An Improved Lawson Local-Optimization Procedure and Its Application

by

Yue Fang
B.Sc., Nanjing University of Posts and Telecommunications, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

An Improved Lawson Local-Optimization Procedure and Its Application

by

Yue Fang
B.Sc., Nanjing University of Posts and Telecommunications, 2012

Supervisory Committee

Dr. Michael D. Adams, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Alexandra Branzan Albu, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

The problem of selecting the connectivity of a triangulation in order to minimize a given cost function is studied. This problem is of great importance for applications, such as generating triangle mesh models of images and other bivariate functions.

In early work, a well-known method named the local optimization procedure (LOP) was proposed by Lawson for solving the triangulation optimization problem. More recently, Yu et al. proposed a variant of the LOP called the LOP with lookahead (LLOP), which has proven to be more effective than the LOP. Unfortunately, each of the LOP and LLOP can only guarantee to yield triangulations that satisfy a weak optimality condition for most cost functions. That is, the triangulation optimized by the LOP or LLOP is only guaranteed to be such that no single edge flip can reduce the triangulation cost. In this thesis, a new optimality criterion named n-flip optimality is proposed, which has proven to be a useful tool for analyzing the optimality property. We propose a more general framework called the modified LOP (MLOP), with several free parameters, that can be used to solve the triangulation-cost optimization problem. By carefully selecting the free parameters, two MLOP-based methods called the $\text{MLOP}_{\mathsf{B}}(L, M)$ and $\text{MLOP}_{\mathsf{C}}(L)$ are derived from this framework. According to the optimality property introduced in the thesis, we have proven our proposed methods can satisfy a stronger optimality condition than the LOP and LLOP. That is, the triangulation produced by our MLOP-based methods cannot have their cost reduced by any single edge flip or any two edge flips. Due to satisfying this stronger optimality condition, our proposed methods tend to yield triangulations of significantly lower cost than the LOP and LLOP methods.

In order to evaluate the performance of our MLOP-based methods, they are compared with two other competing approaches, namely the LOP and LLOP. Experimental results show that the MLOP_{B} and MLOP_{C} methods consistently yield triangulations of much lower cost than the LOP and LLOP. More specifically, our MLOP_{B} and MLOP_{C} methods yield triangulations with an overall median cost reduction of 16.36% and 16.62%, respectively, relative to the LOP, while the LLOP can only yield triangulations with an overall median cost reduction of 11.49% relative to the LOP. Moreover, our proposed methods $\text{MLOP}_{\mathsf{B}}(2, i)$ and $\text{MLOP}_{\mathsf{A}}(i)$ are shown to produce even better results if the parameter i is increased at the expense of increased computation time.

Contents

Supervisory Committee **ii**

Abstract **iii**

Contents **iv**

List of Tables **vi**

List of Figures **vii**

Acknowledgements **viii**

Dedication **ix**

1 Introduction **1**

1.1 Triangulation Connectivity Optimization 1

1.2 Historical Perspective 3

1.3 Overview and Contribution of the Thesis 4

2 Preliminaries **7**

2.1 Overview 7

2.2 Notation and Terminology 7

2.3 Concepts in Computational Geometry 8

2.4 Triangle Mesh Models 10

2.5 Halfedge Data Structure 11

2.6 Triangulation-Cost Optimization Problem 11

2.7 Local Optimization Procedure (LOP) and Look-ahead LOP (LLOP) . 15

2.8 Triangulation Cost Function 15

3 Proposed Methods and Their Evaluation **18**

3.1 Overview 18
3.2 n-Flip Optimality 18
3.3 Proposed LOP Variant: The Modified LOP (MLOP) 19
3.3.1 Additional Notation 23
3.3.2 Good Flip-Sequence Selection Policies 24
3.3.3 Permissible Flip-Sequence Policies 24
3.3.4 Determination of Suspect Edges 28
3.3.5 Proposed Methods for Using the MLOP 28
3.4 Optimality Properties of the MLOP, LOP and LLOP 30
3.5 Evaluation of Proposed LOP 32

4 Conclusions and Future Research **41**
4.1 Conclusions 41
4.2 Future Research 42

A Software User Manual **43**
A.1 Introduction 43
A.2 Installing the Software 43
A.3 Detailed Program Descriptions 44
A.3.1 The `optimize_mesh` Program 45
A.3.2 The `generate_mesh` Program 48
A.3.3 The `two_flip_optimal` Program 51
A.4 Examples of Software Usage 51

Bibliography **54**

List of Tables

Table 3.1 Parameter selections for defining various permissible flip-sequence policies using Algorithm 2. 27
Table 3.2 Triangulation costs obtained using the various optimization methods. (a) Results for a representative subset of the individual test cases. (b) Average rankings taken across all test cases. 34
Table 3.3 Comparison of reduction in triangulation cost obtained with various methods relative to the LOP. 35
Table 3.4 Triangulation costs obtained using $\mathrm{MLOP}_{\mathsf{B}}(2, M)$ for various choices of M 38
Table 3.5 Triangulation costs obtained using $\mathrm{MLOP}_{\mathsf{C}}(L)$ for various choices of L 39

Table A.2 Choices of triangulation-cost function 47
Table A.3 Choices of good flip-sequence selection policies 47
Table A.4 Choices of history level 47
Table A.5 Choices of permissible flip-Sequence policies 48
Table A.6 Choices of response mode when a potential cycles is detected . . 48

List of Figures

Figure 1.1 Examples of (a) triangle mesh model and (b) triangulation of image domain. 2

Figure 2.1 Examples of (a) convex and (b) non-convex sets 8
Figure 2.2 Convex hull example. (a) A set P of points. (b) The convex hull of P. 9
Figure 2.3 Triangulation example (a) a set P of points, (b) a triangulation of P, and (c) another triangulation of P 10
Figure 2.4 Mesh model of a bivariate function ϕ. (a) A bivariate function ϕ, (b) the triangulation in the image domain, and (c) the resulting triangle mesh model. 12
Figure 2.5 Pictorial view of the halfedge data structure. 13
Figure 2.6 An example of parts of a triangulation with flippable and unflippable edges. (a) Flippable edge e, (b) unflippable edge e, and (c) unflippable edge e. 14
Figure 2.7 An example of an edge flip in part of triangulation. (a) Before and (b) after an edge flip is applied to the edge e. 14
Figure 2.8 an quadrilateral $v_iv_jv_kv_l$ with edge $e = \overline{v_iv_j}$ as diagonal 16

Figure 3.1 Computation time versus mesh size for (a) $\mathrm{MLOP}_{\mathsf{B}}(2, 2)$ with SE, (b) $\mathrm{MLOP}_{\mathsf{B}}(2, 2)$ with ABN, (c) $\mathrm{MLOP}_{\mathsf{B}}(2, 2)$ with JND, (d) $\mathrm{MLOP}_{\mathsf{C}}(2)$ with SE, (e) $\mathrm{MLOP}_{\mathsf{C}}(2)$ with ABN, and (f) $\mathrm{MLOP}_{\mathsf{C}}(2)$ with JND. 40

ACKNOWLEDGEMENTS

This thesis would never been done without the help and support from numerous people. I would like to take this opportunity to express my thanks to certain individuals in particular:

To my supervisor Dr. Michael Adams. Thank you for your guidance, encouragement, and extreme patience to me. I would like to express my sincere gratitude and deep appreciate to you for all the things you have taught me. You spent a lot of time and effort on teaching me C++, which is a precious asset for my future career. Without your mentoring in my research project coding, result analysis, and academic writing, this thesis would not have been written. It has been such a pleasure working with you.

To my committe member Dr. Alexandra Branzan Albu. Thank you for being on my supervisory committee and spending time on reviewing my thesis.

To my course instructors. I would like to express my gratitudet to Dr. Alexandra Branzan Albu, Dr. Frank Ruskey, Dr. T. Aaron Gulliver, and Dr. Wu-Sheng Lu. Thank you for offering all the fantastic lectures.

To my friends. I wish to thank my friends, Dan Han, Yue Tang, Xiao Feng, and all other friends, who have helped me during my study. I must also acknowledge our research group members, Ali Mostafavian, Badr El Marzouki, Paul Guo, Jay Luo, Xiao Ma, Zhenmai Hu, and others. I have learnt a lot in our group meetings from your presentation. I am grateful for being in the same research group with you.

To my dearest wife and family . I would like to thank my wife Zhu Tian. Your support has been very important to me. I am so lucky to have you showing up in my life. My dearest parents, Jianglin Fang and Zuanru Yuan, thank you for your unconditional love and support. Your encouragement and support have given me the strength to overcome the difficulty.

DEDICATION

To my family.

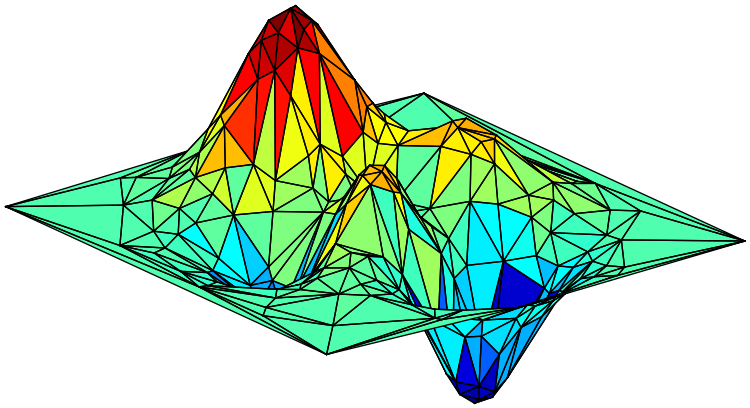
Chapter 1

Introduction

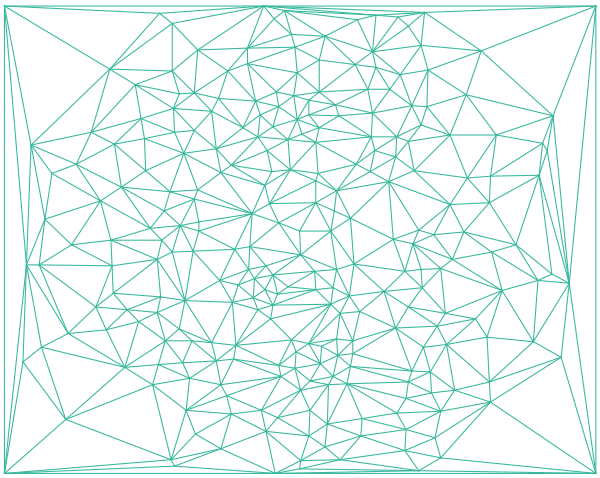
1.1 Triangulation Connectivity Optimization

Triangle meshes are used extensively to represent bivariate functions in a variety of scientific applications, such as, for representing images in signal processing, digital elevation maps in geographic information systems, and math functions in surface modelling. With a triangle mesh model of a bivariate function, the image domain is partitioned by a triangulation into a set of triangle faces and then over each face, an approximating function is constructed. To better illustrate the triangle mesh and triangulation, we provide an example in Figure 1.1. Figure 1.1(a) shows a triangle mesh model for representing a bivariate function. A triangulation is formed by partitioning the image domain of the bivariate function into a set of nonoverlapping triangles, as illustrated in Figure 1.1(b). By constructing an approximation function over each face of this triangulation, the model in Figure 1.1(a) is obtained.

In many geometric algorithms, it is necessary to compute a triangulation of a given set of points. One commonly used triangulation is the Delaunay triangulation [17]. The connectivity (i.e., how the points in the triangulation are connected by edges) chosen by Delaunay triangulation, however, is extremely limiting and almost never the very best choice in practice [34]. Furthermore, in some contexts, it may be necessary to choose the triangulation connectivity in such a way as to minimize, in some sense, a given cost function. For example, such an optimization is often needed in mesh-generation algorithms based on data-dependent triangulations [19, 31, 33, 18, 27, 28]. Although numerous approaches to solving the triangulation connectivity optimization problem are possible, a very popular class of solution techniques is the class based on



(a)



(b)

Figure 1.1: Examples of (a) triangle mesh model and (b) triangulation of image domain.

edge flips. The basic idea in these methods is to repeatedly apply simple transformations, called edge flips, to a triangulation in order to reduce the triangulation cost and ultimately obtain an optimal solution. Finding good computationally-efficient methods for choosing the triangulation connectivity based on the edge flips, however, is quite challenging. In this thesis, this particular connectivity-selection problem is the main interest and typically solved by formulating problem in terms of optimization.

1.2 Historical Perspective

As stated earlier, because of the extensive use of triangulations, many triangulation-connectivity optimization schemes have been developed over the years. One very commonly used triangulation connectivity is that based on the Delaunay triangulation [17]. The connectivity of a Delaunay triangulation is determined solely by the geometry of the points being triangulated. One important feature of the Delaunay triangulation is that a Delaunay triangulation is unique for a given set of sample points if no four points are co-circular. Even if the degeneracy occurs (i.e., four sample points are co-circular), a unique Delaunay triangulation can be obtained easily by applying an appropriate technique, such as the preferred directions scheme [13]. In addition to that, the Delaunay triangulation maximizes the minimum of all the interior angles of the triangles in a triangulation. This tends to avoid long and thin (i.e. sliver) triangles in the triangulation. Nevertheless, sliver triangles are not always a bad choice if they are chosen carefully [34]. For example, sliver triangles are very suitable for modelling functions that have regions with high second-order derivatives in one direction as compared to others. In images, this kind of region often corresponds to image edges. This finding led to the use of data-dependent triangulations. Unlike the Delaunay triangulation, the connectivity of data-dependent triangulations can be chosen arbitrarily based on the information associated with the sample points to be triangulated.

For a data-dependent triangulation, one very popular class of triangulation-connectivity optimization methods is the class based on edge flips. The best known optimization method in this class is the **local optimization procedure (LOP)** proposed by Lawson [25], which works by starting with a given triangulation and then transforming the triangulation by applying a single edge flip in each step. Relatively more recently, Yu, Morse, and Sederberg [44] proposed a variant of the LOP called the LOP with lookahead, which we refer to hereafter as the **lookahead LOP(LLOP)**. The LLOP is similar to the LOP in that the LLOP also applies edge flips to a triangulation until the triangulation is optimal. The LLOP, however, differs from the LOP in that, in addition to allowing a single edge flip in each step, the LLOP also allows a sequence of two edge flips, where the two edges share a common face. In [44], the LLOP was shown to yield triangulations of significantly lower cost than the LOP, but at the expense of increased computation time. Moreover, it was also noted [44, p. 66] that the LLOP is able to produce triangulations of lower cost than

those obtained using the more computationally expensive method in [36], which is based on simulated annealing.

Since the time of their proposal, the LOP and LLOP have been used in many applications, especially those for generating data-dependent triangulations, and have proven useful for constructing mesh models of images, elevation maps, and other bivariate functions [27, 28, 19, 31, 33, 18, 21]. Such mesh models, which are particularly useful for handling content-adaptive sampling, have proven useful in many applications, including: feature detection [14], pattern recognition [30], computer vision [35], image restoration [11], tomographic reconstruction [12], filtering [20], interpolation [37, 38], and image/video coding [7, 32, 26, 41, 15, 22, 6]. For example, two highly-effective mesh-generation frameworks for constructing mesh models of images were proposed in [27, 28] based on the LOP and LLOP. Through the use of the LOP and LLOP, triangulation connectivity can be chosen optimally so as to minimize approximation error. Since the LOP and LLOP are useful in many contexts such as those above, many applications could benefit from any further improvements to the LOP and LLOP, leading to our interest in these methods herein.

1.3 Overview and Contribution of the Thesis

This thesis is concerned with addressing the problem of selecting the connectivity of a triangulation in order to minimize, in some sense, a given cost function. In this thesis, a new notion of optimality, known as n-flip optimality, is proposed, which has proven to be a useful tool for analyzing methods that solve this particular type of optimization problem. We propose a computational framework, called the modified LOP (MLOP) that offers a number of key improvements over the LOP. The MLOP framework has a number of degrees of freedom, which offer great flexibility and the ability to perform much better than the LOP (or LLOP). By carefully selecting the free parameters in our computational framework, we obtain two highly-effective MLOP-based methods called the MLOP_{B} and MLOP_{C} methods, which make different trade offs between triangulation costs and computation time. After that, we prove the conditions needed for a strong optimality property, namely 2-flip optimality, and discuss when the LOP, LLOP and the MLOP-based methods yield triangulations that are 2-flip optimal. Our MLOP-based methods are shown to yield triangulations that, in most cases of practical interest, satisfy a stronger optimality condition (in the sense of n-flip optimality) than the LOP and LLOP methods. The performance of our methods are

also assessed by benchmarking them against the LOP and LLOP for triangulation connectivity optimization. Through experimental results, our MLOP_B and MLOP_C methods are shown to consistently yield triangulations of significantly lower cost than the LOP and LLOP, while still maintaining a reasonable computational cost.

