Graph Based Image Segmentation: A Modern Approach

Jingdong Wang

To my parents, Shihong Wang and Cuihua Ji, and my wife, Nan Li.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of computer vision is producing from images of the external world a description, such as 3D object representation, that is useful to the viewer and not cluttered with irrelevant information. In Marr's vision theory [75], it basically consists of three stages, primal sketch, $2\frac{1}{2}$-D sketch, and 3-D model representation, to derive shape information from images. Image segmentation, which is a process that divides the image into regions that were meaningful, has always been thought as one of the fundamental problems and also a key part in the stage of $2\frac{1}{2}$-D sketch.

Image segmentation is very related to *perceptual organization*, which was created by experimental psychologists (Gestaltists) in the 19th and 20th century. The Gestalt theory, founded by Max Wertheimer [119], identified a set of *laws of grouping* under ideal simplistic settings for artificial stimuli: elements are structured into groups sharing a common feature, e.g. intensity, color, or motion.

Earlier approaches to image segmentation can essentially be categorized into three groups: 1) cluster the low level feature, such as histogram thresholding [81], and k-means/k-centroid [31, 117], mixture of Gaussians (MoG) [23], mixture of probability principal component Analyzers (MPPCA) [33]; 2) edge linking such as dynamic programming [77], relaxation approach [82], and saliency network [109]; or 3) region operation, such as region splitting and merging [54, 55], region growing [22, 76, 1, 112], and region competition [136]. Afterwards, generic priors, such as boundary smoothness, are incorporated into image segmentation: stochastic completion [120] active contours [62, 24], level set and variational methods [96]. The recent developments, top-down approaches, make use of the object specific priors, such as fragment based prior[17, 18, 16, 70], shape based prior [42, 26], and appearance based prior [53, 107, 106, 128].

Recent research on image segmentation has seen great interest in probabilistic graph models (PGMs), or Markov Random Fields (MRFs). Such an approach formulates image segmentation as the Maximum a Posteriori (MAP, or posterior mode) solution to an MRF, which is also equivalent

to partitioning (or labeling) a graph into connected subgraphs. The most representative methods are spectral graph partitioning, combinatorial graph cuts, and statistical inference.

Spectral graph partitioning in essence makes the use of the eigenvectors of the graph Laplacian or its variants to partition the graph. As a hot area of mathematics, it can be traced back to [28, 37, 40], later contributed by computer scientists [3, 15, 99, 100], and now repopular in computer science [85, 51, 25, 30]. It becomes hot in computer vision after normalized cut was invented for image segmentation in [98], and also well studied in machine learning communities [123, 36, 116].

Combinatorial graph cuts is well studied in discrete combinatorial optimization, and matured after the proof of its equivalence to max-flow [41]. Its application to binary segmentation [50] shows its power in computer vision. Powerful performances in image segmentation [56, 19] and stereo matching [91, 66] populated its applications to machine learning [12, 13], computer vision and graphics [67].

Statistical inference is popular in probabilistic graphical models [44]. In computer vision, it was first appeared in [47] using Gibbs sampling for optimization. An alternative sampling approach, the Swendsen-Wang cuts originated in [102], was introduced for image segmentation in [4]. Loopy belief propagation, a variational method, also obtained great successes in vision [84, 78, 45, 59, 43, 38], its generalization [126], and its tree variants in [113, 52, 64] are also proposed. Other variational methods include mean fields [59]. The recent developments, inference by solving a linear system [132, 139], can be categorized into the framework of variational methods.

1.1 Contribution

The research on graph based image segmentation involves the following points:

- Graph structure construction: how to construct or learn a graph to model the relation between the data points.
- Graph weight setting: how to adaptively set the weights from the data points or how to learn the weights from other image segmentation.
- Graph partition inference: how to optimize the objective function to partition the graph.

The three issues are usually interrelated. Basically weight learning and partition inference are dependent on the graph structure. In the probability theory, the graph structure is used to define the prior, called graph based prior.

In this thesis, we focus on graph structure and prior design to contribute to graph based image segmentation

- Pairwise graph: We investigate the key issue, weight setting, in the conventional pairwise graph based image segmentation. I will present a well-designed real system for joint 3D clustering and 2D image segmentation, which makes good use of the states of the art of both spectral graph partitioning and combinatorial graph cuts.

- Hypergraph: A hypergraph is a generalization of conventional pairwise graph, which can model multiple-wise relation to capture high-order statistics. We propose a novel biased hypergraph partitioning approach, called Linear Neighborhood Propagation. A continuous relaxation algorithm is used to solve this problem and leads to a linear systerm.

- Tree: We propose a novel tree partitioning method based on the normalized cut criterion. This method intends to beat the drawbacks of spectral graph partitioning: expensive computational cost and unsatisfied performance due to continuous relaxation and discretization. Our approach runs in linear time, and outperforms normalized cut [98], the state of the art in spectral graph partitioning.

1.2 Organization

This thesis is organized as follows. Chapter 2 outlines the works in this thesis and the research roadmap. In Chapter 3, the joint segmentation framework is given to make the best use of the current two successful pairwise graph partitioning methods. Chapter 4 presents an alternative image segmentation approach based on biased hypergraph partitioning, called Linear Neighborhood Propagation. In Chapter 5, the tree partitioning based on the normalized cut criterion is proposed, and its variants are also presented. Chapter 6 concludes this thesis and discusses some future works.

Chapter 2

Graph Based Prior and Partitioning

Suppose there is an input data set $\mathcal{X} = \{\mathbf{x}_i\}_{i\in\mathcal{N}}$ and a target data set $\mathcal{Y} = \{\bar{\mathbf{y}}_i\}_{i\in\mathcal{L}}$ with $\mathcal{L} \subseteq \mathcal{N} \equiv \{i\}_{i=1}^{N}$, the goal is to find a mapping

$$f : \mathbf{x} \mapsto \mathbf{y}. \tag{2.1}$$

Basically this task can be formulated as a fitting problem:

$$E(\{\mathbf{y}_i\}_{i\in\mathcal{L}}) = \sum_{i\in\mathcal{L}} D_i(f(\mathbf{x}_i)), \tag{2.2}$$

where D_i is a function to measure the penalty that $\mathbf{x}_i$ is mapped to $f(\mathbf{x}_i)$, and the typical function of D_i is the norm function, such as 1-norm ($||\cdot||_1$), 2-norm ($||\cdot||_2$) and so on. This task is particularly a *least square* problem when $D_i(f(\mathbf{x}_i)) = ||f(\mathbf{x}_i) - \bar{\mathbf{y}}_i||_2$ (and $f(\mathbf{x})$ is a parametric function of $\mathbf{x}$).

The fitting problem is sometimes ill-posed (ill-defined). So a regularization term is introduced to well define the problem. Typical regularization terms include the regularization on the function f, or on the domain $\{\mathbf{y}_i = f(\mathbf{x}_i)\}_{i\in\mathcal{N}}$.

In this thesis, we focus on graph regularization, which defines the regularization $R_j(e_j)$ on the edge set $\mathcal{E} = \{e_j\}_{j=1}^{J}$ with $e_j \subseteq \mathcal{N}$. The overall objective function is written as

$$E(\mathbf{y}_1, \cdots, \mathbf{y}_N) = \sum_{i\in\mathcal{L}} D_i(f(\mathbf{x}_i)) + \sum_{j\in\mathcal{E}} R_j(e_j). \tag{2.3}$$

Considering the exponential function $\exp\{-E(\mathbf{y}_1, \cdots, \mathbf{y}_N)\}$, we can reach a probabilistic formulation

$$\begin{aligned} P(\mathbf{y}_1, \cdots, \mathbf{y}_N|\mathbf{x}_1, \cdots, \mathbf{x}_N) &\propto P(\mathbf{x}_1, \cdots, \mathbf{x}_N|\mathbf{y}_1, \cdots, \mathbf{y}_N)P(\mathbf{y}_1, \cdots, \mathbf{y}_N) \\ &= \exp(-\sum_{i\in\mathcal{L}} D_i(f(\mathbf{x}_i)))\exp(-\sum_{j\in\mathcal{E}} R_j(e_j)), \end{aligned} \tag{2.4}$$

where the first term of the right hand side is the likelihood, and the second term is the prior. The prior is defined on a graph structure, and called graph based prior.

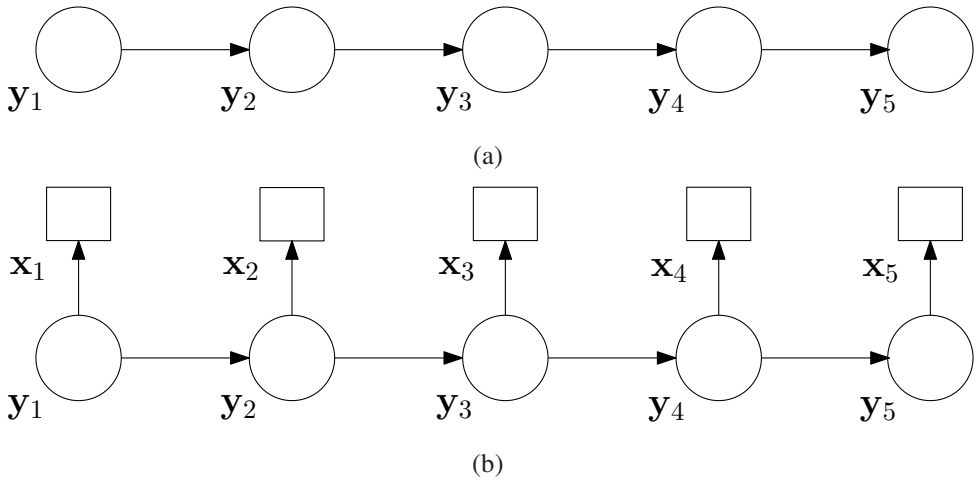


Figure 2.1: Chain based prior. (a) shows the chain structure, and (b) involves the observation model, *i.e*. likelihood.

2.1 Graph Structure

A graph structure can be used to illustrate the prior on the subsets. In the following, we list several graph structures, which can be used to define the prior.

2.1.1 Chain

The simplest graph structure is a chain structure as shown in Fig. 2.1. In the chain structure, the subsets for the prior are defined as $\{e_j\}_{j=1}^{N-1}$ with $e_j = \{\mathbf{y}_j, \mathbf{y}_{j+1}\}$, where e_j is called an edge. Then the prior on edge e_j is defined as a conditional probability $P(\mathbf{y}_{j+1}|\mathbf{y}_j)$. Then the overall objective function, *i.e*. the joint probability, is written as

$$P(\mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \prod_{i=1}^{N} P(\mathbf{x}_i|\mathbf{y}_i) \prod_{j=0}^{N-1} P(\mathbf{y}_{i+1}|\mathbf{y}_i), \tag{2.5}$$

where $P(\mathbf{y}_1, \cdots, \mathbf{y}_N) = \prod_{j=0}^{N-1} P(\mathbf{y}_{i+1}|\mathbf{y}_i)$ is the prior. The joint probability also corresponds to a directed graphical model, such as hidden Markov models, or linear Gaussian models with both likelihood and prior models being Gaussian models. The *Viterbi* algorithm, equivalent to the *dynamic programming* approach, can be used to find the maximum of the joint probability. The marginal probability $P(\mathbf{y}_i|\mathbf{x}_1, \cdots, \mathbf{x}_N)$ can be solved by the *forward-backward* algorithm.

2.1.2 Graph

Another popular structure is just a graph with pairwise edges and usually contains cycles. An example is shown in Fig. 2.2. The general probability model on a graph, called probabilistic graphical models,

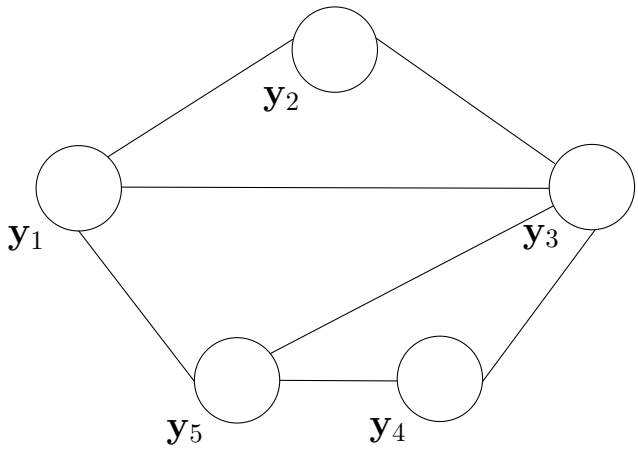


Figure 2.2: Graph based prior. Here we omit the optional observation nodes.

consists of *Markov random fields* (with undirected graph) or *belief networks* (with directed graph). In some cases, those two models can be equivalently transformed. In this thesis, we focus on Markov random fields (MRFs). The general formulation for MRFs can be written as

$$P(\mathbf{y}_1, \cdots, \mathbf{y}_N) \propto \prod_{c \in \mathcal{C}} P_c(\mathbf{y}_c), \tag{2.6}$$

where $\mathbf{x}_c$ is a variable set on the maximal clique $c \in \mathcal{C}$. A clique is defined as a subgraph in which there exists a link between all pairs of nodes in the subset. A maximal clique is a clique that is not a subgraph of any other clique.

In this thesis, we address the labeling problem, and the probabilistic formulation is as the following

$$P(\mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{x}_1, \cdots, \mathbf{x}_N) \propto \prod_{i=1}^{N} P_o(\mathbf{y}_i, \mathbf{x}_i) \prod_{c \in \mathcal{C}} P_c(\mathbf{x}_c), \tag{2.7}$$

where $P_o(\mathbf{y}_i, \mathbf{x}_i) = \exp\left(-D_i(f(\mathbf{x}_i)\right)$. The typical research topics on MRFs include the inference of $\mathbf{y}$, the structure design of graph, and the parameter learning. The popular inference algorithms include combinatorial graph cuts, variational approaches, and relaxation methods. For the design of graph structure, most of current approaches are based on the pairwise cyclic graph, which leads to the difficulty of inference.

2.1.3 Tree

The tree structure is halfway between the graph structure and the chain structure. It is defined as an *acyclic but connected* graph as shown in Fig. 2.3. (An *acyclic and unconnected* graph is instead called a forest.) From the definition, a chain structure is a degraded tree in which a node can connect at most

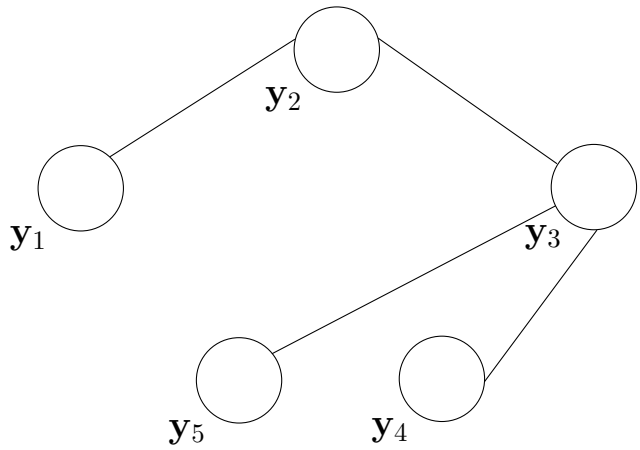


Figure 2.3: Tree based prior. Here we omit the optional observation nodes.

two nodes. For a tree structure, the maximal clique only consists of two nodes, *i.e.*, each edge forms a maximal clique. Hence, the corresponding prior is followed by

$$P(\mathbf{y}_1, \cdots, \mathbf{y}_N) \;=\; \prod_{j=1}^{N} P(\mathbf{y}_j|\mathbf{y}_{p_j}), \tag{2.8}$$

where p_j represents the parent node of node j. In the pre-mentioned chain structure, $p_j = j-1$. Since it is between the graph and chain structure, the tree retains their partial properties. The maximum of joint probability is still efficiently obtained by the *dynamic programming* approach, while the marginal maximum can not directly be obtained by the similar method in the chain structure and is obtained by the sum-produce belief propagation algorithm.

2.1.4 Hypergraph

A hypergraph is a generalization of a graph. One example and its factor graph representation are shown in Figs. 2.4(a) and 2.4(b). A graph can formally be defined $(\mathcal{V}, \mathcal{E})$ with $\mathcal{E} = \{e_k\}$ and $e_k = \{i, j\}$ is a subset of pair nodes, called pairwise edge. A hypergraph generalizes a pairwise edge to a hyperedge (multiple-wise edge) e such that the cardinality of a hyperedge, $|e|$, may be larger than 2. A hypergraph can be represented by a factor graph (a bipartite graph) which consists of data nodes $\mathcal{V}_d$, function nodes $\mathcal{V}_f$ and the edges $\mathcal{E}' = \{(i, j)\}$ with $i \in \mathcal{V}_d$ and $j \in \mathcal{V}_f$ between data nodes and function nodes. Each function node j corresponds to a hyperedge e, and if $i \in e$, $(i, j) \in \mathcal{E}'$. Then the probabilistic formulation is as the following

$$P(\mathbf{y}_1, \cdots, \mathbf{y}_N) \propto \prod_{e} P_e(\mathbf{x}_e), \tag{2.9}$$

where $\mathbf{x}_e$ is the variables on hyperedge e.

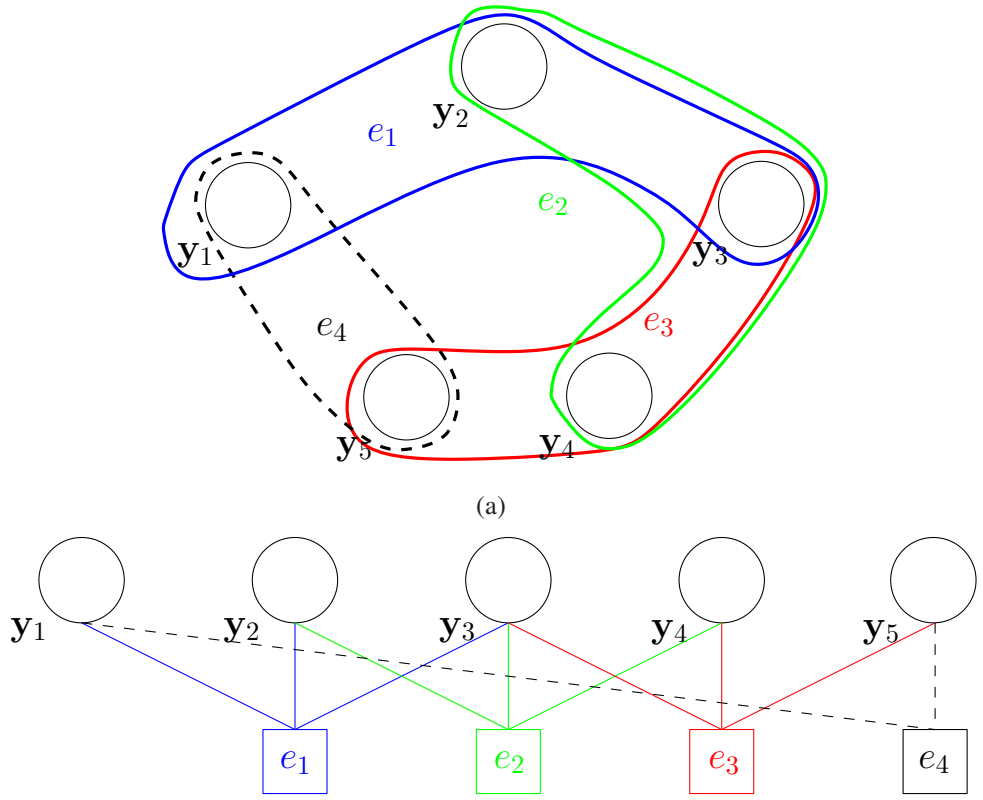


Figure 2.4: Hypergraph based prior. Here we omit the optional observation nodes. In (a), the nodes in different closed curve form a corresponding hyperedge. For instance, nodes $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_3$ in the closed blue curve form a hyperedge e_1. (b) shows the factor graph representation of a hypergraph shown in (a).

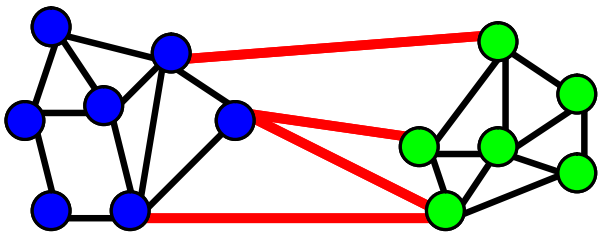


Figure 2.5: The illustration of unbiased graph Partitioning.

2.2 Graph Partitioning

The labeling problem of graph nodes, *i.e.* $\mathbf{y}$ is a discrete partition index, is in some sense equivalent to a graph partitioning problem. The graph partitioning is called *unbiased graph partitioning* if only the prior is used for graph partitioning possibly with the latent constraint that the graph must be separated into at least two subgraphs (or equivalently at least two nodes have different labels). The most representative method for unbiased graph partitioning is spectral graph partitioning based on the normalized cut criterion.

The *biased graph partitioning* is another problem, which involves both terms. The biased information, encoded in the likelihood term, may be supplied from the interactivity or from the training data. Two of the most representative methods are combinatorial graph cuts and continuous relaxation based method. In addition, statistical inference algorithms, such as Swendsen-Wang cuts, the variants of belief propagation and expectation propagation, are also utilized.

2.2.1 Spectral Graph Partitioning

To partition a graph without any biases, we can remove some edges to get several isolated connected subgraph under some certain criterion. Generally, the problem is NP-hard. Some relaxing techniques, such as spectral relaxation and semi-definite relaxation, are usually used to find an approximate solution. Here we only review the most popular method, *i.e.* spectral graph partitioning based on normalized cut criterion.

Normalized cuts aims to minimize the normalized cut value:

$$NC(\mathcal{A},\mathcal{B}) = \frac{cut(\mathcal{A},\mathcal{B})}{assoc(\mathcal{A},\mathcal{V})} + \frac{cut(\mathcal{B},\mathcal{A})}{assoc(\mathcal{B},\mathcal{V})}, \tag{2.10}$$

where $\mathcal{V}$ is the node set, and $\mathcal{A}$ and $\mathcal{B}$ are subsets with $\mathcal{A} \cup \mathcal{B} = \mathcal{V}$, $\mathcal{A} \cap \mathcal{B} = \emptyset$, $cut(\mathcal{A},\mathcal{B}) = \sum_{i\in\mathcal{A},j\in\mathcal{B}} w(i,j)$, and the association is defined as $assoc(\mathcal{A},\mathcal{B}) = \sum_{i\in\mathcal{A},j\in\mathcal{B}} w(i,j)$ with $w(i,j)$ is the affinity on edge (i,j). Considering the graph shown in Fig. 2.5, $cut(\mathcal{A},\mathcal{B})$ is the summation of the affinities on the red edges, $assoc(\mathcal{A},\mathcal{V}) = assoc(\mathcal{A},\mathcal{A}) + cut(\mathcal{A},\mathcal{B})$, $assoc(\mathcal{A},\mathcal{A})$ is double of

the summation of the affinities on the edges in $\mathcal{A}$.

By relaxation trick, Minimizing *normalized cuts* can be formulated as a generalized eigenvalue system $(\mathbf{D}-\mathbf{W})\mathbf{q} = \lambda\mathbf{D}\mathbf{q}$, where $\mathbf{W}$ is an affinity matrix with each entry equal to the weight $w(i,j)$ on the corresponding edge, and $\mathbf{D}$ is a diagonal degree matrix with the diagonal entry $\mathbf{D}_{ii} = \sum_j w(i,j)$. With continuous vector $\mathbf{q}$, $v_i \in A$ if $\mathbf{q}_i > 0$ and $v_i \in B$ otherwise. The generalized eigenvalue system can be written as a standard eigenvalue system:

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z}. \tag{2.11}$$

The matrix $\bar{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ is called Normalized graph Laplacian matrix, note that $\mathbf{L} = \mathbf{D}-\mathbf{W}$ is called graph Laplacian matrix. This method can easily be generalized to k-way partitioning.

2.2.2 Combinatorial Graph Cuts

In combinatorial optimization, the s-t mincut problem is a special graph partitioning problem, in which two nodes, s and t, must be separated into two different subsets. The two-label pairwise MRF can also be formulated as an s-t mincut problem. We augment two nodes, s and t, to the graph representing the prior, such that the two nodes represent the two labels and the augmented edges between label nodes and original nodes indicate the compatibility.

For two-label cases, the optimal solution to biased graph partitioning can be obtained using the max-flow algorithm as in [12]. In multi-label cases, it is NP hard. α-expansion graph cuts or $\alpha\beta$-swap graph cuts can approximately find the suboptimal solution [21]. Theoretically it is much slow. In practice, when $E = O(V)$ with E the edge number and V the node number, the experiments show that the time complexity has less possibilities to reach the worse case V^3.

2.2.3 Continuous Relaxation Optimization

Continuous relaxation optimization methods are popular in semi-supervised learning, which is motivated by practical requirements.

Introduction to Semi-supervised Learning In some practical applications of pattern classification and data mining, one often faces a lack of sufficient labeled data, since labeling often requires expensive human labor and much time. However, in many cases, large numbers of unlabeled data can be far easier to obtain. For example, in text classification, one may have an easy access to a large database of documents (*e.g.* by crawling the web), but only a small part of them are classified manually.

Consequently, semi-supervised learning methods, which aim to learn from partially labeled data, are proposed (for a detailed literature survey see [137]). In general, these methods can be categorized

into two classes: *transductive learning* and *inductive learning*. The goal of transductive learning is only to estimate the labels of the given unlabeled data, whereas inductive learning tries to induce a decision function which has a low error rate on the whole sample space.

The key to semi-supervised learning problems is the consistency [132] (also called *cluster assumption* [27]): 1) nearby points are likely to have the same label; 2) points on the same structure (such as a cluster or a submanifold) are likely to have the same label. Note that the first assumption is local, while the second one is global. The cluster assumption implies us to consider both local and global information contained in the data set during learning.

There are many semi-supervised learning methods, such as Expectation-Maximization with generative mixture models [80, 46], self-training [88], co-training [14], transductive support vector machine [57, 58], and graph based methods.

Relaxation Continuous Optimization Many graph-based methods essentially estimate a function f on the graph such that the function f has two properties: 1) it should be as possible as close to (or the same as) the given label y_L on the labeled nodes, and 2) it should be smooth on the whole graph. The former corresponds to a loss function, the latter can be expressed as a regularizer (smooth function). Almost all the previous are much similar. They differ slightly in cost function and regularization.

Basically, the cost function is as the following:

$$E_c = \sum_{i \in \mathcal{N} \setminus \bar{\mathcal{L}}} (\mathbf{y}_i - \bar{\mathbf{y}}_i)^2 + \lambda \sum_{i \in \bar{\mathcal{L}}} (\mathbf{y}_i - \bar{\mathbf{y}}_i)^2, \tag{2.12}$$

Some methods only used the first term of right hand side such as [139]. Some methods also used the second term such as [132]. In [132], they prefer unbias labeling of unlabeled data by setting $\bar{\mathbf{y}}_i = 0$ when $i \in \mathcal{N} \setminus \mathcal{L}$. Some other got an initial bias $\mathbf{y}_i | \bar{L}$ by estimating their likelihoods.

Often, the regularization is based on Laplacian and its variants. In [7, 139], the combinatorial graph Laplacian $\mathbf{\Delta}$ based regularizer is adopted:

$$E_r = \sum_{i,j} w_{ij} (\mathbf{y}_i - \mathbf{y}_i)^2 = \mathbf{y}^T \mathbf{\Delta} \mathbf{y}. \tag{2.13}$$

In [7], it is mentioned that $\mathbf{y}^T \mathbf{\Delta}^p \mathbf{y}$ with some integer p is used as the regularizer, unfortunately they did not analyze the choice of p. In [132], the normalized Laplacian is used:

$$E_r = \sum_{i,j} w_{ij} (\frac{\mathbf{y}_i}{\sqrt{\mathbf{D}_{ii}}} - \frac{\mathbf{y}_j}{\sqrt{\mathbf{D}_{jj}}}) = \mathbf{y}^T \mathbf{D}^{-\frac{1}{2}} \mathbf{\Delta} \mathbf{D}^{\frac{1}{2}} \mathbf{y}. \tag{2.14}$$

In semi-supervised classification, it aims to find $\mathbf{y}$ such that $E_c + E_r$ is minimized. The function $\mathbf{y}$ is integer valued, and hence the task is in essence a discrete optimization problem. The relaxation

methods relax the discrete problem to a continuous convex optimization problem. Relaxation approaches are faster both in theory and in practice. Usually it can reduce to solving a sparse linear system:

$$\mathbf{A}\mathbf{y} = \mathbf{b}. \tag{2.15}$$

There are many methods to solve a linear system. The time complexity of directly solving is $O(VE)$, and hence is $O(V^2)$ since $E = O(V)$ in practice. The iterative method can be also used to solve the linear system. The basic step is the multiplication of matrix and vector with the time complexity of $O(E) = O(V)$. Therefore the total complexity is $O(TV)$, where T is the iteration number.

This method can be generalized to k-class cases. We can use the one-to-other method, which results in solving k linear systems:

$$\mathbf{A}\mathbf{Y} = \mathbf{B}, \tag{2.16}$$

where $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_k]$, $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_k]$.

2.3 Research Roadmap

In this thesis, I will address the graph partitioning (node labeling) problem and its applications to image segmentation. More precisely, my research will focus on graph construction (including graph structure and prior setting) and its optimization.

2.3.1 Graph

In real applications of pairwise graphs (see Fig. 2.2), at least two issues are required to handle carefully: 1) designing the edge to make the best use of data information 2) proposing an effective procedure to optimize the graph partition.

In Chapter 3, I will present a well-designed real system for joint 3D clustering and 2D image segmentation, which makes good use of the states of the art of both unbiased graph partitioning (spectral graph partitioning) and biased graph partitioning (combinatorial graph cuts). We demonstrated this system in 3D modeling from multiple view images.

2.3.2 Hypergraph

We propose a biased hypergraph partitioning approach to consider multiple-wise relation of data points (or equivalently capture high order information, see Fig. 2.4) instead of only considering pairwise relation in the traditional graph based methods. It is specially invented for biased graph partitioning. In this method, we follow the continuous relaxation based method to transform the problem

into a linear system. Similar to the most relevant two methods [132, 139], our approach can have a differential geometry interpretation, but our approach is related to a biharmonic (bilaplacian) view instead of the harmonic (Laplacian) perspective in [132, 139]. This novel approach is presented in Chapter 4.

