INAUGURAL-DISSERTATION

zur Erlangung der Doktorwürde
der Naturwissenschaftlich-Mathematischen Gesamtfakultät
der Ruprecht-Karls-Universität Heidelberg

vorgelegt von
MSc-Math Ruobing Shen
aus Jiangsu, P.R. China

Tag der mündlichen Prüfung: .....

MILP Formulations for Unsupervised and Interactive Image Segmentation and Denoising

Betreuer:
Prof. Dr. Gerhard Reinelt
Prof. Dr. Christoph Schnörr

To my parents
To my son - Woxin Shen
To my beloved wife - Suqin Weng

Zusammenfassung

Bildsegmentierung und Rauschunterdrückung sind die Schlüsselkomponenten moderner Bilderkennungssysteme. Das Potts Model spielt dabei eine bedeutende Rolle für die Entrauschung von stückweise definierten Funktionen, und Markov Random Field (MRF) Modelle die Potts-Terme benutzen sind sehr beliebt bei der Bildsegmentierungen. Wir präsentieren gemischt ganzzahlige Programme (MILP) für beide Modelle und werden diese mit Hilfe moderne Löser wie CPLEX effizient lösen.

Zuerst untersuchen das diskrete das diskrete Potts-Modell erster Ableitung (stückweise konstant) mit einem ℓ_1 Data-Anteil. Wir präsentieren eine neue MILP Formulierung durch Einführung binärer Kantenvariablen, um die Potts Funktion zu modellieren. Anschließend untersuchen wir die Facetten definierenden Ungleichungen für das zugehörige ganzzahlige Polytop Wir wenden dieses Modell an um Superpixel auf verrauschten Bildern zu erzeugen.

Zweitens präsentieren wir eine MILP-Formulation für das diskrete, stückweise affine Potts-Modell. Um konsistente Partitionen zu erhalten, ist das Hinzufügen vom Multicut-Constraint notwendig. Diese werden iterativ mit Hilfe der Schnittebenenmethode hinzugefügt. Wir wenden diese Modell für die die gleichzeitige Segmentierung und Entrauschung von Bildern an.

MILP-Formulierungen von MRF-Modellen mit globalen Konnektivitätsbeschränkungen wurden zuvor untersucht, aber nur vereinfachte Versionen des Problems wurden gelöst. Wir lösen dieses Problem mit einer Branch-and-Cut-Methode und präsentieren einen Benutzer-interaktive Weg zur Segmentierung.

Unsere vorgeschlagenen MILP's sind im Allgemeinen $\mathcal{NP}$-hard, aber man kann mit ihnen globale optimale Lösungen finden. Wir haben auch drei schnelle heuristische Algorithmen entwickelt die gute Lösungen in sehr kurzer Zeit liefern. Die MILPs können als Post-Verarbeitungsverfahren zusätzlich zu allen Algorithmen benutzt werden, denn sie stellen eine Garantie für die Qualität der Lösung da, und such auch nach besseren Lösungen innerhalb des Branch-and-Cut-Rahmens der Lösers.

Wir zeigen die Stärke und Nützlichkeit unserer Methoden bei Vergleichsrechnungen mit anderen State-of-the-art-Methoden auf synthetischen Bildern, Standard - Bilddatensätzen und auf medizinische Bilder mit trainierten Wahrscheinlichkeitskarten.

Abstract

Image segmentation and denoising are two key components of modern computer vision systems. The Potts model plays an important role for denoising of piecewise defined functions, and Markov Random Field (MRF) using Potts terms are popular in image segmentation. We propose Mixed Integer Linear Programming (MILP) formulations for both models, and utilize standard MILP solvers to efficiently solve them.

Firstly, we investigate the discrete first derivative (piecewise constant) Potts model with the ℓ_1 norm data term. We propose a novel MILP formulation by introducing binary edge variables to model the Potts prior. We look into the facet-defining inequalities for the associated integer polytope. We apply the model for generating superpixels on noisy images.

Secondly, we propose a MILP formulation for the discrete piecewise affine Potts model. To obtain consistent partitions, the inclusion of multicut constraints is necessary, which is added iteratively using the cutting plane method. We apply the model for simultaneously segmenting and denoising depth images.

Thirdly, MILP formulations of MRF models with global connectivity constraints were investigated previously, but only simplified versions of the problem were solved. We investigate this problem via a branch-and-cut method and propose a user-interactive way for segmentation.

Our proposed MILPs are in general $\mathcal{NP}$-hard, but they can be used to generate globally optimal solutions and ground-truth results. We also propose three fast heuristic algorithms that provide good solutions in very short time. The MILPs can be applied as a post-processing method on top of any algorithms, not only providing a guarantee on the quality, but also seek for better solutions within the branch-and-cut framework of the solver.

We demonstrate the power and usefulness of our methods by extensive experiments against other state-of-the-art methods on synthetic images, standard image datasets, as well as medical images with trained probability maps.

Acknowledgement

I yield my first thanks to my supervisor Prof. Gerhard Reinelt for giving me the opportunity to study in his research group and arousing my interest in combinatorial optimization. He is always a supportive and reliable person. Without his help, I would not have had enough motivation to achieve such progress in my research.

My second supervisor, Prof. Christoph Schnörr, introduced me to the field of image processing. I am thankful to him for giving me valuable instructions and providing me with data. I owe special gratitude to the European Marie Curie Project "Mixed Integer Nonlinear Optimization (MINO)" with Project ID 316647, where I got the chance to know 14 awesome colleagues and 11 professors within my field. I thank Prof. Andrea Lodi, Dr. Andrea Tramontani, Prof. Leo Liberti and Prof. Claudia D'Ambrosio for hosting me as a visiting Ph.D. student within this project. Prof. Stéphane Canu introduced me to the field of Total Variation, which had huge impacts on my research direction. Prof. Ismail Ben Ayed is always willing to help in the field of MRF. I would also like to thank Eric Kendinibilir for getting the Boost code running. Xiaoyu Chen and Xiangrui Zheng helped a lot on the benchmark comparison. Last but not least, I thank Prof. Fred A. Hamprecht, Dr. Frank Lenzen and Dr. Martin Storath for valuable discussions.

The members of our research group - Dr. Stefan Wiesberg, Dr. Achim Hildenbrandt, Dr. Markus Speth, Dr. Francesco Silvestri, Dr. Tuan Nam Nguyen, and Xiaoyu Chen - provided a great working atmosphere. The members of the lunch group (including Dr. Hui Li) enriched my working days with discussions about scientific and non-scientific topics. I am grateful to our secretary Mrs. Catherine Proux Wieland, who helps with documenting and reimbursement stuffs for more than 4 years. Thanks also to our system administrators who were responsible for keeping our computers running: Jonas Große Sundrup and Felix Richter. For proof-reading parts of this thesis, I thank Dr. Achim Hildenbrandt and Dr. Martin Storath - your comments were indeed helpful. Of course, all the remaining errors are on my own.

Finally, I would like to express my heartfelt thanks to my parents for their unconditional support in all situations. I owe my deepest thanks to Suqin Weng for her loving support, and her extra efforts to taking care our lovely boy, Woxin Shen.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and Overview

In computer vision, *image segmentation* is a fundamental task that partitions a digital image into multiple segments (set of pixels). The goal is to change the representation of an image from pixels into segments that are semantically more meaningful and easier to analyze. See Figure 1.1 for an example. More precisely, image segmentation is the problem of assigning labels (e.g., *foreground* and *background* labels) to all pixels in an image such that pixels with the same label share certain properties. It is typically applied to locate objects and boundaries within an image.

Figure 1.1 – Illustration of an image segmentation: left is the input image, right is a segmentation into meaningful segments.

Image *noise* is random variation of brightness or color in an image. The presence of noise in images is unavoidable, making it difficult to perform any required image processing tasks. Given a noisy image, *image denoising* is the operation of estimating the clean, original image. As can be seen in Figure 1.2, the right image becomes smooth, but on the downside, the sharp boundaries have also been smeared. Hence, the goal of any denoising algorithm is to remove the noise while still keeping the image sharp.

Figure 1.2 – Illustration of an image denoising: left is the input noisy image, right is the denoised image.

Usually, one first applies denoising method as a pre-processing step before segmentation. But as discussed above, the denoising algorithm may over-smoothen the sharp boundaries and hence harm the segmentation results.

In this thesis, we first combine these two problems into one framework, and propose a novel formulation for simultaneously denoising and segmenting a given noisy image. It is based on the well-known Potts model [1], and can be either a piecewise constant or affine model.

Secondly, we allow users to interact with the image, either to manually label some pixels, or to exclude the known outliers. We propose an interactive image segmentation approach, where the user is required to draw k scribbles, and the output is exactly k connected segments. See Figure 1.3 as an example, where the input is the probability map trained with convolutional neural networks (CNN) of a magnetic resonance image (MRI), which depicts the abdominal aorta. The formulation is based on the Markov Random Field (MRF) model [2] and we introduce a global prior which enforces the connectivity of each label.

All the models in this thesis are formulated as Mixed Integer Linear Programs (MILPs). It is in general $\mathcal{NP}$-hard to solve a MILP, thus fast heuristic algorithms are usually beneficial in that they provide initial solutions to the MILP solver, as well as to the original problem. Sophisticated methods from combinatorial optimization such as branch-and-cut are implemented by default in any modern MILP solver, to get globally optimal solutions. The main advantage of our model over heuristics is that they provide globally optimal solutions if no runtime restrictions are specified, hence provides ground-truth results that could be used as quality assessment for any algorithms.

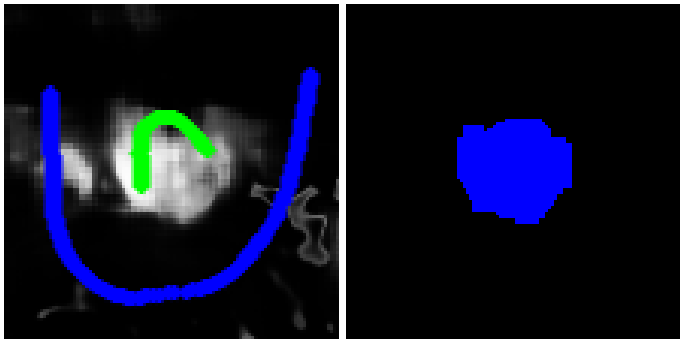


Figure 1.3 – Illustration of an interactive image segmentation: left is the input probability map with 2 user scribbles, right is the desired segmentation.

1.2 Contributions

The main contributions of this thesis are the following:

- We propose novel MILP formulations for both piecewise constant and affine Potts models.
- We explore facet-defining inequalities for the associated integer polytope of the piecewise constant Potts model.
- We conduct extensive benchmark comparison using several state-of-the-art superpixel algorithms applied on noisy images.
- We explore additional constraints that enforce valid segmentation in the MILP formulation of piecewise affine Potts model.
- We revisit the Integer Linear Programming (ILP) formulation for multi-label MRF with connectivity priors and solve it to optimality.

1.3 Organization

This thesis is organized as follows: We start in Chapter 2 by giving basic definitions, starting from graph and polyhedral theory to mathematical programming, linear regression, and finally to superpixels.

In Chapter 3, we first introduce the MILP formulation of the discrete piecewise constant (first derivative) Potts model and the multicut problem [3]. We prove that the multicut constraints are redundant for the the optimal solutions of the Potts model, but they are facet-defining for a special integer polytope. We also introduce a ℓ_1 norm fast heuristic based on the region fusion algorithm [4]. Due to the $\mathcal{NP}$-hardness of the

corresponding optimization problem, we decompose the original image into rectangular blocks (patches) and apply our model within each patch for generating superpixels on noisy images of the BSDS500 dataset [5] against other state-of-the-art superpixel algorithms.

Chapter 4 is devoted to the more general case where the input image is assumed to possess piecewise affine features. We propose a MILP formulation for the piecewise affine Potts model, and prove that the multicut constraints are needed to enforce a valid segmentation. We adapt the heuristic in [4] for the piecewise affine case which provides an initial solution to the MILP solver. Synthetic images as well as real depth images are tested to confirm the usefulness of our model.

In Chapter 5, we focus on the MRF model. We first revisit the ILP formulation of the pairwise MRF and introduce the concept of connected subgraphs. We then formulate the MRF with global connectivity as a MILP. For solution techniques, we again adapt the region fusion algorithm [4] to generate initial feasible solutions. We also discussion the strategy for selecting the cutting planes in the separation problem. Extensive computational experiments on standard image datasets as well as medical images are carried out using different variants of our proposed model.

Finally, we conclude this thesis and point out some future research directions in Chapter 6.

Chapter 2

Preliminaries and Terminologies

This chapter presents basic concepts needed throughout the thesis. Many of them are standard and can be found in the textbooks of the corresponding field, such as [6, 7]. We include them here to make the thesis more self-contained.

2.1 General Notation

A *set* is a collection of distinct elements (also known as members). It is usually denoted by a capital letter, while vectors and scalars use small letters. A set B is called a *subset* of A (denoted $B \subseteq A$) if every member of B is also a member of A. If B is a subset of A but not equal to B, then B is a *proper subset* of A (denoted $B \subset A$). The *power set* of A, denoted by $\mathbb{P}(A)$, is the set of all subsets in A. The *cardinality* of a set A, denoted by $|A|$, is the number of elements of A. The *union* of A and B, denoted by $A \cup B$, is the set of all members of either A or B. The *intersection* of A and B, denoted by $A \cap B$, is the set of all members of both A and B. The *Minkowski sum* of two sets A and B is the set of all elements $x + y$ with $x \in A$ and $y \in B$. A *partition* of a set A is a collection of nonempty sets $\{A_1, A_2, \ldots, A_n\}$, such that $\cup_{i=1}^{k} A_i = A$, and $A_i \cap A_j = \emptyset$, for any $i \neq j$.

We denote the set of real numbers $\mathbb{R}$, the set of nonnegative real numbers $\mathbb{R}^+$, the set of natural numbers $\mathbb{N}$. We also denote $[n]$ the set of integer numbers $\{1, 2, \ldots, n\}$.

A *vector* $a \in \mathbb{R}^{n\times 1}$ is usually a column vector. The *transpose* of the column vector a, denoted by $a^\top \in \mathbb{R}^{1\times n}$, is then a row vector. The *inner product* of two vectors $a = (a_1, a_2, \ldots, a_n)^\top$ and $b = (b_1, b_2, \ldots, b_n)^\top$ is denoted by $a^\top b$, and it equals $\sum_{i=1}^{n} a_i b_i$. The *p-norm*, also know as the ℓ_p *norm* of a vector $a \in \mathbb{R}^n$, is denoted as $\|a\|_p := (\sum_{i=1}^{n} |x_i|^p)^{1/p}$. In particular, the ℓ_1 norm $\|a\|_1 := \sum_{i=1}^{n} |x_i|$, and the ℓ_2 norm $\|a\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$. In contrast, the ℓ_0 norm of a vector a is the number of nonzero entries. We denote a vector of all zeros of appropriate dimensions as $\mathbf{0}$.

A *matrix* $D = (d_{ij}) \in \mathbb{R}^{m\times n}$ is a rectangular array of numbers or any mathematical

elements where operations like addition and multiplication are defined. The *transpose* of the $m \times n$ matrix D is denoted $D^\top = (d'_{ij}) \in \mathbb{R}^{n \times m}$ where $d'_{ij} = d_{ji}$.

For a logical expression ϑ, an *indicator function* $\mathbb{1}(\vartheta)$ is 1 if ϑ is true and 0 otherwise.

2.2 Graph Theory

All the models in this thesis are defined on undirected graphs. We hence introduce concepts from graph theory that we will need later. The definitions are mostly taken from the fundamental part of [8], which is a good reference for further reading.

2.2.1 Undirected graphs

An *undirected graph* $G = (V, E)$ (or simply $G(V, E)$) is a tuple of two basic finite sets V and E such that $E \subseteq \binom{V}{2}$. The elements $v \in V$ are called *vertices* (or *nodes*) and $e \in E$ called edges of the graph G. We denote the set of all nodes by $V(G)$ and the set of all edges of a graph G by $E(G)$, respectively. We will denote an edge $e = \{u, v\} \in E$ simply by uv, so in an undirected graph, $e = uv = vu$. We call the nodes u and v the *endnodes* of $e = uv$. Two nodes u and v are *adjacent* if $uv \in E$, and a node u is *indident* to an edge e, and vice versa, if u is an endnode of e. The *neighborhood* of a node u is the set $nb(u) := \{v \in V \mid uv \in E\}$, and the *degree* of node u is the cardinality of $nb(u)$. From now on, all the graphs discussed in this thesis are undirected graphs.

A graph $G'(V', E')$ is called a *subgraph* of $G(V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. We also say G' is contained in G in this case.

2.2.2 Paths, cycles, and connectivity

In a graph G, a *path* between two nodes u_1 and u_k (which we call a (u_1, u_k)-path) is a subgraph $P(V_P, E_P)$ of G where $V_P = \{u_i \in V \mid i \in [k]\}$ and $E_P = \{u_i u_{i+1} \in E \mid i \in [k-1]\}$. The *length* of a path P is just $|E_P|$.

A *cycle* is a subgraph $C(V_C, E_C)$ of G where $V_C = \{u_i \in V \mid i \in [k]\}$ and $E_C = \{u_i u_{i+1} \in E \mid i \in [k-1]\} \cup \{u_k u_1\}$. The *length* of a cycle C is just $|E_C|$. If there is a cycle C in G, then an edge $e = uv \in E(G)$ is called a *chord* of C in G if $u, v \in V_C$ and $e \notin E_C$. We call a cycle $C \subseteq G$ *chordless* if there is no chord of C in G.

Two nodes u, v in a graph G are *connected* if there is a (u, v)-path in G. G is called *connected* if for any pair of nodes in G, there exists a path between them, otherwise it is *disconnected*. A *connected component* of a graph G is a connected subgraph of G, and is maximal with respect to edge inclusion.

2.2.3 Cuts and multicuts

For a graph $G(V, E)$, the *cut* of G induced by a subset of nodes $D \subseteq V$ is denoted $\delta(D) := \{uv \in E \mid u \in D, v \in V \setminus D\}$. This definition is extended to disjoint sets $D_i \subset V$, for $i \in [n]$ and $n \geq 2$, such that $\delta(D_1, D_2, \ldots, D_n) := \{uv \in E \mid u \in D_i, v \in D_j, i \neq j,\ i, j \in [n]\}$. We call the set $\delta(D_1, D_2, \ldots, D_n)$ a *multicut* of G induced by $D_1, D_2, \ldots, D_n$, if $D_1, D_2, \ldots, D_n$ is a partition of V. The sets $D_1, D_2, \ldots, D_k$ are called the *shores* of the multicut.

2.2.4 Selected classes of graphs

A *grid graph* of size $m \times n$ is a graph $G(V, E)$ such that $V = \{(i, j) \mid i \in [m], j \in [n]\}$ and $E = \{((i, j), (i + 1, j)) \mid i \in [m - 1], j \in [n]\} \cup \{((i, j), (i, j + 1)) \mid i \in [m], j \in [n - 1]\}$.

A *weighted graph* is a tuple $G(V, E, w)$ where $G(V, E)$ is a graph and there exists an associated *weight function* $w : E \to \mathbb{R}$ (or $w : V \to \mathbb{R}$) that assigns a weight $w(e)$ for each edge of E (or $w(v)$ for each node of V).

2.3 Polyhedral Theory

The optimization models we formulate within this thesis will be mixed integer linear programs. However, the algorithms for solving such problems cannot be understood without some basic knowledge of polyhedral theory. We assume the readers are familiar with the standard linear theory such as the real vector space $\mathbb{R}^m$, subspaces, linear independence, scalar product, dimension and so on. The following notations and definitions are mainly based on [7].

2.3.1 Convex and affine subspaces

A *linear subspace* of $\mathbb{R}^m$ is a nonempty set $D \subseteq R^m$ such that for all $x_1, x_2 \in D$ and $a_1, a_2 \in \mathbb{R}$, $a_1 x_1 + a_2 x_2$ is in D. Thus, a linear subspace always contain $\mathbf{0}$. An *affine subspace* of $\mathbb{R}^m$ is a subset $A = x + D$, where $x \in R^m$ and D is a linear subspace of $\mathbb{R}^m$. The *dimension* of an affine subspace is the dimension of its associated linear vector space.

A *linear combination* of a set of vectors $\{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^m$ is defined by the scalar product $\sum_{i=1}^{n} a_i x_i$, where $a_i \in \mathbb{R}$ is called the coefficient. It is called *affine combination* if $\sum_{i=1}^{n} a_i = 1$ and *convex combination* if in addition, $a_i \geq 0$, for $i = 1, 2, \ldots, n$. The vectors $\{x_1, x_2, \ldots, x_n\}$ are called *affinely independent* if $(x_i - x_n)$, for $i \in [n - 1]$, are linearly independent.

A set $S \subseteq \mathbb{R}^m$ is called $convex$ if it contains all convex combinations of finitely many vectors in S. The *convex hull* of a set S is denoted by conv(S), and is the set of all convex combinations of vectors in S. The *affine hull* aff(S) is then defined analogously.

2.3.2 Polytopes and polyhedra

An *linear inequality (constraint)* $a^\top x \leq b$ defines a *halfspace* $\{x \in \mathbb{R}^n \mid a^\top x \leq b\}$ and a corresponding *hyperplane* $H(a, b) = \{x \in \mathbb{R}^n \mid a^\top x = b\}$.

A *polyhedra* P can be defined as the intersection of finitely many halfspaces and is denoted $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. It is called *bounded* if there exists a "box" $Q = \{x \mid l \leq x_i \leq u, \forall i \in [n]\} \subset \mathbb{R}^n$ with some l and u, such that $P \subseteq Q$. A *polytope* is a convex hull of a nonempty finite set, and it is a bounded polyhedra. It is clear that both the polyhedron and polytope are convex sets.

There are two ways to describe a polytope: the convex hull of a nonempty finite set, which is called $\mathcal{V}$-representation, or the bounded intersection of finitely many closed half spaces, also called $\mathcal{H}$-representation.

2.3.3 Vertices, faces and facets

Consider a polytope $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. In order to find out the fewest necessary inequalities to describe P, we need the following definitions.

An equality $a^\top x \leq b$ is called a *valid equality* for P if it holds for all $x \in P$. Its corresponding halfspace and hyperplane $H(a, b)$ is called *supporting* if $H \cap P \neq \emptyset$. If in addition $H \neq P$, we call it a *proper supporting hyperplane*. The set $F := P \cap H(a, b)$ is called the *face* of the polytope P induced by $a^\top x \leq b$. We call F a *proper face* if the corresponding hyperplane H is proper.

The *dimension* of a face F (dim(F)) in P is the dimension of its affine hull aff(F). We call the face of dimension $0, 1$ and dim$(P) - 1$ the *vertex*, *edge* and *facet* of P, respectively. A vetex v of P is often referred to as an *extreme point*.

An equality $a^\top x \leq b$ is called *facet-defining* if $F = P \cap H(a, b)$ is a facet. The facet-defining inequalities are the most important ones to describe the polytope P. In fact, every polytope (fully-dimensional) is the intersection of the halfspaces defined by their facet-defining inequalities. One can show that this description is minimal with respect to the number of halfspaces.

2.4 Algorithms and Complexity Theory

Complexity theory investigates how difficult is the problem and the efficiency of the algorithms used to solve it. We briefly introduce the notations to these topics. For further information, [9] gives a good survey.

2.4.1 Algorithms, decision problems, complexity

An *algorithm* can be interpreted as a finite set of instructions on how to solve a problem, where each instruction contains a sequence of elementary steps on the input data, e.g., plus, minus, etc.

We introduce a function $t_c : \mathbb{N} \to \mathbb{R}$ in order to measure the number of elementary steps needed by an algorithm, and this function is called the *time complexity* of the algorithm, or *complexity* for short. This function provides for each input of size $n \in \mathbb{N}$ the maximal elementary steps $t_c \in \mathbb{R}$. We denote an algorithm has a complexity of $\mathcal{O}(g(n))$ if there exists two constants $c \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$, such that $t_c(n) \leq c \cdot g(n)$, $\forall n \geq n_0$. If $g(n)$ is a polynomial function on n, then we say t_c is a *polynomial-time* algorithm. Or equivalently, the problem is *polynomial-time solvable*.

A *decision problem* can be posed as a "yes" or "no" problem. Decision problems are mainly divided into two classes, namely classes $\mathcal{P}$ and $\mathcal{NP}$. The set of all polynomial solvable decision problems is denoted by $\mathcal{P}$. The class $\mathcal{NP}$ is defined as the set of all decision problems for which each input with an answer "yes" can be verified in polynomial time. Although it is clear that $\mathcal{P} \subseteq \mathcal{NP}$, it is still an open question if the opposite holds. That is, we do not know if $\mathcal{P}$ equals $\mathcal{NP}$ or not.

Given two decision problems D and D^*, D is said to be *polynomial-time reducible* to D^* if there exists a polynomial algorithm that transforms each instance of D to D^* such that they always have the same answer. Informally, we say problem D is not harder than problem D^*.

A decision problem D is called $\mathcal{NP}$-complete if every problem in $\mathcal{NP}$ can be polynomial reducible to D. Informally speaking, $\mathcal{NP}$-complete problems are the most difficult problems in class $\mathcal{NP}$.

2.4.2 Complexity of optimization problems

An *optimization problem* (or more precisely, a minimization problem) can be defined as the following:

$$\min_{x \in X} f(x)$$

where X is a set of feasible solutions to the problem, and f is a real-valued function on X.

We then assign the following decision problem to the optimization problem: "Given a constant $c \in \mathbb{R}$, is there an $x \in X$ such that $f(x) \leq c$". This decision problem is polynomial-time reducible to the optimization problem. Hence, the notion of polynomial-time reducible can also be applied to an optimization problem. We call an optimization problem Opt $\mathcal{NP}$-hard if there exists an $\mathcal{NP}$-complete decision problem that can be polynomial-time reducible to Opt. That is, if every decision problem in $\mathcal{NP}$ is polynomially reducible to Opt.

When solving an optimization problem, an *exact algorithm* is one that always solves the problem to optimality when no time limit is constrained. On the contrast, a *heuristic algorithm* is one that not necessarily finds the optimal solution, or even if it finds the optimal one, there is no proof for that.

2.5 Mathematical Programming

In mathematics, computer science and operations research, *mathematical programming* (MP) or *mathematical optimization*, is to select a best element (with regard to a certain criterion) from a given set of available alternatives. We briefly review two special cases of MP, i.e., linear programming and mixed integer linear programming.

2.5.1 Linear programming

Linear Programming (LP) deals with obtaining the best outcome with respect to some requirements in forms of linear relationships. More formally, it has the following form:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{R}^n, \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Other forms of LP, such as maximization instead of minimization, great than equal to or equal restrictions, can be easily transformed to the above form.

The entries of the vector $x = (x_1, x_2, \ldots, x_n)$ are called *variables* of the LP, the m conditions $Ax \leq b$ *constraints*, and the linear function $x \to c^\top x$ *objective function* of the LP. The set $S := \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is called *feasible region*, and its elements *feasible solutions* of the LP. If $S \neq \emptyset$, the LP is *feasible*. If an LP is feasible, a feasible solution x^* is called *optimal solution* if and only if $c^\top x^* \leq c^\top x$ for any feasible solution x. The corresponding value $c^\top x^*$ is named *optimal value* of the LP. Note that it is possible for more than one feasible solutions x with $c^\top x = c^\top x^*$. If for any $\alpha \in \mathbb{R}$, there exists a feasible solution x such that $c^\top x \geq \alpha$, then the LP is called *unbounded*.

One key observation is that the feasible region of the LP is a polyhedra, and thus convex. If the LP has an optimal solution, this solution always occurs on the vertex.

There exists efficient algorithms for solving linear programming problems. Namely, the *simplex method*, *interior point method*, and *ellipsoid method*. The simplex method has exponential-time complexity in the worst case, while both the interior point and ellipsoid algorithm are polynomial-time solvable. But in practice, the simplex algorithm is found be remarkably efficient and the runtime is often polynomial. Hence, it is widely used to solve LPs.

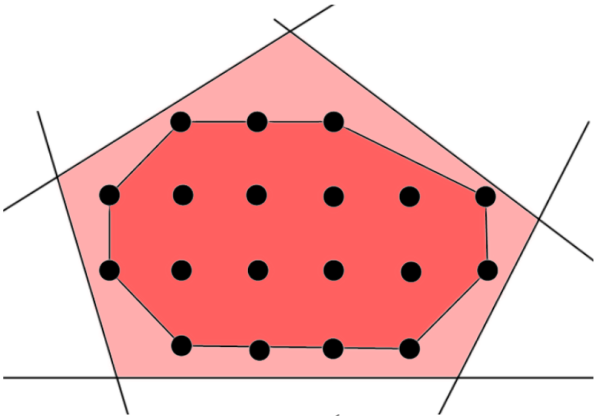


Figure 2.1 – The feasible solutions is plotted in black dots, the integer hull P_I is colored in red, and is contained in the feasible region P, which is colored pink.

2.5.2 Mixed integer linear programming

A *mixed integer linear program* (MILP) can be seen as an extension of a linear program, by exchanging some of its continuous variables to discrete ones. It has the following form:

$$\begin{aligned} \min \quad & c^\top x + d^\top y \\ \text{s.t.} \quad & Ax + By \leq e \\ & x \in \mathbb{R}^m, \ y \in \mathbb{Z}^n, \end{aligned} \tag{2.1}$$

where the requirements $y \in \mathbb{Z}^n$ are called the *integrality constraints*. If there exist only discrete variables, it is called an *integer linear program* (ILP). If $y \in \{0,1\}^n$, we have a *binary linear problem*.

A *combinatorial optimization problem* (COP) is to identify an optimal solution from a finite set of feasible solutions. It covers a wide range of practical problems. Most COPs can be formulated based on graphs, and then defined as MILPs.

Despite the similarity to the LPs, MILPs are much harder: general MILPs belong to the class of $\mathcal{NP}$-hard optimization problems.

For MILPs, the feasible region is naturally defined as $P(A,B,e) := \{x \in \mathbb{R}^m,\ y \in \mathbb{Z}^n \mid Ax + By \leq e\}$, and the LP relaxation of the feasible region as $P_L(A,B,e) := \{x \in \mathbb{R}^m,\ y \in \mathbb{R}^n \mid Ax + By \leq e\}$.

If $P(A,B,e)$ is bounded, the feasible region of the MILP is finite. We define the *integer hull* $P_I(A,B,e)$ as

$$P_I(A,B,e) := \text{conv}(\{x \in \mathbb{R}^m,\ y \in \mathbb{Z}^n \mid Ax + By \leq e\}). \tag{2.2}$$

which is the convex hull of the feasible region.

The associated polyhedra $P_I(A, B, e)$ satisfies

$$P(A, B, e) \subseteq P_I(A, B, e) \subseteq P_L(A, B, e),$$

with the inclusion being proper in general cases. One example is shown in Fig. 2.1, where we only have integer variables.

2.5.3 Solution methods for MILPs

The solution methods can be grouped into two types: exact and *non-exact* methods. Non-exact methods are usually heuristics or approximation methods. For any MILP, one can usually design a greedy or local search algorithm which can be used to compute an initial solution. Unfortunately, under the greedy mechanism, one can easily get trapped in a local minimal. An improved type of heuristic is called *meta-heuristics*, such as *tabu search* and *simulated annealing*.

We will focus on two of the exact methods, namely, the *cutting-plane method* and the *branch-and-bound method*. Before describing these two methods, we first introduce the notions of relaxation and separation problems.

A *relaxation* of an optimization problem $\min_{x \in X} f(x)$ is the optimization problem $\min_{x \in X'} f(x)$, where $X \subseteq X'$. It immediately follows that if x^* is an optimal solution of the original problem and x' is an optimal solution of the relaxation, then

$$f(x') \leq f(x^*), \tag{2.3}$$

i.e., solving the relaxed minimization problem yields a lower bound for the original one.

The reason for considering the relaxation of the original problem is that the relaxed optimization problem may be easier to solve. For a MILP

$$\min \{c^\top x + d^\top y \mid Ax + By \leq e,\ x \in \mathbb{R}^m,\ y \in \mathbb{Z}^n\}$$

a natural relaxation is the so called *linear programming relaxation* (LP-relaxation)

$$\min \{c^\top x + d^\top y \mid Ax + By \leq e,\ x \in \mathbb{R}^m,\ y \in \mathbb{R}^n\}$$

which relaxes the integrality constraints of y to $y \in \mathbb{R}^n$.

We adopt the notation $P(A, B, e)$, $P_I(A, B, e)$ and $P_L(A, B, e)$ from Section 2.1 to denote the feasible set of the MILP, convex hull and LP-relaxation of the feasible set, respectively.

A *separation* problem associated with a MILP $\min\{c^\top x + d^\top y \mid x, y \in P(A, B, e)\}$ is the problem: given $(x', y') \in \mathbb{R}^{m \times n}$, is $(x', y') \in P_I(A, B, e)$? If note, the separation problem finds a valid inequality $\pi^\top x + \omega^\top y \leq \pi_0$ for $P_I(A, B, e)$, but violated by the point (x', y'). This inequality is then added to the problem that "cuts off" the current infeasible solution.

Cutting-plane method

We briefly describe the cutting-plane procedure: when solving a MILP, we first start with solving a relaxation of the problem. If the obtained solution (x', y') satisfies all the constraints of the MILP, it is then also the optimal solution of the original problem. However, this is usually not the case. The basic idea of a cutting-plane method is to come up with a separation problem with respect to the current solution and the original MILP. The valid inequality of this separation problem is called a *cutting plane* or simply a *cut*, since it "cuts" off (x', y') from $P_I(A, B, e)$. We then add this valid inequality to the current relaxation problem and this procedure is iterated, until the obtained solution satisfies all the constraints. This process is shown in detail in Algorithm 2.1.

