# ABSTRACT

Title of dissertation: DISCRETE OPTIMIZATION METHODS FOR SEGMENTATION AND MATCHING

Ming-Yu Liu, Doctor of Philosophy, 2012

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

This dissertation studies discrete optimization methods for several computer vision problems. In the first part, a new objective function for superpixel segmentation is proposed. This objective function consists of two components: entropy rate of a random walk on a graph and a balancing term. The entropy rate favors formation of compact and homogeneous clusters, while the balancing function encourages clusters with similar sizes. I present a new graph construction for images and show that this construction induces a matroid. The segmentation is then given by the graph topology which maximizes the objective function under the matroid constraint. By exploiting submodular and monotonic properties of the objective function, I develop an efficient algorithm with a worst-case performance bound of $\frac{1}{2}$ for the superpixel segmentation problem. Extensive experiments on the Berkeley segmentation benchmark show the proposed algorithm outperforms the state of the art in all the standard evaluation metrics.

Next, I propose a video segmentation algorithm by maximizing a submodular

objective function subject to a matroid constraint. This function is similar to the standard energy function in computer vision with unary terms, pairwise terms from the Potts model, and a novel higher-order term based on appearance histograms. I show that the standard Potts model prior, which becomes non-submodular for multi-label problems, still induces a submodular function in a maximization framework. A new higher-order prior further enforces consistency in the appearance histograms both spatially and temporally across the video. The matroid constraint leads to a simple algorithm with a performance bound of $\frac{1}{2}$. A branch and bound procedure is also presented to improve the solution computed by the algorithm.

The last part of the dissertation studies the object localization problem in images given a single hand-drawn example or a gallery of shapes as the object model. Although many shape matching algorithms have been proposed for the problem, chamfer matching remains to be the preferred method when speed and robustness are considered. In this dissertation, I significantly improve the accuracy of chamfer matching while reducing the computational time from linear to sublinear (shown empirically). It is achieved by incorporating edge orientation information in the matching algorithm so the resulting cost function is piecewise smooth and the cost variation is tightly bounded. Moreover, I present a sublinear time algorithm for exact computation of the directional chamfer matching score using techniques from 3D distance transforms and directional integral images. In addition, the smooth cost function allows one to bound the cost distribution of large neighborhoods and skip the bad hypotheses. Experiments show that the proposed approach improves the speed of the original chamfer matching up to an order of 45 times, and it is

much faster than many state of art techniques while the accuracy is comparable. I further demonstrate the application of the proposed algorithm in providing seamless operation for a robotic bin picking system.

# DISCRETE OPTIMIZATION METHODS FOR SEGMENTATION AND MATCHING

by

Ming-Yu Liu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Larry Davis
Professor K. J. Ray Liu
Professor Min Wu
Professor David Jacobs

# Acknowledgments

This dissertation would not exist without the encouragement and support of many people, to whom I express my deepest gratitude.

Foremost I would like to thank my advisor Prof. Rama Chellappa who provided me the opportunity, and resources to pursue my dissertation. I have learned a lot from Prof. Chellappa about research and life. The amount of enthusiasm Prof. Chellappa shows for all tasks is unmatchable. I always felt full of energy after discussing ideas with him. I am deeply impressed by the broad perspective that Prof. Chellappa brings to research and teaching. I was fortunate to have experienced it as a graduate teaching fellow for the Image and Video Processing course in Spring 2011. I appreciated the way Prof. Chellappa seamlessly connected the theories to the applications, reached out to the class with his humor, and made the challenging material an enjoyable experience for everyone.

I am thankful to Dr. Oncel Tuzel, Dr. Ashok Veeraraghavan, and Dr. Srikumar Ramalingam for their mentorships and for providing me the opportunity to pursue challenging problems at Mitsubishi Electric Research Lab (MERL). I learned a lot from them, from analyzing a problem to designing experiments to validate one's hypothesis. I would also like to thank MERL for providing an excellent academic environment and for the financial support during my visit. I thank Prof. David Jacobs, Dr Aswin Sankaranarayanan, Dr. Amit Agrawal, Dr Yuichi Taguchi, and Dr. Tim Mark for mentoring me at various times during the course of my PhD. I also thank Dr. Xiaoqian Jiang for the intensive discussions on submodular functions

during the time we shared an apartment in Cambridge. Those discussions largely arouse my interest in discrete optimization. Thanks should go to Sofien Bouaziz and Su Yan, who gave me wonderful company during my internship at MERL.

I would also like to thank Prof. Larry Davis, Prof. K.J. Ray Liu, Prof. Min Wu, and Prof. David Jacobs for serving on my dissertation committee and providing valuable feedback. The thesis would not have taken the shape it did without the feedback, encouragement, and support of my fellow group members. I particularly thank Aswin Sankaranarayanan, Dikpal Reddy, Pavan Turaga, Kaushik Mitra, Raghuraman Gopalan, Nitesh Shroff and Ruonan Li. Things would come to a standstill at UMD without the crucial support provided by Janice Perrone, Arlene Schenk, members of ECE staff, UMIACS computing staff, and OIS staff. A big thanks to them.

During my stay at UMD I seldom felt I was away from home. Anyone would feel so with the support of wonderful friends I was fortunate to have. I thank Gimmy Liao, Yuan-Shuo Chang, Chien Min Lin, Raulf Cheng, Peggy Chuo, Yu-Hsuan Chen, Ruei-Ping Kuo, Hui-Ting Wen, Pei-Chun Chen, Yu-Han Yang, Mu-Tien Chang, Lai-Huei Wang, Cindy Wu, Chao-Wei Chen, Ming Du, Qiang Qiu, Ruonan Li, Jingjing Zheng, Jie Ni, Jun-Cheng Chen, Vicky Tu, Evelyn Cheng, Allen Chiang, Dikpal Reddy, and Balaji Vasan. I thank my roommates through the years Yi-Jung Lo, Yuan-Shuo Chang, Vicky Tu, Evelyn Cheng, Yu-Han Yang, Mu-Tien Chang, and Chih-Wei Chang for maintaining a comfortable academic environment at home.

Words will not be enough to express my gratitude to my mother, father, grandmother, aunts, and sister. They have been the foundation of my life and have

sacrificed much for me. This is as much their dissertation as it is mine.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Many tasks in computer vision can be formulated as a discrete optimization problem where one searches for the most meaningful way of assigning each image primitive a label from a finite set of labels. For example, in image segmentation, we assign each pixel a label so that the assignment best represents the scene structure. The object localization problem can be formulated as a problem of finding the maximum object evidence in a discrete space of image grids. In correspondence problems, the goal is to search for a match, which optimally relates feature points in two different images. These problems admit different structures, but they all embrace discrete optimization as a crucial component.

While this discrete optimization perspective provides a unified view of various computer vision problems, it does not lead to a unified approach. Different tasks utilize various problem structures, priors, and objective functions; they often require different discrete optimization methods. Moreover, they tend to result in NP-hard problems where a global optimum is difficult to find. Thus, further progress in computer vision hinges on the analysis of discrete optimization methods for various problem structures and on the development of effective algorithms with provable theoretical guarantees. This is precisely the goal of this dissertation.

In this dissertation, I study discrete optimization methods for several computer vision problems, including image segmentation, video segmentation, and object localization. Particular emphases are placed on the analysis of submodular functions, matroids, integral images, and their applications to the various computer vision problems. Submodular functions are related to convex functions in continuous domains but, at the same time, share similarity with concave functions. A matroid is a combinatorial structure that generalizes the concept of linear independence in vector spaces. Both submodular functions and matroid play an important part in discrete optimization problems. An integral image is an image data structure, which allows efficient evaluation of various image evidence. It finds use in improving computational efficiency for various detection algorithms. In the dissertation, I propose a novel variant of the integral image structure, which enables the integration of image evidence along various directions. Based on this structure, I develop a fast shape matching algorithm.

The main contributions of this dissertation are listed below:

- I designed submodular objective functions for superpixel and video segmentation problems.
- I showed that matroids naturally fit into the superpixel and video segmentation problems. This enables the use of algorithms developed in the discrete optimization society for solving computer vision problems.
- I showed that the Potts energy minimization problem is an instance of maximizing a submodular function subject to a matroid constraint.

- I developed an approach to improve the chamfer matching algorithm both in terms of speed and accuracy.

The dissertation has the following organization. In Chapter 2, I present a review of some necessary background material needed for the following dissertation. Chapter 3 demonstrates the application of submodular functions and matroids for data clustering and superpixel segmentation problems. In Chapter 4, a video segmentation algorithm by maximizing a submodular objective function under a matroid constraint is presented. Chapter 5 is about the fast directional chamfer matching algorithm and its application in robotic bin-picking. Chapter 6 concludes the dissertation.

# Chapter 2

# Background

In this chapter, I first define the discrete optimization problem. It is followed by a brief review to submodular functions and matroids. Most of the materials presented in this chapter are based on the discussions in [97] [78].

## 2.1 Discrete Optimization

Let $E = 1, 2, ..., n$ be a finite set of $n$ elements called the ground set. Let $2^E$ denote the power set of $E$, which is the set of all subsets of $E$. A discrete optimization problem can be formulated as a problem of

$$\max_{A \in \mathcal{I}} f(A) \tag{2.1}$$

where $f : 2^E \to \mathbb{R}$ is a real-valued objective function mapping a subset of $E$ to a real number and $\mathcal{I}$ is a collection of subsets of $E$, $\mathcal{I} \subseteq 2^E$, denoting the set of feasible subsets.

It is sometimes convenient to use a vector of Boolean variables to represent a subset $A \subseteq E$. Let $\mathbf{x} = (x_1, x_2, ..., x_{|E|})^T$ where $x_i$'s are binary variables so that

$$x_i = \begin{cases} 1 & \text{, if } \quad i \in A \\ 0 & \text{, otherwise.} \end{cases} \tag{2.2}$$

An element is in $A$ if and only if the corresponding Boolean variable has the value one. The Boolean vector $\mathbf{x}$ is called the set characteristic vector. It allows us to represent a set function as a pseudo-Boolean function, which maps a Boolean vector to a real number. The discrete optimization problem in (2.1) then can be equivalently represented by a pseudo-Boolean Optimization problem given by

$$\max_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad h(\mathbf{x}) \leq 0 \tag{2.3}$$

where the set function $h : 2^E \to \mathbb{R}$ draws the boundary of feasible subsets: $h(\mathbf{x}) \leq 0$ if and only if the subset $A$ corresponding to $\mathbf{x}$ is in the collection of feasible subsets $\mathcal{I}$.

## 2.2 Submodular Function

Let $E$ be a finite set. A set function $f : 2^E \to \mathbb{R}$ is submodular if

$$f(A_1 \cup \{a\}) - f(A_1) \geq f(A_2 \cup \{a\}) - f(A_2) \tag{2.4}$$

$$f(A_1 \cup \{a\}) - f(A_1) \leq f(A_2 \cup \{a\}) - f(A_2) \tag{2.5}$$

$$f(A_1 \cup \{a\}) - f(A_1) = f(A_2 \cup \{a\}) - f(A_2) \tag{2.6}$$

for all $A_1 \subseteq A_2 \subseteq E$ and $a \in E \setminus A$. The set function $f$ is called supermodular if the reversed inequality holds true for every pair of subsets. Finally, $f$ is called modular if it is both submodular and supermodular.

The submodularity is often referred to as the diminishing return property, which holds the gain obtained by including an element is less when included at a later

stage. Other ways to express the submodularity exist; all of them are equivalent. For example, (2.4) is equivalent to

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{2.7}$$

or

$$f(A) \leq f(B) + \sum_{a \in A \setminus B} \rho_a(B \setminus \{a\}) - \sum_{a \in B \setminus A} \rho_a(A \cup B \setminus \{a\}) \tag{2.8}$$

where $A, B$ are two subsets of $E$ and

$$\rho_a(A) \equiv f(A \cup \{a\}) - f(A) \tag{2.9}$$

is the marginal gain.

Let also define monotonicity. A set function $f$ is monotonically increasing if $f(A) \leq f(A \cup \{a\})$ for all $A \subseteq E$ and $a \in E \setminus A$. For a monotonically increasing submodular function $f$, the inequalities in (2.4) and (2.8) can be enhanced to

$$f(A_1 \cup \{a\}) - f(A_1) \geq f(A_2 \cup \{a\}) - f(A_2) \geq 0 \tag{2.10}$$

and

$$f(A) \leq f(B) + \sum_{a \in A \setminus B} \rho_a(B \setminus \{a\}) \tag{2.11}$$

respectively.

Submodular functions can be related to convex functions through Lovász extension [78]. Let $f : \{0,1\}^{|E|} \to \mathbb{R}$ be a set function. Let $P_f$ be a polyhedron associated with $f$ given by

$$P_f = \{\mathbf{w} \in \mathbb{R}^{|E|} : \sum_{i \in A} w_i \leq f(A) \quad \text{for all} \quad A \subseteq E\} \tag{2.12}$$

where $w_i$ is the $i$th entry of $\mathbf{w}$. The Lovász extension of $f$, $f^L : [0 \quad 1]^{|E|} \to \mathbb{R}$, is a convex function given by

$$f^L(\mathbf{x}) = \max_{\mathbf{w} \in P_f} \mathbf{x}^T \mathbf{w}. \tag{2.13}$$

It is shown that $f(A) = f^L(\mathbf{x}^A)$ for all $A \subseteq E$ where $\mathbf{x}^A$ is the Boolean vector with the $i$th entry equal to one if $i \in A$ and zero otherwise. Lovász further proves that

$$\min_A f(A) = \min_{\mathbf{x} \in \{0\ 1\}^{|E|}} f(\mathbf{x}) = \min_{\mathbf{x} \in [0\ 1]^{|E|}} f^L(\mathbf{x}). \tag{2.14}$$

In other words, minimizing a submodular function can be solved by minimizing a convex function.

While each submodular function induces a convex function, it is also similar to a concave function in many ways. For example, submodularity can be characterized by the monotonically decreasing property of the marginal gain function in (2.9), which is akin to the monotonically decreasing property of the derivative function of a concave function. In another example, a set function $f(|A|)$, a real-valued function defined on the cardinaity of the subset, is submodular if and only if $f$ is a concave function defined $\mathbb{R}_+$.

## 2.3 Matroid

A combinatorial structure, a matroid generalizes the concept of linear independence in vector spaces. It was first discovered by Hassler Whitney in the paper "*On the abstract prosperities of linear dependence*" in 1935. Below, I give a brief introduction to the matroid theory. I leave most of the properties and results stated

in the section unproved and refer the interested readers to the book on matroid theory [97].

### 2.3.1 Definition

A matroid $M$ is an ordered pair $(E, \mathcal{I})$ consisting of a finite ground set $E$ and a collection $\mathcal{I}$ of subsets of $E$ satisfying the following three axioms:

1. $\emptyset \in \mathcal{I}$.
2. If $I_1 \in \mathcal{I}$ and $I_2 \subseteq I_1$, then $I_2 \in \mathcal{I}$.
3. If $I_1$ and $I_2$ are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there is an element $e$ of $I_2 - I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

The third axiom is often referred as the *independence augmentation axiom*. The members of $\mathcal{I}$ are the independent sets of $M$. We shall often write $\mathcal{I}(M)$ for $I$ and $E(M)$ for $E$. Such notations are particularly helpful when several matroids are considered.

### 2.3.2 Basis

A maximal independent set in $M$ is called a basis of $M$ where the maximality is defined with respect to set inclusion. I use $\mathcal{B}(M)$ to denote the collection of bases of $M$. Bases have the same cardinality; i.e., if $B_1, B_2 \in \mathcal{B}(M)$, then $|B_1| = |B_2|$. Given $x \in B_1 \setminus B_2$, one can further show that there is an element $y$ of $B_2 \setminus B_1$ such that

$$(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}(M). \tag{2.15}$$

This condition is referred as the *basis exchange axiom* .

The collection of bases $\mathcal{B}$ provides a more elegant way of specifying a matroid $M$. Let $E$ be a finite set and $\mathcal{B}$ be a collection of subsets of $E$ satisfying:

1. $\mathcal{B}$ is non-empty.

2. The basis exchange axiom in (2.15).

Then $M = (E, \mathcal{I})$ defines a matroid where $\mathcal{I}$ is the collection of subsets of $E$ that are contained in some member of $\mathcal{B}$. Note that $\mathcal{B}$ is the collection of bases of $M$.

### 2.3.3 Circuit

A minimal dependent set in $M$ is a subset of $E(M)$ such that it is not independent and removing any element from it will result in an independent set in $M$. A minimal dependent set in $M$ is also called a circuit of $M$. We shall denote the collection of circuits of $M$ by $\mathcal{C}$ or $\mathcal{C}(M)$. Since the collection of independent sets $\mathcal{I}(M)$ can be derived from the collection of circuits $\mathcal{C}(M)$, one can define a matroid by a ground set and the collection of circuits. Specifically, let $\mathcal{C}$ be a collection of subsets of $E$ satisfying

1. $\emptyset \notin \mathcal{C}$

2. If $C_1$ and $C_2$ are members of $\mathcal{C}$ and $C_1 \subseteq C_2$, then $C_1 = C_2$.

3. If $C_1$ and $C_2$ are distinct members of $\mathcal{C}$ and $e \in C_1 \cap C_2$, then there is a member $C_3$ of $\mathcal{C}$ such that $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.

The third condition is called the *circuit elimination axiom*.

### 2.3.4 Restriction

I now define the matroid restriction operation. Let $M$ be a matroid $(E, \mathcal{I})$. If $X$ is a subset of $E$, then the pair $(X, \mathcal{I}|X)$ defines a matroid where $\mathcal{I}|X$ represents $\{I : I \subseteq X,\ I \in \mathcal{I}\}$. This matroid is called the restriction of $M$ to $X$ or the deletion of $E \setminus X$ from $M$. It is denoted by $M|X$. The collection of circuits for $M|X$ can be easily derived from $\mathcal{C}(M)$.

$$\mathcal{C}(M|X) = \{C : C \subseteq X,\ C \in \mathcal{C}(M)\} \tag{2.16}$$

### 2.3.5 Rank Function

With the matroid restriction operation, we can now derive the rank function for a matroid. A rank function $r$ of a matroid $M = (E, \mathcal{I})$ is a set function that maps a subset of $E$ to a non-negative integer: $r : 2^E \to \mathbb{Z}^+ \cup \{0\}$. Its value is given by the size of a basis $B$ of the restricted matroid $M|X$. Specifically, $r(X) = |B|$ where $B \in \mathcal{B}(M|X)$. A matroid rank function $r$ has the following properties:

1. If $X \subseteq E$, then $0 \leq r(X) \leq |X|$.

2. If $X \subseteq Y \subseteq E$, then $r(X) \leq r(Y)$.

3. If $X$ and $Y$ are subsets of $E$, then

$$r(X \cup Y) + r(X \cap Y) \leq r(X) + r(Y) \tag{2.17}$$

The third property is reminiscent of the identity

$$dim(V + W) + dim(V \cap W) = dim(V) + dim(W) \tag{2.18}$$

that holds true for vector spaces $V$ and $W$. Note well the third inequality is the submodular inequality; every matroid rank function is a submodular function!

We can specify a matroid by a rank function. Let $E$ be a set and $r$ be a function that maps $2^E$ into the set of non-negative integers and satisfies the above three properties. Let $\mathcal{I}$ be the collection of subsets $X$ of $E$ for which $r(X) = |X|$. Then $(E, \mathcal{I})$ is a matroid having rank function $r$.

### 2.3.6 Span

The matroid span operator is a generalization of the span operator in linear algebra. Let $X$ be a subset of $E$ and $r$ be the rank function of the matroid. The span operator is a function that maps a subset in $2^E$ to a subset in $2^E$ given by

$$span(X) = \{x : x \in E, r(X \cup \{x\}) = r(X)\} \tag{2.19}$$

One can show that the span function satisfies the following properties

1. If $X \subseteq E$, then $X \subseteq span(X)$.

2. If $X \subseteq Y \subseteq E$, then $span(X) \subseteq span(Y)$.

3. If $X \subseteq E$, then $span(span(X)) = span(X)$.

4. If $X \subseteq E$, $x \in E$, and $y \in span(X \cup \{x\}) - spspan(X)$, then $x \in span(X \cup \{y\})$.

Similarly, a rank function defines a matroid, so we can also define a matroid by a span function. Let $E$ be a set and *span* be a function from $2^E$ to $2^E$ satisfying the above four properties. Let

$$\mathcal{I} = \{X : X \subseteq E,\ x \notin span(X \setminus \{x\})\, \forall x \in X\}. \tag{2.20}$$

Then $(E, \mathcal{I})$ is a matroid having the span operator *span*.

## 2.4 Matroid Examples

I have briefly introduced the matroid theory. Below, several matroid examples, including vector, cycle, partition, uniform, and transversal matroids, are given.

### 2.4.1 Vector Matroids

Let $A$ be a matrix of $n$ columns, and let $E = \{1, 2, ..., n\}$ be the index set of the columns. Suppose that $\mathcal{I}$ is the collection of subsets of $E$ such that the columns referred by each member of $\mathcal{I}$ are linearly independent. Then $(E, \mathcal{I})$ defines a matroid, which is called a vector matroid. It is often denoted by $M[A]$.

**Example 2.1.** *Let $A$ be the matrix*

$$\begin{array}{ll} \textit{Column index} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ A = & \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} . \end{array} \tag{2.21}$$

*Then $(E, \mathcal{I})$ defines a matroid where*

$$E(M[A]) = \{1, 2, 3, 4, 5\}, \tag{2.22}$$

$$\mathcal{I}(M[A]) = \{\emptyset,\ \{1\},\ \{2\},\ \{4\},\ \{5\},\ \{1, 2\},\ \{1, 5\},\ \{2, 4\},\ \{2, 5\},\ \{4, 5\}\}, \tag{2.23}$$

$$\mathcal{C}(M[A]) = \{\{3\},\ \{1, 4\},\ \{1, 2, 5\},\ \{2, 4, 5\}\}. \tag{2.24}$$

### 2.4.2 Cycle Matroids

Let $E$ be the edge set of a graph $G$, and let $\mathcal{I}$ be the collection of subsets of edges having no cycles. Then $(E, \mathcal{I})$ is a matroid. It is called the cycle matroid of $G$, denoted by $M[G]$. An edge set $A \subseteq E$ is independent in $M[G]$ if and only if $G[A]$, the subgraph induced by $A$, is a forest.

**Example 2.2.** *Let $G$ be a graph consisting of three nodes and five edges as shown in Figure 2.1.*

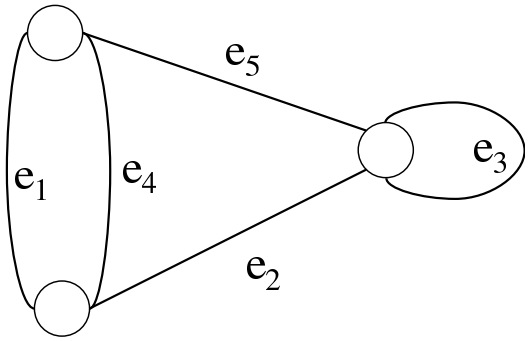



Figure 2.1: Example of a cycle matroid [97].

*Then $(E, \mathcal{I})$ defines a matroid where*

$$E(M[G]) = \{e_1, e_2, e_3, e_4, e_5\}, \tag{2.25}$$

$$\mathcal{I}(M[G]) = \{\emptyset,\ \{e_1\},\ \{e_2\},\ \{e_4\},\ \{e_5\},\ \{e_1, e_2\},\ \{e_1, e_5\},\ \{e_2, e_4\},\ \{e_2, e_5\},\ \{e_4, e_5\}\}, \tag{2.26}$$

$$\mathcal{C}(M[G]) = \{\{e_3\},\ \{e_1, e_4\},\ \{e_1, e_2, e_5\},\ \{e_2, e_4, e_5\}\}. \tag{2.27}$$

*Note well the cycle matroid $M[G]$ is isomorphic—structure preserving mapping—to the vector matroid $M[A]$ in Example 2.1.*

### 2.4.3 Transversal Matroids

Let $E$ be a finite set, and let $J = \{1, 2, ..., m\}$. A family of subsets of $E$ is a finite sequence $\mathcal{A} = (A_1, A_2, ..., A_m) = (A_j : j \in J)$ such that $A_j \subseteq E$ for all $j \in J$. Note that $A_j$s may not be disjoint. A transversal of the family $(A_j : j \in J)$ is a subset $(e_1, e_2, ..., e_m)$ of $E$ such that $e_j \in A_j$ for all $j \in J$. We say a subset $X$ of $E$ is a partial transversal of $(A_j : j \in J)$ if $X$ is a transversal of $(A_j : j \in K)$ for some subset $K$ of $J$. Let $\mathcal{I}$ be the set of partial transversals of $\mathcal{A}$. Then $M = (E, \mathcal{I})$ defines a matroid called a transversal matroid.

The transversal matroid relates to bipartite graph matching. Let $E$ and $J$ denote two sets of nodes of different types in a bipartite graph. Let $\mathcal{A} = (A_1, A_2, ..., A_m)$ be a sequence of subsets of $E$ where $A_j$ consists of nodes in $E$ that can be matched to node $j \in J$. Then, a partial matching of the bipartite graph $G = (E \times J, \mathcal{A})$ is an independent set of the transversal matroid.

**Example 2.3.** *Let* $E = \{x_1, x_2, x_3, x_4\}$, $A_1 = \{x_1, x_3\}$, *and* $A_2 = \{x_1, x_2, x_4\}$. *Then, the bipartite graph for the family of subsets* $\mathcal{A} = (A_1, A_2)$ *can be visualized in Figure 2.2.*

*Note that* $\{x_1, x_2\}$, $\{x_1, x_4\}$, $\{x_3, x_1\}$, $\{x_3, x_2\}$, *and* $\{x_3, x_4\}$ *are transversals of* $\mathcal{A}$, $\{x_1\}$ *and* $\{x_3\}$ *are partial transversals of* $\mathcal{A}$ *for subset* $\{1\}$, *and* $\{x_1\}$, $\{x_2\}$, *and* $\{x_4\}$ *are partial transversals of* $\mathcal{A}$ *for subset* $\{2\}$.

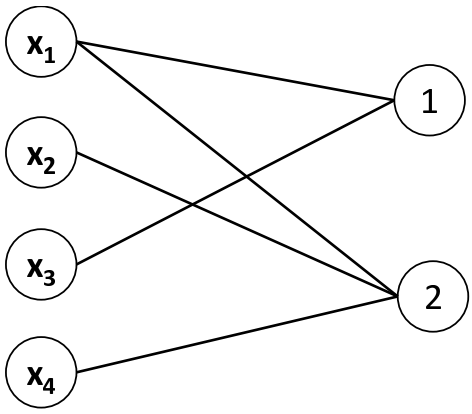



Figure 2.2: Example of a transversal matroid.

### 2.4.4 Partition Matroid

Let $\{E_1, E_2, ..., E_m\}$ be a partition of a finite set $E$ where $E_i$'s are disjoint—$E = \cup_i^m E_i$ and $E_i \cap E_j = \emptyset$ for all $i \neq j$. Suppose that $\mathcal{I}$ is a collection of subsets of $E$ satisfying the property that each member of $\mathcal{I}$ has at most $k_i$ elements from the partition $E_i$ for some nonnegative integers $k_i$'s, that is

$$\mathcal{I} = \{A \subseteq E : |A \cap E_i| \leq k_i, \forall i = 1, ..., m\}. \tag{2.28}$$

Then the ordered pair $M = (E, \mathcal{I})$ is a partition matroid.

**Example 2.4.** *Let $E$ be the product of two node sets of different types, $\{1, 2, 3\}$ and $\{x_1, x_2\}$. Let $E_1 = \{(1, x_1), (1, x_2)\}$, $E_2 = \{(2, x_1), (2, x_2)\}$, and $E_3 = \{(3, x_1), (3, x_2)\}$ be a partition of $E$. Define*

$$\mathcal{I} = \{A \subseteq E : |A \cap E_1| \leq 2, |A \cap E_2| \leq 0, |A \cap E_3| \leq 0\}. \tag{2.29}$$

*Then $M = (E, \mathcal{I})$ is a matroid with the collection of independent sets given by*

$$\mathcal{I} = \{\emptyset, \{(1, x_1)\}, \{(1, x_2)\}, \{(1, x_1), (1, x_2)\}, \}. \tag{2.30}$$

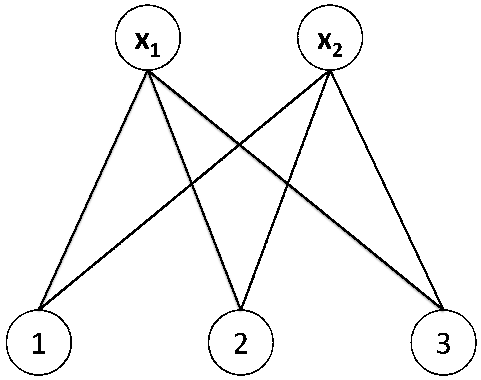



Figure 2.3: Example of a partition matroid.

*as illustrated in Figure 2.3.*

### 2.4.5 Uniform Matroid

Let $m$ and $n$ be non-negative integers such that $m \leq n$. Let $E$ be an $n$-element set and $\mathcal{B}$ be the collection of $m$-element subsets of $E$. Then it is simple to check that $\mathcal{B}$ is the set of bases of a matroid on $E$. We denote this matroid by $U_{m,n}$ and call it a uniform matroid of rank $m$ on an $n$-element set. Clearly

$$\mathcal{I}(U_{m,n}) = \{X : X \subseteq E : |X| \leq m\} \tag{2.31}$$

and

$$\mathcal{C}(U_{m,n}) = \begin{cases} \emptyset, & \text{if } m = n, \\ \{X : X \subseteq E, |X| = m+1\}, & \text{if } m < n. \end{cases} \tag{2.32}$$

# Chapter 3

# Entropy Rate Clustering: Cluster Analysis via Maximizing a Submodular Function Subject to a Matroid Constraint

## 3.1 Introduction

Clustering is a fundamental task in many domains such as machine learning, dimensionality reduction, vector quantization, computer vision, marketing, and biology. In almost every scientific field dealing with empirical data, researchers attempt to obtain a first impression on their data by identifying groups of similar characteristics. Several clustering methods have been proposed in different fields, and many of them have performed promisingly. However, many of the existing approaches are problem dependent; it is difficult to compare one criterion with another. Furthermore, most desirable criteria lead to NP-hard problems. Thus, further progress in clustering hinges on the careful design of new objective functions applicable to

existing or newer problems with provable theoretical guarantees and promising performance on standard datasets. This chapter has precisely that goal.

Among a wide variety of clustering algorithms in the literature, some compute clusters using a single objective function, some obtain clusters recursively using intermediate cost functions, and a few others identify clusters based on a particular projection (subspace, manifold) of data points. This work belongs to the first class. I formulate the clustering problem as a graph topology selection problem where data points and their pairwise relations are respectively mapped to the vertices and edges in a graph. Clustering is then solved by finding a graph topology that maximizes the objective function.

Various objective functions have been proposed to measure a given cluster' quality. However, the notion of a 'good' cluster is problem dependent. It is frequently possible to generate an example for which a given objective function fails. In this chapter, I am interested in obtaining compact, homogeneous, and balanced clusters. In a compact cluster, data points appear close to each other. In a homogeneous cluster, data points share similar inter-element properties. The notion of balanced clusters refers to avoiding large clusters that group samples aggressively. To obtain clusters with these qualities, I propose a novel objective function consisting of two components: 1.) The entropy rate of a random walk on a graph, and 2.) A balancing term on the cluster distribution. The entropy rate favors compact and homogeneous clusters where the balancing term encourages clusters with similar sizes. They are largely motivated by the principle of maximum entropy [55]: I seek a graph topology so the resulting random walk and cluster membership distribution

yields a large uncertainty.

