Learning spectral graph segmentation

Timothée Cour
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Nicolas Gogin
Computer Science
Ecole Polytechnique
91128 Palaiseau Cedex, FRANCE

Jianbo Shi
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract

We present a general graph learning algorithm for spectral graph partitioning, that allows direct supervised learning of graph structures using hand labeled training examples. The learning algorithm is based on gradient descent in the space of all feasible graph weights. Computation of the gradient involves finding the derivatives of eigenvectors with respect to the graph weight matrix. We show the derivatives of eigenvectors exist and can be computed in an exact analytical form using the theory of implicit functions. Furthermore, we show for a simple case, the gradient converges exponentially fast. In the image segmentation domain, we demonstrate how to encode top-down high level object prior in a bottom-up shape detection process.

1 INTRODUCTION

Image segmentation and data clustering are two fundamental operations in computer vision and machine learning. Let $\mathbf{I} = \{\mathbf{x_1}, ..., \mathbf{x_n}\}$ be a set of feature vectors representing n image pixels or data points. The image segmentation process partitions pixels into K disjoint groups. In a 2-way segmentation, we seek an output vector $SEG(\mathbf{I}) = \{y_1, ..., y_n\} \in \{0,1\}^n$, such that a segmentation goodness measure is optimized. We defined segmentation as a mapping from $\mathbf{x_i}$ to $y_i \in \{0,1\}$ to purposely hint its potential connection to a supervised learning method we should propose. Our goal is to *teach* the image segmentation through a set of hand labeled training examples. Given a set of image/segmentation pairs $\{\mathbf{I^i}, SEG^*(\mathbf{I^i})\}$, the system will learn to adjust so that the computed segmentation $SEG(\mathbf{I^i})$ is close to $SEG^*(\mathbf{I^i})$. With a *supervised* image segmentation, we are able to encode top-down object familiarity prior in a bottom-up distributed process. In this paper, we will demonstrate a system that can detect and segment rectangular shaped objects in a clutter image background by learning from examples.

To appreciate why learning image segmentation is difficult, we summarize below its basic principles. Segmentation algorithms are defined by the clustering criteria and computational process to optimize it. For example, in the Markov Random Field (MRF) formulation, the criteria is to maximize $P(SEG(\mathbf{x})|\mathbf{x}) = \frac{1}{Z} exp(\sum -f(y_i, y_j|\mathbf{x}))$, where $f(y_i, y_j|\mathbf{x})$, called clique potential, specifies a local measure of grouping pixel i with j. While each $f(y_i, y_j|\mathbf{x})$ can be easily corrupted, the global optimum of $P(SEG(\mathbf{x})|\mathbf{x})$ must balance preferences on all pairs of $f(y_i, y_j|\mathbf{x})$ and therefore is stable. Spectral graph partitioning, such as Normalized Cut (Ncut)[6][5], has been developed as a computationally efficient alternative to MRF. Image segmentation is mapped to a graph partitioning problem, where the graph consists of the pixels/data points as nodes, and the weighted graph edges $W(i,j)$ serve as the equivalent of Clique Potential $f(y_i, y_j|\mathbf{x})$. The global segmentation criterion Ncut seeks a balanced segmentation and grouping of the pixels. Computationally the solution is derived from the eigenvectors of $Wy = \lambda Dy$, where D is the degree matrix. As in the MRF case, the eigenvectors are implicitly related to the input weight matrix W, and are quite insensitive to random perturbation of W.

While global decision process from local feature comparison brings a stable segmentation, it makes the learning segmentation a difficult task. Treating any segmentation learning algorithm as a black box, one must be able to back-trace error on the output of global segmentation to the input local clique potential or pair-wise weight matrix. Since the global decision is only implicitly related to the input, it is hard to explicitly *assign* a blame to a particular clique potential or weight matrix entry. To account for segmentation error on just *one pixel*, we would potentially need to adjust *all* possible pairs of clique potential or weight

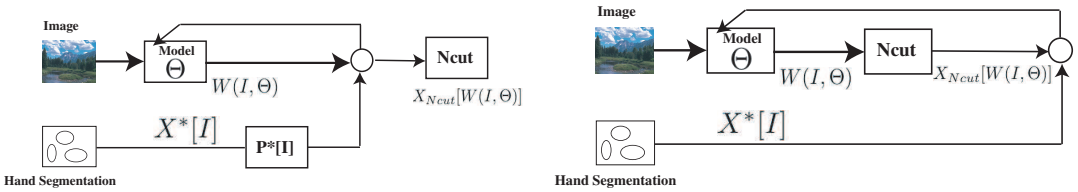


Figure 1: Two alternative algorithms for learning spectral graph partitioning. (A) methods of Meila-Shi[4] optimizes the graph weight $W(I,\Theta)$ by minimizing the KL-divergence between an equivalent random walk matrix $P(I,\Theta)$ and the target $P^*(I,\Theta)$. (B) Our method directly optimize the error on the output Ncut segmentation vector $X_{Ncut}[W(I,\Theta)]$ by gradient descent in the space of all feasible graph weights using explicit computation of the derivatives of eigenvectors.

matrix entries!