Structurally, the remainder of this thesis is organized into three chapters and one appendix. An overview of each of the remaining chapters/appendix is provided in what follows.

Chapter 2 provides some necessary background information for facilitating the understanding of the work in this thesis. First, we introduce some basic notation and terminology. Then, some fundamentals in computational geometry are presented including convex hulls, triangulations, and triangle mesh models. Next, we introduce the halfedge data structure used for describing our proposed algorithm later. After that, we formally introduce the triangulation connectivity selection problem addressed in this thesis, followed by the description of the LOP and LLOP. Lastly, we introduce several well-known triangulation cost functions considered in our work.

Chapter 3 begins with the definition of the n-flip optimality, which can show the advantages of our proposed method later. Then, we introduce the definition of permissible flip sequence as the key free parameter in our algorithm. Next, a new triangulation optimization framework with a number of free parameters is proposed. For each of the free parameters, we provide a detailed description. By carefully selecting these free parameters, three optimization methods, namely $\text{MLOP}_\mathsf{A}(L)$, $\text{MLOP}_\mathsf{B}(L, M)$, and $\text{MLOP}_\mathsf{C}(L)$, are derived from our framework, where L and M are used to control the parameter permissible flip-sequence policy in the framework. After that, we analysis the optimality property of the MLOP-based methods and compare them with the LOP and LLOP in term of n-flip optimality. In this discussion, we first provide the conditions needed for 2-flip optimality and show when the LOP, LLOP and MLOP-based yield 2-flip optimal triangulations. After that, we prove our MLOP-based methods are able to satisfy 2-flip optimality, which is a stronger optimality condition than the LOP and LLOP for most triangulation cost function. In later parts of this chapter, we evaluate our proposed MLOP_B and MLOP_C methods by comparing them to the LOP, LLOP and MLOP_A methods in terms of triangulation cost computed by various cost functions. It is shown that our proposed MLOP_B and MLOP_C methods both yield triangulations of much lower cost than the other competing methods, at a relatively modest computational cost. For example, MLOP_B and MLOP_C methods yield a median cost reduction of 5.85% to 28.34%, and 5.96% to 30.50%, respectively,

relative to the LOP. These median cost reduction of MLOP_{B} and MLOP_{C} methods are also shown to be greater than that of the LLOP by 1.28% to 7.64%, and 1.44% to 8.42% respectively. Moreover, our experiments show that the proposed MLOP_{B} and MLOP_{C} methods can obtain even better results by increasing i in the $\text{MLOP}_{\mathsf{B}}(2, i)$ or in the $\text{MLOP}_{\mathsf{C}}(i)$ at expense of increased computational cost.

Chapter 4 concludes the thesis with a brief summary of the work and results presented herein. Finally, some recommendations for future research are made.

Appendix A provides a brief description of the software that is used to implement the computational framework proposed in the thesis and to collect experimental results. Moreover, our software provides the ability to test a triangulation for the 2-flip optimality condition. The software also implements a mesh-generation framework based on [27] and the proposed MLOP. Some useful instructions and examples of how to use this software are discussed in this appendix.

Chapter 2

Preliminaries

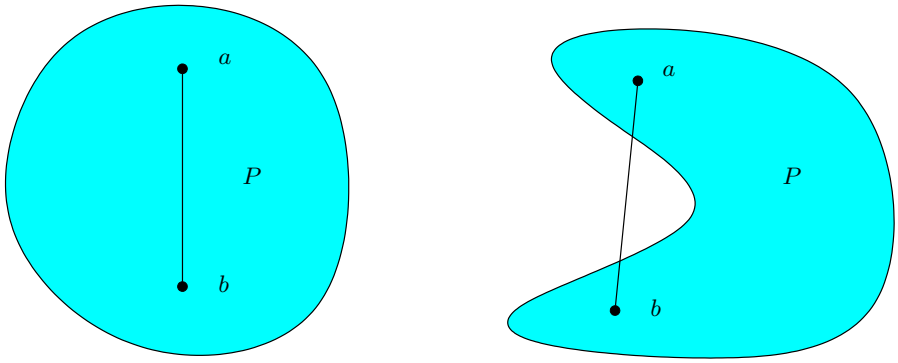
2.1 Overview

In this chapter, we present some essential background information in order to allow the reader to better understand the work presented in this thesis. First, we introduce the notation and terminology used herein. Then, some basic concepts in computational geometry are introduced, including convex hulls and triangulations. After that, triangle meshes and halfedge data structures are discussed. Next, we formally introduce the triangulation-cost optimization problem addressed in this thesis, followed by the description of the LOP and LLOP. Finally, we discuss triangulation cost functions and present the cost functions considered herein.

2.2 Notation and Terminology

Before processing further, it is necessary to introduce some additional basic notation and terminology. The sets of real number and integers are denoted as $\mathbb{R}$ and $\mathbb{Z}$, respectively. For $a, b \in \mathbb{R}$, we use following notation to denote various subsets of $\mathbb{R}$:

$$
\begin{aligned}
(a, b) &= \{x \in \mathbb{R} : a < x < b\}, \\
[a, b) &= \{x \in \mathbb{R} : a \leq x < b\}, \\
(a, b] &= \{x \in \mathbb{R} : a < x \leq b\}, \text{ and} \\
[a, b] &= \{x \in \mathbb{R} : a \leq x \leq b\}.
\end{aligned}
$$



(a) (b)

Figure 2.1: Examples of (a) convex and (b) non-convex sets

For a finite set S, the cardinality of S is denoted as $|S|$. For a vector $v = (v_1, v_2, v_3, ..., v_n)$ in $\mathbb{R}^n$, the 2-norm of v is denoted $||v||$ and defined as

$$||v|| = \sqrt{{v_1}^2 + {v_2}^2 + {v_3}^2 + ... + {v_{n-1}}^2 + {v_n}^2}.$$

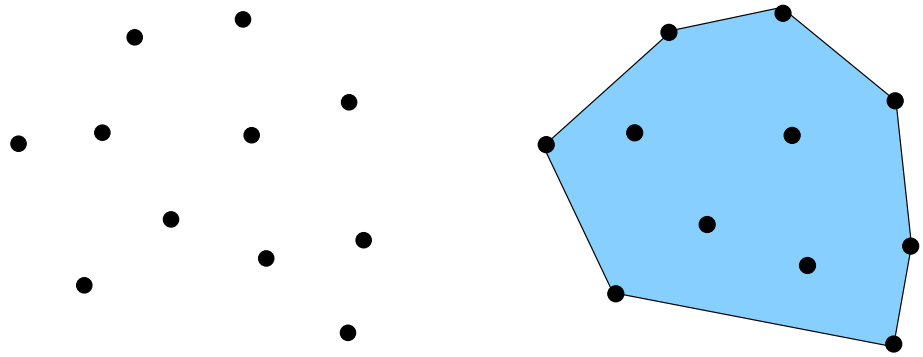
For two points a and b, the line segment joining a and b is denoted $\overline{ab}$. For three non-colinear points a, b and c, $\triangle abc$ denotes the triangle whose vertices are these three points.

2.3 Concepts in Computational Geometry

In this section, we introduce the concept of a triangulation. In order to define this concept, the notion of a convex set and convex hull must first be introduced.

Definition 1. *(Convex set). A set P of points in $\mathbb{R}^2$ is convex if and only if for every pair of points $a, b \in P$, every point on the line segment which joins points a and b is completely contained in P.*

To better illustrate this definition, an example is given. Two different sets are shown in Figure 2.1. The set P in Figure 2.1(a) is convex since each line segment that joins a pair of points a, b in P is completely contained in P, such as $\overline{ab}$. In contrast, the set P shown in Figure 2.1(b) is not convex. For example, we can see that $\overline{ab}$ is not completely contained in P. Having introduced the concept of a convex set, we can now present the definition of the convex hull.



(a) (b)

Figure 2.2: Convex hull example. (a) A set P of points. (b) The convex hull of P.

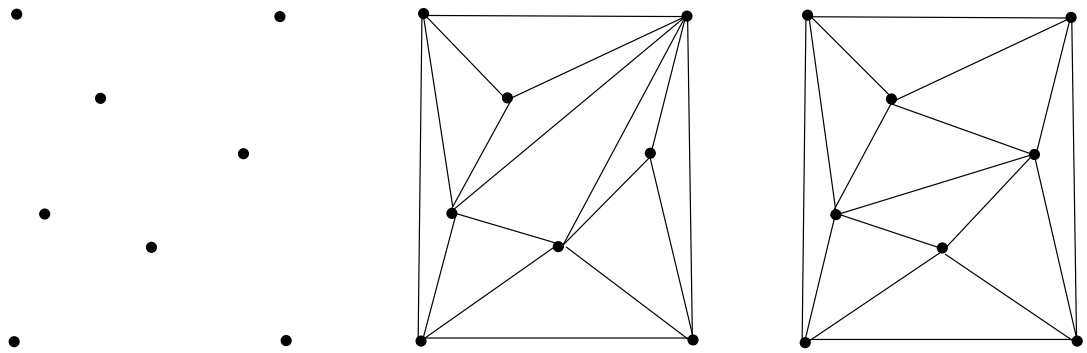
Definition 2. *(Convex hull). The convex hull of a set P of points in $\mathbb{R}^2$ is the intersection of all convex sets that contain P.*

In other words, the convex hull of a set P of points is the "smallest" convex set that contains all the points in P. An example of a convex hull is shown in Figure 2.2. Figure 2.2(a) shows a set P of points, and the shaded area in Figure 2.2(b) is the convex hull of P. The convex hull of P can be viewed as a polygon with vertices in P. A point in P is an **extreme** point (with respect to P) if it is a vertex of the convex hull of P. One way to visualize a convex hull of P is to imagine stretching a rubber band around all the points of P and let the rubber band pull itself taut against the outer points. The resultant polygon formed by the rubber band corresponds to the boundary of the convex hull of these points. With the definition of the convex hull in place, we can now introduce the concept of a triangulation.

Definition 3. *(Triangulation). A triangulation of a finite set P of points in $\mathbb{R}^2$ is a set T of (non-degenerate) triangles satisfying the following conditions:*

1. *the union of all triangles in T is the convex hull of P;*
2. *the union of vertices of all triangles in T is P;*
3. *the interiors of any two triangles in T are disjoint; and*
4. *every edge of a triangle in T joins two and only two points from P.*

For a specified set P of points, a triangulation of P is not usually unique. In fact, the number of possible triangulations typically grows extremely fast with $|P|$. The



(a) (b) (c)

Figure 2.3: Triangulation example (a) a set P of points, (b) a triangulation of P, and (c) another triangulation of P

nonuniqueness of a triangulation is demonstrated in Figure 2.3. The set P of points in Figure 2.3(a) has possible triangulations that include those shown in Figures 2.3(b) and (c).

2.4 Triangle Mesh Models

Before proceeding further, it is necessary to formally introduce the concept of a triangle mesh model. Consider an integer-valued function ϕ defined on $D = [0, W-1] \times [0, H-1]$ and sampled on the truncated two-dimensional integer lattice $\Lambda = \{0, 1, ..., W-1\} \times \{0, 1, ..., H-1\}$ (i.e., a rectangular grid of width W and height H). In the context of our work, a (triangle) mesh model consists of:

1. a set $P = \{p_i\}$ of sample points, where $P \subset \Lambda$;
2. a triangulation T of P; and
3. the function values $\{z_i = \phi(p_i)\}$ for $p_i \in P$.

A mesh model is characterized by a triangulation T covering D (i.e., the domain of function ϕ). Each vertex in T corresponds to a sample point p_i with the function value z_i. In order to ensure that the triangulation T covers all points in Λ, the selected sample points should include all of the extreme convex hull points of D. For example, the extreme convex hull points in D are the four corner points $(0, 0)$, $(W-1, 0)$, $(W-1, H-1)$, and $(0, H-1)$. As a matter of terminology, the **size** and **sampling density** of the model are defined as $|P|$ and $|P|/|\Lambda|$, respectively.

The mesh model is used to represent a function $\hat{\phi}$ that approximates ϕ. In our work, we apply linear interpolation to generate $\hat{\phi}$ based on the function values $\{z_i\}$. In particular, for each face f in the triangulation T of the mesh model, we construct a linear function $\tilde{\phi}$ based on the three vertices of each face to interpolate ϕ. Since the original function ϕ is integer-valued, the function $\hat{\phi}$, which is used to approximate ϕ, should be integer-valued as well. Thus, $\hat{\phi}$ can be defined in terms of the function $\tilde{\phi}$ as $\hat{\phi} = \text{round}(\tilde{\phi})$, where round denotes an operator that rounds to the nearest integer value.

The above mesh modelling process is illustrated in Figure 2.4. Given a bivariate function ϕ shown in Figure 2.4(a), we would like to represent ϕ with a mesh model. A triangulation is first formed by partitioning the image domain of ϕ into a set of triangles, as illustrated in Figure 2.4(b), with the vertices of the triangulation being the sample points. Then, we construct a planar approximating function over each triangle face via linear interpolation. Lastly, these planar approximating functions are combined to form the triangle mesh shown in Figure 2.4(c).

2.5 Halfedge Data Structure

For convenience in specifying algorithms later, we introduce some notation related to the halfedge data structure [16, p. 31] for triangulations. Each (undirected) edge in a triangulation can be viewed as being comprised of a pair of oppositely-oriented directed edges, with each such directed edge being called a **halfedge**. As a matter of notation, for a halfedge h: h.edge denotes the (undirected) edge associated with the halfedge h; h.left denotes the face on the left side of h; h.opp denotes the halfedge associated with h.edge having the opposite orientation to h; h.next and h.prev respectively denote the next and previous halfedge in counterclockwise (CCW) order around h.left. The boolean predicate h.isBorder is defined to be true if h has no left face (due to being on the triangulation boundary) and false otherwise. The meaning of the various halfedge-related notation is further illustrated, by way of an example, in Figure 2.5.

2.6 Triangulation-Cost Optimization Problem

At this point, we would like to formalize the triangulation-cost optimization problem introduced earlier. Formally, a triangulation cost function is a function that maps

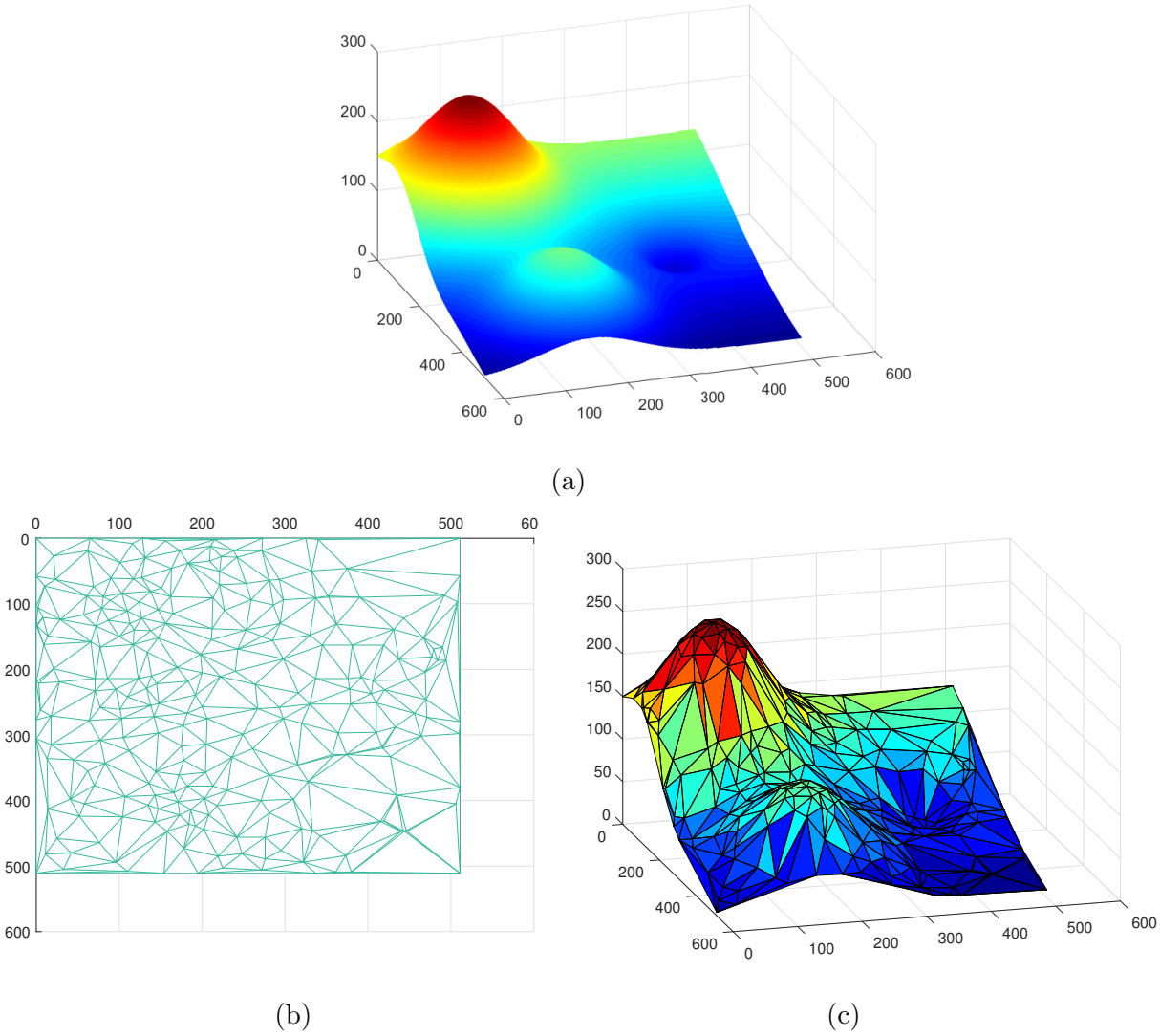


Figure 2.4: Mesh model of a bivariate function ϕ. (a) A bivariate function ϕ, (b) the triangulation in the image domain, and (c) the resulting triangle mesh model.

a triangulation (of some set of points) to a nonnegative real number. With this in mind, the problem that we seek to solve is as follows.