The crucial point is how to solve the separation problem efficiently at each iteration, and if the cutting plane generated could cut off a large area of the relaxation polyhedra $P_L(A, B, e)$. Ideally, good candidates for such cuts are those which define the facets of $P_I(A, B, e)$. However, one needs to balance the efficiency of finding a cut, and the quality of the cut. Another thing to note is that it may not be useful to add many facet-defining inequalities all at once, but only to add those that cut off the current solution of the relaxation problem. Finally, it is not practical to stop the loop of separation problem until no violated cuts can be found. For instance, one could stop once the solution of the relaxation has improved, which could still be useful to the branch-and-bound method to be discussed in the next section.

There exists different ways of generating the cutting planes, and the first cutting plane procedure was developed in the 1950s by Gomory [10]. Gomory was able to specify an easy way to generate such cuts that guarantees to find an feasible solution to $P_I(A, B, e)$ within a finite number of iterations.

Branch-and-bound method

Branch-and-bound method is a general approach for solving MILPs. It is based on the following two conclusions:

- Consider the objective value $z = \min_{x,y} \{c^\top x + d^\top y \mid (x, y) \in P(A, B, e)\}$, let $P = P_1 \cup P_2 \ldots \cup P_K$ be a decomposition of P into subsests, and let $z^k = \min_{x,y} \{c^\top x + d^\top y \mid (x, y) \in P_k\}$ for $k = 1, 2 \ldots, K$. Then $z = \min_k z^k$.

- Let $\underline{z}^k$ be a lower bound on z^k, and $\bar{z}^k$ an upper bound on z^k. Further let $\underline{z} = \min_k \underline{z}^k$ and $\bar{z} = \min_k \bar{z}^k$. Then $\underline{z}$ is a lower bound and $\bar{z}$ an upper bound on z.

As the name suggests, it consists of two main steps: branching and bounding. The key idea behind this algorithm is to split the original hard problem into easier smaller problems, called $subproblems$. The subproblems can be solved, fathomed or split into subproblems again. This leads to the constriction of a branch-and-bound tree with the

Algorithm 2.1 Cutting plane method

Initialize : $t \leftarrow 0, P^0 \leftarrow P_L(A, b, e)$.
while (stopping criterion not reached) **do**
  Solve the LP $(x^t, y^t) = \text{argmin}_{x,y}\{c^\top x + d^\top y \mid (x, y) \in P^t\}$.
  if $y^t \in \mathbb{Z}^n$ **then**
    (x^t, y^t) is an optimal solution.
    break
  else
    Solve the separation problem with respect to (x^t, y^t) and $P_I(A, B, e)$.
    if $\exists$ a valid inequality ${\pi^t}^\top x + {\omega^t}^\top y \leq \pi_0^t$ that cuts off (x^t, y^t) **then**
      $P^{t+1} \leftarrow P^t \cap \{{\pi^t}^\top x + {\omega^t}^\top y \leq \pi_0^t\}$.
      $t \leftarrow t + 1$.
    else
      break
    end if
  end if
end while

LP relaxation of the MILP as the root node. In addition, we keep track of the global lower and upper bound, which helps to fathom some branches of the tree and also provides a relative gap called the *optimality gap*. In a minimization problem, It is computed as follows:

$$optimality\ gap = \frac{|\bar{z} - \underline{z}|}{\bar{z}}.$$

The method terminates if there is no more subproblems to be solved, or if the optimality gap is relativity small. The advantage of this method is that the basic principle is simple, and it can be adopted to solve any MILP. Moreover, one can use any heuristic or approximation algorithm to compute an initial solution as an upper bound.

We can then solve the MILP in the following way. First, we solve the LP-relaxation of the MILP, which serves as the root node of the branch-and-bound tree. If the resulting solution satisfies all the integrality constraints, we have already solved the original problem. If not, we select variable y_i for some $i \in [n]$ which has fractional value y^* in the LP solution. We then split the current problem into two LP subproblems, one with constraint $y_i \leq \lfloor y_i^* \rfloor$, and the other with constraint $y_i \geq \lceil y_i^* \rceil$ added. We then execute the same procedure of splitting current node into two new LP subproblems, if the resulting solution is not feasible to the original MILP. Meanwhile, if a subproblem gets a feasible solution, we update the global upper bound $\bar{z}$. If it has fractional solution, we fathom the current node and update the global lower bound $\underline{z}$. The process stops when there exists no unsolved subproblems, or if $\bar{z} \doteq \underline{z}$. Denote the set of all subproblems $\mathcal{P}$,

the details of this algorithm can be seen in Algorithm 2.2.

Algorithm 2.2 Branch-and-bound method

1: **Initialize** : $\underline{z} \leftarrow -\infty, \bar{z} \leftarrow +\infty, \mathcal{P} \leftarrow P_L(A, b, e)$.
2: **while** $\mathcal{P} \neq \emptyset$ **do**
3: Choose a subproblem $P \in \mathcal{P}$ and solve the LP-relaxation. Denote the optimal solution and objective value as (x^*, y^*) and z^*, if they exist. $\{$*Subproblem selection*$\}$
4: **if** P is infeasible or $z^* \geq \bar{z}$ **then**
5: Fathom P. $\{$*Fathoming*$\}$
6: **else if** $y^* \in \mathbb{Z}^n$ and $z^* < \bar{z}$ **then**
7: $\bar{z} \leftarrow z^*$. $\{$*Bounding*$\}$
8: Fathom P. $\{$*Fathoming*$\}$
9: **else**
10: Select one fractional variable y_i and add two new subproblems: $P \cap \{y_i \leq \lfloor y_i^* \rfloor\}$ and $P \cap \{y_i \geq \lceil y_i^* \rceil\}$ to $\mathcal{P}$. $\{$*Branching*$\}$
11: Fathom P. $\{$*Fathoming*$\}$
12: Compute $\underline{z}^* = \min_k \underline{z}^k$ for the LP-relaxation of both subproblems.
13: **if** $\underline{z}^* > \underline{z}$ **then**
14: $\underline{z} \leftarrow \underline{z}^*$. $\{$*Bounding*$\}$
15: **end if**
16: **end if**
17: **end while**

The critical issue in Algorithm 2.2 is how to select the next subproblem in Step 3 and how to split the current node in Step 10. Different problem selection and branching strategies will lead to various branch-and-bound trees, thus different solution time. The recent paper [11] adopts supervised machine learning approach to learn how to branch from previous experiences, and it outperforms existing heuristic based branching strategies on benchmark instances.

Branch-and-cut method

Branch-and-cut is a technique that combines the branch-and-bound with the cutting plane method. It is the building block of many modern MILP solvers, hence is of great practical importance.

We will only state the basic idea of this method as follows: First, we compute the LP-relaxation of the MILP. Different from the pure branch-and-bound method, before branching, we first successively add violated inequalities (cutting planes) to strengthen the LP relaxation (cutting phase). This phase is stopped when no cuts can be found or a user-set time limit is hit. Then we begin with the branching phase that works as

in Algorithm 2.2 and we obtain two new subproblems. The above procedure is then applied to the next subproblem until no subproblem exists or if the optimality gap is almost 0.

2.6 Constant and Affine Regression

Given n signals $p = (p_1, p_2, \ldots, p_n)$ in a d-dimensional space, we denote their coordinates $z = (z^1, \ldots, z^n)^T \in \mathbb{R}^{n \times d}$ and intensities $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$. In this thesis, we are interested in the cases $d \in \{1, 2\}$, and signals p corresponding to pixels in a regular grid when $d = 2$. In this case, y_i denotes the RGB color grayscale value of pixel p_i. For computational efficiency, we restrict ourselves to the grayscale image in this thesis.

2.6.1 Parametric affine regression

In statistics, *affine (linear) regression* or *linear fitting* is a widely used approach to model the relationship between the dependent variables y and independent variables z. In the parametric model, the relationship is modeled using linear functions and the unknown linear parameters (i.e., slopes and intercepts) β are estimated from the given data according to some objective functions, such as the mean square error (MSE) [12]. It is also called *linear least squares*.

For instance, when $d = 1$, if we denote the fitting value of signal p_i as w_i, then the mean square error $S = \sum_{i=1}^{n}(y_i - w_i)^2$. Here $w_i = \beta_1 * z_i + \beta_0$ and $\beta = (\beta_1, \beta_0)$ is the linear parameter of the fitting line, where $\beta_1 \in \mathbb{R}^d$ is the slope (gradient) and $\beta_0 \in \mathbb{R}$ the intercept. The only two variables here are β_1 and β_0. So, the following minimization problem

$$\min_{\beta_1, \beta_0} \sum_{i=1}^{n} (\beta_1 * z_i + \beta_0 - y_i)^2 \tag{2.4}$$

is an unconstrained quadratic programming problem. We can get the analytical solution by setting its gradient to zero. The partial derivatives of MSE with respect to β_1 and β_0 are:

$$\frac{\partial S}{\partial \beta_1} = 0 = 2 \sum_{i=1}^{n} z_i \, (\beta_1 * z_i + \beta_0 - y_i),$$

$$\frac{\partial S}{\partial \beta_0} = 0 = 2 \sum_{i=1}^{n} (\beta_1 * z_i + \beta_0 - y_i).$$

This results in a system of two equations with two unknowns, which can be easily solved. When $\beta_1 = 0$, it becomes a constant fitting problem.

In general, let $\mathbf{1} = (1, 1, \ldots, 1) \in \mathbb{R}^n$ and $Z = [Z' \ \mathbf{1}] \in \mathbb{R}^{n\times(d+1)}$, where $Z' = (z_1, z_2 \ldots, z_n)^T \in \mathbb{R}^{n\times d}$. The affine least squares is to find the optimal $\beta = (\beta_1, \beta_0) \in \mathbb{R}^{d+1}$ that has the least mean square error of the following overdetermined system:

$$(Z^\top Z)\ \beta = Z^\top y.$$

Finally, β is the coefficient vector of the least-squares hyperplane, expressed as

$$\beta = (Z^\top Z)^{-1}\ Z^\top y.$$

2.6.2 Nonparametric affine regression

Non-parametric affine (linear) models compute w without explicitly modeling the affine parameters β as variables. Let $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$ be the fitting values (unknown variables) of the input signals, the non-parametric linear least squares problem becomes

$$\min \ \sum_{i=1}^{n} (w_i - y_i)^2 \tag{2.5}$$

$$w_i - 2w_{i+1} + w_{i+2} = 0, \quad i \in [n], \tag{2.5a}$$

$$w_i \in \mathbb{R}, \quad i \in [n-2], \tag{2.5b}$$

where (2.5a) implicitly enforces all fitting values lie in the same linear function with respect to the coordinates (to be discussed with more details in Chapter 4). If it is a constant fitting problem, we just need to replace (2.5a) with the equality constraints $w_i - w_{i+1} = 0$, for all $i \in [n-1]$.

Note that we could use any norm other than the ℓ_2 norm. In this thesis, we will use the ℓ_1 as our data term. We will explain this in more details in the next chapter.

2.7 Decomposition via Superpixels

Nowadays, typical sizes of images easily exceed millions of pixels. In this thesis, we are mostly dealing with MILPs, which means the number of integer variables could be millions if we deal with original images. Hence efficient decomposition methods are crucial to us.

On the one hand, one can use typical integer programming decomposition techniques, such as cutting planes and column generation [13] method, which decompose a large problem into simpler ones while still being equivalent to the original. In addition, they can be applied to any general MILP.

On the other hand, since the main application in this thesis is image processing, we will introduce a decomposition technique that is solely developed for this task. This

reduction is implemented on the input data, in particular, a grid of pixels of the image. This preprocessing cannot be reversed in the optimization step of the reduced model, implying that it is not an equivalent reformulation: an optimal solution of the decomposed model is not necessarily an optimal solution of the original one, thus a trade off between complexity and accuracy.

2.7.1 Superpixels

While the exact definition is vague, a *superpixel* is regarded as a group of perceptually meaningful connected blocks of an image. A superpixel should contain pixels similar in color or other properties, which are likely to belong to the same physical world object. The concept of superpixels was introduced by [14], and was motivated by two aspects: firstly, a grid of pixels is not a natural representation of real world scenes, but just a digital imaging "artifact"; and secondly, the huge number of pixels in natural images prevents many computer vision algorithms being computationally efficient or even possible.

Superpixels are usually used as a pre-processing step to speed up computations. Computational efficiency comes from the reduction in the number of elements of a given image, with the superpixel then being treated as a single variable. Superpixels thus have been actively applied for a wide range of applications, and there exist many contributions, see [15, 16] for an overview. Superpixels can be naturally obtained as the results of some image segmentation algorithms. To decrease the risk of the superpixels crossing object boundaries, these algorithms are applied in an over-segmentation mode. Examples are graph based[17], and normalized cuts[18].

Ideally, the following properties are desired for any superpixel algorithms, i.e., superpixels (or superpixel algorithms) should

- adhere well to image boundaries,
- be regular in shape and size, with smooth boundaries,
- be able to control its number and size,
- be non-overlapping and thus each pixel is assigned a label,
- represent connected sets of pixels,
- have few parameters, so that it can be easily adjusted,
- be fast to generate.

There are advantages of superpixels with regular shapes and sizes. Apart from visual appealing, in case a superpixel does cross the boundary, since the size is controlled, the error rate as well.

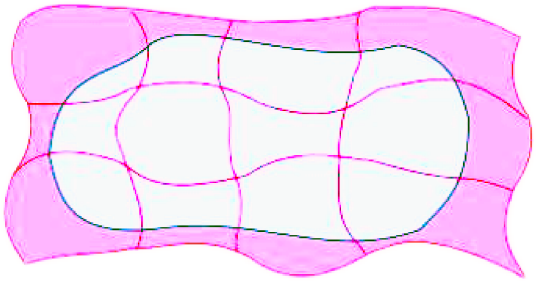


Figure 2.2 – Illustration of the "leakage" of superpixels, which is measured by the undersegmentation error. The ground-truth segment is with the black border, and the red lines are superpixel boundaries. In this example, the areas shaded pink count towards the undersegmentation error.

2.7.2 Evaluation metrics for superpixels

Apart from being visually appealing, there are some quantitative measurements to evaluate different superpixel algorithms. Given ground-truth image segmentations (usually annotated by a human), we will introduce and use three of them in this thesis, namely, the undersegmentation error, boundary recall, and compactness score. Of course, these are not the only evaluation metrics in the literature. For more details on evaluating superpixel algorithms, we refer the reader to a recent survey paper [16].

Undersegmentation Error

The *undersegmentation error* (UE) measures the amount of "leakage" of superpixels when placed over ground-truth segments, which also implicitly measures boundary adherence. Given ground-truth segments $G_1, G_2, \ldots, G_M$ and superpixels of any algorithm $S_1, S_2, \ldots, S_L$, the undersegmentation error UE is defined as

$$\mathrm{UE}(G, S) := \frac{1}{N} \left(\sum_{G_i} \sum_{S_j \cap G_i \neq \emptyset} \min \left\{ |S_j \cap G_i| , |S_j - G_i| \right\} \right), \tag{2.6}$$

where N is the number of pixels of the image. Overall, lower UE is preferred. This is illustrated in Figure 2.2.

Boundary Recall

The *boundary recall* (Rec) is the most commonly used metric to measure the boundary adherence of the superpixel to the ground-truth segmentation. It computes the fraction of pixels on the boundary of ground-truth segments that lies within a small distance of t away from any superpixel boundary, and higher Rec is preferred. Rec is defined as

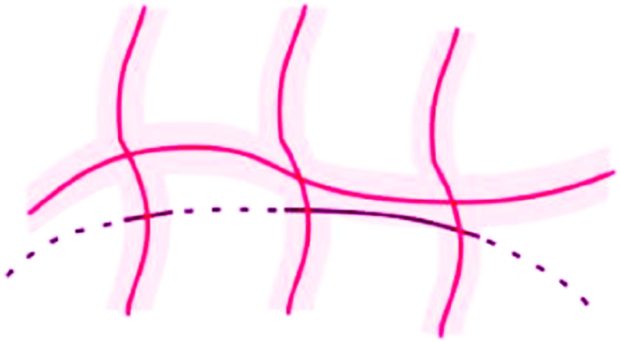


Figure 2.3 – Illustration of the boundary recall: The boundary of a ground-truth segment (black) counts towards the boundary recall (without the dotted black lines) if it is within a distance of t to the boundary of a superpixel (pink).

$$\text{Rec}(G,S) := \frac{\sum_p \mathbb{1}\left[p \in \partial G_i \text{ for some i}\right] \cdot \mathbb{1}\left[d(p, \partial S_j) \leq t \text{ for some j}\right]}{\sum_p \mathbb{1}\left[p \in \partial G_i \text{ for some i}\right]}, \tag{2.7}$$

where ∂G_i and ∂S_j denote the border of the segments, d is the minimal distance of a pixel p to the border of a segment, and t is a user-defined threshold. In this thesis, we will follow the convention in the paper [16] and let $t = 3$. See Figure 2.3 for an example.

Compactness Score

The *compactness score* (CO) compares the area $A(s_i)$ of each superpixel s_i with the area of a circle with the same perimeter $P(s_i)$ of the superpixel. The later forms the most compact 2-dimensional shape, and a higher CO is usually desired. It is invariant to the ground-truth segmentations. CO is computed as

$$\text{CO}(G,S) = \frac{1}{N} \sum_{S_i} |S_i| \frac{4\pi A(S_i)}{P(S_i)}. \tag{2.8}$$

Chapter 3

Piecewise Constant Potts Model for Segmentation and Denoising

Image segmentation and denoising are two fundamental tasks in computer vision and image processing. Graph based models are popular for segmentation and variational models are employed for denoising. Our approach could address both problems at the same time. In this chapter, we propose a novel Mixed Integer Linear Programming (MILP) formulation of the discrete first derivative (piecewise constant) Potts model with ℓ_1 data term, where binary variables are introduced to deal with the ℓ_0 norm of the regularization term. We utilize the branch-and-cut method for global optimum, as well as a fast heuristic algorithm for approximate solutions that is based on the region fusion algorithm [4]. The MILP is solved by a standard off-the-shelf MILP solver, i.e., Cplex [19]. Computational experiments are conducted against the multicut problem and using different variants of our proposed model. We also apply our method to generate superpixels on noisy images. Extensive experiments are carried out on the BSDS500 [20] image dataset and compared with other superpixel methods. Our method achieves the state-of-the-art in terms of a combined score (OP) composed of the under-segmentation error, boundary recall and compactness.

3.1 Background

The image segmentation problem, also known as partitioning, grouping, or clustering, is a fundamental problem in image processing. It contains the task of dividing an image into either fixed or unfixed number of non-overlapping regions. Problem representations are usually based on a graph $G = (V, E)$, where nodes V relate to pixels or superpixels in an image, and E represents the set of edges consisting of unordered pairs of nodes indicating adjacency relations. A segmentation problem can be represented either by

- *node labeling*: assigning a label to each node $v \in V$, or by

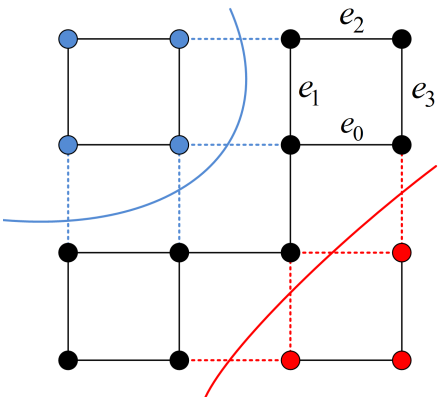


Figure 3.1 – Two representations of an image segmentation: node labeling (by their colors) and edge labeling via multicuts (dashed edges).

- *edge labeling*: a multicut defined by a subset of edges $E' \subseteq E$, which also results in a partition of a set of nodes, as can be seen in Figure 3.1.

We assume that the input is an image with pixels located on a grid. An image segmentation is then a partition of V into sets $\{V_1, V_2, \ldots, V_k\}$ such that $\cup_{i=1}^{k} V_i = V$, and $V_i \cap V_j = \emptyset$, $i \neq j$. So in graph-theoretical terms, the problem of image segmentation corresponds to graph partitioning.

One often distinguishes between *supervised* and *unsupervised* segmentation. In the former case, the number of classes (e.g., person, grass, sky, etc) defined by labels is pre-defined, together with a function (often called the unary data term) measuring how likely a node belongs to each class. Among many existing supervised models, the Markov Random Field (MRF) is well studied and applied, interested readers may refer to [21, 22, 2] for an overview of this field. In the latter unsupervised case, such class information is missing. This introduces ambiguities when node labeling is used. Think about the node labeling in Figure 3.1, if we permute the labels (colors), it will result in the same segmentation. However, edge labeling (e.g., by multicuts) does not exhibit such symmetries and is therefore more appealing in the unsupervised case.

In this chapter, we focus on the problem of partitioning a given image into an unknown number of segments using edge labeling. Exact optimization algorithms such as the *multicut problem* [3, 23] and the *lifted multicut problem* [24] with positive or negative edge weights are based on the Integer Linear Programming (ILP) formulation. They are in general $\mathcal{NP}$-hard, and globally optimal solutions can be solved iteratively using branch-and-cut methods. More efficient fusion move algorithm are adopted in [25, 26].

Denoising of images is one of the most basic image restoration problems. While some denoising approaches [27, 28] either estimate every pixel separately by fusing other "similar" neighboring pixels, or denoise several similar patches simultaneously. Global models like the total variation model [29] by Rudin, Osher, and Fatemi is one of

the best known variational denoising model. Let $D \in \mathbb{R}^d$ denote the signal domain, and $y : D \to \mathbb{R}$ denote the given signals' intensity values possibly with noise. Total variation [29] basically approximates the input signals y with a piecewise smooth function $w : D \to \mathbb{R}$, and is stated as

$$\min_w \int_D (w(z) - y(z))^2 dz + \lambda \int_D |\nabla^1 w| dz. \tag{3.1}$$

Here, z denotes the coordinate system of the signals (horizontal and vertical-axis if the input is a 2D image), and the first part represents the data fidelity term (how well w fits y), while the second is the regularization term, with $\lambda > 0$ the user defined penalty term. Recall that $\nabla^1 = (\partial_{z_1}, \partial_{z_2})$ represent the differential operation of the first order in the 2D case.

With (3.1) being convex, it can be efficiently solved with structured convex solvers. However, the ℓ_1 regularization term sometimes over-penalizes the sharp discontinuities between two distinct regions in an image. In these cases, denoising approach with Potts priors [1] is designed to preserve the sharp discontinuities while removing noises and thus more desirable.

Given two nodes $p, q \in V$, their continuous values $w(p), w(q)$ and a constant λ_{pq} that depends on p, q, the *Potts function*

$$\theta_{pq}(w(p), w(q)) = \lambda_{pq} \cdot \mathbb{1}(w(p) \neq w(q))$$

is discontinuity-preserving and widely used in computer vision. Recall that $\mathbb{1}(\cdot)$ is 1 if its argument is true and 0 otherwise.

We are most interested in the discrete setting of the Potts model [1]. Given n signals $[n]$ (recall that $[n]$ denotes the discrete set $\{1, 2, \ldots, n\}$), the classical (discrete) piecewise constant Potts model (named after R. Potts) has the form

$$\min_w \|w - y\|_k + \lambda \|\nabla^1 w\|_0, \tag{3.2}$$

where w, y denotes the n array vector, the data term measures their ℓ_k norm difference, and the regularization term measures the number of oscillations in w. Recall that the *discrete first derivative* $\nabla^1 w$ of a vector $w \in \mathbb{R}^n$ is defined as the $n-1$ dimensional vector $(w_2 - w_1, w_3 - w_2, \ldots, w_n - w_{n-1})$ and the ℓ_0 norm of a vector is its number of nonzero entries. Various modifications and improvements have been made for the Potts model, see [30] for an overview.

In general, solving the discrete version of the Potts model (3.2) is also $\mathcal{NP}$-hard. Approximate algorithms such as local greedy methods [4] and alternating direction method of multipliers (ADMM) [31] are used instead. Recently, [32] utilizes a MILP formulation to deal with a similar problem in statistics called the *best subset selection problem*. These papers all use the ℓ_2 data term, i.e., $k = 2$. In 1D, the Potts model

with ℓ_1 data term can be efficiently solved by dynamic programming. An $\mathcal{O}(n^2 \log n)$ algorithm is given in [33], an $\mathcal{O}(n^2)$ algorithm in [34], and an $\mathcal{O}(nk)$ algorithm in [35], where n is the number of input signals, and $k \leq n$ is the number of unique values in the signals.

Motivated by the discrete Potts (3.2) and the work of [31], we will look into the problem of simultaneously segmenting and denoising images. However, different from [4, 31] where ℓ_2 norm is used and only approximate solutions are solved, we focus on the ℓ_1 data term and the exact algorithms which leads to globally optimal solutions of (3.2). This is possible in terms of a MILP formulation which is $\mathcal{NP}$-hard. Hence we further introduce a fast region fusion [4] based heuristic with ℓ_1 data term.

Highlights of this chapter.

- We propose a MILP formulation for the discrete Potts model (3.2) with ℓ_1 data term, which solves it to global optimum.
- We prove the multicut constraints [3] are redundant for the optimal solutions of MILP formulation, but also facet-defining for the associated integer polytope.
- We propose a fast region fusion based heuristic algorithm for solving the Potts model with ℓ_1 data term.
- We apply our proposed method for generating superpixels on noisy images, and achieve state-of-the-art results on the proposed OP score.

3.2 MILP Formulation of the Piecewise Constant Potts Model

Given n signals $[n]$ in some interval $D \subseteq \mathbb{R}^d$ with intensities $y = (y_1, \ldots, y_n)$. We are most interested in the case when $d = 2$, where signals become image pixels and y represent pixels' color or gray-scaled values.

We call a function f piecewise constant over D if there exists a partition of D into subintervals $D_1, \ldots, D_k$ such that $D = \cup_{i=1}^{k} D_i$, where $D_i \cap D_j = \emptyset$, and f is constant when restricted to each D_i, for $i \in [k]$. Throughout this chapter, we assume the input signals or images contain noises. We treat the task of segmentation and denoising input signals as a piecewise constant fitting problem. We denote the fitting value for signal i as $w_i = f(i)$.

3.2.1 Formulation of 1D signals

For the 1D signals case, the associated graph $G(V, E)$ simply becomes a chain graph, where $V = \{i \mid i \in [n]\}$ and $E = \{e = (i, i+1) \mid i \in [n-1]\}$. Denote two end nodes

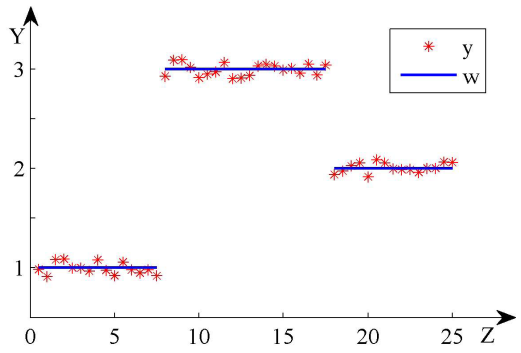


Figure 3.2 – 1D piecewise constant fitting, with 3 segments and 2 active edges.

of an edge e as h_e and t_e. We would like to formulate the Potts model (3.2) as a MILP, this is achieved by introducing $n-1$ binary variables

$$x_e = \begin{cases} 1, & \text{if } h_e, t_e \text{ are in different segments} \\ 0, & \text{otherwise,} \end{cases}$$

and the following properties should hold

$$\nabla^1 w_e = 0 \Leftrightarrow x_e = 0, \quad \forall e \in E, \tag{3.3}$$
$$\nabla^1 w_e \neq 0 \Leftrightarrow x_e = 1, \quad \forall e \in E. \tag{3.4}$$

Here, $\nabla^1 w_e := w_{h_e} - w_{t_e}$. The edge e is called an *active* or *jump* edge if $x_e = 1$, otherwise it is *dormant*. Upon the above assumptions, the fitting values w satisfy the piecewise constant property that we desire, i.e., the signals between two active edges define one segment whose fitting values w share the same intensity and the number of segments equal $\sum_{e\in E} x_e + 1$. See Figure. 3.2 for one example, where there are two active edges and three segments.

The properties (3.3, 3.4) can be modeled via Mixed Integer Programming (MILP) using the "big M" technique, which leads to the following formulation:

$$\min \quad \sum_{i\in V} |w_i - y_i| + \lambda \sum_{e\in E} x_e \tag{3.5}$$
$$|\nabla^1 w_e| \leq M x_e, \qquad \forall e \in E, \tag{3.5a}$$
$$w_i \in \mathbb{R}, \qquad \forall i \in V, \tag{3.5b}$$
$$x_e \in \{0, 1\}, \qquad \forall e \in E, \tag{3.5c}$$

where the first part of the objective function is the data fitting term, and the second is the regularization term. To prevent the model from over-fitting, a user defined parameter

$\lambda > 0$ is introduced to indirectly control the number of segments. The constant M in (3.5a) is usually called a "big M" constant, and it should be large enough so that (3.5a) is always valid when $\nabla^1 w_e \neq 0$.

Lemma 3.1. *The optimal solutions* $(w^\star, x^\star)$ *of problem* (3.5) *satisfy properties 3.3 and 3.4.*

Proof. Let $\vartheta = \sum_{i \in V} |w_i - y_i| + \lambda \sum_{e \in E} x_e$ be the objective value of problem (3.5), we proof sufficient and necessary conditions of 3.3 and 3.4.

1. $\nabla^1 w_e = 0 \Rightarrow x_e = 0$. If $\nabla^1 w_e = 0$, by constraint (3.5a), x_e can be either 0 or 1. But the optimality of the solution will enforce $x_e = 0$ since problem (3.5) is a minimization problem and $\lambda > 0$, and this makes ϑ smaller.

2. $x_e = 0 \Rightarrow \nabla^1 w_e = 0$. If $x_e = 0$, then it immediately follows by constraint (3.5a) that $\nabla^1 w_e \leq 0$, and hence $\nabla^1 w_e = 0$.

3. $\nabla^1 w_e \neq 0 \Rightarrow x_e = 1$. If $\nabla^1 w_e \neq 0$, it immediately follows by constraint (3.5a) that $x_e = 1$, where M is assumed to be big enough for (3.5a) to hold.

4. $x_e = 1 \Rightarrow \nabla^1 w_e \neq 0$. If $x_e = 1$, suppose we have $\nabla^1 w_e = 0$, then by part 1 of this lemma, $x_e = 0$, thus a contradiction.

□

Now we have proved that the optimal solution to (3.5) satisfies the piecewise constant property that we desire. Note that we use the ℓ_1 norm because it is more robust to outliers than ℓ_2 [36]. Moreover, it can be easily modeled with linear constraints. Namely, constraint (3.5a) is firstly replaced by the two constraints $\nabla^1 w_e \leq M x_e$ and $-\nabla^1 w_e \leq M x_e$. Secondly, the term $|w_i - y_i|$ is replaced by $\varepsilon_i^+ + \varepsilon_i^-$ where $w_i - y_i = \varepsilon_i^+ - \varepsilon_i^-$ and $\varepsilon_i^+, \varepsilon_i^- \geq 0$.

Problem (3.5) is then formulated as the following MILP

$$\min \sum_{i \in V} (\varepsilon_i^+ + \varepsilon_i^-) + \lambda \sum_{e \in E} x_e \tag{3.6}$$

$$\nabla^1 w_e \leq M x_e, \qquad \forall e \in E, \tag{3.6a}$$

$$-\nabla^1 w_e \leq M x_e, \qquad \forall e \in E, \tag{3.6b}$$

$$w_i - y_i = \varepsilon_i^+ - \varepsilon_i^-, \qquad i \in V, \tag{3.6c}$$

$$\varepsilon_i^+, \varepsilon_i^- \in \mathbb{R}^+, \qquad \forall i \in V, \tag{3.6d}$$

$$w_i \in \mathbb{R}, \qquad \forall i \in V,$$

$$x_e \in \{0, 1\}, \qquad \forall e \in E.$$

Lemma 3.2. *The MILP formulation* (3.6) *is equivalent to* (3.5)

Proof. The equivalence of constraint (3.5a) with constraints (3.6a, 3.6b) is quite obvious. Here, we concentrate on proving the second replacement.

Suppose $w_i \geq y_i$, let $w_i - y_i = c$ where c is a constant, so we have $|w_i - y_i| = w_i - y_i = c$. By constraint (3.6c), $\varepsilon_i^+ - \varepsilon_i^- = w_i - y_i = c$. Then $\varepsilon_i^+ = \varepsilon_i^- + c$, and $\varepsilon_i^+ + \varepsilon_i^- = 2\varepsilon_i^- + c$.

Since problem (3.5) is a minimization problem and by constraint (3.6d), the optimal solution will have $\varepsilon_i^- = 0$ and $\varepsilon_i^+ = c$. Hence, $\varepsilon_i^+ + \varepsilon_i^- = c = |w_i - y_i|$, which makes the objective function of problem (3.6) and (3.5) the same.

The case when $w_i \leq y_i$ can be proved similarly, and we omit the proof here. □

Note that this technique of equivalently transforming (3.5) to (3.6) will be used widely in this thesis, and from now on, we will just specify the MIP formulation in the form of (3.5).

The optimal solution of (3.6) gives the fitting value w_i for the node (signal) i and the boundaries of two segments (when $x_e = 1$). The segmentation is uniquely defined by the optimal solution $\mathbf{x}$, and we obtained denoised signals with value w simultaneously.

3.2.2 Formulation of 2D images

Given a 2D gray-scaled image, following notations from the previous section, the associated graph $G(V, E)$ is a grid graph. The first derivative Potts model in 2D is again modeled as a MIP, and could be formulated using exactly the same notation as (3.5). However, we propose to formulate it a bit differently as we will need it for introducing the cardinality constraints in Section (3.3.3).