Meila-Shi[4] first studied the problem of learning spectral graph cuts with supervised training data. Their proposed algorithm learned the graph weight W_{ij} by minimizing the KL-divergence between an equivalent random walk matrix P_{ij} and the target P^*_{ij} derived from the hand labeled segmentation. However the formulation provides no explicit constraints on the Ncut eigenvector itself. Bach-Jordan[1] formulated a direct optimization of W with respect to its Ncut eigenvector. They transform the *implicit* relationship between W and Ncut eigenvector into an *explicit* one by making a differentiable approximation of eigenvector using power method. The resulting computation of derivatives of eigenvector is however complex and can be computationally unstable.

We present in this paper a direct method for learning spectral graph cut, based on efficient computation of derivatives of Ncut eigenvectors in *exact analytical form*. We show that there is an *explicit* computation that assigns the segmentation error to the input graph weight matrix. This capability allows us to design parameterized graphs that can encode and detect complex objects. The paper is organized as follows. We describe in Sec. 2 the structure of the graph we use for image and shape segmentation. Sec. 3 describes the learning algorithm and its convergence properties. We show our results in Sec. 4.

2 PROBLEM SETUP

We will demonstrate a learnable segmentation algorithm for detecting and segmenting desired object shape such as a rectangle in an image. The shape detection-segmentation process begins with edge detection. Each edge i is parametrized by (x_i, y_i, θ_i), its location and orientation. Denote $F(I) = \{e_1, ..., e_k | e_i = (x_i, y_i, \theta_i)\}$ the set of edges detected for image I, and F the complete set of possible edges detected in all images. The goal of the segmentation algorithm is to group the edges which form a rectangle, and separate them from background edge clutters, as shown in Fig. 11.

While a rectangle is a relatively simple shape, its aspect can be quite flexible with variable aspect ratio in x, y, and variable orientation. Assuming we have quantized the orientation into N_{angle} angles, for an image size of $N_{pixel} \times N_{pixel}$ a brute force method would need to search over $O(N^4_{pixel}N_{angle}$ possible configurations (for a 100×100 image with 10 orientations quantization, we have 1 billion configurations!). One way to avoid this large scale search is to decompose the rectangles into simple local configurations (corners, lines, parallel lines), and combine them by checking their global consistency. This data-driven bottom-up process only needs to check roughly $O(N_{edge}) = O(|F(I)|)$ local configurations (assuming a fixed neighborhood size). The global integration can be carried out in the grouping framework of graph partitioning such as Ncut, which has empirically a running time of $O(N^{1.5}_{edge})$. Furthermore, the decomposition of a shape into local edge relationships also makes the detection more robust to image background clutter.

2.1 LOCAL SHAPE CONFIGURATIONS

We need to define functions on local configuration goodness, with the hope of discriminating rectangular object vs. background. Since we are using oriented edges, we can favor convex configurations and penalize concave or other impossible configurations, as illustrated in Fig. 2. The function that assesses the goodness of a particular configuration is denoted as clique potential: $f(e_1, \ldots, e_K)$ is high only when $(e_1, \ldots, e_K)$ form a familiar configuration. The problem of designing this clique potential can be quite complex in general. For example, consider the case of binary relationships: we need to find a potential function for all possible pairs of edges (e_1, e_2):

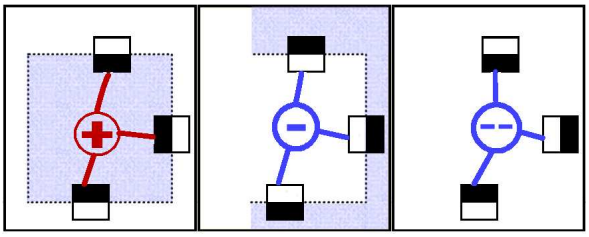


Figure 2: Different oriented edge configurations and associated clique potential. Left: three edges forming a convex object (likely to be found in rectangle shapes). Middle: concave configuration (unlikely in rectangular shapes). Right: impossible configuration (very unlikely to be found in any object).

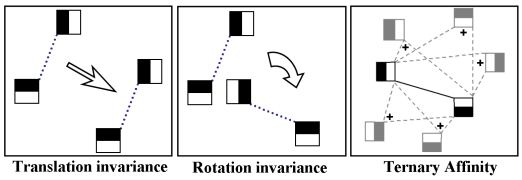


Figure 3: Properties of the clique potential/affinity matrix. Left and middle: translation and rotation invariance, $f(x_1, y_1, \theta_1; x_2, y_2, \theta_2) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$ and, $f(z_1, \theta_1; z_2, \theta_2) = \bar{\bar{f}}((z_2 - z_1)e^{-i\theta_1}, \theta_2 - \theta_1)$. Right: summing up ternary affinities to obtain a binary affinity, $f(e_1, e_2) = \sum_i f_3(e_1, e_2, e_i)$.

