GEOMETRIC ANALYSIS TOOLS FOR MESH SEGMENTATION

by

Beibei Zhou

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2013

Approved by:

Dr. Andrew R. Willis

Dr. Thomas P. Weldon

Dr. Jiang (Linda) Xie

Dr. Deborah Sharer

ABSTRACT

BEIBEI ZHOU. Geometric analysis tools for mesh segmentation. (under the direction of DR. ANDREW WILLIS)

Surface segmentation, a process which divides a surface into parts, is the basis for many surface manipulation applications which include model metamorphosis, model simplification, model retrieval, model alignment and texture mapping. This dissertation discusses novel methods for geometric surface analysis and segmentation and applications for these methods. Novel work within this dissertation includes a new 3D mesh segmentation algorithm which is referred to as the ridge-walking algorithm. The main benefit of this algorithm is that it can dynamically change the criteria it uses to identify surface parts which allows the algorithm to be adjusted to suit different types of surfaces and different segmentation goals. The dynamic segmentation behavior allows users to extract three different types of surface regions: (1) regions delineated by convex ridges, (2) regions delineated by concave valleys, and (3) regions delineated by both concave and convex curves. The ridge walking algorithm is quantitatively evaluated by comparing it with competing algorithms and human-generated segmentations. The evaluation is accompanied with a detailed geometrical analysis of a select subset of segmentation results to facilitate a better understanding of the strengths and weaknesses of this algorithm.

The ridge walking algorithm is applied to three domain-specific segmentation problems. The first application uses this algorithm to partition bone fragment surfaces into three semantic parts: (1) the fracture surface, (2) the periosteal surface and (3) the articular surface. Segmentation of bone fragments is an important computational step necessary in developing quantitative methods for bone fracture analysis and for creating computational tools for virtual fracture reconstruction. The second application modifies the 3D ridge walking algorithm so that it can be applied to 2D images. In this case, the 2D image is modeled as a Monge patch and principal curvatures of the intensity surface are computed

for each image pixel. These principal curvatures are then used by ridge walking algorithm to segment the image into meaningful parts. The third application uses the ridge walking algorithm to facilitate analysis of virtual 3D terrain models. Specifically, the algorithm is integrated as a part of a larger software system designed to enable users to browse, visualize and analyze 3D geometric data generated by NASA's Mars Exploratory Rovers Spirit and Opportunity. In this context, the ridge walking algorithm is used to identify surface features such as rocks in the terrain models.

ACKNOWLEDGMENTS

My Ph.D. studies were supported as research assistantship from the NIH grants P50AR-055533 and 1R21AR054015, and as teaching assistantship from UNC Charlotte Graduate School Graduate Assistant Support Plan (GASP) and the Electrical and Computer Engineering Department. The support of my professors, family, student colleagues and friends, has made the completion of this dissertation possible.

I must first express my gratitude to my advisor, Dr. Andrew R. Willis, whose expertise and understanding added considerably to my graduate experience. I appreciate many things he taught me and his assistance in writing this dissertation. Without the helpful discussions and directions, this dissertation would not have been possible.

I'd like to thank many professors who have taught me and guided me during my PhD education in classes, in projects and even in my daily life: Dr. Thomas Weldon, Dr. Jiang Xie, Dr. Deborah Sharer, and Dr. Martha Cary Eppes. I would like to thank the graduate students I have worked with: Yunfeng Sui, Pengcheng Liu, Elias Mahfoud, Longjiang E and Jungphil Kwon. They each helped make my time in the Ph.D. program more fun and interesting. I look forward to future collaboration with any of them.

Lastly, I wish to thank my husband, Wenyi Shao, my parents, Kai Zhou and Xiuqin Liu, and my parents in law, Yazhu Liu and Jingliang Shao. They supported me, helped me and love me. This dissertation is also dedicated to my son, David Pinyu Shao, I love him so much. He is the best gift I have ever received.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

This dissertation proposes novel segmentation algorithms for segmenting 3D surfaces and 2D images into semantic parts. The algorithm uses geometric shape features to group or separate the surface and image data into parts. The methods are used in two applications: (1) medical image analysis and (2) geographical surface image analysis.

Segmentation is the process of partitioning data into multiple parts or segments. The input to a segmentation algorithm is an unorganized collection of data measurements, $D = \{\cup_i d_i\}$, and the output associates each of these data points to one of N different groups. This generates a label for each data point, (d_i, ζ^j), where ζ^j is a label that specifies the group associated with i^{th} data point. The data can then be re-organized by these groups based on the label values: $S = \{S^1, S^2, \cdots, S^N\}$, $S^i \subseteq D$, $\{(S^1, \zeta^1), (S^2, \zeta^2), \cdots (S^N, \zeta^N)\}$. In general, data within each group is intended to share the same properties, while data in different groups are intended to have different properties.