Problem 1 (Triangulation-connectivity selection problem)**.** *Given a set P of points, a triangulation cost function c, and a mapping S from a triangulation of P to a (possibly improper) subset of all triangulations of P, find a triangulation T of P such that*

$$c(T) \leq c(T') \quad \textit{for all } T' \in S(T). \tag{2.1}$$

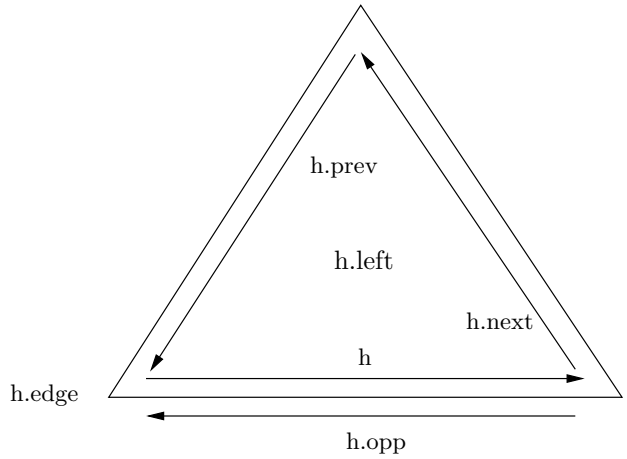


Figure 2.5: Pictorial view of the halfedge data structure.

A triangulation T that satisfies (2.1) is said to be **optimal**. In effect, $S(T)$ specifies a neighbourhood of T in the space of all triangulations of P, and a triangulation T is deemed to be optimal as long as no other triangulation in the neighbourhood $S(T)$ of T has a strictly lower cost than T. If $S(T)$ is chosen as the set of all triangulations of P, then any solution to Problem 1 would be optimal in the global sense (i.e., a global minimizer of the triangulation cost over all possible triangulations of P). Unfortunately, for nearly all cost functions of practical interest, finding a globally optimal solution is computationally intractable. In such cases, $S(T)$ is restricted to a (very small) subset of all possible triangulations of P. When $S(T)$ is chosen as a proper subset of all possible triangulations of P, a solution to Problem 1 is only guaranteed to be locally optimal.

Before proceeding further, it is necessary to introduce some additional notation and terminology relating to triangulations. For an edge e in a triangulation T, $\mathrm{qe}_T(e)$ is used to denote the set of all edges belonging to the (one or two) faces incident on e, and $\mathrm{qf}_T(e)$ is used to denote the set of all (i.e., one or two) faces in T that have e as an edge. An edge e of a triangulation is said to be **flippable** if e has two incident faces (i.e., is not on the triangulation boundary) and the union of these two faces is a strictly convex quadrilateral q. Figure 2.6 presents three examples of flippable and unflippable edges. As we can see in Figure 2.6(a), $\mathrm{qf}_T(e)$ forms a strictly convex quadrilateral, so the edge e is flippable. In each of Figures 2.6(b) and (c), $\mathrm{qf}_T(e)$ does not form a strictly convex quadrilateral, so e is unflippable. For a flippable edge e, an **edge flip** is an operation that replaces the edge e in the triangulation by the other diagonal e' of q, as shown in Figure 2.7. As a matter of terminology, we refer to a

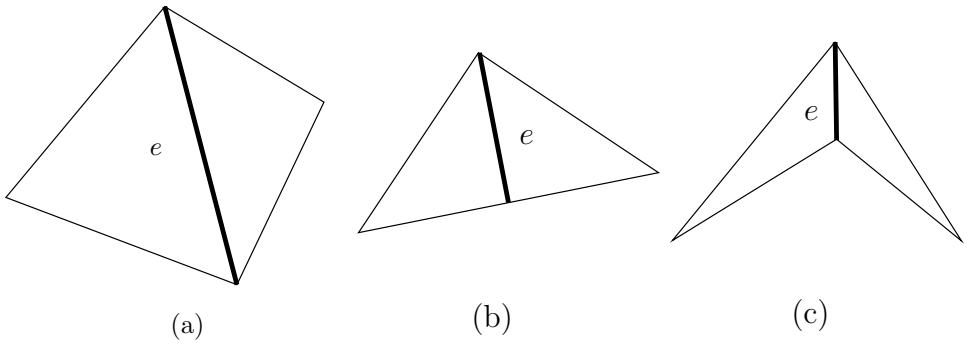


Figure 2.6: An example of parts of a triangulation with flippable and unflippable edges. (a) Flippable edge e, (b) unflippable edge e, and (c) unflippable edge e.

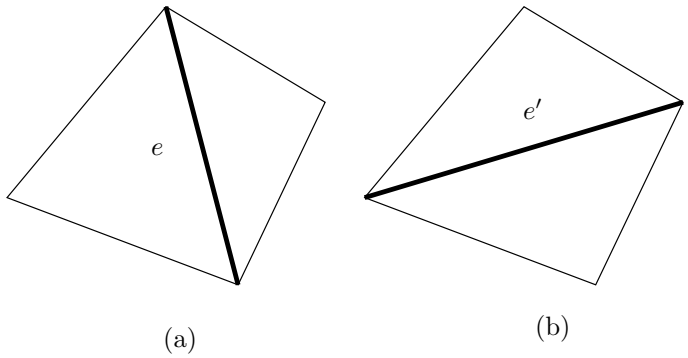


Figure 2.7: An example of an edge flip in part of triangulation. (a) Before and (b) after an edge flip is applied to the edge e.

sequence of edges to which edge flips are to be applied as a **flip sequence**. With an appropriate choice of flip sequence $s = (e_0, e_1, \ldots, e_{n-1})$, a given triangulation $T = T_0$ can be transformed to another triangulation $T' = T_n$ by applying s to T. Let T_i denote the triangulation obtained after applying the first i edge flips in the sequence s to T. If, for each $i \in \{0, 1, \ldots n-1\}$, the edge e_i is a flippable edge in T_i, the flip sequence is said to be **valid** (with respect to triangulation $T = T_0$). A flip sequence can only be applied to a triangulation T if it is valid with respect to T. A triangulation T' is said to be **reachable** from triangulation T by the flip sequence s if applying s to T yields T'.

It is a well-known fact [24, 29] that, for any two triangulations T and T' of a set of points, there exists a finite-length flip sequence that will transform T to T'. In other words, every triangulation of a set of points is reachable from every other triangulation of the same set of points by a finite number of edge flips. This fact leads to a class of approaches to solving Problem 1 that is based on edge flips. In

our work, we consider this particular class of methods exclusively. The basic idea in these approaches is to repeatedly apply edge flips to a triangulation until an optimal triangulation is obtained.

2.7 Local Optimization Procedure (LOP) and Lookahead LOP (LLOP)

One very well known method for solving Problem 1 is the **local optimization procedure (LOP)** proposed by Lawson [25]. Effectively, the LOP chooses $S(T) = S_{\mathsf{LOP}}(T)$ in (2.1), where $S_{\mathsf{LOP}}(T)$ is the set of all triangulations of P that are reachable from T by a single edge flip. Thus, the LOP is guaranteed to yield a triangulation T such that no single edge flip applied to T can yield a new triangulation with a strictly lower cost than T.

Yu, Morse, and Sederberg [44] proposed an interesting variant of the LOP, called the LOP with lookahead, hereafter referred to as the **lookahead LOP (LLOP)**. The LLOP chooses $S(T) = S_{\mathsf{LLOP}}(T)$, where $S_{\mathsf{LLOP}}(T) = S_{\mathsf{LOP}}(T) \cup \Gamma(T)$ and $\Gamma(T)$ is the set of all triangulations that are reachable from T by a valid length-2 flip sequence of the form (e_0, e_1) where $e_1 \in \mathrm{qe}_T(e_0)$ and $\mathrm{qe}_T(e_0)$ is defined as earlier. The LLOP tends to produce optimal triangulations of much lower cost than the LOP, but at an increased computational cost [44].

2.8 Triangulation Cost Function

In Problem 1, many choices are possible for the triangulation cost function c. In practice, however, a triangulation cost function is typically defined as an accumulation of costs for individual triangulation elements, such as edges or faces. This leads to the two general forms for the triangulation cost function considered herein: 1) edge based and 2) face based. An **edge-based cost function** c has the form

$$c(T) = \sum_{e \in \mathcal{E}(T)} \mathrm{edgeCost}(e), \tag{2.2}$$

where $\mathcal{E}(T)$ denotes the set of all edges in T and $\mathrm{edgeCost}(e)$ denotes the cost of the edge e (i.e., the cost of a triangulation is defined as the sum of its edge costs). A

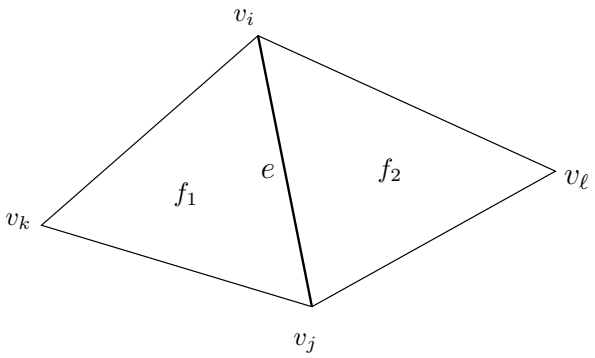


Figure 2.8: an quadrilateral $v_iv_jv_kv_l$ with edge $e = \overline{v_iv_j}$ as diagonal

face-based cost function c has the form

$$c(T) = \sum_{f \in \mathcal{F}(T)} \text{faceCost}(f), \tag{2.3}$$

where $\mathcal{F}(T)$ denotes the set of all faces in T and faceCost(f) denotes the cost of the face f (i.e., the cost of a triangulation is defined as the sum of its face costs). Of these two forms of cost function, the edge-based form is the most common.

For the purposes of discussion herein, we consider the following seven well-known cost functions:

1. **angle between normals (ABN)** [19, Equation 3];
2. **absolute mean curvature (AMC)** [10, Section 2.2] and [42, Section 2];
3. **deviations from linear polynomials (DLP)** [19, Section 3.1];
4. **distances from planes (DP)** [19, Section 3.1];
5. **jump in normal derivatives (JND)** [19, Section 3.1];
6. **squared error (SE)** [31, Equation 1] and [33, Section 2]; and
7. **Yu-Morse-Sederberg (YMS)** [44, Equation 3].

Of the cost functions listed above, SE is face based and all of the others are edge based.

In order to provide the formal definitions of these cost functions, we must first introduce some notation. In the case that edge e is not a boundary edge, the two incident faces of e are denoted as f_1 and f_2. Each vertex in the triangulation is denoted as $v_n = (x_n, y_n)$. We denote $\phi(x_n, y_n)$ as z_n. The faces f_1 and f_2 are represented by $\triangle v_iv_kv_j$ and $\triangle v_iv_jv_\ell$ respectively. This notation is illustrated in Figure 2.8. Let P_1 and P_2 denote the linear interpolant over the faces f_1 and f_2, respectively. The

functions P_1 and P_2 are given by

$$P_1(x, y) = a_1x + b_1y + c_1 \text{ and}$$
$$P_2(x, y) = a_2x + b_2y + c_2.$$

At this point, we would like to provide the definition of edgeCost for each of the triangulation costs considered herein in (2.2). For a non-boundary edge e (which must have two incident faces) in the triangulation T, the edgeCost function in the cost of the ABN, AMC, DLP, DP, JND, and YMS triangulation costs is respectively given by

$$\text{edgeCost}_{\text{ABN}}(e) = \arccos\left[\frac{(a_1, b_1, -1) \cdot (a_2, b_2, -1)}{||(a_1, b_1, -1)||\ ||(a_2, b_2, -1)||}\right], \tag{2.4}$$
$$\text{edgeCost}_{\text{AMC}}(e) = ||v_i - v_j||\,\text{edgeCost}_{\text{ABN}}(e), \tag{2.5}$$
$$\text{edgeCost}_{\text{DLP}}(e) = ||(|P_1(x_\ell, y_\ell) - z_\ell|, |P_2(x_k, y_k) - z_k|)||, \tag{2.6}$$
$$\text{edgeCost}_{\text{DP}}(e) = ||(\text{dist}(P_1, (x_\ell, y_\ell, z_\ell)), \text{dist}(P_2, (x_k, y_k, z_k)))||, \tag{2.7}$$
$$\text{edgeCost}_{\text{JND}}(e) = |(n_x, n_y) \cdot [(a_1, b_1) - (a_2, b_2)]|, \text{ and} \tag{2.8}$$
$$\text{edgeCost}_{\text{YMS}}(e) = ||(a_1, b_1)||\ ||(a_2, b_2)|| - (a_1, b_1) \cdot (a_2, b_2), \tag{2.9}$$

where (n_x, n_y) is a unit vector normal to e, and $\text{dist}(P_m, (x, y, z))$ denotes the distance between the plane P_m and the point (x, y, z). That is, $\text{dist}(P_m, (x, y, z)) = \frac{|P_m(x,y)-z|}{||(a_m,b_m,-1)||}$.

For a boundary edge e (which has only one incident face), we just simply define the edgeCost(e) as 0.

Next, we provide the definition of faceCost used in (2.3). Of the cost functions considered herein, only the cost function of SE is face-based. For a face f, faceCost(f) for the SE triangulation cost is given by

$$\text{faceCost}_{\text{SE}}(f) = \sum_{p \in \mathcal{P}(f)} (\hat{\phi}(p) - \phi(p))^2. \tag{2.10}$$

where ϕ is the function being modelled, $\hat{\phi}$ is a function that approximates ϕ (as defined earlier in Section 2.4), and $\mathcal{P}(f)$ denotes the set of all points in Λ belonging to the face f in the triangulation T.

Chapter 3

Proposed Methods and Their Evaluation

3.1 Overview

In this chapter, we first refine the definition of optimality for the triangulation-connectivity optimization problem by introducing the notion of n-flip optimality. Then, we introduce the proposed computational framework MLOP with a number of free parameters. Some particularly effective choices are recommended for these parameters, leading to the two MLOP-based methods proposed herein, $\text{MLOP}_{\mathsf{B}}(M, L)$ and $\text{MLOP}_{\mathsf{C}}(L)$. After that, we prove the conditions needed for a strong optimality property, namely 2-flip optimality. Then, we discuss when the LOP, LLOP and the MLOP-based methods yield 2-flip optimal triangulations. According to the discussion, our MLOP-based methods are shown to yield triangulations that satisfy a stronger optimality condition than the LOP and LLOP for most triangulation cost functions. Next, the performance of our proposed methods are evaluated by comparing to the other competing schemes, with our proposed methods proving to be superior. Moreover, the MLOP_{B} and MLOP_{C} methods are shown to yield even better results by increasing the parameter i in the $\text{MLOP}_{\mathsf{B}}(2, i)$ or in the $\text{MLOP}_{\mathsf{C}}(i)$.