$f(e_1, e_2) = f(x_1, y_1, \theta_1, x_2, y_2, \theta_2)$. The function takes 4-dimensional inputs, and even in the simple case of 10x10 possible edge locations, with 4 orientations $\{\pi/2, 2\pi/2, 3\pi/2, 4\pi/2\}$, that makes 160,000 different values to design through learning. To make the learning problem more managable, we use the following parameterization that induces translational invariance:

$$f(x_1, y_1, \theta_1; x_2, y_2, \theta_2) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2) \quad (1)$$

If in addition we also require invariance by rotation, the use of complex numbers comes in handy, with $z = x + iy$ we obtain $f(z_1, \theta_1; z_2, \theta_2) = \bar{\bar{f}}((z_2 - z_1)e^{-i\theta_1}, \theta_2 - \theta_1)$. These invariance properties are illustrated in Fig. 3.

2.2 GLOBAL SHAPE DETECTION FROM LOCAL CONFIGURATIONS

With local edge clique function we could eliminate wrong patterns of edges, retrieve the correct edge orientation when ambiguous, and enhance good configurations. However, there are many ambiguous cases in which local properties are insufficient to decide the foreground/background labeling. Think about a weak edge at object boundary, or a strong clutter edge in the background. A direct thresholding technique would fail here. Another example is provided by Fig. 4, where a local approach would favor the wrong edge orientation. As in the case of image segmentation, local grouping measures need to be aggregated to form a global segmentation decision. We will see in the next section how to formulate this precisely in graph framework, through Spectral Graph Partitioning.

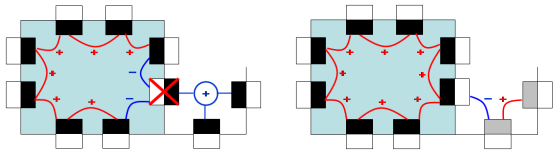


Figure 4: Each edge has 2 hypothesized opposite polarities. We want to inhibit clutter edges and recover correct polarity. Left: local segmentation of edges produces the wrong polarity for one edge (barred), grouping it with clutter. Right: global aggregation of edge affinities yields a correct grouping and inhibits clutter.

2.3 SPECTRAL GRAPH PARTITIONING FORMULATION

Such local relationships between image features are well captured by the notion of graph $G = \langle V, W \rangle$. The graph nodes V consist of the image edge features $F = \{e_i\}$, and the graph edges are the relationships between the edge features with affinity matrix $W \in \mathbb{R}^{n \times n}$ defined by $W_{ij} = f(e_i, e_j)$. Higher-order edge feature relationships can be translated into binary affinities by summing over cliques: $W_{ij} = \sum_{i_1=i, i_2=j, i_3, \ldots, i_K} f(e_{i_1}, \ldots, e_{i_K})$, as illustrated in Fig. 3. We denote $V(I)$ the image edge features, $F(I)$, detected in I; $W(I) = W(V(I), V(I))$, the subgraph affinity induced by image features in I.

Let us recall our goal: we want to partition the graph nodes $V(I)$ into two groups, using an indicator vector X: $X_i = 1$ if detected feature $V(I)_i$ belongs to foreground, and $X_i = -1$ if it belongs to background. The segmentation process should ensure that edge features (nodes) grouped together have high mutual *affinity*, and nodes in different sets have low *affinity*. We will use the Normalized Cuts (Ncut) criterion for the segmentation process. Ncut criterion can be optimized by finding the second generalized eigenvector of $(W(I), D(I))$ ($D(I)$ is the degree matrix of $W(I)$):

$$W(I)X(I) = \lambda_2 D(I)X(I) \quad (2)$$

$X(I)$ is then thresholded to determine the foreground/background labelling. Note that, the solution we obtain for $X(I)$ is an implicit function of the weight matrix $W(I)$, which is defined by the local

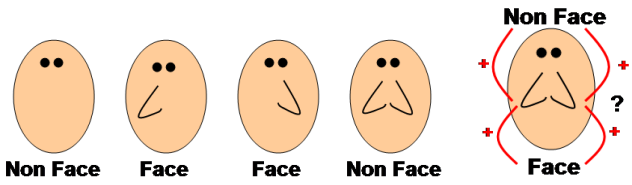


Figure 5: The XOR function. Suppose we have a face detection graph with nodes: Left Nose (LN), Right Nose (RN), Face (F), and Non-Face(NF). The Hebbian learning rule, based on feature coocurrence, would find the following weights: $W(LN,F) = W(RN,F) = W(LN,NF) = W(RN,NF) = \frac{1}{2}$, making it impossible to distinguish between a Face and a Non-Face. The graph learning algorithm we propose does not suffer from this.

clique potentials, $f(e_i, e_j)$. However, its computation is tractable and, as we shall see, we can apply perturbation theory to analyze their effect on the segmentation task. This last point is essential: it means that we can *assign a blame* to the local graph structure, by looking at the global segmentation result. Hence the system is not a black box anymore, we can train it.