Since G is a grid graph, we divide E into its row (horizontal) edge set E^r and column (vertical) edge set E^c so that $E = E^r \cup E^c$ and $E^r \cap E^c = \emptyset$. Suppose the input 2D image is of size $m \times n$. We further denote $E^r = \{E_1^r, E_2^r, \ldots, E_m^r\}$, where E_i^r is the set of row edges in the i-th row and $i \in [m]$. Similarly, we have $E^c = \{E_1^c, E_2^c, \ldots, E_n^r\}$.

Our proposed MIP in 2D is then obtained by formulating the 1D case (3.5) per row and column of the grid.

$$\min \quad \sum_{i \in V} |w_i - y_i| + \lambda \sum_{e \in E} x_e \tag{3.7}$$

$$|\nabla^1 w_e| \leq M x_e, \quad \forall e \in E_i^r, \ \forall i \in [m] \tag{3.7a}$$

$$|\nabla^1 w_e| \leq M x_e, \quad \forall e \in E_j^c, \ \forall j \in [n] \tag{3.7b}$$

$$w_i \in \mathbb{R}, \ \ \forall i \in V, \tag{3.7c}$$

$$x_e \in \{0, 1\}, \ \ \forall e \in E. \tag{3.7d}$$

Recall that $\nabla^1 w_e := w_{h_e} - w_{t_e}$, and Lemma 3.1 still hold true for each row and column of the $2D$ image grid. The solution of (3.7) again gives the fitting value w_i for each

node (pixel) i and the boundaries of two segments (when $x_e = 1$). We will prove in the next section that the image segmentation is uniquely defined by the optimal solution $\mathbf{x}$ of problem (3.7).

3.3 Redundant Constraints and Additional Cuts for the MILP in 2D

It is quite common to add redundant constraints to a MIP for computational speedup. A constraint is called *redundant* to a mathematical programming formulation if it is not necessarily needed for the formulation to be valid. However, it may be beneficial to add them because they forbid some fractional solutions during the branch-and-bound approach, where the MIP solver iteratively solves the linear programming (LP) relaxation. Or sometimes, these constraints impose a structure that help shrink the search space. One example is the symmetry-breaking constraints [37] that are widely used in combinatorial optimization problems.

On the other hand, if one knows some prior knowledge about the problem structure, it usually helps to add them into the formulation. One example is the user input in the interactive image segmentation [38, 39, 40], where the input is formulated as additional constraints in the mathematical formulation. We will just call these additional constraints as "cuts", since they "cut off" some original feasible but not optimal solutions.

3.3.1 The multicut problem

The multicut problem [3] is an unsupervised image segmentation problem, where no unary data term is present. Given the corresponding graph $G(V, E)$, recall that for a partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ of V, the multicut induced by $\mathcal{V}$ is the edge set $\delta(V_1, V_2, \ldots, V_k) = \{uv \in E \mid \exists i \neq j \text{ with } u \in V_i \text{ and } v \in V_j\}$. It formulates the graph partitioning problem as an edge labeling problem, and the result corresponds one-to-one to an image segmentation.

Like in Section 3.2.1, binary edge variables x_e are introduced and the multicut can be represented by a set of active edges. Let the edge weight $c : E \to \mathbb{R}$ represent the absolute differences between two pixels' intensities, the ILP of the multicut problem reads

$$\min \quad -\sum_{e \in E} c_e x_e + \sum_{e \in E} \lambda x_e \tag{3.8}$$

$$\sum_{e \in C \setminus \{e'\}} x_e \geq x_{e'}, \quad \forall \text{ cycles } C \subseteq E, e' \in C, \tag{3.8a}$$

$$x_e \in \{0, 1\}, \quad \forall e \in E, \tag{3.8b}$$

where constraints (3.8a) are called the *multicut constraints* and they enforce the consecutiveness of the active edges. As a result of (3.8a), the multicut uniquely define an image segmentation, and each maximal set of vertices induced only by dormant edges corresponds to one segment.

One example is shown in Figure 3.1, where the multicut of 8 dashed edges uniquely defines a partition of the 4×4-grid graph into 3 segments.

The multicut problem is $\mathcal{NP}$-hard in general and the number of inequalities (3.8a) is exponentially large with respect to $|E|$. One can solve 3.1 in a cutting plane manner, i.e., first ignore the multicut constraints (3.8a), and check the feasibility of the current integer solution. If it is infeasible, violated constraints can be found efficiently using shortest path algorithms. These cuts are added iteratively until the solution is feasible [3, 41].

3.3.2 Cuts and redundant constraints

Let $S := \{(w, x) \mid (w, x)$ is feasible to problem (3.7)$\}$ be the set of all feasible solutions to the discrete Potts model (3.7), and $S_I := \{x| \ (w, x) \in S\}$ denotes the integer set of S. Further denote S^o to be a subset of S, where $(w, x) \in S^o$ has to satisfy the property 3.3 and 3.4. Finally, we let S_I^o be the projection of $(w, x) \in S^o$ to x.

We first prove that the multicut constraints (3.8a) are redundant (thus valid) for S_I^o of the Potts model (3.7). We then point out that the multicut constraints (3.8a) are facet-defining for the convex hull of S_I^o.

Theorem 3.3. *The multicut constraints* (3.8a) *are redundant for S_I^o of the Potts model* (3.7).

Proof. We prove this by stating that every $x \in S_I^o$ satisfies the multicut constraints (3.8a).

As discussed in [3], the multicut constraints (3.8a) are equivalent to

$$\sum_{e \in C} x_e \neq 1, \ \forall \text{ cycles } C \subseteq E.$$

Suppose there exists one solution $(w, x) \in S^o$ and a cycle $C' = \{1, 2, \ldots, n\}$ of length $n - 1$ (here, node 1 and n represent the same node) such that there is only one active edge $e = (i, i + 1) \in C'$, $i \in [n - 1]$ (i.e., $x_{i,i+1} = 1$). By the definition of S^o, we have $w_i \neq w_{i+1}$.

Since all the other edges in cycle C are dormant ($x_e = 0$), by constraints (3.7a,3.7b), we have $w_1 = w_2 = \ldots = w_i$ and $w_i = w_{i+1} = \ldots = w_n$. Since node 1 and n represent the same node, we have $w_1 = w_2 = \ldots = w_n$, thus $w_i = w_{i+1}$, and a contradiction. □

Let $(w^\star, x^\star)$ be an optimal solution to the MIP (3.7). Similar to the proof in Lemma reflemma1, property 3.3 and 3.4 also hold true. Hence the optimal solutions $(w^\star, x^\star)$ of (3.7) lies in S^o, and $x^\star \in S_I^o$.

While the multicut constraints (3.8a) are redundant for the optimal solutions set (may not be unique) of (3.7), it is not necessarily valid for the feasible set of (3.7). For

instance, for the active edge cycle C' we constructed in Theorem 3.3, the solution where $x_{i,i+1} = 1$ and $w_i = w_{i+1}$ is feasible to (3.7). This means there exists a solution $x' \in S_I$ and correspond to this solution, there is one cycle C' such that $\sum_{e \in C'} x_e = 1$. This obviously violates the multicut constraints (3.8a). Hence the multicut constraints (3.8a) "cuts away" some feasible but non-optimal solutions of (3.7).

Since we know the multicut uniquely defines an image segmentation [3], we can now conclude that the optimal solution of the Potts model (3.7) also defines a unique segmentation.

Theorem 3.4. *Let S_m be the set of all feasible solutions to the multicut problem*(3.8) *with respect to the Potts model* (3.7)*, then $S_m = S_I^o$.*

Proof. By Theorem 3.3 we know $S_I^o \subseteq S_m$. The opposite also holds since for any feasible solution $x \in S_m$ of (3.8), x also defines a valid segmentation in the Potts model (3.7), and we can compute the corresponding w by by fitting a constant function within each segment. Thus we have constructed a feasible solution $(w, x) \in S^o$, which implies the existence of $x \in S_I^o$. □

Hence, we can also conclude that $x^\star \in S_m$, where $x^\star$ is the optimal solution of (3.7).

It is well known that if a cycle $C \in G$ is chordless, then the corresponding multicut constraint (3.8a) is facet-defining for the multicut polytope (convex hull of S_m) [3, 41]. In a 2D grid graph, the number of such constraints is still exponential. In Section 3.5, we will test the simple strategy of adding the following 4-edge chordless cycle constraints (see the cycle $e_0 - e_1 - e_2 - e_3$ in Figure 3.1 for an example)

$$\sum_{e \in C \setminus \{e'\}} x_e \geq x_{e'}, \quad \forall \text{ cycles } C \subseteq E, |C| = 4, e' \in C \tag{3.9}$$

to the Potts model (3.7). Because firstly, it is the simplest (in terms of the number of edges) chordless cycle in a grid graph. Secondly, the number of such constraints is only linear to the number of edges.

Of course, there exists more complicated strategy on finding and selecting such cuts, but this is beyond the focus of this chapter. Interested readers may refer to [42, 43, 44] for more information. Finally, although we have defined the set of facet-defining inequalities for S_I^o, we leave the further properties of the chordless cycles constraints with respect to S^o for future research.

3.3.3 Cardinality constraints as additional cuts

If the user has some prior knowledge about the problem and solution, one can add additional cuts to reduce the search space.

One example is the cardinality constraints. In 1D signals case, if one has some prior knowledge or good guess on the number of segments, it may be beneficial to add the following cardinality constraints

$$\sum x_e \leq k, \tag{3.10}$$

where k denotes the upper bound on the number of jumps. For instance, in Figure 3.2, if the user knows the number of segments is less than 3, then by adding $\sum x_e \leq 2$ to (3.5), one shrinks the search space of the combinatorial optimization problem from 2^n to $\frac{n(n-1)}{2} + \frac{n}{2} + 1$.

When working on 2D images, one can add the above cardinality constraints per row and column. We need some strategy of automatically computing the upper bound k though. We will show computational experiments in Section 3.5 on the effects of adding the above two types of constraints (3.9, 3.10) to the MILP (3.7).

3.4 Solution Techniques

Since the resulting MIP (3.7) problem is in general $\mathcal{NP}$-hard to solve, we first present a fast heuristic that could provide a feasible solution in short time. Then it can be adopted as a starting point for the MIP solver, which is then proceeded using the standard branch-and-cut algorithm.

3.4.1 Region fusion based heuristic with ℓ_1 data term

The authors of [4] present a region fusion based greedy algorithm that efficiently solve the Potts model (3.2) with ℓ_2 data term.

In this chapter, we are interested in the the Potts model (3.2) with ℓ_1 data term, and the corresponding energy function similar to that of the paper [4] reads

$$\min_{\mathbf{w}} \sum_{i \in V} \left(\|w_i - y_i\|_1 + \sum_{j \in N_i} \lambda \|w_i - w_j\|_0 \right),$$

where λ is the regularization parameter and N_i denotes the neighboring pixels of i. (recall that in a grid graph, we assume four-connected pixels.)

The algorithm starts with every pixel i belongs to its own group (segment) V_i, and at every iteration, the algorithm just examine the energy of two neighboring groups with current regularization parameter κ. The energy of two groups V_i, V_j equals

$$\min_{\mathbf{w}} \ \tau_i \|w_i - Y_i\|_1 + \tau_j \|w_j - Y_j\|_1 + \kappa \gamma_{ij} \|w_i - w_j\|_0, \tag{3.11}$$

where τ_i denotes the number of pixels in group V_i, and γ_{ij} represents the number of neighboring pixels between two groups V_i and V_j. We use Y_i to indicate the mean of

image data (e.g., color) within group V_i, w_i to denote the fitting value of V_i and κ express the current regularization parameter that increases in every iteration (up to λ).

Three growing strategies are reported in [4], and it is shown that the following non-linear strategy with growing rate 2.2 achieves the overall best result

$$\kappa(iter, K, \lambda) = (\frac{iter}{K})^{2.2}\lambda,$$

where K denotes the maximum iteration number and $iter$ is the current iteration number.

At every iteration, the following condition is checked

$$\tau_i \tau_j \|Y_i - Y_j\|_1 \leq \kappa \gamma_{i,j} (\tau_i + \tau_j) . \tag{3.12}$$

If satisfied, two groups i, j are merged which results in a lower energy, and $w_{i'}$ for the new merged group i' is computed as

$$w_{i'} = \begin{cases} Y_i, & \text{if } \tau_i \geq \tau_j, \\ Y_j, & \text{otherwise,} \end{cases} \tag{3.13}$$

which makes (3.11) minimum. If not, the iteration continues and the above condition is again checked. The overall approach is described in Algorithm 3.1.

3.4.2 Exact branch and cut algorithm

In this chapter, as well as the whole thesis, we use a standard MIP solver (such as Cplex) to solve the resulting MILP problem off the shelf. The main algorithm inside the "black box" solver is the branch and cut method described in Section 2.5.3.

The main problem to solve is the MILP formulation of the discrete Potts model (3.5), with or without the 4-cycle multicut constraints (3.9) as additional cuts.

The advantage of solving a MILP as opposed to any approximate algorithms is that, the MIP solver can use any feasible solution as an initial solution to start with. And by solving the LP-relaxation of the MILP, we obtain an *optimality gap* (defined in Section 2.5.3) that serves as a quality evaluation for the feasible solution.

The solver seeks for better solutions as the branch-and-cut trees proceeds, and one can set any time limit for the whole procedure. When the time limit is reached, one obtain a best feasible solution (given the time limit) as well as an optimality gap. Note that the separation problem with respect to the cutting plane algorithm inside the branch-and-cut is designed within the "black box" solver, and we have little control over it.

Algorithm 3.1 ℓ_0 region fusion algorithm with ℓ_1 data term

Input: Signals of length P, the regularization parameter λ

Initialize: $V_i \leftarrow \{i\}, Y_i \leftarrow y_i, \tau_i \leftarrow 1$
N_i as four-connected neighboring pixels
$\gamma_{i,j} = 1$ if $j \in N_i$, $\gamma_{i,j} = 0$ otherwise
$\kappa \leftarrow 0, iter \leftarrow 0$
repeat
    $i \leftarrow 1$
    while $i \leq P$ **do**
        for all $j \in N_i$ **do**
            if $\tau_i \tau_j \|Y_i - Y_j\|_1 \leq \kappa \gamma_{i,j} (\tau_i + \tau_j)$ **then**
                if $\tau_i \geq \tau_j$ **then**
                    $Y_i \leftarrow Y_i$
                else
                    $Y_i \leftarrow Y_j$
                end if
                $V_i \leftarrow V_i \cup V_j$
                $\tau_i \leftarrow \tau_i + \tau_j$
                Remove j in N_i and delete $\gamma_{i,j}$
                for all $k \in N_j \setminus \{i\}$ **do**
                    if $k \in N_i$ **then**
                        $\gamma_{i,k} \leftarrow \gamma_{i,k} + \gamma_{j,k}$
                        $\gamma_{k,i} \leftarrow \gamma_{i,k} + \gamma_{j,k}$
                    else
                        $N_i \leftarrow N_i \cup \{k\}$
                        $N_k \leftarrow N_k \cup \{i\}$
                        $\gamma_{i,k} \leftarrow \gamma_{j,k}$
                        $\gamma_{k,i} \leftarrow \gamma_{j,k}$
                    end if
                    Remove j in N_k and delete $\gamma_{k,j}$
                end for
                Delete V_j, N_j, τ_j
                $P \leftarrow P - 1$
            end if
        end for
        $i \leftarrow i + 1$
    end while
    $iter \leftarrow iter + 1$
    $\kappa \leftarrow g(iter, K, \lambda)$
until $\kappa \geq \lambda$

for $i = 1 \leftarrow P$ **do**
    for all $j \in V_i$ **do**
        $w_j \leftarrow Y_i$
    end for
end for

Output: Fitting value W of length P

3.5 Experiments on Small Instances

In this chapter and the whole thesis, all computational tests are performed on an Intel i5-4570 quad-core desktop with 16GB RAM. If MILP problems are present, we use IBM Cplex [19] of version $12.6.1$, which is a standard commercial MIP solver.

3.5.1 Multicut problem versus discrete Potts model (3.7)

Both the multicut problem (3.8) and the Discrete Potts model (3.7) are MILPs and $\mathcal{NP}$-hard to solve. However, as we will see later in this section, computational efforts to solve the two problems differ a lot.

We compare three different formulations in this section. The experiments are based on two images from [45], and we resize them to 40×40 and 41×58 respectively. We add Gaussian and salt and pepper noise, and set a time limit of 100 seconds.

- Formulation 1: the multicut problem (3.8).
- Formulation 2: MILP of the discrete Potts model (3.7), with and without the 4-edge cycle constraints (3.9).
- Formulation 3: MILP of the discrete Potts model (3.7) with both (3.9) and (3.10).

Parameter setting. There are in total 3 parameters, i.e., the regularization parameter λ in both (3.8) and (3.7), the big M constant in (3.7), and the upper bound on the number of jumps k in (3.10).

We first compute the average intensity of each 4×4 pixels block of the input image, and then calculate the absolute difference of its maximum and minimum value (denoted Y^*). Hence Y^* somehow represents the "global contrast" of the image. We set the λ in (3.7) to $\frac{1}{4}\sigma_1 Y^*$, where σ_1 is a user defined parameter, and let the constant M equals Y^*. When there exists an extreme outlier, model (3.7) tends not to treat the outlier as a single segment, since doing so would incur a penalty as large as 4λ (equals $\sigma_1 Y^*$).

Denote Y_i^r as the vector of pixels' intensities in row i of the image grid, the constant k_i^r in (3.10) of row i is then set to the number of elements in $\nabla^1 Y_i^r$ that are greater than $\sigma_2 Y^*$, where $0 < \sigma_2 < 1$ is some suitably chosen parameter. Constant k_j^c is computed similarly for each column j. Since the input image is supposed to contain noise, in this setting, we take only the edges with weights greater than $\sigma_2 Y^*$ as potential active ones.

Figure 3.3 shows the input images, detailed setting of the parameters, and the segmentation results for the 3 formulations. As we can see from Figure 3.3, Formulation 1 is sensitive to the parameter λ and it is hard to control the desired number of segments. As a result, it is often over segmented and prune to the presence of outliers. On the

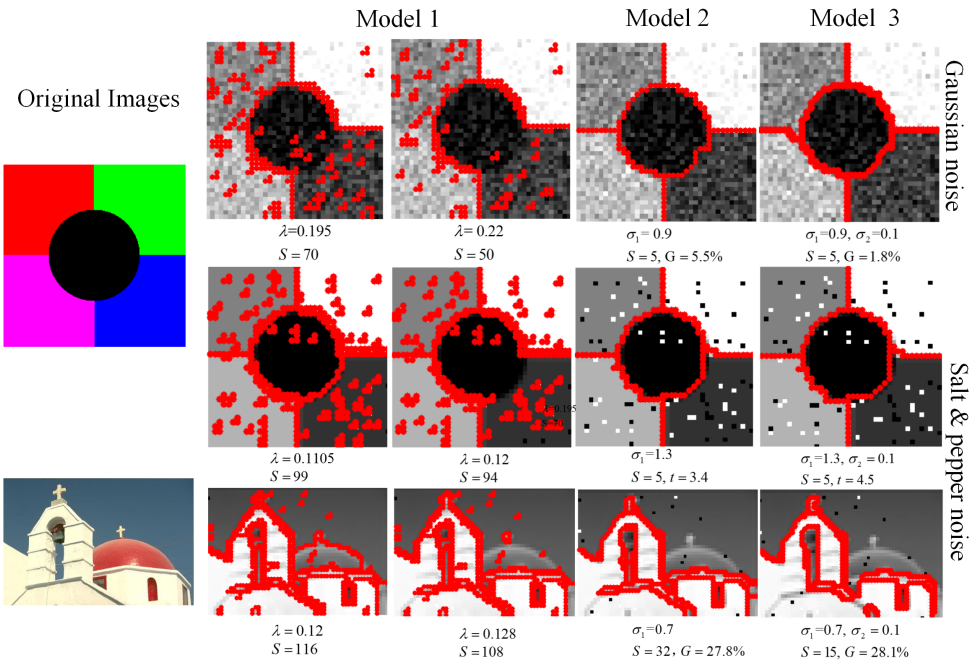


Figure 3.3 – Computational results of three formulations. S: number of segments. t: running time. G: optimality gap when it hits the time limit of 100 sec.

other hand, although requiring more computational time, Formulation 2 and 3 are robust to noise, less sensitive to parameters, and give overall better segmentation results. We report detailed comparison as follows.

With and without (3.9). We first report that Formulation 2 with the 4-edge cycle constraints (3.9) saves 0.9 second in the second instance, and reduces 23.5% of the optimality gap on average (where Cplex hits the time limit in the other two instaces), compared to without (3.9).

In addition to the experiments shown in Figure 3.3, we conduct 10 more experiments with images taken from BSDS500 (resized to 40×50, with 5% salt and pepper noise, time limit 50 seconds). We report that constraints (3.9) help reduce the average optimality gap from 66% to 37%, compared to without them. Hence, we will denote Formulation 2 as (3.7) with (3.9) by default from now on.

Running time and optimality gap. Formulation 1 is very fast to solve, takes less than 0.1 second in all three instances. Formulation 2 and 3 take 2.4 and 4.5 seconds in the second instance and both hit the time limit of 100 seconds in the other two. The optimality gaps for Formulation 2 and 3 on the first and the third instance are 5.5%, 27.8%, 1.8% and 28.1% respectively.

Formulation 2 **versus** 3. We keep the value of σ_1 the same when comparing the effects of adding (3.10) to (3.7). As can be seen from Figure 3.3, there is no clear

	Method 1	Method 2	Method 3
Time	**0.15**	50	50
Gap	Null	23.3%	**19.7%**
Energy	289.0	280.0	**273.8**

Table 3.1 – Average time, optimality gap and energy of 3 proposed methods out of 5 computational tests.

advantage of adding the cardinality constraints (3.10) from the 3 reported instances. For example, it enlarges the optimality gap in the third instance while the solution is visually better (with fewer segments).

Hence, we conclude that it is beneficial to add the 4-edge cycle constraints (3.9), while there is no clear conclusion on whether to add the cardinality constraints (3.10) to (3.7). And we report that when tuning the parameter of σ_2, we found out that by setting σ_2 too large would result in infeasible solutions to (3.7).

3.5.2 Discrete Potts model: heuristic versus MILP

In this section, we will compare the following 3 different methods.

- Method 1: uses only the ℓ_0 region fusion algorithm 3.1 with ℓ_1 data term.
- Method 2: uses only MILP of the discrete Potts model (3.7) with 4-edge cycle constraints (3.9).
- Method 3: similar to Method 2, except we use the result of algorithm 3.1 as initial solution of the MIP solver.

We choose 5 images out of BSDS500, and resize them to 40×50. We add 10% salt and pepper noise, and set $\sigma_1 = 1.8$ for 4 images and 1.5 for one image. A time limit of 50 seconds is set for the MIP solver. For the ℓ_0 region fusion algorithm 3.1, we use the nonlinear growing strategy described in Section 3.4.1, and set the iteration number equals 100 and $\lambda = 0.8$.

The average running time and MIP gaps, as well as the energy function values are reported in Table 3.1. Note that both Method 2 and 3 of the MILP formulation of (3.7) with 4-edge cycle constraints (3.9) are in general very slow to converge. They both reach the time limit of 50 seconds in all 5 instances. We did not report the MIP gap of the heuristic solution, and set it to "Null".

We can conclude from the table that the ℓ_0 heuristic algorithm 3.1 is very efficient. It takes only 0.15 second on average, but provides feasible solutions with not very good quality (equals on average 289.0). While hitting the time limit of 50 seconds, Method 2

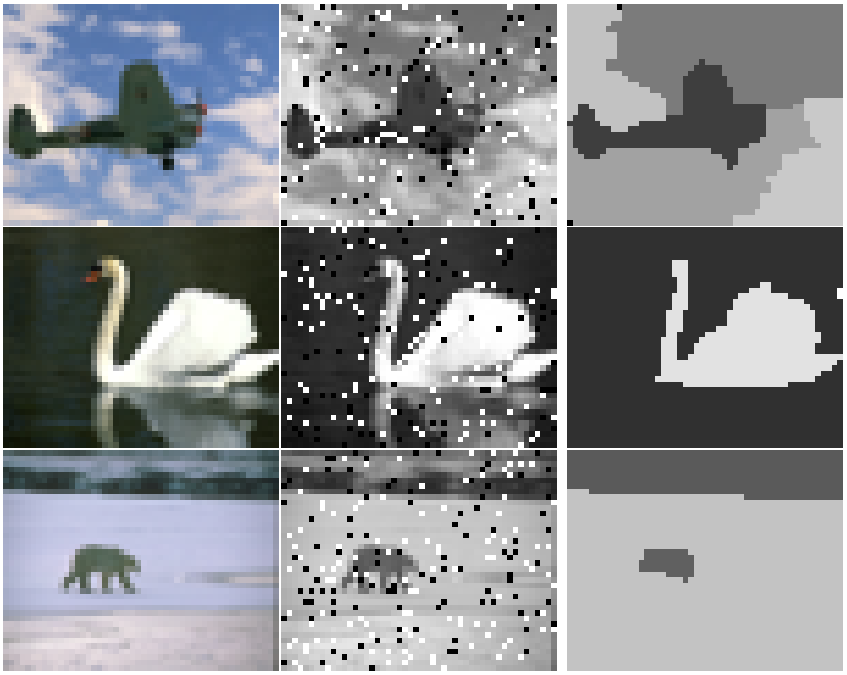


Figure 3.4 – Denoising results of Method 3. From first to third column: the input images of size 40×50, with 10% salt and pepper noise, denoised image using Method 3. The first row uses $\sigma_1 = 1.5$, while $\sigma_1 = 1.8$ for the other two.

provides a better solution than Method 1 (average energy equals 280.0) and has an average optimality gap of 23.3%. Finally, Method 3 adopts the solution of Method 1 as a starting point for the MIP and this increases the overall performance. The average optimality gap reduces from 23.3% to 19.7%, and the average energy further reduces to 273.8, making Method 3 the best over all.

Since from an optimization point of view, the energy (objective function value) is more important than visual appearance of denoising and segmentation. Hence, we only report 3 sample denoising results of Method 3 for illustration purpose in Figure 3.4. The images from left to right columns are original images, images with noise, denoised images using Method 3, respectively. And the first row uses $\sigma_1 = 1.5$, while $\sigma_1 = 1.8$ for the other two. In general, larger σ_1 penalizes more on the number of segments and tends to further smoothen the image. We shall see that in the denoised image of the first row ($\sigma_1 = 1.5$), there exists two one-pixel segments.

We conclude from Figure 3.4 that the MILP formulation of (3.7) with 4-edge cycle constraints (3.9) gives overall satisfying results, which is quite robust to noise.

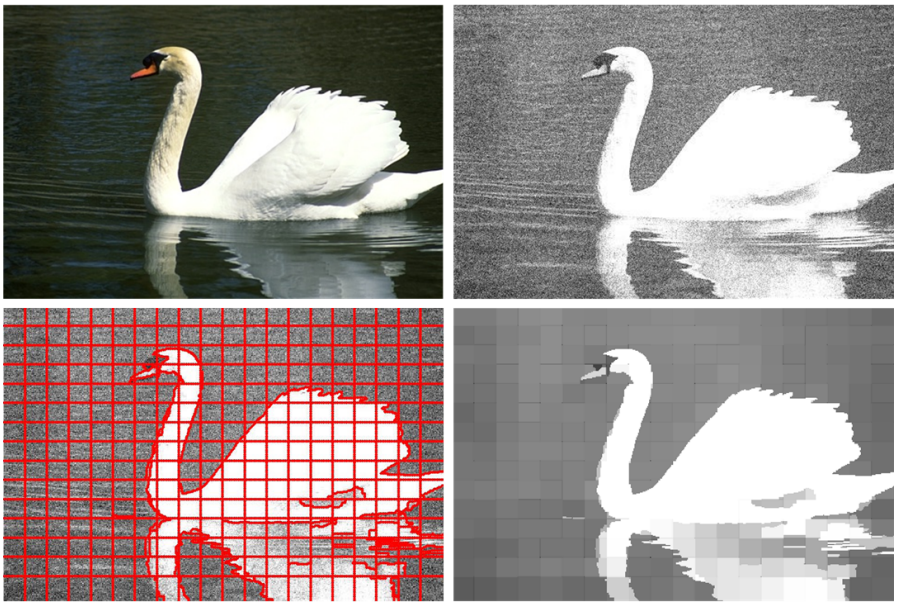


Figure 3.5 – Top left: RGB image. Top right: gray-scale image with 0.5 Gaussian noise. Bottom left: superpixel results of PMcut. Bottom right: denoised output.

3.6 Application on Generating Superpixels

In this section, we first describe our superpixel algorithm (denoted *PMcut*) that is based on the Discrete Potts model (3.7) with the 4-edge cycle constraints (3.9). Then we present a detailed evaluation of our superpixel algorithm towards 4 other state-of-the-art algorithms on noisy images, both quantitatively and qualitatively. The tested superpixel sizes (approximately) are 600, 1200, 1800 and 2400, respectively.

The Berkeley Segmentation Dataset (BSDS500) [20] consists of 500 images (size 321×481) splitting into 200 training, 100 validation and 200 test images. We conduct extensive parameter training before testing on the competing algorithms. For testing, we choose 100 test images to add 0.3 Gaussian noise, and the rest 100 to add 15% salt and pepper ($S\&P$) noise. The test sets are thus divided according to these two types of noises. The superpixel segmentation results are compared with the provided five ground-truth segmentations per image of the BSDS500.

While ours did not train the parameter σ (we set it to 0.5), and the inputs are gray-scaled images (others uses RGB images), we show that it still achieves the best results on noisy images in terms of a combined score (OP) against other algorithms with optimized parameters.

3.6.1 Our superpixel algorithm

The superpixel algorithm is quite straightforward. Like in [46] we start from decomposing the input image into $K \cdot L$ rectangular patches, where K and L are user defined parameters approximating the number of desired superpixels. We then apply PMcut within each image patch. Different from [46], the size of our superpixel is upper bounded by the size of the image patch. See Figure 3.5 as an example of applying PMcut to generate superpixels on a noisy image. It meanwhile denoises the image (shown at the bottom right of the figure).

Since PMcut is $\mathcal{NP}$-hard, smaller image patches (hence smaller MILP problem size) are desired for the sake of computational efficiency. In practice, we notice that with the increase of the number of image patches, the overall computational time decreases since the patch size shrinks.

Note also that every image patch is computed separately, thus parallelism can be fully adopted. In practice, one can also set a time limit or a MIP optimality gap threshold within each patch. As a matter of fact, we often notice the segmentation results are already quite good even though PMcut has not arrived at the optimal solution. For example, in the second experiment of Figure 3.3, PMcut find the output solution in less than 2 secs, but the gap is still more than 1% when it hits the 100 seconds time limit. Hence it also makes sense to create a stopping criterion when the solution is already good enough based on a user defined optimality gap.

3.6.2 Competing superpixel algorithms

We review some state-of-the-art superpixel algorithms, which are either highly ranked in terms of UE and Rec scores, or give regular shapes (high CO score), according to a recent survey paper [16].

ETPS – Extended Topology Preserving Segmentation [47] partition the image into a regular grid as initial superpixel segmentation, and then exchanges pixels between neighboring superpixels iteratively. It uses a coarse-to-fine energy update strategy, and uses block coordinate gradient decent to minimize the energy. However, it produces superpixels with irregular sizes and shapes.

SLIC – Simple Linear Iterative Clustering [15] uses k-means clustering algorithm. It initializes by seeding pixels as cluster centers, and uses color and spatial information for updating. Post-processing is needed to ensure connectivity. This approach offers control over the number of superpixels and the compactness.

CW – Compact Watershed [48] is based on SLIC and [49], and contains two modified algorithms. The first one speeds up SLIC, while the second creates uniformly shaped superpixels as opposed to [49]. It is computational very efficient, takes only 10-33 milliseconds (ms) for segmentation an image of size 321×481.

SEEDS – Superpixels Extracted via Energy-Driven Sampling [46] proposed an approach based on a simple hill-climbing optimization. It starts from an initial superpixel partitioning (e.g., rectangular patches), then continuously refines the superpixels by modifying the boundaries. The energy function is based on enforcing color similarity between the boundaries and the superpixel color histogram. It can control the number of superpixels but not the compactness.

While these superpixel algorithms may work well on clean images, they do not have a systematical way of dealing with noisy images and may suffer from the presence of noise. Moreover, to the best of our knowledge, there have not been any comprehensive comparisons on superpixels algorithms applied on noisy images.

3.6.3 Evaluation benchmark

Among other metrics, the Under-segmentation Error (UE), Boundary Recall (Rec) and Compactness (CO) are probably the most widely used ones. They are standard measures for segmentation boundary adherence.

Given a superpixel segmentation and a ground-truth, under-segmentation error measures the fraction of superpixels that "leak" across the boundary of a ground-truth segment. We adopt the updated formulation of UE (2.6) in [16], that does not over-penalize superpixels which only overlap slightly with ground-truth segment. Overall, under any formulation of UE, the lower score is better.

Boundary Recall (2.7) measures what fraction of the ground-truth edges fall within a local neighborhood of size $t \cdot t$ of a superpixel boundary. Like in [16], here we set $t = 3$. As superpixels are expected to adhere to boundaries, high Rec score is desirable.

The Compactness Score (2.8) compares the area $A(s_i)$ of each superpixel s_i with the area of a circle with the same perimeter $P(s_i)$. The higher CO score means the superpixels are more regular, and hence more desirable.

While UE and Rec depends on the ground-truth segmentation, CO is invariant to it. We again refer readers to [16] for more detailed information on UE, Rec and CO scores.