3.2 n-Flip Optimality

The optimization in Problem 1 can be viewed in terms of a topological space whose underlying set is the set of all triangulations of P. The distance between two triangu-

lations T and T' in this metric space is defined as length of the shortest flip sequence that transforms T to T'. A good way to visualize this space is as a graph, where each node in the graph corresponds to a distinct triangulation, and two nodes are connected by an edge if their corresponding triangulations are reachable from one another by a single edge flip. We suggest that the goodness of an optimal triangulation T can be measured by how far one must move away from T in the solution space in order to obtain a triangulation with a strictly lower cost than T. This view leads to a very natural way to define a notion of optimality, as follows.

Definition 4 (n-flip optimality)**.** *A triangulation T is said to be n-flip optimal with respect to the triangulation cost function c if the following condition holds:*

$$c(T) \leq c(T') \quad \textit{for all } T' \in S_{\mathsf{NFO}}(T, n), \tag{3.1}$$

where $S_{\mathsf{NFO}}(T, n)$ is the set of all triangulations that are reachable from T by a (valid) flip sequence of length $\ell \leq n$.

As n increases, any given n-flip optimal triangulation will tend to more closely approach the globally optimal solution in terms of cost. Moreover, for some finite (but typically very large) n, an n-flip optimal triangulation will be a global minimizer of c.

For any triangulation cost function, the LOP and LLOP are both trivially guaranteed to yield a triangulation that is 1-flip optimal. As we will show later (in Section 3.4), however, for most cost functions of practical interest, the LOP and LLOP are unable to guarantee n-flip optimality for any $n > 1$. The desire to improve upon the 1-flip optimality of the LOP and LLOP served as a partial motivation for the development of our improved version of the LOP (namely, the MLOP) which we introduce next.

3.3 Proposed LOP Variant: The Modified LOP (MLOP)

Before presenting our proposed MLOP approach, we must first introduce the key concept of a permissible flip sequence as well as some other notation and terminology. In the optimization for Problem 1, rather than define S by directly specifying the triangulations in $S(T)$, we elect to instead specify $S(T)$ indirectly in terms of flip

sequences as follows. For any flippable edge e_0 in T, we can define a rule that specifies the set of flip sequences starting with e_0 that are permitted to be applied to T in an attempt to reduce the triangulation cost during optimization. We denote this set of flip sequences as $\text{permFlipSeqs}_T(e_0)$. The rule embodied by permFlipSeqs is known as a **permissible flip-sequence policy**. A flip sequence s to be applied to T is said to be **permissible** if $s \in \text{permFlipSeqs}_T(e_0)$ for some flippable edge e_0 in T. The quantity $S(T)$ is then defined as the set of all triangulations that are reachable by applying a permissible flip sequence to T (according to rule embodied by permFlipSeqs). Thus, an optimal triangulation effectively becomes a triangulation for which no permissible flip sequence exists that can transform the triangulation to a triangulation of strictly lower cost. As for the particular manner in which permFlipSeqs might be defined, we will consider this later. For now, all we will say in this regard is that the set $\text{permFlipSeqs}_T(e_0)$ must be such that, for every flippable edge e_0 in a triangulation T,

$$(e_0) \in \text{permFlipSeqs}_T(e_0). \tag{3.2}$$

(i.e., $\text{permFlipSeqs}_T(e_0)$ always includes the length-1 flip sequence (e_0)). As a matter of terminology, an edge e_0 in a triangulation T is said to be **optimal** if no permissible flip sequence starting with e_0 exists that maps T to T' for which $c(T') < c(T)$. Since every permissible flip sequence must start with a flippable edge, an unflippable edge is, by definition, optimal. Thus, an optimal triangulation simply becomes a triangulation for which all of its flippable edges are optimal.

Next, we introduce the concept of a suspect edge and some related notation. As a matter of terminology, an edge whose optimality is uncertain is said to be **suspect**. Consider a flip sequence s to be applied to the triangulation T to yield the new triangulation T'. When s is applied, the optimality of some edges will potentially be affected, resulting in some edges in T' being suspect (i.e., their optimality is uncertain). The set of all edges in T' whose optimality is potentially affected if the flip sequence s were applied to T is denoted $\text{suspects}(T, s)$. Generally, $\text{suspects}(T, s)$ can potentially include new edges (i.e., edges that are in T' but not T) as well as old edges (i.e., edges that are in both T' and T).

With the above background in place, we are now ready to present the general algorithmic framework of our proposed MLOP. Given a triangulation and triangulation cost function as input, the MLOP produces an optimal triangulation as output

by performing that following processing. Initially, each edge in the triangulation is marked as suspect if flippable, and not suspect otherwise. Then, the algorithm iterates performing the following steps until no suspect edges remain: Select a suspect edge e_0 and mark it as not suspect. Test if there exists a permissible flip sequence starting with e_0 that, if applied to the current triangulation T, would strictly reduce the triangulation cost. If such a flip sequence exists, apply it, and then determine which edges become suspect due to the application of this flip sequence and mark these edges as such. The MLOP is given in more detail in pseudocode form in Algorithm 1.

Algorithm 1 Modified LOP (MLOP)

1: Set the current triangulation T to the input triangulation.
2: For each edge e in T, mark e as suspect if it is flippable; otherwise, mark e as not suspect.
3: **while** suspect edges remain in T **do**
4: Select a suspect edge e_0 in T and mark e_0 as not suspect.
5: Clear the list ℓ of good flip sequences.
6: **if** e_0 is flippable **then**
7: Let p denote the set of all permissible flip sequences starting with e_0, as determined by the permissible flip-sequence policy (i.e., $p = \text{permFlipSeqs}_T(e_0)$).
8: **for** each flip sequence s in p **do**
9: **if** applying s to T would strictly reduce the triangulation cost **then**
10: Add s to ℓ.
11: **endif**
12: **endfor**
13: **if** ℓ is not empty **then**
14: Select a sequence s from ℓ, using the good flip-sequence selection policy (i.e., $s = \text{selGood}(\ell)$).
15: Update T by applying s to T.
16: Determine the set σ of all edges in T that become suspect as a result of s being applied to T (i.e., $\sigma = \text{suspects}(T, s)$).
17: Mark each edge s in σ as suspect (if not already so marked).
18: **endif**
19: **endif**
20: **endwhile**
21: Output T as the final triangulation.

As can be seen from Algorithm 1, the MLOP has two free parameters that control the behavior of the algorithm: 1) the permissible flip-sequence policy and 2) the good

flip-sequence selection policy. The permissible flip-sequence policy permFlipSeqs is used in step 7 of the algorithm to determine which flip sequences must be considered when testing an edge for optimality. The good flip-sequence selection policy selGood is employed in step 14. When an edge is tested for optimality, more than one permissible flip sequence may be found that can be used to reduce the triangulation cost. The good-flip-sequence selection policy determines how this sequence is to be chosen when more than one choice is available. Before presenting all of the particular choices for permFlipSeqs and selGood considered in our work, we need to introduce some additional background and notation. So, we will defer the discussion of these policies until later.

At this point, it is worthwhile to note that the MLOP is essentially a generalization of the LOP and LLOP, as the MLOP includes the LOP and LLOP as special cases. In particular, the LOP and LLOP are equivalent to the MLOP with the permissible flip-sequence policy $\text{permFlipSeqs}_T(e_0)$ chosen as $\text{PFS}_{\mathsf{LOP}}$ and $\text{PFS}_{\mathsf{LLOP}}$, respectively, where

$$\text{PFS}_{\mathsf{LOP}}(e_0) = \{(e_0)\} \quad \text{and} \tag{3.3}$$

$$\text{PFS}_{\mathsf{LLOP}}(e_0) = \text{PFS}_{\mathsf{LOP}}(e_0) \cup \Gamma(e_0), \tag{3.4}$$

and $\Gamma(e_0)$ is set of all valid flip sequences of the form (e_0, e_1) where $e_1 \in \text{qe}_T(e_0)$. As we will see later, however, much better choices for permFlipSeqs are possible than $\text{PFS}_{\mathsf{LOP}}$ and $\text{PFS}_{\mathsf{LLOP}}$.

In practice, MLOP-like algorithms (i.e., the MLOP, LOP and LLOP) are only guaranteed to yield a triangulation that is locally optimal. Furthermore, for triangulation cost functions of practical interest, the optimal triangulation obtained will typically depend quite heavily on the triangulation used as the starting point for optimization. Moreover, some cost functions (such as SE) are such that MLOP-like algorithms are very likely to converge to a very poor local minimum unless some care is exercised in the choice of initial triangulation to be used for optimization. For this reason, it can sometimes be advantageous to apply the MLOP in multiple stages. For example, the MLOP could be applied twice, with different parameters each time (e.g., different triangulation costs and/or permissible flip sequence policies), where the optimal triangulation obtained from the first optimization is then used as the initial triangulation for the second optimization. In such a scenario, the second invocation of the MLOP performs the true optimization of interest, while the first invocation is

simply intended as a preconditioning step that allows the second optimization to be seeded with an initial triangulation that is in a more desirable region of the solution space (i.e., in a region containing lower-cost optimal triangulations). In our work, we found this type of two-stage processing to be quite advantageous, sometimes leading to methods that are both faster and yield lower-cost triangulations. As it turns out, one of the methods that we recommend later in this thesis is based on this type of two-stage processing.

3.3.1 Additional Notation

Before proceeding further, we need to introduce some additional notation and terminology related to the MLOP that is used in various places throughout the remainder of this thesis. For an edge e in a triangulation T and a nonnegative integer i, $\text{layers}_T(e, i)$ denotes the set of edges in T given by

$$\text{layers}_T(e,i) = \begin{cases} e & i = 0 \\ \bigcup_{e \in \text{layers}_T(i-1)} \text{qe}_T(e) & i \geq 1, \end{cases} \tag{3.5}$$

where $\text{qe}_T(e)$ is as defined earlier (on page 13). The distance between the edges e_0 and e_1 in the triangulation T, denoted $d_T(e_0, e_1)$, is defined as the smallest nonnegative integer k for which $e_1 \in \text{layers}_T(e_0, k)$. (Note that $d_T(e_0, e_1) = d_T(e_1, e_0)$ and $d_T(e_0, e_0) = 0$.) Loosely speaking, the distance between edges is the number of edges that must be crossed to move from one edge to the other in the triangulation, subject to the constraint that edges must be crossed in their interiors, not at their endpoints.

We now define the **influence distance** of a cost function c, denoted $\text{inflDist}(c)$. Let S be set of edges (excluding e_0) whose optimality might change as a result of the edge e_0 being flipped if permissible flip sequences were restricted to only single edge flips. Then, $\text{inflDist}(c)$ is the smallest nonnegative integer k for which it is always guaranteed that $S \subset \text{layers}_T(e_0, k)$. Essentially, $\text{inflDist}(c)$ measures the size of the region of influence of an edge. That is, the larger $\text{inflDist}(c)$ is, the more edges can have their optimality affected by the flip of a single edge. For the case of the triangulation cost functions considered herein, it can be shown [27, 28] that

$$\text{inflDist}(c) = \begin{cases} 1 & \text{if } c \text{ is SE} \\ 2 & \text{if } c \text{ is ABN, AMC, DLP, DP, JND, YMS.} \end{cases} \tag{3.6}$$

(Generally speaking, any reasonable cost function c will be such that $\text{inflDist}(c) \geq 1$.)

3.3.2 Good Flip-Sequence Selection Policies

One of the free parameters of the MLOP, as introduced earlier in Algorithm 1, is the good flip-sequence selection policy (i.e., selGood). When the MLOP tests an edge for optimality, more than one permissible flip sequence may be found that can strictly reduce the triangulation cost. When more than one sequence is found, the MLOP must select one of them (in step 14 of the algorithm). In our work, we considered numerous good flip-sequence selection policies, including the following (as well as others):

1. **first found**, which chooses the first good flip sequence that was found;
2. **least cost**, which chooses the good flip sequence that results in a triangulation with the least cost; and
3. **random**, which randomly selects one of the good flip sequences.

Our experiments showed, however, that there was no clear benefit to one of the policies over another. That is, no one policy consistently outperformed all of others. Since, all other things being equal, the first-found policy is the least computationally expensive, we advocate its use in the MLOP and only consider this particular choice (for the good flip-sequence selection policy) in the remainder of this thesis.

3.3.3 Permissible Flip-Sequence Policies

Another one of the free parameters of the MLOP, as introduced earlier in Algorithm 1, is the permissible flip-sequence policy (i.e., permFlipSeqs). This policy determines the particular set of flip sequences that are considered when testing an edge for optimality. In effect, this policy determines the specific sense in which the final triangulation produced by the MLOP is optimal. Since permFlipSeqs is required to satisfy (3.2), the final triangulation is always 1-flip optimal. If this policy is chosen appropriately, however, the MLOP can be guaranteed to produce triangulations that are n-flip optimal for some $n > 1$.

Clearly, even with the (relatively minor) constraint imposed by (3.2), very many choices are possible for the permissible flip-sequence policy permFlipSeqs. In our work, we considered many different policies, and through extensive experimentation, we were able to find three policies that were particularly effective, which we present herein. As it turns out, all three of these policies can be most easily described in terms

of a recursive algorithm that generates permFlipSeqs$_T(e_0)$. So, in what follows, we first introduce this algorithm and its parameters, and then use this algorithm to define the three policies proposed herein.

The recursive algorithm for generating the set permFlipSeqs$_T(e_0)$ is best described by viewing the triangulation in terms of halfedges, as introduced earlier in Section 2.5 (on page 11). To determine permFlipSeqs$_T(e_0)$ for the edge e_0, the algorithm starts by visiting one of the halfedges h of e_0. For each halfedge h visited, a new flip sequence is potentially generated (depending on whether the edge h.edge is flippable and other parameters of the flip sequence generating process). Then, the algorithm recursively visits zero or more of the neighbouring halfedges of h (in particular, halfedges associated with edges in qe(h.edge)). The recursion stops if it would cause the algorithm to attempt to move outside the triangulation or some maximum recursion depth is reached. In more precise terms, the process for generating the set of flip sequences in permFlipSeqs$_T(e_0)$ is given by Algorithm 2. This pseudocode utilizes the functions `flip`, `unflip`, and `append`, which are defined as follows. The function `append(seq, e)` returns the flip sequence formed by appending the edge `e` to the flip sequence `seq`. The functions `flip` and `unflip` each perform an edge flip. Suppose that `h.edge` has two incident faces whose union forms a strictly convex quadrilateral Q. The function `flip(h)` flips the edge `h`.edge by rotating `h` by one vertex around Q in the CCW direction and returns the new rotated halfedge. The `unflip` function is identical to `flip` except that the rotation is in the opposite (i.e., CW) direction. That is, `unflip(h)` undoes the effect of `flip(h)`.

The above algorithm (i.e., Algorithm 2) has four input parameters (which are not passed as explicit function parameters in the pseudocode) that control the flip-sequence generation process: 1) `maxLevel`, 2) `inward`, 3) `skip`, and 4) `maxLength`. The parameter `maxLevel` (which is a nonnegative integer) determines the size of the region in which edge flips can take place and imposes a maximum flip-sequence length of `maxLevel` $+ 1$, while the parameters `inward`, `skip`, and `maxLength` control the precise patterns of edge flips that can occur within this region. In particular, the `maxLength` parameter can be used to restrict the maximum length of the flip sequences generated to a value less than `maxLevel`$+1$. That is, the longest possible flip sequence that can be generated by the above algorithm is $\min\{$`maxLevel`$+1,$ `maxLength`$\}$. The flip sequences generated are placed in the global variable `seqSet`. For a given edge e_0, permFlipSeqs$_T(e_0)$ is generated by invoking `makeSeqs(h)` where `h` is chosen as one of the halfedges of e_0. The generated set is placed by the algorithm in `seqSet`.