3 LEARNING THE GRAPH STRUCTURE

As we have seen, the design of the clique potential is as crucial to the segmentation as it complex. In the Ncut formulation, whether we use a parameterization of the affinity matrix $W(\Theta)$ or a direct representation through its coefficients W_{ij}, real image segmentation tasks will require a large number of parameters to be optimized. Hence the need for a principled algorithm to learn the clique potential.

3.1 WHY ARE SIMPLE LEARNING SCHEMES INSUFFICIENT

One natural idea in learning the graph clique potential is simply to measure the coocurrence of image features accross a set of training images, in accordance to the Hebbian rule. This rule strengthens the weight W_{ij} if feature i and feature j are strongly correlated, according to: $W_{ij} = \sum_I V(I)_i V(I)_j$ in our notation. Though intuitive, this rule is insufficient for our problem. Fig. 5 illustrates a typical situation that the Hebbian rule is unable to handle, namely the XOR boolean function. More generally, the Hebbian rule cannot learn non-linearly separable functions. We have shown in [2][1] that our system does not have this limitation and it could learn XOR.

[1]http://www.seas.upenn.edu/∼timothee/research.html

3.2 PRINCIPLE OF LEARNING

The Maximum Likelihood formulation (ML) tries to adjust the clique potential so that it maximally explains the data (the set of training images). However, this formulation doesn't take into account the graph inference procedure $I \rightarrow X(I)$, as a result, it can produce a probability distribution that cannot be inferenced efficiently. We use a different approach. We adjust the clique potential so that the output of the system gets closer to the desired segmentation. In the following, we assume we are given a set of images I with a target segmentation $X^*(I)$.

3.3 COST FUNCTION FOR LEARNING

Definitions $X_p[W]$, λ_p are the p^{th} largest eigenvector, eigenvalue of $WX = \lambda D_W X$ with $\|X_p[W]\| = 1$ and $D_W = diag(W\mathbf{1})$. $X_p[W]$ is uniquely defined up to polarity, which we disambiguate using a fixed vector Y and forcing $\text{sign}(Y^T X_p[W]) = +1$. This is possible only when $Y^T X_p[W] \neq 0$. We also require λ_p be unique. To satisfy these constraints, we will restrict our attention to weight matrices W in a certain subset $S_n^{2,X^*(I)}$ of symmetric matrices, where $S_n^{p,Y} = \{W \in S_n : W\mathbf{1} > 0, \ker(W - \lambda_p D_W) \not\subset Y^\perp,$ λ_p single$\}$. Note that if $W \in S_n$ and $W\mathbf{1} > 0$, W has probability 1 of being in the feasible space. What's more, $S_n^{2,X^*(I)}$ is open, which implies that any small perturbation of W is allowed.
Define the one-target energy function:

$$\mathcal{E}(W,I) = \frac{1}{2}\|X_2[W(I)] - X^*(I)\|^2, \text{ for } W \in S_n^{2,X^*(I)} \tag{3}$$

The multi-target energy function is defined as $\mathcal{E}(W) = \sum_I \mathcal{E}(W,I)$, for $W \in \cap_I S_n^{2,X^*(I)}$. This error energy function has the following property, which will be useful later on when we try to learn the graph network.

Prop. 3.1 ($\mathcal{E}(W,I)$ has no local minimum) *The single target energy function has all its local minima in $S_n^{2,X^*} \bigcap \{W : \lambda_2(W) \neq -1\}$ equal to the global minimum, 0.*

The proof, in [2], shows that at a critical point, the error vector $X_2 - X^*(I)$ is in the kernel of a certain matrix of rank n-1. This shows in fact that $X_2 - X^*(I)$ is proportional to X_2, which finally leads to $X_2 = X^*(I)$.

3.4 GRADIENT DESCENT ALGORITHM

We minimize the error energy over W by gradient descent: $\Delta W = -\eta \frac{\partial \mathcal{E}}{\partial W} = -\eta \frac{\partial \mathcal{E}}{\partial X_2} \frac{\partial X_2}{\partial W}$. When W is parameterized by Θ, we have instead $\Delta\Theta =$

$-\eta \frac{\partial \mathcal{E}}{\partial X_2} \frac{\partial X_2}{\partial W} \frac{\partial W}{\partial \Theta}$. In the case of rectangle detection, the parameters Θ consist of all the values of the function $f(e_i, e_j) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$, which is a 4 dimensional lookup table.

The main difficulty is to study how the Ncut eigenvector, $X_2[W(\Theta)]$, varies with the graph weight matrix $W(\Theta)$. We will write down a continuous-time PDE describing evolution of the error energy on $X_2[W(\Theta)]$ with respect to Θ. We show that this PDE has an *exact analytical* form, and the resulting PDE *converges*. We have also proved the convergence rate is exponential for a simple case[2]. This result shows that we can minimize the error energy over W or Θ by gradient descent.