3.6.4 Parameter optimization and post-processing

Before testing, we optimized parameters of the competing algorithms on 100 training images where half are added with Gaussian noise and another half with $S\&P$ noise. The noise level is chosen to be the same as the testing images. We conduct discrete grid search, jointly optimizing UE, Rec and CO, i.e., OP = $0.4*$(1-UE) + $0.4*$ Rec + $0.2*$CO. For implementation details, readers can refer to [16].

Since some superpixel algorithms do not ensure connectivity, a connected component algorithm is proposed and used as post-processing by [16], which results in many tiny superpixels. Hence [16] additional propose a merging algorithm, which is designed to merge tiny superpixels into larger neighboring ones. We set the merging threshold

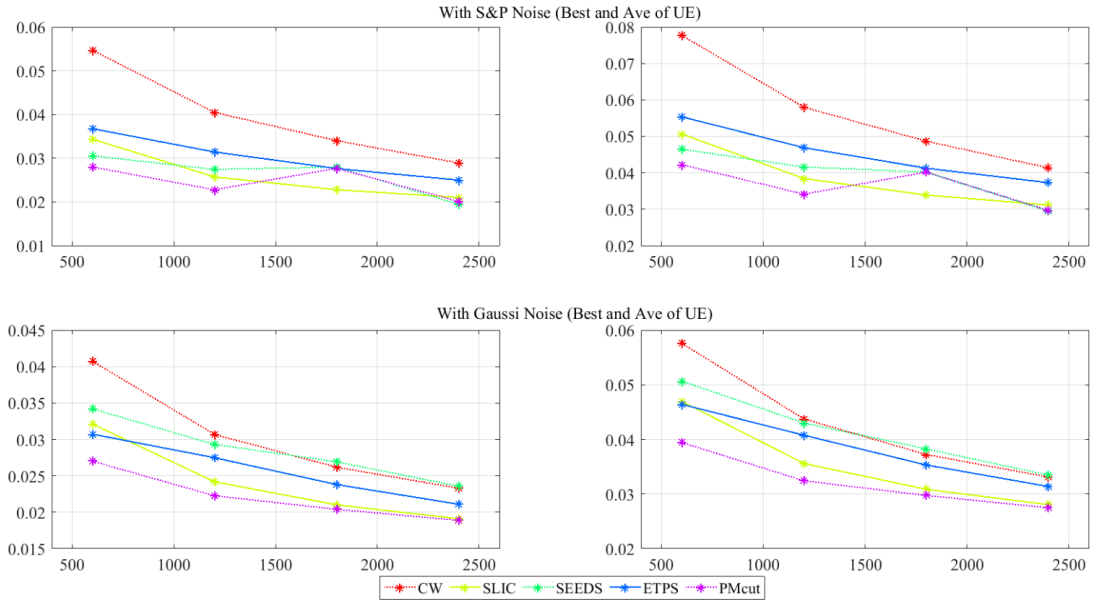


Figure 3.6 – Undersegmentation error of each method (lower is better). Horizontal axis: number of superpixels. Vertical axis axis: UE score.

(minimum number of pixels in any superpixel) to be 10. We adopt the code from [16] for both parameter optimization and post-processing.

3.6.5 Quantitative comparison

For computing the Rec and UE socre, the superpixel results are compared with the provided five ground-truth segmentations per test image, and we choose the best and average out of 5. We then compute the average score of all test images. Since CO is independent of ground-truth, we just compute the average CO score of all test images.

The under-segmentation error (UE)

Figure 3.6 plots the UE score (vertical axis) of each method against the increasing number of superpixels (horizontal axis). The first row depicts results on the 100 images taken from BSDS500 with $S\&P$ noise, and the second row another 100 images with Gaussian noise. These results were obtained by averaging the 100 images of the maximum and average score out of the 5 ground-truth segmentations. PMcut is the clear winner on UE score, especially on images with Gaussian noise. SLIC comes the next place, which even beats PMcut on images with $S\&P$ noise on 1800 superpixels. Meanwhile, CW is the clear loser, which achieves the worst scores in almost all scenarios.

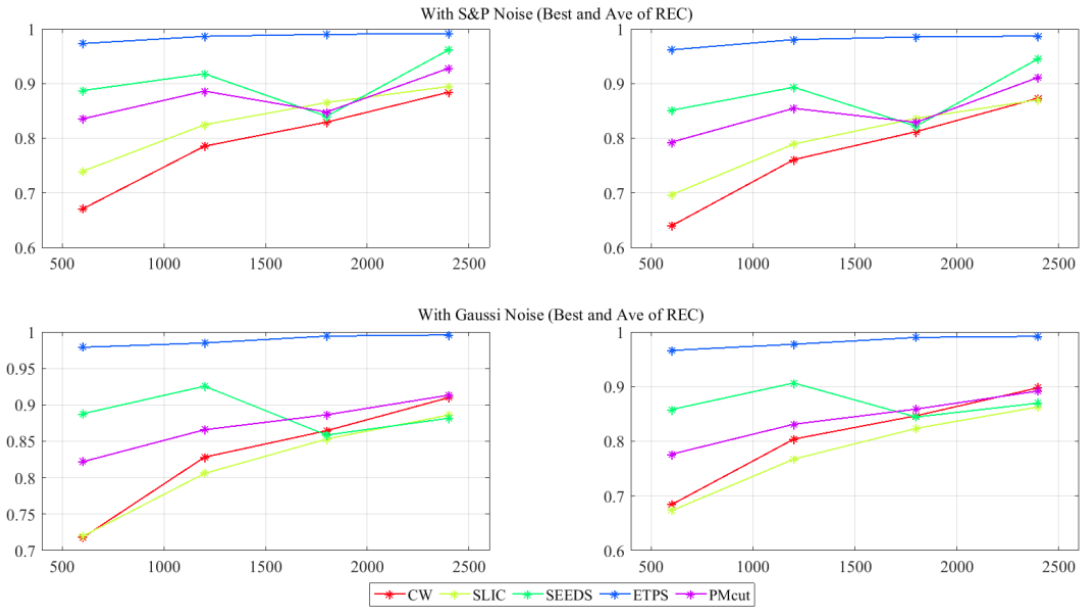


Figure 3.7 – Boundary recall of each method. Horizontal axis: number of superpixels. Vertical axis axis: Rec score.

Finally, there is a clear tread that with increasing number of superpixels, UE score tend to get better (lower).

The boundary recall (Rec)

The boundary recall of each method is plotted similarly in Figure 3.7. Superpixels generated by ETPS demonstrated the overall best boundary recall performance, and SEEDS comes the second place. PMcut also performs quite well, rank 2nd place in 6 scenarios and 3rd in rest. CW performs worst on images with $S\&P$ noise in all scenarios, while SLIC becomes worst on images with $S\&P$ noise in almost all scenarios.

Finally, there is also a tread that with increasing number of superpixels, REC score gets better, with the exception on SEEDS and PMcut in certain cases.

The compactness score (CO)

Furthermore, we evaluate the compactness of superpixels. The comparison result is illustrated in Figure 3.8.

In comparison with other algorithms, the CO score computed by our proposed method (PMcut) show a significant advantage over all competing algorithms, with only one scenario out of eight beaten by SEEDS. On the other hand, although achieving the bset result in terms of Rec, ETPS has the worst CO score amongst all. We will see in

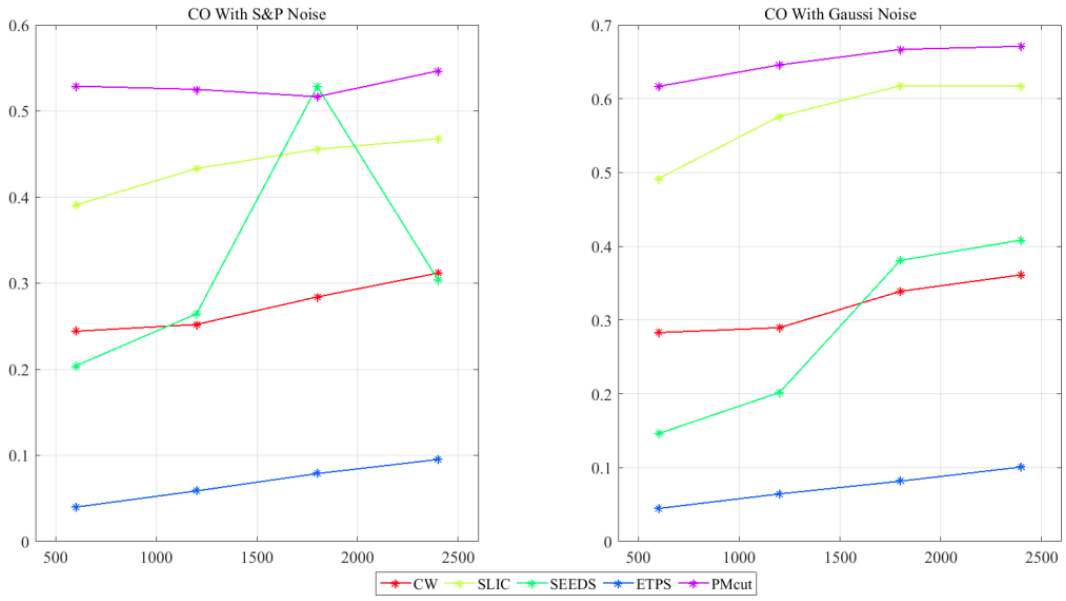


Figure 3.8 – Compactness score of each method. Horizontal axis: number of superpixels. Vertical axis axis: CO score.

Section 3.6.6 that the superpixels generated by PMcut are very regular and compact, while those by ETPS are irregular.

The joint score (OP)

Since all the competing algorithms are trained according to OP, it is thus the most important score in this section. We report that PMcut achieves the 1st place in all 24 scenarios (according to number of superpixels, different noise types and ways to compute scores).

Table 3.2 depicts one of 24 scenarios of the OP scores of 5 superpixel algorithms. In this case, the test images contains Gaussian noise and the methods produce approximately 600 superpixels. We first compute the average score out of 5 ground-truth segmentation and then average all 100 tested images.

CW	SLIC	SEEDS	ETPS	PMcut
0.71	0.76	0.76	0.77	0.83

Table 3.2 – The combined OP score of 5 superpixel methods in one scenario.

Table 3.3 shows the average OP score amongst all 24 scenarios. Recall that OP is computed as $0.4*$(1-UE) + $0.4*$ Rec + $0.2*$CO.

CW	SLIC	SEEDS	ETPS	PMcut
0.75	0.80	0.79	0.79	0.84

Table 3.3 – Average OP score of 5 superpixel methods.

Computational time

Another important aspect of superpixel algorithms is computational time. While all the other superpixel algorithms take less than 1 sec, the running time of PMcut is doomed by the $\mathcal{NP}$-hard nature of the MILP formulation. Among those, CW and SEEDS are the fastest, on average takes only 0.01 second and 0.05 second.

The running time of PMcut is, on average, as follows. The time limit for each rectangular patch is 8 secs, 2 seconds, 0.5 second, and 0.2 second, for the desired superpixel size (also the number of MILPs to solve) of 600, 1200, 1800 and 2400 respectively. The corresponding total running time for the whole image is 272, 134, 40 and 27 seconds. Finally, the average Cplex optimality gap is 12.2%, 9.8%, 10.3% and 9.8%, respectively.

One can notice that as the number of rectangular patches increases, the total time as well as the average Cplex gap decreases. In practice, one can also set different time limit and MILP gap threshold, so that Cplex will terminate earlier to achieve a better running time performance. In addition, parallelism can be fully adopted to speed up computation.

3.6.6 Qualitative comparison

Visual quality can be determined by considering compactness, regularity and smoothness. Here, compactness refers to the area covered by every superpixel, regularity corresponds to both the size and the arrangement of each superpixel, and smoothness refers to the superpixel's boundary.

Figure 3.9 shows superpixel results on three images (with both noises) using all superpixel algorithms, with approximate superpixel number equals 600 and 1200. Note that these two images are intended to be representative, however, keep in mind that superpixel segmentations may vary across different images.

Most of the algorithms provide solid adherence to salient image boundaries, especially when the number of superpixels is large. Compactness and smoothness varies a lot across algorithms and a compactness parameter is beneficial to control the degree of compactness which allows to trade boundary adherence for compactness. Ideally, compactness (CO) should be increased while only slightly sacrificing boundary recall (Rec).

At a first glance, the superpixels generated by PMcut and SLIC are visually more appealing than those of ETPS, SEEDS and CW. Superpixels of PMcut are initialized and bounded by each rectangular image patch, since most of them consist only of back-

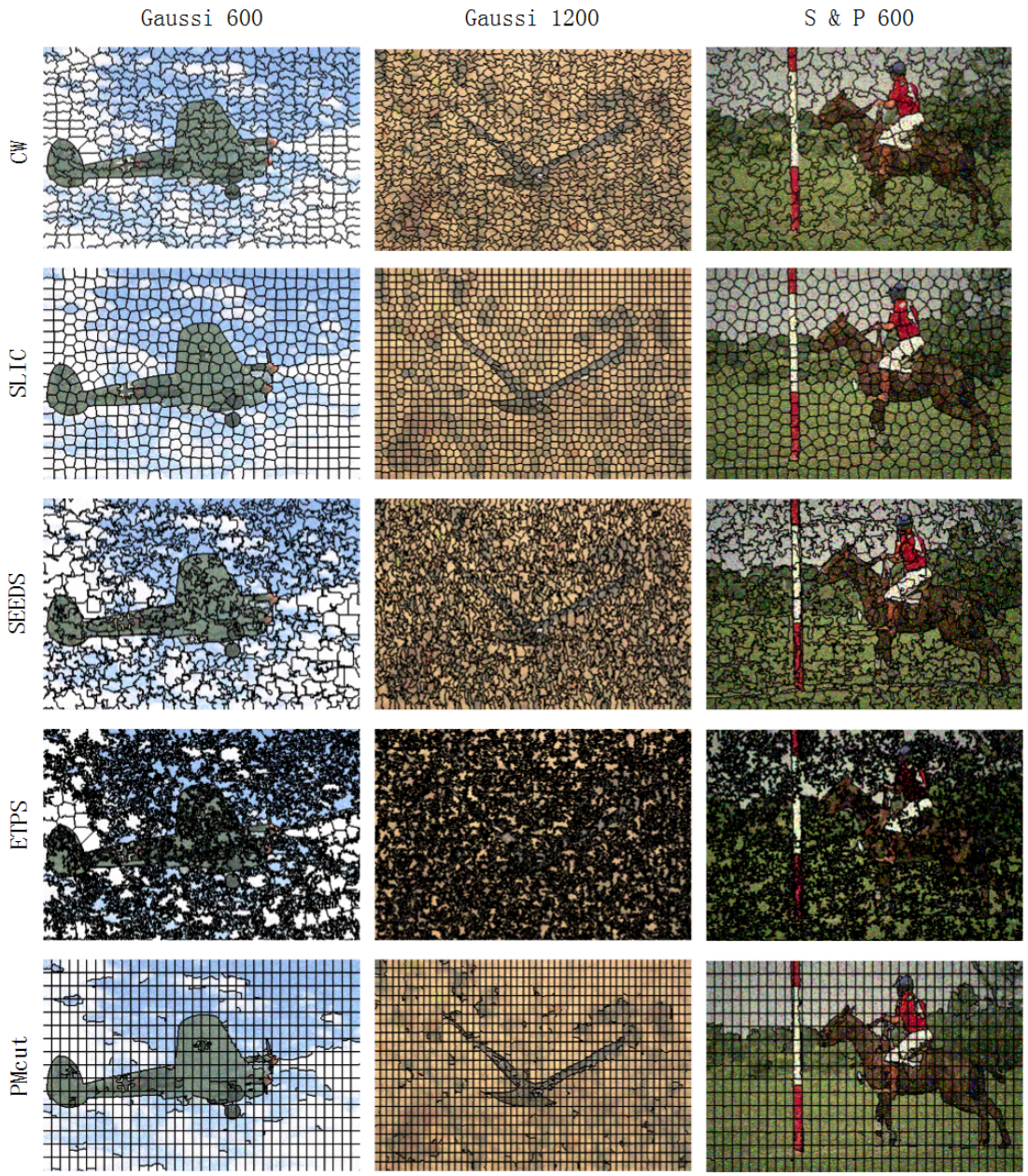


Figure 3.9 – Superpixel results of 5 methods on images with Gaussian and $S\&P$ noise.

ground, the resulting superpixels become the rectangular patches themselves. Hence most of the superpixels are compact and regular in size. PMcut also benefits from the penalty term λ on the boundary length, thus creating smooth superpixels. Although SLIC is also compact and regular in size, it sometimes loses track of some boundary

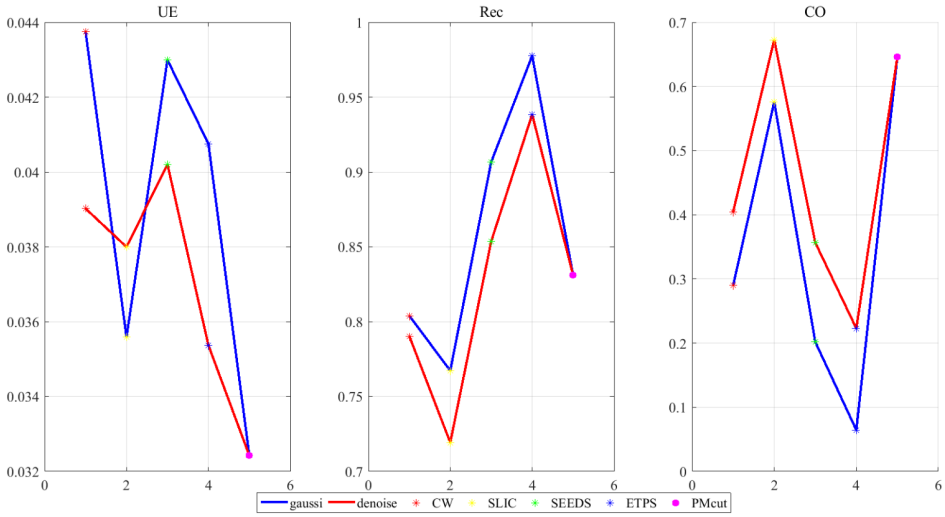


Figure 3.10 – Comparing superpixel methods applied on both Gaussian and denoised images (PMcut only on noisy images), number of desired superpixels: 1200

details, e.g., the helmet in the third image.

In contrast, although being able to capture most of the object boundaries (with high Rec), ETPS and SEEDS produce highly irregular and non-smooth superpixels. Their superpixel size varies a lot, with many tiny superpixels and yet huge ones. One can hardly recognize the original image if the number of superpixel is large, e.g., ETPS and SEEDS segmentation results in the second image.

In conclusion, we find that the evaluated ETPS and SEEDS algorithms show inferior visual quality. On the other hand, PMcut and SLIC demonstrate good superpixel visual quality at the expense of missing some object boundaries.

3.6.7 Competing algorithms applied on denoised images

We conduct additional experiments by first applying a denoising algorithm [50] (we set the two parameters $\sigma_s = 5$ and $h = 0.55\sigma_n$, where σ_n is the noise variance of n pixels.) to denoise the 100 BSDS500 images with Gaussian noise. The competing superpixel algorithms are then run on the 100 denoised images, then we compare their segmentation results against the results obtained by PMcut directly applied to the noisy images. The comparison is shown in Figure 3.10, where the approximate number of superpixels equals 1200 and the scores are computed according to the best of 5 ground-truth segmentations. The blue line denotes the results on images with Gaussian noise, while the red one on noisy images.

As one can probably notice, compared with the results on noisy images, almost all the competing algorithms perform better on the UE and CO scores on denoised images.

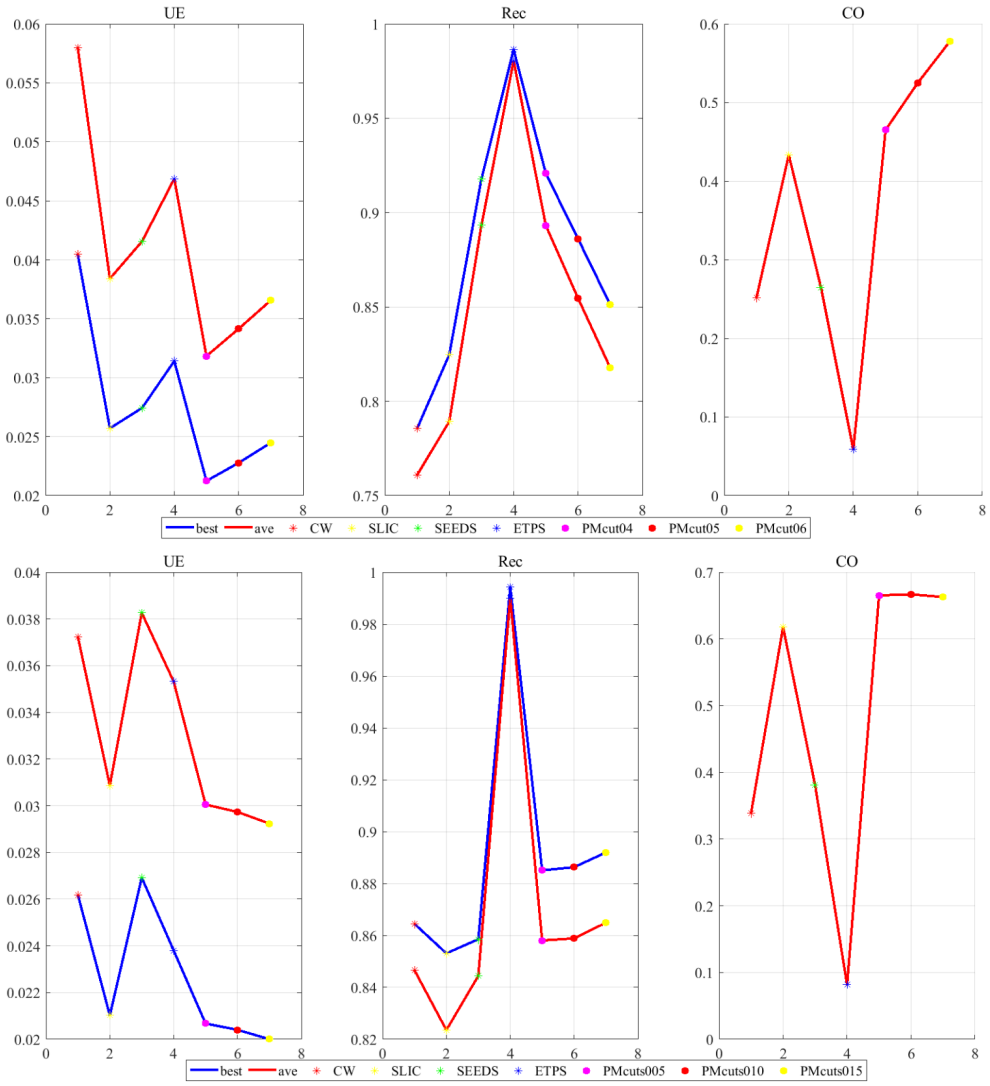


Figure 3.11 – Top: PMcut on $S\&P$ images with different σ, $\# = 1200$. Bottom: PMcut on Gaussian images with different time limit, $\# = 1800$.

However, this does not hold on the Rec score, and it actually becomes worse for all competing algorithms. We think it may be because of the fact that the denoising algorithm smooths the boundary too much. The average running time for the sophisticated denoising algorithm is over 60 secs per image, hence our running time is superior in this case.

3.6.8 Parameter tuning on PMcut

Finally, we conduct experiments of using PMcut on adopting different values for the regularization parameter σ (0.4, 0.5 and 0.6) and time limit (0.05, 0.1 and 0.15 second) within each rectangular patch of the input image.

Figure. 5.3 shows the results of the two experiments. The first one with the number of approximate superpixels equals 1200 and the second one with 1800. Judging from the plots, it is obvious that when the regularization parameter σ grows, the Rec and UE scores of PMcut become worse while CO better. Also, when the time limit increases, almost all three scores benefit,. Thus, it is a trade-off between running time and benchmark scores. We also report that the average MILP gap when adopting the reported 3 time limits are 12.8%, 10.3% and 9.5%, respectively.

3.7 Conclusions and Future Work

We have presented a combined unsupervised image segmentation and denoising framework that is basically the first derivative Potts model and based on the piecewise constant assumption. We formulate it into a MILP and add the 4-edge cycle multicut constraints to cut off feasible but non-optimal solutions. We then apply it for superpixel generation, where each rectangular patch of the original image needs to solve a MILP. We conduct extensive experiments on BSDS500 image dataset with noises. Even without parameter training, our method achieves the best performance against 4 other state-of-the-art superpixel algorithms with optimized parameters on the combined OP score.

Since image noise is unavoidable and there has been so far little work that addresses superpixel algorithms applied on noisy images, we believe this is a research direction that deserves the community's attention.

In the future, better strategies of selecting cuts of PMcut should be investigated. On the other hand, since our MILP formulation of the piecewise constant Potts model is guaranteed to get the optimal solution (if no time limit is specified), it is also interesting to apply it to generate ground-truth results, which could be served as benchmarks for any fast approximate algorithms.

Chapter 4

Piecewise affine Potts Model for Segmentation and Denoising

In this chapter, we assume that the input signals or images contain noise and possess linear trends. The task of segmentation and denoising is then transformed into a piecewise linear fitting problem in 2D and 3D.

The Potts model is a classical segmentation and denoising model. However it is based on the piecewise constant approximation. Many types of images such as disparity maps (inverse of depth images) have linear trends so that piecewise affine models are more suitable. Also, robustness to outliers is often desirable which suggests the use of the ℓ_1 data terms.

In this chapter, we propose a novel solver for the piecewise affine Potts model with the ℓ_1 data fidelity. The key idea is to formulate the problem as a mixed integer program (MIP) which uses edges as binary variables. To obtain consistent partitions (segmentation), the inclusion of the multicut constraints is proposed. Since the resulting problem is $\mathcal{NP}$-hard to solve, we propose a fast region fusion based heuristic that could provide initial solutions for the MIP solver. An advantage of the approach is that it can be used on top of any heuristic algorithms and provide an optimality gap for the quality control. We conduct experiments by some synthetic images as well as some real world disparity maps. We also apply it for over-segmentation, i.e., generating superpixels.

4.1 Overview

Let $D \in \mathbb{R}^d$ denote the signal domain, and $y : D \to \mathbb{R}$ denote the given signals' intensity values possibly with noise. We are most interested in the case when $d = 2$, and y corresponds to natural or depth images.

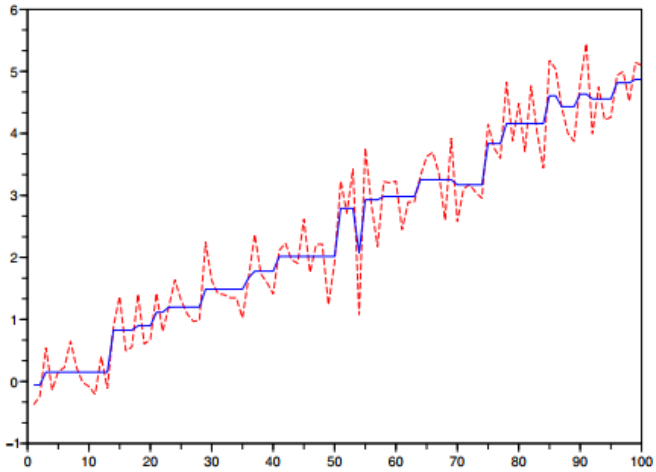


Figure 4.1 – Fitting a noisy ramp using model (4.1), where the stair-casing effect is present. Image taken from [51].

For denoising input signals, we recall the total variation (TV) model [29]

$$\min_{w} \int_D (w(z) - y(z))^2 dz + \lambda \mathrm{TV}(w) \tag{4.1}$$

where the first part is the data fitting term, and second part the regularization term. Here, z represents the coordinates of the input signals (z_1 and z_2-axis if the input is an 2D image), and $\lambda > 0$ the user defined regularization term. The TV regularizers can be either isotropic or anisotropic, they are mathematically given by

$$\mathrm{TV}_{iso}(w) = \int_D \sqrt{|\partial_{z_1} w|^2 + |\partial_{z_2} w|^2}\, dz,$$

and

$$\mathrm{TV}_{ani}(w) = \int_D \left(|\partial_{z_1} w| + |\partial_{z_2} w|\right) dz,$$

where ∂_{z_i} represents the partial derivative of w with respect to z_i, for $i = 1, 2$.

Being a first order model, TV model (4.1) has some shortcomings, most notably the starircasing phenomenon when the input signals contain linear trend (see Figure 4.1) or the true (depth) image contains slanted regions [51, 52]. In this case, the gradient discontinuities of the data cannot be extracted.

Lysaker and Tai [53] provide two second-order regularizers

$$R_1(w) = \int_D \sqrt{w_{z_1 z_1}^2 + w_{z_1 z_2}^2 + w_{z_2 z_1}^2 + w_{z_2 z_2}^2}\, dz,$$

and

$$R_2(w) = \int_D (|w_{z_1 z_1}| + |w_{z_2 z_2}|)\, dz.$$

We will adopt a variant of the discrete version of R_2 in this chapter.

We call a function f piecewise affine over D if there is a partition of D into subsets $D_1, \ldots, D_k$ where $D_i \cap D_j = \emptyset$ and $D = \cup_{i=1}^k D_i$, and furthermore, f is affine when restricted to D_i (denote the function restricted to D_i as f^i). Let $\mathcal{D}$ be the set of all partitions of D, and $\mathcal{F}$ be the set of all piecewise affine functions over D, then any choice of $f \in \mathcal{F}$ pre-defines $\mathcal{D}' \in \mathcal{D}$. Moreover, if the partition $\mathcal{D}'$ is known, the corresponding $f \in \mathcal{F}$ can be easily calculated by conducting an affine regression within each region D_i, for any $i \in [k]$.

In this chapter, we are interested the problem of estimating a piecewise affine function $f \in \mathcal{F}$. In contrast to model (3.2), we consider a piecewise affine Potts model, which we first present the following geometry form of the problem:

$$\min_{f \in \mathcal{F}, \mathcal{D}' \in \mathcal{D}} \sum_{D_i \subset \mathcal{D}'} \int_{D_i} |(f^i(z) - y(z)| dz + \lambda Per(D_i). \tag{4.2}$$

Here, we adopt the ℓ_1 (absolute) data fidelity term and $Per(D_i)$ denotes the perimeter of D_i.

Apart from denoising, we also look into the segmentation problem. The problem is often formulated as an energy minimization problem, which is often derived in the context of Markov Random Fields [54]. The minimum of the energy corresponds to a *maximum a posteriori* (MAP) labeling of each pixel. We focus on decomposing an image into a previously unknown number of segments which do not belong to a predefined set of categories, i.e., the unsupervised image segmentation, where no prototypical features of the image are available.

Recent notable approaches are the (lifted) multicut problems [3, 24, 55] based on the ILP formulations, which label edges instead of pixels and have exponentially many constraints. It is solved by a commercial ILP solver using the cutting plane approaches [3]. The multicut approach is appropriate when one assumes that the input signals y is approximately piecewise constant or differs only a little within a segment. Most of the literature deals with this problem type [3, 23, 39, 55, 56], including the Mumford-Shah model [57].

In this work, we propose a novel MILP formulat for the piecewise affine Potts model with the ℓ_1 data fidelity term (4.2). We will show later that the discrete version of (4.2) alone does not necessarily impose a unique or even valid segmentation in our setting. The inclusion of the multicut constraints is then necessary to obtain consistent partitions.

Figure 4.2 shows a synthetic image with noise that has linear trends and its 3D view, where the horizontal axes (z_1 and z_2) represent the coordinates of the image pixels. The desired outputs are first a denoised image, and second, a segmentation of the image into background and four segments.

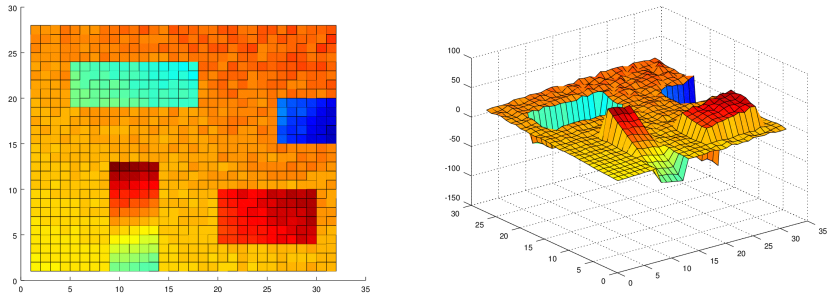


Figure 4.2 – A synthetic 2D image with noise that has linear trend and its 3D view.

4.1.1 Related work

The piecewise affine or second order TV model is not new. Back in [58], Blake and Zisserman proposed a second-order variational model. The work of [59] studies a TV based piecewise affine regularization problem, and they propose a standard convex programming method by employing the augmented Lagrangian and primal and dual variables. Recently, [60] generalized the result of [58] to an arbitrary higher order case.

Since the ℓ_1 regularization term over-penalizes the sharp discontinuities between two regions in an image, Potts model [1] is usually more desirable, but also computationally costly. The discrete Potts is in general $\mathcal{NP}$-hard to solve. [54] was one of the first works to utilize the Potts model with ℓ_2 data fidelity. Recently, [31] adopted the alternating direction method of multipliers (ADMM) method to efficiently solves the Potts model with ℓ_2 data term. In contrast, Potts estimators with ℓ_1 data terms has drawn significantly less attention mainly because of its non-differentiability and hence computationally harder. The authors of [35] propose efficient algorithms (based on the Viterbi algorithm that exploits the special structure of the Potts penalty [61]) that compute Potts estimators of ℓ_1 data terms for real-valued scalar as well as for circle-valued data.