Algorithm 2 Algorithm for generating set of permissible flip sequences.

```
1: // h: halfedge
2: procedure makeSeqs(h)
3:     Clear seqSet (i.e., the set of permissible flip sequences).
4:     makeSeqs2(0, h, {}, true)
5: endprocedure//
6: // level: current level in recursion
7: // h: current halfedge
8: // seq: current flip sequence
9: // doFlip: edge flip should be performed in current state
10: procedure makeSeqs2(level, h, seq, doFlip)
11:     if level > maxLevel then return endif
12:     if doFlip then
13:         seq = append(seq, h.edge)
14:         if length of seq <= maxLength and seq is not in seqSet then add to
    seqSet endif
15:         h = flip(h)
16:     endif
17:     if not h.opp.isBorder then
18:         if h.opp.next is flippable then makeSeqs2(level + 1, h.opp.next,
    seq, true) endif
19:         if h.opp.prev is flippable then makeSeqs2(level + 1, h.opp.prev,
    seq, true) endif
20:         if skip then
21:             makeSeqs2(level + 1, h.opp.next, seq, false)
22:             makeSeqs2(level + 1, h.opp.prev, seq, false)
23:         endif
24:     endif
25:     if not h.isBorder and (inward or level == 0) then
26:         if h.next is flippable then makeSeqs2(level + 1, h.next, seq, true)
    endif
27:         if h.prev is flippable then makeSeqs2(level + 1, h.prev, seq, true)
    endif
28:         if skip then
29:             makeSeqs2(level + 1, h.next, seq, false)
30:             makeSeqs2(level + 1, h.prev, seq, false)
31:         endif
32:     endif
33:     if doFlip then h = unflip(h) endif
34: endprocedure
```

Table 3.1: Parameter selections for defining various permissible flip-sequence policies using Algorithm 2.

Policy	Parameter			
	`maxLevel`	`inward`	`skip`	`maxLength`
$\text{PFS}_{\mathsf{IO}}(\text{L})$	L	`true`	`false`	$\infty^{\dagger}$
$\text{PFS}_{\mathsf{IOS}}(\text{L})$	L	`true`	`true`	$\infty^{\dagger}$
$\text{PFS}_{\mathsf{MLT}}(\text{L})$	L	`false`	`true`	2
$\text{PFS}_{\mathsf{LOP}}$	0	`false`	`false`	1
$\text{PFS}_{\mathsf{LLOP}}$	1	`false`	`false`	2

†Any value greater than or equal to $L+1$ is equivalent to ∞.

With the above algorithm for generating $\text{permFlipSeqs}_T(e_0)$ in place, we can now introduce the three permissible flip-sequence policies proposed herein. These three policies are known as: 1) **inward and outward (PFS_{IO})**, 2) **inward and outward with skip ($\text{PFS}_{\mathsf{IOS}}$)**, and 3) **maximum length two ($\text{PFS}_{\mathsf{MLT}}$)**. Each of these policies has a single control parameter. To denote the policy with a particular choice of parameter, we append a parameter list to the name of the policy. For example, $\text{PFS}_{\mathsf{IO}}(L)$ denotes the PFS_{IO} with its single control parameter chosen as L. The preceding three policies are obtained by using Algorithm 2 with the particular parameter choices specified in Table 3.1. These three policies were chosen to allow for a wide range of trade offs between result quality (i.e., triangulation cost) and computation time. As it turns out, the policies used by the LOP and LLOP (namely, $\text{PFS}_{\mathsf{LOP}}$ and $\text{PFS}_{\mathsf{LLOP}}$) can also be expressed in terms of Algorithm 2, where the necessary parameter choices are also included in Table 3.1.

In step 8 of the MLOP algorithm introduced earlier (i.e., Algorithm 1), the order in which elements of $\text{permFlipSeqs}_T(e_0)$ are processed will influence the optimal triangulation produced. In our implementation of the MLOP, we chose to process the elements of $\text{permFlipSeqs}_T(e_0)$ in an order that corresponds to a breadth-first traversal of the recursion tree associated with the `makeSeqs` function. We made this choice of ordering as it had some efficiency advantages in our implementation and experiments suggest that quality of results obtained from the MLOP do not clearly favor a particular ordering.

3.3.4 Determination of Suspect Edges

As this point, our description of the MLOP is almost complete. What remains is an explanation of how the set of suspect edges is determined in step 16 of Algorithm 1 after a flip sequence is applied to a triangulation. We will now consider this issue.

Let $\text{propagate}_T(F, n)$ denote an n-level recursive process for augmenting a set of faces F in a triangulation T to yield a new set of faces given by

$$\text{propagate}_T(F, n) = \begin{cases} F & n = 0 \\ \text{propagate}_T(F, n-1) \cup \left(\cup_{f \in \text{propagate}_T(F,n-1)} \text{neighFaces}(f)\right) & n \geq 1, \end{cases}$$

where $\text{neighFaces}_T(f)$ denotes the set of all faces in T sharing a common edge with the face f. Essentially, the propagate operator computes n iterations of an outward propagating wavefront of faces in the triangulation T.

Let us consider the application of the flip sequence s to the triangulation T to obtain the new triangulation T'. Let $\text{newEdges}(T, s)$ denote the set of new edges in T' generated by applying s to T (to yield T'). Note that, for the purposes of this definition, an edge in T that is deleted by a flip in s and then reintroduced by a later flip in s is deemed to be new. We define the **flip-affected region** resulting from the application of s to T, denoted $\text{flipAffReg}(T, s)$, as the set of faces in T' given by

$$\text{flipAffReg}(T, s) = \cup_{e \in \text{newEdges}(T,s)} \text{qf}_{T'}(e),$$

where qf is as defined earlier (on page 13). With the above definitions in place, the set of edges that are suspect in the triangulation T' obtained from applying s to T, denoted $\text{suspects}(T, s)$, is given by

$$\text{suspects}(T, s) = \text{edges}\left(\text{propagate}_{T'}(\text{flipAffReg}(T, s), L + \text{inflDist}(c) - 1)\right), \quad (3.7)$$

where $\text{edges}(F)$ denotes the set of all edges belonging to faces in the set F of faces and L is the value of the `maxLevel` parameter for the permissible flip sequence policy being used (as given earlier in Table 3.1).

3.3.5 Proposed Methods for Using the MLOP

Having fully specified the MLOP and defined the various choices for the free parameters of the algorithm, we are now ready to introduce the specific approaches that we

have developed for using the MLOP to solve Problem 1. In our work, we considered numerous combinations of single-stage and multi-stage MLOP approaches with various choices of MLOP free parameters. Extensive experimentation ultimately led us to three particular methods of interest, which we present herein. These three methods are known by the names MLOP_{A}, MLOP_{B}, and MLOP_{C}. All three of these methods use first found as the good flip-sequence selection policy for the MLOP. The difference between these methods is in whether they employ one or multiple stages of the MLOP and which permissible flip-sequence policy is employed for the MLOP. The first method, MLOP_{A}, is a single stage method (i.e., applies the MLOP only once) and has a single control parameter. To denote this method used with the control parameter chosen as L, we write $\text{MLOP}_{\mathsf{A}}(L)$. In the case of $\text{MLOP}_{\mathsf{A}}(L)$, the MLOP is simply applied with the permissible flip-sequence policy $\text{PFS}_{\mathsf{MLT}}(L)$. The second method, MLOP_{B}, is a two-stage method (i.e., applies the MLOP twice) and has two control parameters, one for each stage. To denote this method used with the first and second control parameters chosen as L and M, respectively, we write $\text{MLOP}_{\mathsf{B}}(L, M)$. In the case of $\text{MLOP}_{\mathsf{B}}(L, M)$, the MLOP is first applied with the permissible flip-sequence policy $\text{PFS}_{\mathsf{IO}}(M)$; and then the MLOP is applied to the resulting triangulation with the permissible flip-sequence policy $\text{PFS}_{\mathsf{MLT}}(L)$. The third method, MLOP_{C}, is a single-stage method and has a single control parameter. To denote this method used with the control parameter set to L, we write $\text{MLOP}_{\mathsf{C}}(L)$. In the case of $\text{MLOP}_{\mathsf{C}}(L)$, the MLOP is simply applied with the permissible flip-sequence policy $\text{PFS}_{\mathsf{IOS}}(L)$.

As will be shown later in Section 3.4, for $L \geq 2$, each of our $\text{MLOP}_{\mathsf{A}}(L)$, $\text{MLOP}_{\mathsf{B}}(L, M)$, and $\text{MLOP}_{\mathsf{C}}(L)$ methods is guaranteed to produce 2-flip optimal triangulations for all of the triangulation cost functions considered herein (as well as others with a similar mathematical structure). The methods MLOP_{B} and MLOP_{C} were both chosen due to their effectiveness at producing good (i.e., low cost) optimal triangulations, but make different trade offs between triangulation quality and computational cost. The method MLOP_{A} was chosen solely for comparison purposes, as it is the simplest MLOP-based scheme that is guaranteed to yield 2-flip optimal triangulations for all cost functions considered herein.

3.4 Optimality Properties of the MLOP, LOP and LLOP

Having introduced the MLOP and explained its relationship to the LOP and LLOP, we would now like to study the optimality properties of the MLOP, LOP, and LLOP. Earlier, we claimed that, for most cost functions of practical interest, the LOP and LLOP are only guaranteed to yield triangulations that are 1-flip optimal. Furthermore, we claimed that, for $L \geq 2$, each of our MLOP-based methods $\text{MLOP}_{\mathsf{A}}(L)$, $\text{MLOP}_{\mathsf{B}}(L, M)$, and $\text{MLOP}_{\mathsf{C}}(L)$ is guaranteed to yield 2-flip optimal triangulations for all of the cost functions considered herein. In what follows, we will now prove each of these claims to be true.

To begin, we introduce the following condition on the permissible flip sequence policy that is both necessary and sufficient to guarantee that the MLOP will always yield a 2-flip optimal triangulation:

Theorem 1 (Condition for 2-flip optimality of the MLOP)**.** *The MLOP with triangulation cost function c is guaranteed to yield a 2-flip optimal triangulation if and only if the permissible flip-sequence policy* permFlipSeqs_T *satisfies* (3.2) *and* $\text{permFlipSeqs}_T(e_0)$ *includes the flip sequence* (e_0, e_1) *for each* e_1 *satisfying* $d_T(e_0, e_1) \leq \text{inflDist}(c)$.

Proof. To begin, we consider the sufficiency of condition stated in the theorem. For 2-flip optimality, we must show that the application of any flip sequence of exactly length one or exactly length two cannot strictly reduce the triangulation cost.

First, we consider the case of flip sequences of exactly length one. When the MLOP terminates, it is guaranteed that no permissible flip sequence can strictly reduce the triangulation cost. Since $\text{permFlipSeqs}_T(e_0)$ is required to satisfy (3.2), the set of permissible flip sequences always contains all valid length-1 flip sequences for the triangulation. Thus, no single edge flip could strictly reduce the triangulation cost.

Next, we consider the case of length-2 flip sequences (e_0, e_1). There are three cases to consider:

1. e_0 and e_1 are both edges in the triangulation T to which (e_0, e_1) is being applied; and $d_T(e_0, e_1) > \text{inflDist}(c)$;
2. e_0 and e_1 are both edges in the triangulation T to which (e_0, e_1) is being applied; and $d_T(e_0, e_1) \leq \text{inflDist}(c)$;

3. e_1 is the new edge obtained by flipping e_0.

Trivially, case 3 cannot reduce the cost since flipping e_0 and then e_1 yields the original triangulation T (i.e., the two edge flips cancel). Now, we consider case 1. Since $d_T(e_0, e_1) > \text{inflDist}(c)$, the edges e_0 and e_1 can be considered independently as two length-1 flip sequences. Since the triangulation is 1-flip optimal, any such length-1 flip sequence cannot strictly reduce the triangulation cost. Lastly, we consider case 2. In this case, the edges e_0 and e_1 are sufficiently close together that the optimality of one can depend on the other. Consequently, the MLOP must ensure that both edges are considered jointly as a single length-2 flip sequence. This is guaranteed by the MLOP if the set $\text{permFlipSeqs}_T(e_0)$ includes all length-2 flip sequences (e_0, e_1) for which $d_T(e_0, e_1) \leq \text{inflDist}(c)$.

Now, we consider the necessity of the condition stated in the theorem. The necessity follows from the fact that, if the stated condition is violated, at least one length-1 or length-2 flip sequence must exist for which it is not known whether the triangulation cost would be reduced if that flip sequence were applied to the triangulation. For example, if (3.2) is violated, there must exist at least one flippable edge in the triangulation for which it is not known whether flipping that edge would reduce the triangulation cost. □

Since, as explained earlier, the MLOP includes the LOP and LLOP as special cases, the above theorem also covers the LOP and LLOP. Consequently, from this theorem, we can infer the following result regarding if and when each of the MLOP, LOP, and LLOP approaches are guaranteed to yield 2-flip optimal triangulations:

Corollary 1.1 (2-flip optimality properties of the LOP, LLOP, and MLOP)**.** *The LOP cannot be guaranteed to produce a 2-flip optimal triangulation. The LLOP can only be guaranteed to produce a 2-flip optimal triangulation in the case that the triangulation cost function c satisfies* $\text{inflDist}(c) \leq 1$. *Of the cost functions considered herein only the SE cost function satisfies this condition. Lastly, the MLOP is guaranteed to produce a 2-flip optimal triangulation if the permissible flip sequence policy is chosen as either* $PFS_{\mathsf{MLT}}(L)$ *or* $PFS_{\mathsf{IOS}}(L)$ *and* $L \geq \text{inflDist}(c)$. *If the permissible flip sequence policy is chosen as* $PFS_{\mathsf{IO}}(L)$*, the MLOP is only guaranteed to produce a 2-flip optimal triangulation if* $L \geq 1$ *and* $\text{inflDist}(c) \leq 1$.

Proof. First, we consider the LOP. The proof of the statement for the LOP follows immediately from (3.3). Since, in the case of the LOP, all permissible flip sequences

are of length 1 (i.e., no length-2 flip sequences are considered), the condition that guarantees 2-flip optimality cannot be met.

Next, we consider the LLOP. The LLOP only considers the length-2 flip sequences in Γ in (3.4). This set only includes all length-2 flip sequences (e_0, e_1), where $d(e_0, e_1) \leq 1$. Thus, the LLOP can only be guaranteed to produce 2-flip-optimal triangulations if $\text{inflDist}(c) \leq 1$.

Finally, we consider the MLOP. The proof of the statement for the MLOP follows immediately from the definition of the permissible flip-sequence policies $\text{PFS}_{\mathsf{MLT}}(L)$, $\text{PFS}_{\mathsf{IOS}}(L)$, and $\text{PFS}_{\mathsf{IO}}(L)$. For the $\text{PFS}_{\mathsf{MLT}}(L)$, $\text{PFS}_{\mathsf{IOS}}(L)$ policies, as long as $L \geq \text{inflDist}(c)$, the set $\text{permFlipSeqs}_T(e_0)$ will always include all of the flip sequences necessary to ensure 2-flip optimality as specified in Theorem 1. For the $\text{PFS}_{\mathsf{IO}}(L)$ policy, the set $\text{permFlipSeqs}_T(e_0)$ can only include all of the flip sequences necessary for 2-flip optimality if $\text{inflDist}(c) \leq 1$ and $L \geq 1$. □

From Corollary 1.1, it immediately follows that as long as L is chosen sufficiently large, each of the $\text{MLOP}_{\mathsf{A}}(L)$, $\text{MLOP}_{\mathsf{B}}(L, M)$, and $\text{MLOP}_{\mathsf{C}}(L)$ methods is guaranteed to yield triangulations that are 2-flip optimal. For all of the cost functions considered herein, $L = 2$ is sufficiently large to ensure this. Consequently, we focus primarily on the specific variants $\text{MLOP}_{\mathsf{A}}(2)$, $\text{MLOP}_{\mathsf{B}}(2, 2)$, and $\text{MLOP}_{\mathsf{C}}(2)$ in the remainder of this thesis.

From above, we can see that our three proposed MLOP-based methods (i.e., $\text{MLOP}_{\mathsf{A}}(L)$, $\text{MLOP}_{\mathsf{B}}(L, M)$, and $\text{MLOP}_{\mathsf{C}}(L)$) are superior to the LOP and LLOP in terms of the n-flip optimality criterion. Thus, all other things being equal, we would expect the MLOP to yield lower cost triangulations (on average) than the LOP and LLOP. As we shall see later, this suspicion is confirmed by experimental results.