My formulation leads to an algorithm with a performance bound on the solution. I show that the objective function is a monotonically increasing submodular function. Submodularity appears in many real world applications such as facility location, circuit design, and set covering. It can be related to convexity through the Lovász extension, while also shares some similarities to concavity [78]. Knowing whether a function is submodular enables us to gain a better understand the underlying optimization problem. In general, maximization of submodular functions leads to NP-hard problems, for which the global optimum solution is difficult to obtain. Nevertheless, by using a greedy algorithm and exploiting the matroid in my problem formulation, one obtains a bound of $\frac{1}{2}$ on the optimality of the solution. Recently, maximization of submodular functions with various constraints has been applied in several real word problem domains: sensor placement [49] (subject to a cardinality constraint), outbreak detection in networks [70] (subject to a modular cost constraint) , and word alignment [73] (subject to a matroid constraint).

The proposed algorithm is evaluated using standard datasets in the UCI repository and compared to the state-of-the-art clustering algorithms. In addition, I study a particular clustering problem in computer vision — the superpixel segmentation problem [101]. The superpixel segmentation process divides an image into disjoint and perceptually uniform regions, termed superpixels. A superpixel representation greatly reduces the number of primitives in an image and provides a coherent spatial support for feature computations. It has become a common preprocessing step for many advanced vision algorithms [87][61][51][98]. The desired properties of a

superpixel segmentation algorithm depend on the application. Some general desired properties are listed below:

- Every superpixel should overlap with only one object.
- The set of superpixel boundaries should be a superset of object boundaries.
- The mapping from pixels to superpixels should not reduce the achievable performance of the intended application.
- The above properties should be obtained with as few superpixels as possible.

I demonstrate the proposed objective function effectively models these required properties. Specifically, the entropy rate favors compact and homogeneous clusters — encouraging division of images on perceptual boundaries, whereas the balancing term encourages superpixels with similar sizes — avoiding large superpixels straddling multiple objects.

### 3.1.1 Related Work

A large body of work exists in clustering. Some well-known examples include k-means, mixture models, normalized cut [104], maximum margin clustering [128], correlation clustering [4], and affinity propagation [43]. While these algorithms encode some fundamental notions of clusters, they do not universally apply. Their performances on real datasets depend on how well the mechanism generating the dataset meets underlying assumptions. For example, when data does not exhibit a centroid-like distribution, centroid-based methods such as k-means and k-medians

are not appropriate. In order to bridge the gap, some works propose learning a proximity measure that is tuned to the assumptions of a clustering algorithm [127]. Several attempts have looked to unveil the embedded structure of data [10]. Recently, several works have adapted existing clustering algorithms to incorporate additional knowledge for improving the performance [122][136][11][115]. Below I review only a few related classes of works and refer the interested reader to survey papers such as [54][14][129][40].

#### 3.1.1.1 Graph-theoretic approaches

Cluster analysis from a graph partition perspective has a long history [133][126] [104][4]. These approaches are especially preferred when only the pairwise relations of data are available. In [133], clustering is formulated as a problem of dividing the minimal spanning tree into disjoint sets. The algorithm first constructs a minimal spanning tree from the data graph, then sequentially deletes the edges whose distances are significantly larger than those of their neighbors for partitioning the graph. The edge deletion process uses a single threshold and hence it cannot accommodate intra-cluster variation.

The graph partition problem is usually posed in an optimization framework. In a pioneering paper, Wu and Leahy [126] propose using the min-cut algorithm to bisect the graph iteratively. The min-cut cost can be solved optimally within each iteration. Nevertheless, it prefers dividing a small set of isolated vertices and is vulnerable to outliers. A seminal paper cleverly handled this drawback with

normalized cut (NCut) [104] using a normalization term. NCut relates spectral clustering [60][92]. While NCut is effective, computing an NCut solution requires eigen-decomposition, which is computationally intense for large-scale problems [130].

This work falls in the class of graph-theoretic approach. I formulate clustering as a graph topology selection problem and propose an objective function favoring the formation of compact, homogeneous, and balanced clusters. The proposed algorithm is efficient and can be easily extended to handle large datasets. In addition, it does not require an initial solution, and the computed solution is guaranteed to be within $\frac{1}{2}$ of the optimal objective value. In the discrete optimization field, my problem formulation is closely related to $K$-balanced partitioning problem [66]. Here a graph is partitioned to $K$ connected components where the number of elements in each component remains approximately the same. On the other hand, the balancing function in my problem formulation imposes a soft constraint for obtaining clusters of similar sizes.

#### 3.1.1.2 Random walk modeling

Meilă and Shi [83] discover the link between the NCut objective function and random walk models. They show that solving the NCut partition is equivalent to finding the low conductivity set in a random walk. Harel and Koren [50] propose a separation operator, based on the escape probability in a random walk, to sharpen the distinction between intra-cluster links and inter-cluster links. The operator is applied repeatedly until the graph divides into several disconnected components.

Random walk models can also benefit clustering algorithms by providing a robust distance measure. Yen et al. [132] propose a distance metric called the Euclidean commute time for improving the distance measure between data points. The Euclidean commute time is based on average passing time between two states in a random walk. To compute this metric, one needs to solve the pseudo inverse of the graph Laplacian matrix, which is computationally expensive. In [46], an interactive image segmentation algorithm based on random walk models is presented. With user-specified labels on some pixels, this algorithm computes the probabilities that a random walk reaches these labeled pixels starting at an unlabeled pixel. That pixel is then assigned the label with the largest probability. The process is repeated for all the unlabeled pixels for image segmentation.

The proposed work differs from these approaches in using the entropy rate of a random walk as a clustering objective function, which is easier to compute compared to the escape probability [50].

#### 3.1.1.3 Submodular objective functions

Submodularity has been previously exploited for clustering. Narasimhan et al. [89] present two submodular clustering objective functions. The first one is based on the minimal distance between elements of different clusters; whereas the second is related to the description length of the clusters. Clustering with these objective functions leads to submodular function minimization problems and can be solved optimally in polynomial time. On the contrary, my formulation leads to a

constrained submodular maximization problem. Recently, Jegelka and Bilmes [57] propose a submodular cost function for image segmentation named the cooperative graph cut. It gives biases to the cutting edges exhibiting cooperative patterns. To solve the cooperative cut problem, the authors derive a bounding function and show that the st-cut algorithm [18] can be used iteratively to minimize the bounds to produce the desired graph partitions.

#### 3.1.1.4 Superpixel segmentation

Graph-based image segmentation work of Felzenszwalb and Huttenlocher (FH) [36], mean shift [25], and watershed [119] presents three of the most popular superpixel segmentation algorithms. FH and watershed are extremely fast; mean shift is robust to local variations. However, they produce superpixels with irregular sizes and shapes which tend to straddle multiple objects as discussed in [71][117].

Ren and Malik [101] propose using NCut for superpixel segmentation. NCut has the nice property of producing superpixels with similar sizes and compact shapes which are preferred for some vision algorithms [101][87]. NCut has a drawback with computational requirement— it takes several minutes for segmenting an image of moderate (481x321) size. Levinshtein et al. [71] propose the TurboPixel algorithm as an efficient alternative for achieving a similar regularity. TurboPixel is based on evolving boundary curves from seeds uniformly placed in the image. Recently, Veksler et al. [117] pose the superpixel segmentation problem as a GraphCut [19] problem. The regularity is enforced through a dense patch assignment technique for

allowable pixel labels.

These methods produce good image tessellations, as shown in [101][71][117]. Nevertheless, they tend to sacrifice finer image details owing to their preference for smooth boundaries. This is reflected in the low boundary recall as reported in [71][117]. In contrast, my balancing objective, which regularizes the cluster sizes, avoids the over-smoothing problem, preserving object boundaries.

Moore et al. [85][84] propose an alternative approach for obtaining superpixels aligned with a grid. In [85], a greedy algorithm is used to cut images sequentially along some vertical and horizontal strips; whereas in [84], the problem is solved using a GraphCut algorithm [20].

Superpixel segmentation can also be jointly solved with stereo matching. Taguchi et al. [111] propose an EM-like iterative procedure to estimate scene depth and segmentation jointly using various cues. Bleyer et al. [15] pose the joint estimation problem in an energy minimization framework.

### 3.1.2 Contribution

The main contributions of this chapter are listed below:

- I pose the clustering problem as a maximization problem on a graph and present a novel objective function on the graph topology. This function consists of an entropy rate and a balancing term for obtaining clusters with desired properties.

- I prove that the entropy rate and the balancing function are monotonically

increasing and submodular.

- By embedding the problem in a matroid structure and using the properties of the objective function, I present an efficient greedy algorithm with an approximation bound of $\frac{1}{2}$.

- Using standard datasets, I demonstrate that the proposed algorithm results in improved performance in various clustering performance metrics.

- I show that the proposed algorithm significantly outperforms the state-of-the-art superpixel segmentation algorithms in the standard performance metrics on the Berkeley segmentation benchmark— a reduced undersegmentation error up to 50%, a reduced boundary miss rate up to 40%, and a tighter bound on achievable segmentation accuracy. In addition, the proposed algorithm is highly efficient— taking only about 2.5 seconds to segment an image of size 481x321.

The chapter is organized as follows. The notations and background materials are given in Section 3.2. I present the objective function in Section 3.3 and discuss its optimization in Section 3.4. Extensive experimental validations are provided in Section 3.5. I conclude and discuss some promising future research directions in Section 3.6. A preliminary version of this work appeared as a superpixel segmentation algorithm in [75]. In this chapter, I extend it for tackling the general clustering problem and provide additional experimental validation.

## 3.2 Preliminaries

**Graph representation:** I use $G = (V, E)$ to denote an undirected graph, where $V$ is the vertex set and $E$ is the edge set. The vertices and edges are denoted by $v_i$ and $e_{i,j}$ respectively. The similarity between vertices is given by the weight function $w : E \to \mathbb{R}^+ \cup \{0\}$. In an undirected graph, the edge weights are symmetric, i.e., $w_{i,j} = w_{j,i}$.

**Graph partition:** A graph partition $S$ refers to a division of the vertex set $V$ into disjoint subsets $\mathcal{S} = \{S_1, S_2, ..., S_K\}$ such that $S_i \cap S_j = \emptyset$ for $i \neq j$ and $\bigcup_i S_i = V$. I pose the graph partition problem as a subset selection problem. My goal aims to select a subset of edges $A \in E$ such that the resulting graph $(V, A)$ consists of $K$ connected components/subgraphs.

**Entropy:** The uncertainty of a random variable is measured by entropy $H$. The entropy of a discrete random variable $X$ with a probability mass function $p_X$ is defined by $H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log p_X(x)$, where $\mathcal{X}$ is the support of the random variable $X$. The conditional entropy $H(X|Y)$ quantifies the remaining uncertainty of a random variable $X$ given that the value of a correlated random variable $Y$ is known. It is defined as

$$H(X|Y) = \sum_{y \in \mathcal{Y}} p_Y(y) H(X|Y = y) = -\sum_{y \in \mathcal{Y}} p_Y(y) \sum_{x \in \mathcal{X}} p_{X|Y}(x|y) \log p_{X|Y}(x|y) \quad (3.1)$$

where $\mathcal{Y}$ is the support of $Y$, and $p_{X|Y}$ is the conditional probability mass function.

**Entropy rate:** The entropy rate quantifies the uncertainty of a stochastic process $\boldsymbol{X} = \{X_t | t \in T\}$ where $T$ is some index set. For a discrete random process,

the entropy rate is defined as an asymptotic measure

$$\mathcal{H}(\boldsymbol{X}) = \lim_{t\to\infty} H(X_t|X_{t-1}, X_{t-2}, ..., X_1), \tag{3.2}$$

which measures the remaining uncertainty of the random process after observing the past trajectory. For a stationary stochastic process, the limit in (3.2) always exists. In the case of a stationary $1st$-order Markov process, the entropy rate has a simple form $\mathcal{H}(\boldsymbol{X}) = \lim_{t\to\infty} H(X_t|X_{t-1}) = \lim_{t\to\infty} H(X_2|X_1) = H(X_2|X_1)$. The first equality is due to the $1st$-order Markov property, whereas the second equality is a consequence of stationarity. For more details, one can refer to [26, pp.77].

**Random walks on graphs:** Let $\boldsymbol{X} = \{X_t|t \in T, X_t \in V\}$ be a random walk on the graph $G = (V, E)$ with a nonnegative similarity measure $w$. I use a random walk model described in [26, pp.78]— the transition probability is defined as $p_{i,j} = Pr(X_{t+1} = v_j|X_t = v_i) = w_{i,j}/w_i$ where $w_i = \sum_{k:e_{i,k}\in E} w_{i,k}$ is the sum of incident weights of the vertex $v_i$, and the stationary distribution is given by

$$\boldsymbol{\mu} = (\mu_1, \mu_2, ..., \mu_{|V|})^T = (\frac{w_1}{w_T}, \frac{w_2}{w_T}, ..., \frac{w_{|V|}}{w_T})^T \tag{3.3}$$

where $w_T = \sum_{i=1}^{|V|} w_i$ is the normalization constant. For a disconnected graph, the stationary distribution is not unique. However, $\boldsymbol{\mu}$ in (3.3) is always a stationary distribution. It can be easily verified through $\boldsymbol{\mu} = P^T\boldsymbol{\mu}$ where $P = [p]_{i,j}$ is the transition matrix. The entropy rate of the random walk can be computed by applying (3.1)

$$\begin{aligned}\mathcal{H}(\boldsymbol{X}) = H(X_2|X_1) &= \sum_i \mu_i H(X_2|X_1 = v_i) \\ &= -\sum_i \sum_j \frac{w_{i,j}}{w_T} \log \frac{w_{i,j}}{w_T} + \sum_i \frac{w_i}{w_T} \log \frac{w_i}{w_T}\end{aligned} \tag{3.4}$$

**Submodularity:** Let $E$ be a finite set. A set function $F : 2^E \rightarrow \mathbb{R}$ is submodular if

$$F(A \cup \{a_1\}) - F(A) \geq F(A \cup \{a_1, a_2\}) - F(A \cup \{a_2\})$$

$$\text{or, equivalently,} \quad \delta F_{a_1}(A) \geq \delta F_{a_1}(A \cup \{a_2\}) \tag{3.5}$$

for all $A \subseteq E$ and $a_1, a_2 \in E \setminus A$ where $\delta F_{a_1}(A) \equiv F(A \cup \{a_1\}) - F(A)$ is the marginal gain obtained by adding the element $a_1$ to the set $A$. This property is also referred to as the diminishing return property, which says that the gain of a module is lesser if used in a later stage [91].

## 3.3 Problem Formulation

Clustering is viewed as a graph partitioning problem. To partition the graph into $K$ clusters, one searches for a graph topology that has $K$ connected subgraphs and maximizes the proposed objective function.

### 3.3.1 Graph Construction

I map a dataset to a graph $G = (V, E)$ with vertices denoting the data points, and the edge weights denoting the pairwise similarities given in the form of a similarity matrix. One can generating such a mapping in many ways. Some examples include the fully-connected graph, a local fixed-grid graph, or a nearest-neighbor graph. The proper choice of the graph structure itself presents an important problem in clustering [88]; however, it is not the focus of the dissertation. I simply map a

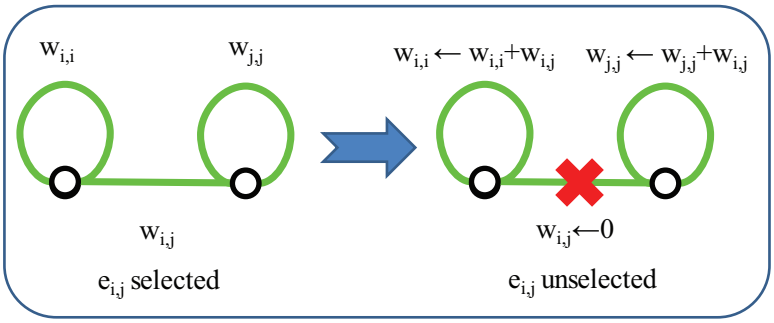



Figure 3.1: Illustration of the graph construction. If an edge $e_{i,j}$ is unselected in cluster formation, its weight is redistributed to the loops of the two vertices.

dataset into a $k$-nearest neighbor graph for clustering. For superpixel segmentation, I exploit the image grid structure and use an 8-connected graph.

My goal aims to partition the graph into disjoint components. It is achieved by selecting a subset of edges $A \subseteq E$ such that the resulting graph, $G = (V, A)$, contains exactly $K$ connected subgraphs. In addition, I also assume that every vertex of the graph has a self-loop. Although the self-loops do not affect graph partitioning, they are necessary for the proposed random walk model. When an edge is not included in $A$, I increase the edge weight of the self-loop of the associated vertices so that the total incident weight for each vertex remains constant (See Figure 3.1).

### 3.3.2 Entropy Rate

I use the entropy rate of the random walk on the constructed graph as a criterion to obtain compact and homogeneous clusters. The proposed construction leaves the stationary distribution of the random walk (3.3) unchanged where the set

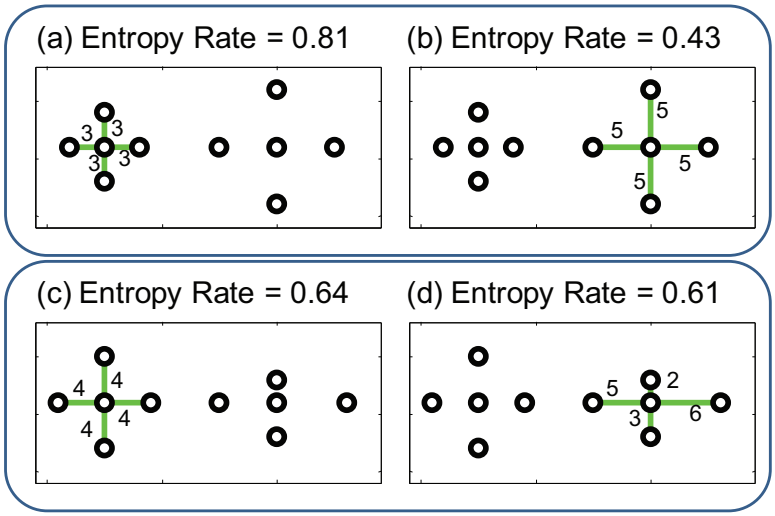



Figure 3.2: I show the role of entropy rate in obtaining compact and homogeneous clusters. I use a Gaussian kernel to convert the distances, the numbers next to the edges, to similarities. Each of these clustering outputs contains six different clusters shown as connected components. As described in Section 3.3, every vertex has a loop that is not shown. The entropy rate of the compact cluster in (a) has a higher objective value than that of the less compact one in (b). The entropy rate of the homogeneous cluster in (c) has a higher objective value than the less homogeneous one does in (d).

functions for the transition probabilities $p_{i,j} : 2^E \to \mathbb{R}$ are given below:

$$p_{i,j}(A) = \begin{cases} \frac{w_{i,j}}{w_i} & \text{if } i \neq j \text{ and } e_{i,j} \in A, \\ 0 & \text{if } i \neq j \text{ and } e_{i,j} \notin A, \\ 1 - \frac{\sum_{j:e_{i,j}\in A} w_{i,j}}{w_i} & \text{if } i = j. \end{cases} \tag{3.6}$$

Consequently, the entropy rate of the random walk on $G = (V, A)$ can be written as a set function:

$$\mathcal{H}(A) = -\sum_i \mu_i \sum_j p_{i,j}(A) \log(p_{i,j}(A)) \tag{3.7}$$

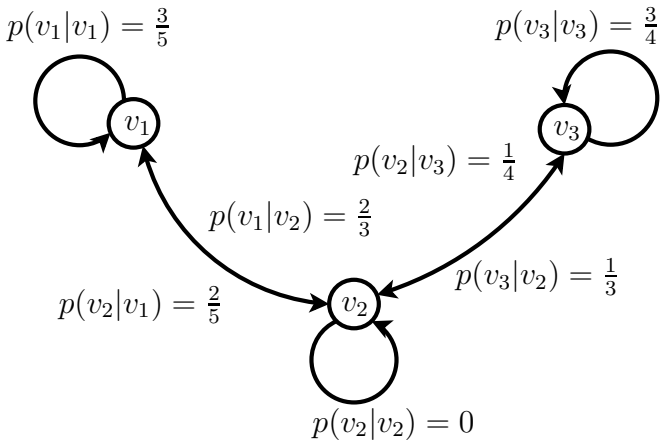



Figure 3.3: Illustration of the transition probabilities, given selecting edges $e_{1,2}$ and $e_{2,3}$.

Although inclusion of any edge in set $A$ increases the entropy rate, this increase is larger when selecting edges that form compact and homogeneous clusters, as shown in Figure 3.2.

I illustrate the computation of the entropy rate under the proposed graph construction using the following example, which is also shown in Figure 3.3. Given a graph with three vertices $\{v_1, v_2, v_3\}$ and the input similarity matrix

$$W = \begin{pmatrix} - & 2.0 & 3.0 \\ 2.0 & - & 1.0 \\ 3.0 & 1.0 & - \end{pmatrix} \tag{3.8}$$

the task is to compute the entropy rate, $\mathcal{H}(\{e_{1,2} \cup e_{2,3}\})$; i.e., the entropy rate of the random walk when selecting the edges $e_{1,2}$ and $e_{2,3}$ (as shown in Figure 3.3). From (3.3) the stationary distribution of the random walk equals $\boldsymbol{\mu} = (\frac{5}{12}, \frac{3}{12}, \frac{4}{12})^T$,

and the transition matrix takes the following form

$$P = \begin{pmatrix} \frac{3}{5} & \frac{2}{5} & 0 \\ \frac{2}{3} & 0 & \frac{1}{3} \\ 0 & \frac{1}{4} & \frac{3}{4} \end{pmatrix}. \tag{3.9}$$

The entropy rate then equals $\mathcal{H}(\{e_{1,2} \cup e_{2,3}\}) = 0.905$.

I establish the following result on the entropy rate of the random walk model.

**Theorem 3.1.** *The entropy rate of the random walk on the graph $\mathcal{H} : 2^E \to \mathbb{R}$ is a monotonically increasing submodular function under the proposed graph construction.*

Clearly, the entropy rate is monotonically increasing, given the inclusion of any edge increases the uncertainty of a jump of the random walk. The diminishing return property stems from the fact that the increase in uncertainty from selecting an edge is less in a later stage because it is shared with more edges. The proof is given in Section 3.7.

### 3.3.3 Balancing Function

I utilize a balancing function that encourages clusters with similar sizes. Let $A$ be the selected edge set, $N_A$ be the number of connected components in the graph, and $Z_A$ be the distribution of the cluster membership. For instance, let the graph partitioning for the edge set $A$ be $\mathcal{S}_A = \{S_1, S_2, ..., S_{N_A}\}$. Then the distribution of $Z_A$ is equal to

$$p_{Z_A}(i) = \frac{|S_i|}{|V|}, i = \{1, ..., N_A\}, \tag{3.10}$$

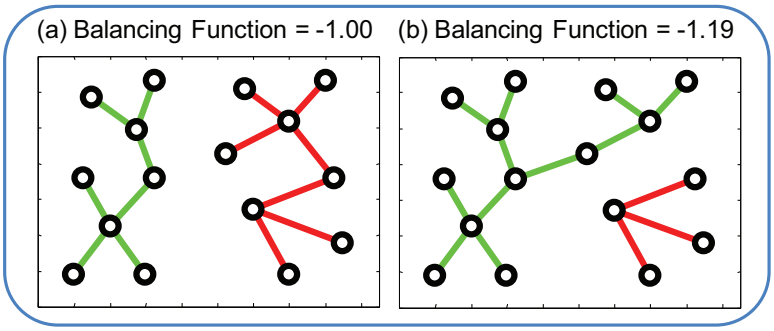



Figure 3.4: I show the role of the balancing function in obtaining clusters of similar sizes. The connected components show the different clusters. The balancing function has a higher objective value for the balanced clustering in (a) compared to the less balanced one in (b).

and the balancing term is given by

$$\mathcal{B}(A) \equiv H(Z_A) - N_A = -\sum_i p_{Z_A}(i) \log(p_{Z_A}(i)) - N_A. \tag{3.11}$$

The entropy $H(Z_A)$ favors clusters with similar sizes; whereas $N_A$ favors fewer number of clusters. In Figure 3.4, I show an example of this preference in which a more balanced partitioning is preferred for a fixed number of clusters.

Similar to the entropy rate, the balancing function is a monotonically increasing and submodular function, as stated in the following theorem:

**Theorem 3.2.** *The balancing function $\mathcal{B} : 2^E \to \mathbb{R}$ is a monotonically increasing submodular function under the proposed graph construction.*

The proof is given in Section 3.7.

The objective function combines the entropy rate and the balancing function, and therefore favors compact, homogeneous, and balanced clusters. Clustering is

achieved via optimizing the objective function with respect to the edge set:

$$\begin{aligned} \max_{A} \quad & \mathcal{H}(A) + \lambda \mathcal{B}(A) \\ \text{subject to} \quad & A \subseteq E \text{ and } N_A \geq K, \end{aligned} \tag{3.12}$$

where $\lambda \geq 0$ is the weight of the balancing term. Linear combination with non-negative coefficients preserves submodularity and monotonicity [91], therefore the objective function is also submodular and monotonically increasing. The additional constraint on the number of connected subgraphs enforces exactly $K$ clusters since the objective function is monotonically increasing.

The proposed formulation is closely related to the principle of maximum entropy, which states that the probability distribution that best represents ones' knowledge of the underlying problem has the largest entropy. This distribution makes the minimal assumption of the problem and is the least biased one [55]. My objective function encourages a graph partition such that the random walk in the graph has a large entropy rate and the cluster membership distribution has a large entropy.

## 3.4 Optimization and Implementation

In this section, I present a greedy optimization scheme, its efficient implementation for the proposed objective function, and analyze its optimality and complexity.

### 3.4.1 Greedy Heuristic

One standard approach for maximizing a submodular function works with a greedy algorithm [91]. The algorithm starts with an empty set (a fully disconnected graph, $A = \emptyset$) and sequentially adds edges to the set. At each iteration, it adds the edge that yields the largest gain. The iterations cease when the number of connected subgraphs reaches a preset number, $N_A = K$.

To achieve additional speedups, I place an additional constraint on the edge set $A$ such that it does not include cycles. This constraint immediately ignores additional edges within a connected subgraph and reduces the number of evaluations in the greedy search. Note: these edges do not change the partitioning of the graph. Although this constraint leads to a smaller solution space (only tree-structured subgraphs are allowed) compared to the original problem, in practice the clustering results are similar.

This cycle-free constraint together with the cluster number constraint $N_A \geq K$ leads to an independent set definition, which induces a matroid $M = (E, \mathcal{I})$. I establish this in the following theorem:

**Theorem 3.3.** *Let $E$ be the edge set, and let $\mathcal{I}$ be the set of subsets $A \subseteq E$ which satisfies: 1.) $A$ is cycle-free and 2.) $A$ constitutes a graph partition with more than or equal to $K$ connected components. Then the pair $M = (E, \mathcal{I})$ is a matroid.*

The proof is given in Section 3.7.

The graph partition problem is then posed as a submodular function maxi-

**Algorithm 1:** Pseudocode of the greedy algorithm. The objective function is defined as $\mathcal{F} \equiv \mathcal{H} + \lambda\mathcal{B}$.

**Data**: $G = (V, E)$, $w : E \to \mathbb{R}^+$, $K$, and $\lambda$

**Result**: $A$

$A \leftarrow \emptyset$, $U \leftarrow E$

**repeat**

$$\hat{a} \leftarrow \underset{a \in U}{arg\max} \quad \mathcal{F}(A \cup \{a\}) - \mathcal{F}(A)$$

**if** $A \cup \{\hat{a}\} \in \mathcal{I}$ **then**

$A \leftarrow A \cup \{\hat{a}\}$

**end**

$U \leftarrow U - \{\hat{a}\}$

**until** $U = \emptyset$

mization problem subject to a matroid constraint:

$$\begin{aligned} \max_{A} \quad & \mathcal{H}(A) + \lambda\mathcal{B}(A) \\ \text{subject to} \quad & A \in \mathcal{I}. \end{aligned} \tag{3.13}$$

Maximization of a submodular function subject to a matroid constraint has been an active subject in combinatorial optimization; it is shown in Fisher et al. [42] that the greedy algorithm gives an $\frac{1}{2}$ approximation bound. Following the same argument, I achieve the same ($\frac{1}{2}$ approximation) guarantee on the proposed greedy algorithm. A pseudocode is given in Algorithm 1.

### 3.4.2 Efficient Implementation

In each iteration, the greedy algorithm selects the edge that yields the largest gain in the objective function subject to the matroid constraint. A naive implementation of the algorithm, as given in Algorithm 1, loops $O(|E|)$ times to add a new edge into $A$. At each loop, it scans through the edge list to locate the edge with the largest gain; therefore the complexity of the algorithm is $O(|E|^2)$.[1] Since each vertex in the graph connects a constant number of few neighbors, the complexity of the algorithm is $O(|V|^2)$. By exploiting the submodularity of the objective function, I can achieve a more efficient implementation that is called lazy greedy [70].

Initially, I compute the gain of adding each edge to $A$ and construct a max heap structure. At each iteration, the edge with the maximum gain is popped from the heap and included with $A$. The inclusion of this edge affects the gains of some of the remaining edges in the heap; therefore, the heap needs to be updated. However, the submodular property allows an efficient update of the heap structure. The key observation is that, throughout the algorithm, the gain for each edge can never increase — a diminishing return property. Therefore, it is sufficient to retain a heap structure where the gain of the top element is updated, but not necessarily the others. Since the top element of the heap is updated and the values for the other elements can only decrease, the top element is the maximum value.

Although the worst case complexity of the lazy greedy algorithm is $O(|V|^2 \log |V|)$, in practice the algorithm runs much faster than the naive implementation. On aver-

[1] Note that an edge gain can be computed in constant time.

age, few updates are performed on the heap at each iteration, hence the complexity of the algorithm approximates $O(|V|\log|V|)$. In my superpixel segmentation experiments, it provides a speedup by a factor of 200–300 for image size 481x321, and on average requires 2.5 seconds.

I present a method to adjust the balancing weight $\lambda$ automatically. Given an initial user-specified value $\lambda'$, the final balancing parameter $\lambda$ is adjusted based on: 1.) The number of clusters, $K$ and 2.) The data dependent dynamic parameter, $\beta$. The cluster number $K$ emphasizes the balancing term more when given a large number of clusters is required. The data dependent term is computed from the input data. It is given by the ratio of the maximum entropy rate increase and the maximum balancing term increase upon including a single edge into the graph $\beta = \frac{\max_{e_{i,j}} \mathcal{H}(e_{i,j}) - \mathcal{H}(\emptyset)}{\max_{e_{i,j}} \mathcal{B}(e_{i,j}) - \mathcal{B}(\emptyset)}$. This choice has the effect of compensating for the magnitude difference between the two terms in the objective function. The final balancing parameter is given by $\lambda = \beta K \lambda'$.

## 3.5 Experiments

I conducted extensive experiments on clustering and superpixel segmentation to evaluate the proposed algorithm. Throughout the experiments, $\lambda' = 0.5$ is used to determine the balancing weight.