2.3.3 Tree

In unbiased graph partitioning spectral graph partitioning has attracted great interest. However, it suffers from at least two drawbacks: expensive computational cost and unpredictable approximation. Those two drawbacks are essentially due to the cycle in the graph. In Chapter 5, we present a novel method to break the cycle by fitting a tree distribution to the graph distribution. For example, the tree in Fig. 2.3 can be used to approximate the graph in Fig. 2.2. We propose a *normalized tree partitioning* following the tree fitting step for the original graph partitioning problem. Our approach is superior over the state of the art in the senses of both computational cost and performance.

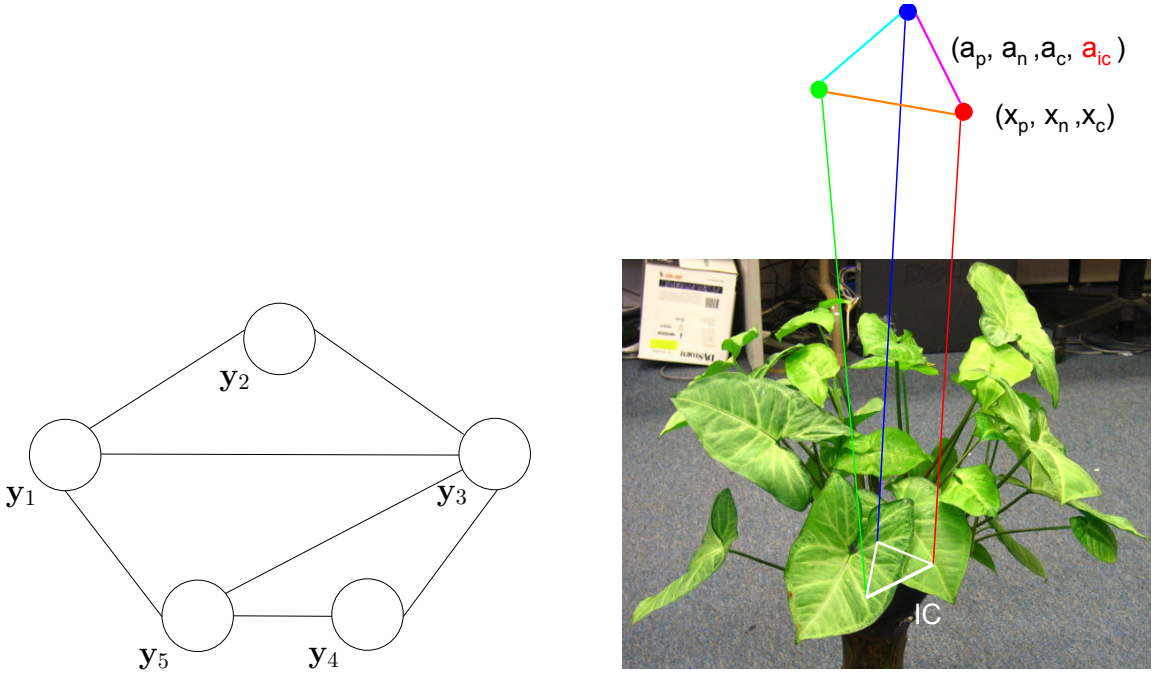
Chapter 3

Pairwise Graph: Joint Segmentation

In this chapter, we demonstrate the conventional pairwise graph partitioning on a real system for joint 3D clustering and 2D image segmentation, called joint segmentation. We address two important issues: 1) designing the edge to make the best use of the joint 2D and 3D information 2) proposing the effective procedure to intensively makes good use of the states of the art of both spectral graph partitioning and combinatorial graph cuts.

This joint segmentation of both 3D and 2D, we argue, is the fundamental stage of 3D modeling from multi-view images. First, we calculated the reconstructed 3D points based on a quasi-dense approach to structure from motion proposed by Lhuillier and Quan [72]. The key advantage of the quasi-dense approach is that it not only delivers the structure from motion in a robust manner for practical modeling purposes, but also it provides a cloud of sufficiently dense 3D points that allows the objects to be explicitly modeled. Second, to structure the available 3D points and registered 2D image information for the subsequent modeling, we propose a probabilistic framework for the joint segmentation. The joint segmentation is modeled as a pairwise graph partitioning problem, in which the affinities are defined by utilizing jointly the 3D and 2D information, and is solved by making use of spectral graph partitioning and combinatorial graph cuts. Moreover, we design a nice and easily-used interactive interface to allow the user to provide the 3D segmentation hints on 2D images in an intuitive manner. Finally, we demonstrate our joint segmentation framework for image based plant modeling.

This chapter is based on the two publications [86, 87] and organized as follows. Sec. 3.1 gives a brief introduction. Sec. 3.2 introduces the joint segmentation framework for image based modeling. The joint prior model is presented in Sec. 3.3. Sec. 3.4 gives the graph partitioning approach using spectral and combinatorial methods. The results are shown in Sec. 3.5. And Sec. 3.6 concludes this chapter.



(a) Pairwise graph

(b) Joint affinities

Figure 3.1: An illustration of joint affinities. In this chapter, we show an illustrative example in which spectral graph partitioning and combinatorial graph cuts are intensively used. The main contribution is the joint affinity definition. (a) shows a pairwise graph. (b) illustrates the joint affinity. $\mathbf{x}_p$, $\mathbf{x}_n$, $\mathbf{x}_c$ are 3D position, normal, color features respectively, and the corresponding affinities are denoted as a_p, a_n, a_c. In addition, The contour information IC in the image edge projected from 3D edge is incorporated into the affinity definition, denoted as a_{ic}.

3.1 Introduction

In Chapter 2, we review the state of the art of graph partitioning. For the traditional pairwise graph partitioning problem, the affinity on each edge (or the distance on the edge) is the most important factor to affect the partitioning performance. In this chapter, we present an real example to show how to set the affinities smartly to obtain the satisfactory performance for further applications. More precisely, we address on 3D points clustering problem in which 3D points are reconstructed from multiple views, and hence can be associated with 3D coordinates and appearance information. In addition, the edge between 3D points may also associate a real edge on the images, which can help 3D points clustering. The typical contribution is the joint affinity definition as shown in Fig. 3.1.

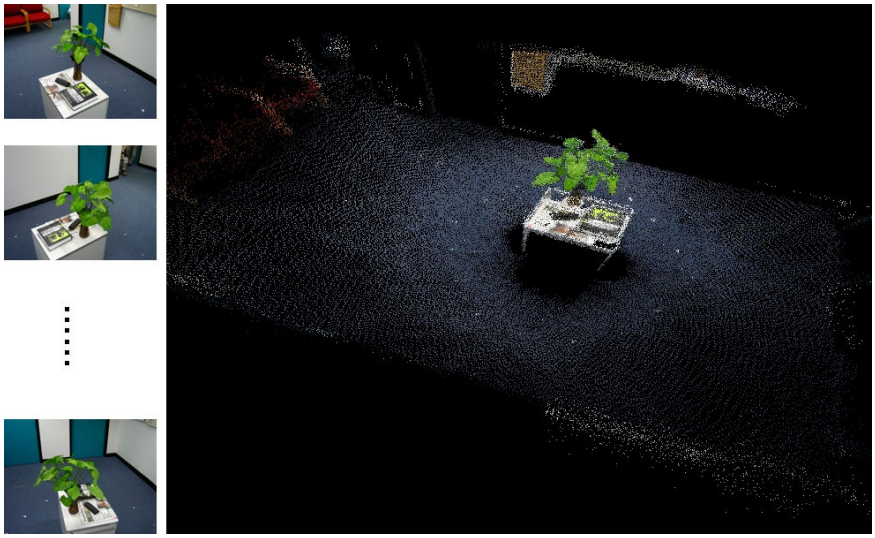


Figure 3.2: One overview on the right of the reconstructed quasi-dense points for the entire scene from 25 images shown on the left.

3.2 Joint Segmentation for 3D Modeling Application

3.2.1 Background

Quasi-dense Approach This sparse structure from motion approach usually requires a dense frame rate and leads to a too sparse set of points to be sufficient for object modeling and depiction. This insufficiency motivated the development of the quasi-dense approach started since 1998 in [71] and matured into the work in [72]. It is robust and handles more distant views. More importantly, it produces a set of semi-dense 3D points that was impossible with the previous methods. The quasi-dense approach was in detailed presented in [72].

One example of quasi-dense approach is shown in Fig. 3.2. The original 25 images used for reconstruction are shown in the left, and one overview of the reconstructed quasi-dense points is shown on the right.

It requires further important processing to obtain the object surface modeling from the 3D points. A variational level set based surface modeling was proposed in [72], by defining the unified functional integrating both 3D quasi-dense points and 2D image information , which was shown to more robust than other approaches using only 2D information or only 3D information. One example from [72] is shown in Fig. 3.3. It usually produces satisfactory results for objects with simple surface. However, it is not proper to reconstruct the surface of complex objects, such as a tree with many leaves.

Joint Segmentation The increased density of the reconstructed 3D points from multiple views, paves the way for the three-dimensional modeling of objects in space, in addition to the recovery

Figure 3.3: The surface reconstruction results by variational level set approach from the quasi-dense points.

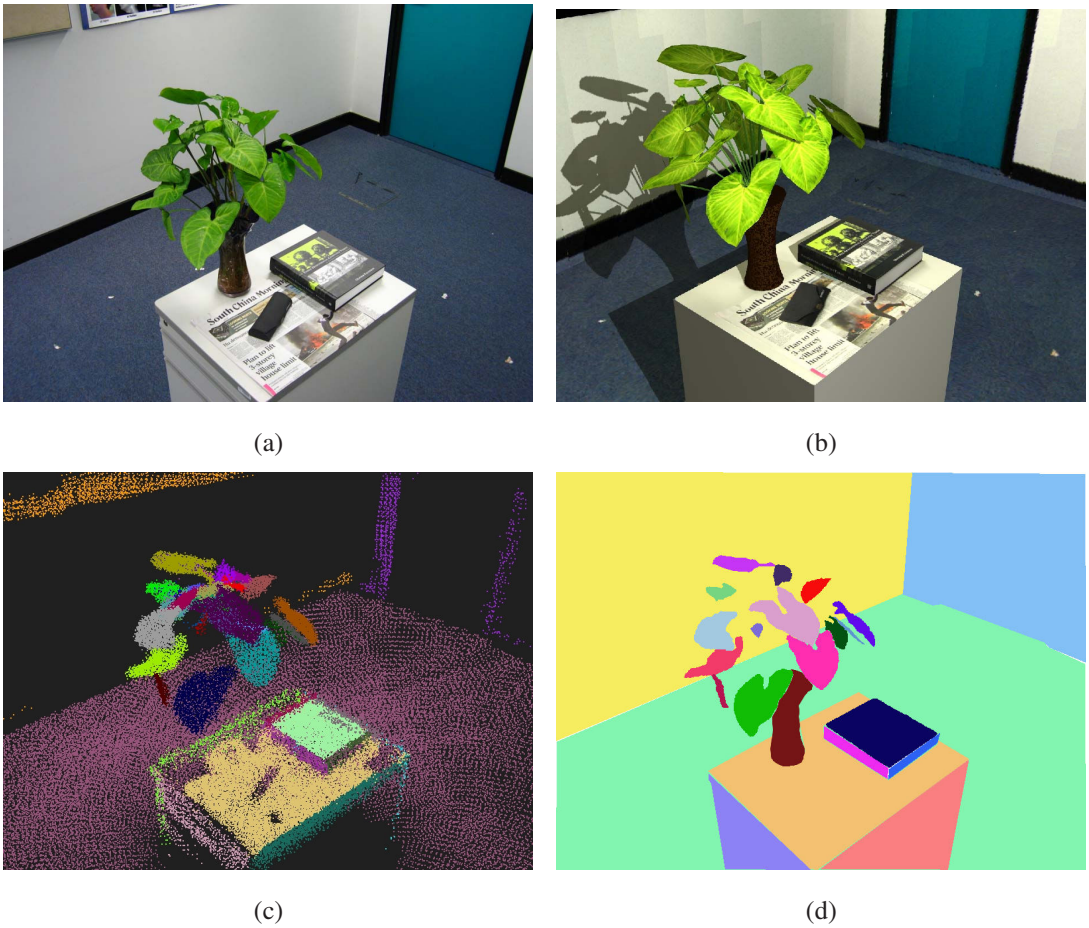
of camera positions. But the 3D points, even semi-dense, are unstructured in space, therefore they are not yet sufficient for creating a geometric model of the underlying complex objects, such as a tree. It is necessary to group the points and pixels that belong to the same *object* into the same cluster of points and pixels. Obviously, the concept of object is subjective: for the example scenario shown in Fig. 3.4(a), the whole plant might be considered as an object for some applications, whereas each individual leaf should be considered as an independent object for the others, depending on the application and the realism details of the modeling required.

This chapter will introduce a pairwise graph based joint segmentation framework for both 3D points and 2D pixels, and look for robust and efficient solutions to it in Sec. **??**. Fig. 3.4(a) shows one of the original 25 images captured by a handheld camera for the example scenario, a rendered image of the final modeling result based on the desired 3D segmentation and 2D segmentation results by a semi-automatic approach developed in [86] using the joint segmentation method presented in this chapter. The segmentation and modeling of such complex objects are almost impossible without the joint segmentation. Our approach defines a joint graph by utilizing both 3D quasi dense points and 2D image information, then partition it interactively to group the 3D quasi dense points and 2D image simultaneously. It has several advantages:

1. It is more robust than the segmentation approach using only 2D image information or only 3D quasi dense points.

2. Our approach provides a nice interface to allow the user to perform some operations on 2D images to help the complex 3D and 2D grouping.

3. The probabilistic segmentation framework can be generalized for general 3D modeling without



(a) (b)

(c) (d)

Figure 3.4: A proper segmentation is fundamental to 3D modeling. (a) One of the 25 original images captured by a handheld camera. (b) A rendered image at a similar viewpoint of the reconstructed 3D model based on the segmentation results in (c) and (d). (c) A desired segmentation of 3D data points from the reconstructed quasi-dense points, and (d) a desired segmentation of an input image.

only being limited on the tree modeling.

Our joint segmentation framework is based on pairwise graph partitioning, which consists of three modules: 1) Joint graph construction; 2) Automatic graph partitioning; 3) Interactive graph partitioning. And the next section will describe the complete 2D image segmentation from the joint segmentation results.

Related Work on Segmentation Image segmentation has been a traditional and fundamental topic in computer vision. There is an abundant literature in both segmentation of 3D range images or depth maps, and segmentation of normal 2D gray level or color images. Segmentation of range images usually looks for more local geometric characterization since it has a high density of 3D points, it leads to different approaches as it has no associated 2D images. Segmentation of normal images uses only pure pixel information, and has attracted lots of recent attentions. Unfortunately it operates only in 2D

space with pure pixel information, and its purposes are often motivated by object recognition. Some representative works include [98, 107]. It is natural that segmentation of multi-view, often calibrated off-line, handles both image and depth information. There are several representative works. The layered approaches originated from [114] usually do not directly adopt 3D reconstruction information. In [121, 124], motion estimation and segmentation on the extracted correspondences between frames are performed, then layer assignment (i.e. pixel label) is obtained through propagating the labels of the corresponding pixels. In [83], joint inference of motion estimation and labeling is solved using the Expectation Maximization (EM) algorithm. The modern stereo matching approach is very similar to the layered approach, and in essence it discretizes the 3D space into a few layers. One of the most representative works is [66]. Bilayer segmentation as a simple layered representation, basically uses the learned appearance model instead of correspondence relations. Stereo cues are probabilistically fused for segmentation in [65]. The edges of the background of last frame are used to attenuate the edges of current frame in [101]. The probabilistic motion model is learnt to help segmentation in [32].

The existing approach may directly integrate the motion constraints into the segmentation, but are often primarily preoccupied with the recovery of depth information rather than with the clustering of the objects. It provides little exploitation of reliable 3D information, probably due to lack of such information. Intuitively, there is much richer information available to explore for object segmentation when both 3D and 2D information is available. For instance, some objects are obviously separable in 3D whereas others are clearly cut out by image boundary information even if they are closely connected in space as shown in Fig. 3.6.

3.2.2 Formulation

Given a set of 3D points, the quasi-dense points in our case such as the example shown in Fig. 3.2, and all the images of the sequence, we would like to segment jointly all 3D points and 2D pixels into groups of meaningful objects.

Let $I = \{I_i\}$ be the set of n images with $i = 1, \ldots, n$. Each image I_i is represented by a set of pixels in RGB space, i.e. $I_i = \{(\mathbf{u}_k, \mathbf{c}_k)\}$ with k up to the number set by the image resolution, and each image point includes its position $\mathbf{u}_k$ in image space and three colors $\mathbf{c}_k$. It is assumed that all the images are fully calibrated with respect to a common coordinate fame. We define a *joint point* $\mathbf{x}$ to be a vector composed of the 3D coordinates (x, y, z) of a point in space and all its corresponding image points $\mathbf{u}_i$ in all images, i.e. $\mathbf{x} = ((x, y, z), (\mathbf{u}_1, \mathbf{c}_1), \cdots, (\mathbf{u}_n, \mathbf{c}_n))$, where each image point satisfies $\mathbf{u}_i = \mathbf{P}_i(x, y, z, 1)^T$ for the projection matrix $\mathbf{P}_i$ of the i-th camera. The correspondence information is encoded in the joint point representation. And each joint point $\mathbf{x}$ is associated with a n-dimensional visibility vector $\mathbf{v}$ with binary values to indicate that $\mathbf{u}_i$ is visible in the i-th image if

the i-th component is 1, and invisible otherwise. A segmentation is a set of labels $L = \{l_k\}$, and each of them l_k assigns a set of joint points to a common group. The number of the labels is unknown. If $X = \{\mathbf{x}_j\}$ is the given set of joint points, and $V = \{\mathbf{v}_j\}$ the given set of visibilities for each joint point, X and V are given by the quasi-dense reconstruction in our case, then L_X is a labeling for the given set of joint points X. The inference of L_X could be treated as a maximum *a posteriori* estimate of the probability $P(L_X|X, V, I)$. A joint segmentation L_X is formally given by

$$L_X = \arg\max_{L_X} P(L_X|X, V, I).$$

This MAP could be solved by representing the posterior probability $P(L_X|X, V, I)$ as a Conditional Random Field (CRF) as in [68],

$$P(L_X|X, V, I) \propto \exp -\{E^l(L_X; X, V, I) + \lambda E^s(L_X; X, V, I)\}, \tag{3.1}$$

where $E^l(\cdot)$ is the energy for the likelihood model of the labeling, and $E^s(\cdot)$ is the energy for the conditional prior model of the labeling.

To define the energy function in Eqn. (3.1), it is necessary to provide the graph structure explicitly, including the connection relation and the edge affinity (prior), and the observation (likelihood) as well. We will describe the three items in the following.

3.2.3 Joint Likelihood

For a joint point, there is no evidence to support the dependency of the 3D position $\mathbf{x}^s = (x, y, z)$ and the image colors $\mathbf{x}^c = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$, the joint likelihood model is then divided into the independent spatial and color terms:

$$E^l(L_X; X, V) = \rho_{l_1} E^l(L_X; X_C, V) + \rho_{l_2} E^l(L_X; X_S), \tag{3.2}$$

where $X_C = \{\mathbf{x}_j^c\}$ is the set of the components of the image colors of the joint points X and $X_S = \{\mathbf{x}_j^s\}$ is the set of the components of 3D coordinates of the joint points X.

Color likelihood The Color likelihood is taken to be

$$E^l(L_X; X_C, V) = -\sum\nolimits_{j=1}^{m} \log p(\mathbf{x}_j^c|l_j),$$

where the color likelihood model for each joint point is defined by $p(\mathbf{x}^c|l) = \prod p(\mathbf{c}_i|l)^{\frac{\mathbf{v}_i}{|\mathbf{v}|_1}}$ as the geometric mean of the likelihoods of all its 2D pixels. The L_1-norm $|\mathbf{v}|_1$ of $\mathbf{v}$ is just the number of visible pixels of the joint point $\mathbf{x}$. The invisible features $\mathbf{c}_i$, with $\mathbf{v}_i = 0$, do not contribute to the

model. In other words, the definition actually only performs the geometric mean on the visible 2D pixels, which equally treats every joint point without over-using it.

The probability density $p(\mathbf{c}|l)$ describes the color distribution of the joint points for the label l. It is natural to take a Gaussian Mixture Model (GMM) [89, 11, 65], whose parameters could be estimated by the Expectation Maximization (EM) algorithm [35]. Again the visible color of each joint point $\mathbf{x}$ is weighted by $\frac{1}{|\mathbf{v}|_1}$ for the estimation to view equally each joint point.

Spatial shape likelihood The spatial shape likelihood is used to model the shape prior of the object. It is desirable to be able to model a general surface patch that will be good for modeling the visible surface of an object, but it is usually computationally difficult. We will take a planar model to approximate the shape model of the objects, this is similar to [105] that uses a plane as a layer. Using a planar model is however insufficient since the 3D points for an object spatially spread rather in a compact manner. To take the compactness into account, we use the Mixture of Probabilistic Principal Component Analysis (MPPCA) [104] to obtain a probability measure. And this shape likelihood term is defined as:

$$E^l(L_X; X_S) = -\sum\nolimits_{j=1}^{m} \log p(\mathbf{x}_j^s|l_j),$$

where $p(\mathbf{x}^s|l)$ is formulated in terms of Probabilistic Principal Component Model (PPCA) [104]. The principal 2D subspace $\mathcal{P}_2$ and the complementary 1D subspace $\mathcal{P}_1$ are first computed through Principal Component Analysis (PCA) for the points with the same label. Then the probability is the combination of the two terms. The first term is to penalize the projection in the complementary subspace $\mathcal{P}_1$, i.e. the residual error; the smaller the probability is, the larger the residual error is. The second term is to measure the compactness of the projected points in the principal plane $\mathcal{P}_2$ through a Gaussian model. We assign a larger weight to the first term than the second, slightly different from [104].

3.3 Joint Prior

3.3.1 Graph Construction

The graph is constructed to model the joint points in an intuitive way. Generally, we can build the graph using the k-Nearest Neighbor (k-NN) technique based on the Euclidean distance of 3D distance of the joint points. Our goal is to embedding all the information of the joint points, including 3D Euclidean information and 2D image information. Hence the edges should ideally be constructed by incorporating both 3D position information and 2D image information. A straightforward way to define the edges is defining the distance by using both the Euclidean distance of 3D space and

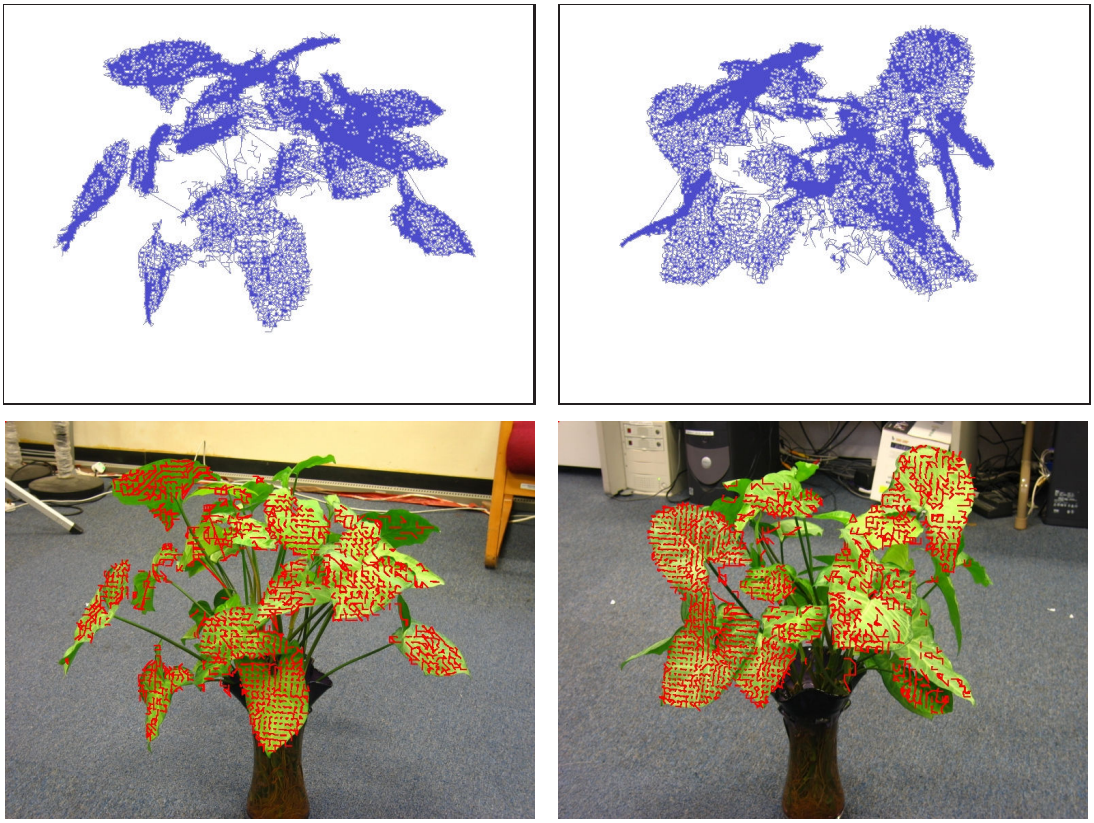


Figure 3.5: An example of joint graph. The first row shows the graph in 3D space in two different views. The second shows the graph in 2D image in the corresponding views.

the color distance of the image space. However, we observed that the 3D points directly reflect the compactness of the reconstructed objects, and color is a complementary feature, and we considered that we can make the better use of 2D image information, such as the image gradient information, not just the color difference, if the edge on the graph can have a real correspondence in the 2D image. we construct the graph by considering the visibility information as well. We first build for each view a k-NN network on the corresponding set of joint points according to the 3D Euclidean distance, and set k to be 5 by default. Then we combine those networks together to reach a graph on the entire joint points. Finally we discard some incident edges with larger distance for each joint point such that the graph is not so dense for efficient computation. It should be noted that the graph construction is critical in that the 2D affinity definition is entirely based on the construction. An example of joint graph is shown in Fig. 3.5.

3.3.2 Joint Prior

Conditional prior model is used to measure the consistency between the labelings of the joint points. The k-way consistency is usually more powerful to describe the prior than the pair-wise consistency,

but it leads to higher computation cost. We adopted the common pair-wise affinity as the prior that is formulated as the following:

$$E^s(L_X; X, V, I) = \sum\nolimits_{(i,j)\in E} E^s(l_i, l_j; X, V, I) = -\sum\nolimits_{(i,j)\in E} a(i,j)\delta(l_i = l_j),$$

where $a(i,j)$ is the affinity function between the points i and j, E is the set of all pairs of points corresponding to linked edges in a graph, and $\delta(a = b)$ is a Dirichlet function such that $\delta(a = b)$ is 1 if $a = b$ holds and 0 otherwise. The quality of a segmentation based on the formulation fundamentally depends on the affinity, we seek therefore to define it jointly from both 3D and 2D features.

3D affinity Points what are closer in space tend to have higher probability of belonging to the same group, i.e. the distance between the points of the same group is smaller than that of the points in different groups. We naturally take this spatial distance as an affinity measure $a_{3d}(i,j) = \exp(-\frac{||\mathbf{s}_i - \mathbf{s}_j||^2}{2\sigma_{3d}^2})$, where $\sigma_{3d} = \mathrm{E}^{1/2}(||\mathbf{s}_i - \mathbf{s}_j||^2)$. The Gaussian function has the desired properties for an affinity, and is popular in spectral clustering and normalized cut [98]. In addition to the 3D Euclidean distance, the normal directions are also important for shape smoothness. We incorporate the difference between normal directions into the affinity and define $a_{3n}(i,j) = \exp(-\frac{||\mathbf{n}_i - \mathbf{n}_j||^2}{2\sigma_{3n}^2})$, where n_j is the normal direction vector of the point j, approximately estimated from its neighbor points, and $\sigma_{3n} = \mathrm{E}^{1/2}(||\mathbf{n}_i - \mathbf{n}_j||^2)$. The final 3D affinity is given by $a_3(i,j) = a_{3d}(i,j)a_{3n}(i,j)$.

2D affinity Since a joint point $\mathbf{x}$ is associated with the image colors, we can define an affinity function encoding the color differences as $a_c(i,j) = \exp(-\frac{||\mathrm{E}(\mathbf{c}_i) - \mathrm{E}(\mathbf{c}_j)||^2}{2\sigma_c^2})$, where $\sigma_c = \mathrm{E}^{1/2}(||\mathrm{E}(\mathbf{c}_i) - \mathrm{E}(\mathbf{c}_j)||^2)$, and $\mathrm{E}(\mathbf{c}) = \frac{1}{|\mathbf{v}|_1}\sum_{i=1}^{n}\mathbf{c}_i$. This color consistency between joint points is intuitively estimated using their average colors, since different points may have different numbers of visible color features. Averaging the colors leads to a more stable solution. However, this affinity function only makes sense between the objects with apparent different colors. In case of apparent similar colors, image contour features, similar to [74], should be incorporated into the affinity as illustrated in Fig. 3.6.

It is assumed at present that each pixel $\mathbf{u}$ in view I_v is associated with a response $g_v(\mathbf{u})$ to show the degree of the pixel lying on a contour point. The endpoints of the edge $(i,j) \in \mathcal{E}$ must both be visible at least in one view, meaning that the line segment $[i,j]$ must correspond to a line segment visible in the same view. We can use the following affinity measurement

$$a_{ic}(i,j) = \exp(-\frac{\mathrm{med}_v\{\max_{t_v\in[i,j]_v} g_v(t_v)\}}{2\sigma_{ic}^2}), \tag{3.3}$$

where the inner term $\max_{t_v\in[i,j]_v} g_v(t_v)$ finds the maximum contour response along the projected line segment $[i,j]_v$ in view v, the outer term $\mathrm{med}_v\{\cdot\}$ tries to seek the median contour response in all

(a) (b) (c) (d)

Figure 3.6: One source image is shown in (a). The zoomed subimage is shown in (b). Assume that the Euclidean distances satisfy $|s_a - s_b| > |s_b - s_c|$ in 3D space. The similarity of the colors for a and b makes them closer in grouping, while the presence of the edge inbetween b and c moves them apart in grouping. A typical example in which the leaves can not be separated with 3D Euclidean distances is shown in (c), but can be separated after considering color and contour information as shown in (d).

possible views, and σ_{ic} is the variance of the median contour responses of all line segments. Different from [86], the median operator is observed in experiment to be more robust than the maximum operator.