In applications with regard to depth image, stereo matching and motion estimation, the input data usually comprises of linear trend, hence piecewise affine models are more suitable. [62, 63] have incorporated depth information on top of the RGB image, which largely improves RGB-D image segmentation accuracy when occlusion is present by modeling the depth image of slanted planars as linear functions. The Blake-Zisserman model is applied in [64] to a digital surface segmentation, and iterated methods based on simple thresholding are used. The work of [52] approximates nearest neighbor of each pixel according to an affine plane in 3D, it achieved top performance among all the local methods in stereo matching with slanted support windows. Recently, [65] proposes a new method to estimate piecewise affine motion fields directly without intermediate segmentation.

Finally, the problem of piecewise affine regression is also studied in the Operations Research community. Continuous piecewise linear fitting models are studied in [66], where the domain partition is in a sense pre-defined, and the fitting function f is restricted to be continuous over the signal domain D. Both parametric (2.4) and nonparametric (2.5) models are studied. A more general n-dimensional data sets with piecewise linear features have been studied in [67], and formulated using MILP as a parametric model. But the assumption that the segments are linearly separable from each other does not hold in many practical applications.

It is worthwhile to highlight some contributions of our method:

- A novel MILP formulation is proposed for the discontinuous piecewise affine Potts model, which aims for global optimum.
- It adopts the ℓ_1 norm of the data fidelity term of the Potts model, which is more robust to outliers than the ℓ_2 norm.
- We add multicut constraints to achieve a valid (depth) image segmentation.
- We propose an approximate and non-parametric model for solving the piecewise linear regression problem.

4.2 MIP of the piecewise affine Potts model: 1D

We first restrict ourselves to the simple 1D signals case, and our goal is find a piecewise linear function f (possibly discontinuous over D) that best fits the original data y. We prefer to work on the discrete space where the signals lie in $[n]$, and similar to Section 3.2, our MIP formulation is also based on graph $G(V, E)$.

4.2.1 Modeling as a MIP

The associated graph $G(V, E)$ for the 1D signals is a chain, and we denote $V = \{i \mid i \in [n]\}$, $E = \{e_i = (i, i+1) \mid i \in [n-1]\}$. We again introduce $n-1$ binary variables defined on edges to indicate if an edge is active or not.

$$x_{e_i} = \begin{cases} 1, & \text{if nodes } i \text{ and } i+1 \text{ are in different segments,} \\ 0, & \text{otherwise.} \end{cases}$$

Recall that an edge e is an *active* or *jump edge* if $x_e = 1$, otherwise it is *dormant*. In this section, we will denote the binary variable x_{e_i} as x_i for simplicity.

Our goal is to fit a piecewise linear function $f \in \mathcal{F}$ to the input data y, and we denote the fitting value $w_i := f(z_i)$, for $i \in [n]$. Similar but different to the piecewise

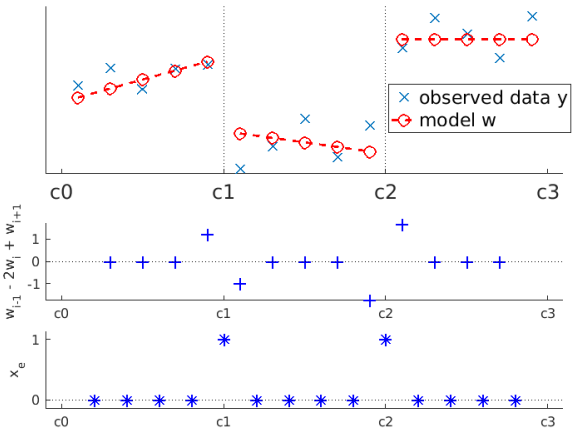


Figure 4.3 – An example with 3 linear pieces and 2 jump edges (blue crosses for y, red circles for w, blue pluses for $\nabla^2 w_i$ and blue stars for x_e).

constant Potts model in Section 3.2.1, we need the following implications:

$$\begin{aligned} \nabla^2 w_i = 0 &\Leftrightarrow x_{i-1} = x_i = 0, \quad i \in [2 : n-1], \\ \nabla^2 w_i \neq 0 &\Leftrightarrow x_{i-1} + x_i \geq 1, \quad i \in [2 : n-1], \end{aligned} \tag{4.3}$$

where $\nabla^2 w_i := w_{i-1} - 2w_i + w_{i+1}$, and $[2 : n-1]$ denotes the discrete set $\{2, 3, \ldots, n-1\}$.

The above implications can be modeled via MIP using the "big M" technique, which leads to the formulation

$$\min \quad \sum_{i=1}^{n} |w_i - y_i| + \lambda \sum_{i=1}^{n-1} x_i \tag{4.4}$$

$$|\nabla^2 w_i| \leq M(x_{i-1} + x_i), \quad i \in [2 : n-1], \tag{4.4a}$$

$$w_i \in \mathbb{R}, \quad i \in [n], \tag{4.4b}$$

$$x_i \in \{0, 1\}, \quad i \in [n-1], \tag{4.4c}$$

where $\lambda > 0$ is the regularization term for the Potts model and again serves as penalty for the number of jumps.

Following the same trick in Section 3.2.1, by introducing two nonnegative variables ε_i^+ and ε_i^- to represent $|w_i - y_i|$, one can formulate (4.4) as a MILP. As a result of (4.4), we get w_i as the fitting value for signal i, and $x_i = 1$ as the boundary edges between two partitions. The pixels between two active edges define one segment, and the signals within one segment share the same linear slope. Although being nonparametric, the linear parameters β for each partition can be easily computed afterwards using w and their coordinates z. The number of segments equals $\sum_{i=1}^{n-1} x_i + 1$.

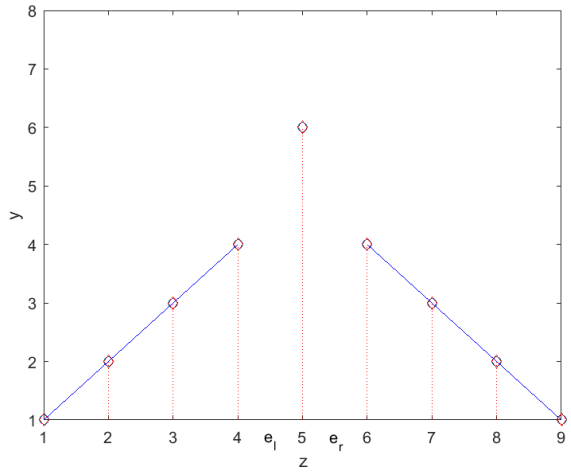


Figure 4.4 – An example where outlier exists, i.e., both e_l and e_r are active.

Lemma 4.1. *The optimal solution $x^\star$ of problem* (4.4) *satisfies properties 4.3.*

Proof. The proof is similar to the proof of Lemma 3.1, and we skip it here. □

Two obvious implications of Lemma (4.1) are: if $\nabla^2 w_i \neq 0$, then $x_{i-1}+x_i$ must be at least one. On the other hand, if $\nabla^2 w_i = 0$, the optimal binary variables x satisfy $x_{i-1} + x_i = 0$ since (4.4) is a minimization problem with positive weights on x. Hence, the signals within the same segment share the same linear function, and different segments are separated by active edges. We can conclude that (4.4) indeed corresponds to a piecewise linear fitting model.

The image on the top of Figure 4.3 shows an example with 3 partitions and 2 active edges. The second row shows the value of $\nabla^2 w_i$, where there are 4 non-zeros, and the third row the value of binary edge variables x_i. Note in this example, the cases where $\nabla^2 w_i \neq 0$ actually induces optimal binary variables x satisfying $x_{i-1} + x_i = 1$, for some $i \in [2, n-1]$. This is again because of the nature of the minimization problem. However, there exists instances where $x_{i-1} + x_i = 2$ for $\nabla^2 w_i \neq 0$. See Figure 4.4 for an example, where the node at position 5 is an outlier (an one signal segment), and $x_{e_l} + x_{e_r} = 2$.

Finally, we observe that problem (4.4) does not necessarily have unique optimal integer solution x. One example is shown in Figure 4.5, where either x_{e_l} or x_{e_r} can be active (but not both), and they yield the same optimal objective value. In this case, the piecewise linear function f is continuous, and the node at position 2 can be either in the first or second partition.

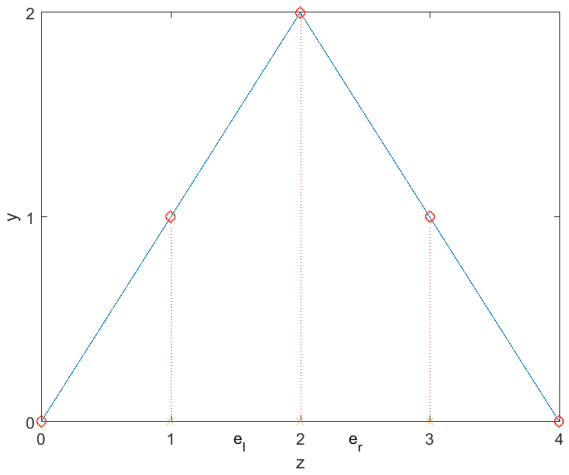


Figure 4.5 – An example with two segments where the optimal solution is not unique, i.e., either e_l or e_r can be active, but not both.

4.3 MIP of the Piecewise Affine Potts Model: 2D

Given an $m \times n$-grid of pixels with coordinates $z_{i,j} = (i, j) \in \mathbb{Z}^2$ for pixel (i, j) and the matrix $Y = (y_{i,j}) \in \mathbb{R}^{m \times n}$ of intensity values (i.e., depth information) of the image, the piecewise affine Potts model is to find a fitting matrix $W = (w_{i,j}) \in \mathbb{R}^{m \times n}$, that approximates a piecewise linear function $f \in \mathcal{F}$.

In contrast to the 1D case, where $w_i = f(z_i)$, we will show that the fitting values w within each segment does not necessarily form an affine plane with respect to their coordinates z. Hence, the approach is not exact, but only an approximation of a piecewise affine regression problem.

4.3.1 Modeling as a MIP

We again divide the edge set E of the grid graph into its horizontal (row) edge set E^r and its vertical (column) edge set E^c. So $E = E^r \cup E^c$, and $E^r \cap E^c = \emptyset$. Denote $e^r_{i,j} \in E^r$ to present edge $((i, j), (i, j + 1))$ and $e^c_{i,j} \in E^c$ to represent $((i, j), (i + 1, j))$. Again for simplicity, we denote the binary edge variables $x^r_{i,j} := x_{e^r_{ij}}$ and $x^c_{i,j} := x_{e^c_{ij}}$.

The approximate piecewise affine Potts model for 2D is obtained by formulating (4.4)

per row and column

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} |w_{i,j} - y_{i,j}| + \lambda \sum_{e \in E} x_e \tag{4.5}$$

$$|\nabla_r^2 w_{i,j}| \leq M(x_{i,j-1}^r + x_{ij}^r), \;\; i \in [m], \; j \in [2:n-1], \tag{4.5a}$$

$$|\nabla_c^2 w_{i,j}| \leq M(x_{i-1,j}^c + x_{ij}^c), \;\; j \in [n], \; i \in [2:m-1], \tag{4.5b}$$

$$w_{ij} \in \mathbb{R}, \quad i \in [m], \; j \in [n], \tag{4.5c}$$

$$x_e \in \{0,1\}, \;\; e \in E, \tag{4.5d}$$

where M is again the big-M constant. Here, $\nabla_r^2 w_{i,j} = w_{i,j-1} - 2w_{i,j} + w_{i,j+1}$, and $\nabla_c^2 w_{i,j} = w_{i-1,j} - 2w_{i,j} + w_{i+1,j}$. That is, the discrete second derivative with respect to z_1 and z_2-axis.

Upon solving (4.5), it serves for the purpose of image denoising by computing w. But two questions still remain: does the integer solution x represent a valid segmentation? If so, does the fitting value w within segment i share the same affine function f^i?

The answers to both questions are "no", unfortunately. We will show in the next two sections that, the first "no" could be fixed by adding the multicut constraints. But the second one could be only addressed as post-processing, thus making the resulting model an approximate one.

4.3.2 Multicut constraints for consistent segmentation

We first illustrate one counter-example where the optimal solution x to (4.5) does not form a valid segmentation.

Lemma 4.2. *The optimal solution $x^\star$ of problem* (4.5) *does not necessarily satisfies the multicut constraints* (3.8a)*, thus does not form a valid segmentation.*

Proof. We prove this lemma by constructing a counter-example as follows:

In the left image of Figure 4.6, the intensities y of all 15 pixels lie exactly on two affine planes with respect to their coordinates $z = (z_1, z_2)$. The affine function of the left plane is $y = 4 - z_2$ and the right one is $y = z_2$. We shall see that the 3 pixels with intensities $y = 2$ lie on both affine planes with respect to the coordinates z.

If we project each column of the pixels in $3D$ into the z_1, y-space, the intensities y of each five pixels lie in the same straight line, hence no jumps occur. If we project it into the z_2, y-space, for every row (5 pixels) of the image grid, it is exactly the case we studied in Figure 4.5. We have showed there that the optimal solution is not unique. For instance, either e_l or e_r can be active, but not both.

Since we construct the problem without adding any noise, the optimal fitting error between w and f (the data term of the Potts model) equals zero, and the objective function value equals 3λ. Note that there are multiple optimal solutions that can achieve the

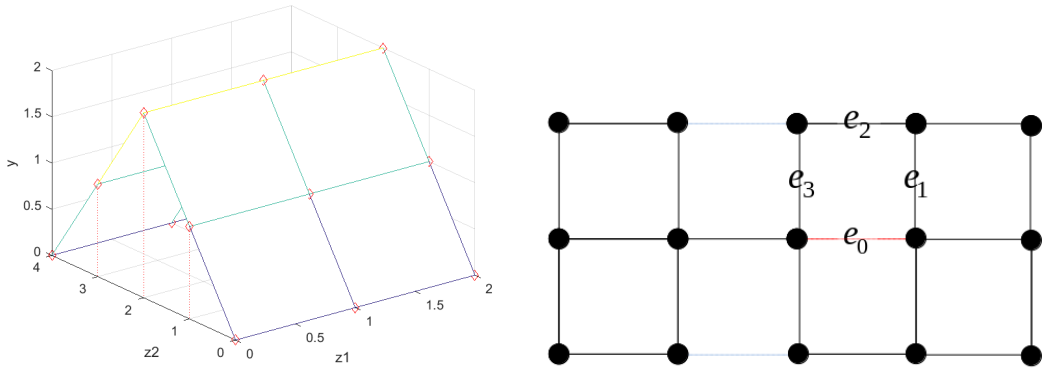


Figure 4.6 – A counter-example where model (4.5) does not form a valid segmentation. Left image: 3D view of input image. Right image: the corresponding graph.

same objective value. For example, the two blue edges plus one red edge in the right image of Figure 4.6. However, this solution does not satisfy the multicut constraints (3.8a), since there exists one cycle (e.g., cycle e_0-e_1-e_2-e_3) that violates it. Hence, we have found an optimal solution x to (4.5) that does not form a valid segmentation. □

Note that in the above illustrated example, it may be possible for the MIP solver to reach at an optimal solution which is also feasible with respect to segmentation (i.e., the 2 dashed blue edges plus the black one in the same column are active). However, this can not be guaranteed, since the two solutions have the same optimal objective value 3λ. This is usually referred to as *symmetry* in combinatorial optimization, and hence the multicut constraints (3.8a) are, in a sense, *symmetry-breaking* inequalities with respect to the denoising problem of problem (4.5). For more literatures on symmetry-breaking constraints in combinatorial optimization, please see [37].

4.3.3 The main formulation in 2D

We thus need to add the multicut constraints (3.8a) to the piecewise affine Potts model (4.5) to form a valid segmentation.

The main formulation of this chapter is then obtained by adding the multicut constraints (3.8a) to the Potts model (4.5)

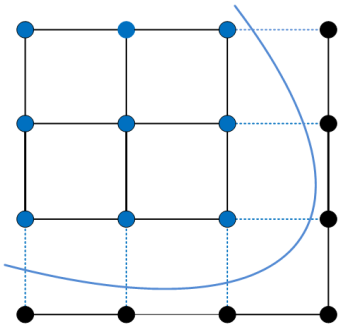


Figure 4.7 – An example with a 9-pixel segment.

$$\min\ (1-\lambda)\sum_{i=1}^{m}\sum_{j=1}^{n}|w_{i,j}-y_{i,j}|+\lambda\sum_{e\in E}x_e \tag{4.6}$$

$$|\nabla_r^2 w_{i,j}| \le M(x^r_{i,j-1}+x^r_{ij}),\ \ i\in[m],\ j\in[2:n-1], \tag{4.6a}$$

$$|\nabla_c^2 w_{i,j}| \le M(x^c_{i-1,j}+x^c_{ij}),\ \ j\in[n],\ i\in[2:m-1], \tag{4.6b}$$

$$\sum_{e\in C\setminus\{e'\}}x_e \ge x_{e'},\ \ \forall \text{ cycles } C\subseteq E, e'\in C, \tag{4.6c}$$

$$w_{ij}\in\mathbb{R},\ \ i\in[m],\ j\in[n], \tag{4.6d}$$

$$x_e\in\{0,1\},\ \ e\in E. \tag{4.6e}$$

Note that the number of multicut constraints (4.6c) is exponential, hence is not possible to include into (4.6) at one time. We will discuss about the solution techniques in Section 4.4, as well as the strategies for computing all the parameters of (4.6).

4.3.4 Approximate model for piecewise affine regression

In this section, we show that formulation (4.6) in 2D does not necessarily correspond to an exact piecewise affine fitting problem as in (2.4) or (2.5), hence an approximate model.

Theorem 4.3. *The optimal solution $w^\star$ of problem* (4.6) *does not necessarily corresponds to a feasible solution in the nonparametric piecewise affine fitting problem* (2.5).

Proof. We prove this theorem by constructing a counter-example where the optimal solution $w^\star$ of (4.6) within one segment does not form an affine plane with respect to the coordinates z.

Suppose an optimal solution $x^\star$ produce the resulting segmentation in Figure 4.7, where 9 pixels on the top left form one segment. We restrict ourselves to this segment where the coordinates of the pixels range from $(0,0)$ to $(2,2)$.

By constraint (4.6a), the $w^\star$ of 3 pixels on each row satisfy the same linear function. Assume in the first row, the linear function $y = a_1 z_1 + b_1$, and $y = a_2 z_1 + b_2$ for the second row. Then the fitting value w of the 6 pixels on the first two rows can be easily computed, and shown as follows:

$$\begin{bmatrix} b_1 & a_1 + b_1 & 2a_1 + b_1 \\ b_2 & a_2 + b_2 & 2a_2 + b_2 \end{bmatrix}.$$

According to constraint (4.6b), we can compute w_{22} using the values of w_{02} and w_{12}, and $w_{22} = 2w_{12} - w_{02} = 4a_2 + 2b_2 - 2a_1 - b_1$. We then test if the $w^\star$ of 3 diagonal pixels of $W \in \mathbb{R}^{3\times 3}$ lie in the same linear function, where

$$w_{00} - 2w_{11} + w_{22} = 2(a_2 - a_1).$$

For any constants $a_1 \neq a_2$, the both sides of equation do not equal to 0, hence the fitting value $w^\star$ of the 9 pixels does not lie in the same affine function.

□

Note that although the optimal solution $w^\star$ of the main model (4.5) does not necessarily satisfy any piecewise affine function $f \in \mathcal{F}$, we can still fit an affine function within each segment once we have a valid segmentation. Hence, we get a valid piecewise affine function $f \in \mathcal{F}$ as post-processing.

4.3.5 Cardinality and bounding constraints

If one has prior knowledge about the MIP problem and its solutions, one can add additional cuts to reduce the feasibility solution set.

Similar to Section 3.3.3, one of such cuts is the cardinality constraint. When working on 2D images, one can add cardinality constraints (3.10) per row and column. In addition to that, we could also add the lower and upper bound constraints for the fitting variable w. Denote the lower and upper bound of w_{ij} variable L_{ij} and U_{ij}. The corresponding bounding constraints for w_{ij} is then the following

$$w_{ij} \in [L_{ij}, U_{ij}].$$

This strategy was reported to achieve better results in [32].

We need some strategy of automatically computing the bounds on the number of jumps k (of the cardinality constraints) and the fitting value w. We will show computational experiments in Section 4.5 on the effects of adding the above two types of constraints to the MIP formulation (4.6).

4.4 Solution Techniques

The resulting problem (4.6) is again a MILP and in general $\mathcal{NP}$-hard to solve, hence a fast heuristic is desired to provide a feasible solution as a starting point for the solver.

4.4.1 Region fusion based heuristic for piecewise affine regression

Like in Section 3.4.1, we adopted the region fusion based greedy algorithm in [4] to provide an initial solution for the piecewise affine Potts model (4.6). The modified algorithm is similar to Algorithm 3.1, except for the following three changes:

- The algorithm starts with all groups of 2×2 squared pixels belonging to one superpixel V_i. If the image can not be perfectly divide by squares, we just merge the rest pixels to their nearest group. We then perform a parametric affine fitting (2.4) within each group V_i, and denote $\beta^i = (\beta_1^i, \beta_0^i)$ as the affine coefficients, where $\beta_1^i \in \mathbb{R}^2$. We then let Y_i in Algorithm 3.1 equals β^i, and hence is a vector of length 3.

- The merging criteria (3.12) is still

$$\tau_i \tau_j \|Y_i - Y_j\|_1 \leq \kappa \gamma_{i,j} (\tau_i + \tau_j),$$

  where $Y_i = \beta^i \in \mathbb{R}^3$ is the affine coefficients of group i after we conduct an affine regression within the group.

- If the above condition holds, we merge group V_i and V_j into $V_{i'}$, and the updated $w_{i'}$ is obtained by conducting an affine fitting of the new group $V_{i'}$.

As apposed to Section 3.4.1, the above method to compute the fitting matrix W no longer corresponds to a minimizer of the following Potts mode:

$$\min_w \ \tau_i \|w_i - Y_i\|_1 + \tau_j \|w_j - Y_j\|_1 + \kappa \gamma_{ij} \|w_i - w_j\|_0 .$$

However, since we would like Y_i to represent the slope of group V_i, this is a better way to preserve the slope of the resulting group. We will adopt the same growing strategy for parameter κ in Section 4.5.

4.4.2 Exact branch and cut algorithm

Different from the piecewise constant Potts model (3.7), the MILP formulation of the piecewise affine Potts model (4.6) has exponential many constraints. This is due to the presence of the multicut constraints (4.6c).

Hence, apart from the usual branch-and-cut algorithm inside the MIP solver as described in Section 2.5.3, we will focus on the cutting planes method that iteratively add constraints of type (4.6c) in this section.

Cutting planes. Similar to the cutting planes method that solves the multicut problem (3.3.1) as described in [3], we start solving the MILP (4.6) by ignoring constraints (4.6c), or with few of them (e.g., the 4-edge cycle constraints (3.9)). We then check the feasibility of the resulting solution. If it is already feasible with respect to (4.6c), we are done and the optimal solution to (4.6) is achieved. Otherwise, we identify current separation problem and then add the corresponding violated constraints (cuts) to (4.6) (to cut off the current infeasible solution). We then resolve the resulting MILP, and this procedure is repeated until the solution of the MILP is feasible.

Separation problem. Although the separation problem for constraints (4.6c) is generally $\mathcal{NP}$-hard, it is polynomial to separate integer infeasible solutions at every iteration. As the branch-and-cut algorithm proceeds within the MIP solver, one gets an integer solution of (4.6) without the multicut constraints (4.6c). A simple algorithm mapping the edge labeling to node labeling of the graph is invented and applied. One can then test the feasibility of the current integer solution (with respect to the multicut constraints (4.6c)) by checking if there exists an active edge ($x_e = 1$) where the two end nodes have the same label.

If so, the current integer solution is infeasible and the separation problem then tries to seek an violated constraint to be added into the original problem, to cut off the current infeasible solution. This could be done by conducting a breath-first search algorithm where the start and end node are just the two nodes of the active edge. For more details on how to detect the feasibility of the current integer solution and identify the violated constraints, please see [23].

Finally, there are two things that could be looked into in future research. Firstly, the cutting plane method could be also applied on fractional solutions, where the user needs to set a threshold so that the edges above it are considered active and vise versa. This way, more cuts will be added. Secondly, the breath-first search algorithm in the separation problem adds a valid inequality to the multicut polytope (S_m), but not necessarily a chordless cycle constraint. As we already know in Section 3.3.2, if the cycle is chordless, the corresponding multicut constraint is facet-defining. Hence, one may design an algorithm that is guaranteed to find a chordless cycle. However, this may increase the complexity of the separation problem. A detailed study is beyond the scope of this chapter and we leave it for future research.

4.5 Computational Experiments on Synthetic Images

Based on the MILP formulation of the piecewise affine Potts model (4.6), we develop and compare several variants and report their computational results. We also compare

model (4.6) against the total generalized variation model [60]. The experiments in this section are based on 2 synthetic images with size 32×32 and 28×32 as shown in Figure 4.8 and 4.9. We normalized the pixels' intensities y to $[0, 1]$.

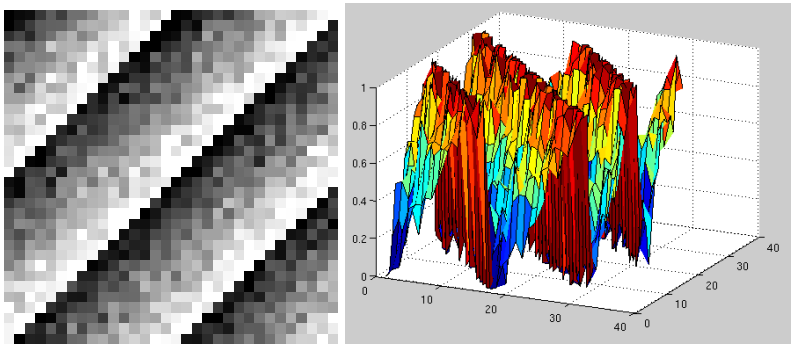


Figure 4.8 – A synthetic image with 4 affine pieces and with Gaussian noise, 2D and 3D view.

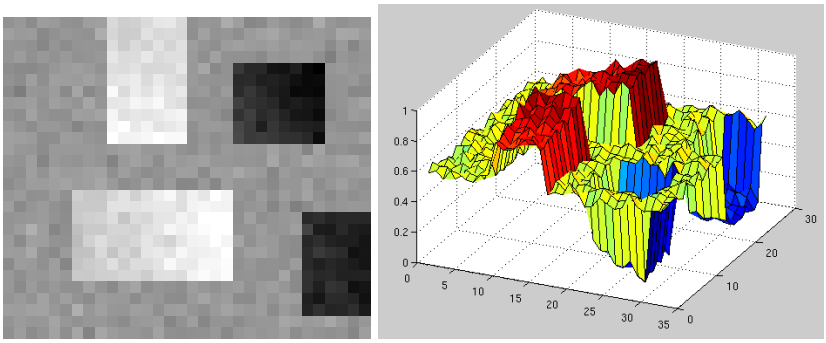


Figure 4.9 – A synthetic image with 4 affine pieces plus background and with Gaussian noise, 2D and 3D view.

We will conduct comparisons of the following variants of models:

- **MP**: The MILP formulation of the piecewise affine Potts model (4.6), solved using cutting plane method described in Seciton 4.4.2.
- **MP-LP**: The linear programming relaxation of problem MP.
- **MP-4C (MC)**: Problem MP with the 4-edge cycle multicut constraints as initial inequalities, which we will later denote MC for simplicity.
- **MC-C**: Problem MC with cardinality constraints introduced in Section 4.3.5.

- **MC-B**: Problem MC with bounding constraints on the fitting value w, also introduced in Section 4.3.5.
- **MC-M**: Problem MC which uses different parameter to compute M automatically.
- **MC-H**: Problem MC which adopts the solution of our heuristic as an initial input for the MILP solver.
- **TGV**: The total generalized variation model introduced in [60], which we will use the second derivative model here.

4.5.1 Automatic computation of parameters

Parameter M is for the "big M" constraint in problem MP (4.6). In principle, it should be big enough so that the "big M" constraints (4.6a,4.6b) are always valid. On the other hand, it should be not too big, or it may harm the tightness of the LP relaxation. Denote M_i^r as the big M constant for row i, and M_j^c for column j, further denote y_i^r be the vector of y values in row i, and y_j^c for column j. We compute M separately for every row and column, and set M_i^r to the largest value of $|\nabla^2 y_i^r|$ for the row i, and M_j^c to the maximum of $|\nabla^2 y_j^c|$. Here, the absolute functional of a vector $|(x_1, x_2, \ldots, x_n)| := (|x_1|, |x_2|, \ldots, |x_n|)$. Recall that the *discrete second derivative* $\nabla^2 y$ of a vector $y \in \mathbb{R}^n$ is defined as the $n-2$ dimensional vector $(y_1 - 2y_2 + y_3, y_2 - 2y_3 + y_4, \ldots, y_{n-2} - 2y_{n-1} + y_n)$. We further multiply it by an user defined factor ξ_1 to make M big enough. So $M_i^r = \xi_1 \cdot \max_i |\nabla^2 y_i^r|$, and $M_j^c = \xi_1 \cdot \max_j |\nabla^2 y_j^c|$.

Parameter $\boldsymbol{\lambda}$ is the regularization term employed to avoid over-fitting in problem MP (4.6). We will set λ independently for each row and column, denoted λ_i^r and λ_j^c. We let $\lambda_i^r = \frac{1}{2}\xi_2 \cdot \max_i |\nabla^2 y_i^r|$ and $\lambda_j^c = \frac{1}{2}\xi_2 \cdot \max_j |\nabla^2 y_j^c|$, where ξ_2 is another user defined parameter.

As we assume the input image contains noise, when there exists an outlier at pixel (i, j), MP will tend not treat it as a one-pixel segment. Since otherwise, the four associated edges to (i, j) are then all active , which will incur the penalty value of $2(\lambda_i^r + \lambda_j^c)$.

Parameter $\boldsymbol{L}$ and **$\boldsymbol{U}$** are the lower and upper bound on the fitting value w associated with the bounding constraints in Section 4.3.5. Denote the upper and lower bound of variable w_{ij} in (4.6) $L_{i,j}$ and $U_{i,j}$. We let $L_{i,j}$ be the smallest y value of the neighboring 8 pixels plus w_{ij} itself, and $U_{i,j}$ be the largest. We further multiply (divide) $U_{i,j}$ ($L_{i,j}$) by an additional user defined factor $\xi_3 \geq 1$ to ensure feasibility.

Parameter $\boldsymbol{J}$ is the upper bound on the number of active edges (jumps) associated with the cardinality constraints in Section 4.3.5. Let J_i^r denote the upper bound on the number of jumps in row i, and J_j^c for column j, respectively. The user then selects a threshold ξ_4, and J_i^r is computed as the number of entries in $|\nabla^2 y_i^r|$ that is no less than $\xi_4 \cdot \max_i |\nabla^2 y_i^r|$. That is, we neglect small values in $|\nabla^2 y_i^r|$ that is below the threshold

second derivative value, and only consider those with large values as potential active edges. The values of J_j^c are computed similarly for every column j.

4.5.2 Detailed comparison of different models

MP versus MP-LP

We first conduct computational experiments on solving MP-LP against MP. When it comes to the separation problem of MP-LP, we will treat any edge with $x_e > 0.0001$ as an active edge, and then the algorithms run just like MP. We set the parameter M to 1, L to 0 and U to 2 to ensure feasibility of (4.6). We further set $\xi_2 = 1$ for the regularization parameter λ. We are most interested in the percentage of fractional solutions of the binary edge variables $x^\star$ of MP-LP.

The percentage of fractional solutions for the 2 images are: 74.5% and 75.9%. The running time of MP-LP for the 2 images are: 0.10 and 0.07 second.

Because of the big M constraints (4.6a,4.6b) and the assumption that M is big enough, constraints (4.6a,4.6b) will hold even if x is very small. This cause a very high percentages of fractional solutions when we just solve the LP. Hence, sophisticated rounding methods must to be applied as post-processing, hoping to get an integer feasible solution of the Potts model (4.6). However, the optimality is not guaranteed, not even the feasibility. On the other hand, by solving MP, we not only get a feasible solution, if no time limit is given, we will also get the optimal solution of (4.6).

We will report the computational analysis of MP in the next section.

MP versus MP-4C (MC)

We conduct computational experiments on solving MP with and without the 4-edge cycle multicut constraints as initial inequalities. Starting from this section, we set a time limit of 50 seconds for the MIP solver. We further let $M = 1$ and $\xi_2 = 0.5$ for computing λ.

Without these constraints, the MIP solver failed to find any active cuts in both images within the time limit, i.e., all x equals 0 and the segmentation contains only the whole image as one segment.

We report the energy, MIP gap and running time of all 4 scenarios in Table 4.1. As we can see, MP-4C greatly improves the results, e.g., reducing the MIP gap from 82.1% to 20.2% in the first experiment. The segmentation results of MP-4C are plotted in Figure 4.10.

Since in both experiments, MP-4C shows promising results. From now on, we will make it a baseline for competing against other models.

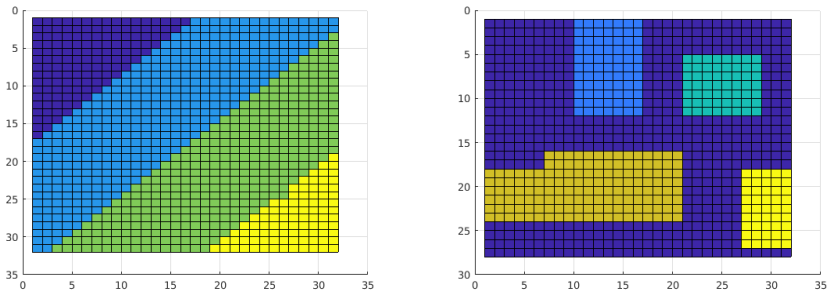


Figure 4.10 – Segmentation results of MP-4C.