Table 3.1: Clustering performance comparison: clustering accuracy.

| Dataset | Proposed | NCut | AP | K-means | CPMMC |
|---|---|---|---|---|---|
| Ionosphere | **92.59** | 83.19 | 70.94 | 70.00 | 75.48 |
| Letters | 94.45 | 94.28 | 91.83 | 93.38 | **95.02** |
| Satellite | **99.51** | 97.50 | 62.30 | 94.10 | 98.79 |
| Digits 0689 | **98.24** | 91.83 | 90.31 | 78.46 | 96.74 |
| Digits 1279 | **95.97** | 91.70 | 85.51 | 89.32 | 94.52 |
| Breast Cancers | 92.97 | 92.09 | **93.32** | 91.04 | n/a |
| Iris | **94.00** | 86.67 | 86.00 | 83.33 | n/a |
| Wine | 96.63 | **98.31** | 93.82 | 96.63 | n/a |
| Glass | 50.93 | **55.14** | 40.19 | 45.33 | n/a |
| Movement Libras | **53.06** | 50.83 | 46.94 | 44.44 | n/a |
| Natural Scenes | 47.36 | **56.36** | 43.64 | 47.70 | n/a |
| MPEG-7 Shapes | **74.00** | 71.64 | 69.14 | n/a | n/a |

Table 3.2: Clustering performance comparison: rand index.

| Dataset | Proposed | NCut | AP | K-means | CPMMC |
|---|---|---|---|---|---|
| Ionosphere | **0.86** | 0.72 | 0.59 | 0.58 | 0.65 |
| Letters | 0.90 | 0.89 | 0.85 | 0.88 | **0.92** |
| Satellite | **0.99** | 0.95 | 0.53 | 0.89 | 0.97 |
| Digits 0689 | **0.98** | 0.93 | 0.92 | 0.87 | 0.97 |
| Digits 1279 | **0.96** | 0.92 | 0.87 | 0.90 | **0.96** |
| Breast Cancers | 0.87 | 0.85 | **0.88** | 0.84 | n/a |
| Iris | **0.93** | 0.86 | 0.85 | 0.83 | n/a |
| Wine | 0.96 | **0.98** | 0.92 | 0.95 | n/a |
| Glass | **0.73** | 0.70 | 0.66 | 0.70 | n/a |
| Movement Libras | **0.92** | **0.92** | 0.91 | 0.91 | n/a |
| Natural Scenes | 0.82 | **0.84** | 0.81 | 0.83 | n/a |
| MPEG-7 Shapes | **0.99** | **0.99** | **0.99** | n/a | n/a |

Table 3.3: Clustering performance comparison: performance rank averages in clustering accuracy and rand index.

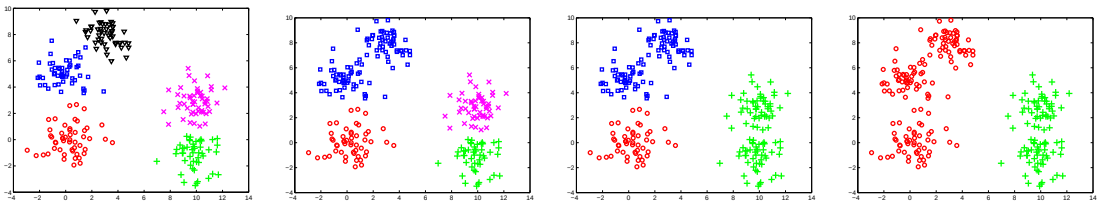
| Algorithm | Proposed | NCut | AP | K-means | CPMMC |
|---|---|---|---|---|---|
| CA | **1.5** | 2.2 | 3.8 | 3.7 | 2.0 |
| RI | **1.4** | 2.1 | 3.6 | 3.6 | 1.8 |

Figure 3.5: Example images from the natural scene recognition dataset [95]. From left to right, the image classes are coast, forest, highway, inside city, mountain, open country, street, and tall building. The images of the same class exhibit great variation due to different imaging conditions such as locations and seasons.


Figure 3.6: Example silhouettes from the MPEG-7 shape dataset [56]. The dataset contains 70 different shape classes and each class has 20 instances in various deformation and articulation. I show four instances for apple, device, elephant, ray, and octopus classes.

### 3.5.1 Clustering experiments

I conducted clustering experiments using both standard and challenging vision datasets. They include the ionosphere, letters, satellite, digits, breast cancers, iris, wine, glass, and movement libras datasets from the UCI repository. In the preprocessing step, the samples were normalized to have a zero mean and unit variance for each feature dimension. To measure the distance between the samples, the Euclidean distance is used. Two vision datasets were also used for performance eval-



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

Figure 3.7: I show the intermediate results of dichotomizing a dataset consisting of five Gaussian clouds. After the first few iterations, I recover the 5 Gaussian clouds in different clusters in (a). The subsequent combinations results in 4 , 3 , and 2 clusters, as shown in (b), (c), and (d) respectively.

uation: the natural scene recognition dataset [95] and the MPEG-7 shape database (MPEG-7) [56]. The natural scene dataset contains images from eight different nature scenes ranging from coast, forest, highway, inside city, mountain, open country, street, to tall building. Some of the images are shown in Figure 3.5. This dataset present many challenges: images of the same scene usually differ greatly due to the various locations and seasons when they were captured, while images of different scenes can be very similar due to the common spatial layout. In order to measure the pairwise similarity, I used GIST descriptors [95], the spatial envelope of the image. I used the Euclidean distance in the GIST descriptor space as the distance measure. The MPEG-7 datasets contains 1,400 shapes evenly distributed among 70 object classes. Some of the shapes appear in Figure 3.6. Samples in the dataset exhibit great intra-class variations including deformation and articulation. I applied the inner distance shape context (IDSC) algorithm [74] to compensate for the intra-class variations.

The proposed algorithm requires pairwise similarity scores as inputs. I use a Gaussian kernel, given by $w(v_i, v_j) = exp(-\frac{d(v_i,v_j)^2}{2\sigma^2})$, to convert the above distance measures to similarity scores, where $d(v_i, v_j)$ is the pairwise distance between samples $i$ and $j$ and $\sigma$ is the kernel bandwidth. I then construct a neighbor graph where each sample is connected to its 30 nearest neighbors prior to clustering.

In the experiments, I set the number of clusters equal to the true number $K$ for all the algorithms. For comparison, I use two standard clustering performance metrics: 1) clustering accuracy and 2) rand index.

- **Clustering accuracy (CA)** is a classification-accuracy like performance metric. Let $\mathcal{C} = \{C_1, C_2, ..., C_K\}$ be the ground truth distributions of clusters where $C_i$ is the set of indices for samples in the $i$th cluster. Similarly, let $\mathcal{S} = \{S_1, S_2, ..., S_K\}$ be the computed cluster distribution with $S_i$ denoting the index set of samples assigned to the $i$th cluster. The clustering accuracy is given by

$$CA = \max_J \frac{1}{n} \sum_i |C_i \cap S_{J(i)}| \tag{3.14}$$

  where $n$ is the total number of samples in the dataset, and $J$ represents any possible permutation of the sequence $\{1, 2, ..., K\}$. Equation (3.14) requires searching for the best permutation solved using the Hungarian algorithm.

- **Rand index (RI)** measures similarity between two clusterings: the ground truth and the estimated. Let $TP$ be the number of sample pairs in the same cluster for both of the ground truth and the estimated clusterings, $TN$ be the number of sample pairs in different clusters for the ground truth and the

estimated clusterings, $FP$ be the number of sample pairs in different clusters for the ground truth clustering but are in the same cluster for the estimated clustering, and $FN$ be the number of sample pairs that are in the same cluster for the ground truth clustering but in different clusters for the estimated clustering. In other words, $TP$, $TN$, $FP$, and $FN$ correspond to the counts of true positive, true negative, false positive, and false negative sample pairs, respectively. The rand index is given by percentage of agreed cluster assignment

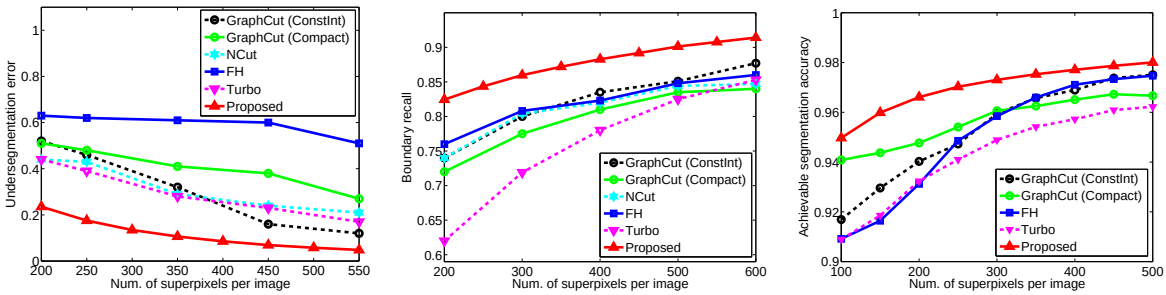
$$RI = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3.15}$$

I compare my results with state-of-the-art clustering algorithms including AP [43], K-means, NCut [104], and the cutting plane maximum margin clustering algorithm (CPMMC) [123]. They represent a variety of clustering methods from example-based, centroid-based, graph-theoretic, to maximum margin-based methods. For AP, the performance parameters implicitly control the number of clusters; a binary search on the parameter value is performed to obtain the output with the desired number of clusters. I used the implementation available from the author's website. The K-means algorithm is sensitive to initialization. I initialized the K-means algorithm with 100 different configurations using the implementation available in MATLAB, and report the best performances achieved. Both the NCut algorithm and the proposed algorithm have a kernel bandwidth parameter. Following the setup in [123], I exhaustively searched a range of the parameter values and report the best performance obtained for each of the algorithms. Specifically, I

computed the minimum and the maximum distance for all the sample pairs prior to clustering. The kernel bandwidth values were then varied from 20% of the minimum distance to the maximum distance linearly in 240 steps. I used the implementation of NCut available provided in [104]. The performance numbers of the CPMMC algorithm were duplicated from a recent paper [123]. The results in clustering accuracy and rand index appear in Table 3.1 and Table 3.2 respectively.

From Tables 3.1 and 3.2, I note that the proposed algorithm produces slightly better performances in clustering. It outperforms the competing algorithms in 7 of the 12 datasets according to the clustering accuracy measure. I also achieve better performance under the rand index criteria: better in 8 of the 12 datasets. For the two challenging vision datasets, all the algorithms did not perform well. This is due mainly to the insufficiency of the descriptors in modeling the intra-class and inter-class variations of the datasets. I obtain a better clustering accuracy for the MPEG-7 shape dataset, despite my results being inferior to NCut in the natural scene clustering task. I summarize the performances using their average performance ranks and present them in Table 3.3. The proposed algorithm has an average performance rank of 1.5 and 1.4 in terms of clustering accuracy and rand index, which performs significantly superior to all the other algorithms.

In the next experiment, I demonstrate the agglomerative nature of the proposed clustering algorithm to dichotomize a dataset consisting of five Gaussian clouds as shown Figure 3.7. Clearly, it first discovers the five Gaussian clouds in Figure 3.7(a) and subsequently combines proximate ones until the number of remaining clusters equal to two, as shown in Figure 3.7(b)(c)(d). The agglomerative property




(a) (b) (c)

Figure 3.8: Performance metrics: (a) undersegmentation error curves, (b) boundary recall curves, and (c) achievable segmentation accuracy curves. The proposed algorithm performs significantly better than the state-of-the-art in all the metrics at all the superpixel counts.

is useful for identifying the internal structure of dataset and scientific visualization.

### 3.5.2 Superpixel segmentation experiments

I conducted experiments on superpixel segmentation using the Berkeley segmentation benchmark [82]. The benchmark contains 300 grey images with human-labeled ground truths. To compute the pairwise similarity between neighboring pixels, I adopt the function $\exp(-\frac{(\|p-q\|_2|I(p)-I(q)|)^2}{2\sigma^2})$ where $p$ and $q$ are pixel coordinates, $\|p-q\|_2$ is their $L_2$ distance, and $|I(p)-I(q)|$ is the absolute value of their intensity difference. The kernel bandwidth is set to $\sigma = 5.0$ throughout the superpixel segmentation experiments.

Superpixel segmentation has a different goal than object segmentation, therefore the performance metrics also differ. I computed three standard metrics that are commonly used for evaluating the quality of superpixels: undersegmentation

Figure 3.9: Superpixel segmentation examples. The images contain 100 superpixels. The ground truth segments are color-coded and blended on the images. The superpixels (boundaries shown in white) respect object boundaries and tend to divide an image into similar-sized regions.

Figure 3.10: Nonphotorealistic rendering using superpixels. The images are divided into 150 superpixels, and each pixel is colored by the average color of the superpixel it belongs to. The balanced-size objective renders an artistic effect capturing the style of thick application of paintbrush common in post-impressionism.

error [71][117], boundary recall [101] and achievable segmentation accuracy [94]. I first describe these metrics for the sake of completeness. I use $\mathcal{G} = \{G_1, G_2, ..., G_{n_\mathcal{G}}\}$ to represent a ground truth segmentation with $n_G$ segments and $|G_i|$ denotes the segment size.

- **Undersegmentation error (UE)** measures fraction of pixel leak across

ground truth boundaries. It evaluates the quality of segmentation based on the requirement that a superpixel should overlap with only one object. I utilize the undersegmentation error metric used in Veksler et al. [117],

$$UE_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_i \sum_{k:S_k \cap G_i \neq \emptyset} |S_k - G_i|}{\sum_i |G_i|}. \tag{3.16}$$

For each ground truth segment $G_i$ I find the overlapping superpixels $S_k$'s and compute the size of the pixel leaks $|S_k - G_i|$'s. I then sum the pixel leaks over all the segments and normalize it by the image size $\sum_i |G_i|$.

- **Boundary recall (BR)** measures the percentage of the natural boundaries recovered by the superpixel boundaries. I compute BR using

$$BR_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_{p \in \delta\mathcal{G}} \mathbb{I}(\min_{q \in \delta\mathcal{S}} \|p - q\| < \epsilon)}{|\delta\mathcal{G}|}, \tag{3.17}$$

which is the ratio of ground truth boundaries that have a nearest superpixel boundary within an $\epsilon$-pixel distance. I use $\delta\mathcal{S}$ and $\delta\mathcal{G}$ to denote the union sets of superpixel boundaries and ground truth boundaries respectively. The indicator function $\mathbb{I}$ checks if the nearest pixel is within $\epsilon$ distance. In my experiments, I set $\epsilon = 2$.

- **Achievable segmentation accuracy (ASA)** is a performance upperbound measure. It gives the highest accuracy achievable for object segmentation that utilizes superpixels as units. To compute ASA, I label each superpixel with the label of the ground truth segment that has the largest overlap. The fraction of correctly labeled pixels is the achievable accuracy,

$$ASA_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_k \max_i |S_k \cap G_i|}{\sum_i |G_i|}. \tag{3.18}$$

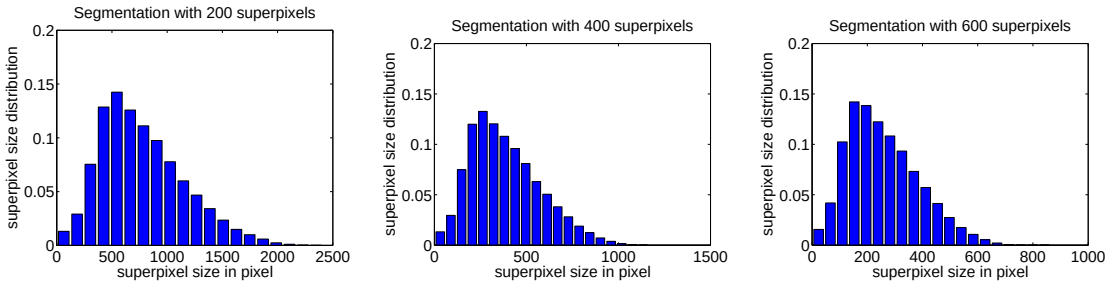



Figure 3.11: Superpixel size distribution. I plot the distributions on superpixel sizes obtained by segmenting the image into (a) 200 superpixels, (b) 400 superpixels, and (c) 600 superpixels. Each of the distributions has a bell shape. The proposed algorithm divides the images into similar-sized regions and avoids producing superpixels with small or large spatial support.

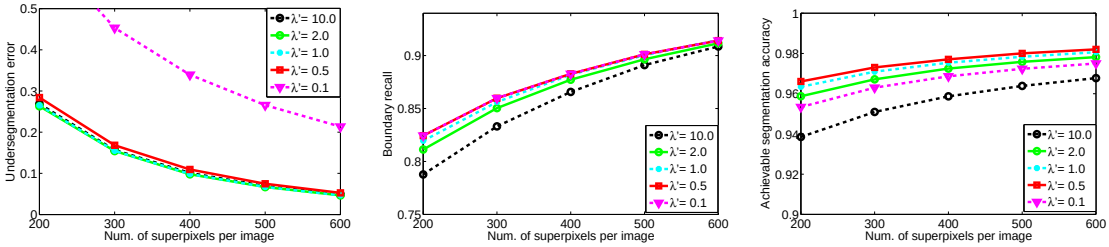



Figure 3.12: Effect of the balancing preference on the performance metrics.

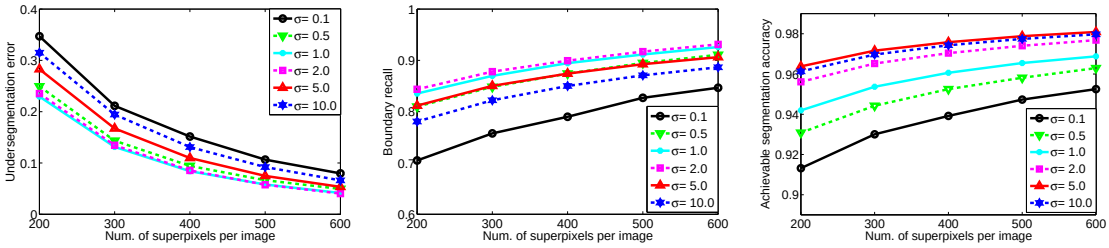



Figure 3.13: Effect of the kernel bandwidth on the performance metrics.

These performance metrics are plotted against the number of superpixels in an image. Algorithms producing better performances with a smaller number of superpixels are preferable.

In the first experiment, I compare my results with FH [36], GraphCut su-

perpixel [117], Turbopixels [71] and NCut superpixel [101] methods using the three evaluation metrics. The results were obtained by averaging all the 300 gray images in the dataset.

Figure 3.8(a) shows the undersegmentation error curves, and the curves for the other methods are duplicates from the original paper [117]. The proposed algorithm outperforms the state-of-the-art at all the superpixel counts where the error rate is reduced by more than 50%. It achieves an undersegmentation error of 0.13 with 350 superpixels while the same performance is achieved with 550 superpixels using GraphCut superpixel segmentation [117]. With 550 superpixels, my undersegmentation error is 0.06.

In Figure 3.8(b), I plot the boundary recall curves. Again, the curves for the other methods are duplicated from the original paper [117]. The proposed algorithm reduces the missed boundaries by more than 30% compared to the state-of-the-art at all the superpixel counts. The recall rates of the presented algorithm are 82% and 92% with 200 and 600 superpixels, respectively. The recall rates with the same superpixel counts are 76% and 86% with FH.

In Figure 3.8(c), I plot the achievable segmentation accuracy curves. In this experiment, I generated the curves for the other methods using the original implementations. The proposed algorithm yields a better achievable segmentation upperbound at all the superpixel counts—particularly for smaller number of superpixels. The ASA is 95% with 100 superpixels where the same accuracy can only be achieved with 200 superpixels for the other algorithms.

In the second experiment, I evaluate the segmentation results visually. Several

examples are shown in Figure 3.9 where the images are partitioned into 100 superpixels. For better visualization, the ground truth segments are color-coded and blended on the images, and the superpixel boundaries recovered by the algorithm are superimposed in white. It is difficult to notice pixel leaks, and the superpixels tend to divide an image into similar-sized regions that are important for region based feature descriptors.

In the third experiment, I illustrate the application of the proposed algorithm for nonphotorealistic rendering. Several examples appear in Figure 3.10, which are computed by first dividing the images into 150 superpixels and coloring each pixel by the average color of the superpixel to which it belongs. Apparently, similar effects can be achieved by other image smoothing techniques, the proposed algorithm renders similar-sized segments and the smoothing effect captures the style of thick application of paintbrush — a style popular in post-impressionism.

In Figure 3.11 I plot the distributions on superpixel size. I applied the proposed algorithm to segment the benchmark images with different numbers of superpixel counts, namely 200, 400, and 600. The superpixels computed using the same count are pooled to obtain the size distribution for the count. Clearly, although theses distributions have different counts, they also have a similar bell shape. Most of the superpixel sizes are close to the average size. The superpixels with very small spatial supports or very large spatial support are rare.

In the last experiment, I analyze the effects of the balancing term, $\lambda'$, and the kernel bandwidth, $\sigma$, parameters on the quality of segmentation. I observe that competitive segmentation results are achieved with a wide range of parameter

selection.

In Figure 3.12, I plot the performance curves for a range of $\lambda'$ values for a fixed $\sigma = 5.0$. I observe that smaller $\lambda'$ results in better boundary recall rates especially for smaller superpixel counts, while the results remain largely invariant to this parameter for larger superpixel counts. I further observed that better performances on undersegmentation error and achievable segmentation accuracy are achieved with a larger $\lambda'$. In general,a tradeoff exists among different metrics based on the $\lambda'$ parameter, and I found empirically that $\lambda' = 0.5$ yields a good compromise among these metrics.

In Figure 3.13, I plot the performance curves for a range of $\sigma$ values for a fixed $\lambda' = 0.5$. I observe that a large range of $\sigma$ values results in comparable performances, namely from 0.5 to 5. The superpixels are largely insensitive to the selection of the $\sigma$ parameter.

The proposed algorithm is among the fastest superpixel segmentation algorithms and takes an average of 2.5 seconds to segment an image on the Berkeley benchmark ($481 \times 321$ pixels) using an Intel Core 2 Duo E8400 processor. Compared to the state-of-the-art methods, it is faster than the Graphcut superpixel [117] (6.4 seconds), turbopixel [71]( 15 seconds), and NCut (5 minutes), whereas it is slower than FH [36](0.5 seconds).

## 3.6 Summary

I presented a novel objective function for cluster analysis, being a conical combination of the entropy rate of a random walk on the data graph and a balancing criterion. The property of this objective function and its optimization were analyzed. I showed that, by exploiting its submodularity and a matroid structure, a simple greedy algorithm can efficiently compute a performance-guaranteed solution. I have achieved significant performance improvements for superpixel segmentation tasks and competitive results compared to the state-of-the-art clustering algorithms on standard datasets.

I plan to explore user-specified constraints in the clustering problem. Another interesting research direction would be to study hyper-graph clustering where edges are defined over a set of vertices.

## 3.7 Proofs

Here I present the proofs of the theorems. They are restated for ease of reference. WLOG is used as the abbreviation for without loss of generality. I use two as the base for the logarithm.

**Theorem 1:** The entropy rate of the random walk on the graph $\mathcal{H} : 2^E \rightarrow \mathbb{R}$ is a monotonically increasing submodular function under the proposed graph construction.

*Proof.* The proof is divided into two parts. The first part proves the monotonicity. In the second part, I show the submodularity.

*Monotonicity:* $\delta\mathcal{H}_{a_1}(A) \geq 0$ for all $A \subseteq E$ and $a_1 \in E \setminus A$. WLOG let assume $a_1 = e_{1,2}$. Under the selected set $A \cup \{e_{1,2}\}$, the loop weights for vertices $v_1$ and $v_2$ are given by $c_1 \equiv w_1 - \sum_{k:e_{1,k} \in A \cup \{e_{1,2}\}} w_{1,k}$ and $c_2 \equiv w_2 - \sum_{k:e_{2,k} \in A \cup \{e_{1,2}\}} w_{2,k}$ respectively. From (3.6) and with some simple algebraic manipulations, the marginal gain from adding $e_{1,2}$ to $A$ is given by

$$\begin{aligned}
\delta\mathcal{H}_{e_{1,2}}(A) = &\big\{\frac{w_{1,2}+c_1}{w_T}\log(\frac{w_{1,2}+c_1}{w_T}) - \frac{w_{1,2}}{w_T}\log(\frac{w_{1,2}}{w_T}) \\
&- \frac{c_1}{w_T}\log(\frac{c_1}{w_T})\big\} + \big\{\frac{w_{2,1}+c_2}{w_T}\log(\frac{w_{2,1}+c_2}{w_T}) \\
&- \frac{w_{2,1}}{w_T}\log(\frac{w_{2,1}}{w_T}) - \frac{c_2}{w_T}\log(\frac{c_2}{w_T})\big\} \qquad (3.19) \\
= &\frac{w_{1,2}+c_1}{w_T}\big\{\frac{w_{1,2}}{w_{1,2}+c_1}\log(\frac{w_{1,2}+c_1}{w_{1,2}}) + \frac{c_1}{w_{1,2}+c_1}\cdot \\
&\log(\frac{w_{1,2}+c_1}{c_1})\big\} + \frac{w_{2,1}+c_2}{w_T}\big\{\frac{w_{2,1}}{w_{2,1}+c_2}\log(\frac{w_{2,1}+c_2}{w_{2,1}}) \\
&+ \frac{c_2}{w_{2,1}+c_2}\log(\frac{w_{2,1}+c_2}{c_2})\big\} \geq 0. \qquad (3.20)
\end{aligned}$$

The terms in the two curly bracket pairs in (3.20) are the entropy values of the two binary random variables with distributions $(\frac{w_{1,2}}{w_{1,2}+c_1}, \frac{c_1}{w_{1,2}+c_1})$ and $(\frac{w_{2,1}}{w_{2,1}+c_2}, \frac{c_2}{w_{2,1}+c_2})$ respectively. Given the entropy of a discrete random variable is nonnegative, I show that $\delta\mathcal{H}_{e_{1,2}}(A) \geq 0$ and prove the monotonicity.

*Submodularity:* $\delta\mathcal{H}_{a_1}(A) \geq \delta\mathcal{H}_{a_1}(A \cup \{a_2\})$ for all $A \in E$ and $a_1, a_2 \in E \setminus A$. Based on whether these edges share a common vertex, I discuss the following two cases.

1.) $a_1$ and $a_2$ have no common vertex. WLOG, let assume $a_1 = e_{1,2}$ and $a_2 = e_{3,4}$. Since $e_{3,4}$ is not connected to either $v_1$ or $v_2$, the addition of $e_{3,4}$ has no

effect on the loop weights of $v_1$ and $v_2$; therefore, I have the following equalities

$$c_1 = w_1 p_{1,1}(A \cup \{e_{1,2}, e_{3,4}\}) = w_1 p_{1,1}(A \cup \{e_{1,2}\}) \tag{3.21}$$

$$c_2 = w_2 p_{2,2}(A \cup \{e_{1,2}, e_{3,4}\}) = w_2 p_{2,2}(A \cup \{e_{1,2}\}) \tag{3.22}$$

where $p_{i,j}$'s are the transition probabilities given in (3.6). As a result, $\delta\mathcal{H}_{e_{1,2}}(A \cup \{e_{3,4}\})$ can also be simplified to (3.19), and $\delta\mathcal{H}_{e_{1,2}}(A \cup \{e_{3,4}\}) = \delta\mathcal{H}_{e_{1,2}}(A)$.

2.) $a_1$ and $a_2$ share a common vertex. WLOG, let assume $a_1 = e_{1,2}$ and $a_2 = e_{1,3}$ where $v_1$ is the shared vertex. Due to the shared vertex, the loop weight for vertex $v_1$ and $v_2$ after the addition of $e_{1,3}$ are given by $d_1 \equiv w_1 - \sum_{k:e_{1,k} \in A \cup \{e_{1,2}, e_{1,3}\}} w_{1,k} = c_1 - w_{1,3}$ and $d_2 \equiv w_2 - \sum_{k:e_{2,k} \in A \cup \{e_{1,2}, e_{1,3}\}} w_{2,k} = w_2 - \sum_{k:e_{2,k} \in A \cup \{e_{1,2}\}} w_{2,k} = c_2$ respectively. The marginal gain from the addition of $e_{1,2}$ to $A \cup \{e_{3,4}\}$ is equal to

$$\begin{aligned}
\delta\mathcal{H}_{e_{1,2}}(A \cup \{e_{3,4}\}) = \big\{ & \frac{w_{1,2} + d_1}{w_T} \log(\frac{w_{1,2} + d_1}{w_T}) \\
& - \frac{w_{1,2}}{w_T} \log(\frac{w_{1,2}}{w_T}) - \frac{d_1}{w_T} \log(\frac{d_1}{w_T}) \big\} + \big\{ \frac{w_{2,1} + d_2}{w_T} \cdot \\
& \log(\frac{w_{2,1} + d_2}{w_T}) - \frac{w_{2,1}}{w_T} \log(\frac{w_{2,1}}{w_T}) - \frac{d_2}{w_T} \log(\frac{d_2}{w_T}) \big\}
\end{aligned} \tag{3.23}$$

By subtracting (3.23) from (3.19), we have

$$\delta\mathcal{H}_{e_{1,2}}(A) - \delta\mathcal{H}_{e_{1,2}}(A \cup \{e_{3,4}\}) \tag{3.24}$$

$$\begin{aligned}
= \big\{ & \frac{w_{1,2} + d_1 + w_{1,3}}{w_T} \log(\frac{w_{1,2} + d_1 + w_{1,3}}{w_T}) \\
& - \frac{d_1 + w_{1,3}}{w_T} \log(\frac{d_1 + w_{1,3}}{w_T}) \big\} - \big\{ \frac{w_{1,2} + d_1}{w_T} \log(\frac{w_{1,2} + d_1}{w_T}) \\
& - \frac{d_1}{w_T} \log(\frac{d_1}{w_T}) \big\} = g(\frac{d_1 + w_{1,3}}{w_T}) - g(\frac{d_1}{w_T}) > 0
\end{aligned} \tag{3.25}$$

The last equation is an application of the strictly increasing property of $g(x) \equiv (x + \xi)\log(x + \xi) - x\log(x)$ where $\xi = \frac{w_{1,2}}{w_T}$ in this case.

From the above case discussions, I show that $\delta\mathcal{H}_{e_{1,2}}(A) \geq \delta\mathcal{H}_{e_{1,2}}(A \cup \{e_{3,4}\})$ and complete the proof. □

**Theorem 2:** The balancing function $\mathcal{B} : 2^E \to \mathbb{R}$ is a monotonically increasing submodular function under the proposed graph construction.

*Proof.* I prove the monotonicity and submodularity separately.

*Monotonicity:* $\delta\mathcal{B}_{a_1}(A) \geq 0$ for all $A \subseteq E$ and $a_1 \in E \setminus A$. WLOG let assume $a_1 = e_{1,2}$, $v_1$ be in cluster $S_i$, and $v_2$ be in cluster $S_j$. From (3.10), the probability that a randomly chosen vertex is in $S_i$ and $S_j$ is given by $p_{Z_A}(i) = \frac{|S_i|}{|V|}$ and $p_{Z_A}(j) = \frac{|S_j|}{|V|}$ respectively.

To prove the monotonicity, I discuss the case where $i \neq j$ — $v_1$ and $v_2$ are in two different clusters given $A$. [2] Let $p_i = p_{Z_A}(i)$ and $p_j = p_{Z_A}(j)$. The addition of $e_{1,2}$ to $A$ merges $S_i$ and $S_j$, and the probability that a randomly chosen vertex is in the merged cluster is equal to $p_{Z_A}(i) + p_{Z_A}(j) = \frac{|S_i|+|S_j|}{|V|}$. The increase in the balancing function is then given by

$$\begin{aligned}
\delta\mathcal{B}_{e_{1,2}}(A) &= H(A \cup \{e_{1,2}\}) - H(A) - N_{\mathcal{A}\cup\{e_{1,2}\}} + N_{\mathcal{A}} \\
&= 1 - (p_i + p_j)\log(p_i + p_j) + p_i \log p_i + p_j \log p_j && (3.26) \\
&= 1 + p_i \log \frac{p_i}{p_i + p_j} + p_j \log \frac{p_j}{p_i + p_j} && (3.27) \\
&\geq 1 + (p_i + p_j)\log(\frac{p_i + p_j}{2(p_i + p_j)}) = 1 - (p_i + p_j) \geq 0. && (3.28)
\end{aligned}$$

Note that the first inequality in (3.28) is an application of the log-sum inequality. From (3.28) I prove the monotonically increasing property.

[2] If $i = j$, then $\delta\mathcal{B}_{a_1}(A) = 0$ and the monotonicity holds.