An edge map in [74] is used to compute the contour responses in this paper. The oriented filter bank based on rotated copies of a Gaussian derivative and its Hilbert transform are used. Let $f_1(x, y) = G''_{\sigma_1}(y)G_{\sigma_2}(x)$, and $f_2(x, y) = H(f_1(x, y))$ is the Hilbert transform of $f_1(x, y)$ along the y axis. The oriented energy at angle 0° is defined as $E_{0^\circ} = (I * f_1)^2 + (I * f_2)^2$. Then the contour response is defined as $g(x, y) = \max_\theta E_\theta(x, y)$.

Finally, we are able to perform simple multiplication of the affinities to define the joint affinity to be $a(i, j) = a_3(i, j) \times a_c(i, j) \times a_{ic}(i, j)$.

3.4 Graph Partitioning

3.4.1 Spectral Step

The final objective function for the joint segmentation is given by

$$E = -\sum_{i=1}^{m}(\rho_1 \log p(\mathbf{x}_i^c|l_i) + \rho_2 \log p(\mathbf{x}_i^s|l_i)) - \lambda \sum\nolimits_{(i,j)\in\mathcal{E}} a(i, j)\delta(l_i = l_j).$$

This problem is, generally speaking, a graph partitioning problem, but the optimization depends on how discriminative the likelihood is.

In the case of separating objects with apparent color dissimilarity, the color likelihood is usually discriminative, but not the shape model. The objective function reduces to the common combinatorial

formulation:

$$E = -\rho_1 \sum\nolimits_{i=1}^{N} \log p(\mathbf{x}_i^c|l_i) - \lambda \sum\nolimits_{(i,j)\in\mathcal{E}} a(i,j)\delta(l_i = l_j). \tag{3.4}$$

This equation is a typical discrete CRF formulation, which can be efficiently optimized using graph cut algorithm [21]. Its complexity is of $O(N^3)$ because the graph is sparse, and there are N points and $O(N)$ edges.

In other cases there exists discriminative spatial shape feature, but not discriminative colors. For instance, we may want to model individual leaves, which requires an individual leaf segmentation, The color is similar to all leaves of the same plant, but each leaf lies on a different surface patch in space. The objective function can be simplified to

$$\begin{aligned} E &= -\rho_2 \sum\nolimits_{i=1}^{m} \log p(\mathbf{x}_i^s|l_i) - \lambda \sum\nolimits_{(i,j)\in\mathcal{E}} a(i,j)\delta(l_i = l_j) \\ &= -\rho_2 \sum\nolimits_{i=1}^{m} \log p(\mathbf{x}_i^s|l_i) + \lambda \sum\nolimits_{(i,j)\in\mathcal{E}} a(i,j)(1 - \delta(l_i = l_j)) + \text{const.} \end{aligned} \tag{3.5}$$

Now the problems, similar to those discussed in [129, 140], can be casted into an unsupervised clustering framework. We can use an iterative algorithm that combines GMM and graph cut. We rewrite the affinity part in matrix form,

$$\sum\nolimits_{(i,j)\in E} a(i,j)(1 - \delta(l_i = l_j) = \text{Trace}(\mathbf{X}^T(\mathbf{D} - \mathbf{W})\mathbf{X}), \tag{3.6}$$

where $\mathbf{W}$ is the symmetric affinity matrix with each entry $\mathbf{W}_{ij}$ corresponding to the affinity value of the edge (i,j), and $\mathbf{W}_{ij} = 0$ if there is no edge between the points i and j, $\mathbf{D}$ is the diagonal degree matrix in which the diagonal entry $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$, $\mathbf{X}$ is a label matrix of size $n \times c$ such that $\mathbf{X}_{il} = 1$ if the label of point i is l and 0 otherwise. The minimization of the above function is in essence a graph min-cut problem. As pointed out in [98], a normalized cut criterion will give a better graph partition. We use the normalized cut method to get a reliable labeling (graph partitioning):

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \text{Trace}(\mathbf{X}^T\mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}\mathbf{X}). \tag{3.7}$$

We optimize Eqn. (3.5) in two steps. First, we use the technique in [98] to solve Eqn. (3.7) to get an initial label $\hat{\mathbf{X}}$. In the second step, the PPCA model can be estimated according to each group of points. Afterwards the MPPCA is used to refine the labeling. Although an iterative algorithm can be used for optimization, its convergence can not be theoretically guaranteed and it does not improve significantly the results. In our experiments, we observed that the two steps without iteration work satisfactorily, thanks to the affinity function. The most computationally expensive step is sparse SVD for $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ in Eqn. (3.7). Its complexity is of $O(N^2)$ as $\mathbf{L}$ has only $O(N)$ nonzero entries. Before performing spectral graph partitioning, we first extract connected components.

Initializing Graph Partitioning by Connected Component Extraction The k-NN graph construction usually can not guarantee that the graph is connected, which means that the graph consists of several subgraphs. This is reasonable because the 3D Euclidean distance may be so large that the two objects are not connected if the two objects are well separated. In this case, it will reduce the computation cost to perform the connected component analysis as initial graph partitioning before spectral clustering. Theoretically speaking, the connected component analysis is equivalent to spectral clustering.

To reduce more computation cost, we disconnected the edges whose distance (the logarithm value of the affinity $d(i,j) = \log(a(i,j))$) is too large. We predefined a threshold based on the variance as

$$d_t = k\sigma_d, \tag{3.8}$$

where k is set to 3 by default.

3.4.2 Combinatorial Step

User Assistance In general, the process of segmentation is subjective; the segments that represent different objects depend on the person's perception of what an object is. The following discussion mainly focuses on the plant leaves segmentation. For this task of partitioning the image into several meaningful objects such as leaves, while the interpretation is much clearer, the problem is nonetheless still very difficult. This is because some objects such as leaves may look much similar, and boundaries between leaves are often very subtle.

We designed simple ways that the user can help refine areas where automatic segmentation fails. It is uneasy to directly operate the joint points in the 3D space. We provide the interface to allow the user to manipulate the joints in the 2D image space. We project the current groups of joint points into the image space as a feedback mechanism. we provides the following several ways to allow the user to input his assistance:

1. Click to confirm grouping: The user can click on an image region to indicate that the corresponding group of joint points is acceptable.

2. Draw to split and refine. The user can select one unsatisfactory group, which can not be split in the automatic segmentation process. Then the user draws a region in one view to separate the points inside the region from the outside points. This triggers the subgraph splitting procedure described in Subsec. 3.4.2.

3. Click to merge. The user clicks on the image to select two groups of joint points by just connecting two points from the two groups. An example of merging two groups is shown

Figs. 3.9 and 3.10.

Subgraph Splitting by s/t Min-cut We assign the joint points inside the region label source (0), and the ones outside label sink (1) in the operated view. This operation only labeled the points which are visible in the current view. We will propagate this label information into all other joint points of this group. We formulate the subgraph splitting as a source and sink graph minimum cut problem, which minimizes the following energy function:

$$E(l) = \sum_{(p,q)\in\mathcal{N}} (1 - \delta_(l_q, l_p)) \frac{1}{d^2(p,q) + \epsilon} + \sum_p D_p(l_p), \tag{3.9}$$

where $\delta(l_p, l_q)$ is 1 if $l_q = l_p$, 0 if $l_q \neq l\_p$, and $l_p, l_q = \{0, 1\}$. ϵ is a very small positive constant set to 0.00001, which is only used for numerical computation problem. The data term $D_p(l_p)$ encodes the user-assigned labels:

$$y = \begin{cases} D_p(0) = 0, \\ D_p(1) = \infty, \end{cases} \text{ if } l_p = 0,$$

and

$$y = \begin{cases} D_p(0) = \infty, \\ D_p(1) = 0, \end{cases} \text{ if } l_p = 1.$$

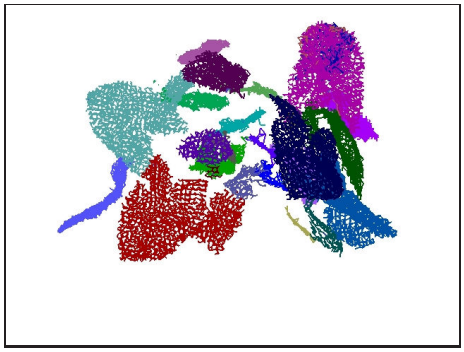
There are many standard algorithms for it. We use the implementation by Boykov, available from ... The complexity of min-cut is $O(N^3)$ in our problem with N the node number. Since each group is usually rather small, the running is immediate, allowing the interface to provide real time interactivity.

An example of such splitting is shown Figs. 3.7 and 3.8. Fig. 3.7 shows the assistance from the user, and Figs. 3.8 and shows the results.

3.5 Implementation and Results

Data acquisition We typically capture about 35 images by moving around a foreground object as discussed in [72]. We choose two typical indoor scenes and three scenes with plants. The choice of plant images is motivated by the fact that the plants are omnipresent in many scenes and are reputed to be difficult in segmentation and modeling, and that the segmentation into individual leaves from pure images is not yet possible even with intensive learning. On the other hand, if we want to obtain a realistic model of a plant as demonstrated in [86], it is inevitable that a proper segmentation should be obtained. Obviously, the examples we choose have relatively large sizes of leaves for the given resolution of the cameras. For the dense foliage of the trees with small leaves, it exhibits texture-like properties and a different methodology should be developed.

(a) The intermediate segmentation shown in 2D image

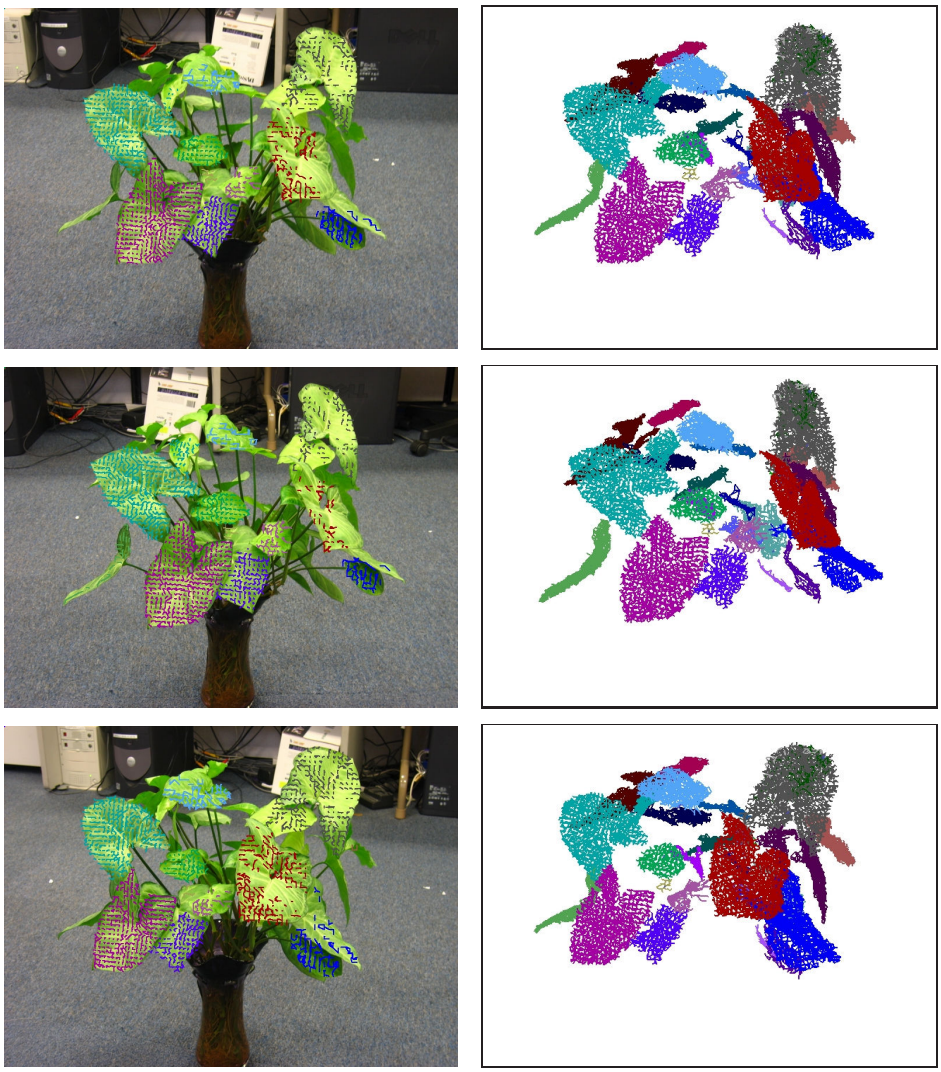


(b) The intermediate segmentation shown in 3D space

(c) The selected one group to be split.

(d) The user hints.

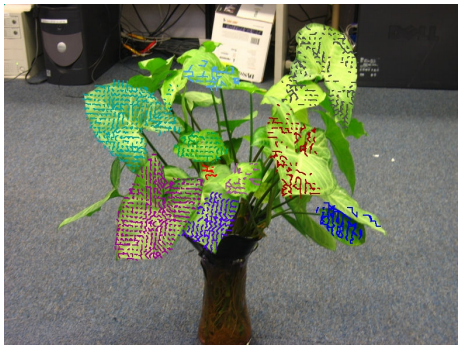
Figure 3.7: The assistance from the user in splitting operation.



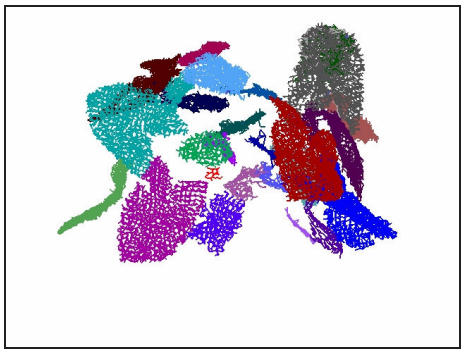
(a) The splitting results in 2D image of different views.

(b) The splitting results in 3D space of different views.

Figure 3.8: The splitting results with user's assistance given in Fig. 3.7.

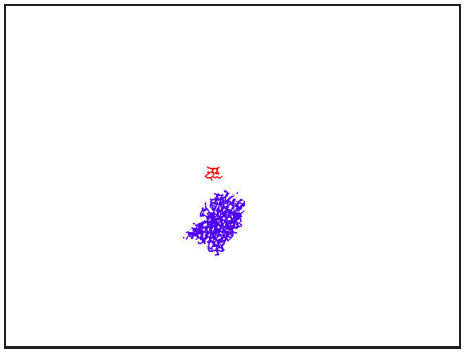


(a) The intermediate segmentation shown in 2D image



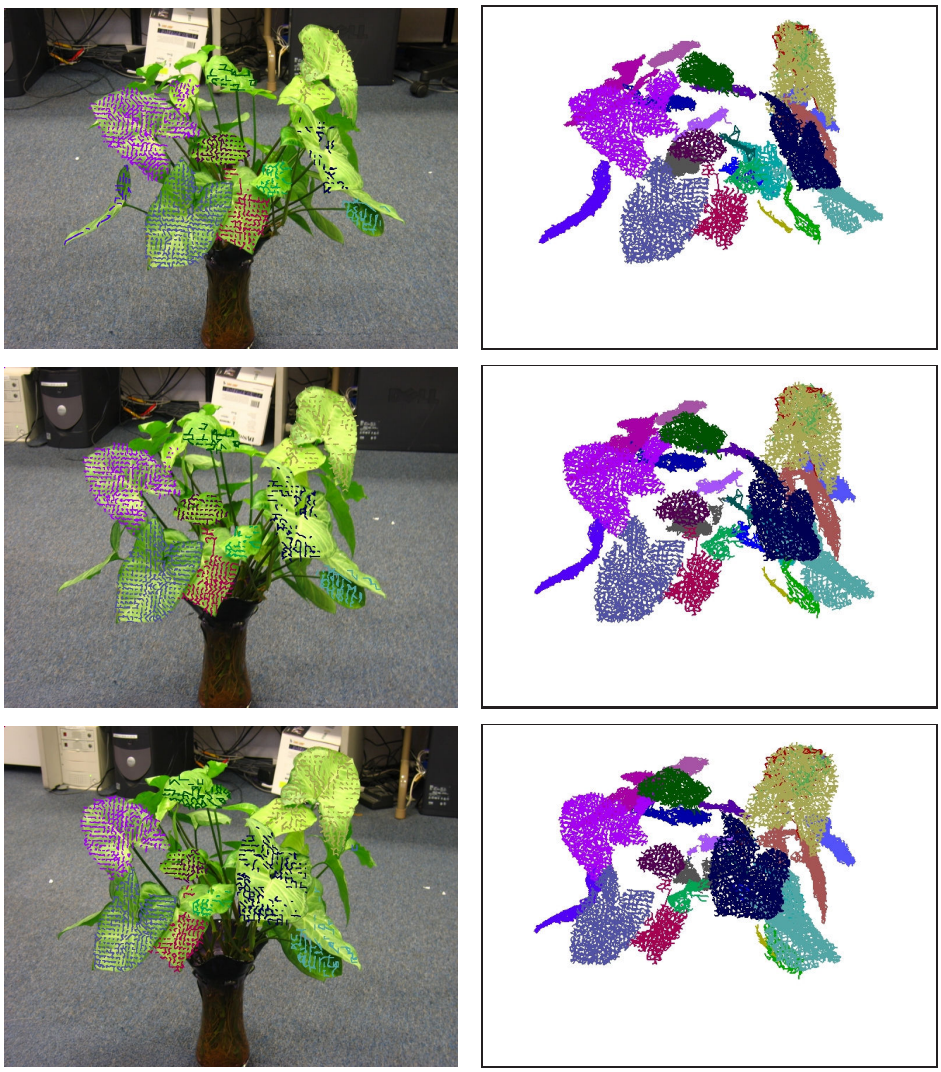
(b) The intermediate segmentation shown in 3D space

(c) The selected two group to be merged.



(d) The selected two group to be merged.

Figure 3.9: The assistance from the user in merging operation.



(a) The splitting results in 2D image of different views.

(b) The splitting results in 3D space of different views.

Figure 3.10: The merging results with user's assistance given in Fig. 3.9.

The two indoor scenes are captured with 48 and 20 images of resolutions 972×1296 and 1024×768. The plants are captured with 35, 35 and 40 for nephthytis, poinsettia and schefflera. The image resolution is 1944×2592 (except for the poinsettia, which is 1200×1600). For the efficiency of structure from motion, we down-sampled the images to 583×777 (for the poinsettia, to 600×800). It took approximately 10 mins for about 40 images on a 1.9GHz P4 PC with 1 GB of RAM. On average, we reconstructed about 100 thousands 3D points for the scene.

Segmentation Propagation The joint segmentation L_X is so far defined for a given set of joint points X, which is, practically, the set of quasi-dense points. Ideally, if there was a dense reconstruction from multiple views, the joint segmentation could have been applied to the set of dense joint points. This only increases the computational cost. As we have argued at the beginning of the paper that the full dense reconstruction is hardly achievable and this motivated the development of our quasi-dense approach. Though the quasi-dense 3D points are the best we can achieve in 3D in terms of computability, but in image space, the object image boundaries are are expected for many geometric reconstruction. This means that the dense image segmentation could have been incorporated into the probabilistic framework. This can be done by extending the definition of labeling to all pixels. Let $L = L_X \cup L_I$ be the collective labeling for the joint points and all image pixels. Then the segmentation is given by

$$L = \arg\max_{L} P(L|X, V, I).$$

The probability $P(L|X, V, I)$ is factored into two terms

$$P(L|X, V, I) = P(L_I|X, V, I, L_X)P(L_X|X, V, I),$$

where the second term is the labeling model of joint points L_X, and the first term is image labeling L_I given the labeling of the joint points L_X or the propagation of L_X into image segmentations L_I. This propagation model can roughly be interpreted as a data augmentation problem [103] in which L_X is treated as hidden variables. Usually an iterative algorithm could be adopted to this optimization problem, however, from the perspective of labeling and the fact that the visibility V of X is given, it is unnecessary to re-estimate L_X from an estimate of L_I. This leads to the following segmentation propagation.

Given the labeling of the joint points L_X, the corresponding (quasi-dense) pixels in each view have been assigned the corresponding labels since the correspondence information and the visibility are provided. The probabilistic dependency relation of joint points and image segmentation is represented as a Bayesian network illustrated in Fig. 3.11(a). According to the conditional independence and Bayesian network properties, the Bayesian network can be exactly divided into n independent networks as shown in Fig. 3.11(b), where $\gamma_v = \{\mathbf{x}\} \subset X$ is the set of the joint points on which the

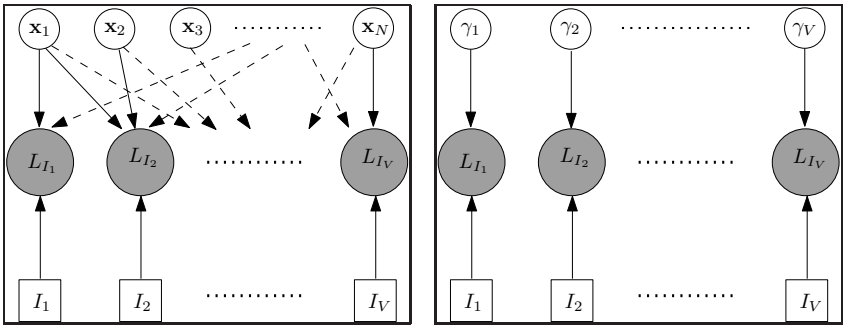


Figure 3.11: (a) The network shows the conditional relation of the joint segmentation probability model for all views. The $a \rightarrow b$ means that b is conditionally dependent on a. (b) The network shows the independence factorization of the network in (a) according to the probability property.

labels of image I_v are dependent. This means that labeling other pixels in each view is dependent on those fixed labels and the image itself. It should be noted that the division holds only if the labels of the joint points are given. Formally, $P(L_I|X, V, I, L_X)$ can be factorized into:

$$\begin{aligned}(L_{I_1}, \cdots, L_{I_n}) &= \arg\max P(L_{I_1}, \cdots, L_{I_n}|X, V, I, L_X) \\ &= \arg\max \prod_{v=1}^{n} P(L_{I_v}|X, V, I, L_X) \\ &= \arg\max \prod_{v=1}^{n} P(L_{I_v}|L_{\gamma_v}, I_v), \end{aligned} \tag{3.10}$$

The independence of views implies that it amounts to solving $L_I = \arg\max P(L_I|L_\gamma, I)$ if we drop the subscript v for simplicity. Again by adopting CRF to model the probabilistic segmentation model, we have

$$P(L_I|L_\gamma, I) \propto \exp -\{E^c(L_I; L_\gamma) + \rho E^l(L_I; I) + \zeta E^s(L_I; I)\},$$

where $E^c(L_I; L_\gamma)$ is used to penalize the inconsistency between the labels of the joint points and its corresponding image pixels, $E^l(L_I; I)$, called image compatibility term, is used to penalize the incompatibility between the labels and color information of pixels, and $E^s(L_I; I)$, called regularization term, is used to bias the smoothness of the labels of neighboring pixels.

This leads to a formulation that is very close to the image segmentation methods proposed in [19, 73, 11, 89]. The difference is that the condition L_μ or the energy E^c term is provided differently in different methods. The existing methods put the user into the loop to provide an iterative and interactive condition by marking up the image, while our method uses the condition L_μ available from the given quasi-dense points and previously computed joint segmentation. We briefly specify the three energy terms to complete the description of the whole procedure.

Consistency term aims at keeping the consistency between the labels of image pixels and the associated joint points. It is given by $E^c(L_I; L_\gamma) = -\eta \sum_{i \in \gamma} \delta(l_k = \hat{l}_i)$, where k is the pixel index

(a) (b)

Figure 3.12: (a) The superimposition of the projections of segmented 3D groups with one image. (b) The image segmentation of one group in blue, representing one leaf, from the associated group of 3D points.

of the joint point i, and $\hat{l}_i$ is the label of the joint point. We impose a strict consistency constraint by setting $\eta = \infty$. It should be noted that the seeds L_γ are semi-dense as shown in Fig. 3.12(a), which makes labeling of the remaining pixels much more robust and effective.

Image compatibility term is defined as $E^l(L_I; I) = -\sum_k \log p(c_k|l_k)$, where $p(c|l)$ is a GMM probabilistic model that describes the color distribution of the pixels with the label l. It is used to penalize the incompatibility between the label and color information of each pixel.

A regularization term is necessary to make the labels of neighboring pixels as smooth as possible. We take a Potts spatial energy model, $E^s(L_I; I) = -\sum_{(m,n)\in\mathcal{C}} a(m,n)\delta(l_m = l_n)$, where $\mathcal{C} = \{(m,n); |(x_m, y_m) - (x_n, y_n)| \leq d\}$ with d being the neighborhood size and typically set as 1, and the affinity between pixels $a(m,n) = \frac{1}{1+\epsilon}(\epsilon + \exp(-\frac{||c_m - c_n||^2}{2\sigma^2}))$, where ϵ is a dilution parameter for color contrast and is set as 0.1 by default, and σ is the standard deviation and estimated as $\sigma = \sqrt{\mathrm{E}(|c_m - c_n|^2)}$. This model avoids the natural tendency for segmentation boundaries to align with colors of high image contrast.

The overall objective function for segmentation propagation is given by

$$\begin{aligned} E &= E^c(L_I; L_\gamma) + \rho E^l(L_I; I) + \zeta E^s(L_I; I) \\ &= -\eta \sum\nolimits_{i\in\gamma} \delta(l_k = \hat{l}_i) - \rho \sum\nolimits_k \log p(c_k|l_k) - \zeta \sum\nolimits_{(m,n)\in\mathcal{C}} \delta(l_m = l_n) a(m,n). \end{aligned} \quad (3.11)$$

It is typically solved by graph cut [19]. For the objects having apparent color similarity, for example when we would segment out individual leaves, the image compatibility term is not discriminative and can be ignored. The energy then reduces to

$$E = -\eta \sum\nolimits_{i\in\gamma} \delta(l_k = \hat{l}_i) - \zeta \sum\nolimits_{(m,n)\in\mathcal{C}} \delta(l_m = l_n) a(m,n).$$

This can still be optimized using graph cut. Different from existing methods in [19, 73, 89, 65, 101, 32], here only hard constraint, i.e. the consistency term, is involved. It appears to be insufficient

for pixel labeling, but the number of the labeled pixels from the joint segmentation is larger than those in [19, 73, 89, 11]. The propagation from L_γ to the whole image is efficient using a graph cut algorithm as shown in Fig. 3.12(b).

Outline of the algorithms For each data set, the computation of the joint segmentation for 3D points and 2D images proceeds as follows:

1. Train the color distributions for foreground and background. We selected several views randomly. And for each selected view the depths of the pixels that have 3D corresponding points are calculated, and normalized to $[0, 1]$. Then two thresholds $d_f = \frac{1}{5}$ and $d_b = \frac{3}{5}$ are chosen to set the pixel whose depth is smaller than d_f as foreground pixel, and larger than d_b as background.

2. Optimize Eqn. (3.4) using graph cut to obtain foreground and background separation. We set $\frac{\lambda}{\rho_1} = 2.5$ by default, the graph cut implementation in [21] is used.

3. Compute the second smallest eigenvector for the normalized Laplacian matrix $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ using sparse singular value decomposition (SVD) algorithm.

4. Threshold the eigenvector to produce a bi-partitioning of the 3D points. We chose 25 different values uniformly spaced within the range of the eigenvector as possible thresholds. Then we chose the one corresponding to a partition which minimizes the normalized cut value. The corresponding partition is accepted.

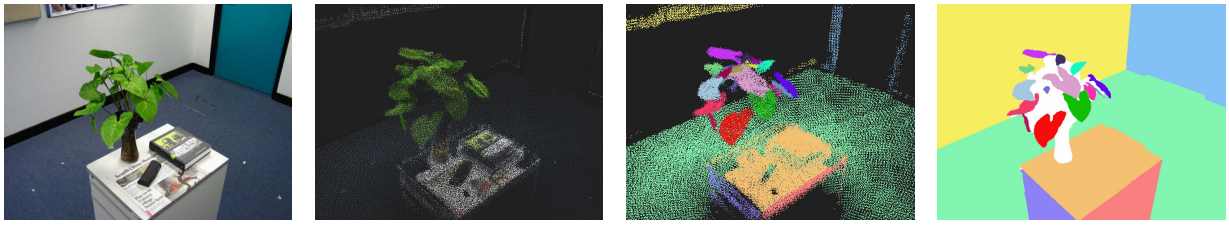
5. Recursively repeat step 3 and 4 for each partition until the normalized cut value is larger than 0.06.

6. Perform MPPCA to refine the 3D segments for the foreground points.

7. Segment each image by optimizing Eqn. (3.11) using graph cut algorithm in [21] given segmented 3D points. And we fix $\eta = \infty$, $\rho = 1$ and $\zeta = 2.5$.

8. Fitting a specific geometric representation to each group, more details are given in the following paragraph.

Segmentation-based Modeling Each segmented group should be further processed to build an appropriate geometric representation as an object.



(a) (b) (c) (d)

Figure 3.13: (a) One of the original images. (b) The 3D points. (c) The joint segmentation results. (d) One of the image segmentations by propagation.

- If a group is representing a regular geometric object, for instances, a plane, a polyhedron, or a cylinder, it is straightforward to use some standard methods of fitting these well-defined geometric models to the given data.

- If a group is representing a smooth surface like a human head or a compact object, we could use a level set approach that integrates all joint points, image information and the object boundaries to build an implicit surface model [72]. Alternatively, we used a graph-cut approach that builds a surface model with more details but with higher computational cost [131].