	Image 1 w/o	Image 1 w.	Image 2 w/o	Image 2 w.
Energy	511.3	**109.8**	534.9	**38.2**
Gap	82.1%	**20.2%**	94.8%	**34.8%**
Time	50	50	50	50

Table 4.1 – Statistics of MP with and without the 4-edge cycle constraints.

MC versus MC-C

In this section, we will test if adding the cardinality constraints (introduced in Section 4.3.5) to MC speeds up the computation. We let $\xi_4 = 0.4$ for computing the upper bound on the number of jumps. And we again set $M = 1$ and $\xi_2 = 0.5$.

We report the energy, MIP gap of all 4 scenarios in Table 4.2. We can see that in both cases, MC-C has worse energy and MIP gap. The segmentation results of MC-C are plotted in Figure 4.11.

Note that we also tried to set ξ_4 very high (thus the upper bound J becomes small), and this resulted in unwanted solutions. For example, we encountered an extreme scenario where $J_1^r = 0$ in Figure 4.9, i.e., it is constrained that there is no active edge in the first row of the image grid (which should be two ideally).

	Image 1 w/o	Image 1 w.	Image 2 w/o	Image 2 w.
Energy	**109.8**	154.4	**38.2**	38.9
Gap	**20.2%**	43.2%	**34.8%**	38.8%

Table 4.2 – Statistics of MC with and without the cardinality constraints (MC-C).

MC versus MC-B

We test if adding the bounding constraints (introduced in Section 4.3.5) to MC speeds up the computation. We let $\xi_3 = 3$ and 1.5 for computing lower and upper bounds for

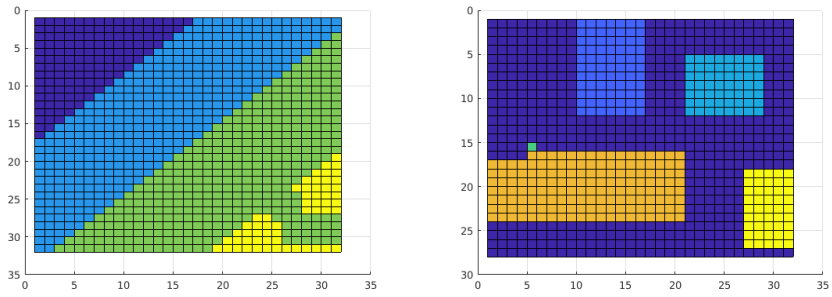


Figure 4.11 – Segmentation results of MC-C.

variables w. And we set $M = 1$ and $\xi_2 = 0.5$.

Table 4.3 reports the energy, MIP gap of 5 different scenarios. We can see from the table that MC-B achieves better performance in terms of both energy and MIP gap on the first image. However, on the second image, only the energy (best feasible solution) improves when $\xi_3 = 1.5$. The segmentation results of MC-C are plotted in Figure 4.12.

	Img1	Img1, $\xi_3 = 3$	Img2	Img2, $\xi_3 = 3$	Img2, $\xi_3 = 1.5$
Energy	109.8	**106.9**	38.2	41.4	**37.8**
Gap	20.2%	**15.7%**	**34.8%**	47.7%	38.6%

Table 4.3 – Statistics of MC with and without the bounding constraints (MC-B).

Note that in theory, we prefer L and U to be as tight as possible. However, this is not possible in practice, as we encountered an extreme scenario with small ξ_3 where Cplex failed to find any feasible solution within the time limit.

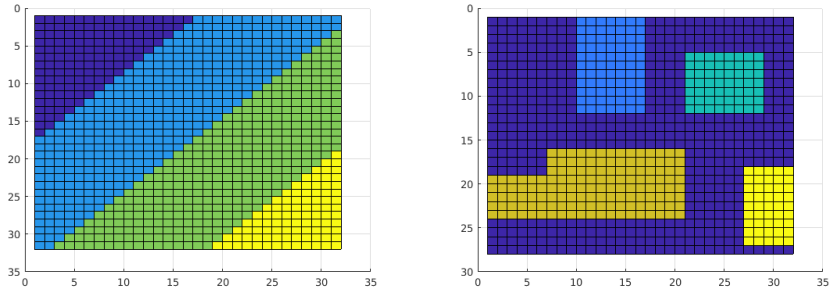


Figure 4.12 – Segmentation results of MC-B with $\xi_3 = 3$ and 1.5 for the second image.

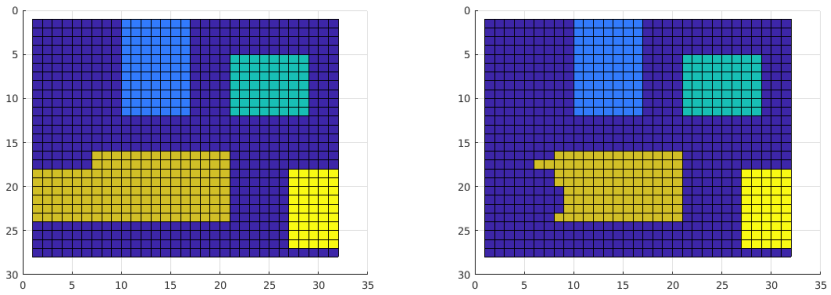


Figure 4.13 – Segmentation results of MC ($\xi_2 = 0.5$) and MC-M ($\xi_1 = 2.5$) on the second image.

MC versus MC-M

In this section, we conduct computational experiments on changing values of M and we are interested if this affects the computation. We let $\xi_1 = 1.5$, 2 and 2.5 for computing M. We again let $\xi_2 = 0.5$ for computing λ.

The energy and MIP gap of 5 different scenarios are reported in Table 4.4, where we notice MC-M improve both scores in the first image. Although both models achieve more or less the same energy (38.2) on the second image, the segmentation results of MC and MC-M are plotted in Figure 4.13, showing a significant difference.

	Img 1	Img 1, $\xi_1 = 2$	Img 2	Img 2, $\xi_1 = 2.5$	Img 2, $\xi_1 = 1.5$
Energy	109.8	**106.9**	**38.2**	**38.2**	64.8
Gap	20.2%	**17.6%**	**34.8%**	39.5%	60.9%

Table 4.4 – Statistics of MC with different big M value.

Note that in theory, M should be big enough for constraints (4.6a,4.6b) to hold. However, if M is too big, it may harm the effectiveness of the LP relaxation. On the other hand, if we set ξ_1 too small, it may happen w could not take its desired value because the right hand side of constraints is not big enough. Hence, cares must be taken when choosing the right value for ξ_1.

MC versus MC-H

In this section, we test the effects of adding initial solutions computed by our region fusion based heuristic (introduced in Section 4.4.1) to the MIP solver. We are interested if this could help improve the computational results. We again let $M = 1$ and $\xi_2 = 0.5$, and the time limit is 50 seconds.

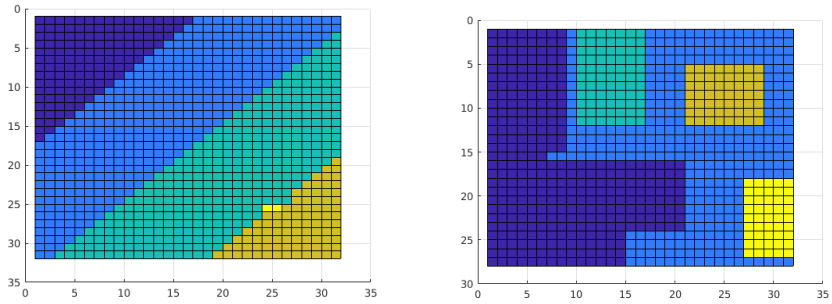


Figure 4.14 – Segmentation results of MC-H.

Our heuristic algorithm is fast to compute, only takes 0.12 and 0.09 second on the tested two images. Keep in mind that we only provide the MIP solver with initial solutions to the binary variables x of problem (4.6). Hence it takes time for the solver to compute the resulting continuous variables w by solving a linear program. We report that it takes 1.1 and 0.8 seconds for the solver to get feasible solutions with objective value 416.9 and 129.4. Moreover, the MIP solver is able to search for new feasible solutions based on the provided initial solutions, and the user can control the efforts of the search by setting different parameters in the solver. We report that the MIP solver found better solutions with objective 142.9 and 52.6 in 5.2 and 4.6 seconds on the first and second image, respectively.

The energy and MIP gap of 4 different scenarios are reported in Table 4.5, where the row "Initial" denotes the energy of the initial solution found by MIP solver's default heuristic (column "w/o") and by our designed heuristic (column "w."). We notice that our region fusion based heuristic provides the MIP solver an initial solution with much lower energy, and this help improve scores on the first image. While it is not beneficial on the second image, we argue it may be related to the extra effort on exploring new solutions based on the initial input. In addition, different branching strategies due to the given initial solution may also cause different computational behaviors.

The segmentation results of MC-H based on initial solutions computed by our region fusion based heuristic are plotted in Figure 4.14.

	Image 1 w/o	Image 1 w.	Image 2 w/o	Image 2 w.
Initial	$\mathbf{511.3}$	416.9	$\mathbf{534.9}$	129.4
Energy	109.8	$\mathbf{109.2}$	$\mathbf{38.2}$	50.6
Gap	20.2%	$\mathbf{19.7}\%$	$\mathbf{34.8}\%$	52.2%

Table 4.5 – Statistics of MC with and without the initial solution (MC-H).

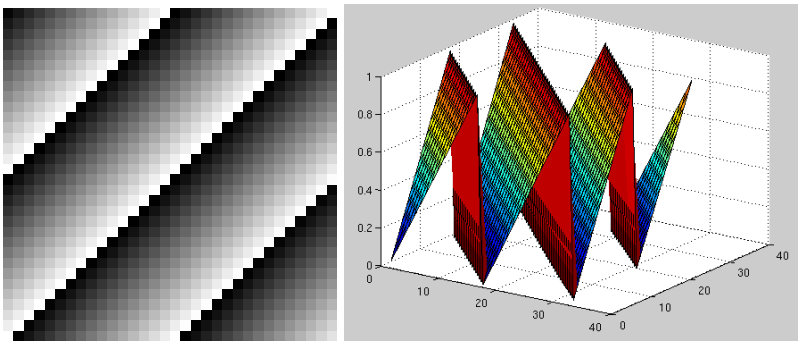


Figure 4.15 – Results of MC-M-B on the first image, 2D and 3D plot.

MC-M-B versus TGV

While TGV is designed for denoising, it does not have a systematical way to partition the given image. Our model, apart from denoising, treats image segmentation as an edge labeling problem, where the active edge denote the boundary edge between two segments. In this section, we present a comprehensive comparison between TGV against our model MC-M-B (MC-M with the bounding constraints).

For the first image, we set $\xi_1 = 1.5$, $\xi_2 = 0.5$, $\xi_3 = 1.5$ and time limit to 50 seconds.

The solution of MC-M-B plotted in Figure 4.15 is found in less than 1 second, and the MIP gap equals 12.4% upon hitting the time limit.

The solution found by TGV is plotted in Figure 4.16, both in 2D and 3D. We choose the parameters $\sigma_1 = 0.8$, $\sigma_2 = 0.1$, and the number of iterations is set to be 5000.

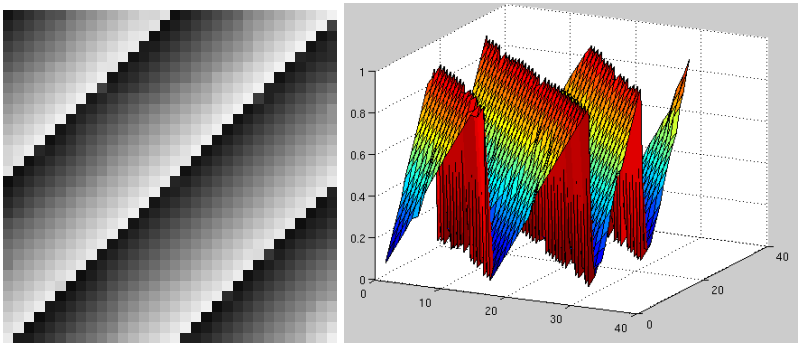


Figure 4.16 – Output of TGV on the first image, 2D and 3D plot.

For image 2, we have the same parameter as the first image. The solution of MC-B-M plotted in Figure 4.17 is found in 41 seconds, and the MIP gap equals 27.6% upon hitting the time limit of 50 seconds.

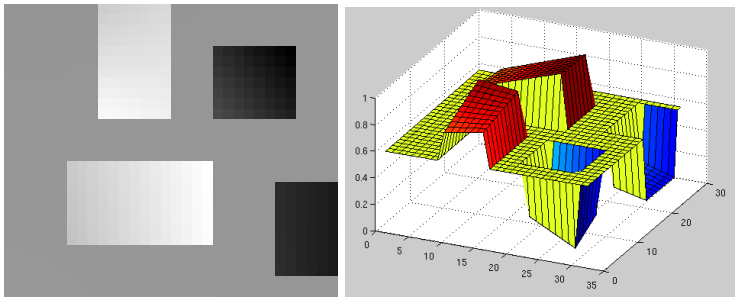


Figure 4.17 – Results of MC-B-M on the first image, 2D and 3D plot.

The solution found by TGV is plotted in Figure 4.18. We set the parameters of TGV $\sigma_1 = 0.6$, $\sigma_2 = 0.05$, and the number of iterations is again set to be 5000.

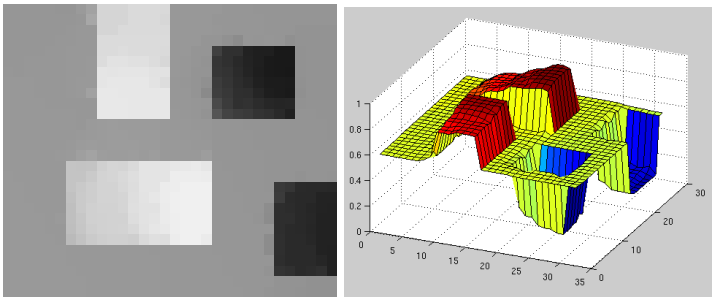


Figure 4.18 – Output of TGV on the second image, 2D and 3D plot.

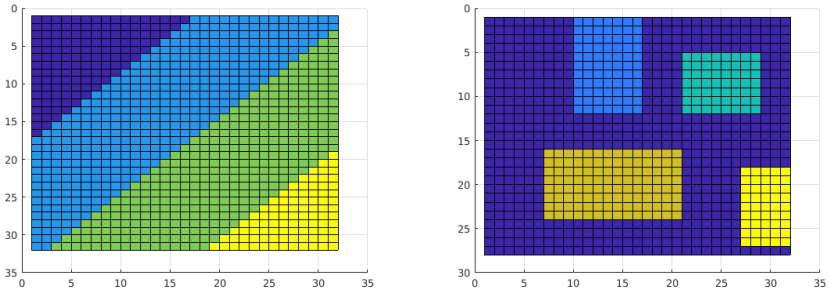


Figure 4.19 – Segmentation results of MC-B-M on two synthetic images.

Concluding from the above comparisons, our model presents approximate piecewise affine results, while TGV does not. However, note that each piece out of our model is not necessarily affine. One example could be seen in the right image of Figure 4.17, where the front red piece is not affine.

Finally, the segmentation results of MC-B-M on both images are plotted in Figure 4.19, presenting a "perfect" partitioning of the input two synthetic images.

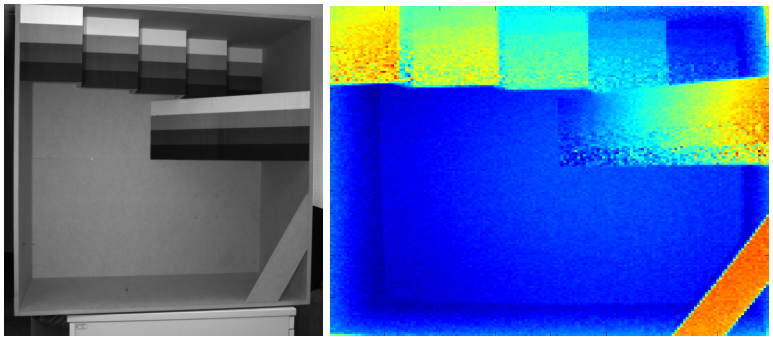


Figure 4.20 – Photo taken by a normal camera of HCIBOX (left) and inverse of the depth image (right).

4.6 Computational Experiments on Depth Images

4.6.1 The HCIBOX depth instances

Additional computational tests are carried out on the HCIBOX depth instances [68] (Figure 4.20).

The left image is taken using a normal camera. A time of flight camera is used to produce a depth image of the same scene, where each pixel now represents the distance of the object to the camera. The right image is obtained by taking the inverse of the depth image. As we can see from the 3D plot in Figure 4.21, it exhibits the piecewise affine property with respect to its coordinates. The image is of size 158×158.

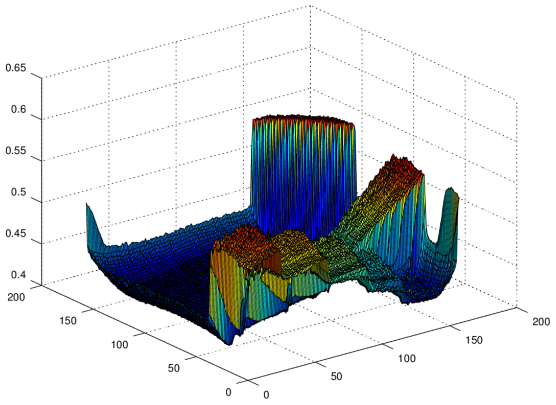


Figure 4.21 – 3D plot of the HCIBOX instance.

We resize the image to 50×50, and adopts MC-M-B for the task of segmentation and denoising. We set $\xi_1 = 15$, $\xi_2 = 6$, $\xi_3 = 10$ and the time limit to 200 seconds. The

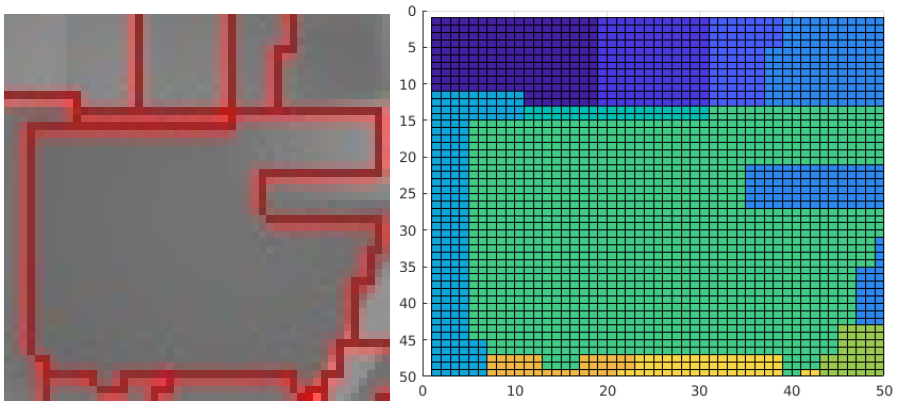


Figure 4.22 – Segmentation result of HCIBOX using MC-M-B.

MIP solver loaded our heuristic solution with objective value 284.9 in 0.86 seconds, and found a better one with reduced objective 56.4 in 12.8 seconds. It did not find any better solution, with MIP gap equals 94.0% given the 200 seconds time limit.

The segmentation results of the HCIBOX instance based on the initial solution computed by our heuristic is plotted in Figure 4.22, where the red lines denote active edges in the left image. As we can see from the result, our model characterize most of the affine shapes, while still missing some details. For example, the top left dark blue piece should be ideally divided into two.

4.6.2 The Middlebury Stereo Dataset

We further conduct experiments using the Middlebury Stereo Datasets in [69], where the instance of "Teddy" is tested. Figure 4.23 shows the original color image along with its disparity map. The later is produced using two images of the same scene taken from different locations.

Figure 4.23 – The Teddy image (left) and the disparity map (right).

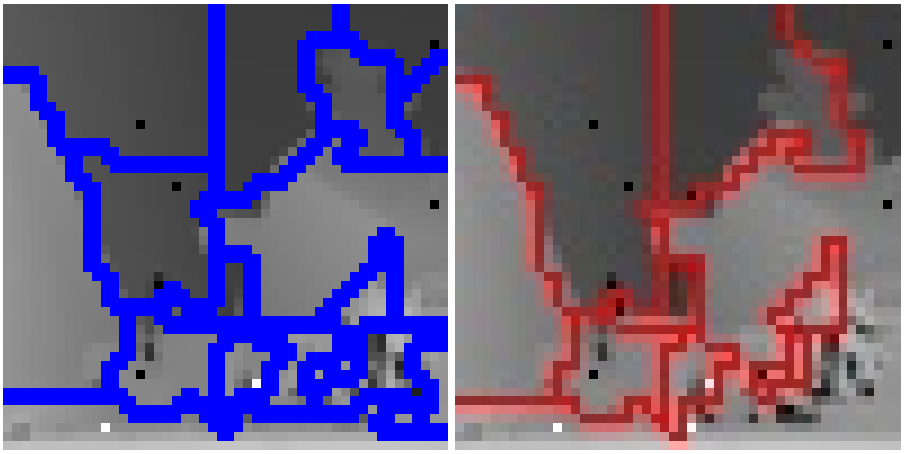


Figure 4.24 – Left: segmentation result using our heuristic. Right: MC-M-B-C result.

We add salt and pepper noise to the ground-truth disparity map. We again resize the image to 50×50, and first apply our region fusion based heuristic, where $\lambda = 0.2$. The heuristic takes only 0.2 seconds, and the segmentation result of the heuristic is plotted in the left image of Figure 4.24.

We adopt MC-M-B-C (MC-M-B with the cardinality constraints) to solve it. We set $\xi_1 = 5$, $\xi_2 = 0.6$, $\xi_3 = 5$ and $\xi_4 = 0.2$ for the cardinality constraints. The time limit is set to 200 seconds.

Due to the presence of the cardinality constraints, the initial solution provided by our heuristic is not feasible to MC-M-B-C. However, the MIP solver is able to "fix" the initial solution and find one with the objective value 96.2 within 10 seconds. It did not find any better solution, and the MIP gap equals 82.1% upon hitting the 200 seconds time limit. Again, some details have not been segmented, like the teddy bear in the top right.

The segmentation results of both our heuristic and MC-M-B-M on the "Teddy" instance is plotted in Figure 4.22, where the blue and red lines denote active edges in the segmentation.

4.6.3 Strategies towards larger images

Since all our models are $\mathcal{NP}$-hard to solve, the computational experiments are so far implemented only on small images up to size 50×50. When dealing with larger images, we could first divide the image into subimages and apply our model within each subimage. Just like in Section 3.6, it could then be used to generate superpixels on depth images or disparity maps.

One example is presented in Figure 4.25, which generates superpixels on the "Teddy" instance. The image is of size 881×734. Here, 180 subimages are first created, then

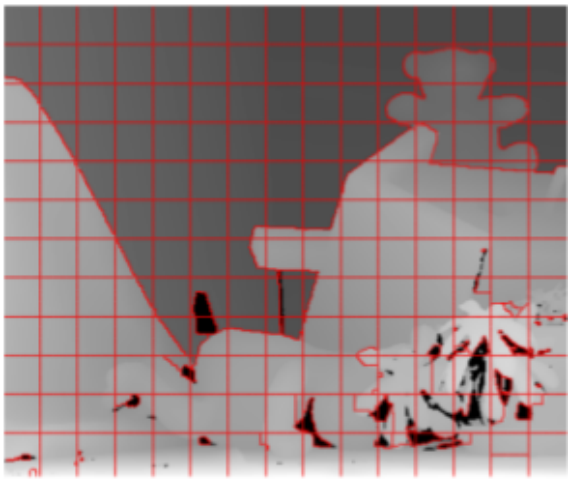


Figure 4.25 – Superpixel generation on the Teddy disparity map.

MC is applied within each subimage, with $\xi_2 = 0.2$.

Due to certain environments which lack object's color information, there have been recently active research efforts on generating superpixels on depth image, without using any RGB features. Examples are the works by [70, 71]. An extensive comparison between our methods with them will be conducted in future research.

4.7 Conclusions

In this chapter, we have presented a combined unsupervised image segmentation and denoising framework that is based on the approximate piecewise affine Potts model. We formulate it as a MIP and solve it with a standard optimization solver.

We conducted extensive experiments on different variants of models and study the effects of adjusting parameters. We have introduced its application to depth image denoising and segmentation, as well as superpixel generation on depth images. Potential fields such as stereo reconstruction and stereo matching are both applicable, since they often meet the piecewise affine assumptions (e.g., slanted surfaces in [72]).

Since our model deals with the more general piecewise affine fitting problem, applications beyond the scope of computer vision are also of interest and will be investigated in the future.

Chapter 5

Multi-label MRF with Connectivity Priors for Interactive Segmentation

Integer Linear Programming (ILP) formulations of Markov random field (MRF) models are standard in computer vision, usually with only local priors [24, 39, 41, 56, 73]. Global connectivity priors were investigated previously [74, 75], but they all restricted themselves to Linear Programing (LP) relaxations [74, 75] or simplified versions [40]. In this chapter, we investigate solving the ILP of the multi-label MRF with connectivity priors to globally optimal solutions, based on some user scribbles input. Since the resulting problem is $\mathcal{NP}$-hard, we propose a fast heuristic algorithm that is adapted from the region fusion method [4]. The commercial ILP solver such as Cplex can thus utilize the solution provided by the heuristic as an initial solution. Since it provides globally optimal solution, it can be used off-line to generate ground-truth labeling, that serves as quality check for any fast heuristic algorithms. Although segmentation serves as the main application in this chapter, denoising could be achieved easily afterwards. We demonstrate the power and usefulness of our method by extensive experiments on the BSDS500 [5] and PASCAL [76] image datasets, with and without noise. We also test on medical images with trained probability maps.

5.1 Overview

In contrast to unsupervised and fully supervised image segmentation, interactive image segmentation deals with partitioning an image based on user-provided input, which could be treated as *semi-supervised* (only a small part of the data is labeled). It is widely used, i.e. almost all major image editing softwares (Adobe Photoshop, GIMP etc.) feature some interactive segmentation algorithms.

Like the name suggests, interactive image segmentation needs human interactions. One has to provide information on what he would like to segment, usually by drawing

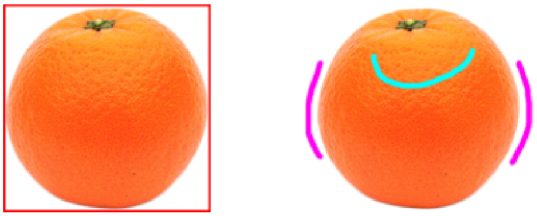


Figure 5.1 – Two ways to provide user input for an interactive image segmentation: the left image uses bounding boxes, while the right one uses scribble based tools.

rectangular bounding boxes [38, 39], scribbles [39, 77, 78], or just single dots [40] on the input image, as illustrated in Figure 5.1. The bounding box is one of the most economical in terms of user interactions. It takes only two mouse clicks and the assumption is, the desired segmentation should be close to each side of the bounding box. The segmentation algorithm then automatically assigns labels to unlabeled pixels and produces a segmentation result based on this input.

Many vision problems, including (semi-)supervised image segmentation, can be formulated using Markov Random Field (MRF). The maximizing a posteriori in an MRF (MAP-MRF) has proven to be successful for many computer vision problems. Applications include image segmentation, denoising, tracking and stereo, among others. Please see [2, 21, 22] for an overview of MRF optimization algorithms and applications in vision.

In the standard version of MRF with pairwise relations (potentials), we have an undirected graph $G = (V, E)$ and the following energy minimizing problem:

$$\mathcal{E}(\mathbf{x}) = \sum\nolimits_{p \in V} \theta_p(x_p) + \sum\nolimits_{(p,q) \in E} \theta_{pq}(x_p, x_q). \tag{5.1}$$

Here, x_p denotes the label of node $p \in V$ got assigned, which belongs to a pre-defined finite set $\mathcal{L} = [k]$ representing k classes (Recall that $[k] = \{1, \ldots, k\}$). $\theta_p(x_p)$ is usually called *unary potential* or *data term*, and is derived from the observed data (like pixels' color information), and it measures how likely label x_p fits node p. $\theta_{pq}(x_p, x_q)$ is often referred to as pairwise potential. It measures the cost of assigning labels x_p, x_q to the adjacent nodes p, q. Typically, this term is used to impose spatial smoothness and to align the solution to object boundaries. The goal is to find a labeling $\mathbf{x}$ (i.e., a mapping from V to $\mathcal{L}$) of all nodes that minimizes $\mathcal{E}(\mathbf{x})$, which corresponds to MAP-MRF.

At object boundaries, adjacent nodes often have different labels and it is crucial that $\mathcal{E}$ does not over-penalize such labellings. This requires that θ_{pq} be a nonconvex function of $|x_p - x_q|$, and this property is called *discontinuity-preserving*. Given two

classes $\alpha, \beta \in \mathcal{L}$ and a constant λ_{pq} that depends on edge (p, q), the *Potts function*

$$\theta_{pq}(\alpha, \beta) = \lambda_{pq} \cdot \mathbb{1}(\alpha \neq \beta)$$

is discontinuity-preserving and widely used in computer vision, among many other models. Recall that $\mathbb{1}(\cdot)$ is 1 if its argument is true and 0 otherwise. However, minimizing energy (5.1) with Potts function is a difficult problem ($\mathcal{NP}$-hard in general).

Because of its $\mathcal{NP}$-hardness, it is common in vision to solve the approximations of (5.1). For example, message passing algorithms [79, 80] and α-expansion [56] with guaranteed approximation ratio of 2. The corresponding condition for the later algorithm is nonnegative edge weights and the so-called *submodularity*, which can be expressed as

$$\theta_{pq}(\beta, \gamma) + \theta_{pq}(\alpha, \alpha) \leq \theta_{pq}(\beta, \alpha) + \theta_{pq}(\alpha, \gamma),$$

for all labels $\alpha, \beta, \gamma \in \mathcal{L}$.

The standard model in (5.1) imposes a limited class of constraints on the solution, since it only incorporates unary and pairwise potentials. Due to this reason, there is an ongoing research effort in computer vision towards encoding high-order constraints in MRF. These include, for example, shape compactness [81], shape convexity [82], curvature regularization [83] and label connectivity [40, 74, 75], among many other high-order priors.

In this chapter, we investigate exact connectedness priors imposed on each label, with one additional "background" label that is possible to be disconnected. More precisely, we are interested in solving (5.1) to global optimality, while adding a high-order (global) prior to (5.1) to explicitly enforce the connectivity of each label (to be made more precise in Sec. 5.3.1). A *k-label segmentation* of the image in this chapter is a partition of the corresponding graph $G(V, E)$ into k connected subgraphs $\{G_1(V_1, E_1), G_2(V_2, E_2), \ldots, G_k(V_k, E_k)\}$ such that $\cup_{i=1}^{k} V_i = V$, and $V_i \cap V_j = \emptyset$, for any $i \neq j$. Without loss of generality, we will assume that the node set V_ℓ of partition G_ℓ is assigned with label ℓ, and we will not distinguish an image segmentation from a graph partitioning.

5.1.1 Related work

Interactive image segmentation is very popular in vision, among which scribbles based user input [39, 77, 78] is widely used. Users are supposed to have some prior knowledge about the segmentation, and the provided annotation can either label certain pixels, or exclude outliers in the image.

Image segmentation under connectivity constraints has been considered in the literature [40, 75, 84, 85], but most of them only solve local or approximate solutions. In [40], a binary MRF with connectivity is considered, and a simplified version (only the user-provided pairs of nodes must be connected) of the problem is proposed. The

problem is then solved with a heuristic-based graph cut algorithm [86], obtaining connected foreground segmentation.

Exact global connectivity potentials are formulated as an ILP in [75], where the authors introduced the connected subgraph polytopes. Due to the high computational cost of solving the corresponding $\mathcal{NP}$-hard problem, the authors of [75] only examined the LP relaxations of the ILP. Although the general ILP formulation works for multi-label MRFs, the authors applied it only to binary MRF problems, and showed the LP-relaxation works well in practice (only a small portion of variables remain fractional). In [85], the authors optimized a linear (unary-potential) objective function subject to connectivity constraint in a binary (two-region) segmentation problem. It is claimed that two medical benchmark datasets are solved to optimality for the first time. The model does not consider the general multi-label pairwise MRF objective in (5.1), which is of wide interest in computer vision applications.

Finally, it is worth to mention that the subgraph connectivity problem has also attracted much attention in the operations research community. For instance, it has been applied in the forest planning problem [84], where each subregion of the forest is constrained to be connected.

5.1.2 Contribution

This chapter extends [40, 74, 75, 85] and investigates interactive multi-label MRF with connectivity constraints. To solve the resulting ILP problem to optimality, we propose a branch-and-cut method, which provably finds globally optimal solutions. This method enforces connectivity priors iteratively by a cutting plane method, where the separation problem at each iteration can be solved efficiently using a breadth-first search algorithm.

To solve the resulting $\mathcal{NP}$-hard problem more efficiently, we propose a fast simple heuristic adapted from [4] that produces very good feasible solution as an initial input for the ILP solver. The solver provides better solutions as the branch-and-cut method proceeds and a guarantee on sub-optimality (in terms of the optimality gap) even if we terminate it earlier. Unlike [74, 75], which examines LP-relaxations of the ILP, our method provides feasible solutions at any time and also global optimum guarantee. Different from [40], we consider exact and global connectivity and do not reduce the problem to connectivity between a given pair of nodes. The proposed ILP is quite general, and can be applied as a post-processing method on top of any existing multi-label segmentation approach with connectivity priors. Furthermore, it can also be used to generate ground-truth proposals for any state-of-the-art weakly supervised semantic segmentation techniques. For instance, those based on partial scribble-based annotations [87].