*Submodularity:* $\delta\mathcal{B}_{a_1}(A) \geq \delta\mathcal{B}_{a_1}(A \cup \{a_2\})$ for all $A \in E$ and $a_1, a_2 \in E \setminus A$. To prove the submodularity, I discuss the cases where the addition of $a_1$ to $A$ combines two clusters. If it does not, both sides of the above equation equal zero and the submodularity holds. WLOG let assume $a_1 = e_{1,2}$ and $a_2 = e_{3,4}$. Depending on whether the addition of $e_{3,4}$ combines two clusters, three derivative cases exist.

1.) Let assume the addition of $e_{3,4}$ combines the clusters $S_i$ and $S_j$. This means $v_1$ and $v_2$ are in the same cluster given $A \cup \{e_{3,4}\}$. Therefore the addition of $e_{1,2}$ to $A \cup \{e_{3,4}\}$ has no effect on the graph partition. Both the number of clusters and the cluster membership distribution remain the same; hence, $\delta\mathcal{B}_{e_{1,2}}(A) \geq \delta\mathcal{B}_{e_{1,2}}(A \cup \{e_{3,4}\}) = 0$.

2.) In the case that $e_{3,4}$ combines some other clusters $S_k$ and $S_m$ where $|\{k, m\} \cap \{i, j\}| = \emptyset$ or $e_{3,4}$ does not combine any clusters. The addition of $e_{1,2}$ will still merge $S_i$ and $S_j$. Moreover, $p_{Z_{A\cup\{e_{3,4}\}}}(i) = p_{Z_A}(i)$ and $p_{Z_{A\cup\{e_{3,4}\}}}(j) = p_{Z_A}(j)$. As a result $\delta\mathcal{B}_{e_{1,2}}(A) = \delta\mathcal{B}_{e_{1,2}}(A \cup \{e_{3,4}\})$.

3.) Suppose that the addition of $e_{3,4}$ combines $S_k$ to $S_i$. Let $p_k = p_{Z_A}(k)$. The marginal gain obtained from adding $e_{1,2}$ to $A \cup \{e_{3,4}\}$ is given by

$$\begin{aligned}\delta\mathcal{B}_{e_{1,2}}(A \cup \{e_{3,4}\}) &= -(p_i + p_j + p_k)\log(p_i + p_j + p_k) \\ &+ (p_i + p_k)\log(p_i + p_k) + p_j \log(p_j) + 1. \end{aligned} \tag{3.29}$$

By subtracting (3.29) from (3.26), we have

$$\begin{aligned}
&\delta\mathcal{B}_{e_{1,2}}(A) - \delta\mathcal{B}_{e_{1,2}}(A \cup \{e_{3,4}\}) \\
&= (p_i + p_j + p_k)\log(p_i + p_j + p_k) - (p_i + p_j)\log(p_i + p_j) \\
&\quad - ((p_i + p_k)\log(p_i + p_k) - p_i \log p_i) \\
&= g(p_i + p_j) - g(p_i) \geq 0
\end{aligned} \tag{3.30}$$

Note that the last inequality is an application of strictly increasing property of the function $g(x) = (x + \xi)\log(x + \xi) - x\log(x)$. □ □

**Theorem 3:** Let $E$ be the edge set, and let $\mathcal{I}$ be the set of subsets $A \subseteq E$ which satisfies: 1) $A$ is cycle-free and 2) $A$ constitutes a graph partition with more than or equal to $K$ connected components. Then, the pair $M = (E, \mathcal{I})$ is a matroid.

*Proof.* The proof is given by showing that $M$ satisfies the three matroid conditions.

1) It is straightforward to note that the empty set is an independent set of the matroid — it contains no cycles and constitutes a $|V|$-partition where $N_\emptyset = |V| > K$.

2) Let $I \in \mathcal{I}$ and $I' \subseteq I$. This implies that $I$ has no cycles and $N_I \geq K$. Since $I' \subseteq I$, the set $I'$ can be obtained by removing edges from the set $I$. The removal of edges does not add cycles and increases the number of connected components. Therefore, the set $I'$ also contains no cycles and $N_{I'} \geq K$.

3) Let $I_1$ and $I_2$ be two independent sets in $\mathcal{I}$ such that $|I_1| < |I_2|$. The independent set assumptions imply the associated connected components of $I_1$ and $I_2$ satisfy the following relations: $N_{I_1} = |V| - |I_1| \geq K$, $N_{I_2} = |V| - |I_2| \geq K$, and $|I_1| < |I_2| \implies |V| - |I_1| > |V| - |I_2| \geq K$. From the above statements we have the

following equations.

$$N_{I_1} > N_{I_2} \tag{3.31}$$

$$N_{I_1} = |V| - |I_1| \geq K + 1 \tag{3.32}$$

I now prove that there exists some $e \in I_2 - I_1$ such that $N_{I_1 \cup \{e\}} \geq K$ and the set $I_1 \cup \{e\}$ is cycle-free; i.e., $I_1 \cup \{e\}$ is an independent set. Since adding one edge to a graph decreases the number of connected components by at most one, (3.32) implies $N_{I_1 \cup \{e\}} \geq K$ for $e \in I_2 - I_1$. The remaining part of the proof is achieved by contradiction.

Let us assume there is no edge $e \in I_2 - I_1$ such that $I_1 \cup \{e\}$ is cycle-free. In other words, adding any edge $e$ in $I_2 - I_1$ to the set $I_1$ will result in a cycle and leaves the number of connected components in the graph unchanged, $N_{I_1 \cup \{e\}} = N_{I_1}$. Thus by adding all the edges from $I_2 - I_1$ to the set $I_1$, the number of connected components will remain as $N_{I_1}$, i.e., $N_{I_1 \cup (I_2 - I_1)} = N_{I_1 \cup I_2} = N_{I_1}$.

We know that the set $I_1 \cup I_2$ can also be obtained by adding edges from $I_1 - I_2$ to $I_2$. Since adding edges to a graph can only decrease the number of connected components, we have the following relation $N_{I_1 \cup I_2} \leq N_{I_2} \implies N_{I_1} \leq N_{I_2}$. This contradicts (3.31), and thus the theorem is proved. □

# Chapter 4

# Submodular Function Maximization with Higher-Order Priors for Video Segmentation

## 4.1 Introduction

Video segmentation methods in computer vision can be classified into two categories: unsupervised [69, 21, 116, 48] and semi-supervised [22, 113, 24, 3, 124, 72]. The proposed algorithm belongs to the latter class where the user provides some annotation such as the segmentation of the first frame, and the goal is to propagate the information to other frames in a spatially and temporally consistent manner.

Most of the existing algorithms for semi-supervised video segmentation are based on minimizing an energy function that models the pixel-wise likelihoods for segmentation in unary terms and smoothness between neighboring pixels in pair-wise terms. Several algorithms exist for this minimization problem, and graph cuts presents one popular choice. Rother et al. [102] developed an interactive image segmentation algorithm based on graph cuts, which was later extended for videos by

Wang et al. [124]. This extension for videos often requires substantial subsequent user annotations. This drawback is later addressed in [3] with local classifiers, but the approach tends to be unreliable for segmenting textureless objects. Recently, Tsai et al. [113] incorporated optical flow in the energy function for video segmentation. In contrast to the existing algorithms, I use a higher-order prior for video segmentation based on histogram similarity; I also develop a novel optimization algorithm.

Graph cuts [19] and message passing [131, 64] are two popular energy minimization algorithms for several vision problems like stereo, segmentation, optical flow, and object recognition. A move-making algorithm like alpha-expansion[19] is widely used for energy functions involving Potts model prior. Recently, higher-order priors have been proven beneficial in several problems such as object class segmentation [62, 67], stereo [125], image segmentation [118, 63], image restoration [53], and single view 3D reconstruction [98, 44]. Kolmogorov and Zabih [65] first showed the transformation of submodular third order functions to pairwise functions, which can be optimized with st-mincut. Several other researchers have considered the problem of higher-order functions by transforming them to pairwise ones [53]. Recently, Jegelka and Bilmes [58] proposed cooperative cut, which can optimize a certain class of higher-order energy functions. This technique was employed to solve image segmentation problem consisting of thin regions.

In this chapter, I propose a higher-order energy function called histogram similarity prior for video segmentation. It is based on the observation that appearances of foreground and background objects tend to remain similar in several consecutive

frames of a video, despite articulation and deformation. Specifically, I encode the appearance of the objects using a dictionary, compute a histogram of dictionary words and pixel labels for each frame, and encourage a segmentation where the histograms of consecutive frames resemble each other. This modeling is particularly useful when the foreground and background objects have similar appearance. In this case, the histogram similarity prior leads to additional robustness over energy functions solely based on the Potts model because it encourages a spatially and temporally coherent segmentation.

I use Kullback-Leibler (KL) divergence to measure the similarity between histograms and propose minimizing the combination of Potts energy model and the histogram similarity prior for video segmentation. While minimizing the Potts energy model alone can be solved efficiently with the graph cuts algorithm [19], minimizing the combination leads to a NP-hard problem. For facilitating the prior, I carefully analyze the resulting optimization problem. In particular, I establish the following results:

1. the minimization of the Potts energy function is equivalent to the maximization of a submodular function subject to a matroid constraint, and

2. the minimization of the histogram similarity prior is equivalent to the maximization of a submodular function subject to a matroid constraint.

Thus, one can minimize an energy function involving both the Potts model and histogram similarity prior by maximizing the associated submodular function subject to a matroid constraint. Maximizing a submodular function subject to a matroid

constraint is NP-hard [91]; however, one can use the greedy algorithm to compute a solution whose worst case performance is bounded by $\frac{1}{2}$ of the optimal solution [42]. The use of the greedy algorithm for maximizing a submodular function subject to a matroid constraint has been recently applied to several problems [49, 70, 73, 75]. In this chapter, I propose a branch and bound algorithm that can potentially lead to the global optimum efficiently. This uses the greedy result as the initial solution, and the bounding function is based on the submodularity of the associated set function. Since the initial solution is bounded by $\frac{1}{2}$, the incumbent—the current best solution maintained in the branch and bound procedure—is also bounded by $\frac{1}{2}$.

### 4.1.1 Contribution

The chapter's contribution is as follows:

- I propose the histogram similarity prior for video segmentation.
- I show that minimizing the Potts energy model, the histogram similarity prior, and their combination can all be mapped to a problem of maximizing a submodular function subject to a matroid constraint.
- The standard approach to maximize a submodular function subject to a matroid constraint is based on the greedy algorithm, which is a $\frac{1}{2}$ approximation algorithm for this NP-hard problem. I further develop a branch and bound procedure to improve the solution. The branch and bound procedure uses the greedy solution as the initial solution; hence the worst case performance is also bounded by $\frac{1}{2}$.

- I evaluate the proposed algorithm in a standard dataset and achieve comparable or better performance with the state of the art.

The chapter is organized as follows. Background materials are reviewed in Section 4.2. In Section 4.3, I establish the equivalence between the Potts model energy minimization problem and a submodular function maximization problem. The histogram similarity prior is introduced in Section 4.4. The optimization algorithm is discussed in Section 4.5. I present the experiment results in Section 4.6 and conclude the chapter in Section 4.7.

## 4.2 Preliminaries

For ease of explanation, throughout the chapter I refer to pixels as the basic elements of the image/video. However, any other primitives can also be defined such as superpixels, supervoxels, and image patches. In my experiments, I use supervoxels. I use $\mathbb{R}$ to denote the set of real numbers and $\mathbb{R}_+$ to denote the set of positive real numbers.

**Matroids:** A matroid $M = (S, \mathcal{I})$ is a combinatorial structure consisting of a ground set $S$ and a collection $\mathcal{I}$ of subsets of $S$ satisfying the following three axioms [97]: (I1) $\emptyset \in \mathcal{I}$, (I2) If $I_2 \in \mathcal{I}$ and $I_1 \subseteq I_2$, then $I_1 \in \mathcal{I}$, and (I3) If $I_1$ and $I_2$ are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there is an element $s$ of $I_2 - I_1$ such that $I_1 \cup \{s\} \in \mathcal{I}$.

A member of $\mathcal{I}$ is called an independent set of $M$. Note that the third axiom implies that every maximal[1] independent set is maximum or, equivalently, all maxi-

[1] inclusion-wise

mal independent sets have the same cardinality. A maximal independent set is also called a basis $B$. The collection of bases of $M$ is denoted by $\mathcal{B}$. There are several types of matriods, and I use the partition matroid in this paper.

**Partition matroids:** Let $\{S_1, S_2, ..., S_n\}$ be a partition of a finite set $S$, i.e., $S = \cup_{i=1}^{n} S_i$ and $S_i \cap S_j = \emptyset$ for all $i \neq j$. A partition matroid is defined as $M = (S, \mathcal{I})$, where $\mathcal{I}$ is a collection of subsets of $S$ satisfying the property that each member of $\mathcal{I}$ has at most $\kappa_i$ elements from the partition $S_i$ for some nonnegative integers $\kappa_i$'s. In other words, the collection of independent sets is

$$\mathcal{I} = \{A \subseteq S : |A \cap S_i| \leq \kappa_i, \forall i = 1, ..., n\}. \tag{4.1}$$

A basis is a maximal independent set, and the collection of bases is given by

$$\mathcal{B} = \{A \subseteq S : |A \cap S_i| = \kappa_i, \forall i = 1, ..., n\} \tag{4.2}$$

**Submodular functions:** Let $S$ be a finite set. A set function $f : 2^S \to \mathbb{R}$ is submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{4.3}$$

for all $A, B \subseteq S$. The set function $f$ is called supermodular if the reversed inequality holds true for every pair of subsets. Finally, $f$ is called modular if it is both submodular and supermodular. One can express the submodular inequality in other ways; for example, (4.3) is equivalent to

$$f(A) \leq f(B) + \sum_{a \in A \setminus B} \rho_a(B \setminus \{a\}) - \sum_{a \in B \setminus A} \rho_a(A \cup B \setminus \{a\}) \tag{4.4}$$

where $\rho_a(A) \equiv f(A \cup \{a\}) - f(A)$ is the marginal gain.

**Monotonically increasing functions:** A set function $f$ is monotonically increasing if $f(A) \leq f(A \cup \{a\})$ or $\rho_a(A) \geq 0$ for all $A \subseteq S$ and $a \in S \setminus A$.

**Pseudo-Boolean function (PBF):** A PBF takes a Boolean vector as input and returns a real number [16]. Let us consider a Boolean vector $\mathbf{x} = (x_1, x_2, ..., x_{|S|}) \in \{0,1\}^{|S|}$. Note that $\mathbf{x}$ can also be seen as a subset $A$ of a set $S$ where $x_i = 0$ when $i \notin A$ and $x_i = 1$ when $i \in A$. Thus, a PBF is a function $\phi : \{0,1\}^{|S|} \rightarrow \mathbb{R}$. In this paper, I use PBF to express the energy function for segmentation.

**Energy minimization:** Let $\mathcal{V} = \{1, 2, ..., n\}$ be a set of image pixels and $L = \{1, 2, ..., |L|\}$ be a set of labels. Many computer vision problems computes the label vector $\mathbf{y} = (y_1, y_2, ..., y_n) \in L^n$ so an associated energy function $E : L^n \rightarrow \mathbb{R}_+ \cup \{0\}$ is minimized. The energy function usually consists of many terms; each is defined on a subset $C$ of $\mathcal{V}$, called a clique. It is written as $E(\mathbf{y}) = \sum_{C \in \mathcal{C}} E_C(\mathbf{y}_C)$, where $\mathcal{C}$ is the set of all cliques and $\mathbf{y}_C$ are the entries in $\mathbf{y}$ corresponding to the primitives in $C$. The energy minimization problem is a problem of

$$\arg\min_{\mathbf{y}} \sum_{C \in \mathcal{C}} E_C(\mathbf{y}_C). \tag{4.5}$$

When the clique size is one or two, they are usually referred to as unary or pairwise terms, respectively.

## 4.3 Submodular Function Maximization

In this section, I design a new PBF in a way that the optimal solution of certain class of energy functions in (4.5) can also be computed by maximizing the PBF under a constraint. This is achieved by using a set of Boolean variables to denote the label

vector $\mathbf{y}$. I demonstrates this PBF is submodular and the introduction of Boolean variables leads to a partition matroid constraint.

### 4.3.1 Pseudo-Boolean Representation

I consider minimizing an energy function consisting of unary and pairwise energy terms, i.e.,

$$E(\mathbf{y}) = \sum_{i \in \mathcal{V}} E_i(y_i) + \sum_{(i,j) \in \mathcal{N}} E_{ij}(y_i, y_j) \tag{4.6}$$

where $\mathcal{N}$ is the set of adjacent pixels. I assume a Potts pairwise model where $E_{ij}(y_i, y_j) = 0$ when $y_i = y_j$ and $E_{ij}(y_i, y_j) = w_{ij}$ when $y_i \neq y_j$. Here, $w_{ij}$'s are nonnegative location-dependent weights.

I will now show that the minimization problem given in (4.6) is equivalent to a maximization problem of a carefully constructed monotonically increasing submodular PBF. I represent the label vector $\mathbf{y}$ by a Boolean vector $\mathbf{x} = \{x_{il} : (i,l) \in \mathcal{V} \times L\}$ where

$$x_{il} = \begin{cases} 1, & y_i = l \\ 0, & y_i \neq l \end{cases} \tag{4.7}$$

With this representation, I define a new quadratic PBF given by

$$\phi(\mathbf{x}) = \sum_{i \in \mathcal{V}, l \in L} K_i x_{il} - \sum_{i \in \mathcal{V}, l \in L} E_i(l) x_{il} - \sum_{(i,j) \in \mathcal{N}, l_1 \in L, l_2 \in L} E_{ij}(l_1, l_2) x_{il_1} x_{jl_2}. \tag{4.8}$$

where

$$K_i \equiv \max_{l \in L} E_i(l) + \sum_{(i,j) \in \mathcal{N}} |L| w_{i,j}, \tag{4.9}$$

that ensure monotonicity of the set function $\phi$. A quadratic PBF is submodular if all the coefficients of the quadratic terms are non-positive [16]. Since the coefficients

of quadratic terms in (4.8) are either $-w_{ij}$ or 0, $\phi(X)$ is submodular.

I further note that a 1-1 correspondence between $\mathbf{y}$ and $\mathbf{x}$ can be established by the following constraint:

$$\sum_{l \in L} x_{il} = 1, \quad \forall\, i \in \mathcal{V}, \tag{4.10}$$

which ensures that the Boolean vector $\mathbf{x}$ corresponds to a label vector $\mathbf{y}$ where every pixel takes exactly one label. Given the constraint (4.10), $\phi(\mathbf{x})$ and $E(\mathbf{y})$ is related by

$$\phi(\mathbf{x}) = K - E(\mathbf{y}) \tag{4.11}$$

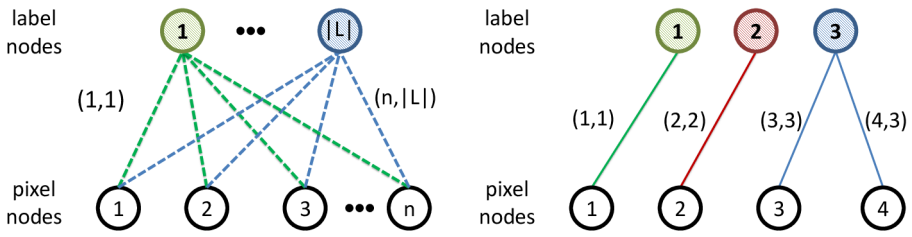
where $K = \sum_{i \in \mathcal{V}} K_i$ is a constant. Thus the problem of minimizing $E(\mathbf{y})$ is also equivalent to maximizing a submodular function $\phi(\mathbf{x})$ under the above constraint (4.10). In what follows, I will show that the constraint (4.10) can be replaced by a partition matroid constraint.

### 4.3.2 Partition Matroid Constraint

I consider a partition matroid $M = (S, \mathcal{I})$ whose bases are related to $\mathbf{y}$. Each element $(i, l)$ in the ground set $S$ denotes an assignment of a label $l$ to a pixel $i$. Let $\{S_1, S_2, ..., S_n\}$ be a partition of the ground set $S$ where elements in $S_i$ represent all the possible label assignments for pixel $i$, i.e., $\{(i, l), (i, 2), ..., (i, |L|)\}$. The collection of independent sets of the partition matroid $M$ is given by

$$\mathcal{I} = \{A \subseteq S : |A \cap S_i| \leq 1, \forall\, i\}, \tag{4.12}$$

which constrains each pixel to have at most one label assignment. Fig. 4.1 illustrates the graphical representation of the partition matroid where pixel nodes $i$ are




(a) (b)

Figure 4.1: (a) Graphical representation of the partition matroid $M = (S, \mathcal{I})$. Each node in the bottom row represents an image pixel; it has edges connecting to all the label nodes in the top row. The edge is denoted by an ordered pair $(i, l)$ where the first index refers to the $i$th pixel node and the second index refers to the $l$th label node. Edges connecting to a pixel node forms a partition of $S$, $S_i = \{(i, l),\ \forall\, l \in L\}$. The independent set is given by subsets of $S$ that have at most one edge for each partition, $\mathcal{I} = \{A \subseteq S : |A \cap S_i| \leq 1, \forall i\}$. (b) I illustrate the graphic representation of an independent set containing elements $\{(1, 1), (2, 2), (3, 3), (4, 3)\}$. This independent set is a maximal independent set or a basis. It corresponds to a label vector of $\mathbf{y} = (1, 2, 3, 3)$.

connected to label nodes $l$ with edges $(i, l)$.

A basis of a matroid is a maximal independent set, and therefore the collection of bases in the above partition matroid is given by

$$\mathcal{B} = \{A \subseteq S : |A \cap S_i| = 1, \forall\, i\}. \tag{4.13}$$

Now, it is easy to see that a 1-to-1 correspondence exists between bases and the Boolean vector $\mathbf{x}$ under the constraint (4.10). Let $f : 2^S \rightarrow \mathbb{R}$ be a set function corresponding to the pseudo-Boolean function $\phi(\mathbf{x})$. Maximizing $\phi(\mathbf{x})$ under the constraint (4.10) is equivalent to solving the following problem

$$\arg\max_{B \in \mathcal{B}} f(B). \tag{4.14}$$

Furthermore, the set function $f$ is monotonically increasing and the details are shown in the supplementary material. Thus, the bases or the maximal independent sets of the partition matroid coincide with the local maxima of $f$. As a result, the solution set of the problem in (4.14) is the same as the solution set of the problem

$$\arg\max_{A \in \mathcal{I}} f(A). \tag{4.15}$$

I have shown that the energy minimization problem in (4.6) is equivalent to maximizing a submodular function subject to a partition matroid constraint.

## 4.4 Histogram Similarity Prior

The appearance distributions of foreground and background objects remain approximately the same in short periods of time in a video. To model this high order interaction, I introduce histogram similarity prior which penalizes the dissimilarity of an appearance histogram of a given image segmentation from a reference histogram. Let $D$ be a dictionary of visual words and $\mathbf{d} = (d_1, d_2, ..., d_n) \in D^n$ be a vector mapping the image pixels to these visual words. The appearance histogram of a given segmentation $\mathbf{y} \in L^n$, $H(\mathbf{y})$, contains $|L| \times |D|$ bins representing the joint appearance and segmentation frequencies of $|D|$ dictionary words, and $|L|$ segmentation labels in the image. I illustrate the histogram similarity prior in Fig 4.2.

One standard approach to measure the dissimilarity between two histograms uses Kullback-Leibler (KL) divergence [26]. Let $H^1$ and $H^2$ be two histograms. The

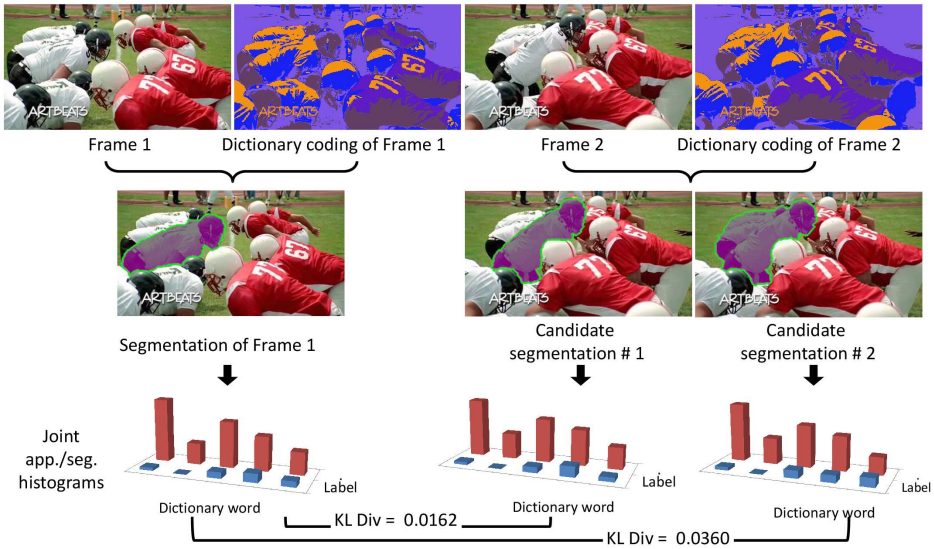



Figure 4.2: Illustration of the histogram similarity prior. I compare the histogram similarity between two candidate segmentations. Frame 1 is the reference frame where the segmentation is given,which I use to guide the segmentation of Frame 2. I use a dictionary to encode each pixel. Now for each candidate segmentation (corresponding to a label vector), a histogram is induced. The histogram similarity prior prefers to the preference of a segmentation with a histogram similar to the reference image.

KL divergence is given by

$$KL(H^1||H^2) = \sum_k H_k^1 \log \frac{H_k^1}{H_k^2} \tag{4.16}$$

where the subscript $k$ refers to the $k$th bin of the histogram. The KL divergence is not symmetric, but a symmetric version can be obtained by defining

$$KL_{sym}(H^1, H^2) = KL(H^1||H^2) + KL(H^2||H^1). \tag{4.17}$$

Let $H^r$ be the reference histogram obtained from the segmentation of the first frame, and $H^t$ be the histogram of following frames.[2] The histogram similarity prior

[2]Note that this reference histogram can be also learned from a training set.

is modeled as a higher order term involving the set of all the pixels $\mathcal{V}$.

$$E_{\mathcal{V}}(\mathbf{y}) \equiv \sum_{t} KL_{sym}(H^r, H^t(\mathbf{y})). \tag{4.18}$$

I establish the following result that enables us to minimize the above higher order term using the same maximization formulation.

**Theorem 4.1.** *The minimizing of the histogram similarity prior in (4.18) can be solved by maximizing a monotonically increasing submodular function* $g : 2^S \rightarrow \mathbb{R}$ *subject to the partition matroid constraint* $M = (S, \mathcal{I})$.

The proof is given in the supplementary material.

From the established equivalence in Theorem 4.1 and Section 4.3.2, the minimization of an energy function comprising a Potts model and the histogram similarity prior can be formulated as the maximization of a submodular function subject to a matroid constraint

$$arg\max_{A \in \mathcal{I}} \big(f(A) + \lambda g(A)\big), \tag{4.19}$$

where $\lambda \geq 0$ is a weighting factor.

## 4.5 Optimization

I present a greedy algorithm to maximize the set function given in (4.19). The algorithm starts with an empty set $A = \emptyset$ (all pixels are unlabeled). It adds an edge $a$ to $A$ (assigns a label to an unlabeled pixel) at a time based on the maximum increase in the set function (4.19) without violating the matroid constraint (every pixel is assigned to, at most, one label). The iterations terminate when the set $A$

is a maximal independent set of the matroid $M$ (when all the pixels are labeled). It is shown in [42] that the greedy algorithm yields a solution that is guaranteed to be within $\frac{1}{2}$ of the optimal solution for maximizing a monotonically increasing submodular function subject to a matroid constraint. Since the set function (4.19) is monotonically increasing and submodular, this bound holds for my optimization procedure.

I further present a branch and bound procedure to improve the optimization in a limited time budget or to find the global optimum. This procedure is based on a search tree structure where the root node corresponds to the original problem. The root node has $|L|$ child nodes, which correspond to subproblems after fixing the label of a single pixel to one of the $|L|$ segmentation labels. Further intermediate nodes in the tree hierarchy correspond to reduced subproblems where labels of more pixels are fixed, and the leaf nodes of the tree correspond to full labeling of the image. Note that the search tree's depth is $n$ being the number of pixels.

I use a depth first search strategy—the deepest node in the tree is selected and branched for further exploration at each iteration according to the greedy heuristic. Therefore, the greedy solution serves as the initial incumbent, or the initial best solution for the branch and bound procedure. For each branch, I compute a bound for the subproblem. Whenever the bound is better then the incumbent, the branched node remains in a pool of live nodes. However, if the bound is worse than the incumbent, the branch is discarded because no feasible solutions of the subproblem can be better than the incumbent. When a leaf node yields an objective value better than the incumbent, I update the incumbent with the leaf node solution.

The algorithm reaches a global optimum when the pool of live nodes is empty.

The branch and bound algorithm requires a bounding function to discard branches of the search tree that cannot contain an optimal solution. As explained above, each intermediate node of the tree corresponds to a subproblem where labels of exactly $p$ pixels are fixed, and $p$ is the depth of the node. I only need to compute a bound on the label assignment for the remaining $n-p$ pixels, which correspond to the nodes from the current node down to the leaf node in the search tree. Let $A_s^*$ be the best solution for the subproblem where the labels of the first $p$ pixels are fixed: $A_s^* = A_p \cup A_{n-p}^*$ where $A_p$ corresponds to the labels of the first $p$ pixels and $A_{n-p}^*$ are the best labels for the remaining pixels. By using the submodularity property given in (4.4), I have

$$f(A_s^*) \leq f(A_p) + \sum_{a \in A_s^* \backslash A_p} \rho_a(A_p) - \sum_{a \in A_p \backslash A_s^*} \rho_a(A_p \cup A_s^* - \{a\}) \tag{4.20}$$

$$= f(A_p) + \sum_{a \in A_s^* \backslash A_p} \rho_a(A_p) = f(A_p) + \sum_{a \in A_{n-p}^*} \rho_a(A_p) \tag{4.21}$$

$$\leq f(A_p) + \sum_{i=p+1}^{n} \big( \max_{a \in S_i} \rho_a(A_p) \big) \tag{4.22}$$

where $S_i$ is the partition corresponding to the pixel at depth $i$ in the search tree. Note that the third term in (4.20) equals zero because $A_p \backslash A_s^* = \emptyset$, and the inequality in (4.22) is simply due to the maximum operation. Therefore, the optimal solution for the subproblem $f(A_s^*)$ is bounded by (4.22).

Like many branch and bound algorithms, the efficiency of the proposed algorithm depends on the tightness of the bounding function—a problem-specific property and hard to analyze in general. Hence, I have no guarantee on the to-

tal execution time of the algorithm. However, for time-constrained applications, one can simply terminate the branch and bound computation at any instance and use the current incumbent as the final solution. Although the incumbent may not be the global optimal solution, it is guaranteed to be within $\frac{1}{2}$ of the global optimum since the initial solution is from the greedy heuristic.

## 4.6 Experiments

**Implementation:** I divide the input video into several overlapping clips; each clip contains five consecutive frames where the first frame overlaps with the last frame of the preceding clip. I assume the segmentation of the first frame of the video is given. For the following clip, I use the segmentation output of the last frame of the preceding clip as the segmentation of the first frame. To reduce the complexity of the optimization algorithm, I initially use a supervoxel segmentation procedure that produces temporally and spatially coherent 3D volumes. This procedure extends the superpixel segmentation algorithm [75] for video by constructing a spatiotemporal graph whose temporal connection is based on optical flow. With this over-segmentation, a clip is divided into several hundred supervoxels, each containing about $1,000$ pixels. The final video segmentation then becomes a task of finding an optimal label assignment for the supervoxels.

