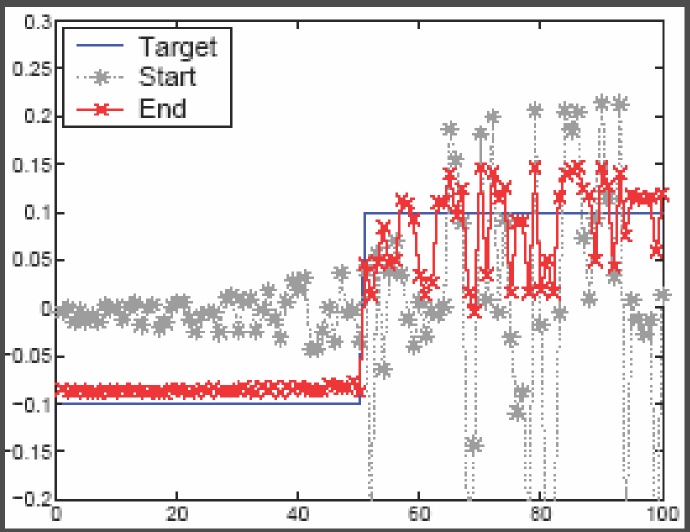
$$E(\sigma_x, \sigma_y) = \| X\left(W(\sigma_x, \sigma_y)\right) - X^* \|^2$$

Learning 2D point segmentation

$$\text{Learn} \quad W_{ij} = \exp\left(-\sigma_x (x_i - x_j)^2 - \sigma_y (y_i - y_j)^2\right)$$







$$E(\sigma_x,\sigma_y) = \| X\left(W(\sigma_x,\sigma_y)\right) - X^*\|^2$$

Ground truth segmentation X^* and X(W) learned

Proposition : Exponential convergence

The PDE $\dot{W} = -\dfrac{\partial E}{\partial W}$ either

- converges to a global minimum W_∞ :

$$E\left(W(t)\right) \to 0, \text{ exponentially}$$

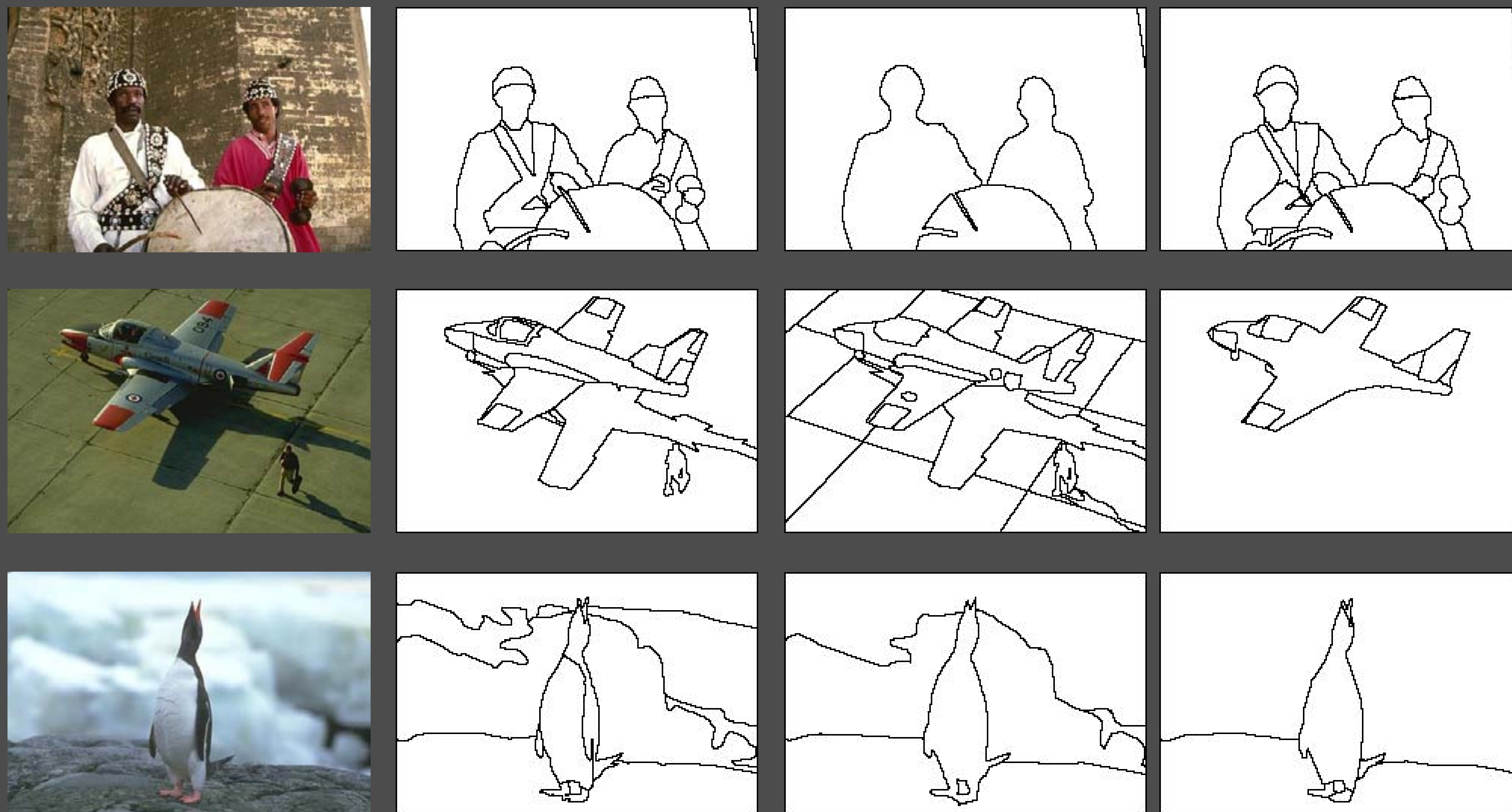
- or escapes any compact $K \subset \Omega$