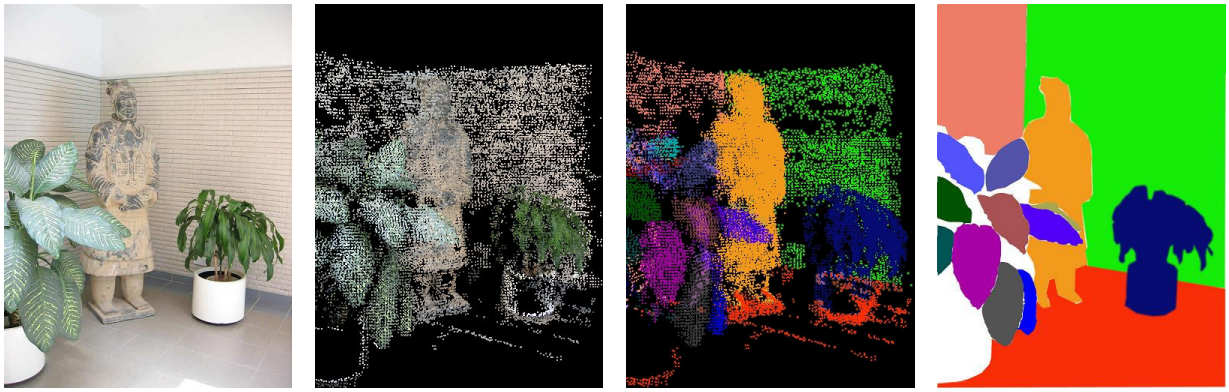
- If a group is representing the specific hair of a given person, a combination of synthesis and analysis method could be used to reconstruct each hair fiber as a curve represented as a set of connected line segments by following the edge orientation in the images [118].

- If a group is representing an individual leaf on a plant, then we can build a generic leaf model for each plant, and we have developed a method of fitting a generic deformable model to the data in [86].

We did not develop any specific fitting method in this paper, we used the combination of the above mentioned methods to build the final models illustrated in Figs. 3.4 and 3.17.

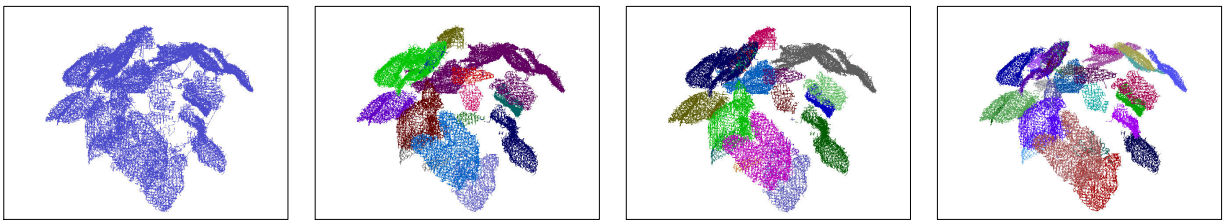
Results For the example scenario shown in Fig. 3.4, the joint segmentation results are given in Fig. 3.13. The walls, the small desk, and individual leaves have been successfully segmented, whereas the book and the eyeglasses box are not due to lack of discriminant reconstructed 3D points associated with them. These automatic results are then edited, using a semi-automatic system as described in [86], to create the final model of the scene rendered by Maya in Fig. 3.17.

Another example is shown in Fig. 3.14, which segments out the different walls, the statue, and the leaves of the plant. The second smaller plant is segmented out as a whole, not into leaves and flower



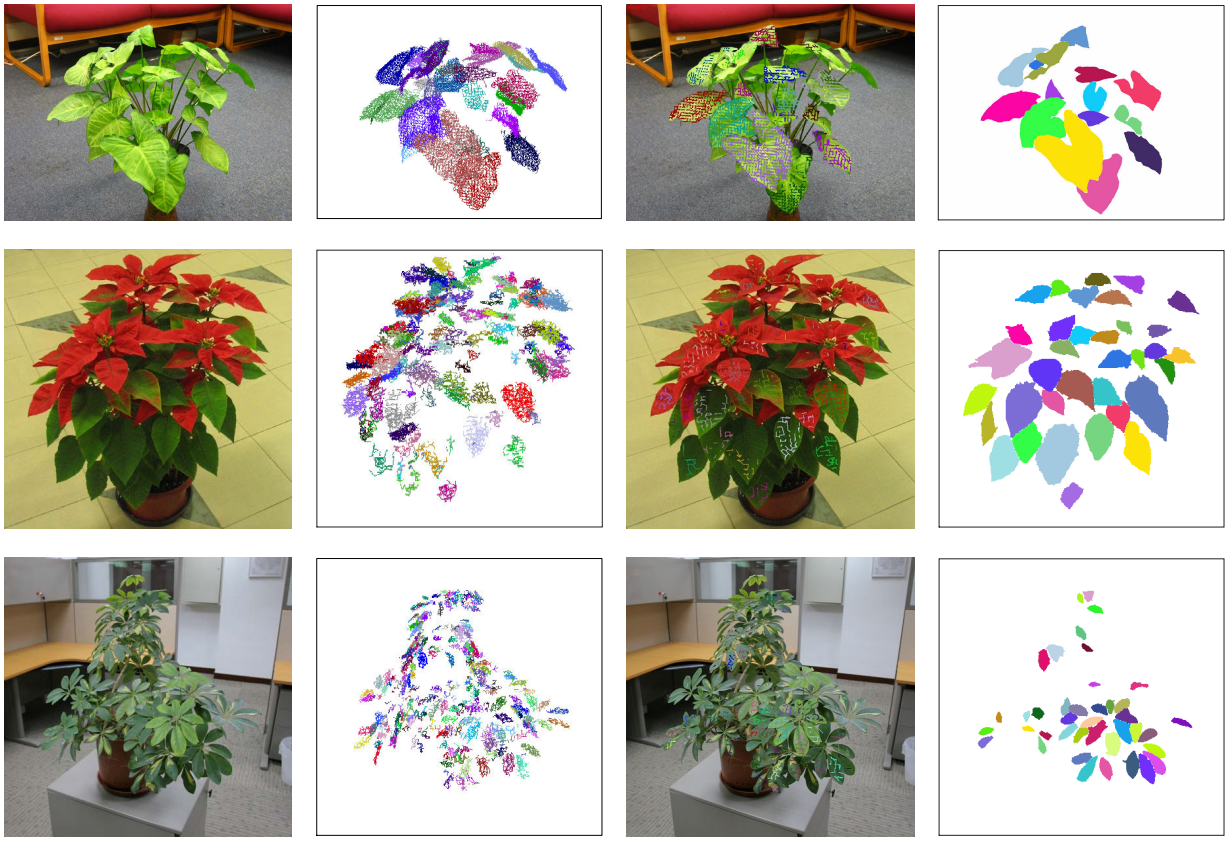
(a) (b) (c) (d)

Figure 3.14: (a) One of the original images. (b) The 3D points. (c) The segmented 3D groups by 3D segmentation. (d) One of the image segmentations by propagation.



(a) (b) (c) (d)

Figure 3.15: Two intermediate 3D segmentation results of the nephthytis example to illustrate the successive splitting of larger groups into the smaller ones.



(a) (b) (c) (d)

Figure 3.16: The first row is the nephthytis plant, the second is the poinsettia plant, and the third is the schefflera plant. (a) One of the original images. (b) Joint segmentation shown for 3D points, each group is coded with a different color. (c) Joint segmentation shown for the visible 2D points at one view. (d) Propagated image segmentation.

pot. This is due to that the same scale, and therefore the same parameters, is used for the whole scene; it will be possible by changing the setting of the parameters.

For the difficult segmentation of a plant into leaves, when the leaves of the plant tend to be sufficiently large, as is the case of the nephthytis as shown in the first row of Fig. 3.16, the segmentation becomes relatively easier since the spatial distance is larger between the points of different leaves, and the image contour becomes strong as well. The segmentation results are excellent since all visible leaves have been successfully partitioned. The intermediate results of the recursive strategy in step 3-4 of the above procedure are shown in Fig. 3.15. When the leaves of the plants become smaller, as in the cases of the poinsettia and schefflera shown in Fig. 3.16, the occlusion between leaves makes the segmentation more difficult, and therefore the success rate is lower than for the nephthytis example. The statistics of the results are reported in Tab. 3.1.

The segmentation results are not yet as perfect as we may expect, but fortunately many small

Figure 3.17: One of the original images of the schefflera plant on the left. Rendered at a similar viewpoint of the reconstructed full model of the plant on the right.

	nephthytis	poinsettia	schefflera
# images	35	35	40
# total 3D pts	128,000	103,000	118,000
# foreground pts	53,000	83,000	43,000
# segmented groups of the foreground	29	97	343
ground truth of # leaves	30	≈120	≈450

Table 3.1: The statistics for the 3D segmentation results on the three plants jointly using 3D and 2D information. The last row gives a rough estimate of the true number of the leaves for each plant by inspection.

objects such as individual leaves on the various plants have been successfully segmented out. The segmentation results have been successfully explored in a semi-automatic modeling approach in [86] to produce realistic 3D models of the plants as shown in Fig. 3.17. Of course, the joint segmentation approach is general, not restricted to plants, and geometric fitting after the segmentation might be different for different types of objects.

3.6 Conclusion

This chapter describes a workable approach to segment the 3D points and 2D image jointly. Given the availability of both 3D point data and 2D image data, we proposed a joint segmentation approach that is formulated within a probabilistic framework. This segmentation is based on the conventional pairwise graph partitioning with well-defined affinities, which makes the best use of the current states of the art including spectral graph partitioning and combinatorial graph cuts. The results obtained

on real data could be explored in an interactive modeling approach. Future directions include the development of more efficient computation methods of the proposed joint segmentation formulation.

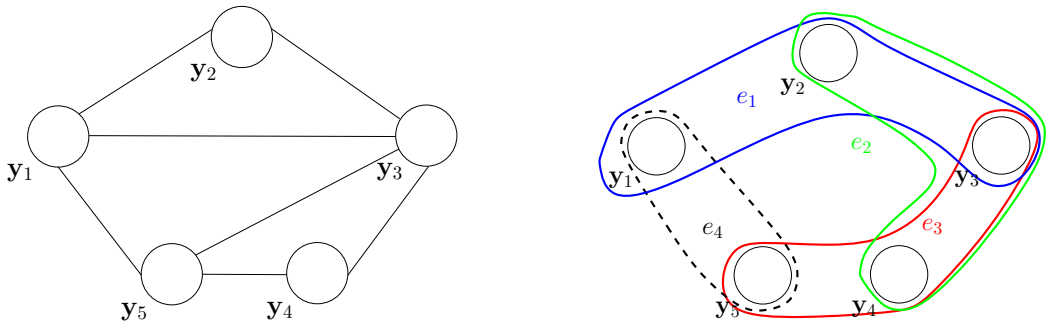
Chapter 4

Hypergraph: Linear Neighborhood Propagation

In this chapter, we address the graph partitioning problem on the direction of considering multiple-wise relation instead of pairwise relation as depicted in Fig. 4.1. We propose a new biased partitioning approach by separating a hypergraph constructed on the data points. It intends to propose a good model by considering multiple-wise relation between the data points and hence capturing high-order statistics. This novel approach finally results in solving a linear system that can have a differential geometry interpretation.

The proposed method is called Linear Neighborhood Propagation (LNP). Different from the traditional approaches, LNP provides a hypergraph based method by introducing multiwise edges instead of pairwise edges, and presents a method to learn the multiwise weight function. Intuitively it aims to predict the labels according to the neighbors in a linear way, which can be casted into a second order intrinsic Gaussion Markov Random field and results in a biharmonic solution. Experimental results on image segmentation and semi-supervised classification demonstrate the effectiveness and efficiency of LNP.

This chapter is organized as follows. The motivation of the approach is introduced in Sec. 4.1. The *LNP* algorithm is proposed in Sec. 4.2, and the analysis of *LNP* will be presented in Sec. 4.3. Sec. 4.4 extends LNP. The implementation is presented in Sec. 4.5. The image segmentation results are given in Sec. 4.7.



(a) Pairwise graph (b) Hypergraph

Figure 4.1: In this chapter, we consider multiple-wise relation *i.e*. the nodes in the hyperedge shown in (b)) instead of pairwise relation shown in (a). Then we partition the hypergraph to obtain the nodes separation.

4.1 Introduction

In Chapter 2, we have reviewed the graph partitioning approaches. This chapter will address the biased graph partitioning problem, and the proposed approach is quite related to the continuous relaxation optimization methods [139, 132], which are popular in semi-supervised learning. Although they have achieved great successes, there are several drawbacks:

1. Few of previous approaches touch the study of graph construction. Practice shows that it is more important to construct a good graph than to choose a good inference method. This is also pointed in [137]. Here, the graph construction consists of two folds. One is graph structure itself. But graph structure construction is difficult because there is less information for learning the structure as in general graphical model learning. One hot topic is graph approximation using tree structures. But almost all the methods still tries to construct pairwise graph structure. In this paper, we will consider multiple-wise relation. The other one is the relation function, i.e, the regularizer. Almost all the approaches just set the relation function to be Radial Basis function (RBF), whose results are sensitive to the parameter setting, and one example is shown in Fig. 4.2. It is expected to be ideal that the relation function is learned from the data.

2. The penalty weight of the loss function for different data is homogeneous in nearly all the approaches except [58]. Homogeneous penalty is much sensitive to noise. A more robust way is to take a inhomogeneous weight with less confidence on possible noisy data.

4.1.1 Hypergraph prior

In this chapter, we try to touch the less-studied area: graph structure, regularization function and inhomogeneous penalty weight. Moreover, in order to clearly relate the graph based approach for

semi-supervised learning to statistics inference, we also reviewed the probabilistic framework of semi-supervised learning.

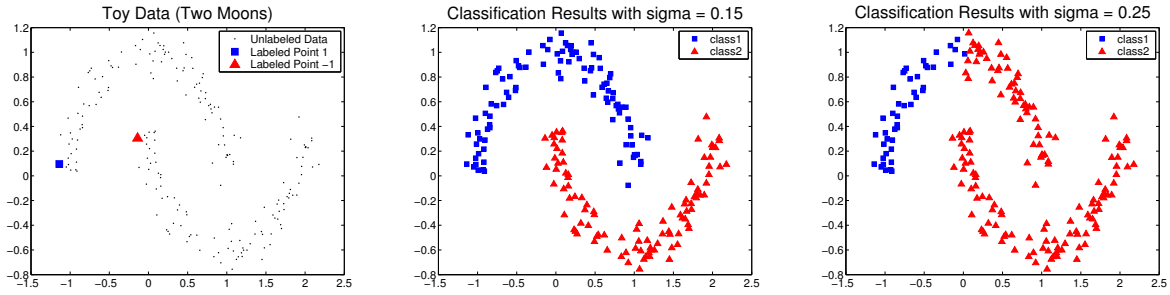
To address the above issues, we propose a novel method called *Linear Neighborhood Propagation* (*LNP*) under the second order intrinsic Gaussian Markov Random Fields. *LNP* first build a hypergraph with a series of overlapped linear neighborhood patches being the hyperedges, and the weights on each hyperedge can be solved by a standard quadratic programming procedure. Then all the edge weights will be aggregated together to obtain a weight matrix of the whole graph. We prove theoretically that the "aggregated" matrix is essentially an approximation to the *biLaplacian* matrix of a standard weighted undirected graph. Therefore, this approximated *biLaplacian* matrix can be used as a smoothness matrix as in standard graph-based semi-supervised learning algorithms. We also give an easy method for extending *LNP* to out-of-sample data points. Finally the experiments on digits and text classification are presented to show the effectiveness of *LNP*.

In this paper, we propose a novel semi-supervised learning approach. It is adopted to almost all the applications of existing approaches. Our approach however offers the following improvements over existing methods:

1. We build multiple wise graph instead of pairwise graph. To be convenient, we use bipartite graph to show their difference. In traditional pairwise relation, the function node links at most two data nodes while the function node in multiple wise case can link any number of data nodes. This multiple-wise relation provides a way to relate one data to more data together, and hence is potentially more powerful.

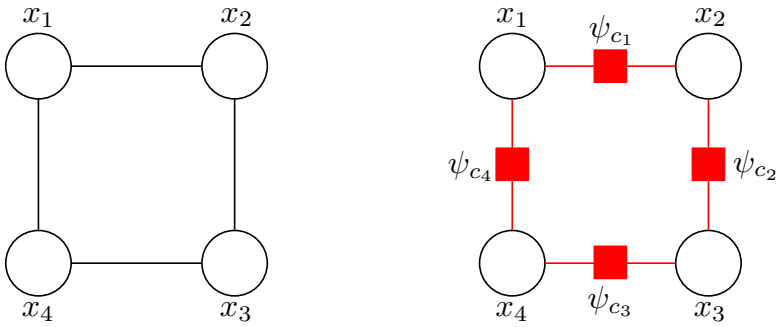
2. Generally the multiple wise relation function can be arbitrary. The relation function should reflect the cluster assumption as mentioned. It is more practical that there also exists an efficient and effective optimization approach. Here we express the relation function using a linear construction function, which is very similar to locally linear embedding (LLE) developed in recent years [90]. However we constrain the weights to be nonnegative to guarantee that the learnt weights are meaningful and reflect the relation between points.

3. Some theoretical analysis is presented to compare our approach with traditional graph based methods. Traditional approach essentially tries to find a harmonic function $\mathbf{f}$ such that $\mathbf{\Delta f} = 0$. But our goal is to find a function $\mathbf{f}$ to minimize the norm-2 value of $\mathbf{f}$, which is more relaxative and potentially more powerful.



(a) (b) (c)

Figure 4.2: Classification results on the two-moon pattern using the method in [132] with the edge weights computed by a Gaussian function. (a) toy dataset with two labeled points; (b) classification results with $\sigma = 0.15$; (c) classification results with $\sigma = 0.25$. It can be seen that a small variation of σ will cause a dramatically different classification result.



(a) (b)

Figure 4.3: A sample Markov Random Field. (a) shows a simple first-order MRF. (b) shows its factor graph representation.

4.1.2 Preliminaries

Markov Random Field A Markov Random Field, also known as an undirected graphical model or a Markov network, is a graph in which the nodes represent variables and arcs represent compatibility constraints between them. Assuming all the probabilities are nonzero, the Hammersley-Clifford theorem guarantees that the probability distribution can be factorized into a product of compatibility functions of the maximal cliques of the graph. Denoting by $\mathbf{x}$ the values of all variables in the graph, the factorization has the form:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_c \psi(\mathbf{x}_c), \tag{4.1}$$

where $\mathbf{x}_c$ is a subset of $\mathbf{x}$ that form a clique in the graph, and Z is a normalization constant.

As an example, Fig. 4.3(a) shows a first-order MRF, in which there includes four cliques. And an MRF can also be represented as a factor graph as shown Fig. 4.3(b) by adding a function node for each clique.

Gaussian Markov Random Field A Gaussian Markov Random Field (GMRF) is an MRF in which the joint distribution is Gaussian. Mathematically, the compatibility function $\psi(\mathbf{x}_c)$ in Eqn. (4.1) is an exponential as

$$\psi(\mathbf{x}_c) = \exp\{-E(\mathbf{x}_c)\}$$

with

$$E(\mathbf{x}_c) = \frac{1}{2}(\mathbf{x}_c - \boldsymbol{\mu}_c)^T \mathbf{Q}_c (\mathbf{x}_c - \boldsymbol{\mu}_c).$$

Thus the joint probability, $p(\mathbf{x})$, is Gaussian as follows:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{Q} (\mathbf{x} - \boldsymbol{\mu})\}, \tag{4.2}$$

where $Z = (2\pi)^{-n/2} \|\mathbf{Q}\|^{1/2}$, $\boldsymbol{\mu}$ is the mean and $\mathbf{Q}$ is the precision matrix, i.e. the inverse covariance matrix.

Entry $\mathbf{Q}_{ij} = 0$, means that there is no edge between nodes i and j, or that $\mathbf{x}_i$ and $\mathbf{x}_j$ are conditionally independent. The GMRF, as shown Fig. 4.3(a) must have the following form of its precision matrix:

$$\mathbf{Q} = \begin{bmatrix} \times & \times & & \times \\ \times & \times & \times & \\ & \times & \times & \times \\ \times & & \times & \times \end{bmatrix}.$$

Intrinsic Gaussian Markov Random Field A GMRF is called an intrinsic Gaussian Markov Random Field (IGMRF) if precision matrix $\mathbf{Q}$ is not of full rank [92]. Suppose $\mathbf{Q}$ is the Laplacian matrix of the graph in Fig. 4.3(a) as the following:

$$\mathbf{Q} = \mathbf{L} = \begin{bmatrix} 1 & -0.5 & 0 & -0.5 \\ -0.5 & 1 & -0.5 & 0 \\ 0 & -0.5 & 1 & -0.5 \\ -0.5 & 0 & -0.5 & 1 \end{bmatrix}.$$

It is well-known that Laplacian matrix $\mathbf{L}$ is of rank $n - 1$. The IGMRF with the Laplacian matrix or its variants as the precision matrix has wide applications in real problems.

4.2 Hypergraph Partitioning via Linear Neighborhood Propagation

Data labeling has been a traditional problem in machine learning and pattern recognition, such as unsupervised, semi-supervised, and supervised classification. Many computer vision problems, such

as stereo matching, segmentation, colorization, and matting, can be formulated into data labeling framework. In the following, we will propose an intrinsic GMRF based method, called Linear Neighborhood Propagation.

4.2.1 Problem Formulation

Suppose there are n data points $\{\mathbf{x}_i\}_{i=0}^{n}$, and l points $\{\mathbf{x}_i\}_{i=0}^{l}$ are labeled as $\{\bar{y}_i\}_{i=0}^{l}$. The task is to assign labels $\{y_i\}_{i=l+1}^{n}$ to other points $\{\mathbf{x}_i\}_{i=l+1}^{n}$. Let $u = n - l$ be the number of unlabeled points. We organize the points in the form of a graph $G = (V, E)$ with nodes V corresponding to the n data points, with nodes $V = L \cup U$, and $L = \{1, \cdots, l\}$ corresponding to the labeled points with labels $y_1, \cdots, y_l$, and $U = \{l+1, \cdots, l+u\}$ corresponding to the unlabeled points. Here E is the edge set $E = \{e_1, \cdots, e_k\}$, with e_i a vertex/point set representing an edge. In the following, we will formulate the problem in the intrinsic Gaussian Markov Random field framework by defining the edge set E.

4.2.2 First-order IGMRF

An IGMRF is called first-ordered if the precision matrix $\mathbf{Q}$ is of rank $n-1$ where $\mathbf{Q1} = 0$. Here, we consider the construction of IGMRFs of first order. Let's suppose each edge e in E is presented by a pair of vertexes $\{i, j\}$. We define a normal independent "increment" for each edge:

$$y_i - y_j \sim \mathcal{N}(0, 1/(w_{ij})), \tag{4.3}$$

where w_{ij} is the edge weight. Then the joint probability becomes:

$$p(\mathbf{y}) \propto \exp(-\sum_{(i,j)} w_{ij}(y_i - y_j)^2). \tag{4.4}$$

By this definition, the precision matrix $\mathbf{Q}$ is just the *combinatorial Laplacian* $\mathbf{\Delta}$, given in matrix form as $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ where $\mathbf{D} = \text{Diag}(d_i)$ is the diagonal matrix and $d_i = \sum_j w_{ij}$, and $\mathbf{W} = [w_{ij}]$ is the weight matrix. It can be easily verified that $\mathbf{\Delta 1} = 0$. From Eqn. (4.4), we can easily get the dependent expectation

$$\mathbf{E}(\mathbf{y}_u|\mathbf{y}_l) = -\mathbf{Q}_{uu}^{-1}\mathbf{Q}_{ul}\mathbf{y}_l, \tag{4.5}$$

where $\mathbf{Q}_{uu} = [\mathbf{Q}_{ij}]_{i\in U, j\in U}$ is a submatrix of $\mathbf{Q}$, and $\mathbf{Q}_{ul}$ is similarly defined. And in the following, such submatrices are similarly defined. It should be noted that the dependent expectation of $\mathbf{y}_u$ is also the mode (maximum) of the dependent probability $p(\mathbf{y}_u|\mathbf{y}_l)$. In essence, the approaches in [139, 132, 48, 49] can be casted into this first order IGMRF framework.

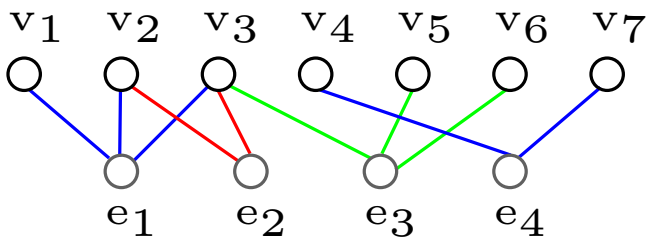


Figure 4.4: A bipartite graph representation of hyper graph. e represents a hyper edge, and the vertexes connecting it form a hyper edge.

4.2.3 Second-order IGMRF

Direct construction of a second-order IGMRF is a little difficult. We start from the intuition of the label smoothness of neighboring points. We did not directly construct IGMRF from the pairwise graph, i.e. each edge e consists of two vertexes. Instead we construct another graph $G' = (V, E')$ in which V corresponds to all the points similarly in the pairwise graph G, while each edge e in E' is a hyperedge that may include more than two vertexes. Particularly we construct the hyperedge by each vertex and its neighbor vertexes. Hence we index each hyperedge as e_v using its center vertex v, and e_v is a set of v and its neighbors N_v. A graph with hyperedges is called a hyper graph. It cannot be easily depicted in a similar way to pairwise graph since a hyperedge is not easily illustrated. We use a pairwise graph with augmented vertexes by introducing additional vertexes to represent hyperedges and placing an (pairwise) edge between a hyperedge point and the vertexes in the hyperedge. An example is shown in Fig. 4.4.

Similarly, we define an increment for a hyperedge,

$$\sum_{i \in N_v} w_{vi} y_i - y_v, \tag{4.6}$$

where w_{vi} is the normalized weight such that $\sum_{i \in N_v} w_{vi} = 1$, and it should reflect the local similarity of the labels. We suppose the increment satisfies an i.i.d. normal distribution:

$$\sum_{i \in N_v} w_{vi} y_i - y_v \sim \mathcal{N}(0, 1). \tag{4.7}$$

Then we can get the joint distribution of all the increments

$$p(\mathbf{y}) \sim \exp(-\frac{1}{2} \sum_v (\sum_{i \in N_v} w_{vi} y_i - y_v)^2). \tag{4.8}$$

The exponent part is apparently a Gaussian MRF with $\mathbf{Q} = (\mathbf{1} - \mathbf{W})^T (\mathbf{1} - \mathbf{W})$ in which $\mathbf{W} = [w_{ij}]$.

Connection to Hypergraph The second-order IGMRF is constructed based on the concept of the hypergraph by defining an energy on a hyperedge, which is a novel topic in machine learning such as

[133, 2]. This is our initial goal. Generally speaking, there is no approach to generally set the weight on the hyperedge, and, the inference on the hypergraph is often difficult. In our approach, we define a quadratic energy for a hyperedge, which finally can be reduced into a pairwise graph, and it can be easily optimized. Moreover, we provide a practical way to define and adaptively learn the parameters of the quadratic energy from the data.

4.2.4 Learning the Weights

Suppose $\mathbf{x} \in \mathbb{R}^m$, we can define an affinity between two points

$$\bar{w}_{ij} = \exp(-\sum_{d=1}^{m} \frac{(\mathbf{x}_{id} - \mathbf{x}_{jd})^2}{\sigma_d^2}), \tag{4.9}$$

where $\mathbf{x}_{id}$ is the d-th component of instance $\mathbf{x}_i$, and $\sigma_1, \cdots, \sigma_2$ are length scale hyper-parameters for each dimension. Then we normalize the weights. This method sounds fine. However, as pointed out by [132], there is no reliable approach for model selection if only very few labeled points are available, i.e. it is hard to determine an optimal σ. Moreover, we found in our experiments that even a small perturbation of σ could make the classification results dramatically different. Thus we derive a more reliable and stable way to construct the graph weight.

Let's consider the increment of a hyperedge as Eqn. (4.7) independently. In the mode, we have

$$y_v = \sum_i w_{vi} y_i. \tag{4.10}$$

This means intuitively that the label can be linearly constructed by its neighbors. We impose the cluster assumption in a linear way, i.e., the label space and the data space share the same local linear reconstruction weights.

Then we can get the linear construction weights by minimizing

$$F = \sum_i \left\| \mathbf{x}_i - \sum\nolimits_{j:\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} w_{ij} \mathbf{x}_j \right\|^2, \tag{4.11}$$

where $\mathcal{N}(\mathbf{x}_i)$ represents the neighborhood of $\mathbf{x}_i$, and w_{ij} is the contribution of $\mathbf{x}_j$ to $\mathbf{x}_i$. This objective function is similar to LLE, in which it is assumed that the low dimensional coordinates share the same linear construction weights with the high dimensional coordinates. Differently, we assume the sharing relation between the labels and the features. To avoid the negative contribution, we further constrain $\sum_{j \in \mathcal{N}(\mathbf{x}_i)} w_{ij} = 1$, $w_{ij} \geqslant 0$. Obviously, the more similar $\mathbf{x}_j$ to $\mathbf{x}_i$, the larger w_{ij} will be. Considering an extreme case when $\mathbf{x}_i = \mathbf{x}_k \in \mathcal{N}(\mathbf{x}_i)$, then $w_{ik} = 1,\ w_{ij} = 0,\ j \neq k,\ \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)$ is the optimal solution. Thus w_{ij} can be used to measure how similar $\mathbf{x}_j$ to $\mathbf{x}_i$.