3.5 Evaluation of Proposed LOP

Having introduced our proposed MLOP-based methods (i.e., MLOP_{A}, MLOP_{B}, and MLOP_{C}), we now evaluate their performance by comparing them to the LOP and LLOP schemes for the task of solving Problem 1 (i.e., the triangulation-connectivity optimization problem). In this comparison, the seven cost functions introduced earlier are considered. The specific variants of our proposed MLOP-based methods used in this evaluation are $\text{MLOP}_{\mathsf{A}}(L)$, $\text{MLOP}_{\mathsf{B}}(L, M)$, and $\text{MLOP}_{\mathsf{C}}(L)$ with the L and M

parameters chosen as $L = M = 2$. This choice of L and M was made as it is sufficient to ensure that each of our methods will yield 2-flip optimal triangulations for all of the cost functions under consideration. All of the experimental results presented herein for the LOP, LLOP, and our proposed MLOP-based schemes were obtained from a software implementation written in C++ that was developed by the author of this thesis. Although some effort was made to ensure that this implementation was reasonably efficient, it could certainly be optimized further (e.g., some extra work performed by the code to facilitate more thorough testing and a more general computational framework than what is considered herein could be eliminated). So, any execution times given herein (especially ones for our proposed methods) should be viewed as upper bounds on what could be achieved with a reasonably efficient implementation.

For evaluation purposes, we employed 29 meshes for test data, which were produced using several highly-effective mesh generators, including the error-diffusion [43], ID1 [9], and GPRFS-ED [8] methods. The meshes were generated from a variety of grid-sampled functions (e.g., images and elevation maps) taken mostly from standard data sets such as [40, 23, 39] as well some test functions in [19]. The sizes of these meshes vary from approximately 900 to 16000 vertices.

For each of our 29 test meshes and each of the seven cost functions considered herein (for a total of $29 \cdot 7 = 203$ test cases), each of the five methods under consideration was used to optimize the mesh's triangulation connectivity and the resulting triangulation cost was measured. For each of the 203 test cases, the triangulation costs resulting from the five methods were ranked from 1 to 5, with a rank of 1 corresponding to the best (i.e., lowest) triangulation cost and a rank of 5 corresponding to the worst. Then, the average and standard deviation of the ranks for each method were computed for each cost function as well as overall, with the results shown in Table 3.2(b). (The standard deviations are the numbers shown in parentheses in the table.) Individual results obtained for a representative subset of the test cases are given in Table 3.2(a). In particular, for each of the seven cost functions, results are given for three meshes. Lastly, the median reduction in the triangulation cost relative to the LOP is given in Table 3.3 for each of the LLOP, MLOP_{A}, MLOP_{B}, and MLOP_{C} methods. To assist with the interpretation of the data, in each of Tables 3.2(a), 3.2(b), and 3.3, the best result for each table entry is highlighted in bold font.

Proposed methods vs. LOP. First, we compare our proposed MLOP-based methods

Table 3.2: Triangulation costs obtained using the various optimization methods. (a) Results for a representative subset of the individual test cases. (b) Average rankings taken across all test cases.

(a)

Cost Func.	Mesh	Triangulation Cost				
		LOP	LLOP	$\text{MLOP}_{\mathsf{A}}(2)$	$\text{MLOP}_{\mathsf{B}}(2,2)$	$\text{MLOP}_{\mathsf{C}}(2)$
ABN	lena@ED@2	11916.60	10271.55	10089.16	9539.82	**9417.04**
ABN	ct@ID@2	8185.67	6957.90	6858.16	6466.56	**6389.13**
ABN	n36-w113@0.5	7349.00	7002.41	6972.81	6905.48	**6897.76**
AMC	lena@ED@2	106401.89	101248.28	100792.96	99810.94	**99481.09**
AMC	ct@ID@2	71694.69	68165.35	67972.92	67091.76	**66883.94**
AMC	n36-w113@0.5	6918326.24	6868876.57	6857626.68	6853149.95	**6851738.72**
DLP	lena@ED@2	973861.70	749538.76	751934.44	**693624.80**	693765.19
DLP	ct@ID@2	9241106.72	7316894.76	6996901.67	6491381.83	**5819313.77**
DLP	n36-w113@0.5	7673511.72	7065988.61	7008287.94	6890063.14	**6838935.10**
DP	lena@ED@2	99480.75	77684.29	78242.12	**67863.62**	70426.29
DP	ct@ID@2	125744.00	95138.91	88144.27	77252.90	**74717.28**
DP	n36-w113@0.5	5909169.34	5312215.54	5262803.93	5122322.64	**5082415.04**
JND	lena@ED@2	203874.69	190001.90	188589.17	187041.87	**186462.70**
JND	ct@ID@2	1641991.36	1507048.46	1506550.02	**1485052.35**	1495526.76
JND	n36-w113@0.5	9831.93	9519.02	9479.73	9431.13	**9422.08**
SE	lena@ED@2	22684438	19285947	19134957	**18297542**	18333058
SE	ct@ID@2	82791830	78502285	78196476	77696882	**77625618**
SE	n36-w113@0.5	899889559	874421792	873570760	**869878114**	869931291
YMS	lena@ED@2	3056206.81	2792159.74	2781951.65	2727975.11	**2724628.16**
YMS	ct@ID@2	159766983.95	54517395.48	54128494.76	56343039.01	**51937420.49**
YMS	n36-w113@0.5	2049.38	1791.59	1748.12	1685.08	**1680.91**

(b)

Cost Func.	Average Rank (with Standard Deviation†)				
	LOP	LLOP	$\text{MLOP}_{\mathsf{A}}(2)$	$\text{MLOP}_{\mathsf{B}}(2,2)$	$\text{MLOP}_{\mathsf{C}}(2)$
ABN	5.00 (0.00)	3.86 (0.34)	3.14 (0.34)	1.79 (0.41)	**1.21** (0.41)
AMC	5.00 (0.00)	3.97 (0.18)	3.03 (0.18)	1.72 (0.45)	**1.28** (0.45)
DLP	5.00 (0.00)	3.93 (0.25)	3.07 (0.25)	1.62 (0.49)	**1.38** (0.49)
DP	5.00 (0.00)	3.76 (0.43)	3.21 (0.48)	1.62 (0.55)	**1.41** (0.49)
JND	5.00 (0.00)	3.97 (0.18)	3.03 (0.18)	1.83 (0.38)	**1.17** (0.38)
SE	5.00 (0.00)	3.72 (0.45)	3.24 (0.50)	1.55 (0.50)	**1.48** (0.56)
YMS	5.00 (0.00)	3.45 (0.72)	3.07 (0.83)	2.14 (0.90)	**1.34** (0.54)
Overall	5.00 (0.00)	3.81 (0.44)	3.11 (0.46)	1.75 (0.58)	**1.33** (0.49)

†The standard deviation is given in parentheses.

to the LOP. From the overall statistical results in Table 3.2(b), each of our $\text{MLOP}_{\mathsf{A}}(2)$, $\text{MLOP}_{\mathsf{B}}(2,2)$, and $\text{MLOP}_{\mathsf{C}}(2)$ methods outperforms the LOP in all 203 test cases. This can be seen from the fact that the LOP consistently has an overall rank of last (i.e., 5th) place with a standard deviation of exactly 0 (i.e., for every test case, the

Table 3.3: Comparison of reduction in triangulation cost obtained with various methods relative to the LOP.

Cost Func.	Median Reduction in Triangulation Cost Relative to LOP (%)			
	LLOP	$\text{MLOP}_\text{A}(2)$	$\text{MLOP}_\text{B}(2,2)$	$\text{MLOP}_\text{C}(2)$
ABN	13.80	14.76	19.72	**19.77**
AMC	4.30	4.64	5.85	**5.96**
DLP	13.53	14.57	**18.21**	17.89
DP	20.70	21.35	**28.34**	28.25
JND	6.14	6.63	7.42	**7.59**
SE	5.63	5.72	**7.11**	7.07
YMS	22.08	24.10	27.58	**30.50**
Overall	11.49	12.18	16.36	**16.62**

LOP has a rank equal to its average rank of 5). The poor performance of the LOP is also evident from the individual results shown in Table 3.2(a). Examining Table 3.3, we can see that, depending on the cost function used, our MLOP_A, MLOP_B, and MLOP_C methods yield a median cost reduction of 5.72% to 24.10%, 5.85% to 28.34%, and 5.96% to 30.50%, respectively, relative to the LOP. Note that these values are medians. So, in many cases, the actual improvement is greater than these median values, and sometimes much greater (e.g., as high as about 74%). So, clearly, our proposed MLOP-based methods offer very substantially better results than the LOP.

Proposed methods vs. LLOP. Next, we compare our proposed MLOP-based methods to the LLOP. From the overall statistical results given in Table 3.2(b), we can see that each of our MLOP_A, MLOP_B, and MLOP_C methods achieves a better average rank than the LLOP. This is consistent with individual results shown in Table 3.2(a). As it turns out, a more detailed analysis of the data shows that our MLOP_B and MLOP_C methods always outperform the LLOP in every test case for all of the cost functions, except YMS. In the case of the YMS cost function, the MLOP_B and MLOP_C methods outperform the LLOP in 24/29 (83%) and 27/29 (93%) of the test cases, respectively. Again, a more detailed analysis of the data shows that, except for the DP, SE, and YMS, cost functions, MLOP_A outperforms the LLOP in 86% to 96% of the test cases, while in the case of the DP, SE, and YMS cost functions, MLOP_A outperforms the LLOP in 69% to 76% of the test cases. Examining Table 3.3, we can see that, depending on the cost function, our MLOP_A, MLOP_B, MLOP_C methods offer a median cost reduction that is greater than that of the LLOP by 0.09% to 2.02%, 1.28% to 7.64%, and 1.44% to 8.42%, respectively. Again, note that these values

are medians. So, in many cases, our proposed methods (especially MLOP_B and MLOP_C) beat the LLOP by a larger margin than these values. In particular, the MLOP_A, MLOP_B, and MLOP_C methods beat the LLOP by up to 14.35%, 19.06%, and 19.57%, respectively, in our 203 test cases. Clearly, our proposed MLOP-based methods offer a considerable benefit over the LLOP.

Proposed methods relative to each other. Next, we compare our proposed MLOP methods (i.e., MLOP_A, MLOP_B, and MLOP_C) to each other in terms of performance. From Table 3.2(b), we can see that, in terms of overall average rank, the MLOP_C method performs best followed by the MLOP_B and MLOP_A schemes in that order. Furthermore, this relative ordering can be seen to be consistent across each of the seven cost functions. For the most part, the standard deviations are sufficiently small that the average ranks alone give an accurate picture of the results. The only exception to this is in the case of the MLOP_B and MLOP_C methods for the DLP, DP, and SE cost functions. In the case of these three cost functions, although the MLOP_C method has a better average rank than the MLOP_B scheme, the MLOP_B and MLOP_C methods are actually fairly close in terms of performance. As it turns out, in the case of these three cost functions, although the MLOP_C method beats the MLOP_B scheme in more cases, the MLOP_B scheme has a tendency to perform better in cases where the improvement relative to LOP is greater. This leads to the fact that, for these three cost functions, the MLOP_B is able to achieve a higher median cost reduction (relative to the LOP) than the MLOP_C method. This can be seen by examining the results of Table 3.3, where the MLOP_B method has higher values for the DLP, DP, and SE cost functions than the MLOP_C scheme. A more detailed analysis of the data shows that, in the cases of these three cost functions, the MLOP_C method beats the MLOP_B scheme in a much smaller fraction of the test cases, compared to the other cost functions. In particular, for the DLP, DP, and SE cost functions, the MLOP_C method beats the MLOP_B scheme in only 62%, 58%, and 55% of the test cases, respectively. In the case of the other cost functions, this percentage is significantly higher, namely, above 72% in all cases. The individual results given in Table 3.2(a) can be seen to be consistent with the overall statistical results, with the MLOP_C method most frequently yielding the best result followed by the MLOP_B and MLOP_A schemes in that order. Clearly, the MLOP_C and MLOP_B methods perform better than the MLOP_A scheme. Again, the MLOP_A scheme was only introduced herein as an additional point of comparison (due to its relative conceptual simplicity). Thus, we conclude that the MLOP_C method is best, except for the cases of the DLP, DP,

and SE cost functions, where the MLOP_B scheme yields better results.

Comments on computation time. At this point, a few comments are worthwhile regarding the computation times of the various methods under consideration. On very modest hardware, namely a six-year-old notebook with a 1.90 GHz Intel Core i7 CPU and 6 GB RAM, the time required for the LOP, LLOP, MLOP_A, MLOP_B, and MLOP_C for each of the test cases in Table 3.2(a) lies in the ranges 0.46 to 1.14 s, 1.55 to 4.50 s, 4.72 to 20.74 s, 4.15 to 14.01 s, and 11.39 to 38.19 s, respectively. In this regard, the first observation that we can make is that all of the methods are reasonably fast, with all of them requiring less than 40 seconds of computation time. Generally, the LOP is fastest, followed by the LLOP, and then the MLOP_B, MLOP_A, and MLOP_C schemes in that order. Although the LOP and LLOP are usually faster than our proposed MLOP-based methods, the better results produced by our methods can easily justify this extra computational cost (which is not too exorbitant) for many applications. Interestingly, in the vast majority of test cases, the MLOP_B method is faster than the MLOP_A scheme. Although the second stage of processing in the MLOP_B method uses the same parameters as the single stage of processing in the MLOP_A scheme, the second stage converges very quickly due to the preconditioning achieved by the first stage. This, combined with fast convergence of the first stage of processing, allows the MLOP_B method to require less time than the MLOP_A scheme.

Recommendations. Considering all of the above results, we recommend the use of $\text{MLOP}_\text{B}(2, 2)$ for cost functions other than ABN, AMC, JND, and YMS as well as in situations where the lesser computation time for MLOP_B (relative to MLOP_C) is desired. Otherwise, we recommend the use of $\text{MLOP}_\text{C}(2)$.

Other remarks. Of the cost functions considered herein excluding SE, the $\text{MLOP}_\text{A}(2)$ scheme performs the minimal amount of extra work beyond that done by the LLOP to ensure 2-flip optimality. This makes the MLOP_A scheme an interesting one to use for comparison purposes (hence, our reason for considering the MLOP_A scheme herein). The fact that $\text{MLOP}_\text{A}(2)$ is able to outperform the LLOP in the preceding cases demonstrates that 2-flip optimality by itself is quite beneficial to have. Moreover, the fact that the MLOP_B and MLOP_C methods beat the LLOP by even larger margins (than the MLOP_A scheme) shows that the more sophisticated permissible-flip-sequence policies they employ are also highly effective.

In passing, it is worth mentioning that, at expense of increased computational cost, even better results can be obtained with our MLOP-based methods. In particular, one can use $\text{MLOP}_\text{B}(2, i)$ and $\text{MLOP}_\text{C}(i)$ for values of i greater than 2. Table 3.4 shows

Table 3.4: Triangulation costs obtained using $\text{MLOP}_{\mathsf{B}}(2, M)$ for various choices of M

Cost Func.	Mesh	Triangulation Cost			
		$M = 2$	$M = 3$	$M = 4$	$M = 5$
ABN	lena@ED@2	9539.82	9235.90	8950.96	**8881.31**
ABN	ct@ID@2	6466.56	6181.52	6073.77	**5969.44**
ABN	n36-w113@0.5	6905.48	6877.19	6862.76	**6840.55**
AMC	lena@ED@2	99810.94	98830.81	98579.41	**98540.33**
AMC	ct@ID@2	67091.76	66734.55	66521.99	**66348.92**
AMC	n36-w113@0.5	6853149.95	6852002.26	6851130.46	**6848777.86**
DLP	lena@ED@2	693624.80	689832.86	678211.34	**631582.68**
DLP	ct@ID@2	6491381.83	6171134.41	6119682.15	**5713933.08**
DLP	n36-w113@0.5	6890063.14	6817818.32	**6718116.16**	6721997.02
DP	lena@ED@2	67863.62	66484.86	64318.27	**62851.59**
DP	ct@ID@2	77252.90	73002.61	**72021.63**	72374.42
DP	n36-w113@0.5	5122322.64	5001998.30	4962355.40	**4933473.46**
JND	lena@ED@2	187041.87	185816.54	185728.60	**185482.90**
JND	ct@ID@2	1485052.35	1479519.63	1476350.99	**1475829.41**
JND	n36-w113@0.5	9431.13	9408.14	9392.62	**9376.06**
SE	lena@ED@2	18297542	17832396	17735184	**17431940**
SE	ct@ID@2	77696882	77301345	77196481	**77179921**
SE	n36-w113@0.5	869878114	868600878	867165140	**866664396**
YMS	lena@ED@2	2727975.11	2691297.40	**2657434.01**	2661270.97
YMS	ct@ID@2	56343039.01	53622523.01	**46175699.34**	53861453.89
YMS	n36-w113@0.5	1685.08	1651.44	1634.73	**1623.68**

the triangulation costs obtained with $\text{MLOP}_{\mathsf{B}}(2, i)$ for the test cases from Table 3.2(a) with i ranging from 2 to 5. Similar results are also given for $\text{MLOP}_{\mathsf{C}}(i)$ in Table 3.5. Examining the results of both of these tables, we can see that, for both $\text{MLOP}_{\mathsf{B}}(2, i)$ and $\text{MLOP}_{\mathsf{C}}(i)$, the trend is for the triangulation cost to decrease as i increases. This demonstrates that even better results can be obtained using our MLOP-based approaches if one is willing to incur the expense of more computation.