Theorem 3.2 (Derivative of Ncut eigenvector) *The map $W \rightarrow (X_p, \lambda_p)$ is C^∞ over $S_n^{p,Y}$, and we can express the derivatives over any C^1 path $W(t)$ as:*

$$\begin{aligned} \frac{dX_p[W(t)]}{dt} &= -(W - \lambda_p D_W)^\dagger \\ &\quad (W' - \lambda_p D'_W - \frac{d\lambda_p}{dt} D_W) X_p \\ \frac{d\lambda_p}{dt} &= \frac{X_p^T (W' - \lambda_p D'_W) X_p}{X_p^T D_W X_p} \end{aligned}$$

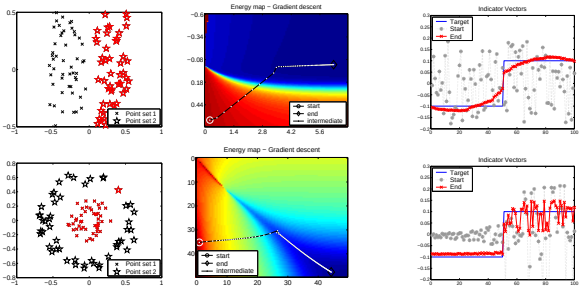
We obtain an analog theorem for the **derivative of standard eigenvectors**, by simply replacing D_W with I_n. The proof in [2] uses the implicit function theorem to show $X_p[W]$ is C^∞, then differentiates $WX_p = \lambda_p D_W X_p$ to obtain $(W - \lambda_p D_W)X'_p + (W' - \lambda_p D'_W - \lambda'_p D_W)X_p = 0$.

Computation of the partial derivatives $\frac{\partial X_2}{\partial W}$ alone requires $O(n^3)$ time because of the pseudo-inverse term $(W - \lambda_p D_W)^\dagger$ in each gradient direction. We remove this bottleneck by first left-multiplying by $\frac{\partial \mathcal{E}}{\partial X_2}$. We introduce $Y = -(W - \lambda_2 D_W)^\dagger (X_2 - X^*(I))$, which we showed how to compute efficiently in [2], and obtain a $O(n^2)$ gradient update rule:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial W}_{ij} &= X_{2,i} Y_j + X_{2,j} Y_i - \lambda_2 (X_{2,i} Y_i + X_{2,j} Y_j) \\ &\quad -\lambda'_{2ij} Y^T D_W X_2 \\ \text{with } \lambda'_{2ij} &= \frac{2X_{2,i}X_{2,j} - \lambda_2 ({X_{2,i}}^2 + {X_{2,j}}^2)}{X_2^T D_W X_2} \end{aligned}$$

3.5 PROPERTIES OF THE LEARNING ALGORITHM

Empirically, we observe that $\mathcal{E}(W(t))$ converges to 0 exponentially fast when $W(t)$ follows the gradient path, even if the number of training examples grows as $O(n)$. We will prove this fact in the case of a single target. The convergence of $\mathcal{E}(W(t))$ however does not imply that of $W(t)$. Indeed, one can construct functions for which gradient descent leads to limit cycle oscillations. The following proposition shows that this cannot happen here.



(A) Point set (B) Error Energy (C) Ncut eigenvector

Figure 6: Learning point set clustering. $W(i,j) = \exp(-\sigma_x(x(i) - x(j))^2) + \exp(-\sigma_y(y(i) - y(j))^2)$. A) 2D layout of the points. The first set is the cross set and the second is the star set. The resulting clustering can be identified by the red and black colors. B) Energy landscape of $\mathcal{E}(\sigma_x, \sigma_y)$, and gradient path taken by Eq.4, $(\dot{\sigma_x}, \dot{\sigma_y}) = -(\frac{\partial \mathcal{E}}{\partial \sigma_x}, \frac{\partial \mathcal{E}}{\partial \sigma_y})$. C) Target vector comparing with initial and final learned Ncut vector. The graph nodes are ordered according to their x-axis position (first row), and to their distance to origin (second row).

Prop. 3.3 (Exponential convergence of $\mathcal{E}(W, I)$) *The 1-target energy PDE $\dot{W} = -\frac{\partial \mathcal{E}}{\partial W}$ either converges to a global energy minimum W_∞, or it escapes any compact $K \subset S_n^{2,X^*}$. In the first case, $\mathcal{E}(W(t)) \rightarrow 0$ exponentially.*

Our proof in [2] shows that $\|\frac{\partial \mathcal{E}}{\partial W}\| \geq b\sqrt{\mathcal{E}}$, leading to the convergence of $W(t)$, and then $\frac{d}{dt}\mathcal{E}(W(t)) \leq -b^2\mathcal{E}$, which shows the exponential decay of $\mathcal{E}(W(t))$.

Pathological non-convergence cases. As stated in the proposition, $W(t)$ could potentially hit the boundary of S_n^{2,X^*}. This arises in 2 pathological cases: 1) $\lambda_2(t) \rightarrow 1$ or $\lambda_2(t) - \lambda_3(t) \rightarrow 0$, and 2) $D_{W(t)}(i,i) \rightarrow 0$ for some i. Note that $X^*(I)^T X_2[W(t)] \rightarrow 0$ cannot happen, because initially $X^*(I)^T X_2[W(t)] > 0$ and $\mathcal{E}(W, I)$ decreases. There are ways to alleviate those problems through weight parameterization, but in practice they only occur when learning a lot of target vectors.