It can be easily inferred that

$$\begin{aligned}\varepsilon_i &= \left\|\mathbf{x}_i - \sum\nolimits_{j:\mathbf{x}_j\in\mathcal{N}(\mathbf{x}_i)} w_{ij}\mathbf{x}_j\right\|^2 \\ &= \left\|\sum\nolimits_{j:\mathbf{x}_j\in\mathcal{N}(\mathbf{x}_i)} w_{ij}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right\|^2 \\ &= \sum\nolimits_{j,k:\mathbf{x}_j,\mathbf{x}_k\in\mathcal{N}(\mathbf{x}_i)} w_{ij}w_{ik}\left(\mathbf{x}_i - \mathbf{x}_j\right)^T\left(\mathbf{x}_i - \mathbf{x}_j\right) \\ &= \sum\nolimits_{j,k:\mathbf{x}_j,\mathbf{x}_k\in\mathcal{N}(\mathbf{x}_i)} w_{ij}G^i_{jk}w_{ik}, \end{aligned} \tag{4.12}$$

where G^i_{jk} represents the (j,k)-th entry of the local Gram matrix $(\mathbf{G}^i)_{j,k} = \left(\mathbf{x}_i - \mathbf{x}_j\right)^T\left(\mathbf{x}_i - \mathbf{x}_j\right)$ at point $\mathbf{x}_i$, where $(\cdot)_{j,k}$ represents the (j,k)-th entry of a matrix. Thus the reconstruction weights of each data object can be obtained by solving the following n standard quadratic programming problems

$$\begin{aligned}\min_{w_{ij}} \quad & \sum\nolimits_{j,k:\mathbf{x}_j,\mathbf{x}_k\in\mathcal{N}(\mathbf{x}_i)} w_{ij}G^i_{jk}w_{ik} \\ \text{s.t.} \quad & \sum_j w_{ij} = 1,\ w_{ij} \geqslant 0. \end{aligned} \tag{4.13}$$

After all the reconstruction weights are computed, we will construct a sparse matrix $\mathbf{W}$ by

$$(\mathbf{W})_{i,j} = w_{ij}. \tag{4.14}$$

The matrix $\mathbf{W}$ can represent the prior function of the bipartite graph. Intuitively, the way we construct the whole graph is to first shear the whole graph into a series of overlapped linear patches, and then pasted them together. Then, the solution can be obtained as

$$\mathbf{E}(\mathbf{y}_u|\mathbf{y}_l) = -\mathbf{Q}_{uu}^{-1}\mathbf{Q}_{ul}\mathbf{y}_l \tag{4.15}$$

by solving the linear system

$$\mathbf{Q}_{uu}\mathbf{y}_u = -\mathbf{Q}_{ul}\mathbf{y}_l. \tag{4.16}$$

4.3 Analysis and Connection

4.3.1 Biharmonic and Harmonic

The approach in [139] aims at solving a Laplacian equation $\mathbf{\Delta y} = 0$ with boundary conditions given on the labeled data, and results in a harmonic solution. In the form of differential operation, the combinatorial Laplacian operator $\mathbf{\Delta}$ sources from a continuous differential operator

$$\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{4.17}$$

In contrast, the construction of our second-order IGMRF can be interpreted from the bilaplacian (biharmonic) differential operator, i.e., the square of Laplacian operator. In the two dimensional space, the biharmonic differential operator is written as

$$(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})^2 = \frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4}. \tag{4.18}$$

It is not straightforward to directly discretize the bilaplacian operator. Instead, we have squared the discrete Laplacian operator and then obtain it. The precision matrix $\mathbf{Q} = \mathbf{\Delta}^T\mathbf{\Delta}$ is called *combinatorial BiLaplacian* operator on discrete manifold. Therefore, our approach essential aims at solve the BiLaplacian equation $\mathbf{\Delta}^T\mathbf{\Delta}\mathbf{y} = 0$. The solution of BiLaplacian equation is biharmonic function. It is shown that a harmonic function must be biharmonic, but not the other way around. This means that the local smoothness of BiLaplacian equation is more relaxant than Laplacian equation, and hence it potentially may be more suitable as the local smoothness constraint. This relaxation can be also observed from another view. The harmonic solution is the solution of $\mathbf{\Delta}\mathbf{y} = 0$, while the biharmonic solution can be viewed as the argument to minimize $||\mathbf{\Delta}\mathbf{y}||^2$.

4.3.2 Relation to Manifold Approaches

Most previous graph based semi-supervised learning approaches can be casted into the manifold regularization framework. Those approaches start from a graph with pairwise edges, and impose different regularization terms on the edges. The approaches in [9, 139] lead to a linear system based on the *combinatorial graph Laplaican*, and the approach in [132] uses the *normalized graph Laplacaian* $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. However, our approach is beyond the pairwise regularization, and defines the regularization over the hyperedges, and results in the BiLaplacian regularization. In the case that the Laplacian $\mathbf{\Delta}$ is symmetric $\mathbf{\Delta}^T\mathbf{\Delta} = \mathbf{\Delta}^2$. Our approach will be similar to the p-Laplacian operator $\mathbf{\Delta}^p$. It is also mentioned [9] that the 2-Laplacian operator works well in practice.

4.4 Extension

4.4.1 Incorporating the Bias

Often we may have label bias of the unlabeled points, which is constructed from on the labeled data alone. For example, we can build an external classifier from the unlabeled data. In this section, this can be combined into the proposed second-order IGMRF model by augmenting the nodes. For each unlabeled data v_u, we attach an "observable" node which is labeled with value the bias b_u. Similarly, we define an normal independent increment on the edge between unlabeled point and its observable

node:

$$y_u - b_u \sim \mathcal{N}(0, 1/\lambda). \tag{4.19}$$

Accordingly we revise the variance of the increments on the hyperedges to $1/(1-\lambda)$. It is easy to show that the augmented graph with the observable, labeled and unlabeled nodes and the augmented edges and hyperedges still form an IGMRF. And its precision matrix is in the form of

$$\mathbf{Q} = \begin{bmatrix} (1-\lambda)\mathbf{Q_{ll}} & (1-\lambda)\mathbf{Q_{lu}} & \mathbf{0} \\ (1-\lambda)\mathbf{Q_{ul}} & (1-\lambda)\mathbf{Q_{uu}} + \lambda\mathbf{1} & -\lambda\mathbf{1} \\ \mathbf{0} & -\lambda\mathbf{1} & \lambda\mathbf{1} \end{bmatrix}.$$

Then we can solve the augmented linear system $\mathbf{Q}[\mathbf{y}_l^T, \mathbf{y}_u^T, \mathbf{b}_u^T]^T = 0$ with $\mathbf{y}_l$ and $\mathbf{b}_u$ given. To take consideration of the data noise, we attach the confidence map $\boldsymbol{\alpha}$ for the bias then the precision matrix is transformed into

$$\mathbf{Q} = \begin{bmatrix} (1-\lambda)\mathbf{Q_{ll}} & (1-\lambda)\mathbf{Q_{lu}} & \mathbf{0} \\ (1-\lambda)\mathbf{Q_{ul}} & (1-\lambda)\mathbf{Q_{uu}} + \lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} & -\lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} \\ \mathbf{0} & -\lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} & \lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} \end{bmatrix}.$$

In the above, we assume the labeling on the labeled data is noise free, and so it is used as hard constraint. Considering the noise in the labelling, we attach a visible node to each labeled point, denoted as n_l with its value $\bar{y}_l$. Then we can easily get a new IGMRF on all the nodes with the precision matrix

$$\mathbf{Q} = \begin{bmatrix} (1-\lambda)\mathbf{Q_{ll}} + \lambda\mathbf{1} & (1-\lambda)\mathbf{Q_{lu}} & -\lambda\mathbf{1} & \mathbf{0} \\ (1-\lambda)\mathbf{Q_{ul}} & (1-\lambda)\mathbf{Q_{uu}} + \lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} & \mathbf{0} & -\lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} \\ -\lambda\mathbf{1} & 0 & \lambda\mathbf{1} & 0 \\ \mathbf{0} & -\lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} & \mathbf{0} & \lambda\text{Diag}(\boldsymbol{\alpha})\mathbf{1} \end{bmatrix},$$

where λ is the confidence on the labelling data. Then the final labeling can be obtained by $\mathbf{y} = -\mathbf{Q}_{yy}^{-1}\mathbf{Q}_{yb}\mathbf{b}$, or minimizing $\mathbf{y}^T\mathbf{Q}_{yy}\mathbf{y} + 2\mathbf{y}^T\mathbf{Q}_{yb}\mathbf{b}$.

4.4.2 Extension to Multi-label

It is easy to extend *LNP* to multi-class classification problems. Suppose there are c classes and the label set becomes $\mathcal{L} = \{1, 2, \cdots, c\}$. Let $\mathcal{M}$ be a set of $n \times c$ matrices with non-negative real-value entries. Any matrix $\mathbf{F} = \left[\mathbf{f}_1^T, \mathbf{f}_2^T, \cdots, \mathbf{f}_c^T\right]^T \in \mathcal{M}$ corresponds to a specific classification on $\mathcal{X}$ which labels $\mathbf{x}_i$ as $y_i = \arg\max_{j \leqslant c} \mathbf{F}_{ij}$. Thus $\mathbf{F}$ can also be viewed as a function which assigns the label for each data point. We use the one-to-other methodology. First, we precompute the matrix $\mathbf{P} = -\mathbf{Q}_{uu}^{-1}\mathbf{Q}_{ul}$. Second, we run c times. For the k-th time, we let the entry of $\mathbf{y}_l$ 1 if the corresponding label is k and 0 if the corresponding label is not k. Then $\mathbf{f}_k = \mathbf{P}\mathbf{y}_l^k$.

4.4.3 Induction for Out-of-Sample Data

We have introduced the transduction procedure. In this section we will propose an effective method to extend *LNP* to out-of-sample data.

According to [34], we need to do two things for generalizing *LNP* to out-of-sample data: (1) use the same type of smoothness criterion for a new testing point $\mathbf{x}$; (2) the inclusion of $\mathbf{x}$ will not affect the original $\mathcal{Q}(f)$ value of the training data set.

The smoothness criterion for a new test point $\mathbf{x}$ is

$$\mathcal{Q}(f(\mathbf{x})) = \sum_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{x})} w(\mathbf{x}, \mathbf{x}_i)(f(\mathbf{x}) - f_i)^2. \tag{4.20}$$

Since $\mathcal{Q}(f(\mathbf{x}))$ is convex in $f(\mathbf{x})$, it is minimized when

$$f(\mathbf{x}) = \sum\nolimits_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{x})} w(\mathbf{x}, \mathbf{x}_i) f_i. \tag{4.21}$$

Interestingly, this is exactly the formula when the label of $\mathbf{x}$ can be optimally reconstructed from the labels of its neighbors in the training dataset, *i.e.*

$$f(\mathbf{x}) = \min_{f(\mathbf{x})} \left\| f(\mathbf{x}) - \sum_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{x})} w(\mathbf{x}, \mathbf{x}_i) f_i \right\|^2. \tag{4.22}$$

To illustrate the effectiveness of this induction method, we also use the problem shown in Fig. 4.2(a). This is a two-class classification problem with only two points labeled, one for each class. We first adopt the standard *LNP* procedure to predict the labels of the unlabeled points, then Eqn. (4.21) is used for inducting the labels of all the points in the region $\{(x, y)|x \in [-1.5, 2.5], y \in [-0.8, 1.2]\}$. The results can be seen in Fig. 4.5.

4.5 Implementation

We have shown that there exists a closed-form solution for $\mathbf{y}$. However, it involved the calculation of the inverse of $\mathbf{Q}_{yy}$, which is often costly. Therefore we apply the iterative method to solve the linear sparse system $\mathbf{Q}_{yy}\mathbf{y} = -\mathbf{Q}_{yb}\mathbf{b}$. First, let's simplify The biLaplacian matrix $(\mathbf{1} - \mathbf{W})^T(\mathbf{1} - \mathbf{W}) = \mathbf{1} + \mathbf{W}^T\mathbf{W} - \mathbf{W}^T - \mathbf{W}$. We let $\mathbf{K} = \mathbf{W}^T + \mathbf{W} - \mathbf{W}^T\mathbf{W}$. Then, we have

$$\mathbf{Q}_{yy} = \mathbf{1} - (1 - \lambda)\mathbf{K} - \lambda(1 - \text{Diag}(\tilde{\boldsymbol{\alpha}}))\text{Diag}(\mathbf{e}) = \mathbf{1} - (1 - \lambda)\mathbf{K} - \lambda\mathbf{C}, \tag{4.23}$$

where $\mathbf{e} = [\mathbf{0}_l^T \mathbf{e}_u^T]^T$ and $\mathbf{0}_l^T$ is a zero vector with the dimension the number of labeled points, and $\mathbf{e}_l^T$ is a ones vector with the dimension the number of unlabeled points, $\tilde{\boldsymbol{\alpha}} = [\mathbf{0}_l^T \boldsymbol{\alpha}^T]^T$. Then the update equation is written as

$$\mathbf{y}^{(t)} = ((1 - \lambda)\mathbf{K} + \lambda\mathbf{C})\mathbf{y}^{(t-1)} - \mathbf{Q}_{yu}\mathbf{b}. \tag{4.24}$$

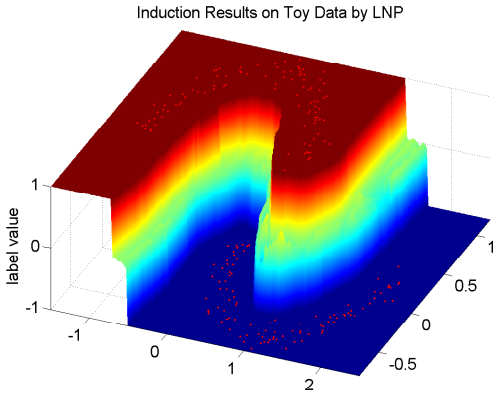


Figure 4.5: Induction results for all the data points in $\{(x, y)|x \in [-1.5, 2.5], y \in [-0.8, 1.2]\}$ using Eqn. (4.21). The red dots represent the data points in the training data set. The z-axis indicates the predicted label values associated with different colors. We can see that the induced decision boundary is intuitively satisfying since it is in accordance with the intrinsic structure of the training data set.

At iteration t, we have

$$\mathbf{y}^{(t)} = ((1-\lambda)\mathbf{K} + \lambda\mathbf{C})^t\mathbf{y}^{(0)} - \sum_{i=0}^{t-1}((1-\lambda)\mathbf{K} + \lambda\mathbf{C})^i\mathbf{Q}_{yu}\mathbf{b}. \tag{4.25}$$

It can be easily shown that the iteration is converged at $\mathbf{y}_\infty = -(1-(1-\lambda)\mathbf{K} - \lambda\mathbf{C})^{-1}\mathbf{Q}_{yu}\mathbf{b}$ if the eigenvalues of $((1-\lambda)\mathbf{K} + \lambda\mathbf{C})$ are in $[-1, 1]$. From $\alpha \in [0, 1]$, we have that the eigenvalues of $\mathbf{C}$ lie in $[0, 1]$, and the eigenvalue range of $\mathbf{K}$ can be shown to be [-3, 1]. Then we can selecte some λ to guarantee the convergence.

4.5.1 Speed up

Sequential Updating The vector form of the iteration equation leads to a synchronous updating scheme. This means that $\mathbf{y}^{(t+1)}$ is updated only using $\mathbf{y}^{(t)}$. For fast convergence, a sequential iteration scheme can be used. The basic idea is that once one label is updated and it is immediately used to update other labels. We write the component form of the iteration:

$$\mathbf{y}_j = [((1-\lambda)\mathbf{K} + \lambda\mathbf{C})]_i\mathbf{y} - [\mathbf{Q}_{yu}\mathbf{b}]_i = [\mathbf{A}]_i\mathbf{y} + [\mathbf{b}']_i, \tag{4.26}$$

where $[\cdot]_i$ denotes the i-th row. This iteration equation results in the fact that it requires only one vector to store the label vector, and the new value overlays the old value once it is updated. This updating scheme can be interpreted as the Gauss-Seidel iteration.

Cooling Scheme In the experiment, we found that some labels may reach convergence faster and almost keep unchanged after several iterations. We design a scheme to deal with this case. We attach

each label a status variable valued as "active" or "inactive". "Active" means that the corresponding label is not convergence. "Inactive" means that the label is temporarily stable. If all the labels are inactive, the iteration reach converged, i.e. permanently stable. If one value keeps unchanged several iterations we set it inactive. And it is not updated in the successive s steps. After s steps, we re-update it and inspect whether it remains temporarily stable, and reactivate it if not. We run the iteration with the stability checking until all the labels are inactive.

Multi-grid The multi-grid approach is designed to solve linear systems fast. To use the approach, we should design the methods to construct coarser grids, prolongation (interpolation), restriction, and coarse-grid operator. In the following, we will discuss the four steps in detail.

1. **Graph Coarsening**: We use the *maximal matching* method described in [61] to coarsen the graph. The vertexes in the coarse graph corresponds to two vertexes of an edge or a single vertex in the immediately finer graph. And each vertex has a unique corresponding vertex in the coarser graph.

2. **Restriction**: The value on the vertex in the coarse graph is just the average of the corresponding vertexes in the immediate finer graph. This mapping could be represented by a transform matrix $\mathbf{T}_{fc}$ such that $\mathbf{x}_c = \mathbf{T}_{fc}\mathbf{x}_f$.

3. **Prolongation**: First, the values are copied in the finer graph directly from the coarser graph. Then, we update the value for each vertex according to its neighbors. This mapping could be represented by a transform matrix $\mathbf{T}_{cf}$ such that $\mathbf{x}_f = \mathbf{T}_{cf}\mathbf{x}_c$.

4. **Coarser-grid Operator**: We denote $v_{f(i)}$ as the vertex set in the finer graph corresponding to vertex i in the coarser graph. Then for edge (i, j) in the coarser graph we find the edges linking $v_{f(i)}$ and $v_{f(j)}$ in the finer graph. And we average the weights on all the edges to get the weight on the edge in the coarser graph.

4.5.2 Relation to Other Algorithms

Connection to Mean Field The "mean field" approach is a variational method to a complicated probabilistic inference problem, which is usually difficult. Basically, the "mean field" aims to approximate a probability $p(\mathbf{x})$ using a factored probability

$$q(\mathbf{x}) = \prod_i q_i(\mathbf{x}_i). \tag{4.27}$$

It can be shown that there is a closed form solution

$$q_i(\mathbf{x}_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \mathbf{Q}_{ii}^{-1}) \tag{4.28}$$

if $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. From this sense, the solution of our approach is equivalent to the mean field approach. it was shown in [138] that the iteration of Eqn. (4.25) is an approximation to the mean field iteration if we assume $p(\mathbf{x}) \propto \mathbf{x}^T\mathbf{Q}\mathbf{x}$.

Connection to Iterated Conditional Mode The *iterated conditional mode* method in [10], or ICM, is a simple iterative technique, which is simply an application of coordinate-wise gradient method, or alternating optimization approach. The idea is first to initialize all the variables. Then we take one variable at a time and find its optimum value, keeping all other variables fixed. Particularly, when solving a linear system, ICM is equivalent to the Jacobi method. For example, to solve

$$\mathbf{A}\mathbf{x} = \mathbf{c}, \tag{4.29}$$

at time t, we update the value of $\mathbf{x}_i^{(t)}$ according to the formula

$$\mathbf{x}_i^{(t)} = \frac{1}{\mathbf{A}_{ii}}\mathbf{c}_i - \sum_{j \neq i} \mathbf{A}_{ij}\mathbf{x}_j. \tag{4.30}$$

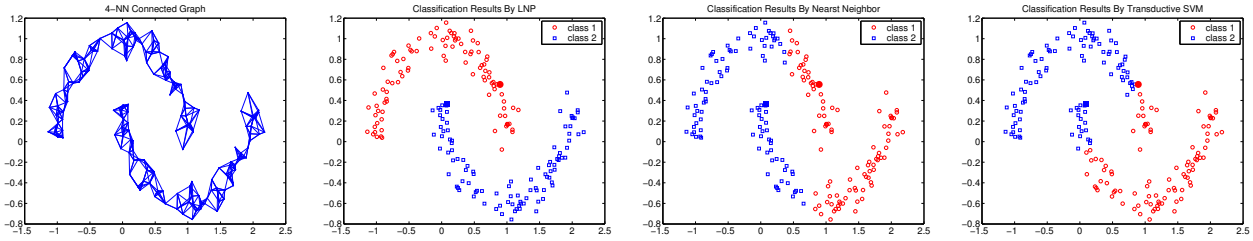
The convergence of ICM applied on solving the linear system is guaranteed if the diagonal element is dominant in the system matrix $\mathbf{Q}$. Our algorithm can be viewed as a modified version to ICM in which the convergence is controlled.

4.6 Illustration by Toy Examples

In this section, we illustrate the power of our algorithm on three toy examples. Subsec. 4.6.1 tests the basis biharmonic method as solving Eqn. (4.16). Subsec. 4.6.2 and 4.6.3 test the bias incorporated method as Eqn. (4.4.1), in which the bias element in $\mathbf{b}_l$ for labeled data is set to 1 or -1, the bias element in $\mathbf{b}_u$ for unlabeled data is set to 0, $\boldsymbol{\alpha}$ is set as ones vector, and $\lambda = 0.1$.

4.6.1 Comparison with Transductive SVM

The data set with its 4-nearest neighbor graph is shown in Fig. 4.6(a), which is also used in [132, 9]. The task is to separate the two half-circles. There are only one labeled point and 99 unlabeled points in each class. The classification results of *LNP* with the parameter k set to 4, nearest neighbor classifier using Euclidean distance and transductive SVM (*TSVM*) [57], are shown in Fig. 4.6. From the example we can clearly see that *LNP* is capable of learning the intrinsic structure of the data set, and thus can classify the data points more correctly.



(a) (b) (c) (d)

Figure 4.6: A toy example compared with transductive SVM.

4.6.2 Comparison with Graph Laplacian and Harmonic Method

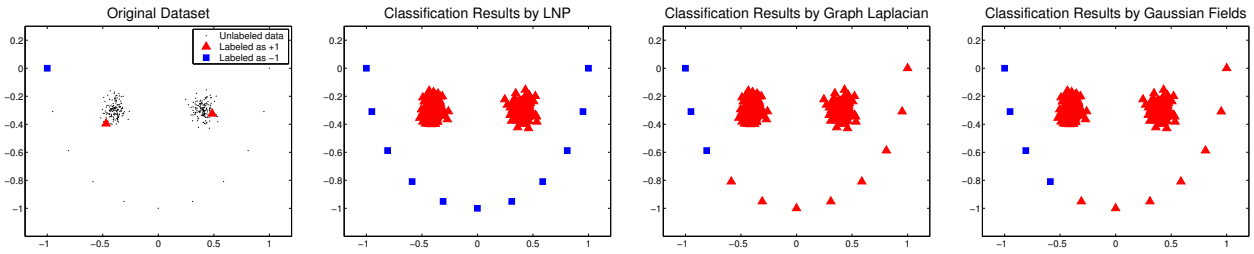
This subsection presents the comparison results with Graph Laplacian and Harmonic Method on the glass data set as shown Fig. 4.7. This data set contains 311 two-dimensional points with two hidden clusters shown in Fig. 4.7(a). The outside cluster shaped like a parabola, composed of 11 points, is uniformly sampled from a half circle with radius 1 at point $(0, 0)$. The inside cluster, consisting of 300 2D points, includes two subclusters which are generated from a mixture of two Gaussians:

$$p(\mathbf{x}) = \frac{1}{2}\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \sigma\mathbf{1}) + \frac{1}{2}\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_2, \sigma\mathbf{1}), \tag{4.31}$$

where $\boldsymbol{\mu}_1 = [-0.4, 0.3]^T$, $\boldsymbol{\mu}_2 = [0.4, 0.3]^T$ and $\sigma^2 = 0.05$.

The task is to separate the two hidden clusters by labelling partial points. Here we only manually labeled three points as shown in Fig. 4.7(a) where the point with □ is labelled as one cluster, and the two points with △ are labelled as the other cluster. We compare our approach with other two popular approaches: the standard Laplacian [8] and harmonic function [139]. The standard *graph Laplacian* is used as the smoothness regularizer [8], *i.e.* $\sum_{i,j} w_{ij}(f_i - f_j)^2$ with $w_{ij} = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)\}$, where σ is well adjusted to achieve the best result. Figs. 4.7(b), 4.7(c) and 4.7(d) show the results of the three approaches, respectively. It can be observed that *LNP* can successfully discover the latent clusters contained in the data set, while the other two methods fail to capture the outside parabola shaped cluster.

The success of *LNP* may come from that the weight matrix $\mathbf{W}$ in Eqn. (4.14) is normalized (*i.e.* $\sum_j w_{ij} = 1$, and learnt from the data. Since the density of this toy data set varies substantially across different clusters, the classification results may be affected when using non-normalized smoothness regularizer (such as the ones used in Figs. 4.7(c) and 4.7(d). This similar problem has also been addressed in [134] and an alternative symmetrically normalized smoothness regularizer was adopted.



(a) (b) (c) (d)

Figure 4.7: Transduction results on the doll toy data. The blue squares and red triangles represent two classes. (a) the original dataset, with only three data points labeled; (b) classification results by *LNP*, and the neighborhood graph is constructed with neighborhood size $k = 5$; (c) classification results using standard *graph Laplacian* as the smoothness regularizer with $\sigma = 0.15$; (d) classification results using *harmonic function* with $\sigma = 0.2$.

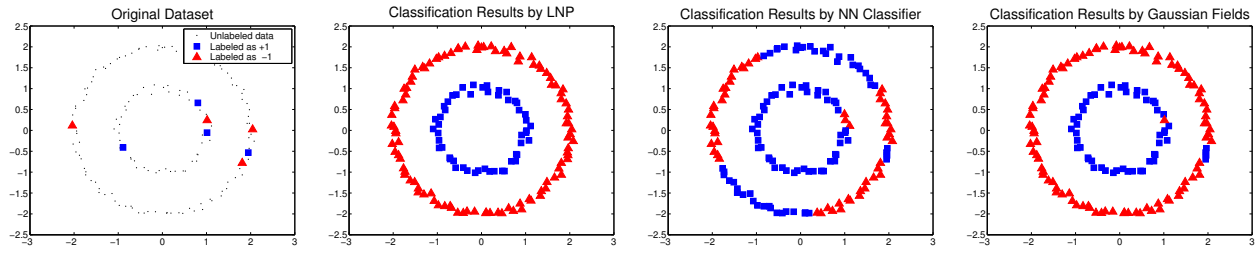
4.6.3 Robustness Comparison

Possibly due to the tiredness or careless of the annotator one may be prone to get a labeled data set with some noise. But inspecting such a data set needs extra human labor and time, hence designing a robust classifier is of vital importance. In the second toy example the robustness to such labeling noise of *LNP* is demonstrated, i.e., our approach can help to erase the labelling noise automatically. For example, Fig. 4.8(a) shows the toy data set of two circles. Ideally this toy data should be classified into two clusters: the inner circle as one class and the outer circle as the other class. Suppose the labelling is given as shown in Fig. 4.8(a), it is obvious that two labeled points are not as our common sense, which can be viewed as labelling noise. Hence it is expected that the methods can handle the noise. Figs. 4.8(b), 4.8(c) and 4.8(d) show the results of our approach, the nearest neighbor (*NN*) classifier and harmonic function, respectively. We can observe that all the points are correctly classified only by our approach. It is still difficult for *NN* to detect the noise data since it is essentially inductive. The *harmonic* method is also brittle here in that the labels of the labeled points remain fixed unchangeable during its optimization.

4.7 Application to Image Segmentation

Image segmentation is an integral part of image processing applications. Although in recent years fully automated image segmentation methods have seen great success, the segmentation results are not desirable as people expect. However, semi-automated image segmentation is much more practical, and hence has been more popular.

The semi-automated or interactive image segmentation methods borrows users' assistance to perform hard image segmentation. Representative works include graph cuts [19], random walker [48, 49]



(a) (b) (c) (d)

Figure 4.8: Transduction results on the two ringed toy data. The blue squares represent class 1, and the red triangles represent class 2. (a) the original data set; (b) classification results with *LNP*, and the neighborhood graph is constructed with neighborhood size $k = 5$; (c) classification results using the nearest neighbor classifier; (d) classification results using *harmonic function* with $\sigma = 0.28$.

and lazy snapping [73]. In essence, interactive image segmentation could be cast into the semi-supervised classification framework. Naturally the proposed approach, *LNP*, can be applied to interactive image segmentation. The user only needs to label a few pixels to respectively indicate the foreground and the background, then *LNP* automatically predicts the labels of the remaining pixels. Particularly, the neighborhood of a pixel is spatially defined to be a 5×5 patch with the pixel as the center. And each pixel is associated with a three-tuple RGB feature. The medical image and natural image segmentation results are shown in Figs. 4.9 and 4.10.

4.8 Conclusions and Future Works

In this chapter, we propose a novel semi-supervised learning algorithm called *Linear Neighborhood Propagation* (*LNP*), which is novel in constructing a hypergraph and learning the parameter adaptively. It can discover the structure of the whole dataset through synthesizing the linear neighborhood around each data object. We also analyzed theoretically that the resulted data labels can be sufficiently smooth with respect to the data structure. Finally we provide many experiments to show the effectiveness of our method, from which we also find that *LNP* also have a high parameter stability.

Appendix: Application to Transductive Pattern Recognition

4.8.1 Transductive Face Recognition

LNP is performed on the popular ORL database [93], in which there are 40 distinct faces and each object has ten gray images of size 92×112. Fig. 4.11 shows the sample face images.

For computational efficiency, all the faces were downsampled to 23×28. By comparison, other

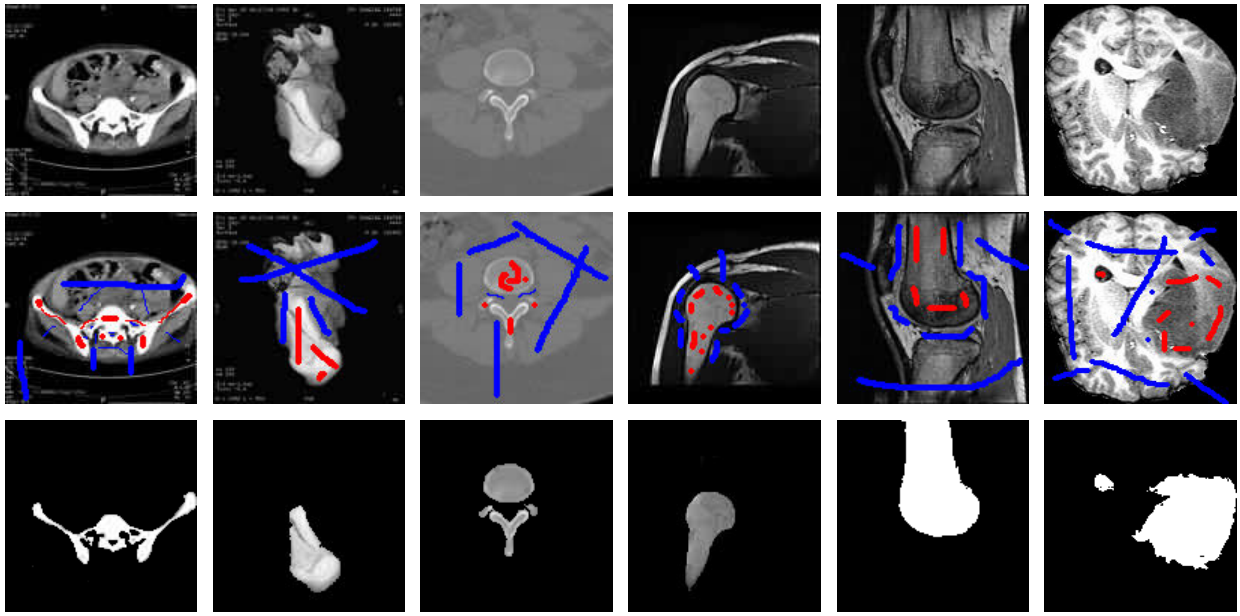


Figure 4.9: Medical images segmentation results. The first row are the original images. The second row are the partially labeled images with the red pixels representing foreground and the blue pixels representing background. The third row are the segmentation results with the background black. The original images in the first five columns are from DICOM image samples [5], and the last column is from [122].