My unary terms are based on the Gaussian mixture model as proposed in [102]. To compute the pairwise terms, I first construct a supervoxel adjacency graph in

the spatiotemporal domain. The pairwise term is then given by

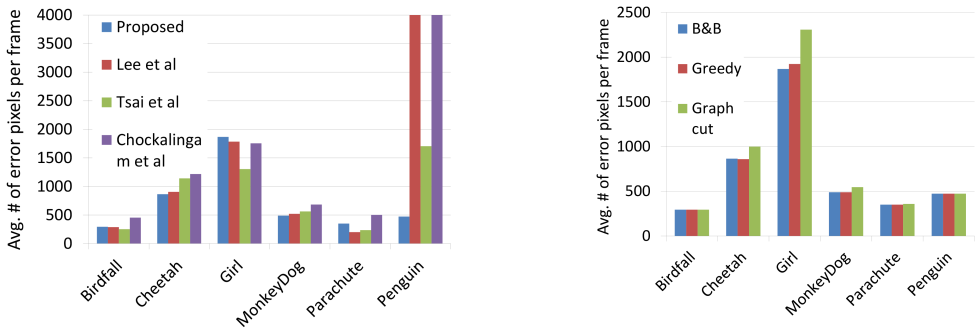
$$\gamma exp(-\frac{1}{\beta^2}||\mathbf{c}_i - \mathbf{c}_j||^2) \tag{4.23}$$

where $\mathbf{c}_i$ and $\mathbf{c}_j$ are the average RGB color of the supervoxels and $\beta$ is the average color difference of all pairs of neighboring supervoxels. I set $\gamma = 500$ in all my experiments.

I construct a dictionary via clustering pixels in the first frame of each clip. I used simple image features where pixels are represented by their RGB color and normalized image coordinates.[3] These vectors are divided into 30 clusters using K-means. The cluster centers are used as dictionary words, and pixels in all the clip's frames are encoded using nearest neighbor assignment. Since I use $|D| = 30$ visual words and the segmentation problem contains $|L| = 2$ classes, the joint appearance and segmentation histogram contains 60 bins. The video segmentation is achieved by solving the optimization problem given in (4.19) over the supervoxel representation. Throughout the experiments, I set $\lambda = 5,000$.

**Results:** I evaluated the proposed algorithm using the SegTrack dataset [113], which contains six videos, namely Birdfall, Cheetah, Girl, Monkey–Dog, Parachute, and Penguin. Pixel-level ground truth segmentations for all the frames in the videos are available. The dataset is highly challenging, including similar foreground and background colors, motion blur, occlusion, articulation, and compression artifacts. The task aims to segment the video with a pre-specified binary segmentation of the first frame.

[3]The coordinates are normalized by the image width.




(a) comparison with state-of-the-art (b) comparison on optimization techniques

Figure 4.3: (a) Performance comparison with the state-of-the-art on the SegTrack dataset [113]. The proposed algorithm yields better average segmentation accuracy and achieves best performance on three of the six challenging videos. (b) Performance comparison of the proposed energy function and optimization schemes with the Potts model using graph-cuts optimization.

I compared my algorithm with the state-of-the-art and reported in Fig. 4.3(a); the performance of the other algorithms were duplicated from a recent paper [69]. The performance metric used is the average number of error pixels per frame as proposed in [113]. On three of six videos (Cheetah, Monkey–Dog, and Penguin), the proposed algorithm achieved better segmentation accuracy whereas the performance was slightly inferior to Lee et al. [69] for the Parachute video, and to Tsai et al. [113] for the Birdfall and Girl videos. The Birdfall and Girl videos contained significant motion blur, which resulted in errors in initial supervoxel segmentation, reducing the performance of the proposed algorithm. On average, the proposed algorithm achieved a better error rate of 723 pixel per frame over six video sequences, whereas the second best method (Tsai et al. [113]) achieved 867 pixel per frame. I show several segmentation examples in Fig. 4.4; and more can be found in the

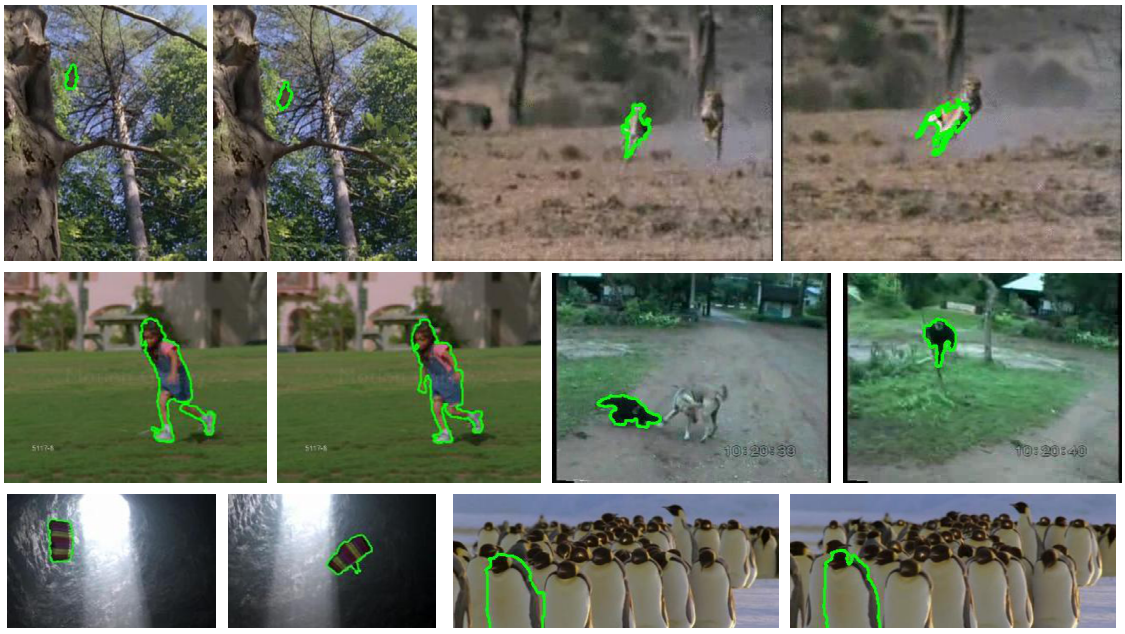
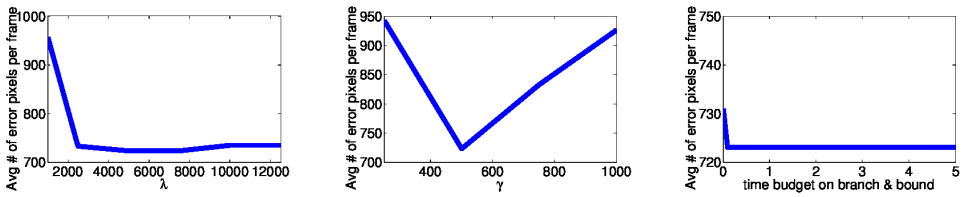


Figure 4.4: Several examples of segmentation results from the six videos: Birdfall, Cheetah, Girl, Monkey–Dog, Parachute, and Penguin. The foreground boundaries are superimposed on the images. More examples can be found in the supplementary material.




(a) $\lambda$ (b) $\gamma$ (c) B&B time (sec.)

Figure 4.5: Parameter sensitivity analysis with respect to (a) the weighting factor on the histogram similarity energy, $\lambda$, (b) the weighting factor on the pairwise term, $\gamma$, and (c) time budget for the branch and bound procedure.

supplementary material.

In Fig 4.3(b), I illustrate the effect of the histogram similarity prior by comparing the proposed energy function with the standard Potts model. I used the graph

cut [19] to optimize the Potts energy function, which produces the optimal result for the given binary segmentation problem. In addition, I compared two suboptimal optimization schemes for the proposed Potts and histogram similarity energy function: 1) Greedy, and 2) Branch and bound. Note that I terminated the branch and bound algorithm after 1 second and returned the current best solution; hence it only guarantees a $\frac{1}{2}$ approximation bound inherited from the greedy algorithm. As shown in Fig 4.3(b), the incorporation of the histogram similarity prior leads to an improved performance over the Potts model even though a simple feature representation is used. The improvement using branch and bound procedure over the greedy scheme appears minor which I believe relates to the already good solution found by the greedy algorithm and the limited time budget.

In Fig 4.5, I analyzed the segmentation's sensitivity with to the parameter values. I found that the proposed algorithm is largely insensitive to exact tuning of the parameters and produces good results for a wide range of parameter values. In Fig. 4.5(c), I analyzed the segmentation accuracy with respect to the time spent on the branch and bound algorithm. The result indicated that the algorithm found a suboptimal solution quickly and did not improve further in a limited time constraint.

## 4.7 Conclusion

I show an approach to convert the Potts energy minimization problem to a matroid-constrained submodular function maximization problem. I introduce the histogram similarity prior for video segmentation and show that it can be naturally

handled with the maximization framework. My framework is not limited to the energy functions studied in the chapter; it can be applied to solve a higher-order energy function that has an associated submodular set function in the maximization form. In future, I plan to study the class of hard constraints occurred in computer vision that can be enforced via matroids.

## 4.8 Proof

In this section, I present the proofs of the claims in the main paper.

### 4.8.1 Monotonically Increasing Property of $f$

I show that the set function $f$ defined in Section 4.3 is monotonically increasing. It is achieved by showing that the marginal gain, shown below, is always nonnegative for all $A \in S$ and $(i,l) \in S \setminus A$.

$$f(A \cup \{(i,l)\}) - f(A) = K_i - E_i(l) - \sum_{(i,j)\in\mathcal{N},(j,k)\in A,k\neq l} w_{ij}$$

$$= \big(\max_{l'\in L} E_i(l') - E_i(l)\big) + \big(\sum_{(i,j)\in\mathcal{N}} |L| w_{ij} - \sum_{(i,j)\in\mathcal{N},(j,k)\in A,k\neq l} w_{ij}\big) \geq 0 \qquad (4.24)$$

Note that $K_i$ is given by

$$K_i \equiv \max_{l\in L} E_i(l) + \sum_{(i,j)\in\mathcal{N}} |L| w_{i,j}, \qquad (4.25)$$

and the inequality in (4.24) is due to

$$\max_{l'\in L} E_i(l') \geq E_i(l) \qquad (4.26)$$

and

$$\sum_{(i,j)\in\mathcal{N}} |L| w_{ij} \geq \sum_{(i,j)\in\mathcal{N},(j,k)\in S_j,k\neq l} w_{ij} \geq \sum_{(i,j)\in\mathcal{N},(j,k)\in A,k\neq l} w_{ij}. \tag{4.27}$$

### 4.8.2 Proof of the submodularity of the KL Divergence

I present the proof of Theorem 4.1. I first prove two lemmas where the first lemma shows that the logarithm function [4] of a monotonically increasing modular function is a monotonically increasing submodular function, and the second lemma shows that the set function $-q \log q$ is submodular if $q$ is a monotonically increasing modular function.

#### 4.8.2.1 Proof of Lemma 1

**Lemma 4.1.** *Let $S$ be a finite set and $q : 2^S \to \mathbb{R}$ be a monotonically increasing modular function with $q(\emptyset) = 0$. Then the set function $\log(q)$ is monotonically increasing and submodular.*

*Proof.* Let $A$ be a subset of $S$ and $a \in S \setminus A$. The marginal gain of the set function $\log(q)$ is given by

$$\log(q(A \cup \{a\})) - \log(q(A)) = \log(\frac{q(A \cup \{a\})}{q(A)}) \geq 0 \tag{4.28}$$

This is because the monotonically increasing property of the set function $q$, i.e., $q(A \cup \{a\}) \geq q(A)$.

[4] The logarithm function is defined on $\mathbb{R}_+$. In order to avoid $\log(0)$, I assume that a small constant $\epsilon > 0$ is added to the argument of the logarithm function. For the ease of presentation, I use $\log(x)$ to denote $\log(x + \epsilon)$ where $x \geq 0$ and omit $\epsilon$ for the rest of the supplementary material.

To prove the submodularity, I show the marginal gain in (4.28) is monotonically decreasing (Proposition 2.1 (iii) in [91]). Specifically, I consider

$$\Big(\log(q(A\cup\{a\}))-\log(q(A))\Big)-\Big(\log(q(A\cup\{a,b\}))-\log(q(A\cup\{b\}))\Big) \tag{4.29}$$

where $b\in S\backslash(A\cup\{a\})$. With some algebraic manipulation, one can show that (4.29) is equal to

$$\log(\frac{q(A\cup\{a\})q(A\cup\{b\})}{q(A\cup\{a,b\})q(A)}) \tag{4.30}$$

$$=\log(\frac{\Big(q(A)+q(\{a\})\Big)\Big(q(A)+q(\{b\})\Big)}{\Big(q(A)+q(\{a\})+q(\{b\})\Big)q(A)}) \tag{4.31}$$

$$=\log(\frac{q(A)q(A)+\Big(q(\{a\})+q(\{b\})\Big)q(A)+q(\{a\})q(\{b\})}{q(A)q(A)+\Big(q(\{a\})+q(\{b\})\Big)q(A)})\geq 0. \tag{4.32}$$

Note that the inequality in (4.32) is due to $q(\{a\})q(\{b\})\geq 0$ (from the monotonically increasing property of the set function $q$ and $q(\emptyset)=0$). □

#### 4.8.2.2 Proof of Lemma 2

**Lemma 4.2.** *Let $S$ be a finite set and $q:2^S\to\mathbb{R}$ be a monotonically increasing modular function with $q(\emptyset)=0$. Then the set function $-q\log(q)$ is submodular.*

*Proof.* Let $A$ be a subset of $S$ and $a\in S\setminus A$. The marginal gain of the set function $-q\log(q)$ is given by

$$\delta_a(A)\equiv -q(A\cup\{a\})\log q(A\cup\{a\})+q(A)\log q(A) \tag{4.33}$$

$$=-\big(q(A)+q(\{a\})\big)\log\big(q(A)+q(\{a\})\big)+q(A)\log q(A). \tag{4.34}$$

For a fixed $a$, the marginal gain in (4.34) is a function of $q(A)$. Moreover, the marginal gain is a monotonically *decreasing* function of $q(A)$. Since $q$ is a monotonically *increasing* function of $A$, the marginal gain is a monotonically *decreasing* function of $A$. From Proposition 1.1 in [78], the set function $-q\log(q)$ is submodular. □

#### 4.8.2.3 Proof of the submodularity of the KL Divergence

*Proof.* I first observe that the negation of the energy function is equal to

$$-E_{\mathcal{V}}(\mathbf{y}) = \sum_{t}(-KL(H^r||H^t(\mathbf{y})) - KL(H^t(\mathbf{y})||H^r)) = \tag{4.35}$$

$$\sum_{t}\sum_{k}\left(H_k^r \log H_k^t(\mathbf{y}) - H_k^r \log H_k^r + H_k^t(\mathbf{y}) \log H_k^r - H_k^t(\mathbf{y}) \log H_k^t(\mathbf{y})\right) \tag{4.36}$$

where $H_k^r$ is constant, which denotes the percentage of pixels in the $k$th bin of the histogram associated with the reference frame. Note that my joint appearance and segmentation histogram is a 2-dimensional one where the two dimensions are dictionary word index $d$ and the segmentation label $l$, respectively. Hence, the subscript $k$ represents a tuple $(d, l)$.

I represent the label vector $\mathbf{y}$ by a Boolean vector $\mathbf{x} = \{x_{il} : (i, l) \in \mathcal{V} \times L\}$ where $x_{il} = 1$ when $y_i = l$ and $x_{il} = 0$ when $y_i \neq l$ as defined in the main paper. With this representation, I design a pseudo-Boolean function $\psi(\mathbf{x})$ associated with

$E_{\mathcal{V}}(\mathbf{y})$ given by

$$\psi(\mathbf{x}) = \sum_t \sum_k \Bigg( \underbrace{\sum_{i\in\mathcal{V},\, l\in L} \xi_i x_{il}}_{\text{modular}} + \underbrace{H_k^r \log \sum_{i\in\mathcal{V},\, l\in L} H_k^t(i,l) x_{il}}_{\text{submodular}}$$

$$\underbrace{- H_k^r \log H_k^r}_{\text{constant}} + \underbrace{\sum_{i\in\mathcal{V},\, l\in L} H_k^t(i,l) x_{il} \log H_k^r}_{\text{modular}}$$

$$+ \Big( \underbrace{\sum_{i\in\mathcal{V},\, l\in L} -H_k^t(i,l) x_{il} \Big) \log \sum_{i\in\mathcal{V},\, l\in L} H_k^t(i,l) x_{il}}_{\text{submodular}} \Bigg) \tag{4.37}$$

where

$$\xi_i = - \min_{l'\in L} \big( H_k^t(i,l') \log H_k^r \big) - \big( \sum_{(j,m)\in S} H_k^t(j,m) \big) \log \big( \sum_{(j,m)\in S} H_k^t(j,m) \big)$$

$$+ \max_{l'\in L} \big( \sum_{(j,m)\in S\setminus\{(i,l')\}} H_k^t(j,m) \big) \log \big( \sum_{(j,m)\in S\setminus\{(i,l')\}} H_k^t(j,m) \big), \tag{4.38}$$

which ensures the monotonicity of the set function $\psi$ (shown later in the proof), and $H_k^t(i,l) = 1$ if pixel $i$ is assigned to the $k$th bin of the histogram $H^t$, and $H_k^t(i,l) = 0$ otherwise. Note that given the constraint

$$\sum_{l\in L} x_{il} = 1, \quad \forall\, i \in \mathcal{V}, \tag{4.39}$$

I have the following relation

$$\sum_{i\in\mathcal{V},\, l\in L} H_k^t(i,l) x_{il} = H_k^t(\mathbf{y}), \tag{4.40}$$

and the pseudo-Boolean function $\psi(\mathbf{x})$ and the energy function $E_{\mathcal{V}}(\mathbf{y})$ is related by $\psi(\mathbf{x}) - \sum_t \sum_k \sum_{i\in\mathcal{V}} \xi_i = -E_{\mathcal{V}}(\mathbf{y})$. I further note that $\sum_{i\in\mathcal{V},\, l\in L} H_k^t(i,l) x_{il}$ in (4.40) is a monotonically increasing *modular* function since $H_k^t(i,l) \geq 0$.

We can now prove that the set function $\psi$ is submodular. From Lemmas 4.1 and 4.2, I know that the second and fifth terms in (4.37) are submodular. In addition,

the first and fourth terms in (4.37) are modular, which are also submodular. The third term is simply a constant. Since submodularity is preserved under canonical combinations, $\psi$ is submodular.

I now show that $\psi$ is monotonically increasing. Let $g : 2^S \to \mathbb{R}$ be the corresponding set function of the pseudo-Boolean function $\psi$. The marginal gain of $g$, shown below, is always nonnegative for all $A \in \mathcal{I}$ and $(i,l) \in S \setminus A$.

$$
\begin{aligned}
&g(A \cup \{(i,l)\}) - g(A) = \\
&\xi_i + H_k^r \Bigg( \log \big( H_k^t(i,l) + \sum_{(j,m)\in A} H_k^t(j,m) \big) - \log \big( \sum_{(j,m)\in A} H_k^t(j,m) \big) \Bigg) \\
&+ H_k^t(i,l) \log H_k^r \\
&+ \Bigg( - H_k^t(i,l) - \sum_{(j,m)\in A} H_k^t(j,m) \Bigg) \log \Bigg( H_k^t(i,l) + \sum_{(j,m)\in A} H_k^t(j,m) \Bigg) \\
&- \Bigg( \sum_{(j,m)\in A} -H_k^t(j,m) \Bigg) \log \sum_{(j,m)\in A} H_k^t(j,m) \qquad (4.41) \\
\geq &\xi_i + H_k^t(i,l) \log H_k^r \\
&+ \Bigg( - H_k^t(i,l) - \sum_{(j,m)\in A} H_k^t(j,m) \Bigg) \log \Bigg( H_k^t(i,l) + \sum_{(j,m)\in A} H_k^t(j,m) \Bigg) \\
&- \Bigg( \sum_{(j,m)\in A} -H_k^t(j,m) \Bigg) \log \sum_{(j,m)\in A} H_k^t(j,m) \qquad (4.42)
\end{aligned}
$$

Note that the derivation from (4.41) to (4.42) follows the monotonically increasing property of the logarithm function. Let define a set function $z$ given by

$$
z(A) = \Bigg( \sum_{(j,m)\in A} -H_k^t(j,m) \Bigg) \log \Bigg( \sum_{(j,m)\in A} H_k^t(j,m) \Bigg), \qquad (4.43)
$$

which will be used to simplify the notation. Then, $\xi_i$ in (4.38) can be written as

$$
\xi_i = - \min_{l' \in L} \big( H_k^t(i,l') \log H_k^r \big) - z(S) + \max_{l' \in L} z(S \setminus \{(i,l')\}) \qquad (4.44)
$$

Equation (4.42) can then be rewritten as

$$
\begin{aligned}
&= H_k^t(i,l)\log H_k^r - \min_{l' \in L}\left(H_k^t(i,l')\log H_k^r\right) + \\
&\quad z(A \cup \{(i,l)\}) - z(A) - z(S) + \max_{l' \in L} z(S \setminus \{(i,l')\}) && (4.45)\\
&\geq z(A \cup \{(i,l)\}) - z(A) - z(S) + \max_{l' \in L} z(S \setminus \{(i,l')\}) && (4.46)\\
&\geq z(A \cup \{(i,l)\}) - z(A) - z(S) + z(S \setminus \{(i,l)\}) && (4.47)\\
&\geq 0 && (4.48)
\end{aligned}
$$

The derivations from (4.45) to (4.46) and from (4.46) to (4.47) are due to the min and max operations respectively. The last inequality in (4.48) is due to the submodularity of the set function $z$ (Lemma 4.2).

I have shown that the set function $g$ is monotonically increasing and submodular. Using a similar argument as presented in Section 4.3, the energy minimization problem

$$\arg\min_{\mathbf{y}} E_{\mathcal{V}}(\mathbf{y}) \tag{4.49}$$

can be solved by maximizing $g$ subject to the partition matroid constraint

$$\arg\max_{A \in \mathcal{I}} g(A) \tag{4.50}$$

□

# Chapter 5

# Fast Object Localization and Pose Estimation in Heavy Clutter for Robotic Bin Picking

## 5.1 Introduction

Building smarter, more flexible, and independent robots that can interact with the surrounding environment is a fundamental goal of robotics research. Potential applications are wide-ranging, including automated manufacturing, entertainment, in-home assistance, and disaster rescue. One of the long-standing challenges in realizing this vision remains the difficulty of "perception" and "cognition"—i.e., providing the robot with the ability to understand its environment and make inferences that allow it to take appropriate actions. Perception through inexpensive contact-free sensors, such as cameras are essential for continuous and rapid robot operation. In this chapter, I address the challenge of robot perception in the context of industrial robotics.

### 5.1.1 Visual Perception in Industrial Robotics

Computer vision has made rapid progress in the last decade, moving closer to definitive solutions for longstanding problems in visual perception, such as object detection [28, 120, 114], object recognition [33, 95, 32], and pose estimation [1, 108, 87]. While the huge strides made in these fields lead to important realizations, most of these methods cannot be readily adapted to industrial robotics because many of the common assumptions are either violated or invalid in such settings.

**Material properties:** One of the most common assumptions in traditional vision algorithms relates to the characterization of the reflectance of materials in the scene. Most vision algorithms characterize materials as Lambertian [7], i.e., the appearance (radiance) of a single surface point is invariant to the location of the observer (camera). A reasonable assumption in many scenarios, it does not apply well to industrial vision tasks. Several common materials handled in industrial settings such as metal, glass, ceramics, and some plastics differ wildly from Lambertian. Hence, using the Lambertian assumption for such objects tends to result in poor performance. This necessitates industrial robots to possess the ability to understand and make inferences about objects that have complex reflectance characteristics.

**Environmental challenges:** The types of errors that afflict vision-based systems in industrial settings also differ greatly from those in natural environments. Several industrial assembly and manufacturing tasks must be accomplished in dark or dimly lit environments with dust, dirt, grime, and grease. It is essential for vision-based techniques to cope with such sources of error to attain success in such

environments.

**Variable appearance:** The most popular methods for object detection, recognition, and pose estimation are based on the idea of feature descriptors, such as Scale Invariant Feature Transform (SIFT) [81], Histogram of Gradients (HOG) [28], and SURF [8]. The basic idea is to detect several keypoint locations on the surface of each object and compute these feature descriptors at these keypoint locations. The features of each object are then stored in a database. After acquiring a test image, the keypoint locations and the feature descriptors for the test image are computed and matched to the features stored in the database. An appropriately computed matching score is used to detect and recognize objects, and the geometric relationship between the matched keypoint locations in the test image and the database are used to make inferences about the objects' poses. This general principle is quite popular, as seen in several object recognition methods [110, 68, 93]. Unfortunately, this successful paradigm cannot be adapted easily to industrial robotics because visual appearance features are unreliable in industrial settings. Variable material properties, as well as uncontrolled illumination and environmental conditions, make appearance-based descriptors unreliable and preclude the use of such techniques in most industrial applications.

**Background Clutter:** The objects in a factory environment are usually stacked in part containers, which produces additional challenges such as overlapping parts, occlusions, cast shadows, and complex backgrounds. Therefore, most commercial vision systems assume that parts are separated in a kitting stage before operation. Machine vision systems capable of handling clutter, occlusions, and com-

plex backgrounds would eliminate the need for kitting stages, allowing such systems to handle a complex bin of parts.

**Model-based estimation:** While industrial settings have the challenging above-mentioned factors, they are also more structured and allow opportunities to exploit this structure. For example, 3D CAD models of most of the industrial parts are readily available. Even if some of them are not, the fact that most industrial assembly lines repeatedly handle a finite set of discrete parts many times (on the order of millions) makes CAD model acquisition cost-effective. The 3D CAD models provide a reliable source of information, potentially overcoming the challenging reflectance and environmental conditions.

### 5.1.2 A Practical Vision-Based Robotic Bin-Picking System

In this chapter, I present a practical vision-based robotic bin-picking system that overcomes the challenges described in Section 5.1.1. The system performs detection and estimation of the 3D poses of objects that are stacked in a part container, retrieves the parts from the container using an industrial robot arm, performs pose verification and refinement while holding the part in the gripper, and inserts the chosen part at a designated position. I have introduced two novel ideas that allow us to achieve a reliable, fast, and accurate operation: (1) Novel imaging hardware that provides reliable geometric features regardless of the object's material and surface characteristics; (2) Fast, robust, and accurate 3D pose estimation based on the Fast Directional Chamfer Matching (FDCM) algorithm.

The fundamental challenges that arise due to non-Lambertian materials (e.g., metal, glass, ceramic), textureless parts (e.g., uniformly painted parts), and greasy and dirty environments lead to the fact that photometric features such as color and appearance descriptors are not sufficiently robust in industrial settings. This motivates the need to develop features that are dependent on the geometry of the part rather than its photometry. I use an inexpensive camera design, the multi-flash camera (MFC) [99], that provides reliable geometric features: depth edges. The location of depth edges on an object is dependent only on the pose of the object with respect to the observer and the object geometry. Therefore, depth edges can be used to determine the pose of the object uniquely. In addition, these geometric descriptors allow easy and efficient incorporation of the 3D CAD models into my system. Given the 3D CAD model of an object, I retrieve the object's pose that provides the best match between the observed features and the depth edges from the CAD model. This allows us to bypass a time-consuming training phase for each of the objects that would otherwise be necessary. A new part can be integrated into my system in less than 10 minutes.

Although many shape matching algorithms have been proposed over the decades, chamfer matching (CM) [6] remains among the fastest and most robust approaches in the presence of clutter. I adapt traditional chamfer matching with a host of techniques to improve reliability, accuracy, and speed. First, I exploit the geometric redundancies in the 3D structure of industrial parts by approximating the edge features using line segments. This, along with a 3D integral distance transform representation, allows us to both reduce the memory footprint and speed the matching

algorithm by orders of magnitude. Second, I incorporate a directional error term in the distance transform definition that significantly improves the reliability, robustness, and accuracy of matching. Further, I improve the pose estimation accuracy using a continuous optimization procedure. The resulting system is capable of real-time operation for several industrial assembly applications.

While the primary goal of this research is to develop a robust and reliable vision system for industrial robotics, the FDCM algorithm also achieves state-of-the-art performance in shape matching. I present two additional application domains that benefit from FDCM: deformable object detection using a hand-drawn shape, and human pose estimation.

The chapter is organized as follows. I briefly overview the related literature in Section 5.2. I present the shape matching algorithm and its optimization in Section 5.3. Pose estimation and the robotic bin-picking system are described in Section 5.4. I report on my extensive experimental validation of the proposed system and compare it to the state of the art in Section 5.5. The chapter concludes in Section 5.6. An earlier version of this chapter was published in [76, 77].

## 5.2 Related Work

In recent decades, a considerable amount of work has occurred on automating the process of part assembly using vision systems [105, 90, 2]. Though vision systems can successfully in identify, inspect, and locate parts in carefully engineered manufacturing settings, it remains a great challenge to extend their applicability

to more general, unconstrained settings. Also, they often make use of simple geometric features such as lines, circles, or ellipses and their spatial organization [121]. Changing a target object in the assembly process would require significant manual adjustments or algorithmic modifications.

**Model-based vision systems** exploit 3D CAD models of objects, along with either acquired 2D images or range sensor data, for image interpretation. Such methods provide means for efficient detection, recognition, and pose estimation of objects in cluttered environments [79, 80, 109, 59, 5, 31]. Methods such as [79, 80, 30] rely on establishing correspondences between 2D image features and points in the 3D models in order to obtain an initial estimate of object pose. The estimate is later refined using iterative algorithms. These correspondences are, however, difficult to obtain reliably.

Since establishing 3D-to-2D correspondences using images is a difficult task, several systems rely either directly or indirectly on 3D information. Such methods greatly simplify the correspondence problem at the cost of increased hardware requirements. The most common approach use a 3D range sensors either based on structured light [103] or on time of flight [27]. This provides the ability to establish 3D-to-3D point correspondences by matching 3D point descriptors from the CAD model to those in the acquired point cloud data. Several 3D point descriptors [109, 59, 5] have been proposed for matching the 3D scene points to the model points. To remove false matches, an interpretation tree procedure [47] can be applied to find mutually consistent pairs. With these consistent pairs, one can use Horn's method [52] to estimate the object's 3D pose. Unfortunately, these descriptors are

less reliable for pose estimation of industrial parts because these objects are mostly made of planar surfaces, which leads to very few and uninformative features.

Recently, the use of the multi-flash camera [99] for object pose estimation was proposed in [2]. The MFC, which was originally developed in the context of non-photorealistic rendering, provides depth edge features that can be used for pose estimation tasks. In this chapter, I significantly expand the scope and impact of MFC for industrial robotics. While [2] presented a system capable of handling isolated parts, here I present a system that can handle multitudes of parts randomly placed in a cluttered bin. Further, I also present a novel shape-matching algorithm that results in better accuracy and orders-of-magnitude improvement in matching speed, allowing real-time system performance.

One of the main technical contributions of this work is the development of the fast directional chamfer matching algorithm, which is widely applicable to several problems that currently use shape matching. Below, I briefly discuss related approaches in shape matching.

**Shape matching** has been an active area in robotic vision research. Several authors have proposed shape representations and similarity measures that aim to be invariant to object deformations [12, 74]. These methods actively handle intra-class shape variations and achieve good performance in object recognition. However, they require a clean segmentation of the target object. This renders them less suitable for dealing with unstructured scenes due to the difficulty in foreground-background separation.

Recent studies focus on the recognition and localization of object shapes in

cluttered images. In [13], the shape matching problem is posed as finding the optimal correspondences between feature points, which leads to an integer quadratic programming problem. In [39], a contour segment network framework is proposed in which shape matching is formulated as finding paths on the network similar to model outlines. In [37], Ferrari et al. propose a family of scale-invariant local shape descriptors (pair-of-adjacent-segment features) formed by $k$-connected nearly straight contour fragments in the edge map. These descriptors are later utilized in a shape matching framework [38] through a voting scheme on a Hough space.