  this happens when

  1) $\lambda_2(t) \to 1$ or $\lambda_2(t) - \lambda_3(t) \to 0$

  2) $D_{W(t)}(i,i) \to 0$ for some i

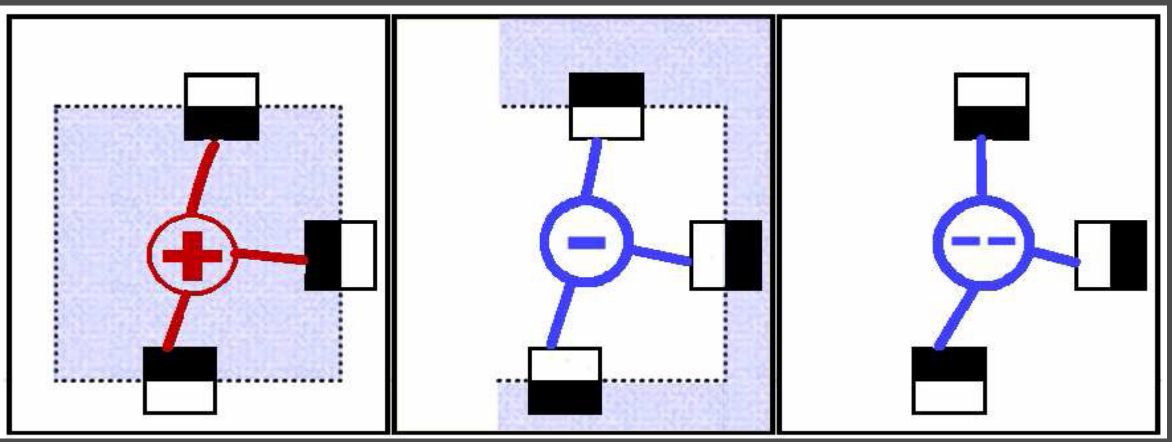
Is there a ground truth segmentation ?



Graph nodes $= \left\{ \text{edge } e_i \text{ at } (x_i, y_i) \text{ with angle } \alpha_i \right\}$