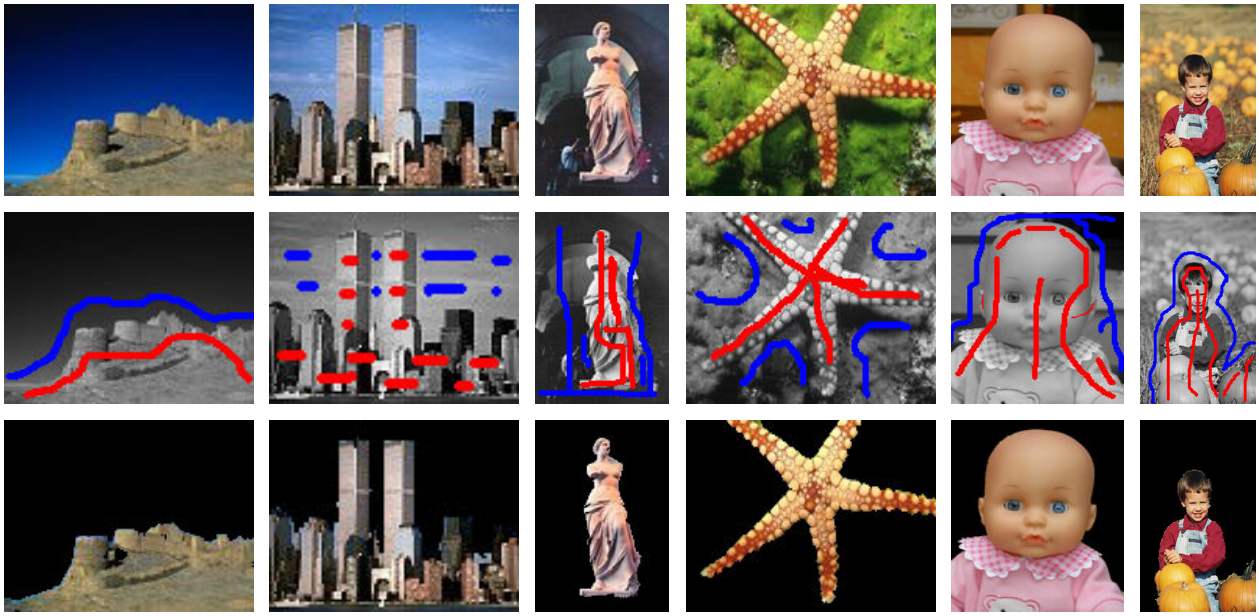


Figure 4.10: Natural color images segmentation results. The first row are the original images. The second row are the partially labeled images with the red pixels representing foreground and the blue pixels representing background. The third row are the segmentation results with the background black.

Figure 4.11: Sample faces of ORL face image data base.

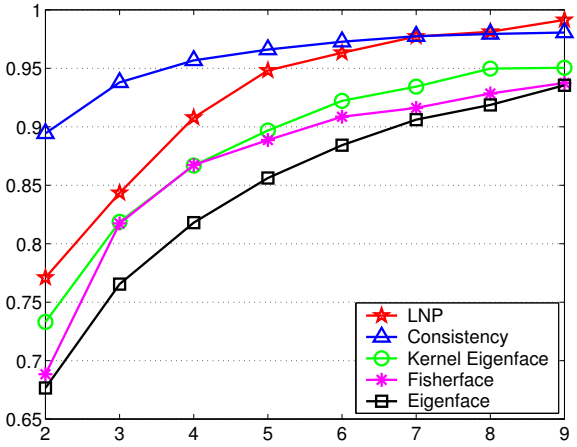


Figure 4.12: Recognition accuracies on ORL.

results, such as the *consistency* method [132] and three commonly used face recognition methods, *eigenface* [108], *fisherface* [6] and *kernel eigenface* [125], are also presented. The parameter k in *LNP* and the *consistency* method is set to 5, and the final dimensionality of the three other methods is set to 10. The variance of the Gaussian kernel in the *consistency* method is adjusted to achieve the best performance ($\sigma = 0.55$). The comparison results are illustrated in Fig. 4.12, where the horizontal axis represents the number of randomly labeled face images per subject. The vertical axis is the corresponding recognition accuracy, which is an average value of 50 independent runs. The results show that LNP outperforms traditional inductive methods and is comparable to the *consistency* method. It should be noted that the result of the *consistency* method is obtained by best tuning the parameter.

4.8.2 Transductive Visual Object Recognition

In this subsection, *LNP* is applied to visual object recognition on COIL-20 database [79], which consists of gray-scale images of 20 objects. For each object there are 72 images of size 128×128. The images were taken around the object at the pose interval of 5 degrees. Fig. 4.13 shows the 20

Figure 4.13: Sample images of COIL-20.

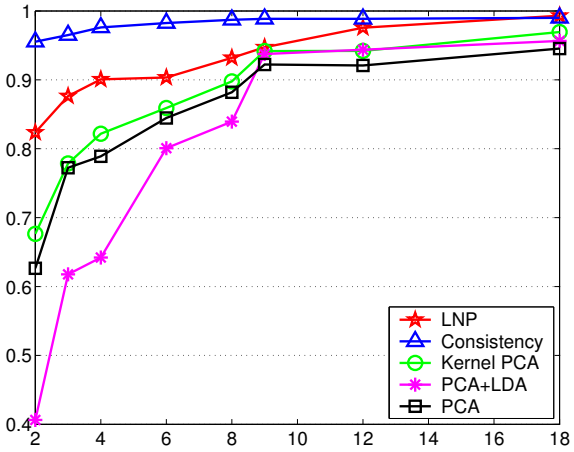


Figure 4.14: Recognition accuracies on COIL.

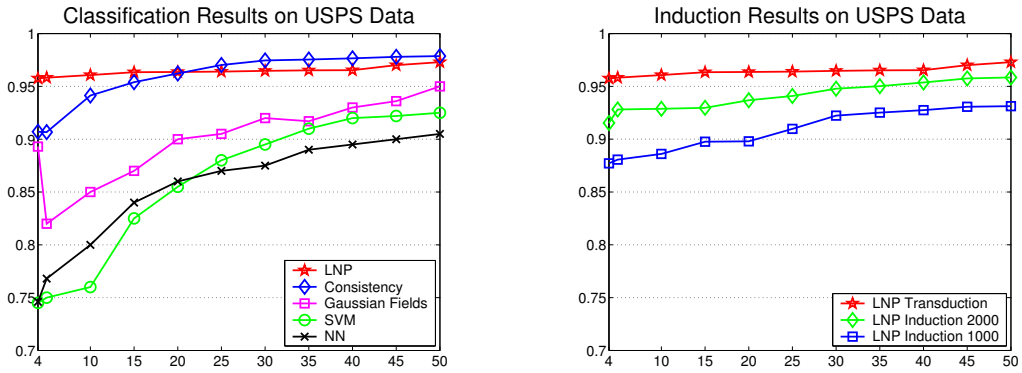
sample images.

For comparison, we also tested the recognition accuracy from three other recognition methods, Principal Component Analysis *(PCA)*, Linear Discriminant Analysis *(LDA)*, Kernel PCA and the *consistency* method. The parameter settings of all these methods are the same as in Subsec. 4.8.2 (for the *consistency* method, $\sigma = 50$). The only difference is that we select to label the images for each subject uniformly, since the images for each object are also taken uniformly around the object. The detailed procedure of the selection of the training set and description of the three methods above can be referred to [**?**]. Fig. 4.14 shows the recognition results. The results show that LNP outperforms traditional inductive methods and is comparable to the *consistency* method. It should be noted that the result of the *consistency* method is obtained by best tuning the parameter.

4.8.3 Transtuctive Digit Recognition

In this case study, we will focus on the problem of classifying hand-written digits. The data set we adopt is the USPS handwritten 16x16 digits data set [1]. The images of digits 1, 2, 3 and 4 are used in this experiments as four classes, and there are 1269, 929, 824 and 852 examples in each class, with a

[1] `http://www.kernel-machines.org/data.html`.



(a) (b)

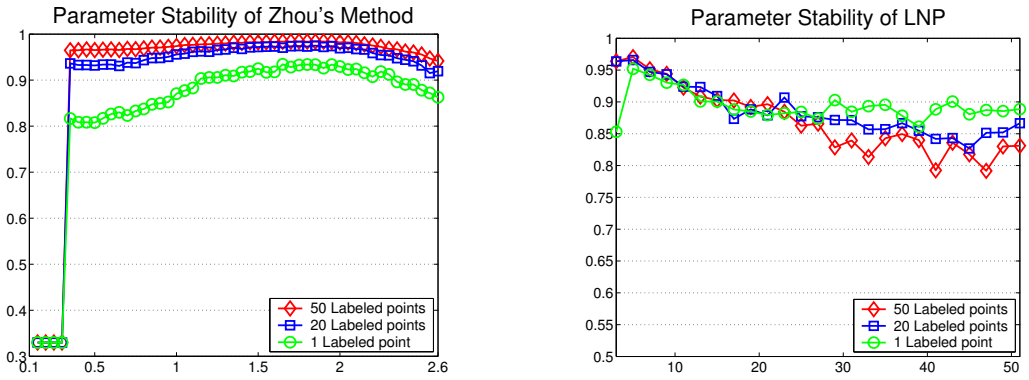
Figure 4.15: Digit recognition on the USPS dataset. A subset containing digits from 1 to 4 was adopted. (a) shows the recognition accuracies for different algorithms; (b) shows the recognition accuracies with 1000 and 2000 points selected for training, and the remaining data points for testing (induction). In both figures, the horizontal axis represents the number of randomly labeled data in the training set (we guarantee that there are at least one labeled point in each class), and the vertical axis is the total recognition accuracy value averaged over 50 independent runs.

total of 3874.

We used the *Nearest Neighbor* classifier and *one-vs-rest SVMs* [94] as baselines. The width of the RBF kernel for *SVM* was set to 5. I n *LNP*, the number of nearest neighbors k was set to 5 when constructing the graph. For comparison, we also provide the classification results achieved by the *consistency* method [134] and the *harmonic* approach [139]. The affinity matrix in both methods were constructed by a radial basis function (RBF) with variance 1.25. Note that the diagonal elements of the affinity matrix in the consistency method were set to 0. The recognition accuracies averaged over 50 independent trials are summarized in Fig. 4.15(a), from which we can clearly see the advantages of *LNP* and the consistency method.

It is interesting to discover that our *LNP* method is very stable, *i.e.* even when we only label a very small fraction of the data, it can still get a high recognition accuracy. We believe this is because the image data of different digits reside on different submanifolds (while the image data of the same digit may reside on the same submanifold [90]). *LNP* can effectively reveal these manifold structures so that only a few labeled points are sufficient for predicting the labels of the remaining unlabeled points.

We also test the effectiveness of the induction method for *LNP* introduced in Subsec. 4.4.3. First we split the whole dataset into two non-overlapping subsets, one for training (transduction), and the other for testing (induction). In the transduction phase, we simply perform *LNP* on the training set with the number of randomly labeled points varying from 4 to 50, and the resultant label matrix is used for induction on the testing set using Eqn. (4.21). Fig. 4.15(b) illustrates the induction results.



(a) (b)

Figure 4.16: Parameter stability testing results. In both figures, the vertical axis represents the average recognition accuracy of 50 independent runs, and size of the randomly labeled points is set to 1, 20, 50 respectively. (a) shows the results achieved by the *consistency* method, where the horizontal axis is the variance of the RBF kernel; (b) shows the results achieved by our *LNP* method, where the horizontal axis represents the number of nearest neighbors.

We fix the size of the training set to be 1000 and 2000, and the recognition accuracies on these two data sets are plotted by the blue and green lines. For comparison, the results achieved by standard *LNP* (*i.e.* using the whole data set for transduction) are also plotted in this figure. It is clearly observed that we can still get a high recognition accuracy using our induction method.

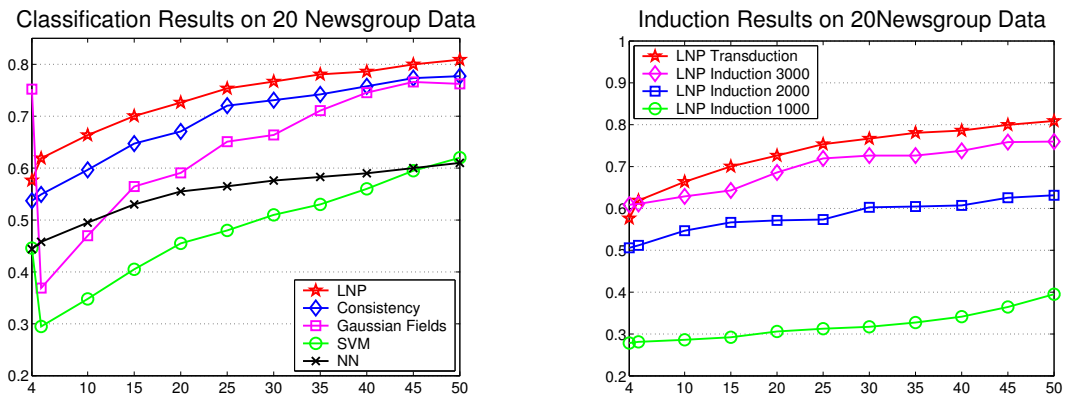
In the third part of this experiment, we studied the stability of parameter, *i.e.* the number of the nearest neighbors k in *LNP*. For comparison, we also tested the stability of the variance σ of the RBF kernel in the consistency method. The results are shown in Fig. 4.16. It seems that for this problem the consistency method is stable as long as σ is not too small, and our *LNP* method is also stable if k is not too large (since a large k will cause a confusion of different digit image manifolds). Doubtlessly, k is easier to tune since it is selected from only positive integers.

4.8.4 Transductive Text Classification

In this experiment, we addressed the task of text classification using 20-newsgroups dataset [2]. The topic *rec* containing *autos*, *motorcycles*, *baseball* and *hockey* was selected from the version 20news-18828. The articles were preprocessed by the same procedure as in [134]. The resulted 3970 document vectors were all 8014-dimensional. Finally the document vectors were normalized into *TFIDF* representation.

We use the inner-product distance to find the k nearest neighbors when constructing the neighborhood graph in *LNP*, *i.e.* $d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \mathbf{x}_i^T \mathbf{x}_j / (\|\mathbf{x}_i\| \|\mathbf{x}_j\|)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are document vectors.

[2] http://people.csail.mit.edu/jrennie/20Newsgroups/



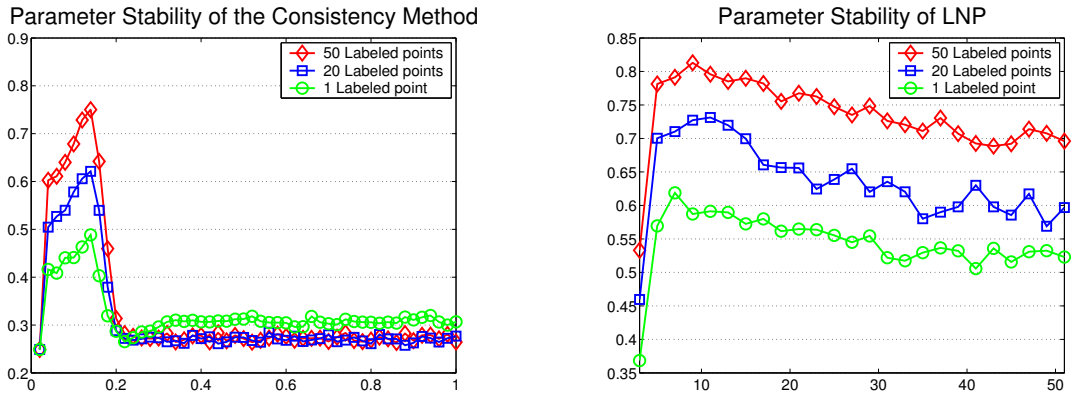
(a) (b)

Figure 4.17: Classification accuracies on 20-newsgroup data. A subset of topic *rec* was adopted. (a) shows the recognition accuracies for different algorithms; (b) shows the recognition accuracies with 1000, 2000 and 3000 points selected for training, and the remaining data points for testing (induction). In both figures, the horizontal axis represents the number of randomly labeled data in the training set (we guarantee that there are at least one labeled point in each class), and the vertical axis is the total recognition accuracy value averaged over 50 independent runs.

And the value of k is set to 10. For the *consistency* and the *Gaussian fields* methods, the affinity matrices were all computed by $(\mathbf{W})_{ij} = exp\left(-(1/2\sigma^2)(1-\mathbf{x}_i^T\mathbf{x}_j/(\|\mathbf{x}_i\|\|\mathbf{x}_j\|))\right)$ with $\sigma = 0.15$. The *SVM* and *Nearest Neighbor* classifiers were also served as the baseline algorithms, and the width of the *RBF* kernel in *SVM* is set to 1.5. The *accuracy vs. number of labeled points* plot is shown in Fig. 4.17(a), where the accuracy values are averaged over 50 independent trials. From this figure we can clearly see the effectiveness of *LNP*.

Fig. 4.17(b) illustrates the induction results of *LNP* on 20-newsgroup data. We fixed the size of the training set to be 1000, 2000 and 3000, and the remaining data were used for induction. Clearly, 1000 data points are not enough for describing the structure of the whole data set, as the classification accuracies are dramatically poor. And 3000 points are sufficient for discovering this structure in that the classification accuracies can approximate the accuracies achieved by standard *LNP* which uses the whole data set for classification.

We also test the parameter stability in the consistency and our *LNP* methods. The experimental results are shown in Fig. 4.18, where the meanings of the axes are the same as in Fig. 4.16. We may find that the consistency method is very unstable in this experiment, since it can only achieve a high classification accuracy when σ falls into a very small range (between 0.1 and 0.2). By contrast, our *LNP* method is much more stable, and it can hold a high classification accuracy as long as k is not too small.



(a) (b)

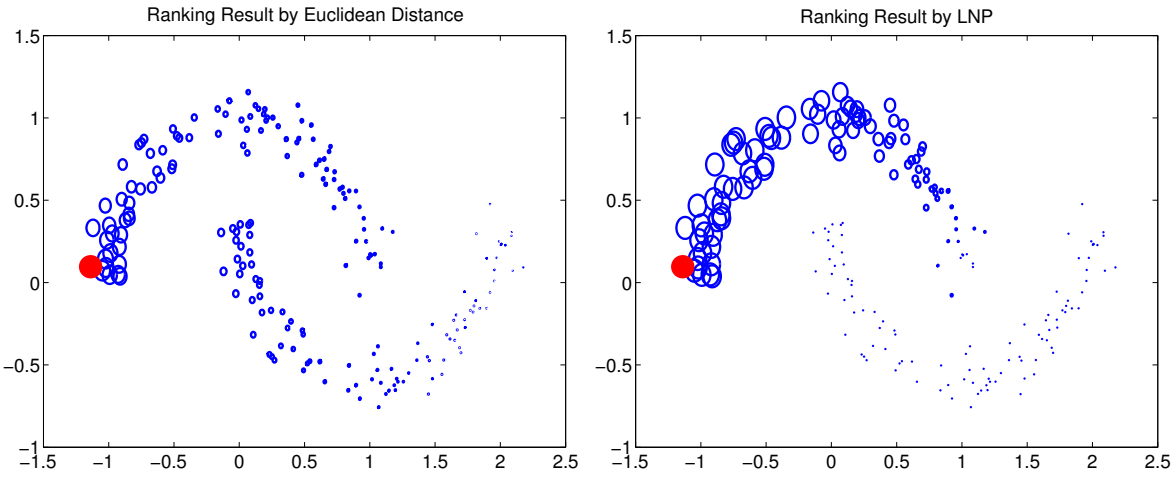
Figure 4.18: Parameter stability testing results. In both figures, the vertical axis represents the average recognition accuracy of 50 independent runs, and size of the randomly labeled points is set to 1, 20, 50 respectively. (a) shows the results achieved by the *consistency* method, where the horizontal axis is the width of the *RBF* kernel; (b) shows the results achieved by our *LNP* method, where the horizontal axis represents the number of nearest neighbors.

4.8.5 Information Retrieval

Information retrieval (*IR*) is a hot research area which has attracted considerable interests recently. In a typical *IR* system, the user raises some queries, then the computer responses by listing several ordered results according to the relevance to the queries.

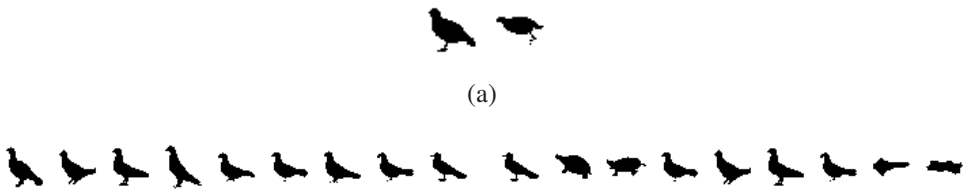
The information retrieval problem can be cast into the semi-supervised classification framework, and particularly only few positive examples are given. As an intuitive illustration, we give a toy example (which is also used in [135]) shown in Fig. 4.19. The only query is represented by a red filled circle. Fig. 4.19(a) is the retrieval result based on Euclidean distance, where the relevant scores of the data are computed by the reciprocal of the Euclidean distance between them and the query. Fig. 4.19(b) is the retrieval result using *LNP*. It is easy seen that the retrieval result by *LNP* is more consistent with our perception in that the points on the same structure have higher relevances.

We also test our approach on the image retrieval task using shape information. The database we used contains 216 different shapes [95]. Two bird shape images are randomly selected as the queries as shown in Fig. 4.20(a). Fig. 4.20(b) shows the first 9 shape images retrieved by *LNP*, and Fig. 4.20(c) shows the first 9 shape images retrieved based on Euclidean distance. Clearly the shapes retrieved by *LNP* are all quite like birds, while the shapes retrieved based on Euclidean distance contain the ones that are not bird shapes.



(a) (b)

Figure 4.19: Retrieval on a toy example.



(a)

(b) (c)

Figure 4.20: Retrieval on a shape database. (a) shows the queries, (b) shows the result of our approach, and (c) shows the result based on Euclidean distance.

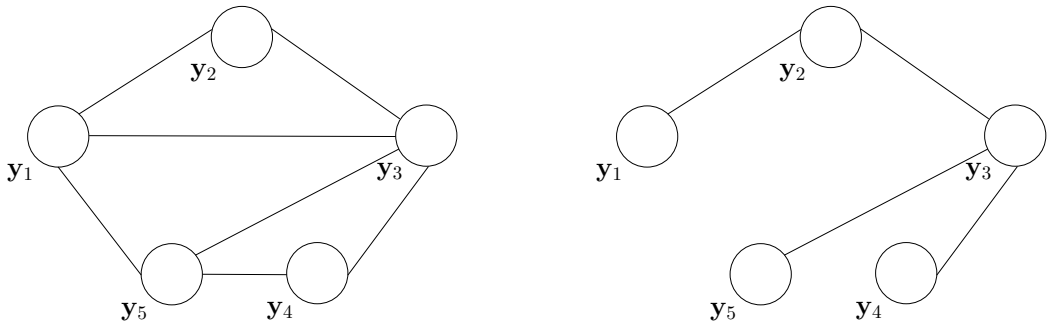
Chapter 5

Tree: Normalized Partitioning

In this chapter, we address the graph partitioning problem on the direction of simplifying a cyclic graph into a tree as depicted in Fig. 5.1. Then we propose a new unbiased tree partitioning method, normalized tree partitioning. It intends to find an efficient and effective algorithm for graph partitioning and as well reduce the approximations to potentially improve the performance.

The proposed novel method is called *Normalized Tree Partitioning*. It is a combinatorial optimization method to partition a tree with the normalized cut criterion. It has several advantages over spectral clustering. 1) It produces the exact global optimal bipartition, 2) it introduces fewer approximations and might potentially produce superior segmentation performances, 3) it is faster and runs in liner time. Then, we extend it to k-way tree partitioning by proposing a best-first recursive tree bisection scheme. Moreover, two variants, augmented tree partitioning and iterated tree partitioning, are presented. Furthermore, a novel probabilistic tree fitting approach is designed to approximate the image graph. Last, experimental results demonstrate that our approaches are more efficient and effective compared with traditional graph based approaches.

This chapter is organized as follows. Sec. 5.1 presents the previous work motivating this tree partitioning approach. Sec. 5.2 introduces a maximum likelihood framework of fitting a tree to a general distribution. Then Sec. 5.3 presents the normalized tree partitioning method. In Sec. 5.4 introduces two variants of tree partitioning. Sec. 5.5 presents the detail of fitting a tree to an image graph. The segmentation results in Sec. 5.6 demonstrate the effectiveness and efficiency of our approaches. Sec. 5.7 concludes this chapter.



(a) Pairwise graph (b) Tree

Figure 5.1: In this chapter, we simplify a cyclic graph in (a) to an acyclic graph, *i.e*. tree, shown in (b). Then we partition the tree to obtain graph partitions.

5.1 Motivation

Graph based approaches usually describe the image as a connected spares graph with pixels as nodes and edges connecting neighboring pixels. Then image segmentation can be realized by partitioning the graph. Chapter 2 has reviewed the graph based approaches. In the following, we present a short analysis of spectral clustering, which is most relevant to our work.

The basic idea of spectral clustering is relaxing discrete labels into continuous variables, and the problem can be solved using numerical methods. By spectral clustering, the original segmentation problem is transferred to an eigendecomposition problem and the eigenvectors are used for segmentation. The differences of various spectral clustering methods are cut criterions. The normalized cut [98] criterion takes the self-similarity of regions into account, and provides a significant advantage over other criterions from a theoretical and practical point of view. However, the normalized cut criterion (on grid graphs) still yields an NP-hard computational problem. So they can only obtain approximate segmentation due to spectral relaxation. Unfortunately, the approximation error is not well understood. In practice, the approximate segmentation is still hard to compute due to the expensive computation cost of eigendecomposition.

5.1.1 Tree Prior

The previous graph based approaches to image segmentation almost view the image a cyclic graph, and essentially result in solving NP problems [20, 98] except s/t graph cuts for binary segmentation. Some works in [110, 130] proposed tree based approaches, but the tree is spanned simply by minimizing the sum of intensity differences, and the cut criterions are not well adaptable for image segmentation.

In this paper, we fit a tree to the image graph and propose a probabilistic tree partitioning frame-

work for image segmentation. The main contributions are summarized as the following:

- Starting from the maximum likelihood (ML) formulation of fitting a tree distribution to a general distribution, we present a maximum negentropy spanning tree to approximate the image graph in Sec. 5.2.

- A novel *normalized tree partitioning* method is proposed. It runs in linear time and can produce the exact global optimal bipartition. A best-first recursive bisection scheme is proposed for k-way partitioning with linear time complexity. Segmentation results show its superiority over spectral clustering. This method is presented in Sec. 5.3.

- Two variants of tree partitioning are presented. Augmented tree partitioning aims to partition the tree with extra augmented nodes as shown in Fig. 5.2(b), and can be applied to interactive image segmentation. Iterated tree partitioning extends tree partitioning by combing generative clustering methods, and can efficiently be solved by the Expectation-Maximization algorithm. The two methods are detailed in Sec. 5.4.

5.2 Probabilistic Tree Fitting

This section presents a maximum likelihood solution to fitting a tree distribution to a general distribution.

5.2.1 Tree Distribution

A tree structure $\mathcal{T} = \{\mathcal{V}, \mathcal{E}\}$ is a graph that is acyclic and connected[1]. For instance, a tree structure is shown in Fig. 5.2(a). A tree distribution T is a decomposable model defined on a tree structure $\mathcal{T}$ and can be represented in terms of conditional probabilities

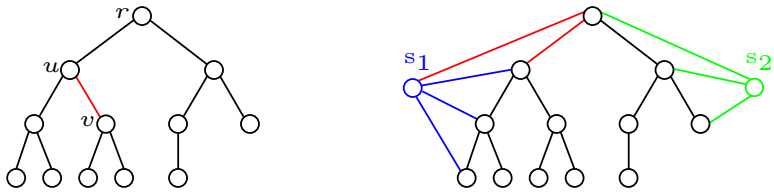
$$T(\mathbf{x}) = \prod\nolimits_{v \in \mathcal{V}} T_{v|pa(v)}(x_v | x_{pa(v)}), \tag{5.1}$$

where $pa(v)$ is the parent node of v, and $\mathcal{V}$ is the node set.

5.2.2 The Objective

Our objective is to find a tree distribution $T(\mathbf{x})$ to fit an arbitrary distribution $P(\mathbf{x})$ in the sense of maximum likelihood, which is equivalent to minimizing the Kullback-Leibler divergence between P

[1]By comparison, an acyclic and unconnected graph is called a forest.



(a) A tree (b) An augmented tree

Figure 5.2: The tree structure.

and T as mentioned in [29]. Formally it is equivalent to maximizing the logarithm of the likelihood as:

$$T^* = \arg\max_T \int P(\mathbf{x}) \log(T(\mathbf{x}))d\mathbf{x}. \tag{5.2}$$

5.2.3 Theory Derivation

Fitting a tree T to a distribution P consists of searching both the best tree structure $\mathcal{T} = \{(v, pa(v))\}_v$ and parameters $\{T_{v|pa(v)}\}$.