4 RESULTS

4.1 POINT SET CLUSTERING

In experiment 1, figure 6, we examine our spectral graph learning algorithm on simple 2D point set clustering examples. The graph weight matrix $W_{ij} =$

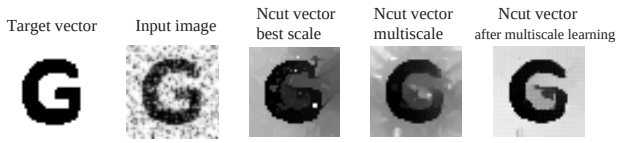


Figure 7: A simple example of multiscale learning. The input image is 40 by 40 and 4 narrow scales are used. The best result (minimum of energy) with one scale is displayed as well as the result with all four scales set up with the same weight, and with the learned weight. The use of multiscale enable to segment correctly the inside of the G. The lowest energy is achieved after learning.

$\exp(-\sigma_x(x_i - x_j)^2) + \exp(-\sigma_y(y_i - y_j)^2)$ has two parameters σ_x, σ_y which we aim to optimize. We update (σ_x, σ_y) as follows:

$$\Delta\sigma_x = -\eta(X - X^*)^T \frac{\partial X}{\partial \sigma_x} \quad (4)$$

$$\Delta\sigma_y = -\eta(X - X^*)^T \frac{\partial X}{\partial \sigma_y} \quad (5)$$

We use directly the derivatives given in Sec. 3.4 with the following expressions of W':

$$\frac{\partial w_{ij}}{\partial \sigma_x} = -(x_i - x_j)^2 \, e^{-\sigma_x (x_i - x_j)^2} \quad (6)$$

$$\frac{\partial w_{ij}}{\partial \sigma_y} = -(y_i - y_j)^2 \, e^{-\sigma_y (y_i - y_j)^2} \quad (7)$$

The experiments on simple clustering show a fast convergence of the gradient descent. We also tested the algorithm with radial distributed point sets.

4.2 MULTISCALE IMAGE SEGMENTATION

In this experiment, we focus on an application of spectral learning in image segmentation. The aim is to provide a powerful tool to find the best scales of edge extraction in Ncut segmentation[3]. Basically the idea of multiscale segmentation is to use several edge scales find a consistent segmentation of the image across scales. The simultaneous use of various scale levels is interesting for complex and big images which mixes textures with sharp and soft contours. In those images, meaningful boundaries may exist at weak contours or between textures that do not rise to edges. Using simultaneously several scales of edges enable to face this problem. The global affinity matrix is the sum of r-affinity matrices at different scales:

$$W_{i,j}^{(I)} = \sum_{r=1}^{k} \alpha_r \exp\left(-\sigma_r \Delta_{i,j}^{(r)}(I)\right) \quad (8)$$

where $\Delta_{i,j}^{(r)}(I)$ is a matrix of the same size as W which expresses a distance measure in a specific scale. Learning on the α coefficients of the scales enables to select

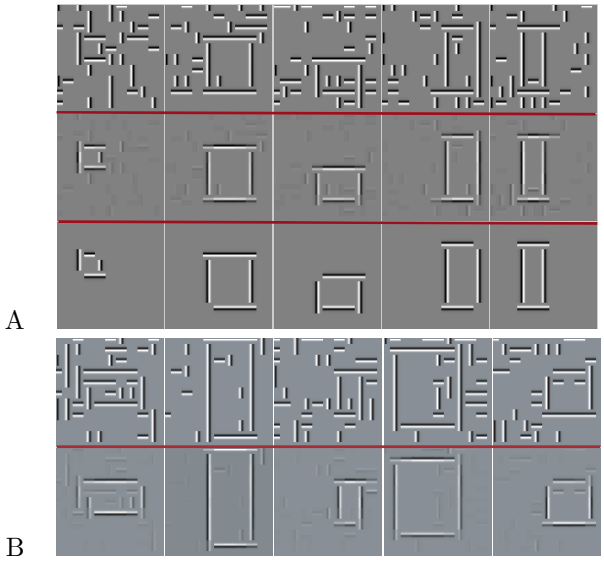


Figure 8: A: Training. Row 1: training input vector. Row 2: Ncut vector after learning. Row 3: target vector. B: Testing. Row 1: testing input vector, Row 2: Ncut vector after learning. For each edge, 2 polarities are hypothesized (only 1 is displayed in Row 1 of Training/Testing). Notice that after learning, not only clutter edges are suppressed but also the correct edge polarities are recovered.

the scales and to set up the weighting coefficients when more than two scales are required. Learning on σ coefficients enables to find the sensitivity to edge strength at a given scale.