Zhu et al. [135] formulate shape detection as a subset selection problem on a set of salient contours. Due to the NP-hardness of the selection problem, they compute an approximate solution using a two-stage linear programming procedure. In [35], a hierarchical object contour representation is proposed to model shape variation, and the matching is performed using dynamic programming. In [100], a multi-stage approach is employed in which coarse detections, which are established by matching subsets of contour segments, are pruned by building the entire contour using dynamic programming.

These algorithms yield impressive results for matching shapes in cluttered images. However, they share a common drawback, high computational complexity, which makes them unsuitable for time-critical applications. Although proposed decades ago, chamfer matching [6] remains the preferred method when speed and accuracy are required, as discussed in [112]. In this chapter, I propose an improved version of chamfer matching and demonstrate its superiority with respect to other variants [45, 106]. My approach improves the accuracy of chamfer matching while

greatly reducing its time complexity, leading to a speed increase of up to two orders of magnitude in several application scenarios.

## 5.3 Fast Directional Chamfer Matching

In this section, I introduce my *fast directional chamfer matching* algorithm, which I use for object detection and pose estimation in industrial robotics and other application areas.

### 5.3.1 Chamfer Matching

First, I briefly explain standard chamfer matching (CM) [6], a popular technique for finding the best alignment between a template edge map and a query edge map. Let $U = \{\mathbf{u}_i\}$, where $i = 1, 2, ..., |U|$, be the set of edge pixels from a template edge map, and let $V = \{\mathbf{v}_j\}$, where $j = 1, 2, ..., |V|$, be the set of edge pixels from a query image edge map. The *chamfer distance* between $U$ and $V$ is defined as the average over all pixels $\mathbf{u}_i \in U$ of the distance between $\mathbf{u}_i$ and its nearest pixel in $V$:

$$d_{\mathrm{CM}}(U, V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \min_{\mathbf{v}_j \in V} \|\mathbf{u}_i - \mathbf{v}_j\|. \tag{5.1}$$

where $n$ is the number of template edge pixels, $n = |U|$.

Let $W$ be a warping function defined on the image plane that is parameterized by $\mathbf{s}$. For instance, if $W$ is a 2D Euclidean transformation, then $\mathbf{s} \in \mathrm{SE}(2)$ can be written as $\mathbf{s} = (\theta,\ \bar{t}_x,\ \bar{t}_y)$, where $\bar{t}_x$ and $\bar{t}_y$ are translations parallel to the $x$ and $y$ axes, respectively, and $\theta$ is the in-plane rotation angle. Its action on each image

point $\mathbf{x} \in \mathbb{R}^2$ is given via the transformation

$$W(\mathbf{x}; \mathbf{s}) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \mathbf{x} + \begin{pmatrix} \bar{t}_x \\ \bar{t}_y \end{pmatrix}. \tag{5.2}$$

The best alignment parameter $\hat{\mathbf{s}}$ between the two edge maps is then given by

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \mathrm{SE}(2)} d_{\mathrm{CM}}(W(U; \mathbf{s}), V) \tag{5.3}$$

where $W(U; \mathbf{s}) = \{W(\mathbf{u}_i, \mathbf{s})\}$, $i = 1, 2, ..., |U|$.

The chamfer matching cost can be computed efficiently using the distance transform image

$$\mathrm{DT}_V(\mathbf{x}) = \min_{\mathbf{v}_j \in V} \|\mathbf{x} - \mathbf{v}_j\|, \tag{5.4}$$

which specifies the distance from each pixel $\mathbf{x}$ in the distance transform image to the nearest edge pixel in $V$. The distance transform can be computed in two passes over the image using dynamic programming [34]. Using the distance transform, the cost function (5.1) can be evaluated in linear time $O(n)$ via

$$d_{\mathrm{CM}}(U, V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \mathrm{DT}_V(\mathbf{u}_i). \tag{5.5}$$

Chamfer matching provides a fairly smooth measure of fitness and can tolerate small rotations, misalignments, occlusions, and deformations. However, it becomes less reliable in the presence of background clutter due to an increase in the proportion of false correspondences. To improve its robustness, several variants of chamfer matching have been introduced that exploit edge orientation information. In [45, 29], the template and query image edges are quantized into discrete orientation channels, and individual matching scores across channels are summed. Although

these methods improve performance in cluttered scenes, the cost function is sensitive to the number of orientation channels and becomes discontinuous across channel boundaries. In [106], the chamfer distance is augmented with an additional cost for orientation mismatch, which is given by the average difference in orientations between template edges and their nearest edge points in the query image. The method is known as oriented chamfer matching (OCM).

### 5.3.2 Directional Chamfer Matching

Instead of an explicit formulation of the orientation mismatch, I generalize the chamfer distance to points in $\mathbb{R}^3$ to match directional edge pixels. Each edge pixel $\mathbf{x}$ is augmented with a direction term, $\phi(\mathbf{x})$, and the directional chamfer matching (DCM) score is given by

$$d_{\mathrm{DCM}}(U, V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \min_{\mathbf{v}_j \in V} \left( \|\mathbf{u}_i - \mathbf{v}_j\| + \lambda \|\phi(\mathbf{u}_i) - \phi(\mathbf{v}_j)\|_\pi \right) \tag{5.6}$$

where the parameter $\lambda$ is a weighting factor between the location and orientation terms. To compute the direction terms, I fit line segments to the edge points (as explained in Section 5.3.3), and $\phi(\mathbf{x})$ is the orientation of the line segment associated with point $\mathbf{x}$. Note that the directions are written modulo $\pi$: $0 \leq \phi(\mathbf{x}) < \pi$, and the orientation error is defined as the minimum circular difference between the two directions:

$$\|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)\|_\pi = \min \left\{ |\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)|, \left| \, |\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)| - \pi \right| \right\}. \tag{5.7}$$

In Figure 5.1, I illustrate the differences between DCM and OCM [106]. The proposed matching cost, DCM, is a piecewise smooth function of both the translation

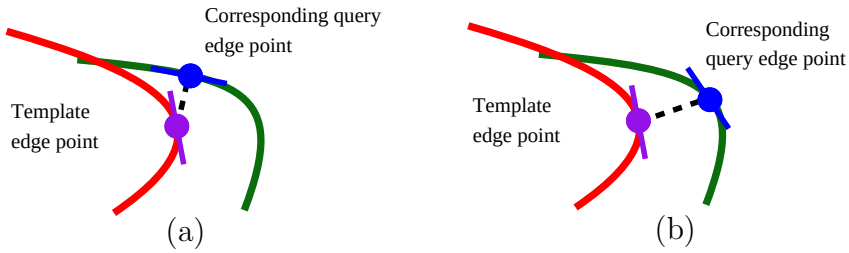



Figure 5.1: Matching costs for an edge point. (a) Oriented chamfer matching (OCM) [106]. (b) Directional chamfer matching (DCM, proposed in this chapter). Whereas in OCM the location error is augmented with the orientation difference from the nearest edge point, DCM jointly minimizes location and orientation errors.

$(t_x, t_y)$ and the rotation $(\theta)$ of the template pose. It is more robust to clutter, missing edges, and small misalignments.

The computational complexity of existing chamfer matching algorithms is linear in the number of template edge points. Even though DCM includes an additional direction term, my algorithm (derived in this section) computes the *exact* DCM score with *sublinear* complexity.

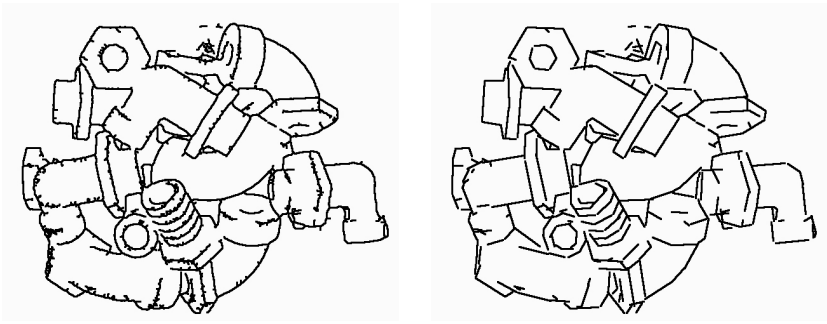
### 5.3.3 Line-Based Representation

The edge map of a scene is not an unstructured binary pattern. On the contrary, the object contours comply with certain continuity constraints that can be retained by combining line segments of various lengths, orientations, and translations. Based on this observation, I represent an edge image as a collection of $m$ line segments. Compared with a set of points which has cardinality $n$, its line-based representation is more concise. Encoding an edge map using the line-based represen-

tation requires only $O(m)$ memory size, where $m << n$, and is particularly suitable when the storage space for templates is limited. When the object exhibits a curved contour, more segments are required for good approximation, but the line-based representation is still more concise than the set of edge pixels.

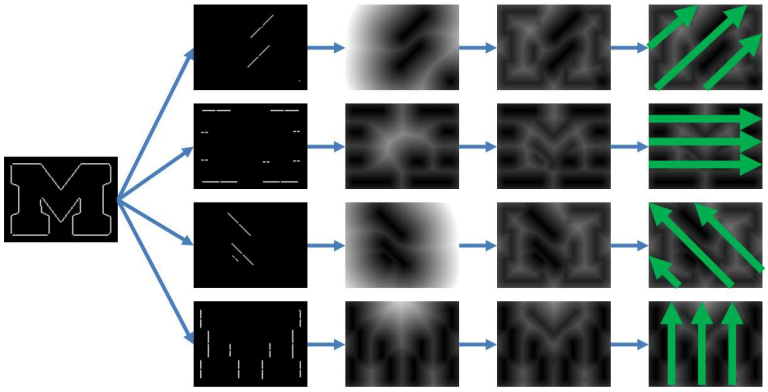
I use a variant of the RANSAC [41] algorithm to compute the line-based representation of an edge map. The outline of the algorithm is as follows. The algorithm initially hypothesizes a variety of line segments by selecting a small subset of edge points and their directions. The support of each line segment is given by the set of points that satisfies the line's equation up to a small residual, $\nu \geq 0$, and form a continuous structure. The line segment with the largest support is retained, and its supporting points are removed from the set of edge points. The procedure is repeated with the reduced set of edge points, until the support of the longest line candidate becomes smaller than a few points.

The algorithm retains only edge points with continuity and sufficient support; therefore, the noise and isolated edges are filtered. In addition, the directions recovered through the line fitting procedure are more precise than would be obtained using local operators, such as image gradients. An example of the line-based representation appears in Figure 5.2, where a set of 11,542 points is modeled with 300 line segments.



(a) (b)

Figure 5.2: Line-based representation. (a) Edge image. The image contains 11,542 edge points. (b) Line-based representation of the edge image. The image contains 300 line segments.



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

Figure 5.3: Computation of the integral distance transform tensor. (a) The set $V$ of points in the query edge map is mapped into a set of line segments through a line-fitting procedure. (b) Edges are quantized into discrete orientation channels. (c) Two dimensional distance transform of each orientation channel. (d) The three-dimensional distance transform, $\mathrm{DT3}_V$, is updated based on the orientation cost. (e) The 3D distance transform is integrated along the discrete edge orientations, and the integral distance transform tensor, $\mathrm{IDT3}_V$, is computed.

### 5.3.4 Three-Dimensional Distance Transform

The matching score given in (5.6) requires finding the minimum matching cost over location and orientation terms for each template edge point. Therefore, the computational complexity of the brute-force algorithm is quadratic in the number of template and query image edge points. Here, I present a three-dimensional distance transform representation ($\mathrm{DT3}_V$) for computing the matching cost in linear time. A similar structure was also used in [96] for fast evaluation of Hausdorff distances.

This representation is a three dimensional image tensor in which the first two dimensions are locations in the image plane and the third dimension belongs to a discrete set of edge orientations. I evenly quantize the edge orientation into $q$ discrete channels, $\hat{\Phi} = \{\hat{\phi}_i\}, i = 1, 2, ..., q$, which evenly divide the range $[0 \quad \pi)$. Each element of the tensor encodes the minimum distance to an edge point in the joint location and orientation space:

$$\mathrm{DT3}_V(\mathbf{x}, \phi(\mathbf{x})) = \min_{\mathbf{v}_j \in V} \left( \|\mathbf{x} - \mathbf{v}_j\| + \lambda \|\hat{\phi}(\mathbf{x}) - \hat{\phi}(\mathbf{v}_j)\|_\pi \right), \tag{5.8}$$

where $\hat{\phi}(\mathbf{x})$ is the nearest quantization level in the orientation space $\hat{\Phi}$ to the edge orientation $\phi(\mathbf{x})$.

I present an algorithm to compute the $\mathrm{DT3}_V$ tensor in $O(q)$ passes over the image by solving two dynamic programs consecutively. Equation (5.8) can be rewritten as

$$\mathrm{DT3}_V(\mathbf{x}, \phi(\mathbf{x})) = \min_{\hat{\phi}_i \in \hat{\Phi}} \left( \mathrm{DT}_{V\{\hat{\phi}_i\}} + \lambda \|\hat{\phi}(\mathbf{x}) - \hat{\phi}_i\|_\pi \right) \tag{5.9}$$

where $\mathrm{DT}_{V\{\hat{\phi}_i\}}$ is the two dimensional distance transform of the edge points in $V$ that have edge orientation $\hat{\phi}_i$.

Initially, I compute $q$ two-dimensional distance transforms $\mathrm{DT}_{V\{\hat{\phi}_i\}}$, which requires $O(q)$ passes over the image using the standard distance transform algorithm [34]. Subsequently, the $\mathrm{DT3}_V$ tensor (5.9) is computed by using a second dynamic program for each image pixel separately. The tensor is initialized with the two dimensional distance transforms, $\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i) = \mathrm{DT}_{V\{\hat{\phi}_i\}}(\mathbf{x})$, and is updated with a forward recursion

$$\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i) = \min\{\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i), \mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_{i-1}) + \lambda\|\hat{\phi}_{i-1} - \hat{\phi}_i\|_\pi\} \tag{5.10}$$

and a backward recursion

$$\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i) = \min\{\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_i), \mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_{i+1}) + \lambda\|\hat{\phi}_{i+1} - \hat{\phi}_i\|_\pi\} \tag{5.11}$$

for each pixel $\mathbf{x}$. Unlike the standard distance transform algorithm, special handling is required for the circular orientation. The forward and backward recursions do not terminate after a full cycle, $i = 1, \ldots, q$ or $i = q, \ldots, 1$ respectively, but the values of the tensor entries continue to be updated in a circular form until the value for a tensor entry is not changed. Note that, at most, 1.5 cycles are needed for each of the forward and backward recursions, therefore the worst time computational cost is $O(q)$ passes over the image. I illustrate the computation of the three dimensional distance transform in Figure 5.3(a)–(d). Using $\mathrm{DT3}_V$, the directional chamfer matching score of any template $U$ can be computed as

$$d_{\mathrm{DCM}}(U, V) = \frac{1}{n} \sum_{\mathbf{u}_i \in U} \mathrm{DT3}_V(\mathbf{u}_i, \hat{\phi}(\mathbf{u}_i)), \tag{5.12}$$

where the complexity is linear in $n$, the number of edge points in $U$.

### 5.3.5 Integral Distance Transform Tensor

Let $l_{[\mathbf{x}_1,\mathbf{x}_2]}$ represent the line segment in the image plane connecting pixels $\mathbf{x}_1$ and $\mathbf{x}_2$. Let $L_U = \{l_{[\mathbf{s}_j,\mathbf{e}_j]}\}$, $j = 1,\ldots,m$, be the line-based representation of template edge points $U$, where $\mathbf{s}_j$ and $\mathbf{e}_j$ are the start and end locations of the $j$th line segment, respectively. For ease of notation, I sometimes refer to a line segment with only its index, $l_j = l_{[\mathbf{s}_j,\mathbf{e}_j]}$. I assume the line segment directions are restricted to $q$ discrete channels $\hat{\Phi}$, which is enforced in the line-based representation. I choose the number of directions $q$ large enough (in my experiments, $q = 60$) to avoid quantization artifacts. The line-based representation of Figure 5.2(b) is generated from the edge image in Figure 5.2(a) using $q = 60$ directions.

Since the edge points in a line segment all have the same orientation, which is the direction of the line segment $\hat{\phi}(l_j)$, the directional chamfer matching score (5.12) can be rearranged as

$$d_{\mathrm{DCM}}(U,V) = \frac{1}{n}\sum_{l_j \in L_U}\sum_{\mathbf{u}_i \in l_j} \mathrm{DT3}_V(\mathbf{u}_i, \hat{\phi}(l_j)). \tag{5.13}$$

In this formulation, the $k$th orientation channel of the $\mathrm{DT3}_V$ tensor, $\mathrm{DT3}_V(\mathbf{x}, \hat{\phi}_k)$, is only used for evaluating the matching scores of the line segments having the direction $\hat{\phi}_k$, achieved by summing over the points in the line segments.

Integral images are intermediate image representations used for fast calculation of region sums [120] and linear sums [9]. Here, I present an integral distance transform representation ($\mathrm{IDT3}_V$) to evaluate the summation of costs over any line segment in $O(1)$ operations, as shown in Figure 5.3(e).

Let $\mathbf{x}_0$ be the intersection of an image boundary with the line that passes

through $\mathbf{x}$ and has direction $\hat{\phi}_i$. Each entry of the $\mathrm{IDT3}_V$ tensor is given by

$$\mathrm{IDT3}_V(\mathbf{x}, \hat{\phi}_i) = \sum_{\mathbf{x}_j \in l_{[\mathbf{x}_0, \mathbf{x}]}} \mathrm{DT3}_V(\mathbf{x}_j, \hat{\phi}_i). \tag{5.14}$$

The $\mathrm{IDT3}_V$ tensor can be computed in one pass over the $\mathrm{DT3}_V$ tensor. Using this representation, the directional chamfer matching score of any template $U$ can be computed in $O(m)$ operations via

$$d_{\mathrm{DCM}}(U, V) = \frac{1}{n} \sum_{l_{[\mathbf{s}_j, \mathbf{e}_j]} \in L_U} \mathrm{IDT3}_V(\mathbf{e}_j, \hat{\phi}(l_{[\mathbf{s}_j, \mathbf{e}_j]})) - \mathrm{IDT3}_V(\mathbf{s}_j, \hat{\phi}(l_{[\mathbf{s}_j, \mathbf{e}_j]})). \tag{5.15}$$

### 5.3.6 Search Optimization

In this section I present two search optimization techniques based on the bounds on the matching cost, and empirically show that the number of evaluated line segments is *sublinear* in the number of template points $n$.

#### 5.3.6.1 Bound in the Hypotheses Domain

The $O(m)$ complexity is only an upper bound on the number of computations. FDCM can be used for object detection and for localization. For object detection, I need only to retain the hypotheses for which the template matching cost is less than a detection threshold. For localization, I need only to retrieve the hypothesis with the lowest matching cost.

I order the template line segments with respect to their lengths and start the summation (5.15) from the longest line segment. A hypothesis is eliminated during the summation if the cost is greater than the detection threshold or the current best

hypothesis. Since the lengths of the line segments roughly decay exponentially, for most of the hypotheses only a few arithmetic operations are performed.

#### 5.3.6.2 Bound in the Spatial Domain

The DCM cost function (5.6) is smooth and bounded in the spatial domain. I utilize this fact to reduce significantly the number of hypotheses evaluated. Let $\boldsymbol{\delta} \in \mathbb{R}^2$ be a translation of the model $U$ in the image plane. The DCM cost variation due to translation is given by

$$\begin{aligned} d_{\mathrm{DCM}}(U+\boldsymbol{\delta}, V) &= \frac{1}{n} \sum_{\mathbf{u}_i \in U} \min_{\mathbf{v}_j \in V} \|\mathbf{u}_i + \boldsymbol{\delta} - \mathbf{v}_j\| + \lambda \|\phi(\mathbf{u}_i) - \phi(\mathbf{v}_j)\|_\pi \\ &\leq \frac{1}{n} \sum_{\mathbf{u}_i \in U} \min_{\mathbf{v}_j \in V} \|\mathbf{u}_i - \mathbf{v}_j\| + \|\boldsymbol{\delta}\| + \lambda \|\phi(\mathbf{u}_i) - \phi(\mathbf{v}_j)\|_\pi = \|\boldsymbol{\delta}\| + d_{\mathrm{DCM}}(U, V). \end{aligned} \tag{5.16}$$

From Equation (5.16), the cost variation is bounded by the spatial translation $|d_{\mathrm{DCM}}(U+\boldsymbol{\delta}, V) - d_{\mathrm{DCM}}(U, V)| \leq \|\boldsymbol{\delta}\|$. If the detection threshold is $\tau$ and the cost of the current hypothesis is $\psi > \tau$, then a target cannot reside within the $\|\boldsymbol{\delta}\| = |\psi - \tau|$ pixel range that has a matching cost lower than the detection threshold. Therefore, I can skip the evaluation of the hypotheses within this region.

#### 5.3.6.3 Empirical Evidence of Sublinear Complexity

Clearly, the sublinear complexity holds in the case of scaling the template shape. As the number of edge points, $n$, increases with the template size, the cardinality $m$ of the line-based representation of the template remains the same. Hence, the same number of arithmetic operations is required to compute the matching cost, which means the matching complexity remains constant irrespective of the number

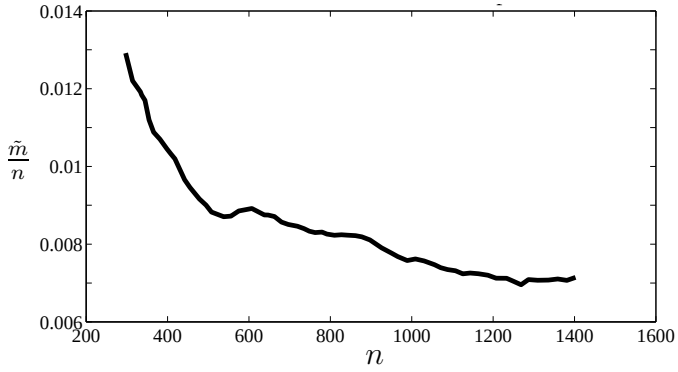



Figure 5.4: Empirical evidence of sublinear time complexity in the number of template points. The graph plots the ratio of the number of evaluated lines $\tilde{m}$ to the number of template points $n$ vs. the number of template points. (If the complexity were linear in $n$, the graph would be a horizontal line.)

of edge points.

I also provide empirical evidence that in a more general setup, as well, the matching complexity is a sublinear function of the number of template points. As explained above, the $O(m)$ complexity is only an upper bound on the number of evaluations, and, on average, I need to evaluate only a fraction of the $m$ lines. Empirically, I evaluate the $\tilde{m}$ longest lines, where $\tilde{m}$ is chosen to fit $20\% - 30\%$ of the template points. Most of the energy is concentrated in only a few lines, and I find that $\tilde{m}$ grows sublinearly with $n$. In Figure 5.4, I plot the number of template points, $n$, on the $x$-axis and the ratio of the number of evaluated lines to the number of template points, $\frac{\tilde{m}}{n}$, on the $y$-axis. For this graph, $\tilde{m}$ is selected as the number of lines that fit 30% of the template points. The curve is generated using 1,000 shape images from the MPEG-7 dataset. I observe that as the number of template points increases, the fraction of evaluated lines decreases, which provides empirical

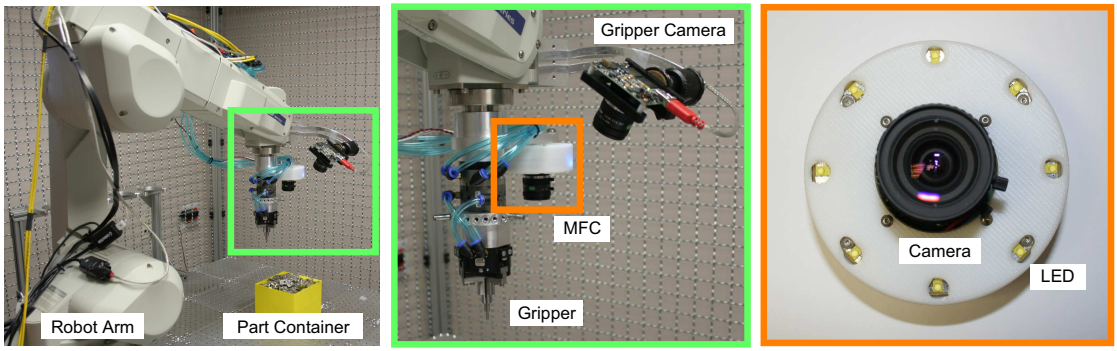



Figure 5.5: My robotic grasping system. A multi-flash camera (MFC), shown in detail in (c), is mounted on the robot arm and used to perform detection and pose estimation of objects (parts) placed in a container. The gripper camera is a standard camera mounted above the robot's wrist joint and aimed at the tip of the gripper. The gripper camera performs error detection after an object is retrieved from the container.

evidence that the algorithm is sublinear in the number of template points $\big( < O(n) \big)$.

## 5.4 Pose Estimation for Robotic Bin Picking

In this section, I present my robotic bin-picking system that uses the shape matching algorithm described in Section 5.3.

### 5.4.1 System Overview

Figure 5.5 shows my system configuration. I mount an MFC and a standard camera on an industrial robot arm. The MFC performs object detection and pose estimation of objects that are randomly arranged in a part container. The robot arm

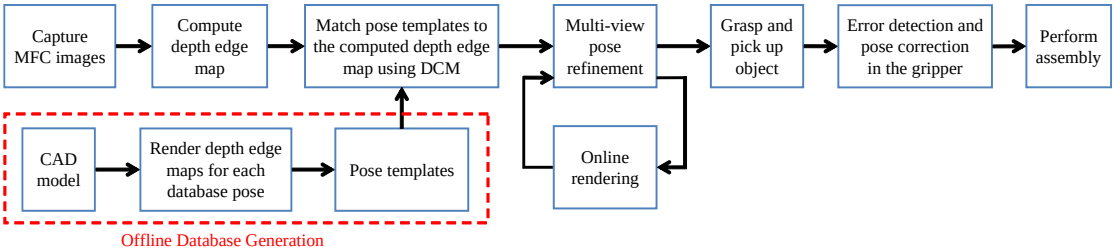



Figure 5.6: System flowchart.

uses the object's estimated pose to grasp the object and lift it out of the container. The standard camera, the gripper camera, focuses on the tip of the gripper performs error detection after the object is retrieved. Both cameras are calibrated offline using a checkerboard. The calibration determines internal parameters of the cameras, as well as the poses of the cameras with respect to the robot coordinate system (hand-eye calibration).

The flowchart in Figure 5.6 provides a summary of my system. I give an overview of my algorithm below and explain the details of each process in the following subsections.

1. **Offline database generation** (Section 5.4.3): For each object, I render the 3D CAD model according to a set of hypothesized poses, extract depth edges, and compute the line-based representation (which was presented in Section 5.3.3) of the depth edges.

2. **MFC imaging and depth edge extraction** (Section 5.4.2): I capture nine images, using the eight different flashes of the MFC and one image without any flash. The depth edges in the scene are computed using these images.

3. **Object detection and pose estimation**: Using the FDCM algorithm (which was presented in Section 5.3), I retrieve the database pose and its in-plane transformation parameters that have the minimum matching cost and use them as a coarse pose estimate. The matching algorithm is accelerated using a heuristic called *one-dimensional search* (Section 5.4.4). Further improvement of the coarse estimate is achieved via a multi-view pose refinement algorithm (Section 5.4.5).

4. **Grasping and picking up the object**: I use the estimated 3D pose to grasp the object with the gripper and lift it out of the part container.

5. **Error detection and pose correction** (Section 5.4.6): I use the gripper camera to detect grasping errors. I evaluate/re-estimate the object pose in the gripper. The object pose is corrected if necessary.

6. **Assembly**: The pose-corrected object is ready for the next step of the assembly task.

### 5.4.2 MFC Imaging and Depth Edge Extraction

I use an MFC [99] to detect depth edges (depth discontinuities) in the scene. A depth edge is a robust geometric feature. It is invariant to the surface properties of objects (textured, textureless, shiny, etc.) and is unaffected by oil, grime, or dirt on the object surface, which are common in industrial environments. The MFC is equipped with eight point light sources made of light-emitting diodes (LEDs). They are evenly distributed around the camera in a circular fashion, as shown in

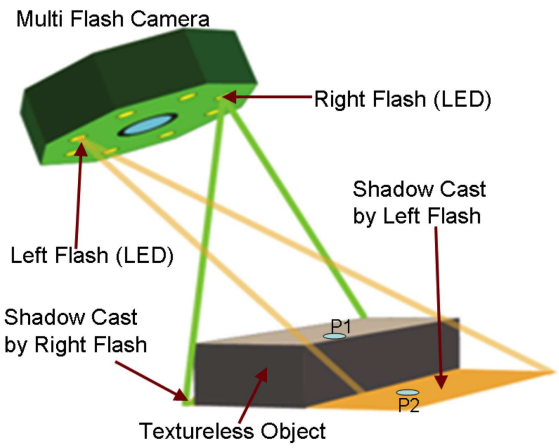



Figure 5.7: Illustration of the principle of multi-flash camera (MFC) imaging. The scene is illuminated by flashing one LED at a time. Due to the different positions of the LEDs, the shadows cast by the object change. While intensity values of points such as $P1$ (on the top surface of the object) remain nearly constant when illuminated by different LEDs, intensity values of points such as $P2$ (which is in shadow for some of the LEDs) change. This property is exploited to detect depth edges.

Figure 5.7. During the MFC imaging, these LEDs are sequentially switched on to illuminate the scene. Only one LED lights at a time, and an image is taken. This is repeated for each of the eight LEDs. I also take an image with all the LEDs turned off to record the ambient illumination. The various LED positions result in different illumination directions, so the positions of shadows change across the eight images. This property can be exploited to detect the depth edges in the scene, as discussed below.

Let $I_i$ denote the image illuminated by the $i$th LED, after subtracting the

(a) Flash from right (b) Flash from left (c) Depth edges (d) Canny edges

Figure 5.8: Comparison between depth edges extracted using an MFC and standard Canny edges for a simple scene (top) and a highly cluttered scene (bottom). (a, b) Two out of eight flash images captured with an MFC. Note the different shadow locations. (c) Extracted depth edges. (d) Standard intensity edges computed by using a Canny edge detector on an image captured without flash. Note that the Canny edge results include both texture and depth edges and are affected by shadows due to ambient lights.

ambient image. First, I construct the maximum image, $I_{\max}$, where the shadows cast by the flashes are removed. I consider each pixel location and find the maximum intensity value at that location across the eight images:

$$I_{\max}(x, y) = \max_i I_i(x, y). \tag{5.17}$$

Next, I compute the ratio images

$$RI_i = \frac{I_i}{I_{max}}. \tag{5.18}$$

Ideally, if a pixel exists in a shadow region of image $I_i$ (e.g., point $P2$ in Figure 5.7), the ratio should be 0 because the contribution of the illumination from the ambient

source has been removed. In contrast, the ratio in other non-shadow regions (e.g., point $P1$ in Figure 5.7) should be close to 1 because these regions are illuminated by all the flashes. Notice that a depth edge corresponds to a point of transition from a non-shadow region to a shadow region along the illumination direction defined by the LED position in each image. Therefore, for each ratio image, I detect the transition using a Sobel filter whose direction is aligned with the illumination direction, followed by non-maximum suppression. I then add the filter responses across different flash images and use hysteresis thresholding similar to the Canny edge detector [23] to obtain a depth edge image.

**Comparison with Intensity Edges:** Figure 5.8 compares depth edges extracted using an MFC to standard intensity edges, which were computed by using a Canny edge detector on an image captured without flash. Note that the Canny edge results include texture edges (e.g., the artificially painted object surface in the top row, and small scratches on the surface of the shiny objects in the bottom row). They are also affected by shadows due to ambient light (note the difference of detected edge locations between the MFC depth edge results and the Canny edge results). In contrast, my approach using MFC imaging provides depth edges only, which can be used as robust geometric features for object detection and pose estimation.