Parameter estimation Suppose the tree structure $\mathcal{T}$ is fixed, and expand the right-hand side of Eqn. (5.2):

$$\begin{aligned} & \int P(\mathbf{x}) \log(T(\mathbf{x}))d\mathbf{x} \\ = & \int P(\mathbf{x}) \log \prod\nolimits_{v\in\mathcal{V}} T_{v|pa(v)}(x_v|x_{pa(v)})d\mathbf{x} \\ = & \sum\nolimits_{v\in\mathcal{V}} [\int P_{pa(v)}(x_{pa(v)})dx_{pa(v)} \\ & \int P_{v|pa(v)}(x_v|x_{pa(v)}) \log T_{v|pa(v)}(x_v|x_{pa(v)})dx_v], \end{aligned}$$

where the terms that depend on T are of the form

$$\int P_{v|pa(v)}(x_v|x_{pa(v)}) \log T_{v|pa(v)}(x_v|x_{pa(v)})dx_v, \tag{5.3}$$

which is maximized by

$$T_{v|pa(v)}(x_v|x_{pa(v)}) = P_{v|pa(v)}(x_v|x_{pa(v)}) \ \ \forall v \in \mathcal{V}. \tag{5.4}$$

Intuitively, for a fixed structure $\mathcal{T}$, the best tree parameters $T_{v|pa(v)}$ in the sense of maximum likelihood are obtained by copying the corresponding conditional distributions $P_{v|pa(v)}$. This also means that the joint distribution $T_{uv} = P_{uv}$ for $\forall(u, v) \in \mathcal{T}$.

Structure optimization With the above results in mind, we now proceed to the maximization of the likelihood over the tree structures $\mathcal{T}$:

$$J(\mathcal{T}) = \int P(\mathbf{x}) \log(T^{\mathcal{T}}(\mathbf{x}) d\mathbf{x}). \tag{5.5}$$

By the similar deduction to [29], we can reach:

$$J(\mathcal{T}) = \sum\nolimits_{(u,v)\in\mathcal{T}} I_{uv} - \sum\nolimits_{v\in\mathcal{V}} H(P_v), \tag{5.6}$$

where I_{uv} is the mutual information

$$I_{uv} = \int P_{uv}(x_u, x_v) \log \frac{P_{uv}(x_u, x_v)}{P_u(x_u)P_v(x_v)} dx_u dx_v, \tag{5.7}$$

and $H(P_v)$ is the entropy that is independent of the tree. Maximizing $J(\mathcal{T})$ reduces to maximizing the first term. In other words, the goal is to find a tree such that the sum of the weights is maximized, a.k.a *Maximum Weight Spanning Tree* problem with weights $W_{uv} = I_{uv}$. Sec. 5.5 will discuss the detail to approximate the image graph.

5.3 Normalized Tree Partitioning

A tree partitioning can be defined as separating the nodes $\mathcal{V}$ into two disjoint sets, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{A} \cup \mathcal{B} = \mathcal{V}$, $\mathcal{A} \cap \mathcal{B} = \emptyset$, by simply removing edges connecting the two parts.

The cut $cut(\mathcal{A}, \mathcal{B}) = \sum_{u\in\mathcal{A}, v\in\mathcal{B}} a(u, v)$ can simply be adopted as the criterion for a partition. This criterion, as used in [130], usually produces unsatisfactory results, and results in cutting small areas out as shown in Fig. 5.3(d). The normalized cut criterion has been proved to be better, and is mathematically formulated as

$$NC = \frac{cut(\mathcal{A}, B)}{assoc(\mathcal{A}) + cut(\mathcal{A}, \mathcal{B})} + \frac{cut(\mathcal{B}, \mathcal{A})}{assoc(\mathcal{B}) + cut(\mathcal{B}, \mathcal{A})}, \tag{5.8}$$

where $assoc(\mathcal{A}) = \sum_{u,v\in\mathcal{A}} a(u, v)$ is the association. The NC criterion can have a probability interpretation that it is a normalized version of spatial coherence prior $P(L)$ [97].

Minimizing NC is NP hard in cyclic graphs such as grid graphs as analyzed in [98]. The spectral relaxation technique, although it has achieved great success in image segmentation, has at least two drawbacks: unpredictable relaxation approximation and expensive computation. And segmentation results are not always satisfactory.

In this section, we will first prove the theorem that it is sufficient to remove one edge to optimize the criterion in Eqn. (5.8), then present the algorithm, and finally give the analysis and extension.

5.3.1 Computing the Optimal Partition

Theorem: The global optimum for the normalized cut criterion must result in two connected subtrees: $\mathcal{A}$ and $\mathcal{B}$ are connected, respectively.

Analysis: The theorem is equivalent that only one edge is required to remove to obtain the global optimum for normalized cut. We prove it by contradiction.

Proof: Suppose there exists an optimum where m (> 1) edges, $\{(u_i, v_i)\}_{i=1}^m$, $u_i \in \mathcal{A}$, $v_i \in \mathcal{B}$, are removed, and this leads to $m+1$ connected subtrees $\{\mathcal{V}_j\}_{j=0}^m$. Then the normalized cut can be written as

$$\begin{aligned} NC &= \frac{\sum_i a(u_i, v_i)}{a_{\mathcal{A}} + \sum_i a(u_i, v_i)} + \frac{\sum_i a(u_i, v_i)}{a_{\mathcal{B}} + \sum_i a(u_i, v_i)} \\ &= \frac{a_{\mathcal{T}} \sum_i a(u_i, v_i)}{(a_{\mathcal{A}} + \sum_i a(u_i, v_i))(a_{\mathcal{B}} + \sum_i a(u_i, v_i))}, \end{aligned} \tag{5.9}$$

where $a_{\mathcal{T}} \equiv a_{\mathcal{A}} + \sum_i a(u_i, v_i) + a_{\mathcal{B}} + \sum_i a(u_i, v_i) \equiv assoc(\mathcal{A} \cup \mathcal{B}, \mathcal{A} \cup \mathcal{B}) \equiv assoc(\mathcal{V}, \mathcal{V})$ by definition. Consider the m possible partitions, $\{(\mathcal{A}_i, \mathcal{B}_i)\}_{i=1}^m$, corresponding to splitting one single edge from $\{(u_i, v_i)\}_{i=1}^m$. The normalized cut of $(\mathcal{A}_i, \mathcal{B}_i)$ is written as

$$\begin{aligned} NC_i &= \frac{a(u_i, v_i)}{a_{\mathcal{A}_i} + a(u_i, v_i)} + \frac{a(u_i, v_i)}{a_{\mathcal{B}_i} + a(u_i, v_i)} \\ &= \frac{a_{\mathcal{T}} a(u_i, v_i)}{(a_{\mathcal{A}_i} + a(u_i, v_i))(a_{\mathcal{B}_i} + a(u_i, v_i))}. \end{aligned} \tag{5.10}$$

Denote $\overline{NC}_i = \frac{NC_i}{a_{\mathcal{T}}}$. The following inequality holds

$$\begin{aligned} & \min(\{\overline{NC}_i\}_{i=1}^m) && (5.11) \\ \leq\ & \frac{\sum_i a(u_i, v_i)}{\sum_i (a_{\mathcal{A}_i} + a(u_i, v_i))(a_{\mathcal{B}_i} + a(u_i, v_i))} && (5.12) \\ <\ & \frac{\sum_i a(u_i, v_i)}{(a_{\mathcal{A}} + \sum_i a(u_i, v_i))(a_{\mathcal{B}} + \sum_i a(u_i, v_i))}. && (5.13) \end{aligned}$$

The inequality from Eqn. (5.11) to Eqn. (5.12) can easily be validated. The inequality from Eqn. (5.12) to Eqn. (5.13) holds when $\sum_i (a_{\mathcal{A}_i} + a(u_i, v_i))(a_{\mathcal{B}_i} + a(u_i, v_i)) > (a_{\mathcal{A}} + \sum_i a(u_i, v_i))(a_{\mathcal{B}} + \sum_i a(u_i, v_i))$, which is in detail proved in Appendix. This means that at least one partition $(\mathcal{A}_i, \mathcal{B}_i)$ has smaller normalized cut value than $(\mathcal{A}, \mathcal{B})$, and this is in contradiction to the assumption. Consequently, the theorem holds.□

With the above theorem in mind, one edge needs to be removed to optimize the normalized cut criterion. Consider the tree in Fig. 5.2(a), which is rooted from node r and denoted as $\mathcal{T}_r$, we can just remove edge (u, v), then the tree is partitioned into two parts: one, denoted as $\mathcal{T}_v$, is rooted from node v, and the other one, denoted as $\mathcal{T}_{r \setminus v}$ and called complementary subtree, is still rooted from the

original root r but excludes $\mathcal{T}_v$ and edge (u, v). For convenience, the removed edge (u, v) is used to represent such a bisection.

The (connected) tree structure only consists of $n - 1$ edges, where n is the number of the nodes. So it only takes $\Theta(n)$ time to find the optimal edge to be split by just traversing all the edges, if all the possible associations and cuts are pre-computed. Because there is only one edge linking two complementary subtrees, the cut value can be directly obtained just as the similarity of the edge. In Fig. 5.2(a), the cut value of the partition, $cut(\mathcal{T}_{r\setminus v}, \mathcal{T}_v)$, is the similarity $a(u, v)$ of nodes u and v.

Consider the tree structure as shown in Fig. 5.2(a), it is obvious that $a_{r\setminus v} = a_r - a_v - 2a(u, v)$, where $a_{r\setminus v} = a_{\mathcal{T}_r\setminus v}$ and $a_r = a_{\mathcal{T}_r}$. Hence it is sufficient to calculate association values for all subtrees $\mathcal{T}_v$. Moreover, we can observe that the association of subtree $\mathcal{T}_v$ can be calculated from the associations of the subtrees, rooted from v's child nodes, and the cut values between v and v's child nodes. Mathematically, it can be written as a recursive formulation:

$$a_v = \sum\nolimits_{w\in C_v} (a_w + 2a(v, w)), \tag{5.14}$$

where C_v represents the set of v's child nodes. By this recursion, the associations of all the subtrees can be evaluated in a bottom-up manner. In a summary, the bisection of a tree consists of two steps:

1. **Association evaluation**: recursively calculate the association a_v for subtree T_v according to Eqn. (5.14), and the association $a_{r/v}$ of subtree $T_{r/v}$.

2. **Tree traversal**: traverse all the edges to find the optimal partition.

5.3.2 Analysis and Comparison

Complexity analysis The main computation cost lies in the evaluation of all the associates, *i.e.*Eqn. (5.14). Consider the acyclic property of the tree, it means that each node has a unique parent except the root node, and apparently each node is involved only one time in the RHS of Eqn. (5.14). Consequently, the computation complexity of association evaluation is $\Theta(n)$. And it is natural that tree traversal takes $\Theta(n)$ time. Therefore the total time complexity of tree bisection is $\Theta(n)$.

Comparison with spectral clustering The basic implementation of spectral clustering in [98] takes $O(n^2)$ time, while ours takes only $\Theta(n)$. This is the first advantage of our approach over spectral clustering.

Furthermore, spectral clustering is difficult to directly obtain the segmentation result from the eigenvectors, and hence the postprocessing step, such as discretizing the eigenvectors [127] or clustering on the eigenvectors, is necessary to approximate the final segmentation. In contrast, our approach directly obtains the segmentation without any postprocess. In a word, our tree partitioning framework

introduces the approximation only once in transferring an image graph to a tree, while spectral clustering introduces at least two approximations: relaxation and discretization. This difference implies normalized tree partitioning has the potential to get superior performances over spectral clustering. Experimental results demonstrate the superiority. Fig. 5.3 shows a comparison result of our approach with spectral clustering, and it can be observed that spectral clustering wrongly groups the leaf near the tail into the dog figure as shown Fig. 5.3(c), which possibly is due to the large approximation in the clustering process, while our approach can obtain a satisfactory segmentation shown Fig. 5.3(b).

Comparison with minimum tree partitioning Compared with the approach based on minimum cut in [110, 130], we optimize the normalized cut criterion, which was shown to be better in [98], and quite adaptable for image segmentation. We implemented the algorithms in [110, 130], and the segmentation usually results in cutting small regions. In the example shown Fig. 5.3(d), the small nose is cut out with the approach in [130]. This is consistent to the analysis in [39].

5.3.3 Best-first Recursive Bisection

Multi-way partitioning of a tree is still NP-hard. We propose a best-first recursive bisection algorithm to produce multi-way partition. The recursive procedure for p-way partitioning is as follows.

1. Bisect the input tree $\mathcal{T}$ into $\mathcal{A}_1$ and $\mathcal{A}_2$. Set the number of current partitions $k = 2$.
2. Try to bisect all the current subtrees $\{\mathcal{A}_i\}_{i=1}^{k}$,
3. Find the subtree $\mathcal{A}_t$ that produces the smallest normalized cut value, denote its bisected subtrees $\mathcal{B}_1$ and $\mathcal{B}_2$, and let $\mathcal{A}_t = \mathcal{B}_1$ and $\mathcal{A}_{k+1} = \mathcal{B}_2$, increment k by 1.
4. Output the p-way partitions if $k = p$, otherwise go to step 2.

5.4 Variants of Tree Partitioning

5.4.1 Biased Image Segmentation

The s/t graph cuts approach to object segmentation is introduced in [20]. Different from spectral clustering, this approach often requires certain types of topological constraints, for instance, the user marks some pixels as the label bias. Lazy snapping [73] and grabcut [89] mainly focused on providing convenient interactivities. Some other studies in [60, 63] mainly worked on speeding up the s/t graph cuts implementation. The s/t graph cuts approach can get the global optimum for two-label

(a) Original image (b) Our approach (c) Spectral clustering (d) Approach in [130]

Figure 5.3: A comparison of segmentation results. (a) shows the original image (b) shows the result of normalized tree partitioning, which is satisfactory. (c) is the result of spectral clustering with well-tuned parameters, in which we can observe that the leaf near the tail is wrongly grouped into the dog figure. (d) shows the result of the tree based approach in [130].

segmentation. But it can only be used to approximately solve multi-label segmentation [20]. Some statistical inference approaches are alternatively used to solve multi-label problems, such as generalized Swendsen-Wang cuts in [4] and generalized belief propagation in [97].

A Gaussian Markov random field can be also used to model image segmentation. Its MAP solution leads to a quadratic programming problem, which is deduced to solving a linear sparse system. Such an approach can lead to a closed-form solution to soft segmentation [69]. And it can be extended to solve hard segmentation [48, 132, 139].

5.4.2 Augmented Tree Partitioning

Considering a cyclic graph shown in Fig. 5.2(b), we call it augmented tree since it is formed by augmenting several extra nodes $\{s_i\}_{i=1}^k$ and connecting them with nodes $\mathcal{V}$ in the tree, and denote it $\mathcal{T}' = (\mathcal{V} \cup \{s_i\}_{i=1}^k, \mathcal{E} \cup \mathcal{E}_a)$ with $(\mathcal{E}_a = \{(v, s)\})$, $v \in \mathcal{V}$ and $s \in \{s_i\}_{i=1}^k$. For convenience, Fig. 5.2(b) just draws a few augmented edges. A partitioning on such an augmented tree is defined as separating the nodes into k disjoint subsets, $\{\mathcal{V}_i \cup \{s_i\}\}_{i=1}^k$, such that $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$, $\cup_{i=1}^k \mathcal{V}_i = \mathcal{V}$, by simply removing some edges. Different from normalized tree partitioning, there are additional constraints: *i.e.*the augmented nodes must lie in different subsets. Then we adopt the minimum cut criterion to partition such an augmented tree. It can easily be validated that minimum cut is equivalent to assigning nodes $\mathcal{V}$ with labels $\{s_i\}_{i=1}^k$, which maximizes the following formula:

$$P(L|I) = \prod\nolimits_v P(s_{l_v}|l_v) \prod\nolimits_v T(l_v|l_{pa(v)}), \tag{5.15}$$

where $P(s_{l_v}|l_v)$ encodes the likelihood that node $v \in \mathcal{V}$ is connected to s_{l_v}. In our experiment, this likelihood is evaluated by learning a Gaussian mixture model (GMM) from the labeled pixels. An efficient dynamic programming procedure can be adopted to maximize Eqn. (5.15), and we follow the description in [111].

We present the dynamic programming algorithm on a tree by following the description [111]. Let $r \in \mathcal{V}$ be root node, and its depth be 0, and the depth of all other $v \in \mathcal{V}$, denoted as d_v, be the number of the edges on the shortest path from r to v. Considering each node v except root node r and its parent node $pa(v)$, it is obvious that $d_v = d_{pa(v)} + 1$. This can be easily verified in Fig. 5.2(b).

Let's start the description of the algorithm. Consider subtree T_v rooted from node v, we define a function $q_v(l_v)$ with label l_v of node v as the argument:

$$q_v(l_v) = \max_{l_*} p(l_v, l_*), \tag{5.16}$$

where l_* represents the possible labels of all the nodes in T_v except node v, and $p(l_v, l_*) = P(L_{T_v}|I_{T_v})$. If v is a leaf node, then $q_v(l_v) = p(l_v) = P(s_{l_v}|l_v)$. Therefore for a leaf v, $q_v(l_v)$ can be evaluated directly. Now we move to the internal nodes. From the Markov and acyclic properties, we obtain the recursive calculation of the posterior probability:

$$\begin{aligned} q_v(l_v) &= \max_{\{l_w, w \in C_w\}} P(s_{l_v}|l_v) \prod\nolimits_{w \in C_v} T(l_w|l_v) q_w(l_w) \\ &= P(s_{l_v}|l_v) \prod\nolimits_{w \in C_v} \max_{l_w} T(l_w|l_v) q_w(l_w). \end{aligned} \tag{5.17}$$

$q_v(l_v)$ for all the internal nodes and root node can be evaluated in a recursive bottom-up way. Suppose the maximum depth of the tree is D. The nodes with depth D are just leaves, and their posterior probabilities $q_v(l_v)$ can be directly evaluated as discussed before. With the posterior probabilities of all the leaves given, we can evaluate $q_v(l_v)$ for all the nodes with depth $D-1$ using Eqn. (5.17). Similarly, we can proceed to process the nodes in a decreasing depth order until reaching the root node.

The optimal labeling can be found in a top-down way from the root node to leaf nodes. The optimal label assignment for root node r can be written as $l_r^* = \arg\max_{l_r} q_r(l_r)$. Then we use the optimal value at root r to find the labels of its children $w \in C_r$ by replacing $\max$ with $\arg\max$ in Eqn. (5.17). This $\arg\max$ can be recorded in the process of bottom-up posterior probability evaluation. Then we can go down the tree in order of increasing depth to compute the optimal label assignment of each child node w, using the precomputed $\arg\max_{l_w}$.

Summarily, the method includes two passes on the tree:

1. **Bottom-up pass**: evaluate the posterior probabilities in a depth decreasing order starting from the leaf nodes.

2. **Top-down pass**: assign the optimal labels in a depth increasing order starting from the root node.

Analysis The dynamic procedure to maximize Eqn. (5.15) takes only $\Theta(nk^2)$ time, and consumes only $\Theta(nk)$ memories. The two are superior over graph cuts. Moreover, the dynamic programming procedure always reaches the global optimum even in multi-label cases, while graph-cuts produces the optimum only in two-label cases.

Relation Recently several inference methods, such as (loopy) belief propagation [84] and tree reweighted belief propagation [113, 64], generalize the belief propagation on a tree to on a loopy (cyclic) graph to obtain an approximate solution. Belief propagation on a tree has been proved to be equivalent to dynamic programming. Our total procedure consists of tree fitting and dynamic programming, which might introduce a bit more approximation error on finding the solution of the original loopy graph that models image segmentation. However, Our approach is quite suitable for interactive image segmentation because it can bring immediate feedbacks to the user and produce competitive performance in practice, which also validates that a good solution of the loopy graph model is not completely equivalent to a good segmentation result.

5.4.3 Iterated Tree Partitioning

Reconsidering partitioning the tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ into $\{\mathcal{V}_i\}_{i=1}^k$, it is desirable that the nodes in each partition $\mathcal{V}_i$ satisfies a generative probabilistic model such as Gaussian mixture models (GMM). This objective can be written as

$$P(\mathcal{L}, \boldsymbol{\theta}) = \prod_v P(s_{l_v}|l_v, \boldsymbol{\theta}) \prod_v T(l_v|l_{pa(v)}), \tag{5.18}$$

where $\mathcal{L} = \{l_v\}_{v\in\mathcal{V}}$, and $\boldsymbol{\theta} = \{\theta_l\}$ is the GMM parameters for each partition. Similar to [89, 129], an Expectation-Maximization (EM) algorithm is used to maximizing Eqn. (5.18). Differently, we initialize the segmentation using normalized tree partitioning, and use the augmented tree partitioning algorithm for perform the maximization step in the EM algorithm. The algorithm is outlined as follows:

1. **Initialization**: Initialize the segmentation using normalized tree partitioning.

2. **Expectation**: Estimate the GMM parameters $\bar{\boldsymbol{\theta}}$ for all partitions.

3. **Maximization**: Run augmented tree partitioning procedure, *i.e.*optimizing $P(\mathcal{L}, \boldsymbol{\theta} = \bar{\boldsymbol{\theta}})$

4. **Loop**: loop step 2 and 3, until convergence.

5.5 Tree Fitting for Image Graph

We use a first-order Markov random field to describe the image graph:

$$P(L) = \frac{1}{Z}\prod_{(u,v)\in\mathcal{E}} P(l_u, l_v), \tag{5.19}$$

where $\mathcal{E}$ is the set of all the connected neighboring pixels, $P(l_u, l_v)$ is the local prior, and Z is the normalization constant. The pairwise probability $P(l_u, l_v) \propto P(\delta_{[l_u=l_v]})$, where $P(\delta_{[l_u=l_v]})$ is a Potts model:

$$P(\delta_{[l_u=l_v]}) = \frac{1}{Z}\begin{cases} 1 & \delta_{[l_u=l_v]} = 1 \\ 1-\exp(-\lambda g(f_u, f_v)) & \delta_{[l_u=l_v]} = 0, \end{cases} \tag{5.20}$$

where $g(f_u, f_v)$ as the distance measure of pixels u and v with f_u as the RGB feature, Z is the normalization parameter, λ is set 1 by default, and $\delta_{[l_u=l_v]}$ is an indicator function.

We will use the method in Sec. 5.2 to fit a tree to the image graph. The difficulty is calculating I_{uv} in Eqn. (5.7) in that it requires that marginal and joint probabilities, P_u, P_v and P_{uv}, are precomputed, while estimating those probabilities is NP-hard in cyclic graph. In our MRF model of image segmentation, we approximate joint probability P_{uv} using smoothness prior $P(l_u, l_v)$. Without any bias considered, marginal distribution $P(l_u)$ can be simply approximated as a uniform distribution. Then the mutual information is approximated as

$$\begin{aligned} I_{uv} &\approx \sum\nolimits_{l_u,l_v} P(l_u, l_v)\log\frac{P(l_u, l_v)}{(1/M)(1/M)} \\ &= \sum_{l_u,l_v} P(l_u, l_v)\log P(l_u, l_v) + 2\log M\sum_{l_u,l_v} P(l_u, l_v) \\ &\approx \sum\nolimits_{\delta_{[l_u=l_v]}} P(\delta_{[l_u=l_v]})\log P(\delta_{[l_u=l_v]}) \\ &\quad +2\log M\sum\nolimits_{l_u,l_v} P(l_u, l_v) \\ &= -H_{uv} + 2\log M. \end{aligned} \tag{5.21}$$

Here $\log M$ is constant for any edge (u, v) with M the number of labels, hence it can be omitted. Therefore $W_{uv} = -H_{uv}$. Since the edge weight is negentropy, we call the spanning tree as *maximum negentropy spanning tree*.

In summary, the tree fitting algorithm includes two steps:

1. **Structure construction** Mutual information for all edges (u, v) is estimated according to Eqn. (5.21). Maximum negentropy spanning tree $\mathcal{T}$ can be constructed by maximum weight spanning tree algorithms, such as Prim's or Kruskal's algorithms. Generally the time complexity is $O(|\mathcal{E}|\lg|\mathcal{V}|)$. The image graph is sparse with $|\mathcal{E}| = \Theta(|\mathcal{V}|)$. Hence the time complexity is $O(|\mathcal{V}|\lg|\mathcal{V}|)$.

2. **Parameter assignment** The tree parameters $T_{v|pa(v)}$ are directly assigned as $P_{v|pa(v)}$.

Relation Fitting a tree to an image graph has also been applied to stereo matching in [111], which just intuitively finds a spanning tree based on the minimum distance criterion and has no probability interpretation. In contrast, The tree in our approach is fitted in the probability framework. Considering weight $W_{uv} = -H_{uv}$, through simple analysis of the entropy from the definition of $P(\delta_{[l_u=l_v]})$, the value of H_{uv} is a monotonically increasing function of the distance $g(f_u, f_v)$. In other words, weight W_{uv} is made to be a monotonically decreasing function of the distance. From this sense, the minimum distance spanning tree used in [111] can be roughly casted into our probabilistic tree fitting framework.

5.5.1 Prior Model on Superpixels

To make tree partitioning more practical, a graph coarsening step can be performed before tree fitting. Particularly, the image graph can be coarsened by building the graph on the superpixels of the image. This will bring at least two advantages: 1) The memory complexity of the graph is reduced; 2) The time complexities of tree construction and inference on the tree are reduced. However, this poses a new problem of defining the distance g between two superpixels.

We borrow the idea of pairwise predicate presented in [39], and define a relative external distance to internal difference as:

$$g(C_1, C_2) = \max(d(C_1, C_2)/Int(C_1), d(C_1, C_2)/Int(C_2)).$$

The internal difference is defined on a superpixel C:

$$Int(C) = \max_{(u,v)\in MST(C)} g(u, v), \tag{5.22}$$

where the maximization is done over the edges in the minimum distance spanning tree of superpixel C, $MST(C)$.

The external difference between two superpixels C_1, C_2 is defined to be the minimum weight among spatial neighboring pixels:

$$d(C_1, C_2) = \min_{u\in C_1, v\in C_2, (u,v)\in\mathcal{N}} g(u, v). \tag{5.23}$$

5.6 Experimental Results

We demonstrate the proposed tree partitioning framework on image segmentation. The results based on tree partitioning are all obtained by segmenting the superpixels. The superpixels are generated

Figure 5.4: A comparison between normalized tree partitioning and spectral clustering. The top row shows the results of normalized tree partitioning, and the bottom row shows the results of spectral clustering in [127].

by oversegmenting the image with the watershed algorithm proposed in [112]. The Markov random field is constructed by setting the superpixels as the nodes and connecting two superpixels iff they are spatial neighbors. Then the maximum negentropy spanning tree is constructed to approximate the MRF. In our experiments, the ranges for RGB value in color image are all normalized to [0, 1]. All the running times are given on 1.9 GHz Pentium 4 desktop PC.

5.6.1 Normalized Tree Partitioning

In this subsection, we present the comparison results of normalized tree partitioning with spectral clustering in [127]. We run the matlab implementation of spectral clustering, which is downloaded from `http://www.cis.upenn.edu/\~jshi/software/`. The affinity function in tree partitioning is defined as $a(u, v) = \exp[-\alpha g(f_u, f_v)]$, and in our experiments α is picked in [8, 12]. All the results with image size about 400×400 are obtained in 0.05 second with un-optimized c++ implementation after the superpixels are obtained, while the results of spectral clustering usually take over one minute.

The comparison results are shown in Figs. 5.3 and 5.4. In Fig. 5.4 the top row shows the segmentation results by our approach, and the bottom shows the results by spectral clustering. It can be observed that our results are better than spectral clustering. This is because spectral clustering involves two approximations: relaxing the discrete label to continuous value, and discretizing the continuous optima to generate the discrete label. In contrast, our approach only introduces the approximation in the tree fitting step, and is expected to get better results.

(a) Original image (b) Our approach (c) Graph cuts (d) LNP

Figure 5.5: Interactive segmentation for figure-ground separation.

5.6.2 Augmented Tree Partitioning

We test dynamic programming based tree partitioning on interactive image segmentation. We follow the user interactivity [20], and let users draw several scribbles to mask the pixels as some certain object shown in Figs. 5.5(a) and 5.6(a). Similarly, we set the masked pixels as hard constraints. To impose hard constraints, we set $P(i_v|l_v) = 0$ if l_v is not as the label indicated by the user, and otherwise $P(i_v|l_v) = 1$. Weight λ in Eqn. (5.20) is set to 5 for all the results in this subsection. We have analyzed that our approach is more efficient, and experimental results demonstrate this efficiency. The optimization with images of size 500×500, after obtaining the likelihood, requires about 0.02 second for dynamic programming (DP) while about 0.2 second for graph cuts (GC) in figure-ground separation cases. In multi-object cases, the iterative α-expansion graph cuts algorithm is adopted for optimization, which often takes more than 1 second, while DP only takes less than 0.1 second.

Figs. 5.5 and 5.6 show our results in (b) and GC based results in (c). It should be noted that all the results are obtained with the same scribbles as shown in (a). The results indicate that our approach can get satisfactory performance. And the times shown in Tab. 5.1, consumed in the optimization of DP and GC after obtaining the likelihood, demonstrate that DP based tree partitioning is more efficient.

size	470×318	481×321	574×384		
DP	0.02	0.03	0.03	0.03	0.02
GC	0.24	0.32	0.37	0.45	0.46
LNP	0.60	0.60	2.00	1.30	1.50

size	425×521	600×603	535×513	800×533
DP	0.04	0.07	0.04	0.05
α-GC	1.22	1.84	1.41	4.173
LNP	2.00	3.00	2.30	6.52

Table 5.1: A quantitative comparison of optimization times. The left table shows image sizes and optimization times (in seconds) of DP and GC for the images shown in Fig. 5.5 in the same order, and the right table corresponds to Fig. 5.6.