The update rules for α_r, σ_r are as following:

$$\Delta\alpha_r = -\eta Y(\frac{\partial W}{\partial \alpha_r} - \lambda\frac{\partial D}{\partial \alpha_r} - \frac{\partial \lambda}{\partial \alpha_r}D)X \quad (9)$$

$$\Delta\sigma_r = -\eta Y(\frac{\partial W}{\partial \sigma_r} - \lambda\frac{\partial D}{\partial \sigma_r} - \frac{\partial \lambda}{\partial \sigma_r}D)X \quad (10)$$

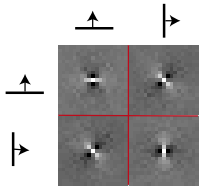


Figure 9: The learned shift-invariant graph clique function $\bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$ with $\theta_1 = 0$, and $\theta_2 = \pi/2$. Each 2D function corresponds to a fixed (θ_1, θ_2) pair. The clique function learns to favor good continuation of the edges with $(\theta_1 = 0, \theta_2 = 0)$, $(\theta_1 = \pi/2, \theta_2 = \pi/2)$, and corner configurations $(\theta_1 = \pi/2, \theta_2 = 0)$, $(\theta_1 = 0, \theta_2 = \pi/2)$

4.3 SHAPE DETECTION

We first generate random rectangles in synthetic images of 100 by 100, see Fig. 8. The edges extracted, e_i are specified by its quantized location (x_i, y_i), orientation θ_i, and polarity p_i. Three graph weight clique potential functions are implemented:

1. unconstrained $f(e_1, e_2) = f(x_1, y_1, \theta_1, x_2, y_2, \theta_2)$
2. translational invariant $f(e_1, e_2) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$
3. translational invariant with ternary clique potential $f(e_1, e_2) = \frac{1}{N} \sum_k g(x_1 - x_k, y_1 - y_k, x_2 - x_k, y_2 - y_k, \theta_1, \theta_2, \theta_k)$.

We apply the following graph learning algorithms to train segmentation algorithm to detect rectangles.

1. Generate random rectangles with one noise-free and one noisy version per example. Generate a random affinity matrix W to start with
2. For each image I, extract edge features from the noisy image to compute subgraph $V(I)$, and compute target $X^*(I)$ from the noise-free image
3. initialize $\mathcal{E} = 0$; for each image I,
   (a) $W(I) = W(\mathrm{V}(I), \mathrm{V}(I))$, using one of the clique potential function $f(e_i, e_j)$ described above.
   (b) Compute $X_2(I)$, second generalized eigenvector of $(W(I), D_{W(I)})$
   (c) Update $W(I)$ with gradient update and propagate updating to each $f(e_i, e_j)$
   (d) Update $\mathcal{E} := \mathcal{E} + \mathcal{E}_I$ with the partial energy $\mathcal{E}_I = \frac{1}{2}||X_2(I) - X^*(I)||^2$
4. Go back to step 3 until $\mathcal{E} < threshold$

Fig.8 display the results of the training and testing using shift-invariant clique function $\bar{f}$. Figure 4.2 shows the shift-invariant clique function learned on a pair of horizontal and vertical edges.

4.4 COMPARISON BETWEEN THE DIFFERENT CLIQUE POTENTIAL FUNCTIONS

We have applied the three methods to random rectangles in 100 by 100 images. The unconstrained affinity matrix has 40000 entries, shift-invariant clique function $\bar{f}$ has 1444 entries and triplet clique function g has 1042568 entries. Several simulations have been run

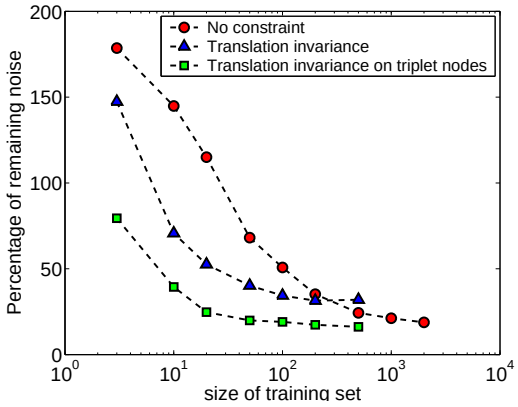


Figure 10: Square shape detection and enhancement. For each simulation, the table indicates the percentage of remaining noise (100 is the initial amount of noise) on 2000 testing examples. These results have been obtained for different sizes of training set, according to three methods: 1- learning on full affinity matrix, 2- invariance by translation, 3- mean on third node with invariance by translation.

for each training set and the result displayed in fig.10 have been averaged. We noticed a very low standard deviation on our training sets.

We see that the best results are achieved with the triplet clique method involving summation of ternary affinities over a third node. With only 20 training examples, an average of 75% of the noise was eliminated in the 2000 testing examples. We achieved the best result with a training data set of 500 squares. 15 iterations were enough to reach energy convergence and it took 4 minutes. This fast convergence can be explained by the averaging on a third node: when the affinity between two nodes is updated, all ternary affinities involving this pair are updated in a single pass. Also, ternary clique potentials carry out a stronger, more robust cue than binary affinities.

4.5 RECTANGLE DETECTION ON REAL IMAGES

This rectangle detection algorithm can be applied directly on real images, fig.12. We just have to adapt the filter parameters to have a good edge extraction. The amount of noise on real images turns out to be frequently below the one we used in the learning step, thus giving those encouraging results.