5.2 MRF with Pairwise Priors

We first review the standard ILP formulation of the pairwise MRF. Following the notation of [73], where $d(a, b)$ denotes any label distance on $a, b \in \mathcal{L}$, the ILP of the general pairwise MRF reads

$$\min_x \sum_{a\in\mathcal{L}}\sum_{p\in V}\theta_p(a)x_p(a) + \sum_{(p,q)\in E}\lambda_{pq}\sum_{a,b\in\mathcal{L}} d(a,b)x_{pq}(a,b) \tag{5.2}$$

$$\sum\nolimits_a x_p(a) = 1, \quad \forall p \in V, \tag{5.2a}$$

$$\sum\nolimits_a x_{pq}(a,b) = x_q(b), \quad \forall b \in \mathcal{L},\ (p,q) \in E, \tag{5.2b}$$

$$\sum\nolimits_b x_{pq}(a,b) = x_p(a), \quad \forall a \in \mathcal{L},\ (p,q) \in E, \tag{5.2c}$$

$$x_p(\cdot),\ x_{pq}(\cdot,\cdot) \in \{0,1\}, \tag{5.2d}$$

where the binary variable $x_p(a)$ indicates if vertex p is assigned label a, and $x_{pq}(a, b)$ indicates if vertices p, q are assigned label a, b. The first constraint (5.2a) express the fact that each vertex must receive exactly one label, and constraints (5.2b) and (5.2c) maintain consistency between variables $x_p(\cdot), x_q(\cdot)$ and $x_{pq}(\cdot,\cdot)$.

In this chapter, we assume the label distance function $d(a, b)$ is a Potts function, and λ_{pq} remains the same for every labels pair. Further assume $|V| = n$ and $|\mathcal{L}| = k$, then the corresponding ILP formulation (5.2) boils down to

$$\min_x\ (1-\lambda)\sum_{\ell=1}^{k}\sum_{i=1}^{n} c_i^\ell x_i^\ell + \lambda\sum_{\ell=1}^{k}\sum_{(i,j)\in E}|x_i^\ell - x_j^\ell| \tag{5.3}$$

$$\sum\nolimits_{\ell=1}^{k} x_i^\ell = 1, \quad \forall i \in [n], \tag{5.3a}$$

$$x_i^\ell \in \{0,1\}, \quad \forall i \in [n], \quad \ell \in [k], \tag{5.3b}$$

where the binary variable x_i^ℓ indicate whether node i is assigned label ℓ ($x_i^\ell = 1$ in this case), c_i^ℓ denotes the unary data term for node i assigning label ℓ, and $\lambda \in [0, 1]$ is a positive constant weighting the pairwise smoothness term. Constraint (5.3a) enforces that each node is assigned exactly one label.

The resulting problem (5.3) is nonlinear because of the absolute term $|x_i^\ell - x_j^\ell|$. Using the same trick as in Section 3.2.1, it can be transformed into an ILP. It is in general $\mathcal{NP}$-hard to solve (5.3) to optimality, hence it is common in vision to solve the corresponding LP relaxation [73, 88] by relaxing constraint (5.3b) into $x_i^\ell \in [0, 1]$. However, we are interested in solving the ILP (5.3) to global optimum.

5.3 MRF with Connectivity Priors

Apart from the standard pairwise prior, we are interested in adding a global (high-order) connectedness priors imposed on each label, to explicitly enforce the connectivity of each label (to be made more precise in Sec. 5.3.1). The mathematical programming problem of enforcing the connectivity prior is proven to be $\mathcal{NP}$-hard in [40].

5.3.1 Connected subgraph polytope

In this section we introduce the connected subgraph polytope, where a connected subgraph $G_\ell(V_\ell, E_\ell)$ is a subgraph of G where all the nodes in V_ℓ are labeled ℓ and is connected.

Connected subgraph. Without loss of generality, we assume the graph $G(V, E)$ is connected. Otherwise, we can add corresponding edges that have 0 pairwise cost, so that the energy function of (5.3) remains unchanged. We call $G_\ell(V_\ell, E_\ell)$ a connected subgraph with label ℓ if G_ℓ is connected, where $V_\ell = \{i \in V : x_i^\ell = 1\}$ and $E_\ell = \{(i,j) \in E : i,j \in V_\ell\}$. Recall from Section 2.2.2 that a subgraph $G'(V', E')$ is connected if for all pairs of nodes $i, j \in V'$, there exists a path in G' that connects i and j. We call a node $i \in V$ *active* in label ℓ if $x_i^\ell = 1$, i.e., if it is labeled ℓ.

Connected subgraph polytope. Denote $\mathbf{x}^\ell := (x_1^\ell, x_2^\ell, \ldots, x_n^\ell) \in \{0,1\}^n$, for $\ell \in [k]$, then $\mathbf{x} := (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k) \in \{0,1\}^{kn}$. Let $\mathcal{C} = \{\mathbf{x} : G_\ell(V_\ell, E_\ell) \text{ connected}, \ \forall \ell \in [k]\}$ be the finite set of labellings such that every subgraph G_ℓ is connected. Further let $\mathcal{C}_\ell = \{\mathbf{x}^\ell \in \{0,1\}^n : G_\ell(V_\ell, E_\ell) \text{ connected}\}$ denote the finite set of connected subgraphs in label ℓ. Then we call the convex hull $\text{conv}(\mathcal{C}_\ell)$ the connected subgraph polytope of label ℓ. It is shown in [40] that optimizing over this polytope is $\mathcal{NP}$-hard.

Vertex-separator set. Given a subgraph $G_\ell = (V_\ell, E_\ell)$, for any pair of active nodes $i, j \in V_\ell, \ i \neq j, \ (i,j) \notin E$, the set $S_\ell \subseteq V \setminus \{i,j\}$ is said to be a vertex-separator set with respect to vertices $\{i,j\}$ and label ℓ, if the removal of S_ℓ from G disconnects i and j in the original graph $G(V,E)$. It also means that there will exist no path between vertices i and j in the reduced graph $G' = (V \setminus S_\ell, E \setminus (S_\ell \times V))$ upon removing S_ℓ. As an additional definition, a set $\bar{S}_\ell$ is said to be a *minimal vertex-separator set* in label ℓ if it is a vertex-separator set with respect to a pair of active nodes $\{i,j\}$ in V_ℓ and any strict subset $T_\ell \subset \bar{S}_\ell$ is not.

Let $\mathcal{S}_\ell(i,j) = \{S_\ell \subset V : S_\ell \text{ is a vertex-separator set with respect to active vertices pair } \{i,j\} \text{ in } V_\ell\}$ be the collection of all $\{i,j\}$ vertex-separator sets in label ℓ, and let $\bar{\mathcal{S}}_\ell(i,j) \subset \mathcal{S}_\ell(i,j)$ be the collection of all minimal vertex-separator sets.

It is proved in [75] that $\text{conv}(\mathcal{C}_\ell)$ can be described exactly by the following set of linear inequalities:

$$x_i^\ell + x_j^\ell - 1 \leq \sum\nolimits_{s \in S_\ell} x_s^\ell, \ \ \forall i,j \in V : (i,j) \notin E, \ \forall S_\ell \in \mathcal{S}_\ell(i,j), \tag{5.4}$$

where $x_i^\ell \in \{0,1\}$, $i \in V$. In other words, if two nodes i and j are active in V_ℓ (left hand side of (5.4) becomes 1), then any vertex separator S_ℓ must contain at least one active node in V_ℓ. Otherwise, i, j cannot be connected in G_ℓ since any path in G_ℓ from i to j must pass through at least one node in S_ℓ.

In [75], the authors also prove inequalities (5.4) are facet-defining for conv($\mathcal{C}_\ell$) if $\mathcal{S}_\ell(i,j)$ is replaced by $\bar{\mathcal{S}}_\ell(i,j)$. Hence we have characterized the convex hull of the connected subgraph polytope. However, in general the number of linear constraints (5.4) is exponential with respect to $|V|$.

5.3.2 Rooted case

Following the work of [85], we assume every label has a so called *root node*. This is reasonable in our setting, since in this chapter we require the user to draw scribbles for all labels, so that at least one root node (denote r_ℓ the root node for label ℓ) is identified within each label. Then, it suffices to check connectivity of every active node x_i^ℓ to the root node r_ℓ instead of all pairs of active nodes (as in (5.4)). Thus, constraints (5.4) is reduced to

$$x_i^\ell \leq \sum\nolimits_{s \in S_\ell} x_s^\ell, \quad \forall i \in V : (i, r_\ell) \notin E, \ \forall S_\ell \in \mathcal{S}_\ell(i, r_\ell), \tag{5.5}$$

where $\mathcal{S}_\ell(i, r_\ell)$ is the set of all $\{i, r_\ell\}$ vertex-separator sets in label ℓ. In practice, the number of constraints (5.5) is still exponentially large for any label ℓ, hence cannot be considered all simultaneously for graphs of large sizes.

Theorem 5.1. *It is still $\mathcal{NP}$-hard to optimize over the connected subgraph polytope $\mathcal{C}_\ell$ even if one root node r_ℓ is given.*

Proof. Consider the problem of binary MRF with connectivity, and we only enforce the connectivity on the foreground label. This problem is proved to be $\mathcal{NP}$-hard in [40].

Suppose by fixing one root node, this problem becomes polynomial solvable. Then we can randomly assign a node $v \in V$ to be the root node r_ℓ, and solve the resulting problem in polynomial time. Since there are n ($n = |V|$) possible root nodes, by trying out all n possible root nodes in polynomial time, we are sure to find the optimal solution. Thus a contradiction. □

Despite the $\mathcal{NP}$-hardness and the need of exponentially many constraints (5.5) to exactly model conv($\mathcal{C}_\ell$), given an integer labeling x^ℓ, we can identify a subset of the violated constraints (vertex-separator sets) in polynomial time. We then iteratively add them to the ILP while searching for new integer solutions. This is known as *cut generation* in the cutting plane approach. We will look into this in detail in Sec. 5.4.2.

5.3.3 Proposed model: ILP-PC

Once we have the exact mathematical description of the connected subgraph polytope, we can model this problem exactly using mathematical programming, i.e., using MIP.

Let y_i denotes the observed image feature (e.g., gray-scaled color) at spatial location (pixel or superpixel) i. We assume the user draws k scribbles as seeds for the k labels, as shown in the left image of Fig. 5.5. We assume image observations follow a piecewise constant model within each region[1]. Let F_ℓ denotes the average color of the seeds that are label ℓ. In this case, unary potential $c_i^\ell = |y_i - F_\ell|$ represents how well label ℓ fits node i.

The MIP of the multi-label MRF with connectivity constraints becomes

$$\min_{\mathbf{x}} \ (1-\lambda)\sum_{\ell=1}^{k}\sum_{i=1}^{n} c_i^\ell x_i^\ell + \lambda \sum_{\ell=1}^{k}\sum_{(i,j)\in E} |x_i^\ell - x_j^\ell| \tag{5.6}$$

$$\sum\nolimits_{\ell=1}^{k} x_i^\ell = 1, \ \ \forall i \in [n], \tag{5.6a}$$

$$x_i^\ell = 1, \ \ \forall i \text{ within the scribble of label } \ell, \tag{5.6b}$$

$$\mathbf{x}^\ell \in \mathcal{C}_\ell, \ \ \forall \ell \in [k], \tag{5.6c}$$

$$x_i^\ell \in \{0,1\}, \ \ \forall i \in [n], \ \ell \in [k], \tag{5.6d}$$

where constraints (5.6b) denotes that the nodes of scribbles (seeds) are manually labeled by the user as hard constraints, and the connectivity constraints (5.6c) can be expressed by the rooted vertex-separator constraints (5.5). Following the trick in Section 3.2.1, we can introduce two nonnegative variables $\varepsilon_i^{\ell+}$ and $\varepsilon_i^{\ell-}$ to model $|x_i^\ell - x_j^\ell|$, and the MIP becomes an ILP.

In the case of a superpixel graph (which we adopt in the computational experiments), we represent relations between neighboring superpixels by introducing the corresponding *Region Adjacency Graph* (RAG) $G = (V, E)$, where V is the set of superpixels, and E contains edges between pairs of adjacent superpixels. We then multiply the unary and the pairwise data term in (5.6) by τ_i and γ_{ij}. Here, τ_i represents the number of pixels contained in node (superpixel) i, and γ_{ij} denotes the number of neighboring pixels between i and j.

When we deal with depth or disparity images, where the underlying assumption is piecewise affine, a few changes have to be made on the data term c_i^ℓ in formulation (5.6). In this case, y_i denotes the observed image feature (e.g., pixel's depth information), and let β^ℓ be the affine coefficients of the seeds that are labeled ℓ (i.e., we conduct an affine regression on the scribbled pixels). The unary potential c_i^ℓ then equals $|y_i - \beta^\ell z|$, where $z = (z_1, z_2, 1)$ and (z_1, z_2) present pixel's coordinates.

[1]We assume a piecewise constant model for simplicity. However, our formulation extends to any other probabilistic assumptions of observation models, e.g., piecewise linear.

5.3.4 ILP-PCB: ILP-PC with background label

In many images, it is reasonable to assume a background (not necessarily connected) exists, and the connectivity constraints can be ignored on this specific label, which we call the *background label*.

One example is shown in the left image of Fig. 5.7, where the black background is allowed to have 4 disconnected components. On the other hand, the right image of Fig. 5.7 depicts results without the background label.

5.3.5 ILP-PCO: ILP-PC with ordering prior

If one has some prior knowledge about the ordering (with respect to the number of pixels) of the label, one can add the following *ordering constaints*

$$\sum\nolimits_{i\in V} x_i^{\ell} \geq \sum\nolimits_{i\in V} x_i^{\ell+1}, \;\; \forall \ell \in [n-1]. \tag{5.7}$$

Here, one assumes that the number of pixels in segment ℓ is at least as large as that of segment $\ell+1$. In case of a superpixel graph, one needs to multiply every variable x_i by τ_i (the number of pixels in superpixel i).

These constraints are redundant if no unary data is present (un-supervised), and they impose a structure in the solution that breaks symmetries, thus enforcing an order of the segments that is generally beneficial to eliminate symmetric solutions [37]. Indeed, without constraints (5.7), two solutions obtained by swapping labels in any pair of segments ℓ, k are perfectly equivalent.

In case of a supervised MRF (5.6), it might still be beneficial, but sometimes also harmful, as will be shown in Section 5.5.

5.4 Solution Techniques

In this section, we first propose a fast greedy heuristic algorithm that can provide a feasible solution to the MIP formulation (5.6). We then introduce a strategy that finds a subset of the vertex separator sets that could be used to formulate constraints (5.5) in polynomial time. This allows us to implement the cutting plane method which grantees global optimum if no time limit is present.

5.4.1 L_0-H: a region fusion based heuristic

Although originally designed for the Potts model (unsupervised), we again adopted the idea of the heuristic from the region fusion algorithm [4]. Our algorithm works by iteratively merging groups of nodes until exactly k groups remain, and we ensure connectivity constraints at each iteration.

At the beginning, every scribble of seeds and all other nodes not covered by any scribbles are in their own groups. Then at every iteration, we merge two neighboring groups, if the following condition holds and if the merging does not result in the seeds of two different scribbles being in the same group:

$$\tau_i \tau_j |Y_i - Y_j|^2 \leq \kappa \gamma_{ij} (\tau_i + \tau_j). \tag{5.8}$$

Here τ_i denotes the number of pixels in group (segment) i, and γ_{ij} represents the number of neighboring pixels (also known as boundary length) of two groups i and j. We use Y_i to indicate the mean of image data (e.g., color) within group i, and κ to express the regularization paramter at each iteration.

By increasing κ in (5.8) in every iteration (to possibly large enough) , we terminate the algorithm when exactly k groups remain. Let *iter* be the current iteration number and η be a user-defined parameter, we adopt the following exponentially growing strategy in our algorithm:

$$\kappa = (\frac{\text{iter}}{100})^{2.2} \eta.$$

Having a good heuristic is two-folded. On the one hand, we get a good feasible solution at any time. On the other hand, this solution will be an upper bound (in a minimization problem) to the branch-and-bound method the MIP solver uses, thus helps to prune a lot of unnecessary branching nodes, and improve the running time of the solver. We will show in Section 5.5 that L_0-H is fast and generates good results most of the time, sometimes even optimal.

5.4.2 Branch and cut method: towards global optimum

The most widely used exact method for solving an ILP is branch and cut. In this section, we focus on the cutting plane method because the branch-and-bound method is implemented by default in any modern ILP solver. We are interested in exact connectivity via cutting plane method and we focus on the rooted case (5.5), since we assume at least one root node r_ℓ is fixed for each label ℓ by the user scribble.

Cutting plane method

We concentrate on enforcing the connectivity prior (5.6c) for one label only (e.g., label ℓ). Then, the same approach will be repeated for other labels until they are all connected. In the case of a background label, we ignore its connectivity and simply jump to the next label.

The basic idea is similar to that of Algorithm 2.1. When solving problem (5.6), we first omit (5.6c) (i.e., constraints (5.5), since they are exponentially many) initially and explore the branch-and-bound tree (since it is an ILP) of the system until an integer

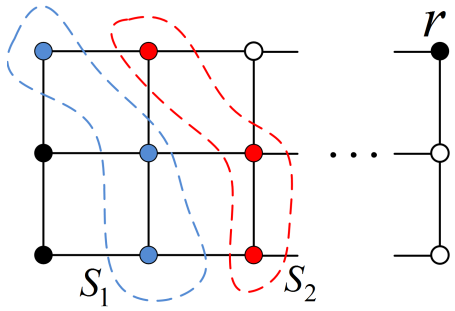


Figure 5.2 – The *K-nearest cut generation strategy*. Active nodes are shown in black, and the two separator sets are marked in red and blue.

solution is found. Then we check the feasibility of this solution (i.e., if $x^\ell \in \mathcal{C}^\ell$). If not feasible, violated constraints are identified (to be discussed in next section) and added to (5.6), to "cut off" the current infeasible solution. This procedure is repeated until $x^\ell \in \mathcal{C}^\ell$.

Algorithm 5.1 Cutting plane method for solving problem (5.6)

1: **Initialize**: Solve (5.6) without the connectivity prior (5.6c) and obtain an integer solution x
2: **while** $(\exists \ell, \text{s.t. } x^\ell \notin \mathcal{C}^\ell)$ **do**
3: Identify subsets of the vertex-separator sets $S_\ell \in \mathcal{S}_\ell$ of the current solution and add the corresponding cuts (5.5) to (5.6).
4: Solve the resulting ILP system (5.6).
5: **end while**

Here, we treat each individual connected component as one entity (see Figure 5.2 where the left two black nodes form one entity), since establishing connectivity between all nodes in one component and r_ℓ automatically connects all the nodes. Identifying violated constraints (5.6c) then boils down to finding a vertex separator set S_ℓ between each active disconnected component and the root component (which contains r_ℓ) of the current solution.

At the heart of the cutting plane technique is that only a subset of constraints (5.6c) will be needed at the optimum of (5.6). However, depending on the choice of the violated inequalities that we choose in each step, we may require a different number of them and the number of iterations will vary.

An overview of the cutting plane algorithm for solving (5.6) is listed in Algorithm 5.1.

The separation problem and cut generation strategies

So far, we still need an algorithm for the separation problem in the cutting plane method, i.e., given $x^\ell \notin \mathcal{C}^\ell$, identify subsets of the vertex-separator sets $S_\ell \in \mathcal{S}_\ell$ of the current solution. Among the many ways of selecting such cuts in the rooted case, we list two strategies among the many proposed in [75, 85]. We denote the set of active components of label ℓ as $\{H_0^\ell, H_1^\ell, \ldots\}$, where H_r^ℓ is the component that contains the root node r^ℓ.

Minimal separator strategy. Given $x^\ell \notin \mathcal{C}^\ell$ and an active components H_r^ℓ, one first construct a directed graph with duplicated vertices and edge capacities (see [75] for more details). The minimal vertex separator set S_ℓ is then obtained by solving a max-flow problem. This strategy was applied in [75].

K-nearest strategy. In the K-nearest strategy, one first runs a breath-first search on any active component H_i^ℓ to collect K (disjoint) sets composed of all nodes with identical distance. The search terminates if K equals the number of nodes in H_r^ℓ or if another active node is reached. The idea is illustrated in Figure 5.2, where the active nodes are depicted in black and r denotes the root node. The two vertex separator sets are marked in red and blue. Here $K = 2$ since it is the number of nodes in H_r.

This strategy is reported in [85] to be one of the most successful (among five) in terms of solved instances and computational efficiency. We will adopt this strategy in this chapter and integrate this into Algorithm 5.1.

5.5 Computational Experiments

In this section, we show computational experiments on medical images, where the unary potentials are based on the probability maps of each pixel belonging to given labels, which were trained using convolutional neural networks (CNN) [89]. The sizes range from 96×96 to 256×256.

We also use images from the Berkeley Segmentation Dataset [20] (BSDS500) of size 321×481 and the PASCAL VOC 2012 set [76] (PASCAL, image size around 500×400). We first apply the SLIC [15] superpixel algorithm on them to get an over-segmentation, where the number of superpixels is around 1000.

Using superpixels instead of pixels has several advantages. First, the complexity of the optimization problem (5.6) is drastically reduced with only a negligible segmentation accuracy loss (especially when the superpixels number is large). Second, the information in each superpixel is more discriminative, and also help reduce the effects of outliers. As shown in a recent survey paper for superpixel algorithms [16], a few advanced superpixel algorithms can achieve very accurate over-segmentation results with around 1000 superpixels.

List of models to be compared

We will conduct a comprehensive comparison of the following different optimization models:

- L_0-H: Our proposed L_0 region fusion based heuristic, which was motivated by [4] and we modified it to generate exactly k connected segments.
- ILP-PC. Our proposed ILP formulation (5.6) of multi-label MRF with the global connectivity constraints.
- ILP-PCB: ILP-PC with the "background" label, where this special label is not required to be connected.
- ILP-PCO: ILP-PC with the ordering constraints (5.7), where the number of pixels is constrained to not increase when the label grows.
- LP-PC: The LP relaxation of ILP-PC, which was introduced and applied in [75, 85].
- ILP-P: The ILP formulation of (5.6) without the connectivity constraints (5.6c), which is widely used in vision (e.g., graph cuts [56]).

In this section, if there is no further explanation, the default setups are the following: the pairwise term λ is 0.2, the time limit for ILP-PC and its extensions (e.g., ILP-PCB and ILP-PCO) are 100 seconds , and the L_0-H parameter η is 0.1. When we report energy E, we use the objective function of (5.6).

5.5.1 Ground-truth generation

Although our proposed ILP solver is $\mathcal{NP}$-hard, it provides globally optimal solution for the multi-label MRF with connectivity prior. Thus, it could be used to generate ground-truth labeling on the input images, which serves as a quality check for any fast algorithms.

We conduct experiments on three instances taken from the PASCAL dataset, where the scribbles of all $11k$ training images are online available and provided by the authors of [87]. We set λ equals 0.2 for the first two image and 0.3 for the third one. We use ILP-PC, and it takes only 0.08 and 0.15 second on the first two instances and 31.3 seconds on the third one to get the global optimal solution. The optimal energies are 5927.4, 12220.1 and 28238.5 respectively. The outputs are shown in Figure 5.3.

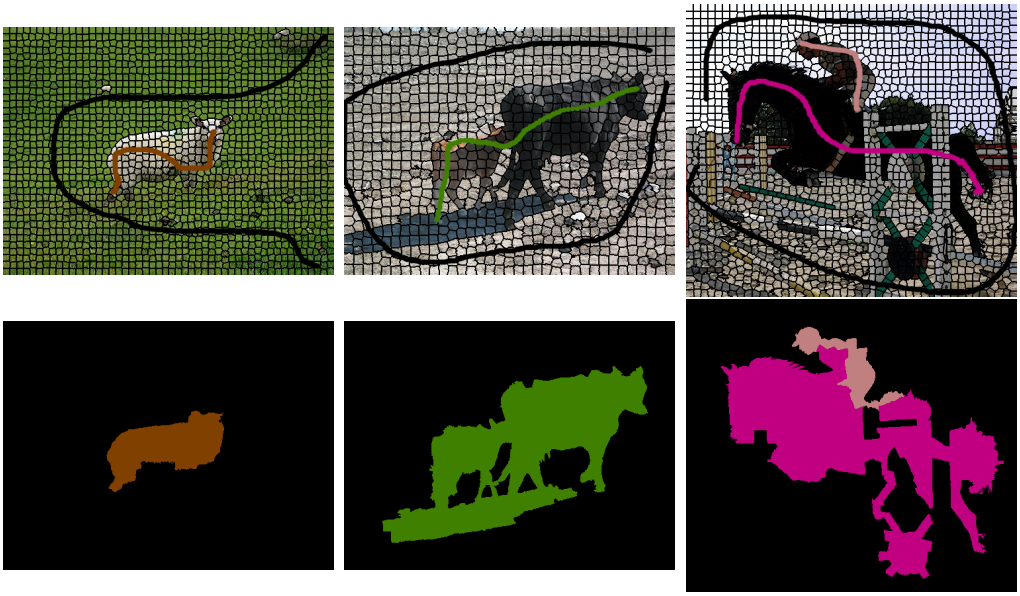


Figure 5.3 – Ground-truth generation on 3 images taken from PASCAL. Computation time $t = 0.08$, 0.15 and 31.3 secs, and energy $E = 5927.4$, 12220.1 and 28238.5.

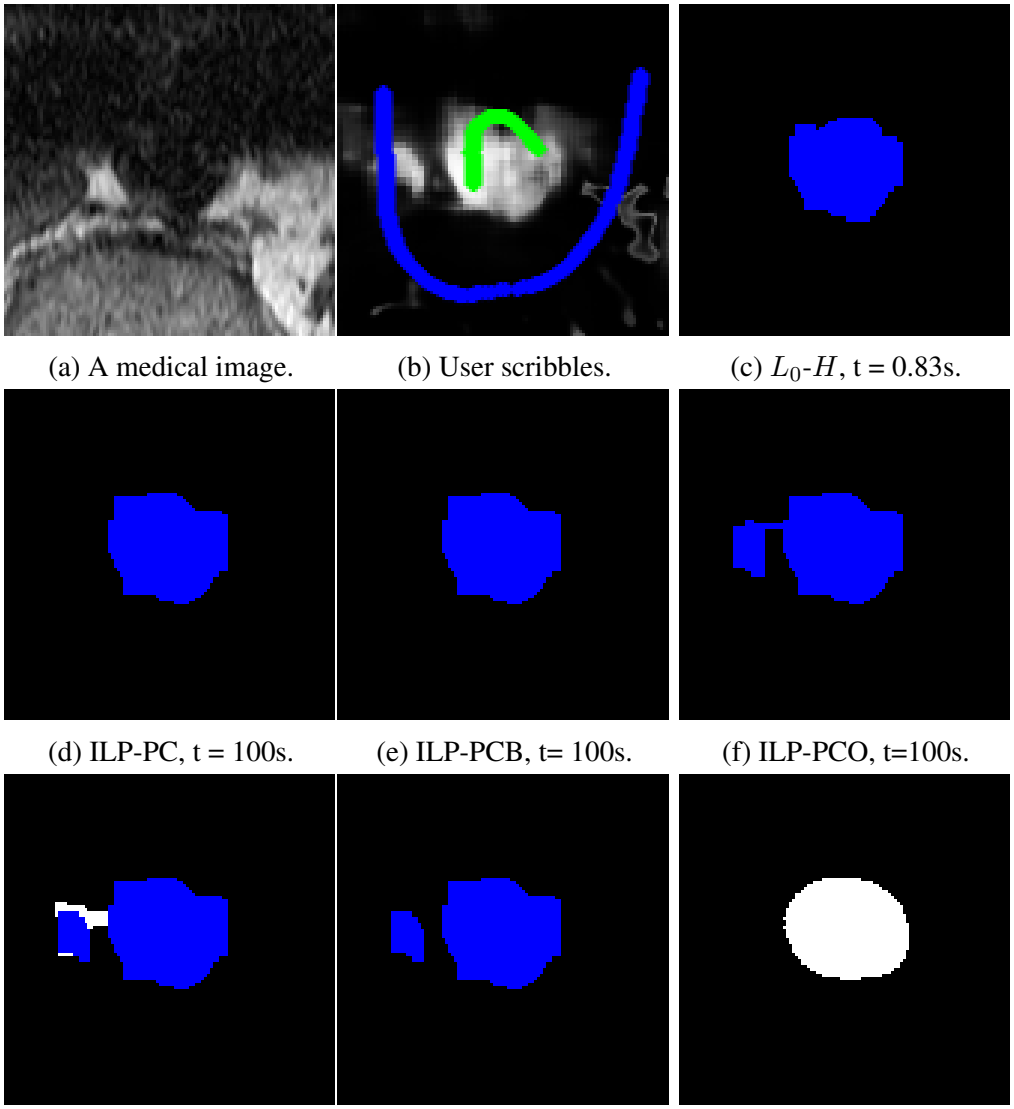
5.5.2 Detailed comparison of different models

In this section, we first report a detailed computational experiment on a medical image with probability maps using the proposed 5 models. We then compare ILP-PC with and without using the initial solution provided by L_0-H on an image from BSDS500. We finally report 3 more experiments on BSDS500 images.

Medical images with probability maps

We report a medical image segmentation example, whose unary term is based on the probability maps of each pixel belonging to different labels. The probability map was trained using convolutional neural networks (CNN) in [89]. One wants to obtain a binary (two-region) segmentation of an magnetic resonance image (MRI), which depicts the abdominal aorta [81]. In this example, the CNN probability maps yielded unsatisfying disconnected region due to imaging noises, the lack of boundary contrast and also limited training datasets.

The input image is of size 111×111, and the computation time is reported in Figure 5.4. In this example, it takes L_0-H 0.83 second to converge. Given the initial solution provided by L_0-H, ILP-PC, ILP-PCB and ILP-PCO all failed to converge within the time limit, with 2.8%, 2.8%, and 0.9% optimality gap respectively. On the other hand, LP-PC takes 1.19 seconds to solve the LP-relaxation, and it is surprising to see that ILP-P (with no connectivity prior) only takes 0.46 second to find the optimal solu-



(a) A medical image. (b) User scribbles. (c) L_0-H, t = 0.83s.

(d) ILP-PC, t = 100s. (e) ILP-PCB, t= 100s. (f) ILP-PCO, t=100s.

(g) LP-PC, t=1.19s. (h) ILP-P, t = 0.46s. (i) The ground-truth.

Figure 5.4 – Comparison of all 5 models on an MRI, where user scribble is applied on the probability map. Their energies are reported in Table. 5.1, and t denotes the time spent. In LP-PC, 0.62% of the pixels remains unlabeled (fractional) , and are colored in white. Both ILP-PC and ILP-PCB have 2.8% gap, while ILP-PCO has 0.9% , with lower energy (but not necessarily closer to the ground-truth).

tion. The solution provided by L_0-H is of high quality, only within 1.22% of the best solution found by ILP-PC in 100 seconds.

As we see in Figure 5.4, LP-PC has 0.62% fractional solution (depicted in white). Although a post-processing rounding heuristic can be applied, it is not guaranteed to

even find a feasible solution. The result of ILP-P gives two separated regions (since no connectivity prior is enforced), which is far away from the ground-truth.

The energies of all 5 models are reported in Table 5.1. We notice that both LP-PC and ILP-P give lower energy than ILP-PC, where LP-PC is the LP-relaxation and ILP-P is the relaxation on the connectivity prior. Hence, they both provide lower bounds for ILP-PC.

L_0-H	ILP-PC	ILP-PCB	ILP-PCO	LP-PC	ILP-P
864.5	854	854	839	829.5	826.8

Table 5.1 – Energy values of 5 proposed models.

The inclusion of background label is not beneficial in this example, with the same energy and optimality gap. While the ordering constraints (5.7) help find lower energy which reduces the optimality gap of ILP-PC from 2.8% to 0.9%, it does not necessarily means the solution is closer to the ground-truth. It is worth to mention that unwanted solutions like ILP-PCO in this case can be avoided by drawing slightly different scribbles.