Computation time vs. mesh size. During the evaluation of the $\text{MLOP}_{\mathsf{B}}(2, 2)$ and $\text{MLOP}_{\mathsf{C}}(2)$ methods, we also examined how the size of mesh affects computation time. In this experiment, for a number of images, we generated meshes with Delaunay connectivity ranging in size from 500 to 10000 vertices. Then, we measured the time required to optimize the mesh connectivity using $\text{MLOP}_{\mathsf{B}}(2, 2)$ and $\text{MLOP}_{\mathsf{C}}(2)$ with the various cost functions. Graphs showing a representative subset of the results are given in Figure 3.1. The graphs in Figures 3.1(a), (b) and (c) show the computation time plotted against mesh size for $\text{MLOP}_{\mathsf{B}}(2, 2)$ with cost function SE,

Table 3.5: Triangulation costs obtained using $\mathrm{MLOP}_{\mathsf{C}}(L)$ for various choices of L

Cost Func.	Mesh	Triangulation Cost			
		$L = 2$	$L = 3$	$L = 4$	$L = 5$
ABN	lena@ED@2	9417.04	9190.62	8946.22	**8781.89**
ABN	ct@ID@2	6389.13	6165.62	5999.04	**5962.38**
ABN	n36-w113@0.5	6897.76	6831.35	6814.38	**6783.87**
AMC	lena@ED@2	99481.09	98694.11	98497.80	**98420.13**
AMC	ct@ID@2	66883.94	66562.35	66378.11	**66298.59**
AMC	n36-w113@0.5	6851738.72	6848643.96	6846949.52	**6846406.06**
DLP	lena@ED@2	693765.19	677303.21	665675.13	**664895.44**
DLP	ct@ID@2	5819313.77	6449558.64	5312767.51	**4998525.22**
DLP	n36-w113@0.5	6838935.10	6755761.23	6676171.30	**6630737.61**
DP	lena@ED@2	70426.29	65600.47	62485.53	**61098.12**
DP	ct@ID@2	74717.28	72560.62	**63390.78**	68631.84
DP	n36-w113@0.5	5082415.04	4953480.91	4918212.10	**4842517.34**
JND	lena@ED@2	186462.69	185334.12	184809.10	**184459.31**
JND	ct@ID@2	1495526.76	1475114.75	1472661.70	**1471682.01**
JND	n36-w113@0.5	9422.07	9375.31	9362.66	**9346.87**
SE	lena@ED@2	18333058	17895627	17648332	**17337996**
SE	ct@ID@2	77625618	77211125	77096945	**77007871**
SE	n36-w113@0.5	869931291	867594566	866558588	**866244494**
YMS	lena@ED@2	2724628.16	2666925.87	2657377.21	**2650449.03**
YMS	ct@ID@2	51937420.49	50633138.21	48262464.43	**46035990.21**
YMS	n36-w113@0.5	1680.91	1621.14	1586.29	**1566.21**

ABN and JND, respectively. Similar graphs are also given in Figures 3.1(d), (e) and (f) for $\mathrm{MLOP}_{\mathsf{C}}(2)$. From Figure 3.1, it is clear that the computation time grows approximately linearly as the size of mesh increases for all six cases. We also found a similar growth rate for computation time for the other cost functions considered in our work. This demonstrates that the computation times of $\mathrm{MLOP}_{\mathsf{B}}(2,\ 2)$ and $\mathrm{MLOP}_{\mathsf{C}}(2)$ grow approximately linearly with mesh size (at least when starting from meshes with Delaunay connectivity).

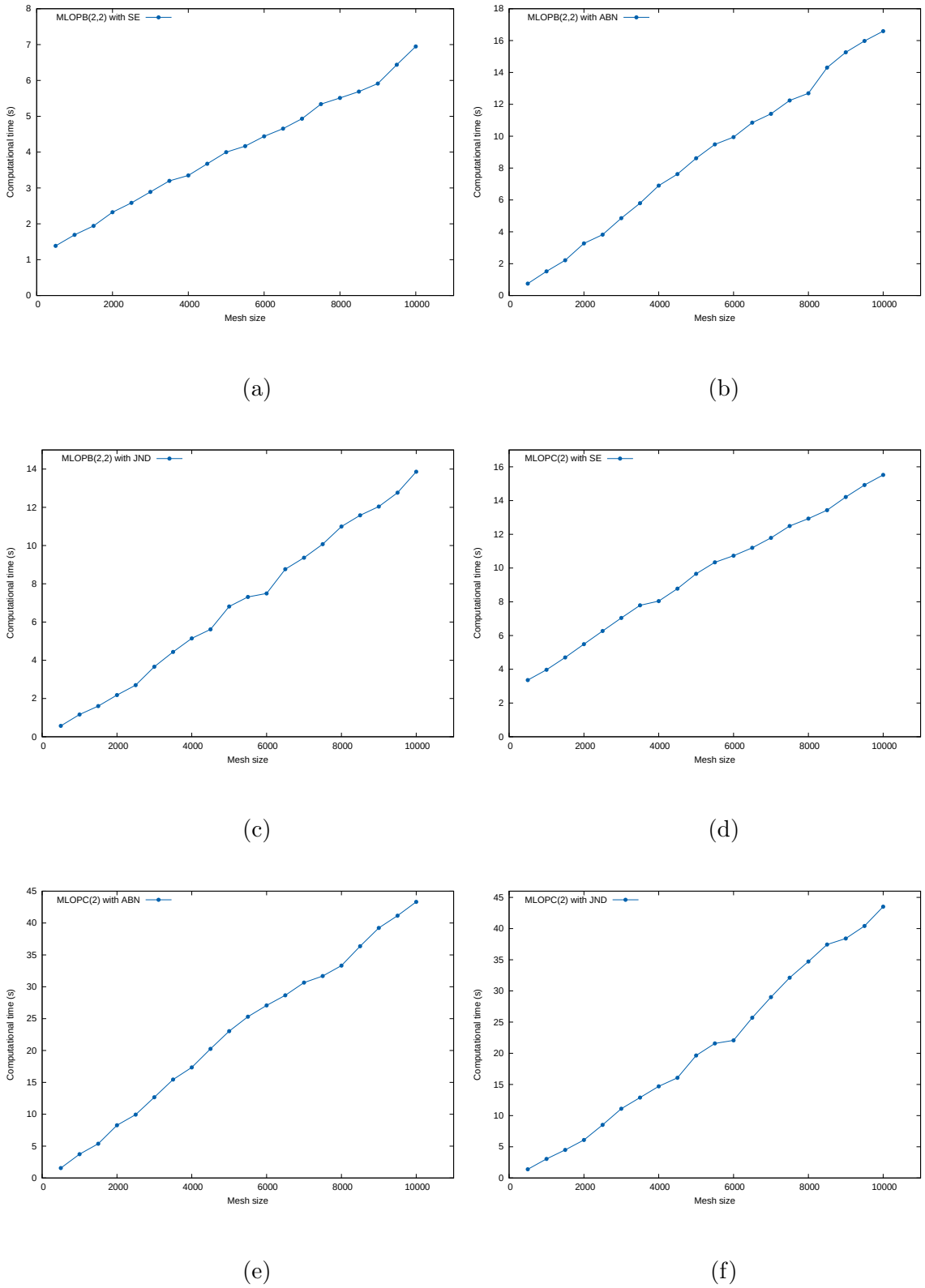


Figure 3.1: Computation time versus mesh size for (a) $\mathrm{MLOP_B}(2,\ 2)$ with SE, (b) $\mathrm{MLOP_B}(2,\ 2)$ with ABN, (c) $\mathrm{MLOP_B}(2,\ 2)$ with JND, (d) $\mathrm{MLOP_C}(2)$ with SE, (e) $\mathrm{MLOP_C}(2)$ with ABN, and (f) $\mathrm{MLOP_C}(2)$ with JND.

Chapter 4

Conclusions and Future Research

4.1 Conclusions

In this thesis, we have considered the problem of selecting the connectivity of a triangulation to minimize a given cost function. The notion of n-flip optimality was introduced and then used to guide the development of an improved solution technique for this problem. In particular, a computational framework for an improved version of the LOP, called the MLOP, was proposed. The MLOP framework has a number of degrees of freedom (e.g., the choice of permissible-flip-sequence policy), which were explored, leading to the proposal of two highly-effective MLOP-based methods known as $\text{MLOP}_{\mathsf{B}}(L, M)$ and $\text{MLOP}_{\mathsf{C}}(L)$. We showed that each of the $\text{MLOP}_{\mathsf{B}}(L, M)$ and $\text{MLOP}_{\mathsf{C}}(L)$ methods is guaranteed to yield 2-flip optimal triangulations for arbitrary cost functions if L is chosen sufficiently large. Furthermore, we also showed that, for most cost functions, the LOP and LLOP cannot make such a guarantee.

The triangulation-connectivity selection problem was considered in detail for the case of seven well-known cost functions. In this context, we focused our attention on the specific variants of our MLOP-based methods $\text{MLOP}_{\mathsf{B}}(L, M)$ and $\text{MLOP}_{\mathsf{C}}(L)$ obtained when $L = M = 2$ (i.e., $\text{MLOP}_{\mathsf{B}}(2, 2)$ and $\text{MLOP}_{\mathsf{C}}(2)$). Unlike the LOP and LLOP, our $\text{MLOP}_{\mathsf{B}}(2, 2)$ and $\text{MLOP}_{\mathsf{C}}(2)$ schemes are guaranteed to yield 2-flip optimal triangulations for all seven of the cost functions under consideration. Through experimental results, we demonstrated that our $\text{MLOP}_{\mathsf{B}}(2, 2)$ and $\text{MLOP}_{\mathsf{C}}(2)$ methods are able to produce triangulations of significantly lower cost than the LOP and LLOP schemes, while still maintaining a reasonable computational cost (e.g., from a few seconds to a few tens of seconds for problems of practical interest). Of our

MLOP-based methods, we found $\mathrm{MLOP}_{\mathsf{B}}(2,2)$ to be the best choice if the cost function is DP, DLP, or SE, or if computational cost is more of a concern; otherwise, $\mathrm{MLOP}_{\mathsf{C}}(2)$ is recommended. Lastly, we demonstrated that by using $\mathrm{MLOP}_{\mathsf{B}}(2,i)$ and $\mathrm{MLOP}_{\mathsf{C}}(i)$ with $i \geq 2$, one can achieve even better results than with $i = 2$ at the expense of increased computational cost.

Since the triangulation-connectivity selection problem can arise in numerous contexts, effective solution techniques for this problem are of great practical interest. In particular, methods for the generation of meshes based on data-dependent triangulations can benefit substantially from improvements in such solution techniques. Thus, the MLOP framework and MLOP-based schemes proposed herein have great potential as a tool for developing improved mesh-generation and other methods.

4.2 Future Research

Although this thesis has made significant contributions to the solution of the triangulation-connectivity selection problem, further work in this area could still be done. In what follows, some potential areas for future work are discussed.

As we introduced in Corollary 1.1, the MLOP is guaranteed to produce a 2-flip optimal triangulation if the permissible flip sequence policy is chosen as either $\mathrm{PFS}_{\mathsf{MLT}}(L)$ or $\mathrm{PFS}_{\mathsf{IOS}}(L)$ and $L \geq \mathrm{inflDist}(c)$. Since any given n-flip optimal triangulation will tend to more closely approach the globally optimal solution in terms of cost as n increases, the MLOP could be further improved by applying some other permissible flip sequence policy to yield n-flip optimality triangulation, where $n > 2$. We suspect that such a permissible flip sequence policy can be achieved by carefully selecting the free parameter in Algorithm 2 with respect to the influence distance of the triangulation cost function. Proving the n-flip optimality, where $n > 2$, however, is more challenging. A permissible flip sequence policy yields n-flip optimal triangulation, where $n > 2$, would be potentially valuable if the computational cost is not too high.

Another potential improvement is the computational cost of the MLOP. During the optimization procedure, we noticed applying different flip sequences might result in the triangulations with the same connectivity. Therefore, the algorithm could compute the cost of a triangulation, whose cost has been computed before. If some efficient scheme is employed to avoid these redundant computations, the computational efficiency could be improved.

Appendix A

Software User Manual

A.1 Introduction

As a part of this research project, software that implements the proposed triangulation-connectivity optimization methods was developed by the author with the guidance from his supervisor. The software also implements a mesh generation method like the one in [27] based on the MLOP. The software was written in C++ under the Linux OS, and consists of more than 6000 lines of code.

Our software project contains three executable programs as follows.

1. `optimize_mesh` performs the triangulation-connectivity optimization for a triangulation based on the MLOP.

2. `generate_mesh` produces a triangle mesh model for the input image based on the MLOP.

3. `two_flip_optimal` performs the explicit 2-flip optimality test on a triangulation for a triangulation cost function specified by the user.

In the remainder of this appendix, we will introduce how to install and use the above software in detail. Several examples are also provided for illustrating software usage.

A.2 Installing the Software

Since our program utilize many features of `C++11/14`, the compiler should be compatible with `C++11/14`. We recommend to use the `GCC 6.1.0` or higher version as

compiler. Our software makes heavy use of some C++ libraries, including the Computational Geometry Algorithm Library (CGAL) [3], Boost Library [1], Signal Processing Library (SPL) [5], and SPL Extensions Library (SPLEL). These libraries should be correctly installed before building the software. The versions of these libraries that have been verified to work correctly with our programs are:

- Boost 1.59.0
- CGAL 3.8.2
- SPL 2.0.4
- SPLEL 2.0.5

In order to install our software, user should first install the `CMake`[2] tool with version 3.2.2 or later. To build and install our software, the following steps are required (in order):

1. Choose an installation directory. Let `$INSTALL_DIR` denotes this directory.
2. Change the current working directory to the top level of the source tree (i.e., the directory that the contains the `CMakeLists.txt`)
3. Generate native build files by executing the command:
   `cmake -H. -Btmp -DCMAKE_INSTALL_PREFIX=$INSTALL_DIR`
4. To build and install the software, run the command:
   `cmake --build tmp --clean-first --target install`
   Note that this step may require special administrator privileges depending on the target directory for installation.

Once the software has been installed successfully, the directory `tmp` under the current working directory can be deleted by executing the command `rm -r tmp`.

A.3 Detailed Program Descriptions

As mentioned earlier, the software consists of three programs: `optimize_mesh`, `generate_mesh`, and `two_flip_optimal`. In what follows, we provide a detailed description for each of these programs.

A.3.1 The `optimize_mesh` Program

SYNOPSIS

`optimize_mesh [OPTIONS]`

DESCRIPTION

The `optimize_mesh` program reads a triangle mesh in OFF format [4] from standard input, and optimizes the triangulation connectivity of this mesh based on the method specified by the user. Then, the program writes the optimized mesh model to standard output in OFF format.

OPTIONS

The following options are supported by the `optimize_mesh` program:

`-b $selPolicy`	Set the good flip-sequence selection policy to `$selPolicy`. This option must be provided. The valid values for `$selPolicy` are listed in Table A.3.
`-C $maxFlips`	Set the maximum flip count of an edge to `$maxFlips`. When an edge has been flipped more than `$maxFlips` times, a cycle is assumed to have been encountered.
`-c $triCostFun`	Set the triangulation-cost function to `$triCostFun`. This option must be provided. The valid values for `$triCostFun` are listed in Table A.2.
`-f $originalFunc`	Set `$originalFunc` as the original function ϕ being modeled. This option is only valid when the triangulation-cost function is SE.