References

[1] Francis R. Bach and Michael I. Jordan. Learning spectral clustering. *Advances in Neural Information Processing Systems (NIPS)*, 2003.

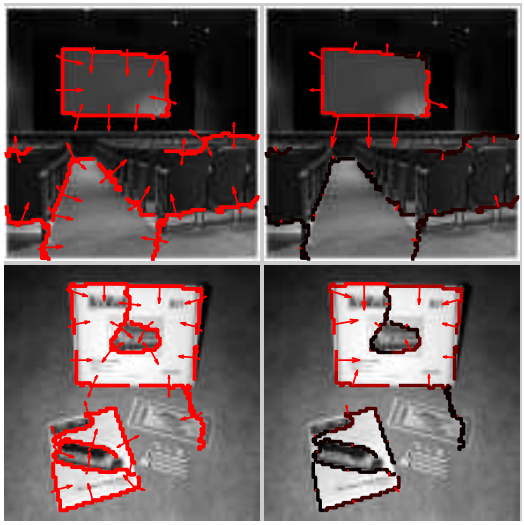


Figure 11: Examples of rectangle detection on real images. Left: edges detected, with arrows indicating orientation (2 opposite polarity hypothesis for each edge). Right: segmentation of foreground edges (in red) versus background clutter edges (in dark).

[2] Timothee Cour and Jianbo Shi. A learnable spectral memory graph for recognition and segmentation. Technical Report MS-CIS-04-12, University of Pennsylvania CIS Technical Reports, Philadelphia, PA, June 2004.

[3] Charless Fowlkes, David Martin, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 26(5):530–549, 2004.

[4] Marina Meila and Jianbo Shi. Learning segmentation with random walk. *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[5] Andrew Y. Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[6] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 22(8):888–905, 2000.

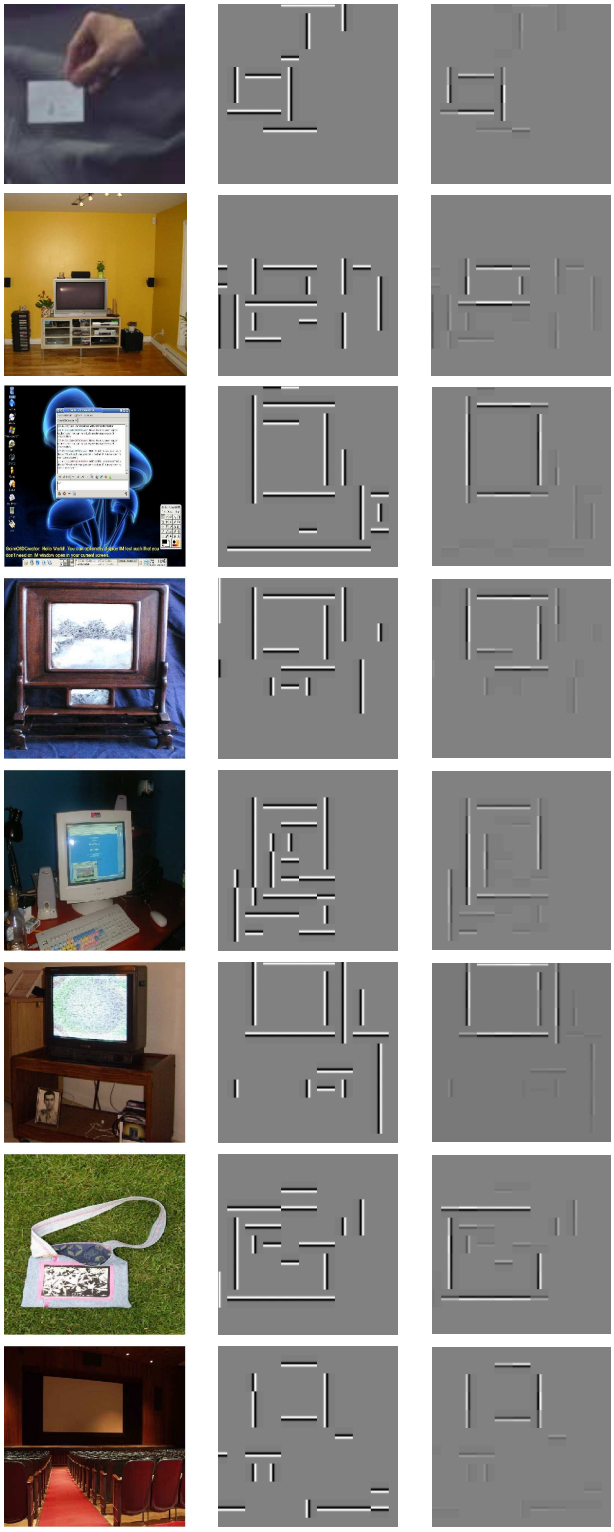


Figure 12: Rectangle detection on real images. First column: image; second column: edges detected; third column: rectangle detection using the graph. Graph weights are learned with random rectangles in background noise.