5.6.3 Iterated Tree Partitioning

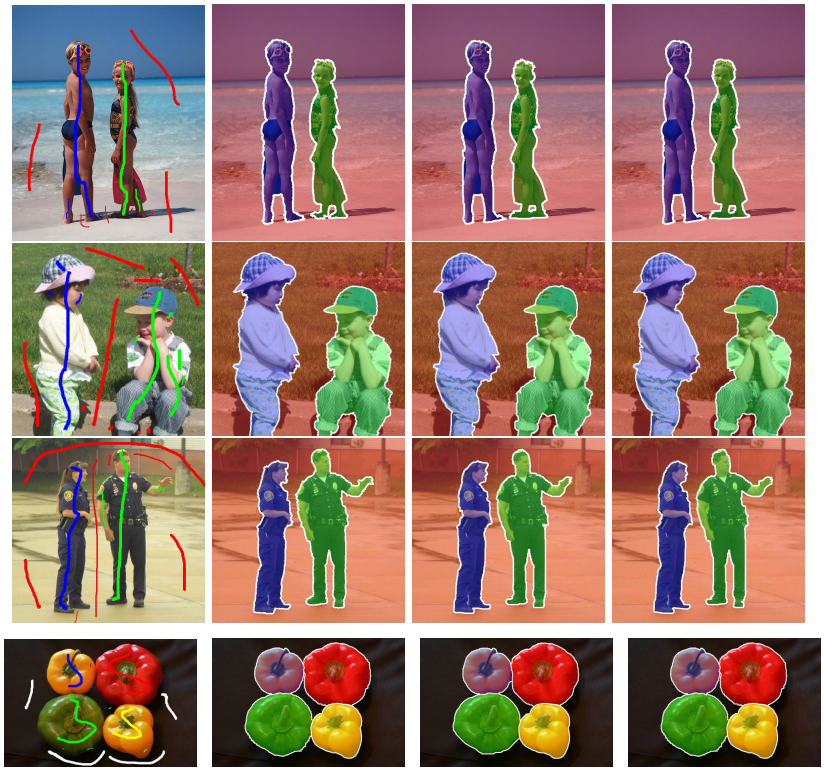
This subsection shows image segmentation results using iterated tree partitioning. Fig. 5.7 shows the iteration procedure. (a) shows the original image, (b) shows the initial segmentation by normalized tree partitioning without any fine parameter tuning, (c) shows one intermediate result, and the final segmentation is shown in (d). To demonstrate the convergence, Fig. 5.7(e) shows the convergence process of the logarithm of the posterior probability, which is optimized in the dynamic programming procedure.

The grabcut method [89] used a similar iterative mechanism. But we provide initial segmentation automatically instead of masking a rectangle. Fig. 5.8 shows comparison results, from which we can see that automatic segmentation by our method is satisfactory as well. More results are shown in Fig. 5.9, in which original images and converged segmentations are both shown. In our experiment, the convergence is much fast, and only requires several iterations.

5.7 Conclusion

In this chapter, a maximum likelihood solution to fitting a tree to an image graph is presented, and a tree partitioning framework, normalized tree partitioning, augmented tree partitioning, and iterated tree partitioning, is proposed. Normalized tree partitioning is originally proposed, it is theoretically proved that normalized tree bipartition can produce the exact global optimum. It is faster than spectral clustering, and obtains superior performance. The iterated tree partitioning can automatically extract objects in an iterative manner, while provides an automatic and practical initialization using normalized tree partitioning.

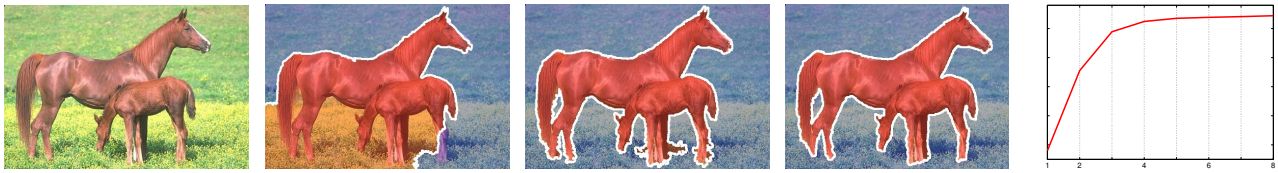
The proposed tree partitioning methods can be easily generalized for un/semi-supervised learning



(a) Original image (b) Our approach (c) Graph cuts (d) Graph cuts

Figure 5.6: Interactive segmentation for multiple object extraction.

problems. Error bound analysis of our approaches is a future topic.



(a) Original image (b) Initial result (c) Intermediate result (d) Converged result (e) Convergence curve

Figure 5.7: Segmentation results using iterated tree partitioning. (e) shows the convergence curve of the logarithm of the posterior probability.

Figure 5.8: A comparison between iterated tree partitioning and grabcut. From left to right, the figure shows the interaction for grabcut, the result by grabcut, the result by normalized tree partitioning, and the converged result by iterated tree partitioning.

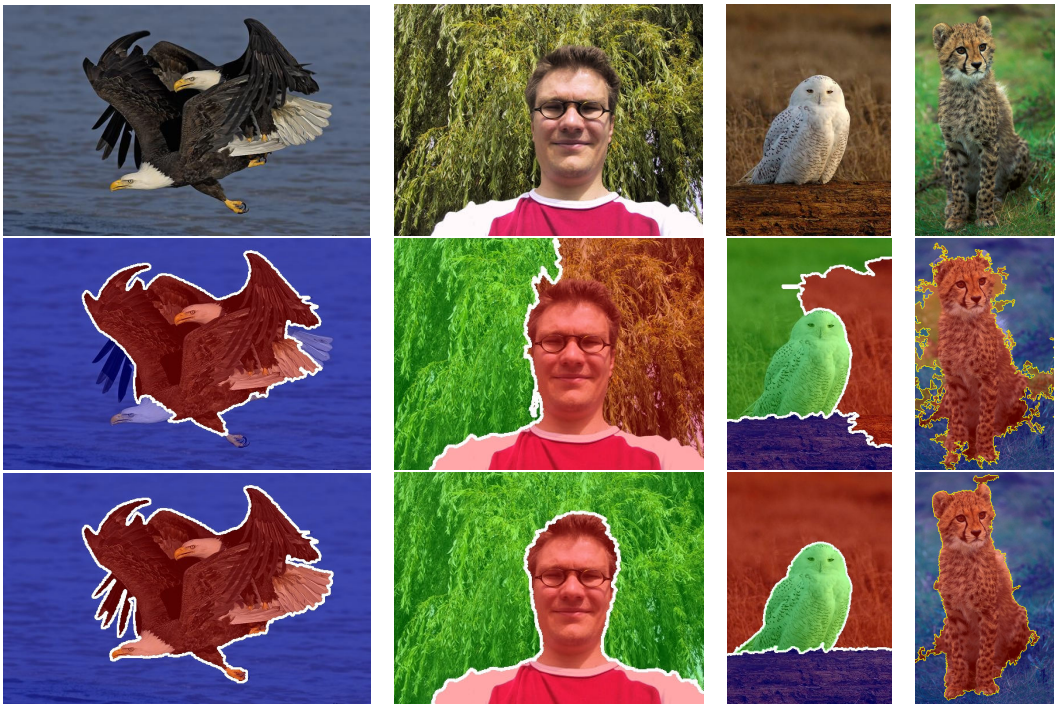


Figure 5.9: Segmentation results using iterated tree partitioning. In the first three columns, the top row shows initial segmentation by normalized tree partitioning, and the converged results shown in the bottom row take 3, 3 and 6 iterations, respectively. The last column shows a failure case due to the ambiguous boundary between object figure and background.

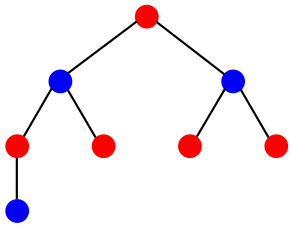


Figure 5.10: The super tree $(\{\mathcal{V}_j\}_{j=0}^m, \{(u_i, v_i)\}_{i=1}^m)$.

Appendix

Lemma:

$$\begin{aligned} &\sum_{i=1}^m (a_{\mathcal{A}_i} + a(u_i, v_i))(a_{\mathcal{B}_i} + a(u_i, v_i)) \\ > &(a_{\mathcal{A}} + \sum_{i=1}^m a(u_i, v_i))(a_{\mathcal{B}} + \sum_{i=1}^m a(u_i, v_i)) \end{aligned} \tag{5.24}$$

Proof: Interestingly, subtrees $\{\mathcal{V}_j\}_{j=0}^m$ and edges $\{(u_i, v_i)\}_{i=1}^m$ can be viewed as a super tree $\overline{\mathcal{T}}$ with subtrees as super nodes connected by edges $\{(u_i, v_i)\}_{i=1}^m$. Then the following two statements holds: (1) $\mathcal{A}$ and $\mathcal{B}$ correspond to the subsets of super nodes with odd depths (blue nodes in Fig. 5.10) and even depths (red nodes in Fig. 5.10), respectively; (2) $(\mathcal{A}_i, \mathcal{B}_i)$ can be viewed as a partition of the super tree $\overline{\mathcal{T}}$ by removing edge (u_i, v_i). With those two observations, the following inequality can be easily validated

$$a_{\mathcal{A}} a_{\mathcal{B}} = (\sum_{\mathcal{V}_o \in \mathcal{A}} a_{\mathcal{V}_o})(\sum_{\mathcal{V}_e \in \mathcal{B}} a_{\mathcal{V}_e}) < \sum_{i=1}^m (\sum_{\mathcal{V}_j \in \mathcal{A}_i} a_{\mathcal{V}_j} \sum_{\mathcal{V}_j \in \mathcal{B}_i} a_{\mathcal{V}_j}).$$

Transferring and expanding the left hand side of Eqn. (5.24), we can have:

$$\begin{aligned} & \sum_{i=1}^m (a_{\mathcal{A}_i} + a(u_i, v_i))(a_{\mathcal{B}_i} + a(u_i, v_i)) \\ = & \sum_{i=1}^m [\sum_{\mathcal{V}_j \in \mathcal{A}_i} a_{\mathcal{V}_j} + 2 \sum_{(u_j, v_j) \in \mathcal{A}_i} a(u_j, v_j) + a(u_i, v_i)][\sum_{\mathcal{V}_j \in \mathcal{B}_i} a_{\mathcal{V}_j} + 2 \sum_{(u_j, v_j) \in \mathcal{B}_i} a(u_j, v_j) + a(u_i, v_i)] \\ > & \sum_{i=1}^m [\sum\nolimits_{\mathcal{V}_j \in \mathcal{A}_i} a_{\mathcal{V}_j} \sum\nolimits_{\mathcal{V}_j \in \mathcal{B}_i} a_{\mathcal{V}_j} + (\sum\nolimits_{\mathcal{V}_j \in \mathcal{A}_i} a_{\mathcal{V}_j} + \sum\nolimits_{\mathcal{V}_j \in \mathcal{B}_i} a_{\mathcal{V}_j}) a(u_i, u_i) \\ & + (2 \sum\nolimits_{(u_j, v_j) \in \mathcal{A}_i} a(u_j, v_j) + a(u_i, v_i)) \times (2 \sum_{(u_j, v_j) \in \mathcal{B}_i} a(u_j, v_j) + a(u_i, v_i))] \\ > & a_{\mathcal{A}} a_{\mathcal{B}} + \sum_{j=0}^m a_{\mathcal{V}_j} \sum_{i=1}^m a(u_i, u_i) + \sum_{i=1}^m [a^2(u_i, v_i) + a(u_i, v_i) \sum\nolimits_{j \neq i} a(u_j, v_j)] \\ = & a_{\mathcal{A}} a_{\mathcal{B}} + (a_{\mathcal{A}} + a_{\mathcal{B}}) \sum_{i=1}^m a(u_i, v_i) + [\sum_{i=1}^m a(u_i, v_i)]^2 \\ = & (a_{\mathcal{A}} + \sum_{i=1}^m a(u_i, v_i))(a_{\mathcal{B}} + \sum_{i=1}^m a(u_i, v_i)). \end{aligned}$$

Therefore, the lemma holds. □

Chapter 6

Conclusions

Image segmentation plays a fundamental role in computer vision. High level vision problems, such as object recognition, image understanding and so on, can essentially benefit from the segmentation results. However, up to now, its solutions still remain unsatisfactory. The basic difficulties of image segmentation lie in two folds. The first one is what pixels should be grouped together. There is no general solution to this problem since it is subjective, and different persons have different ideas of it. Now researcher seems to reach the same point, *i.e*. combining the recognition into the segmentation. The other one is what algorithm is efficient to infer the segmentation with the criterion given. Historically, clustering methods, region based approaches, and contour based methods, have paved the progresses. Recently graph based approaches have attracted much interest.

The application of image segmentation not only lies in the vision such as image based modeling, but also is extended into image retrieval, in which image segmentation is viewed as the necessary step to roughly understand image contents. We have applied image segmentation for image management [115].

In this thesis, we addressed the graph based prior design problem for image segmentation from the following perspectives:

- We investigate the prior design in the conventional pairwise graph approaches. We design a joint prior in the joint segmentation for 3D clustering and 2D image segmentation. This problem makes good use of the current states of the art in both unbiased and biased graph partitioning.
- We generalize the pairwise graph to the hypergraph that can model multiple-wise relation among the data points. We propose a biased hypergraph partitioning approach, called Linear Neighborhood Propagation, which leads to a linear system with an efficient solution.
- We degrade the cyclic graph based prior to a connected acyclic graph based prior, *i.e*. tree based prior. We propose a tree partitioning method based on the normalized cut criterion,

called Normalized Tree Partitioning. It runs in linear time, and potentially results in superior performance over spectral graph partitioning due to less approximation.

The later two novel methods are proposed from two different directions. Linear neighborhood propagation is proposed to capture high-order statistics to model the relation of the data points better. It can be applied to not only image segmentation to obtain better results, but also many pattern recognition problems. The tree partitioning approaches are proposed to find an efficient algorithm to make image segmentation faster and more practical. The result is even improved possibly due to more effective solutions though the starting point is for efficiency. Similar to linear neighborhood propagation, tree based methods can also be used for pattern recognition problems, particular for large scale data clustering. One future work is to apply the two approaches to other applications besides image segmentation.

Bibliography

[1] R. Adams and L. Bischof. Seeded Region Growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6):641–647, 1994.

[2] S. Agarwal, K. Branson, and S. Belongie. Higher Order Learning with Graphs. In *ICML*, pages 17–24, 2006.

[3] N. Alon. Eigenvalues and Expanders. *Combinatorica*, 6(2):83–96, 1986.

[4] A. Barbu and S.-C. Zhu. Generalizing Swendsen-Wang to Sampling Arbitrary Posterior Probabilities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1239–1253, 2005.

[5] S. Barré. http://www.barre.nom.fr/medical/samples.

[6] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):711–720, 1997.

[7] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and Semi-supervised Learning on Large Graphs. In *COLT*, pages 624–638, 2004.

[8] M. Belkin and P. Niyogi. Laplacian Eigenmaps and Spectral Techniques for Embedding and clustering. In *NIPS*, pages 585–591, 2001.

[9] M. Belkin, P. Niyogi, and V. Sindhwani. On Manifold Regularization. pages 17–24. Society for Artificial Intelligence and Statistics, 2005. (Available electronically at http://www.gatsby.ucl.ac.uk/aistats/).

[10] J. Besag. On The Statistical Analysis of Dirty Pictures. *J. Roy. Stat. Soc*, 48(3):259–302, 1986.

[11] A. Blake, C. Rother, M. Brown, P. Pérez, and P. Torr. Interactive Image Segmentation Using an Adaptive GMMRF Model. In *ECCV (1)*, pages 428–441, 2004.

[12] A. Blum and S. Chawla. Learning from Labeled and Unlabeled Data using Graph Mincuts. In *ICML*, pages 19–26, 2001.

[13] A. Blum, J. D. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised Learning Using Randomized Mincuts. In *ICML*, 2004.

[14] A. Blum and T. M. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *COLT*, pages 92–100, 1998.

[15] R. B. Boppana. Eigenvalues and Graph Bisection: An Average-Case Analysis. *Proc. 28th Symp. Foundations of Computer Science*, pages 280–285, 1987.

[16] E. Borenstein, E. Sharon, and S. Ullman. Combining Top-Down and Bottom-Up Segmentation. In *2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, 2004.

[17] E. Borenstein and S. Ullman. Class-Specific, Top-Down Segmentation. In *ECCV (2)*, pages 109–124, 2002.

[18] E. Borenstein and S. Ullman. Learning to Segment. In *ECCV (3)*, pages 315–328, 2004.

[19] Y. Boykov and M.-P. Jolly. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In *ICCV*, pages 105–112, 2001.

[20] Y. Boykov and M.-P. Jolly. Graph Cuts and Efficient N-D Image Segmentation. *Int. J. Comput. Vision*, 70(2):109–131, 2006.

[21] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[22] C. R. Brice and C. Fennema. Scene Analysis Using Regions. *AI*, 1(3-4):205–226, 1970.

[23] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image Segmentation Using Expectation-Maximization and Its Application to Image Querying. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(8):1026–1038, 2002.

[24] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic Active Contours. *Int. J. Comput. Vision*, 22(1):61–79, 1997.

[25] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral -Way Ratio-Cut Partitioning and Clustering. In *DAC*, pages 749–754, 1993.

[26] T. Chan and W. Zhu. Level Set Based Shape Prior Segmentation. In *CVPR (2)*, pages 1164–1170, 2005.

[27] O. Chapelle, J. Weston, and B. Schölkopf. Cluster Kernels for Semi-Supervised Learning. In *NIPS*, pages 585–592, 2002.

[28] J. Cheeger. A Lower Bound for the smallest Eigenvalue for the Laplacian. *Problems in Analysis.*, (5):195–199, 1970.

[29] C. K. Chow and C. N. Liu. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Trans. Information Theory*, 14(3):462–467, May 1968.

[30] F. R. K. Chung. *Spectral Graph Theory.* Am.Math. Soc., 1997.

[31] M. C. Clark, L. O. Hall, D. B. Goldgof, L. P. Clarke, R. P. Velthuizen, and M. S. Silbiger. MRI Segmentation Using Fuzzy Clustering Techniques. *IEEE Engineering in Medicine and Biology Magazine*, 13(5):730–742, 1994.

[32] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer Segmentation of Live Video. In *CVPR*, 2006.

[33] D. de Ridder, J. Kittler, and R. P. W. Duin. Probabilistic PCA and ICA Subspace Mixture Models for Image Segmentation. In *BMVC*, 2000.

[34] O. Delalleau, Y. Bengio, and N. L. Roux. Efficient Non-Parametric Function Induction in Semi-Supervised Learning. pages 96–103. Society for Artificial Intelligence and Statistics, 2005. (Available electronically at http://www.gatsby.ucl.ac.uk/aistats/).

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. of the Royal Statistical Society*, Series B:1–38, 1977.

[36] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. In *ICDM*, pages 107–114, 2001.

[37] W. E. Donath and A. J. Hoffman. Lower Bounds for the Partitioning of Graphs. *IBM J. Research and Development.*, (6):420–425, 1973.

[38] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Belief Propagation for Early Vision. In *CVPR (1)*, pages 261–268, 2004.

[39] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.

[40] M. Fiedler. A Property of Eigenvectors of Nonnegative Symmetric Matrics and Its Applicatoin to Graph Theory. *Czech. Math. J.*, pages 619–633, 1975.

[41] L. R. Ford and D. R. Fulkerson. *Flow in Networks.* Princeton University Press, 1962.

[42] D. Freedman and T. Zhang. Interactive Graph Cut Based Segmentation with Shape Priors. In *CVPR (1)*, pages 755–762, 2005.

[43] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning Low-Level Vision. *Int. J. Comput. Vision*, 40(1):25–47, 2000.

[44] B. J. Frey and N. Jojic. A Comparison of Algorithms for Inference and Learning in Probabilistic Graphical Models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(9):1392–1416, 2005.

[45] B. J. Frey and D. J. C. MacKay. A Revolution: Belief Propagation in Graphs with Cycles. In *NIPS*, 1997.

[46] A. Fujino, N. Ueda, and K. Saito. A Hybrid Generative/Discriminative Approach to Semi-Supervised Classifier Design. In *AAAI*, pages 764–769, 2005.

[47] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and The Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Machine Intell.*, 6(6):721–741, Nov. 1984.

[48] L. Grady. Multilabel Random Walker Image Segmentation Using Prior Models. In *CVPR (1)*, pages 763–770, 2005.

[49] L. Grady. Random Walks for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1768–1783, 2006.

[50] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact Maximum A Posteriori Estimation for Binary Images. *Journal of the Royal Statistical Society. Series B (Methodological).*, 51(2):271–279, 1989.

[51] L. W. Hagen and A. B. Kahng. A New Approach to Effective Circuit Clustering. In *ICCAD*, pages 422–427, 1992.

[52] F. Hamze and N. de Freitas. From Fields to Trees. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 243–250, Arlington, Virginia, 2004. AUAI Press.

[53] D. Hoiem, A. A. Efros, and M. Hebert. Geometric Context from a Single Image. In *ICCV*, pages 654–661, 2005.

[54] S. L. Horowitz and T. Pavlidis. Picture Segmentation by a Tree Traversal Algorithm. *JACM*, 23(2):368–388, April 1976.

[55] S. L. Horowitz and T. Pavlidis. A Graph-Theoretic Approach to Picture Processing. 7(2):282–291, April 1978.

[56] H. Ishikawa and D. Geiger. Segmentation by Grouping Junctions. In *CVPR*, pages 125–131, 1998.

[57] T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *ICML*, pages 200–209, 1999.

[58] T. Joachims. Transductive Learning via Spectral Graph Partitioning. In *ICML*, pages 290–297, 2003.

[59] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

[60] O. Juan and Y. Boykov. Active Graph Cuts. In *CVPR (1)*, pages 1023–1029, 2006.

[61] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[62] M. Kass, A. P. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *Int. J. Comput. Vision*, 1(4):321–331, 1988.

[63] P. Kohli and P. H. S. Torr. Effciently Solving Dynamic Markov Random Fields Using Graph Cuts. In *ICCV*, pages 922–929, 2005.

[64] V. Kolmogorov. Convergent Tree-Reweighted Message Passing for Energy Minimization. In *AISTATS*, 2005.

[65] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-Layer Segmentation of Binocular Stereo Video. In *CVPR (2)*, pages 407–414, 2005.

[66] V. Kolmogorov and R. Zabih. Multi-camera Scene Reconstruction via Graph Cuts. In *ECCV (3)*, pages 82–96, 2002.

[67] V. Kwatra, A. Schödl, I. A. Essa, G. Turk, and A. F. Bobick. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003.

[68] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289, 2001.

[69] A. Levin, D. Lischinski, and Y. Weiss. A Closed Form Solution to Natural Image Matting. In *CVPR (1)*, pages 61–68, 2006.

[70] A. Levin and Y. Weiss. Learning to Combine Bottom-Up and Top-Down Segmentation. In *ECCV (4)*, pages 581–594, 2006.

[71] M. Lhuillier. Efficient Dense Matching for Textured Scenes using Region Growing. In *BMVC*, 1998.

[72] M. Lhuillier and L. Quan. A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):418–433, 2005.

[73] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy Snapping. In *Proceedings of ACM SIGGRAPH*, pages 303–308, 2004.

[74] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and Texture Analysis for Image Segmentation. *Int. J. Comput. Vision*, 43(1):7–27, 2001.

[75] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982.

[76] O. Monga. An Optimal Region Growing Algorithm for Image Segmentation. *PRAI*, 1(4):351–375, December 1987.

[77] U. Montanari. On the Optimal Detection of Curves in Noisy Pictures. *Commun. ACM*, 14(5):335–345, 1971.

[78] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[79] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library: COIL-20. Technical report, Department of Computer Science, Columbia University.

[80] K. Nigam. *Using Unlabeled Data to Improve Text Classification*. PhD thesis, Pittsburgh, US, 2001.

[81] N. Otsu. A Threshold Selection Method from Gray Level Histograms. *IEEE Trans. Systems, Man and Cybernetics*, 9:62–66, Mar 1979.

[82] P. Parent and S. W. Zucker. Trace Inference, Curvature Consistency, and Curve Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(8):823–839, 1989.

[83] I. Patras, E. Hendriks, and R. Lagendijk. Video Segmentation by MAP Labeling of Watershed Segments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(3):326–332, 2001.

[84] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[85] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990.

[86] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. Image-based Plant Modeling. *ACM Trans. Graph.*, 25(3):599–604, 2006.

[87] L. Quan, J. Wang, P. Tan, and L. Yuan. Image-based Modeling by Joint Segmentation. *Int. J. Comput. Vision*, 75(1):135–150, October 2007.

[88] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-Supervised Self-Training of Object Detection Models. In *WACV/MOTION*, pages 29–36, 2005.

[89] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. In *Proceedings of ACM SIGGRAPH.*, pages 309–314, 2004.

[90] S. Roweis and L. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science.*, 290(5500):2323–2326, 2000.

[91] S. Roy and I. J. Cox. A Maximum-Flow Formulation of the N-Camera Stereo Correspondence Problem. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 492, Washington, DC, USA, 1998. IEEE Computer Society.

[92] H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 2005.

[93] F. Samaria and A. Harter. Parameterisation of a Stochastic Model for human Face Identification. In *IEEE Workshop on Applications of Computer Vision*, Sarasota (Florida), December 1994.

[94] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[95] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of Shapes by Editing Shock Graphs. In *ICCV*, pages 755–762, 2001.

[96] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[97] N. Shental, A. Zomet, T. Hertz, and Y. Weiss. Learning and Inferring Image Segmentations using the GBP Typical Cut Algorithm. In *ICCV*, pages 1243–1250, 2003.

[98] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[99] A. Sinclair and M. Jerrum. Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains. *Information and Computation*, pages 93–133, July 1989.

[100] D. A. Spielman and S.-H. Teng. Disk Packings and Planar Separators. In *Symposium on Computational Geometry*, pages 349–358, 1996.

[101] J. Sun, W. Zhang, X. Tang, and H. Shum. Background Cut. In *ECCV*, 2006.

[102] R. H. Swendsen and J.-S. Wang. Nonuniversal Critical Dynamics in Monte Carlo Simulations. *Physical Review Letters*, 58:86–88, 1987.

[103] M. A. Tanner and W. H. Wong. The Calculation of Posterior Distributions by Data Augmentation (with discussion). *Journal of the American Statistical Association*, 82:528–550, 1987.

[104] M. Tipping and C. Bishop. Mixtures of Probabilistic Principal Component Analysers. *Neural Computation*, 11(2):443–482, 1999.

[105] P. Torr, R. Szeliski, and P. Anandan. An Integrated Bayesian Approach to Layer Extraction from Image Sequences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(3):297–303, 2001.

[106] Z. Tu, X. Chen, A. L. Yuille, and S. C. Zhu. Image Parsing: Unifying Segmentation, Detection, and Recognition. *Int. J. Comput. Vision*, 63(2):113–140, 2005.

[107] Z. Tu and S. C. Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):657–673, 2002.

[108] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[109] S. Ullman and A. Sha"ashua. Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network. Technical report, Cambridge, MA, USA, 1988.

[110] R. Urquhart. Graph Theoretical Clustering Based on Limited Neighborhood Sets. *Pattern Recognition*, 15(3):173–187, 1982.

[111] O. Veksler. Stereo Correspondence by Dynamic Programming on a Tree. In *CVPR (2)*, pages 384–390, 2005.

[112] L. Vincent and P. Soille. Watersheds in Digital Spaces: an Efficient Algorithm Based on Immersion Simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598, June 1991.

[113] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-reweighted Belief Propagation and Approximate ML Estimation by Pseudo-moment Matching. In *AISTATS*, 2003.

[114] J. Wang and E. Adelson. Representing Moving Images with Layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[115] J. Wang, J. Sun, L. Quan, X. Tang, and H.-Y. Shum. Picture Collage. In *CVPR (1)*, pages 347–354, 2006.

[116] S. Wang and J. M. Siskind. Image Segmentation with Ratio Cut. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(6):675–690, 2003.

[117] A. R. Weeks and G. E. Hague. Color Segmentation in the HSI Color Space Using the K-means Algorithm. In *Proc. SPIE Vol. 3026, p. 143-154, Nonlinear Image Processing VIII, Edward R. Dougherty; Jaakko T. Astola; Eds.*, pages 143–154, apr 1997.

[118] Y. Wei, E. Ofek, L. Quan, and H.-Y. Shum. Modeling Hair from Multiple Views. *ACM Trans. Graph.*, 24(3):816–820, 2005.

[119] M. Wertheimer. Laws of Organization in Perceptual Forms (partial translation). *A source book of Gestalt psychology (1938).*, 4.

[120] L. R. Williams and D. W. Jacobs. Stochastic Completion Fields: A Neural Model of Illusory Contour Shape and Salience. In *ICCV*, pages 408–415, 1995.

[121] J. Wills, S. Agarwal, and S. Belongie. What Went Where. In *CVPR (1)*, pages 37–44, 2003.

[122] Q. Wu, W. Dou, Y. Chen, , and J.-M. Constans. Fuzzy Segementaion of Cerebral Tumorous Tissues in MR Images via Support Vector Machine and Fuzzy Clustering. In *IFSA*, 2005.

[123] Z. Wu and R. M. Leahy. An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1101–1113, 1993.

[124] J. Xiao and M. Shah. Motion Layer Extraction in the Presence of Occlusion Using Graph Cut. In *CVPR (2)*, pages 972–979, 2004.

[125] M.-H. Yang. Kernel Eigenfaces vs. Kernel Fisherfaces: Face Recognition Using Kernel Methods. In *FGR*, pages 215–220, 2002.

[126] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized Belief Propagation. In *NIPS*, pages 689–695, 2000.

[127] S. X. Yu and J. Shi. Multiclass Spectral Clustering. In *ICCV*, pages 313–319, 2003.

[128] S. X. Yu and J. Shi. Object-Specific Figure-Ground Segregation. In *CVPR (2)*, pages 39–45, 2003.

[129] R. Zabih and V. Kolmogorov. Spatially Coherent Clustering Using Graph Cuts. In *CVPR (2)*, pages 437–444, 2004.

[130] C. Zahn. Graph Theoretic Methods for Detecting and Describing Gestalt Clusters. *IEEE Trans. Computers*, 20.

[131] G. Zeng, S. Paris, L. Quan, and F. X. Sillion. Accurate and Scalable Surface Representation and Reconstruction from Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):141–158, 2007.

[132] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with Local and Global Consistency. In *NIPS*, 2003.

[133] D. Zhou, J. Huang, and B. Schoelkopf. Learning with Hypergraphs: Clustering, Classification, and Embedding. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[134] D. Zhou and B. Schölkopf. Learning from Labeled and Unlabeled Data Using Random Walks. In *DAGM-Symposium*, pages 237–244, 2004.

[135] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on Data Manifolds. In *NIPS*, 2003.

[136] S. C. Zhu and A. L. Yuille. Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(9):884–900, 1996.

[137] X. Zhu. Semi-supervied Learning Literature Survey. *Computer Sciences Technical Report, 1530, University of Wisconsin-Madison*, 2006.

[138] X. Zhu and Z. Ghahramani. Learning from Labeled and Unlabeled Data with Label Propagation. Technical Report tech report CMU-CALD-02-107, 2002.

[139] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *ICML*, pages 912–919, 2003.

[140] X. Zhu and J. Lafferty. Harmonic Mixtures: Combining Mixture Models and Graph-based Methods for Inductive and Scalable Semi-supervised Learning. In *ICML*, pages 1052–1059, 2005.