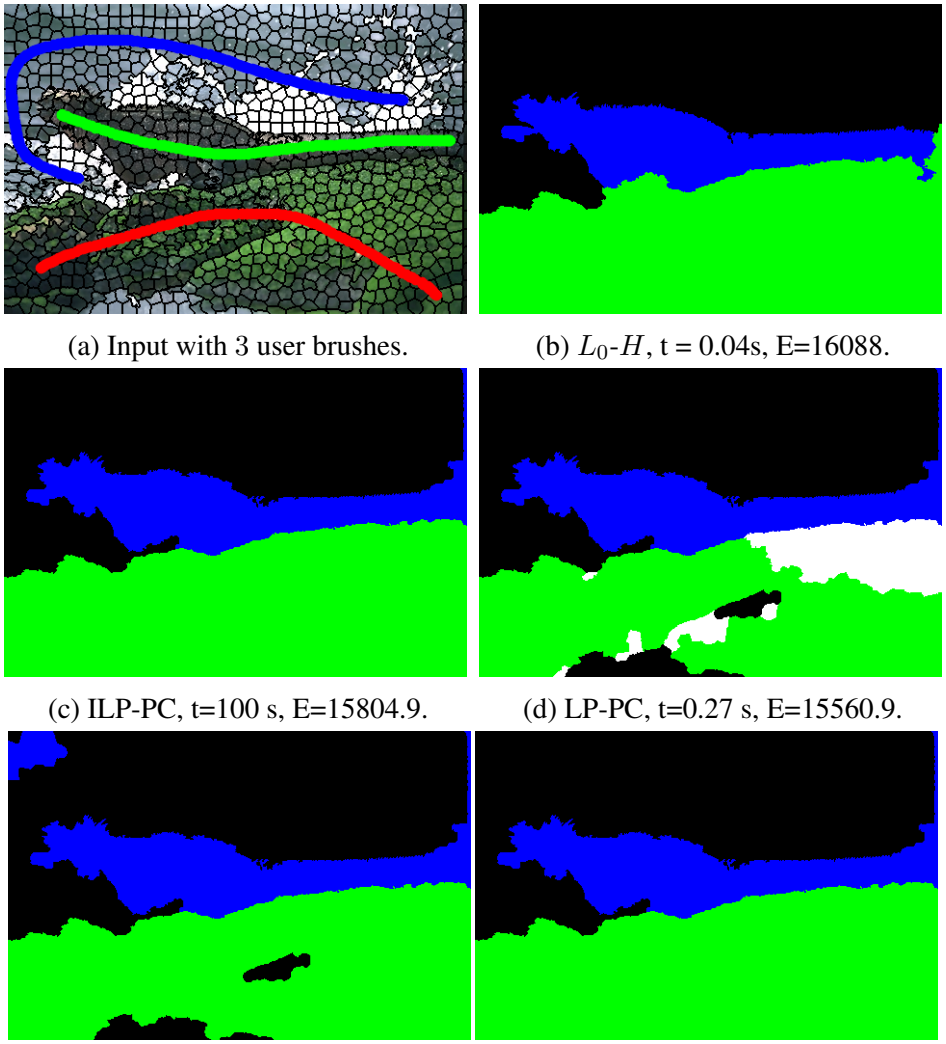
Superpixels of images from BSDS500

In this section, we introduce another model which we call ILP-PCW. It is exactly the same mathematical model as ILP-PC, but without using the initial solution provided by L_0-H. The purpose here is to test whether the ILP solver is able to achieve good results by itself.

While it does support our argument, Figure 5.5 depicts an example, where ILP-PC with L_0-H does not converge within the time limit (100 seconds) but ILP-PCW finds the provably global optimal solution in 61 seconds. Note that ILP-PC and ILP-PCW have the same energy, meaning they found exactly the same solution, but ILP-PC failed to get the tightest lower bound, with an optimality gap of 0.3%. A closer look into the log file of Cplex shows that, given the initial solution of L_0-H, ILP-PC found the "best solution" in less than 1 sec, while ILP-PCW found the same solution in 18 sec. However, ILP-PCW found the best lower bound quicker than ILP-PC.

The solution time and energies are reported in Figure 5.5. Among the 5 reported models, L_0-H is the fastest, takes only 0.04 second, followed by ILP-P and LP-PC, which takes 0.08 and 0.27 second, respectively. The energy of the starting solution provided by L_0-H is again very good, within 1.76% of the optimal solution (found by ILP-PCW).

The inclusion of the integrality constraints and the connectivity prior greatly improves solution quality. As many as 5.9% superpixel values of LP-PC are factional (depicted in white in the right figure of the second row). In the output of ILP-P, the blue and black labels have several disconnected regions, resulting in a much worse solution.



(a) Input with 3 user brushes. (b) L_0-H, t = 0.04s, E=16088.

(c) ILP-PC, t=100 s, E=15804.9. (d) LP-PC, t=0.27 s, E=15560.9.

(e) ILP-P, t=0.08s, E=15232.5. (f) ILP-PCW, t=61s, E=15804.9.

Figure 5.5 – Comparison of 4 models on an image from BSDS500, plus ILP-PCW (ILP-PC without initial solution from L_0-H). t: time spent, E: energy. Note that 5.9% of the nodes remains unlabeled in LP-PC, colored in white. The L_0-H's solution is within 1.76% of the optimal solution found by ILP-PCW.

More experiments on BSDS500 images

We report computational results and their illustrations on 4 more BSDS500 images in Figure 5.6. Note that in the first column, the pairwise term λ is set to 0.1 to encourage segmenting thin branches of the tree. All the other parameters remain at their default values.

In these experiments, we draw much fewer scribbles in the right two columns, to

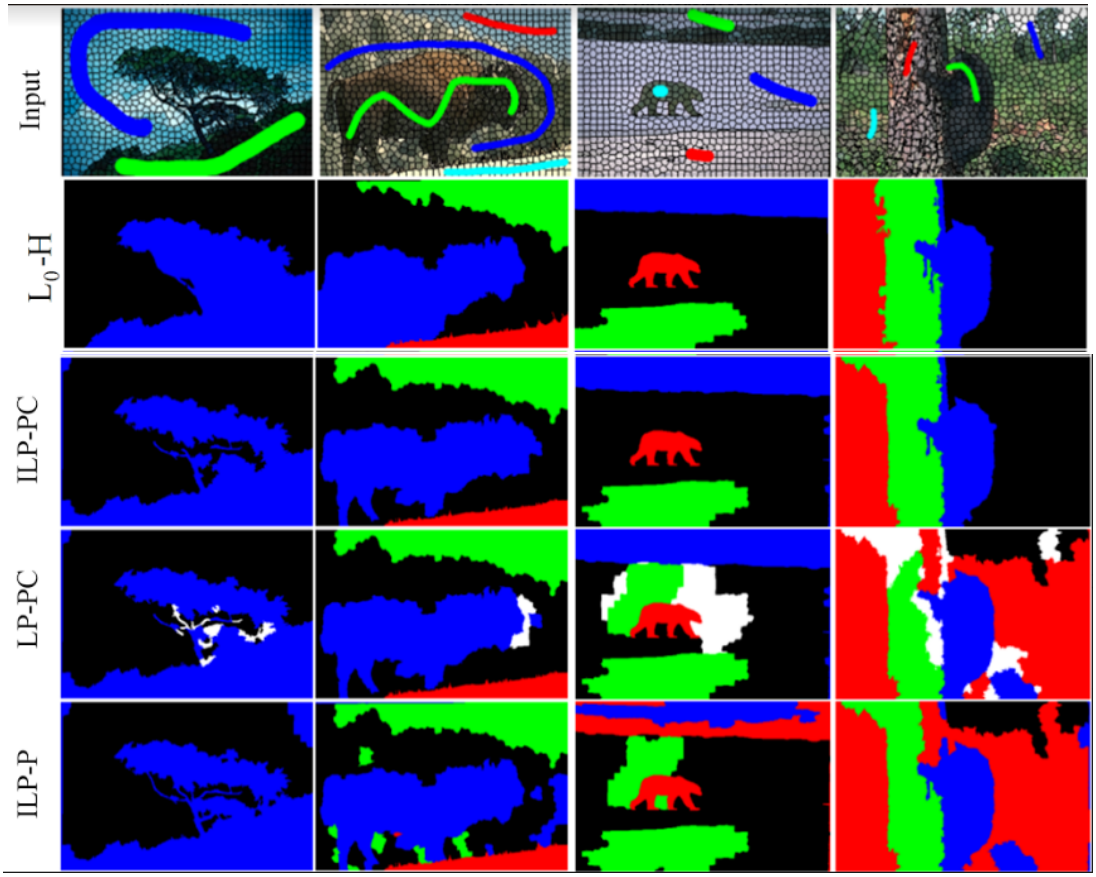


Figure 5.6 – More experiments on 4 BSDS500 images. The pairwise term λ is set to 0.1 to encourage segmenting thin branches of the tree in the first column, while all other parameters remain default. The user draws fewer scribbles in the right two columns on purpose. The while areas in LP-PC denote fractional solutions, and ILP-P is without connectivity constraints.

show the robustness of our models (to be discussed in Section 5.5.4 with more details). Note again that the white areas in the row of LP-PC denote fractional solutions, and ILP-P is without connectivity prior, thus allowing disconnected regions with the same label. We observe that L_0-H gives again quite good results in the right two images, but not so satisfying in the left two cases.

Finally, we conclude that our proposed model ILP-PC achieves the best overall results.

5.5.3 Quantitative comparison on 5 models

In this section, we conduct a detailed analysis of all the 5 models with respect to energy, computational time (optimality gap if not converge in 100 seconds time limit) and some other parameters. They are based on computational experiments of 15 images from BSDS500 and 4 real medical images with probability maps from [81].

Time, ILP optimality gap and fractional solutions of LP-PC

We report the average running time of all models in the second row of Table 5.2. If one experiment hits the time limit of 100 seconds, we just set $T = 100$ seconds. The average ILP optimality gap is shown in the third row of Table 5.2, where "Null" means no optimality gap exists (e.g., it is not an ILP).

	L_0-H	ILP-PC	ILP-PCB	ILP-PCO	LP-PC	ILP-P
Time	0.7	62.3	39.2	71.8	1.4	0.3
Gap	Null	3.7%	1.9%	2.1%	Null	0

Table 5.2 – Average time and optimality gap of 5 proposed models.

We can observe that ILP-P without connectivity prior is the fastest method (0.3 sec on average) amongst all, followed by L_0-H which takes on average 0.7 second. On the other hand, ILP-PC often hits the 100 seconds time limit and has an average running time of 62.3 seconds. The inclusion of the "background label" clearly helps on improving computational efficiency, reducing the average solving time to 39.2 seconds. However, the ILP-PCO with the ordering constraints spends 9.5 seconds more time than ILP-PC on average.

The model ILP-P is surprisingly fast and solve every instance to optimality, despite its $\mathcal{NP}$-hardness. On the other hand, it takes ILP-PC a lot of effort to enforce the connectivity prior, and given the 100 seconds time limit, the average optimality gap is 3.7%. Moreover, the inclusion of the "background label" (ILP-PCB) and ordering constraints (ILP-PCB) both help in reducing the optimal gap, to 1.9% and 2.1% respectively.

Apart from running time and optimality gap, we also report that among all the tested images, an average of 3.5% pixels found by LP-PC remain unlabeled (they have fractional solutions). Hence, we argue that LP-PC is not applicable in practice.

ILP-PC against L_0-H

In the conducted 15 experiments, L_0-H is fast to solve and could provide a feasible solution to the ILP solver as an initial solution. ILP-PC adopts this initial solution, and tries to finder better ones as the branch-and-cut tree proceeds. On average, the ILP-PC

is able to improve 6.4% solution quality based on the initial solution provided by L_0-H within 100 seconds time limit.

More experiments with 2 seconds time limit

We further conduct experiments on the same 15 instances by setting the time limit of the ILP solver to 2 seconds. L_0-H is again applied as pre-processing, thus providing an initial solution to the solver. We observe that ILP could improved 12 out of the 15 instances, and on average improve 4.4% of the given initial solutions.

Hence, we argue that our proposed ILP model can be beneficial even within very short time, and thus applicable in much wider scenarios.

ILP-PCB against ILP-PC

Based on the computational experiments, we report the gain and loss on introducing additional background label (ILP-PCB) against ILP-PC. The scores are calculated as follows.

If both ILP-PC and ILP-PCB solve the problem to optimality, we report the computational time difference. If both models failed to solve the problem within the time limit, we report the ILP optimality gap difference. In the cases where one model solves the problem to optimality while the other reaches the time limit of 100 seconds, we report the computational time difference (we treat the latter takes 100 seconds).

On 12 instances where the background label makes sense, 8 instances benefit, 3 lose while one instance remains the same. The average net gain in computation time equals 28.6 seconds, where there exist two instances that reduce the running time of 100 seconds (reaches the time limit) from ILP-PC to less than one second. The average net gain in ILP optimality gap equals 0.9%. This computational efficiency gain results from "relaxing" one connected label, thus no separation problems (5.5) need to be solved on this label.

Apart from the speed gain, it can be beneficial in practice to use ILP-PCB when a clear background with disconnected regions exists. See Figure 5.7 for an example where the images in the bottom row has a background label colored in black.

ILP-PCO against ILP-PC

We use the same methodology as above. On 10 instances where the ordering constraints (5.7) (ILP-PCO) make sense to be applied, 7 instances benefit while 3 lose. The average net loss on computation time equals 1.2 seconds, while the average net gain in ILP optimality gap is 1.6%. Although we do not observe a clear advantage of applying ILP-PCO, it often works better on hard instances (in which the ILP solver converges very slowly and hits the time limit).

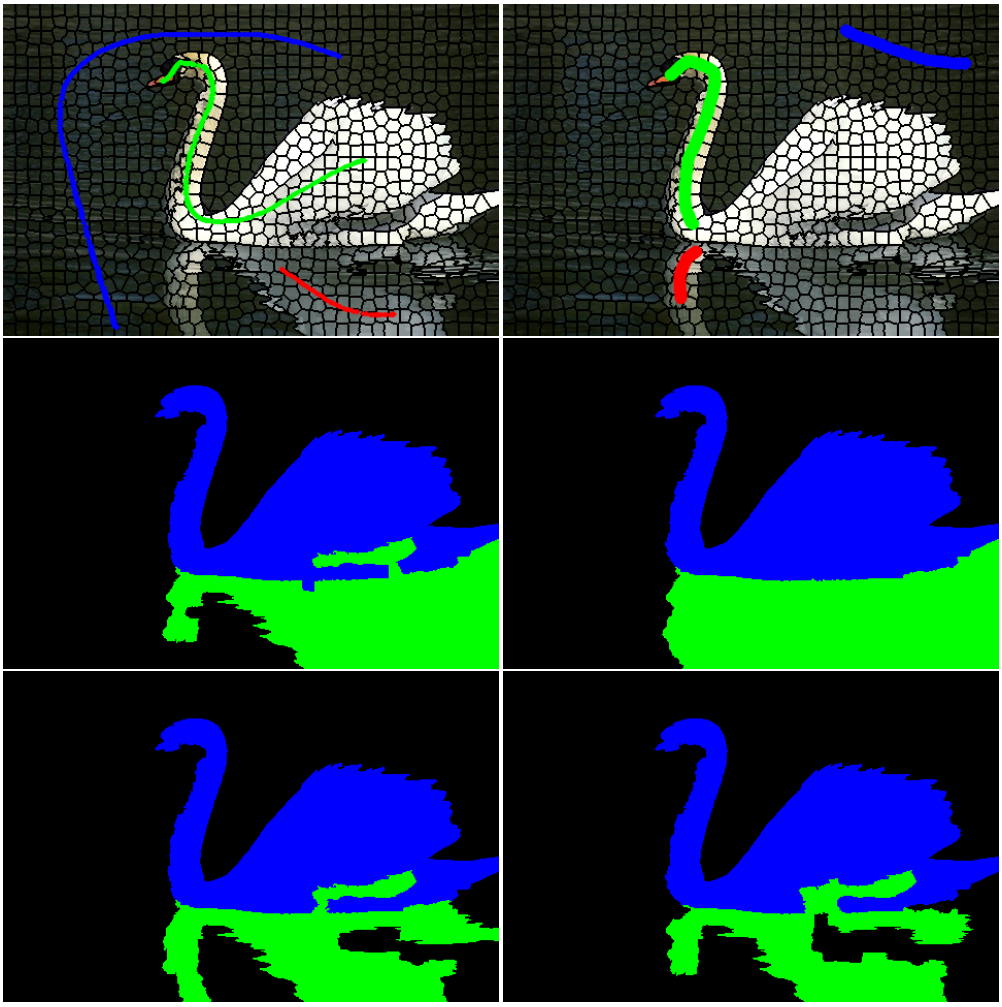


Figure 5.7 – Segmentation results with different user scribbles on the same superpixel image. Second row: ILP-PC with E=9544.2 and 13422.8, both reaching 100 seconds time limit. Bottom row: ILP-PCB with E=8615.3 and 10482.3, t= 2.1s and 0.4s. Note that the background label (shown in black) can be disconnected, while the other labels (blue and green) are enforced to be connected.

It is worth to mention that we encounter one instance where no obvious ordering exists, it took ILP-PCO 100 seconds and the solver still did not find any feasible solution. Hence it is not advised to use ILP-PCO if one is not sure about the ordering of the label.

5.5.4 Analysis on using different user scribbles

The user scribbles, on the one hand, are used to compute the average color of each label, which is used in the ILP formulation (5.6) as the unary term. On the other hand, they

also enforce hard labeling constraints (5.6b) into the ILP, which helps fix some of the binary variables. It thus helps pruning the branch-and-bound search trees within the ILP solver. Moreover, in case of difficult situations (such as outliers), scribbles can also be used to exclude outliers from one label.

We show in Figure 5.7 that changing the user scribbles does not result in significantly different segmentations, hence our model is quite robust. While ILP-PC in the second row reaches the time limit in both cases, ILP-PCB gets the reported optimal solution in only 2.1 (bottom left) and 0.4 (bottom right) seconds. The energy differences between the two cases are due to two factors: different scribbles resulted not only in different hard labeling constraints, but also different unary terms.

5.5.5 Analysis on different L_0-H parameters

L_0-H is very fast to compute, only taking 0.7 seconds on average to produce a feasible solution. It is then used as an initial solution for the ILP solver. Other than its efficiency, the quality in terms of energy is on average only 7.20% with respect to the best solution ILP-PC can find within 100 seconds time limit.

For all previous experiments reported in this chapter, the L_0-H parameter η is set to be 0.1. Figure. 5.8 shows an experiment on adopting different η. We adopt 5 different values of η, ranging from 0.1 to 0.7, with an interval of 0.15 between each pair of successive values. The computation time decreases when we increase η, reducing gradually from 0.08 to 0.05 second. One can observe from Figure. 5.8 that L_0-H is also robust to its parameter η, with only slight changes in the segmentation results.

5.6 Conclusions

In this chapter, we revisit the ILP formulation of the multi-label MRF with connectedness priors, and propose a cutting plane method to exactly enforce the connectivity constraints. A fast region fusion based heuristic is designed to provide a good initial solution for the ILP solver. Although $\mathcal{NP}$-hard to solve, the ILP formulation provides a feasible solution (often better than the initial solution) with a guarantee on the sub-optimality even if we terminate it earlier.

The ILP can also be applied as a post-processing method on top of any existing multi-label segmentation methods. It seeks for better solutions during its search in the branch-and-bound tree within the ILP solver. Furthermore, our ILP model can be applied to generate ground-truth labeling and segmentation, thus providing a quality assessment for any fast algorithms.

We demonstrated the power and usefulness of different variants of our ILP model by extensive experiments in the BSDS500, PASCAL datasets, and medical images with trained probability maps. We show that with moderate-sized images or superpixels

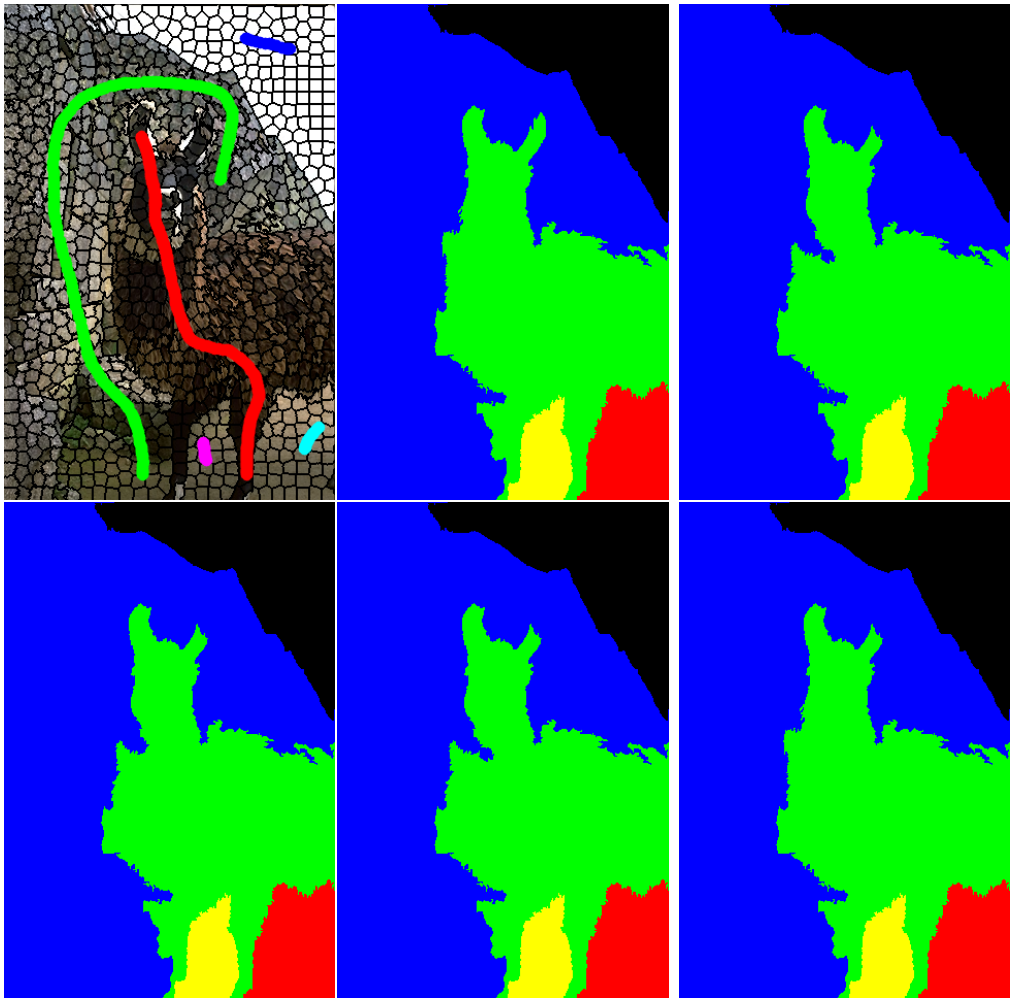


Figure 5.8 – Experiments on L_0-H with different parameter η. From top left to bottom right: superpixel image with scribbles, segmentation results with parameter η changes from 0.1 to 0.7 (with interval 0.15). The computational time reduces from 0.08 to 0.05 sec.

(thousands of them) of large ones, our proposed model achieves the best overall performance, yielding provably global optimum in some instances.

Chapter 6

Conclusions and Future Work

Recent years' hardware improvements plus the algorithmic advances in solving Mixed Integer Programming (MIP) problems have resulted in a enormous speedup in solving MIPs. In this thesis, we (re)visited the MIP formulations of three classical models, namely, the piecewise constant and affine Potts models, and the maximum a posteriori of the Markov Random Field (MRF) model.

The MIPs are based on the associated graph of the input image, and we have characterized the convex hulls of the corresponding integer polytopes in terms of facet-defining inequalities. Although the number of inequalities are exponential with respect to the number of edges (or nodes), it is polynomial to generate the separation problem at each iteration.

The proposed MIPs are in general $\mathcal{NP}$-hard to solve, and can be applied as a post-processing method on top of any algorithms. We design fast heuristic algorithms that are well suited for the above three problems, and the MIP solver provides a guarantee on the quality upon solving the linear programming relaxation. Besides, the MIP solver seeks for better solutions and can be stopped at anytime. Furthermore, the MIPs can be applied to generate ground-truth results, thus providing a quality assessment for any fast algorithms.

We have demonstrate the power and usefulness of our models by synthetic images, and instances from standard datasets. We have also compared them with some state-of-the-art methods, and showed the advantages of our methods.

In the future, we will study better separation strategies of selecting cutting planes. Besides, MRF with connectivity priors can be applied, for instance, to generate ground-truth benchmarks in the field of semantic segmentation. Furthermore, since the underlying problems of our proposed formulations are piecewise constant (affine) regression, applications beyond image processing are also of great interest.

Bibliography

[1] R. B. Potts and C. Domb. Some generalized order-disorder transformations. *Proceedings of the Cambridge Philosophical Society*, 48:106–109, 1952.

[2] R. Kindermann and J. L. Snell. *Markov random fields and their applications*. Providence, R.I. : American Mathematical Society, 1980.

[3] J. H. Kappes, M. Speth, B. Andres, G. Reinelt., and C. Schnörr. Globally optimal image partitioning by multicuts. In *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 31–44, 2011. 1.

[4] R. M. H. Nguyen and M. S. Brown. Fast and effective l0 gradient minimization by region fusion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 208–216, 2015.

[5] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[6] H. Anton and C. Rorres. *Elementary linear algebra: Applications version*. Wiley, New Jersey, 2005.

[7] G.M. Ziegler. *Lectures on Polytopes*. Springer-Verlag New York, 1995.

[8] B. Bollobás. *Modern Graph Theory*. Springer-Verlag New York, 1998.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[10] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[11] E. B. Khalil, P. L. Bodic, L. Song, G. Nemhauser, and B. Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 724–731. AAAI Press, 2016.

[12] H. L. Seal. Studies in the history of probability and statistics. xv: The historical development of the gauss linear model. *Biometrika*, 54(1-2):1–24, 1967.

[13] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[14] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 10–17, 2003.

[15] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[16] D. Stutz, A. Hermans, and B. Leibe. Superpixels: an evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, pages 1–32, 2017.

[17] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

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

[19] IBM ILOG. Cplex 12.6, https://www.ibm.com/de-en/marketplace/ibm-ilog-cplex.

[20] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[21] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1286–80, 2008.

[22] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 115(2):155–184, 2015.

[23] B. Andres, J. H. Kappes, T. Beier, U. Köthe, and F. A. Hamprecht. Probabilistic image segmentation with closedness constraints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2611–2618, 2011.

[24] A. Horňáková, J.H. Lange, and B. Andres. Analysis and optimization of graph decompositions by lifted multicuts. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1539–1548, 2017.

[25] T. Beier, F. A. Hamprecht, and J. H. Kappes. Fusion moves for correlation clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3507–3516, 2015.

[26] T. Beier, B. Andres, U. Köthe, and F. A. Hamprecht. An efficient fusion move algorithm for the minimum cost lifted multicut problem. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume LNCS 9906, pages 715–730. Springer, 2016.

[27] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV)*, pages 839–846, 1998.

[28] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising with block-matching and 3d filtering. In *Proceedings of the Electronic Imaging*, 2006.

[29] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[30] T. Chan, S. Esedoglu, F. Park, and A. Yip. Total variation image restoration: Overview and recent developments. In Nikos Paragios, Yunmei Chen, and Olivier Faugeras, editors, *Handbook of Mathematical Models in Computer Vision*, pages 17–31. "Springer", 2006.

[31] M. Storath, A. Weinmann, J. Frikel, and M. Unser. Joint image reconstruction and segmentation using the potts model. *Inverse Problems*, 31(2):025003, 2015.

[32] D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016.

[33] F. Friedrich, A. Kempe, V. Liebscher, and G. Winkler. Complexity penalized m-estimation: Fast computation. *Journal of Computational and Graphical Statistics*, 17(1):201–224, 2008.

[34] A. Weinmann, M. Storath, and L. Demaret. The l^1-potts functional for robust jump-sparse reconstruction. *SIAM Journal on Numerical Analysis*, 53(1):644–673, 2015.

[35] M. Storath, A. Weinmann, and M. Unser. Jump-penalized least absolute values estimation of scalar or circle-valued signals. *Information and Inference: A Journal of the IMA*, 6(3):225–245, 2017.

[36] P. J. Huber. The place of the l1-norm in robust estimation. *Computational Statistics & Data Analysis*, 5(4):255–262, 1987.

[37] V. Kaibel and M. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

[38] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV)*, pages 277–284, 2009.

[39] C. Rother, V. Kolmogorov, and A. Blake. Grabcut -interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (SIGGRAPH)*, pages 309–314, 2004.

[40] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[41] J.H. Kappes, M. Speth, G. Reinelt, and C. Schnörr. Higher-order segmentation via multicuts. *Computer Vision and Image Understanding*, 143:104 – 119, 2016. Inference and Learning of Graphical Models Theory and Applications in Computer Vision and Image Analysis.

[42] D. Sontag, D.K. Choe, and Y. Li. Efficiently searching for frustrated cycles in map inference. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, pages 795–804, Arlington, Virginia, United States, 2012. AUAI Press.

[43] J.K. Johnson. *Convex Relaxation Methods for Graphical Models: Lagrangian and Maximum Entropy Approaches*. PhD thesis, Cambridge, MA, USA, 2008. AAI0821536.

[44] D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for map-mrf inference: A local primal-dual gap based separation algorithm. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[45] M. Storath and A. Weinmann. Fast partitioning of vector-valued images. *SIAM J. Imaging Sci.*, 7:1826–1852, 2014.

[46] M. V. d. Bergh, X. Boix, G. Roig, B. d. Capitani, and L. V. Gool. Seeds: Superpixels extracted via energy-driven sampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 13–26, 2012.

[47] J. Yao, M. Boben, S. Fidler, and R. Urtasun. Real-time coarse-to-fine topologically preserving segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2947–2955, 2015.

[48] P. Neubert and P. Protzel. Compact watershed and preemptive slic: on improving trade-offs of superpixel segmentation algorithms. In *Proceedings of the 22nd International Conference on Pattern Recognition*, pages 996–1001, 2014.

[49] F. Meyer. Color image segmentation. *1992 International Conference on Image Processing and its Applications*, pages 303–306, 1992.

[50] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 60–65, 2005.

[51] V. Caselles, A. Chambolle, and M. Novaga. The discontinuity set of solutions of the tv denoising problem and some extensions. *Multiscale Modeling & Simulation*, 6(3):879–894, 2007.

[52] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo - stereo matching with slanted support windows. In *Proceedings of the British Machine Vision Conference*, pages 1–11, 2011.

[53] M. Lysaker and X. Tai. Iterative image restoration combining total variation minimization and a second-order functional. *International Journal of Computer Vision*, 66(1):5–18, 2006.

[54] S. Geman and D. German. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[55] J.H. Kappes, M. Speth, G. Reinelt, and C. Schnörr. Towards efficient and exact map-inference for large scale discrete computer vision problems via combinatorial optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1752–1758, 2013.

[56] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[57] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. Pure Applied Mathematics*, pages 577–685, 1989.

[58] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, USA, 1987.

[59] J. Yuan, C. Schnörr, and G. Steidl. Total-variation based piecewise affine regularization. In Xue-Cheng Tai, Knut Mørken, Marius Lysaker, and Knut-Andreas Lie, editors, *Proceedings of the Scale Space and Variational Methods in Computer Vision*, pages 552–564, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[60] K. Bredies, K. Kunisch, and T. Pock. Total generalized variation. *SIAM Journal on Imaging Sciences*, page 492–526, 2010.

[61] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[62] C. Erdogan, M. Paluri, and F. Dellaert. Planar segmentation of rgbd images using fast linear fitting and markov chain monte carlo. In *Proceedings of the Ninth Conference on Computer and Robot Vision*, pages 32–39, 2012.

[63] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using rgb-d cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317. Springer, 2012.

[64] M. Zanetti and A. Vitti. The blake-zisserman model for digital surface models segmentation. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5(2):355–360, 2013.

[65] D. Fortun, M. Storath, D. Rickert, A. Weinmann, and M. Unser. Fast Piecewise-Affine Motion Estimation Without Segmentation. *ArXiv e-prints*, 2018.

[66] A. Toriello and J. Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219:86–95, 2012.

[67] E. Amaldi, S. Coniglio, and L. Taccari. Discrete optimization methods to fit piecewise affine models to data points. *Computers and Operations Research*, pages 214–230, 2016.

[68] HCIBOX. https://hci.iwr.uni-heidelberg.de/benchmarks/the_hcibox_depth_evaluation_dataset.

[69] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 195–202, 2003.

[70] A.S. Agoes, Z. Hu, and N. Matsunaga. Dslic: A superpixel based segmentation algorithm for depth image. In Chu-Song Chen, Jiwen Lu, and Kai-Kuang Ma, editors, *Proceedings of Asian Conference on Computer Vision (ACCV)*, pages 77–87, Cham, 2017. Springer International Publishing.

[71] D. Morris, S. Imran, J. Chen, and D. Kramer. Growing depth image superpixels for foliage modeling. In *Proceedings of the 13th Conference on Computer and Robot Vision (CRV)*, pages 406–409, 2016.

[72] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 489–495, 1999.

[73] N. Komodakis and G. Tziritas. Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453, 2007.

[74] S. Nowozin and C.H. Lampert. Global interactions in random field models: A potential function ensuring connectedness. *SIAM Journal on Imaging Sciences*, 3(4):1048–1074, 2010.

[75] S. Nowozin and C. H. Lampert. Global connectivity potentials for random field models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 818–825, 2009.

[76] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2010.

[77] J. Santner, T. Pock, and H. Bischof. Interactive multi-label segmentation. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Proceedings of Asian Conference on Computer Vision (ACCV)*, pages 397–410, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[78] Y. Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 105–112, 2001.

[79] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[80] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. pages 239–269. Morgan Kaufmann Publishers Inc., 2003.

[81] J. Dolz, I. B. Ayed, and C. Desrosiers. Unbiased shape compactness for segmentation. In *Medical Image Computing and Computer Assisted Intervention - MICCAI (1)*, pages 755–763, 2017.

[82] L. Gorelick, O. Veksler, Y. Boykov, and C. Nieuwenhuis. Convexity shape prior for binary segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):258–271, 2017.

[83] C. Nieuwenhuis, E. Toeppe, L. Gorelick, O. Veksler, and Y. Boykov. Efficient squared curvature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4098–4105, 2014.

[84] R. Carvajal, M. Constantino, M. Goycoolea, J. P. Vielma, and A. Weintraub. Imposing connectivity constraints in forest planning models. *Operations Research*, pages 824–836, 2013.

[85] M. Rempfler, B. Andres, and B.H. Menze. The minimum cost connected subgraph problem in medical image analysis. In *Proceedings of the Medical Image Computing and Computer Assisted Intervention (MICCAI) (3)*, pages 397–405, 2016.

[86] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[87] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3159–3167, 2016.

[88] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552, 2011.

[89] J. Dolz, C. Desrosiers, and I. B. Ayed. 3d fully convolutional networks for subcortical segmentation in mri: A large-scale study. *NeuroImage*, pages 457–470, 2017.