Task 2: Learning rectangular shape

$$W(e_i, e_j) = f(x_i, y_i, \alpha_i; x_j, y_j, \alpha_j)$$



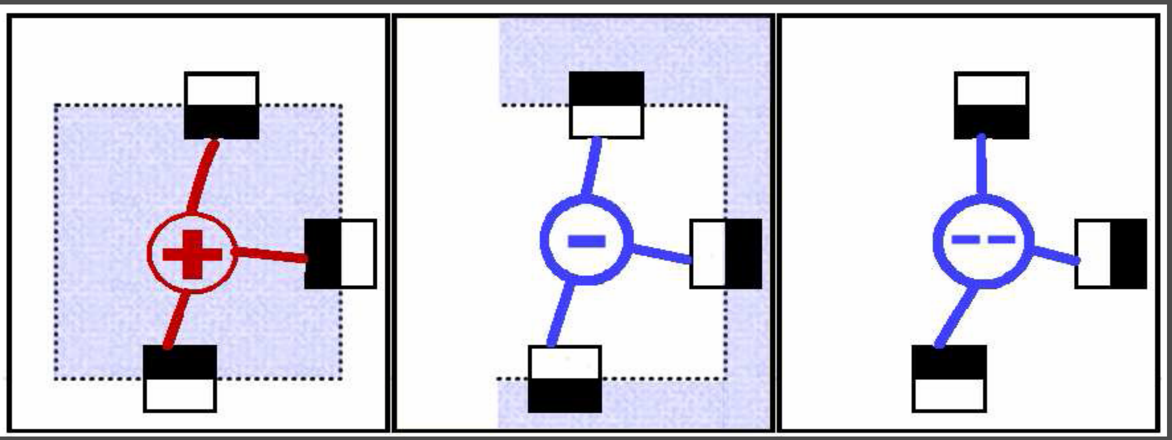
Convex Concave Impossible

Edge location x_i, y_i

Edge orientation α_i

Task 2: Learning rectangular shape

$$W(e_i, e_j) = f(x_i, y_i, \alpha_i; x_j, y_j, \alpha_j)$$



Convex Concave Impossible

Edge location x_i, y_i

Edge orientaion α_i

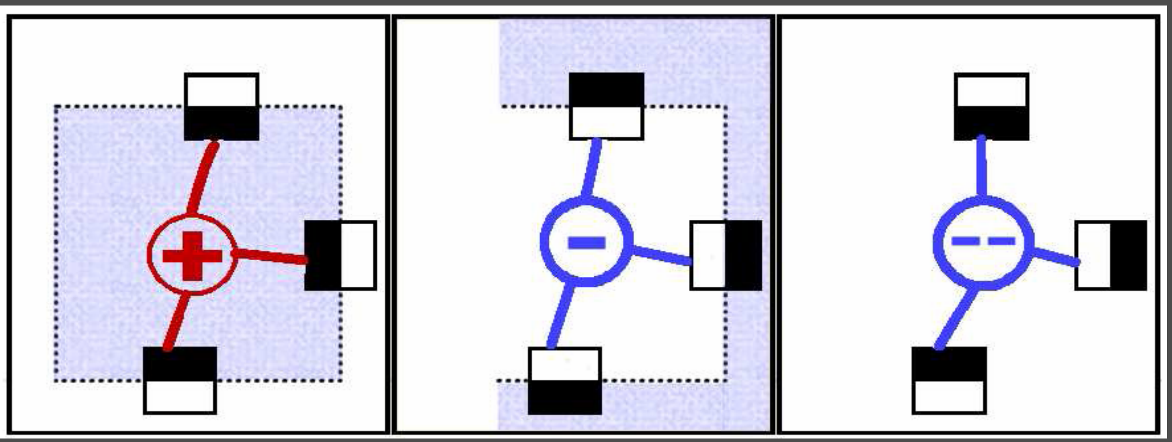
10×10 edge locations

4 edge angles

$$|W| = \underbrace{\left(n_x \cdot n_y \cdot n_\alpha\right)^2}_{} = 160{,}000 \text{ parameters}$$

Task 2: Learning rectangular shape

$$W(e_i, e_j) = f(x_i, y_i, \alpha_i; x_j, y_j, \alpha_j)$$



Convex Concave Impossible

Edge location x_i, y_i

Edge orientaion α_i

Invariance through parameterization : $W(e_i, e_j) = \bar{f}(x_j - x_i, y_j - y_i, \alpha_i, \alpha_j)$