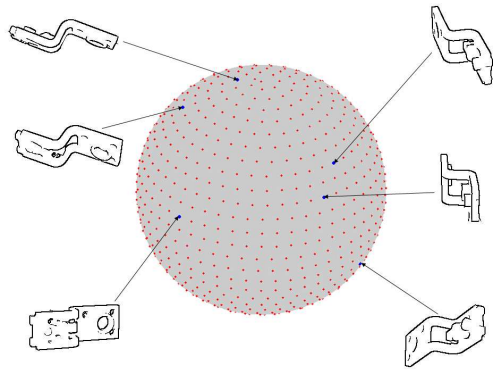


Figure 5.9: Database generation. I uniformly sample the rotation angles ($\theta_x$ and $\theta_y$) on the 2-sphere. Rendering the CAD model of the object at each of the sampled rotations generates the template database.

### 5.4.3 Database Generation

An object exhibits different silhouettes in different poses. Although the matching algorithm in Section 5.3 models in-plane rotation and translation, it does not model rotations in depth, which can change an object's depth-edge silhouette. To accommodate these variations, I generate a set of templates across the range of possible rotations in depth, denoting this set of templates by $\{U_k\}$. The search problem in (5.3) is generalized to find the best-matching template in this set, as follows:

$$\arg \min_{k, \mathbf{s} \in \mathrm{SE}(2)} d_{DCM}(W(U_k; \mathbf{s}), V). \tag{5.19}$$

Given a CAD model of the object, I generate a database of depth-edge templates by detecting the depth discontinuities in the model. In this simulation, a virtual camera with the same internal parameters as the real camera is placed at the origin, and its optical axis is aligned with the $z$-axis of the world coordinate system. The CAD model of the object is then placed on the $z$-axis at a distance $t_z$

from the virtual camera, which is equal to the actual distance of the part container from the real MFC in the setup. The virtual flashes are switched on individually, acquiring eight renderings of the object (including cast shadows). The depth edges are detected using the procedure described in Section 5.4.2.

An arbitrary 3D rotation can be decomposed into a sequence of three elemental rotations about three orthogonal axes. I align the first of these axes to the camera optical axis and refer to the rotation about this axis as *in-plane rotation* ($\theta_z$). The other two axes are on a plane perpendicular to the camera optical axis, and I call the rotation about these two axes *out-of-plane rotation* ($\theta_x$ and $\theta_y$). Note that an in-plane rotation results simply in an in-plane rotation of the observed images, and the effect of an out-of-plane rotation depends on the 3D structure of the object. Due to this distinction, I only include out-of-plane rotations of the object in the database. I sample $K$ out-of-plane rotations ($\theta_x$ and $\theta_y$) uniformly on the 2-sphere, $S^2$, as shown in Figure 5.9, and generate the depth-edge template $U_k$ for each rotation $k \in \{1, \ldots, K\}$.

### 5.4.4 One-dimensional Search

To retrieve the object's coarse pose in the scene using (5.19), I sequentially search all the database templates, $U_k$, where $k = 1, \ldots, K$. For each template $U_k$, searching for the best alignment $\hat{\mathbf{s}} = (\theta_z, \bar{t}_x, \bar{t}_y)$ is computationally intensive (three degrees of freedom). I present a heuristic method to reduce the search space greatly (from three degrees to one degree of freedom), which exploits the fact that under

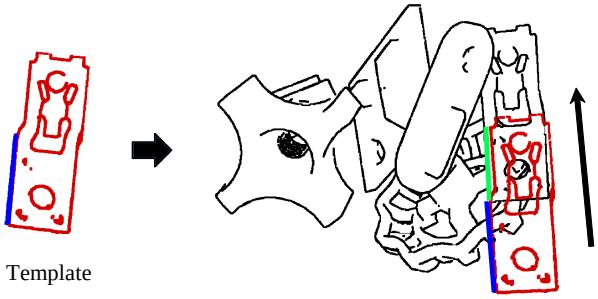



Figure 5.10: One-dimensional search. A template is rotated and translated such that one template line segment (the blue line segment) is aligned with one line segment in the query image (the green line segment). The template is translated along the query line segment, and the directional chamfer matching cost is evaluated for each translation.

the best alignment, the template and query image line segments are well aligned. Additionally, the major lines of the template and the query images are reliably detected during the line-fitting process, since the algorithm favors line segments with larger support.

I order the sets of template and query line segments from longest to shortest, and use a few major lines from the ordered sets of template and query line segments to guide the hypothesis search. The template is initially rotated and translated so a template line segment is aligned with the direction of a query image line segment and the template line's end point is translated to match the start point of the query line segment, as illustrated in Figure 5.10. The template is then translated along the query line segment direction, and the cost function is evaluated only at locations where an overlap occurs between two segments. This procedure reduces the three-

dimensional search (in-plane rotation and translation) to a one-dimensional search along only a few directions. The search time is invariant to the image size and is only a function of the number of templates and query image lines and their lengths. With this heuristic, I can efficiently find the minimum-cost template and its alignment parameters.

### 5.4.5 Multi-View Pose Refinement

The minimum-cost template, together with its in-plane transformation parameters $(\theta_z,\ \bar{t}_x,\ \bar{t}_y)$, provide a coarse estimate of the 3D object pose. Let $\theta_x$, $\theta_y$ be the out-of-plane rotation angles, and let $t_z$ be the distance from the camera, which are used to render the template. I back-project the in-plane translation parameters to 3D using the camera calibration matrix $\mathbf{K}$, and obtain the initial 3D pose of the object, $\mathbf{p}^0$, as the three Euler angles $(\theta_x, \theta_y, \theta_z)$ and a 3D translation vector $(t_x, t_y, t_z)^T$. The 3D pose $\mathbf{p}$ can also be written in matrix form

$$\mathbf{M_p} = \begin{pmatrix} \mathbf{R_p} & \mathbf{t_p} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathrm{SE}(3), \tag{5.20}$$

where $\mathbf{R_p}$ is the $3 \times 3$ rotation matrix computed by a sequence of three rotations around the $x$–$y$–$z$ axes, $\mathbf{R}_{\theta_z}\mathbf{R}_{\theta_y}\mathbf{R}_{\theta_x}$, and $\mathbf{t_p}$ is the 3D translation vector.

The precision of the initial pose estimation is limited by the discrete set of out-of-plane rotations included in the database. Below, I present a continuous optimization method to refine the pose estimate. The proposed method is a combination of the iterative closest point (ICP) [134] and Gauss-Newton [17] optimization algorithms. It can work with any number (one or more) of views with known camera

poses.

**Refinement Algorithm:** Let $\mathbf{M}^{(j)} \in \mathrm{SE}(3)$ be the 3D rigid transformation matrix representing the pose of the camera corresponding to the $j$th view in the world coordinate system, and let $\mathbf{P} = (\mathbf{K} \;\; \mathbf{0})$ be the $3 \times 4$ projection matrix. As explained in Section 5.4.1, the projection matrix is known through camera calibration, and the camera poses are known through hand-eye calibration and the motion of the robot. The edge points detected in the $j$th view are given by the set $V^{(j)} = \{\mathbf{v}_i^{(j)}\}$.

First, I establish a set of correspondences between the 3D CAD model points $\tilde{\mathbf{u}}_i^{(j)}$ and the 2D detected edge points $\mathbf{v}_i^{(j)}$. I find these 3D-to-2D point correspondences via closest-point assignment on the image plane. To do so, I simulate the multi-camera setup and render the 3D CAD model with respect to the current pose estimate $\mathbf{p}$. Let $U^{(j)} = \{\mathbf{u}_i^{(j)}\}$ be the sets of detected edge points in the $j$th synthetic view and $\tilde{U}^{(j)} = \{\tilde{\mathbf{u}}_i^{(j)}\}$ be the corresponding 3D CAD model points in the $j$th camera coordinate system. For each point $\mathbf{u}_i^{(j)} \in U^{(j)}$, I search for the nearest point in $V^{(j)}$ with respect to the directional chamfer matching cost as

$$\arg \min_{\mathbf{v}_k^{(j)} \in V^{(j)}} \left\| \mathbf{u}_i^{(j)} - \mathbf{v}_k^{(j)} \right\| + \lambda \left\| \phi\big(\mathbf{u}_i^{(j)}\big) - \phi\big(\mathbf{v}_k^{(j)}\big) \right\|_{\pi} \tag{5.21}$$

and establish 3D-to-2D point correspondences $\big(\tilde{\mathbf{u}}_i^{(j)}, \mathbf{v}_i^{(j)}\big)$.

Using the found correspondences, my optimization algorithm minimizes the sum of squared projection errors simultaneously in all the views:

$$\varepsilon(\mathbf{p}) = \sum_j \sum_{\tilde{\mathbf{u}}_i^{(j)}} \left\| \mathbf{P}\mathbf{M}^{(j)}\mathbf{M}_{\mathbf{p}}{\mathbf{M}^{(j)}}^{-1}\tilde{\mathbf{u}}_i^{(j)} - \mathbf{v}_i^{(j)} \right\|^2. \tag{5.22}$$

Both the 3D points $\tilde{\mathbf{u}}_i^{(j)}$ and their projections are expressed in homogeneous coordinates, while the corresponding edge points are expressed in Cartesian image

coordinates. With a slight abuse of notation, in this formulation, I assume that the projections of the 3D points have been converted to 2D image coordinates before measuring the distances.

The nonlinear least squares error function given in (5.22) is minimized using the Gauss-Newton algorithm. Starting with the initial pose estimate $\mathbf{p}^0$, I improve the estimation via the iterations

$$\mathbf{p}^{t+1} = \mathbf{p}^t + \Delta\mathbf{p}. \tag{5.23}$$

The update vector $\Delta\mathbf{p}$ is given by the solution of the normal equations

$$(\mathbf{J}_{\boldsymbol{\varepsilon}}^T \mathbf{J}_{\boldsymbol{\varepsilon}})\Delta\mathbf{p} = -\mathbf{J}_{\boldsymbol{\varepsilon}}^T \boldsymbol{\varepsilon}, \tag{5.24}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^N$ is the residual vector comprising the $N$ summed error terms in (5.22), and $\mathbf{J}_{\boldsymbol{\varepsilon}}$ is the $N \times 6$ Jacobian matrix of $\boldsymbol{\varepsilon}$ with respect to $\mathbf{p}$, evaluated at $\mathbf{p}^t$. Similar to the ICP algorithm, the correspondence and minimization problems are solved iteratively until convergence.

**Implementation for Bin Picking:** The pose refinement method could be used with a single view to refine the coarse pose estimate. However, I found that the estimation accuracy obtained using a single view is not sufficient for accurate grasping. To increase the accuracy, I use a two-view approach. I move the robot arm to a second location and capture the scene again with the MFC. The second location is determined depending on the coarse pose estimate of a detected object, such that in the second view the object is captured at the center of the image and from a different out-of-plane rotation angle.

Since I perform the coarse pose estimation on the first view, typically the projection errors in the second view are larger than those in the first view. This holds particularly true when the distance between the camera and the objectdiffers from the hypothesized distance $t_z$ that generated the database (Section 5.4.3). To improve convergence, I first perform the refinement using the first view only, for several iterations, and then jointly using both views. In general, I found that 20 iterations suffice for convergence.

### 5.4.6 Error Detection and Pose Correction in the Gripper

The estimated 3D pose of the object is used to grasp the object with the gripper and lift it from the part container. The grasping will fail if the estimated pose is inaccurate. Even if the estimated pose is correct, the grasping process may introduce errors because of slippage and interference from the other objects. These can result in a grasping failure (object is not retrieved) or in the object having the wrong pose in the gripper, which would make it impossible to perform subsequent assembly tasks. Therefore, after grasping, I use the gripper camera to detect these errors and correct the object pose before the next stage in the assembly.

The goal of this error detection and pose correction process aims to determine whether the object is grasped with the correct pose. I use a standard camera as the gripper camera and mount it above the wrist joint of the robot arm, as shown in Figure 5.5. The gripper camera is focused on the gripper's tip and captures an image of the object after it is lifted out the part container. I use the Canny edge detector

to extract edges from the image acquired by the gripper camera. The extracted edges include both texture and depth edges, which are not as robust as the depth edges extracted using the MFC. However, since the object is already isolated in the gripper, these edges work well for error detection.

Since I know the ideal pose of the object in the gripper (the pose that would occur if there were no error in the initial pose estimation and gripping process), I use this ideal pose as the initial guess and apply the pose refinement algorithm described in Section 5.4.5. If the refined pose is very different from the ideal pose or the matching cost becomes larger than a threshold, then I detect it as an error and drop the object back into the part container. Otherwise, I use the refined pose for the subsequent assembly task. Since the gripper camera is located above the robot's wrist joint, I can obtain a second view of the object in the gripper with a different pose by rotating the wrist and capturing another image using the gripper camera. In the experiments (see Section 5.5.1.4), I show that single-view pose estimation is sufficient for detecting errors, but for pose correction, two-view pose estimation is preferable due to the higher accuracy required.

**Foreground Extraction:** I exploit robot motion to make the pose estimation in the gripper more accurate and efficient. The idea is to move the robot arm during the exposure time of the camera, while keeping the relative pose between the camera and the gripper fixed. This can be achieved by fixing the joints of the robot that are between the gripper and the arm segment to which the camera is attached (not moving the wrist joint) and moving the other joints. This robot motion introduces blur only in the background while keeping the foreground object sharp. As shown

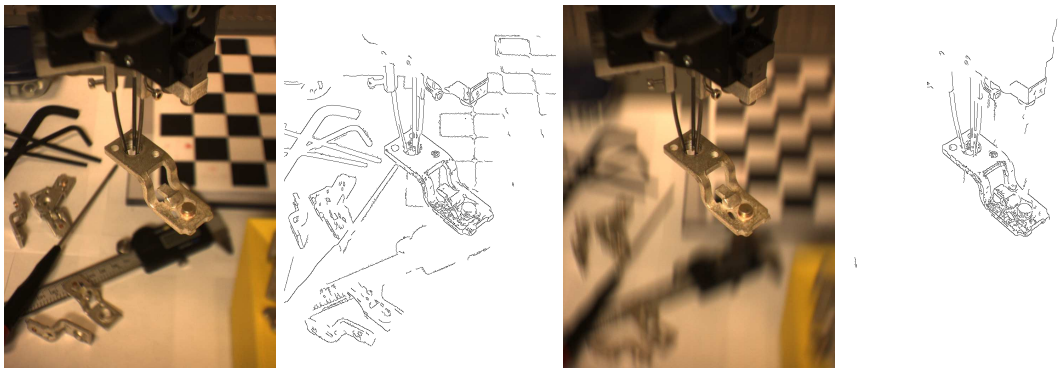



Figure 5.11: Foreground extraction. Images captured by the gripper camera (a) while the robot arm is fixed and (c) during a robot arm motion in which the relative pose between the object and the gripper camera is fixed. Note that in (c), the background is blurred due to the motion, while the foreground object remains sharp. Corresponding Canny edge detection results using the same threshold are shown in (b) and (d).

in Figure 5.11, images captured during such a robot motion produce sharp edges only on the foreground object (which is stationary relative to the camera), leading to accurate and efficient pose estimation. I call this *foreground extraction*, because it is essentially the reverse of standard background subtraction.

## 5.5 Experiments

I conducted extensive evaluations of the proposed algorithm for several applications using challenging real and synthetic datasets. In this section, I first demonstrate results for the robotic bin-picking system described in Section 5.4 and then present results of the proposed shape matching algorithm (described in Section 5.3)

for other applications: deformable object detection using a hand-drawn shape, and human pose estimation.

Note that in all of my experiments, I emphasize my FDCM algorithm's improvement in accuracy and speed compared to CM and OCM. For deformable object detection and human pose estimation, the performance of FDCM is roughly comparable to state-of-the-art methods, and if desired the FDCM estimates could be further refined by using them as initial hypotheses for more computationally expensive point registration algorithms.

In all my experiments, I used $q = 60$ orientation channels. I set the weighting factor $\lambda = \frac{180}{6 \cdot \pi}$, which means that a 6° error in line orientation carries the same penalty as a 1-pixel distance in image plane.

### 5.5.1 Pose Estimation for Robotic Bin Picking

#### 5.5.1.1 Synthetic Examples

I quantitatively evaluated the accuracy of the proposed matching algorithm to detect and localize objects in highly cluttered scenes on an extensive synthetic dataset. The synthetic dataset was generated using 3D models of 6 objects, with 3D shapes of varying complexity, which were placed randomly one over the other to generate several cluttered scenes. I computed depth-edge images by simulating the MFC and its cast shadows in software using OpenGL. The average occlusion of each part in the dataset was 15%, while the maximum occlusion was 25%. Moreover, in order to simulate missing depth edges and other imperfections in MFC imaging,

a small fraction (about 10–15%) of the depth edges were removed. Furthermore, the depth-edge images were corrupted with significant noise by adding uniformly sampled line segments. There were a total of 606 such synthetic images rendered under this setup, six of which are shown in Figure 5.12.

For each object in this experiment, I generated a database containing $K = 300$ shape templates, one for each of the uniformly sampled out-of-plane rotations (see Section 5.4.3). For each query image, I retrieved the best template pose using a brute force search scheme (over all in-plane rotation and translation parameters). Full 3D pose of the objects were then recovered for a known depth using the estimated in-plane transformation parameters together with the out-of-plane rotation parameters that generated the template poses. An estimation was labeled as correct if the position was within 5 mm, and the three estimated angles were each within 10°, of the ground truth pose.

**Detection and Localization:** I compared the performance of my proposed FDCM to CM (described in Section 5.3.1) and OCM [106]. The detection failure rates and processing times are shown in Table 5.1. Whereas CM had an average detection failure rate of 0.24, the proposed FDCM algorithm had a failure rate of only 0.05. It also improved upon the error rate of the competing state-of-the-art matching formulation (OCM) by a factor of 2. Notice that objects with discriminative shapes, such as the diamond part and the circuit breaker part, are easier to detect and localize. In contrast, the T-nut object, which has a simple shape, is relatively difficult to detect, since false edges from clutter and other objects frequently mislead the optimization algorithm. The FDCM algorithm is 40× faster than CM, and 90×

Table 5.1: Detection failure rates and processing time in highly cluttered scene with multiple objects.

| Algorithm | Circuit Breaker | Diamond Part | Ellipse Part | T-Nut | Knob | Wheel | Avg. | Time (sec) |
|---|---|---|---|---|---|---|---|---|
| FDCM (ours) | **0.03** | **0.01** | **0.05** | **0.11** | **0.04** | **0.08** | **0.05** | **0.71** |
| OCM [106] | 0.05 | 0.05 | 0.14 | 0.17 | **0.04** | 0.17 | 0.10 | 65.3 |
| CM | 0.11 | 0.22 | 0.26 | 0.34 | 0.26 | 0.22 | 0.24 | 29.1 |

faster than OCM: The average detection time of FDCM was 0.71 seconds, compared to 29.1 seconds for CM and 65.3 seconds for OCM. Several examples of successful detections for various objects in challenging scenarios are shown in Figure 5.12.

**Robustness to Occlusion:** I further quantitatively evaluated the robustness of the FDCM algorithm to varying degrees of occlusion, from no occlusion to an average occlusion of 30%. The results are presented in Figure 5.13. I achieved greater than 99% detection rate up to 5% occlusion, and about 85% detection rate when one-fourth (25%) of the object is occluded.

**Two-View Pose Refinement:** In this experiment, I evaluate the accuracy of the pose refinement algorithm described in Section 5.4.5. Using the same set of 6 objects, I render one object at a time in random poses. After a coarse pose estimate was computed, both the refinement schemes using one view and that using two views were applied independently to further refine the estimates. The final pose estimates were compared to the ground truth pose. The results in Table 5.2, averaged over 6 objects and 100 trials each, demonstrate that the two-view approach outperformed

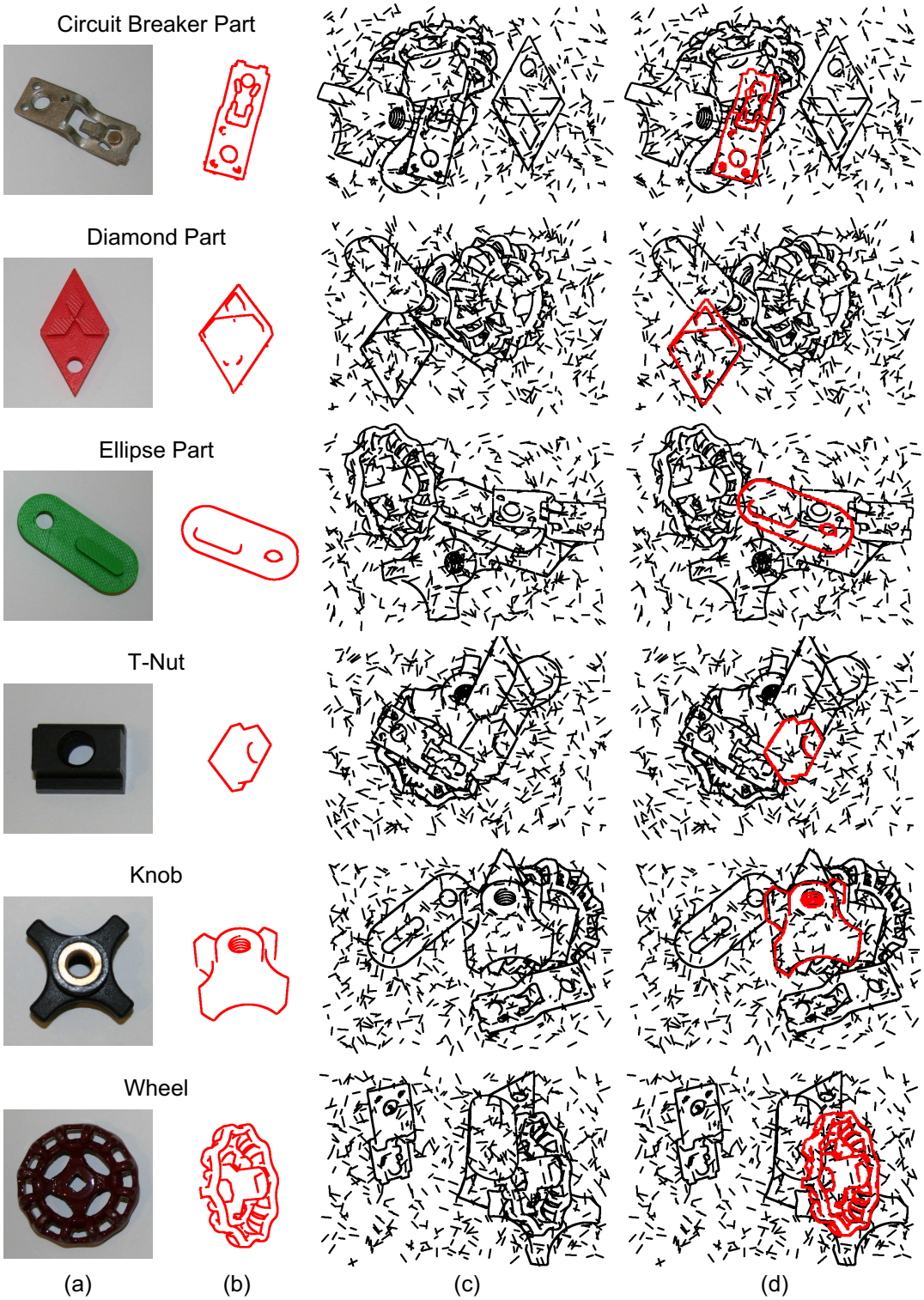



Figure 5.12: Examples of successful pose estimation on the synthetic dataset. (a) Photo of each part. (b) Sample depth-edge template. (c) Rendered query image. (d) Pose estimation result.

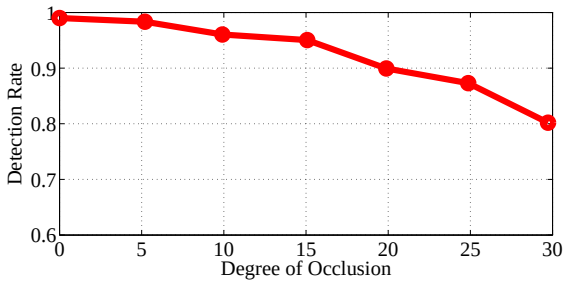



Figure 5.13: Detection rate versus percentage of occlusion.

Table 5.2: Comparison of the average absolute pose estimation error between the one-view and two-view approaches.

| Average absolute error | $t_X$ mm | $t_Y$ mm | $t_Z$ mm | $\theta_X$ degree | $\theta_Y$ degree | $\theta_Z$ degree |
|---|---|---|---|---|---|---|
| 1 View | 0.127 | 0.165 | 1.156 | 0.674 | 0.999 | 0.349 |
| 2 View | 0.094 | 0.096 | 0.400 | 0.601 | 0.529 | 0.238 |

the one-view approach. In the rendered images, 1 mm corresponded to about 6.56 pixels on the image plane, indicating that the two-view estimate achieves sub-pixel accuracy.

#### 5.5.1.2 Real Examples

**Object Detection and Pose Estimation in Cluttered Scenes:** To quantitatively evaluate performance, I performed several real experiments. Six different types of objects were laid one on top of another in a cluttered manner as shown in Figure 5.14. I then extracted depth edges using the MFC and performed object detection and pose estimation on the resulting depth-edge images. In each trial of

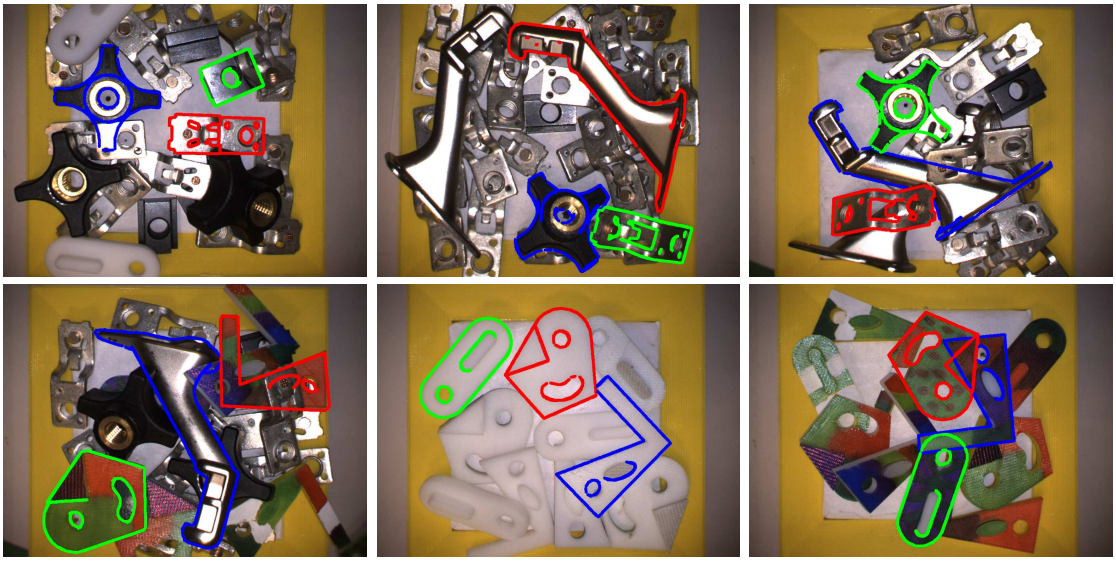


Figure 5.14: Results using real examples. The system detected and accurately estimated the pose for specular (shiny metal) objects, textureless objects (such as the ones in the bottom center image), and objects that have potentially misleading texture painted on them (such as the ones in the bottom right image). Overlaid on each image is the top detector output for each of three different object types.

this experiment, I used the system to detect a single instance of an object type. Over several hundred trials, the average detection rate was 95%. Shown in Figure 5.14 are some typical example trials of this experiment. On each image, I overlay the silhouettes of the detector outputs for three different object types. Notice that some of the parts have no texture, while others are quite specular. In such challenging scenarios, methods based on traditional image edges (e.g., Canny edges) usually fail, but the MFC enables us to robustly extract depth edges. Also notice that since the depth edge features are not affected by texture, my method works robustly even for parts that have artificial texture painted on them. This indicates that the method

can work in the presence of oil, grime, or dirt (which are all common in industrial environments), all of which add artificial texture to the surface of objects.

**Statistical Evaluation:** In order to statistically evaluate the accuracy of the proposed system, I need a method of independently obtaining the 3D ground truth pose of the object. Since there was no simple way of obtaining this (especially when objects were stacked on top of each other or piled in a bin), I instead devised a method to evaluate the consistency of pose estimate across multiple viewpoints of the camera. I placed an object in the scene and commanded the robot arm to move to several rotations and translations, so that data are collected when the camera is pointing at the object using many different camera poses. The camera poses were maintained such that the distance along the $z$-axis between the camera and the object is $\pm 10$ mm from the hypothesized distance $t_z$ that was used to generate the database (Section 5.4.3). From each camera pose, MFC images were captured, and my algorithm was used to estimate the pose of the object in the camera coordinate system. Since the object is static, the estimated pose of the object in the world coordinate system should be identical irrespective of the viewpoint of the MFC. For each view, the estimated pose of the object was transformed to the world coordinate system using the known position and orientation of the robot arm. I repeated this experiment for 7 different objects, with 25 trials for each object (the object was placed in a different pose for each trial). During each of these independent trials, the robot arm was moved to 40 different viewpoints in order to evaluate the consistency of the pose estimates. The histogram of the deviations from the median pose estimate is shown in Figure 5.15. The results demonstrate that the algorithm

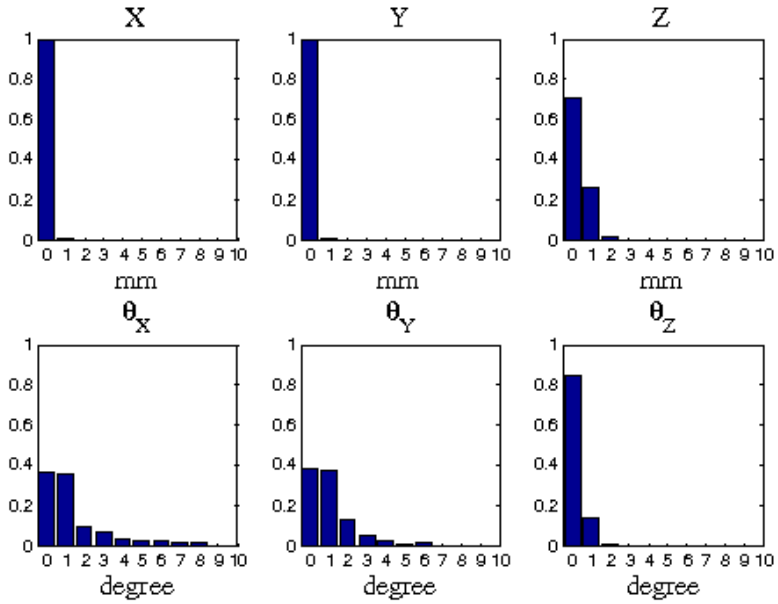



Figure 5.15: Results from real examples. Histograms of deviations from the median pose estimate, in mm (top) and degrees (bottom), across multiple trials of pose estimation.

computes consistent estimates, with a standard deviation of less than 0.5 mm in the in-plane directions $(x, y)$ and about 2 degrees in each of the three orientation angles. The standard deviation of the estimate in the $z$ direction (along the optical axis of camera) is slightly larger (approximately 1.2 mm).

**Effect of Depth Variation:** In my experiments, the system was optimized for a part container with a depth variation of 40 mm and a distance along the $z$-axis of 275 mm from the camera to the top of the part container. As explained in Section 5.4.3, the pose estimation algorithm requires a rough value of the distance, $t_z$, from the camera to the objects along the $z$-axis. In this experiment, I analyze how deviations of the true object distance from the hypothesized distance $t_z$ affect pose estimation accuracy.