`-H`	`$historyLevel`	Set the level of history output to `$historyLevel`. If `$historyLevel` is not specified, no history data will be output. The valid values for `$historyLevel` are listed in Table A.4.
`-h`	`$historyFile`	Write the history output to the file `$historyFile`. The history output records all of the operations applied to the mesh in a particular format useful for testing.
`-l`	`$lookahead`	Set the number of lookahead levels to `$lookahead` (where `$lookahead` is nonegative integer). This option must be provided.
`-O`	`$cacheEnabled`	Set the enabling of the cost cache to `$cacheEnabled`. If `$cacheEnabled` is 0, the cache is disabled; otherwise, the cache is enabled. The default value is 0.
`-r`	`$resultFile`	Write some information about the results of mesh-optimization process to the file `$resultFile`. This information includes the triangulation cost function, the value of `maxLevel`, the good flip-sequence selection policy, the triangulation cost before and after the optimization procedure, and the mesh-optimization time in seconds.
`-s`	`$cycleAlert`	Set the response mode to `$cycleAlert` when a potential cycle is detected. The two available choices for `$cycleAlter` are listed in Table A.6. The default mode is `always_fail`.
`-t`	`$triFile`	Write the optimized triangulation of the image plane in OFF format to the file `$triFile`.

Table A.2: Choices of triangulation-cost function

Policy	Description	Reference
`abn`	angle between normals (ABN)	Section 2.8
`amc`	absolute mean curvature (AMC)	Section 2.8
`dlp`	deviations from linear polynomials (DLP)	Section 2.8
`dp`	distances from planes (DP)	Section 2.8
`jnd`	jump in normal derivatives (JND)	Section 2.8
`se`	squared error (SE)	Section 2.8
`yms`	Yu-Morse-Sederberg (YMS)	Section 2.8

Table A.3: Choices of good flip-sequence selection policies

Policy	Description
`ff`	first found: choose the first good flip sequence that was found
`lc`	least cost: choose the good flip sequence that results in a triangulation with the least cost
`rd`	random: randomly select one of the good flip sequences

Table A.4: Choices of history level

Number	Type of information output
0	Point insertion, edge optimality test result and edge flip operation.
1	Level 0 information plus cost invalidation information if option `-O` is specified and triangulation cost before and after applying each flip sequence.
2	Level 1 information plus cycle detection infomation and infomation of each flip sequence applied to the triangulation.

`-z $permFlipSeq` Set the permissible flip-sequence policy to `$permFlipSeq`. This option must be provided. The valid values for `$permFlipSeq` are listed in Table A.5.

Table A.5: Choices of permissible flip-Sequence policies

Policy	Description	Reference
`out_noskip`	outward without skip	Section 3.3.3
`inout_noskip`	inward and outward without skip	Section 3.3.3
`out_skip`	outward with skip	Section 3.3.3
`inout_skip`	inward and outward with skip	Section 3.3.3
`stop_at_length_two`	maximum length two	Section 3.3.3

Table A.6: Choices of response mode when a potential cycles is detected

Mode	Description
`always_fail`	Print the iteration number and the two end points of the edge that causes the cycle, and exit the program.
`never_fail`	Continue the program and skip the edge that causes the cycle in the reminder of the optimization procedure.

A.3.2 The `generate_mesh` Program

SYNOPSIS

```
generate_mesh [OPTIONS]
```

DESCRIPTION

The `generate_mesh` program reads an image in PNM format from the standard input stream, and generates a mesh with a specified sampling density with the specified mesh-optimization method. Then, the program writes the optimized mesh in OFF format to standard output.

OPTIONS

The following options are supported by the `optimize_mesh` program:

`-b $selPolicy` — Set the good flip-sequence selection policy to `$selPolicy`. This option must be provided. The valid values for `$selPolicy` are listed in Table A.3.

`-C`	`$maxFlips`	Set the maximum flip count of an edge to `$maxFlips`. When an edge has been flipped more than `$maxFlips` times, a cycle is assumed to have been encountered.
`-c`	`$triCostFun`	Set the triangulation-cost function to `$triCostFun`. This option must be provided. The valid values for `$triCostFun` are listed in Table A.2.
`-d`	`$sampDensity`	Set the sampling density for the mesh to be generated to `$sampDensity`. The value should be in [0,1]. Either this option or the `-n` option must be provided.
`-H`	`$historyLevel`	Set the level of history output to `$historyLevel`. If `$historyLevel` is not specified, no history data will be output. The valid values for `$historyLevel` are listed in Table A.4.
`-h`	`$historyFile`	Write the history output to the file `$historyFile`. The history output records all of the operations applied to the mesh in a particular format useful for testing.
`-l`	`$lookahead`	Set the number of lookahead levels to `$lookahead` (where `$lookahead` is nonegative integer). This option must be provided.
`-m`		Specify that any subsequent `-b`, `-c`, and `-l` options apply to main processing.
`-n`	`$numSamples`	Set the number of samples in the mesh to be generated to `$numSamples`. Either this option or the `-d` option must be provided.

`-O`	`$cacheEnabled`	Set the enabling of the cost cache to `$cacheEnabled`. If `$cacheEnabled` is 0, the cache is disabled; otherwise, the cache is enabled. The default value is 0.
`-p`		Specify that any subsequent `-b`, `-c`, and `-l` options apply to post processing.
`-r`	`$resultFile`	Write some information about the results of the mesh-generation process to the file `$resultFile`. This information includes the triangulation cost in terms of the main triangulation-cost policy, triangulation cost in terms of the postprocessing triangulation cost policy, the total mesh-generation time, the main part of mesh-generation time in seconds, and the postprocessing part of mesh-generation time in seconds.
`-s`	`$cycleAlert`	Set the response mode to `$cycleAlert` when a potential cycle is detected. The two available choices for `$cycleAlter` are listed in Table A.6. The default mode is `always_fail`.
`-t`	`$triFile`	Write the optimized triangulation of the image plane in OFF format to the file `$triFile`.
`-z`	`$permFlipSeq`	Set the permissible flip-sequence policy to `$permFlipSeq`. This option must be provided. The valid values for `$permFlipSeq` are listed in Table A.5.

A.3.3 The `two_flip_optimal` Program

SYNOPSIS

`two_flip_optimal [OPTIONS]`

DESCRIPTION

The `two_flip_optimal` program reads a triangualtion mesh in OFF format from the standard input stream, and tests if the triangulation is 2-flip optimal in terms of the specified cost function. If the triangulation is 2-flip optimal, the program outputs the the message "True" as the test result. Otherwise, the program outputs the message "False" and some additional details. These details include all of the length-two flip sequences that can reduce the triangulation cost and the triangulation cost before and after applying such a flip sequence.

OPTIONS

The following options are supported in `two_flip_optimal` program:

`-c`	`$triCostFun`	Set the triangulation-cost function to `$triCostFun`. This option must be provided. The valid value for `$triCostFun` are listed in Table A.2.
`-i`	`$originalFunc`	Set the `$originalFunc` as the original function ϕ being modeled. This option is only valid when the triangulation-cost function is SE.

A.4 Examples of Software Usage

We provide some examples in what follows to illustrate how to use the software with different options.

Example A

Suppose that we want to optimize a triangle mesh in the file `mesh.off` and write the optimized mesh to the file `optMesh.off`. During the optimization procedure, the cycle response mode is desired to be `never_fail` with the maximum flip count of an edge to be 200 while the cost cache is enabled. Once the optimization procedure finish, the result information is to be written in the file `results.txt` and the triangulation in image planes is to be written in the file `tri.off`. The MLOP is desired to meet the following requirements:

- set the triangulation-cost function to be `JND`;
- set the number of lookahead level to be 2;
- set the good flip-sequence selection policy to be first found(`ff`);
- set the permissible flip-sequence policy to be inward and outward without skip.

The above objective can be achieved by running the command `optimize_mesh` as follows:

```
optimize_mesh -c jnd -l 2 -b ff -O 1 -t tri.off -r results.txt \
-z inout_noskip -C 200 -s never_fail < mesh.off >optMesh.off
```

Example B

Suppose we want to generate a mesh from the image `lena.pnm` with sampling density of 1% and output the mesh to the standard output stream to the file `lenaMesh.off`. Besides the mesh, we also want to output the results information to the file `results.txt` and output the triangulation in image plane to the file `tri.off`. During the mesh-generation procedure, the good flip-sequence selection policy is to be `ff` and the cycle response mode is to be `always_fail` while the cost cache is enabled. The mesh-generation method is desired to use the parameters as follows:

- set the triangulation-cost function to be JND for the main processing;
- set the number of lookahead level to be 0 for the main processing;
- set the triangulation-cost function to be SE for the post processing;

- set the number of lookahead level to be 3 for the post processing;
- set the permissible flip-sequence policy to be inward and outward with skip.

The above task can be accomplished by invoking the command `generate_mesh` as follows:

```
generate_mesh -d 0.01 -m -c jnd -l 0 -b ff -p -c se -l 3 -b ff -O 1 \
-t tri.off -r results.txt -z inout_skip -s always_fail \
< lena.pnm >lenaMesh.off
```

Example C

Suppose that we want to test if a triangulation T is 2-flip optimal based on a specified triangulation-cost function. The triangulation-cost function is to be ABN and the file `mesh.off` represents the triangle mesh model generated by T. Such a test can be accomplished with the following command:

```
two_flip_optimal -c abn <mesh.off
```

If the triangulation-cost function is defined as SE, the user should also specify the original bivariate function ϕ, such as `oriImage.pnm`, from which the input mesh is generated. This test can be accomplished with the following command:

```
two_flip_optimal -c se -i oriImage.pnm <mesh.off
```

Once the program executes successfully, the program will output the test results and detailed information to the standard output.

Bibliography

[1] Boost C++ library. `http://www.boost.org`. Accessed: 2018-01-10.

[2] CMake tool. `https://camke.org`. Accessed: 2018-01-10.

[3] Computational geometry algorithms library. `http://www.cgal.org`. Accessed: 2018-01-10.

[4] Object file format OFF. `http://shape.cs.princeton.edu/benchmark/documentation/off_format.html`. Accessed: 2018-01-10.

[5] Signal processing library. `https://github.com/mdadams/SPL`. Accessed: 2018-04-24.

[6] M. D. Adams. An efficient progressive coding method for arbitrarily-sampled image data. *IEEE Signal Processing Letters*, 15:629–632, 2008.

[7] M. D. Adams. Progressive lossy-to-lossless coding of arbitrarily-sampled image data using the modified scattered data coding method. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1017–1020, Taipei, Taiwan, April 2009.

[8] M. D. Adams. A flexible content-adaptive mesh-generation strategy for image representation. *IEEE Trans. on Image Processing*, 20(9):2414–2427, September 2011.

[9] M. D. Adams. A highly-effective incremental/decremental Delaunay mesh-generation strategy for image representation. *Signal Processing*, 93(4):749–764, April 2013.

[10] L. Alboul, G. Kloosterman, C. Traas, and R van Damme. Best data-dependent triangulations. *Journal of Computational and Applied Mathematics*, 119:1–12, 2000.

[11] J. G. Brankov, Y. Yang, and N. P. Galatsanos. Image restoration using content-adaptive mesh modeling. In *Proc. of IEEE International Conference on Image Processing*, volume 2, pages 997–1000, 2003.

[12] J. G. Brankov, Y. Yang, and M. N. Wernick. Tomographic image reconstruction based on a content-adaptive mesh model. *IEEE Trans. on Medical Imaging*, 23(2):202–212, February 2004.

[13] C.Dyken and M. S. Floater. Preferred directions for resolving the non-uniqueness of delaunay triangulations. *Computational Geometry—Theory and Applications*, 34:96–101, 2006.

[14] S. A. Coleman, B. W. Scotney, and M. G. Herron. Image feature detection on content-based meshes. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 844–847, 2002.

[15] F. Davoine, M. Antonini, J.-M. Chassery, and M. Barlaud. Fractal image compression based on Delaunay triangulation and vector quantization. *IEEE Trans. on Image Processing*, 5(2):338–346, February 1996.

[16] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, New York, NY, USA, 2nd edition, 2000.

[17] B. Delaunay. Sur la sphere vide. *Bulletin of the Academy of Sciences of the USSR, Classes des Sciences Mathematiques et Naturelle*, 7(6):793–800, 1934.

[18] N. Dyn. Data-dependent triangulations for scattered data interpolation and finite element approximation. *Applied Numerical Mathematics*, 12:89–105, 1993.

[19] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10:137–154, 1990.

[20] M. A. Garcia and B. X. Vintimilla. Acceleration of filtering and enhancement operations through geometric processing of gray-level images. In *Proc. of IEEE International Conference on Image Processing*, volume 1, pages 97–100, Vancouver, BC, Canada, 2000.

[21] M. Garland and P.S.Heckbert. Fast polygonal approximation of terrains and height field. Technical report, School of Computer Science, Carnegie Mellon University, pittsburgh, PA, USA, September 1995.

[22] K.-L. Hung and C.-C. Chang. New irregular sampling coding method for transmitting images progressively. *IEE Proceedings Vision, Image and Signal Processing*, 150(1):44–50, February 2003.

[23] Kodak lossless true color image suite. `http://r0k.us/graphics/kodak`, 2011.

[24] C. L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3:365–372, 1972.

[25] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, New York, NY, USA, 1977.

[26] P. Lechat, H. Sanson, and L. Labelle. Image approximation by minimization of a geometric distance applied to a 3D finite elements based model. In *Proc. of IEEE International Conference on Image Processing*, volume 2, pages 724–727, 1997.

[27] P. Li and M. D. Adams. A tuned mesh-generation strategy for image representation based on data-dependent triangulation. *IEEE Trans. on Image Processing*, 22(5):2004–2018, May 2013.

[28] X. Ma and M. D. Adams. An improved error-diffusion approach for generating mesh models of images. *Signal Processing*, 117:17–32, December 2015.

[29] E. Osherovich and A. M. Bruckstein. All triangulations are reachable via sequences of edge-flips: an elementary proof. *Computer Aided Geometric Design*, 25:157–161, 2008.

[30] M. Petrou, R. Piroddi, and A. Talebpour. Texture recognition from sparsely and irregularly sampled data. *Computer Vision and Image Understanding*, 102:95–104, 2006.

[31] E. Quak and L. L. Schumaker. Least squares fitting by linear splines on data dependent triangulations. In P. J. Laurent, A. Le Mehaute, and L. L. Schumaker,

editors, *Curves and Surfaces*, pages 387–390. Academic Press, Boston, MA, USA, 1991.

[32] G. Ramponi and S. Carrato. An adaptive irregular sampling algorithm and its application to image coding. *Image and Vision Computing*, 19:451–460, 2001.

[33] S. Rippa. Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1123–1141, 1992.

[34] S. Rippa. Long and thin triangles can be good for linear interpolation. *SIAM Journal on Numerical Analysis*, 29(1):257–270, 1992.

[35] M. Sarkis and K. Diepold. A fast solution to the approximation of 3-D scattered point data from stereo images using triangular meshes. In *Proc. of IEEE-RAS International Conference on Humanoid Robots*, pages 235–241, Pittsburgh, PA, USA, November 2007.

[36] L. L. Schumaker. Computing optimal triangulations using simulated annealing. *Computer Aided Geometric Design*, 10:329–345, 1993.

[37] D. Su and P. Willis. Demosaicing of colour images using pixel level data-dependent triangulation. In *Proc. of the Theory and Practice of Computer Graphics*, pages 16–23, 2003.

[38] D. Su and P. Willis. Image interpolation by pixel-level data-dependent triangulation. *Computer Graphics Forum*, 23(2):189–201, 2004.

[39] United States Geological Survey. Shuttle radar topography mission void-filled dataset. `http://lta.cr.usgs.gov/SRTMVF`, 2017.

[40] University of Southern California. USC-SIPI image database. `http://sipi.usc.edu/database`, 2011.

[41] Y. Wang, O. Lee, and A. Vetro. Use of two-dimensional deformable mesh structures for video coding, part II–the analysis problem and a region-based coder employing an active mesh representation. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(6):647–659, December 1996.

[42] J. Weisz and R. Bodnar. A refined "angle between normals" criterion for scattered data interpolation. *Computers and Mathematics with Applications*, 41:531–534, 2001.

[43] Y. Yang, M. N. Wernick, and J. G. Brankov. A fast approach for accurate content-adaptive mesh generation. *IEEE Trans. on Image Processing*, 12(8):866–881, August 2003.

[44] X. Yu, B. S. Morse, and T. W. Sederberg. Image reconstruction using data-dependent triangulation. *IEEE Computer Graphics and Applications*, 21(3):62–68, May 2001.