$20 \times 20 \times 4 \times 4 = 6400$ parameters

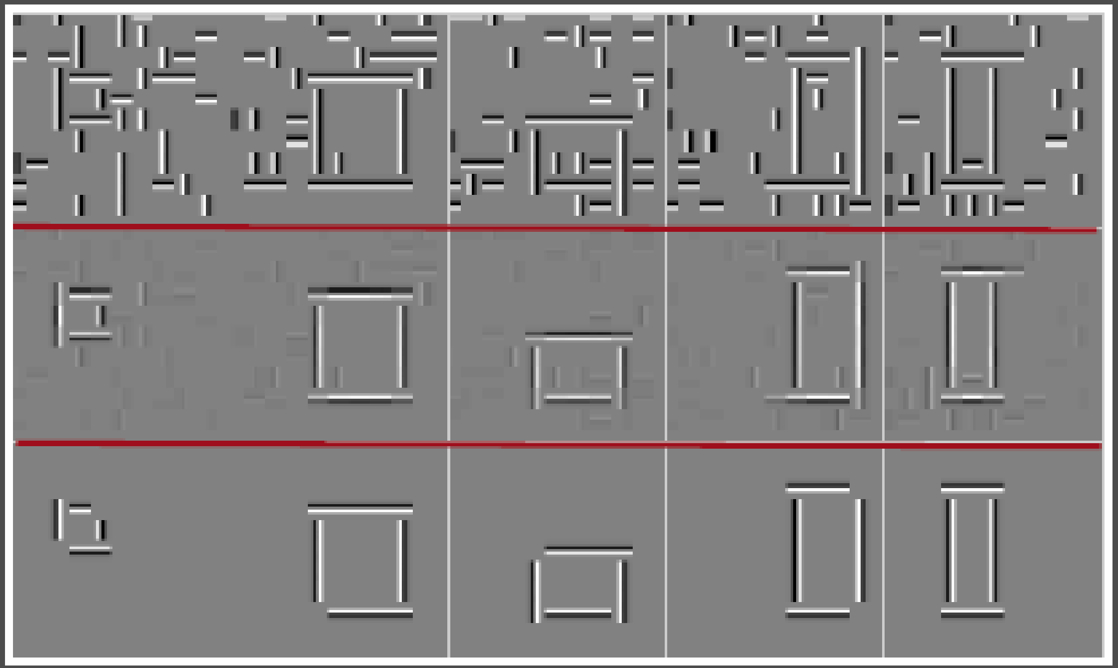
Synthetic case

training

Input edges

Ncut eigenvector
after training

target

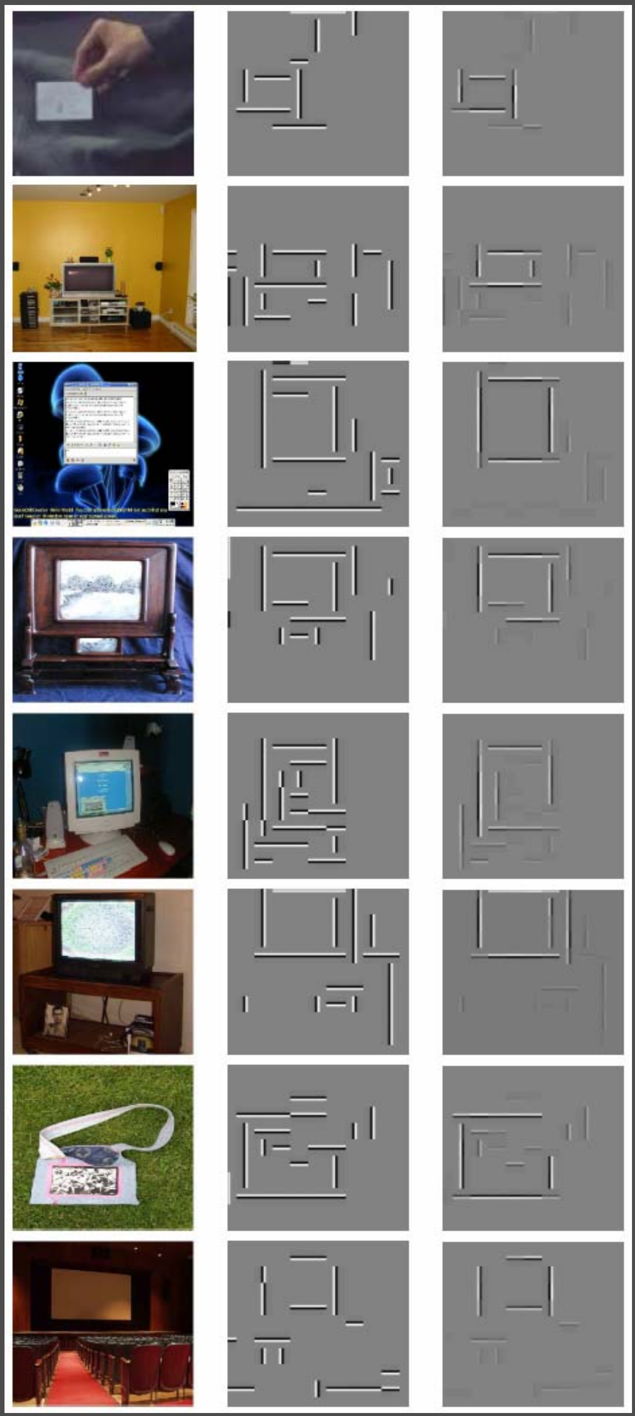


testing

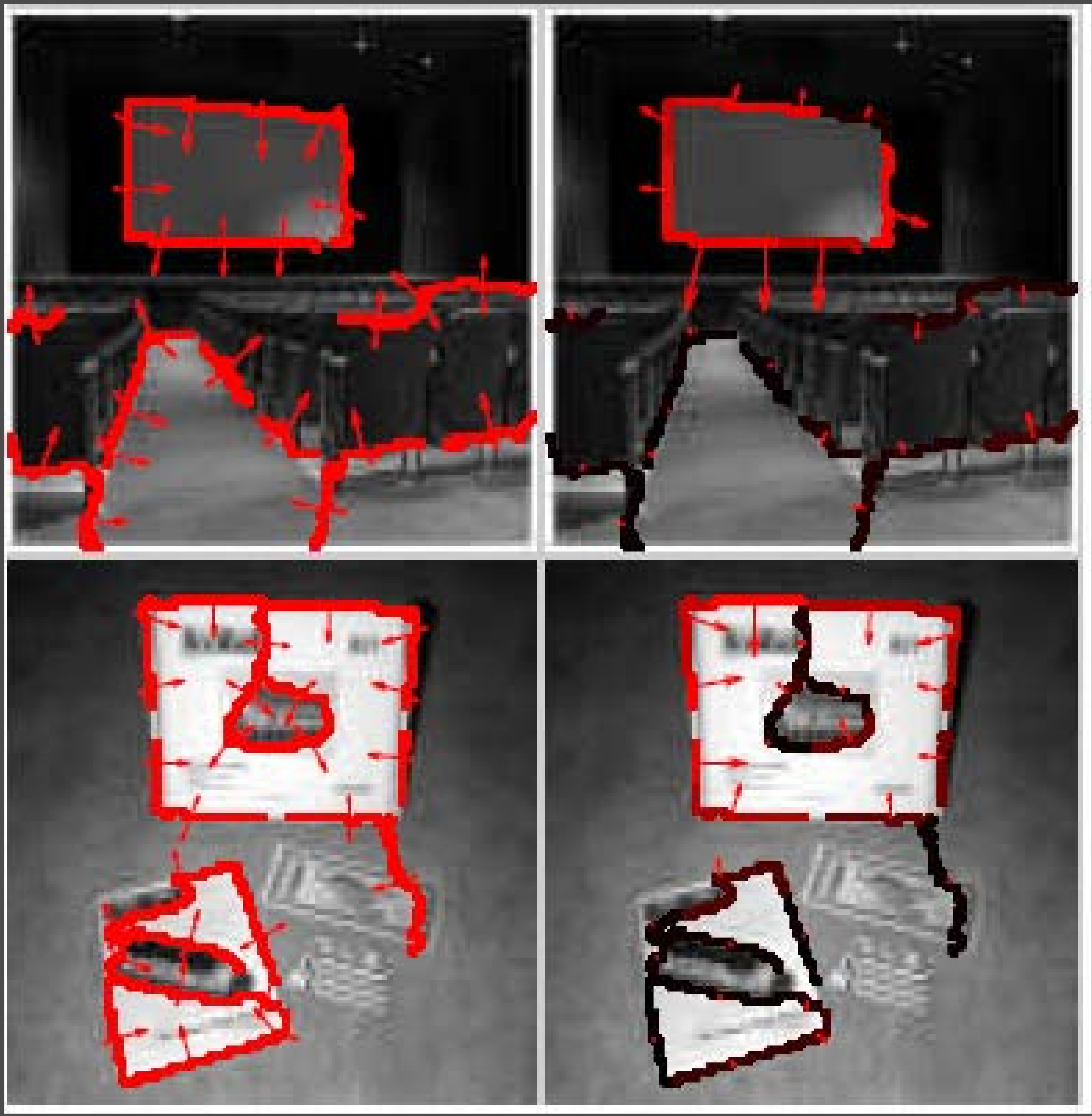
Input edges

Ncut eigenvector

Testing on real images



Learning with precise edges



Comparison of the spectral graph learning schemes

Meila and Shi,
Fowlkes, Martin and Malik

$$\arg\min_W \| D^{-1}W - D^{-1}W(X^*) \|$$

Bach-Jordan

$$\arg\min_W J(W, X^*)$$

Our method

$$\arg\min_W \| X(W) - X^* \|$$

Bach and Jordan (2003)

Learning spectral clustering

Use a differentiable approximation of eigenvectors

Our work

Exact analytical form

computationally efficient solution

Conclusion

- Supervise the un-supervised spectral clustering
- Exact and efficient learning of graph weights using derivatives of eigenvectors