I placed a single object in the scene and performed pose estimation at several different camera poses with offsets along the $z$-axis from the hypothesized distance of 275 mm. At each $z$ offset (height), I repeated the pose estimation for 100 trials by randomly changing the camera pose in the $(x, y)$ directions. As in the previous experiment, I used the median pose estimate as the ground truth pose. An estimate is labeled as correct if the translation error, computed as the Euclidean distance between the $(x, y, z)$ translation vectors, is less than 3 mm and the rotation error, computed as the geodesic distance between two 3D rotations, is less than 8°. The accuracy of the system is shown in Figure 5.16. The pose estimation algorithm is quite robust to depth variations between $[-20, +50]$ mm, which is significantly larger than my target capture range. Outside of this range, my two-view pose refinement algorithm failed to converge to the true solution for several trials, due to the incorrect distance assumption causing large projection errors. This experiment suggests that for part containers with larger depth variations, coarse pose estimation should be performed at multiple scales, targeting different depths, to get a better initial depth estimation. Alternatively, I could move the robot arm and change the height of the capture position based on previous object pose estimates in order to maintain a roughly constant distance between the camera and the objects.

#### 5.5.1.3 Bin-Picking System Performance

I evaluated the performance of bin picking using the robotic system shown in Figure 5.5. Figure 5.5 shows a part container (bin) containing a large number of

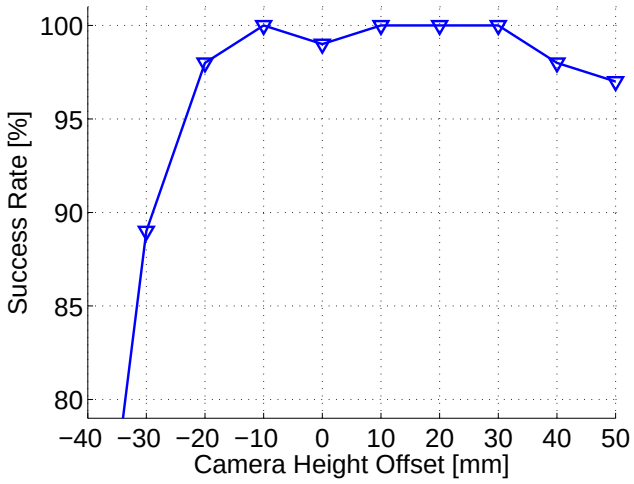



Figure 5.16: Effect of depth variation on pose estimation.

circuit breaker parts. The gripper (end effector) of the robot arm is designed to grasp each of the objects by first inserting its three metal pins in the closed state through a hole in the object. The gripper then opens by moving the three pins radially outward, thereby exerting outward horizontal forces on the inside edges of the hole. The gripper has a diameter of 3 mm in its closed state, while the hole in the object has a diameter of about 6 mm. Therefore, in order to successfully insert the gripper inside the hole (before lifting the object), the pose estimate error in the $(x, y)$ directions must be less than 1.5 mm. If the pose estimate error is greater, the pins will not be inserted into the hole, resulting in a failure to grasp the object.

My system was able to successfully guide the robot arm in the grasping task, achieving a grasping success rate of 94% over several hundred trials. There were two main causes for the grasping failures in 6% of the trials: (1) This particular target object has very similar depth edges when it is flipped upside-down, which occasionally led to inaccurate pose estimates; (2) Even when the pose estimation was

correct, the hole of the object was occasionally occluded by other objects, resulting in a grasping failure. It is important to note that all of these grasping failures were detected by the error-detection process using the gripper camera, so they did not affect the subsequent assembly task. Among the instances of successful grasping, a few trials resulted in the object being picked up by the gripper in an incorrect pose, due to interference from neighboring objects during the pickup process. These cases were also detected and were corrected automatically by my system using the gripper camera and my process for pose estimation and correction in the gripper (described in Section 5.4.6).

**Processing Time:** The entire pose estimation process requires less than 1 second for an object in an extremely cluttered environment (on an Intel quad-core 3.4 Ghz CPU with 3 GB memory). The decomposition of processing time is 0.6 seconds for FDCM and 0.3 seconds for the multi-view pose refinement algorithm. Almost all of the computation occurs during robot motion, so the computation time has almost no effect on the system operation speed. In environments with minimal clutter, the algorithm runs about twice as fast, since there are significantly fewer edges in the captured images.

#### 5.5.1.4 Pose Estimation in the Gripper

To evaluate the system's potential for error correction in the gripper, I measure the accuracy of pose estimation in the gripper using different numbers of views. In this experiment, I picked up circuit breaker parts from the part container as

described in Section 5.5.1.3. After each pickup, I captured 8 images at different wrist rotation angles, as shown in Figure 5.17, using the gripper camera and foreground extraction during robot motion (described in Section 5.4.6). I performed the pose refinement algorithm (Section 5.4.5) using from 1–8 views (for these experiments, I used the ideal pose of the object as the initial guess).

Figure 5.18 shows pose estimation errors using different numbers of views. Since the ground truth of the object pose in the gripper is not available, I used the pose that was estimated using all 8 views as the ground truth, and compared it with the poses estimated using different numbers (1–7) of views. The plots show the average estimation errors and standard deviations of these estimation errors (error bars) over 100 trials. Similar to my observations from synthetic data (shown in Table 5.2), the translation errors on real data are smaller for two-view estimation than for one-view estimation (see Figure 5.18, left). Two-view pose estimation is sufficient for accurate pose correction (error less than 0.5 mm), and using more than two views does not further improve the estimation. The rotation estimation errors are roughly the same ($< 2$ degrees) for each number of views. Figure 5.19 shows a typical example of two-view pose estimation. The estimated pose of the object matches the object's outline in both views quite closely, illustrating the high accuracy of the estimated pose. The difference between this estimated pose and the ideal pose provides an idea of the typical size of the initial pose error in the gripper. My system automatically estimates and corrects this error in the gripper.

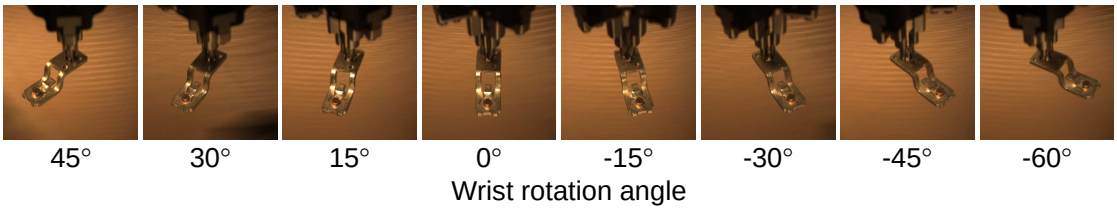



Figure 5.17: Example eight views captured with different wrist rotation angles.

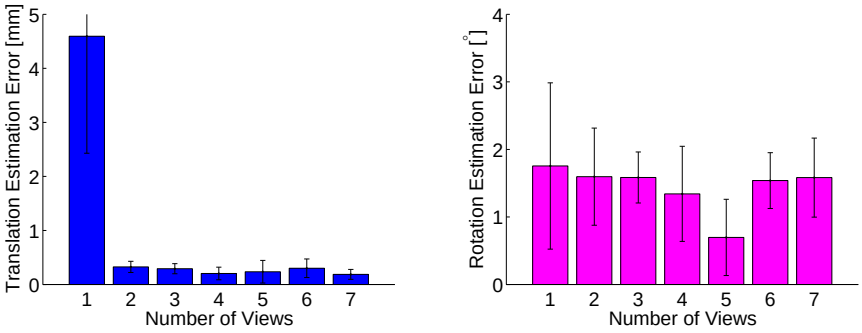



Figure 5.18: Pose estimation errors in the gripper using different numbers of views.

### 5.5.2 Deformable Object Detection

I applied the FDCM algorithm to object detection and localization on the ETHZ shape class dataset [39]. The dataset consists of 255 images, each of which contains one or more objects from five different object classes: apple logos, bottles, giraffes, mugs, and swans. The objects have large variations in appearance, viewpoint, size, and non-rigid deformation. I followed the experimental setup proposed in [39, 38], in which a single hand-drawn shape for each class is used to detect and localize its instances in the dataset.

My detection system is based on scanning using a sliding window. I retain all the hypotheses whose matching costs are less than the detection threshold. I densely sampled the hypothesis space and searched the images at 8 different scales

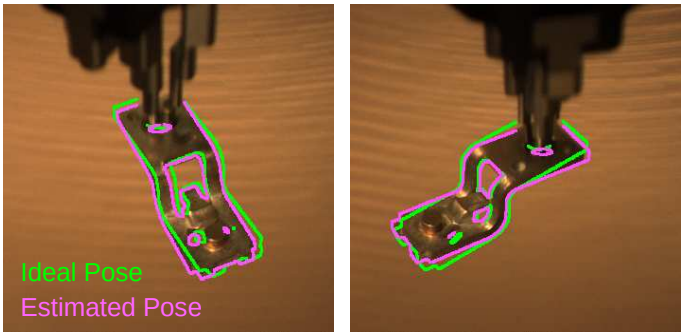



Figure 5.19: Two-view pose estimation in the gripper. The ideal pose is used as the initial guess and refined to give the estimated pose. (Both poses are superimposed on the input images of the two views).

and 3 different aspect ratios. The ratio between two consecutive scales is 1.2 and between consecutive aspect ratios is 1.1. I performed non-maximal suppression by retaining only the lowest-cost hypothesis among any group of detections that have significant spatial overlap.

In Figure 5.20, I plot detection rate vs. false positives per image. The curve is generated via altering the detection threshold for the matching cost. I compared my approach with OCM [106] and two recent studies by Ferrari et. al. [39, 38]. My approach outperforms OCM at all the false positive rates and is comparable to [38]. Compared to [38], my results are better for two classes (giraffes and bottles) and slightly worse for the swans class, while for two other classes (apple logos and mugs), the numbers are almost identical. As shown in the detection examples (Figure 5.21), object localization is highly accurate. Note that [135] and [100] report slightly better performance on this dataset, but I could not include their results in my graphs because their results were only reported in graphical format (as precision-recall

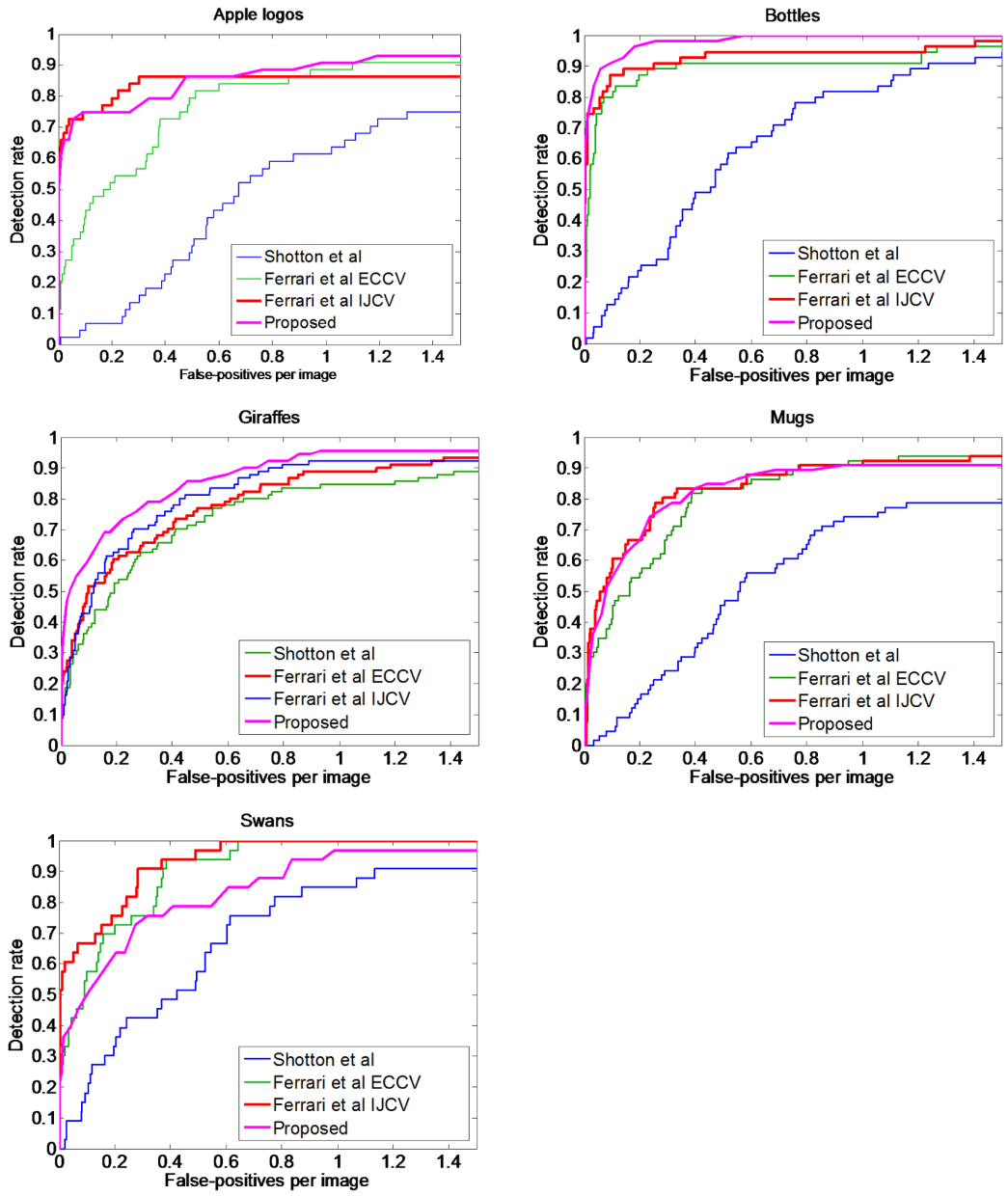



Figure 5.20: Receiver operating characteristic (ROC) curves on the ETHZ shape dataset comparing my proposed approach to OCM [106] and two recent studies by Ferrari et. al. [39, 38].

curves). Also note that these methods are orders of magnitude slower than FDCM.

**Complexity Comparison:** The average number of points in the shape templates were $1,610$, computed over five classes. My line-based representation used an average of 39 line segments per class. Note that the number of lines per class provides an upper bound on the number of computations required. Since the algorithm retrieves only the hypotheses having a smaller cost than the detection threshold, the summation was terminated for a hypothesis if the cost exceeded this value. By using this bound in the hypothesis domain (see Section 5.3.6.1 for more details), on average only 14 line segments were evaluated per hypothesis.

The average evaluation time for a single hypothesis was 0.40 $\mu s$ using FDCM, whereas this process took 51.50 $\mu s$ for OCM and 17.59 $\mu s$ for CM. The proposed method is 43$\times$ faster than chamfer matching and 127$\times$ faster than oriented chamfer matching. Note that the speed up is more significant for larger-sized templates, since my cost computation is insensitive to the template size, whereas the cost of standard chamfer matching increases linearly.

On average, I evaluated 1.05 million hypotheses per image, which took 0.42 seconds. Using the bound in the spatial domain presented in Section 5.3.6.2 enabled 91% of the hypotheses to be skipped, reducing the average evaluation time per image to 0.39 seconds. Note that the speedup is not proportional to the fraction of hypotheses skipped because in order to use the bound in the spatial domain, I could no longer use the bound in the hypothesis domain (Section 5.3.6.1).

Table 5.3: Pose estimation errors on three action sequences. Errors are measured as the mean absolute pixel distance from the ground truth marker locations.

| Algorithm | Walking | Jogging | Boxing | Average |
|---|---|---|---|---|
| FDCM (ours) | **7.3** | **12.5** | **9.7** | **9.8** |
| OCM [106] | 15.0 | 15.3 | 13.6 | 14.6 |
| CM | 9.3 | 13.6 | 10.6 | 11.2 |

### 5.5.3 Human Pose Estimation

I utilized my shape-matching framework for human pose estimation, which is a highly challenging task due to the large set of possible articulations of the human body. As proposed in [86], I matched a gallery of human shapes that have known poses to each test image. Due to articulation, the size of the pose gallery needed for accurate pose estimation is large. Hence, it becomes increasingly important to have an efficient matching algorithm that can cope with background clutter.

The experiments were performed on the HumanEva dataset [107], which contains video sequences of multiple human subjects performing various activities captured from different viewing directions. The ground truth locations of human joints at each image were extracted using attached markers. Shape gallery templates were acquired in two steps. First, I computed the human silhouettes via HumanEva background subtraction code. Then, using the Canny edges around the extracted silhouette outlines, I obtained the shape templates. I performed the experiment on video sequences from one subject of three actions: walking, jogging, and boxing. For

each action, I included all of the images from the subject's training sequence (about $1,000$–$2,000$ images) in the shape gallery. I used this to estimate the subject's pose in the corresponding validation sequence. As I extracted Canny edges directly from the validation images, they included us significant amount of background clutter. The best shape template, together with its scale and location, were then retrieved via the matching framework. I quantitatively evaluated the mean absolute error between the ground truth marker locations and the estimated pose on the image plane. The results, presented in Table 5.3, demonstrate significant improvements in accuracy compared to OCM and CM. My proposed approach can evaluate more than 1.1 million hypotheses per second, whereas CM and OCM can evaluate only $31,000$ and $14,000$ hypotheses per second, respectively. Examples of pose estimation are shown in Figure 5.22.

## 5.6 Conclusion

I presented a practical robotic system that uses novel computer vision hardware and algorithms for detection and 3D pose estimation of industrial parts in a cluttered bin. My implementation of the system on a robotic arm achieves accuracies on the order of 1 mm (reduced to less than 0.5 mm with automatic error correction) and 2°, with a total processing time of less than 1 second. Given a CAD model, a new object can be integrated into my system in less than 10 minutes. My goal with this line of research is to make substantial progress towards more versatile and easy-to-customize robotic bin-picking systems.

Figure 5.21: Several localization results on the ETHZ shape dataset. The images are searched using a single hand-drawn shape shown in the lower right of the images in the rightmost column.

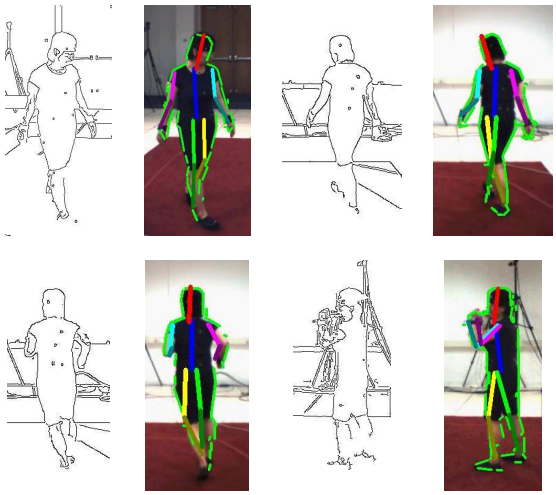


Figure 5.22: Human pose estimation results. *First row:* Walking sequence. *Second row:* Jogging and boxing sequences. Estimated poses and contours are overlayed on the images.

# Chapter 6

# Conclusion

In this dissertation, several discrete optimization problems that arise in superpixel segmentation, video segmentation, and object localization were studied. For superpixel segmentation, I proposed a clustering objective function which encourages the formation of compact, homogeneous, and balanced clusters. I proved that the objective function is monotonically increasing and submodular and showed that by enforcing a matroid constraint, one obtains additional efficiency and worst-case performance guarantee by using the greedy algorithm.

The dissertation proposed minimizing an energy function for video segmentation. The energy functions consists of the Potts pairwise terms and a higher-order histogram similarity prior term. I carefully studied the resulting optimization problem and showed that the minimization of the Pott energy, the higher-order histogram similarity prior, and their combination can all be achieved by maximizing an associated submodular function subject to a partition matroid constraint. This discovery revealed the application of greedy algorithms in solving the video segmentation problem. I further developed a branch and bound algorithm to further refine the segmentation result.

I proposed the fast directional chamfer matching algorithm for the object lo-

calization problem, which approximates shapes of query objects using line segments and evaluates the object evidence by accumulating the matching scores of the line segments. I further developed a directional integral image structure and showed that by evaluating the matching scores of longer line segments first one can significantly improve the computational efficiency.

# Bibliography


[1] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[2] Amit Agrawal, Yu Sun, John Barnwell, and Ramesh Raskar. Vision-guided robot system for picking objects by casting shadows. *International Journal of Robotics Research*, 29:155–173, February 2010.

[3] Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH)*, 2009.

[4] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[5] Prabin Bariya and Ko Nishino. Scale-hierarchical 3d object recognition in cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[6] Harry G Barrow, Jay Martin Tenenbaum, Robert Coy Bolles, and Helen C Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of the 5th international joint conference on Artificial intelligence*, pages 659–663, 1977.

[7] Ronen Basri and David W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc J. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[9] Csaba Beleznai and Horst Bischof. Fast human detection in crowded scenes by contour integration and local shape estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2246–2253, 2009.

[10] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *The Neural Information Processing Systems (NIPS) Foundation*, pages 585–591, 2001.

[11] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[12] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[13] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 26–33, 2005.

[14] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, 2002.

[15] Michael Bleyer, Carsten Rother, Pushmeet Kohli, Daniel Scharstein, and Sudipta Sinha. Object stereo — joint stereo matching and object segmentation. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[16] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Journal of Discrete Applied Mathematics*, 123, 2002.

[17] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[18] Yuri Boykov and Vladimir 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.

[19] Yuri Boykov, Olga Veksler, and Ramin Zabih. Efficient approximate energy minimization via graph cut. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 20(12):1222–1239, 2001.

[20] Yuri Boykov, Olga Veksler, and Ramin Zabih. Efficient approximate energy minimization via graph cut. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 20(12):1222–1239, 2001.

[21] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *Proceeding of European Conference on Computer Vision*, 2010.

[22] Ignas Budvytis, Vijay Badrinarayanan, and Roberto Cipolla. Semi-supervised video segmentation using tree structured graphical models. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[23] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679 –698, 1986.

[24] Prakash Chockalingam, Nalin Pradeep, and Stan Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *Proceeding of IEEE International Conference on Computer Vision*, 2009.

[25] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[26] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley, 2 edition, 1991.

[27] Yan Cui, Sebastian Schuon, Chan Derek, Sebastian Thrun, and Christian Theobalt. 3d shape scanning with a time-of-flight camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[28] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[29] Oscar Danielsson, Stefan Carlsson, and Josephine Sullivan. Automatic learning and extraction of multi-local features. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[30] Daniel DeMenthon and Larry Davis. Model-based object pose in 25 lines of code. In *Proceedings of the European Conference on Computer Vision*, pages 335–343, 1992.

[31] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 998–1005, 2010.

[32] Pinar Duygulu, Kobus Barnard, João F. G. de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the European Conference on Computer Vision*, pages 97–112, 2002.

[33] Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[34] Pedro Felzenszwalb and Daniel Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2004.

[35] Pedro Felzenszwalb and Joshua Schwartz. Hierarchical matching of deformable shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[36] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[37] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008.

[38] Vittorio Ferrari, Frederic Jurie, and Cordelia Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, 87:284–303, 2010.

[39] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *Proceedings of the European Conference on Computer Vision*, volume 3953, pages 14–28, 2006.

[40] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recogn.*, 41:176–190, January 2008.

[41] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6):381–395, 1981.

[42] Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. An analysis of the approximations for maximizing submodular set functions - ii. *Mathematical Programming*, pages 73–87, 1978.

[43] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[44] Andrew C. Gallagher, Dhruv Batra, and Devi Parikh. Inference for order reduction in markov random fields. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[45] Dariu M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 439–444, 1998.

[46] Leo Grady. Random walks for image segmentation. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[47] Eric Grimson and Tomás Lozano-Pérez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3:3–35, 1984.

[48] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[49] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, pages 235–284, 2008.

[50] David Harel and Yehuda Koren. On clustering using random walks. In *Foundations of Software Technology and Theoretical Computer Science*, volume 2245, pages 18–41. Springer-Verlag, 2001.

[51] Derek Hoiem, Andrew N. Stein, Alexei A. Efros, and Martial Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007.

[52] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 4(4):629–642, 1987.

[53] Hiroshi Ishikawa. Higher-order clique reduction in binary graph cut. 2009.

[54] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[55] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957.

[56] S. Jeannin and M. Bober. Description of core experiments for mpeg-7 motion/shape. *Technical Report ISO/IEC JTC 1/SC29/WG 11 MPEG99/N2690, MPEG-7*, 1999.

[57] Stefanie Jegelka and Jeff Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[58] Stefanie Jegelka and Jeff Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[59] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[60] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clustering: Good, bad and spectral. In *Proceedings. 41st Annual Symposium on Foundations of Computer Science*, 2000.

[61] Pushmeet Kohli, Lubor Ladicky, and Philip Torr. Robust higher order potentials for enforcing label consistency. *International Journal on Computer Vision*, 82:302–324, 2009.

[62] Pushmeet Kohli, Lubor Ladicky, and Philip Torr. Robust higher order potentials for enforcing label consistency. *International Journal on Computer Vision*, 82:302–324, 2009.

[63] Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. $\mathcal{P}^3$ & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 31:1645–1656, 2009.

[64] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[65] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[66] Robert Krauthgamer, Joseph Naor, and Roy Schwartz. Partitioning graphs in to balanced components. In *In Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[67] Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip H. S. Torr. Graph cut based inference with co-occurrence statistics. Proceeding of European Conference on Computer Vision, 2010.

[68] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.

[69] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *Proceeding of IEEE International Conference on Computer Vision*, 2011.

[70] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *SIGKDD*, pages 420–429, 2007.

[71] Alex Levinshtein, Adrian Stere, Kiriakos N. Kutulakos, David J. Fleet, and Sven J. Dickinson. Fast superpixels using geometric flows. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009.

[72] Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. *ACM Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH)*, 2005.

[73] Hui Lin and Jeff Bilmes. Word alignment via submodular maximization over matroids. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Short Papers*, pages 170–175, 2011.

[74] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 29(2):286–299, 2007.

[75] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[76] Ming-Yu Liu, Oncel Tuzel, Ashok Veeraraghavan, Rama Chellappa, Amit Agrawal, and Haruhisa Okuda. Pose estimation in heavy clutter using a multi-flash camera. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[77] Ming-Yu Liu, Oncel Tuzel, Ashok Veeraraghavan, Yuichi Taguchi, Tim K. Marks, and Rama Chellappa. Fast object localization and pose estimation in heavy clutter for robotic bin picking. 2012.

[78] L. Lovász. Submodular functions and convexity. *Mathematical Programming - State of the Art*, pages 235–257, 1983.

[79] David Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

[80] David Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991.

[81] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[82] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[83] Marina Meilă and Jianbo Shi. A random walks view of spectral segmentation. In *IEEE Interntaional Conference on Artificial Intelligence and Statistics*, 2001.

[84] Alastair P. Moore, Simon J.D. Prince, and Jonathan Warrell. "lattice cut" – construcitng superpixles using layer constraints. In *CVPR*, 2010.

[85] Alastair P. Moore, Simon J.D. Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In *CVPR*, 2008.

[86] Greg Mori and Jitendra Malik. Estimating human body configurations using shape context matching. In *Proceedings of the European Conference on Computer Vision*, volume 3, 2002.

[87] Greg Mori, Xiaofeng Ren, Alexei A. Efros, and Jitendra Malik. Recovering human body configurations: Combining segmentation and recognition. In *CVPR*, 2004.

[88] Miguel Carreira-Perpi nán and Richard S. Zemel. Proximity graphs for clustering and manifold learning. In *The Neural Information Processing Systems (NIPS) Foundation*, pages 225–232. MIT Press, 2004.

[89] Mukund Narasimhan, Nebojsa Jojic, and Jeff Bilmes. Q-clustering. In *The Neural Information Processing Systems (NIPS) Foundation*, pages 979–986. MIT Press, 2006.

[90] Bradley Nelson, N.P. Papanikolopoulos, and Pradeep Khosla. Robotic visual servoing and robotic assembly tasks. *IEEE Robotics and Automation Magazine*, 3(1):23–31, June 1996.

[91] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, pages 265–294, 1978.

[92] Andrew Ng, Micheal Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *The Neural Information Processing Systems (NIPS) Foundation*, pages 849–856. MIT Press, 2001.

[93] Juan Carlos Niebles and Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[94] Sebastian Nowozin, Peter Gehler, and Christoph Lampert. On parameter learning in crf-based approaches to object class image segmentation. In *ECCV*, 2010.

[95] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal on Computer Vision*, 42(3):145–175, 2001.

[96] Clark F. Olson and Daniel P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):103–113, 1997.

[97] James Oxley. *Matroid Theory*. Oxford University Press, 1992.

[98] Srikumar Ramalingam, Pushmeet Kohli, Karteek Alahari, and Philip H. S. Torr. Exact inference in multi-label crfs with higher order cliques. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[99] Ramesh Raskar, Kar-Han Tan, Rogerio Feris, Jingyi Yu, and Matthew Turk. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics*, 23(3):679–688, 2004.

[100] Saiprasad Ravishankar, Arpit Jain, and Anurag Mittal. Multi-stage contour based detection of deformable objects. In *Proceedings of the European Conference on Computer Vision*, pages 483–496, 2008.

[101] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *ICCV*, 2003.

[102] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH)*, 23(3):309–314, 2004.

[103] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

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

[105] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5(2):99 –108, 1973.

[106] Jamie Shotton, Andrew Blake, and Roberto Cipolla. Multi-scale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1270–1281, 2008.

[107] Leonid Sigal and Michael J. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical report, Brown University, 2006.

[108] Leonid Sigal and Michael J. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2048, 2006.

[109] Fridtjof Stein and Gérard Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125 –145, 1992.

[110] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1331–1338, 2005.

[111] Yuichi Taguchi, Bennett Wilburn, and C. Lawrence Zitnick. Stereo reconstruction with mixed pixels using adaptive over-segmentation. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[112] Arasanathan Thayananthan, Bjorn Stenger, Philip H. S. Torr, and Roberto Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 127–133, 2003.

[113] David Tsai, Mathew Flagg, and James M. Rehg. Motion coherent tracking with multi-label mrf optimization. In *British Machine Vision Conference*, 2010.

[114] Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1713–1727, 2008.

[115] Oncel Tuzel, Fatih Porikli, and Peter Meer. Kernel methods for weakly supervised mean shift clustering. In *Proceeding of IEEE International Conference on Computer Vision*, pages 48–55, 2009.

[116] Amelio Vazquez-Reina, Shai Avidan, Hanspeter Pfister, and Eric Miller. Multiple hypothesis video segmentation from superpixel flows. In *Proceeding of European Conference on Computer Vision*, 2010.

[117] Olga Veksler and Yuri Boykov. Superpixels and supervoxels in an energy optimization framework. In *ECCV*, 2010.

[118] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Joint optimization of segmentation and appearance models. Proceeding of IEEE International Conference on Computer Vision, 2009.

[119] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 13(6):583 –598, 1991.

[120] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[121] VisionPro. http://www.cognex.com/visionpro/. *Cognex*.

[122] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning*, pages 577–584, 2001.

[123] Fei Wang, Bin Zhao, and Changshui Zhang. Linear time maximum margin clustering. *IEEE Transactions on Neural Networks*, 21(2):319–332, 2010.

[124] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH)*, 2005.

[125] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. 2008.

[126] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[127] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. Distance metric learning with application to clustering with side-information. In *The Neural Information Processing Systems (NIPS) Foundation*, pages 505–512, 2002.

[128] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *The Neural Information Processing Systems (NIPS) Foundation*, 2004.

[129] Rui Xu and II Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645 –678, May 2005.

[130] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *The ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 907–916, 2009.

[131] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In *The Neural Information Processing Systems (NIPS) Foundation*, 2000.

[132] Luh Yen, Denis Vanvyve, Fabien Wouters, Francois Fouss, Michel Verleysen, and Marco Saerens. Clustering using a random-walk based distance measure. In *In Proceedings European Symposium on Artificial Neural Networks*, 2005.

[133] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971.

[134] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[135] Qihui Zhu, Liming Wang, Yang Wu, and Jianbo Shi. Contour context selection for object detection: A set-to-set contour matching approach. In *Proceedings of the European Conference on Computer Vision*, pages 774–787, 2008.

[136] Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.