Segmentation is a problem that has different realizations depending upon the nature of the data. For the problem of 3D surface segmentation, the input data is a mesh surface which consists of mesh vertices, edges and faces. The output may be collections of vertices, edges or faces where elements in the same segment should have similar properties (such as curvature, appearance attributes, etc.) and other segments should have significantly different values for these same properties. Surface segmentation algorithms are important because they simplify and/or change the representation of the data into something that is more semantically meaningful and easier to analyze [1]. Segmented 3D surfaces are needed for a wide range of problems, such as reverse engineering [2], object modeling [3], skeleton extraction [4], surface animation [5], medical treatment [6] and geological field

measurement [7].

Image segmentation algorithms take as input a digital image consisting of pixels which have intensity/color measured on an (x, y) grid. The output of image segmentation algorithms is a collection of pixel groups where pixels in the same group are intended to have similar properties (such as color, intensity or texture). Image segmentation algorithms are important because they can be used in a wide range of applications, such as medical diagnosis [8], human face recognition [9], and fingerprint recognition [10].

1.1 Structure of this Dissertation

This dissertation is composed of seven chapters which collectively detail my investigations on the use of shape for segmentation of 3D surfaces and 2D imagery.

Chapter 1 defines the general segmentation problem and the specific areas of investigation for this dissertation. For each investigation, the specific problem, motivation, and contribution of the work in this dissertation are briefly discussed.

Chapter 2 discusses a novel geometric segmentation method for 3D surfaces referred to as *"ridge walking"*. The method computes contours that separate, i.e., segment, the surface into semantic parts by following contours that traverse ridge and valley substructures of the surface. The ridge walking segmentation algorithm is described and results are shown for several different surfaces.

Chapter 3 quantitatively evaluates the ridge walking algorithm by comparing its results with ground truth segmentation examples for a large dateset of 3D models. Quantitative evaluations are also computed for several competing methods by comparing their segmentation results with the same ground truth examples. The geometric properties of segmentations generated by ridge walking algorithm are also analyzed in detail to better understand the strengths and weakness of the ridge walking segmentation algorithm.

Chapter 4 introduces a new method for surface segmentation that integrates appearance information to improve the results of geometric segmentation. Specifically, the method uses CT voxel intensities as appearance attributes for vertices of bone fragment surfaces

segmented from the CT images. By using these intensities to classify the bone tissues in the vicinity of the surface, the segmentation result can be improved and can also be transformed to include physiological semantic information.

Chapter 5 discusses adaptation of the 3D ridge walking algorithm for the use on 2D images. Here the 2D image is modeled as a Monge patch and the ridge walking algorithm is modified to accommodate this special structure. The 2D image segmentation results are quantitatively compared with several ground truth segmentations and several competing segmentation algorithms using two different evaluation metrics.

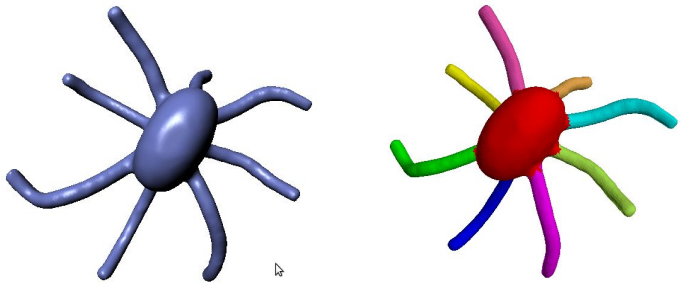
Chapter 6 discusses an application of the developed surface segmentation algorithm as part of a system for geological analysis of 3D Martian surface data. The software system is designed to enable users to browse, visualize and analyze image data generated by NASA's Mars Exploratory Rovers Spirit and Opportunity. In this context, the ridge walking algorithm is used to identify surface features such as rocks in the terrain models.

Chapter 7 concludes the dissertation, summarizes the accomplishments of this dissertation, and discusses areas of potential future investigations.

1.2 Areas of Investigation

Each area of investigation defines a problem to be solved, the motivations to solve the problem and the contribution associated with the approach discussed in this dissertation. The investigated problems include:

1. Theory: Geometric 3D surface segmentation and performance evaluation (Chapter 2 and 3),
2. Application: Improving bone fragment geometric segmentation using appearance data (Chapter 4),
3. Theory: Geometric methods for image segmentation (Chapter 5),
4. Application: A software system for geological analysis of 3D Martian surface data (Chapter 6).



(a) (b)

Figure 1.1: An example of mesh surface segmentation. The input of the segmentation is a mesh surface as shown in (a), the segmentation algorithm provides a decomposition of the model surface into meaningful parts as shown in (b) as surface patches having distinct colors.

These four investigations include two different theoretical formulations of the ridge walking segmentation algorithm and two different applications for its use. A summary of each investigation is introduced in the following sections (§1.2.1-§1.2.4).

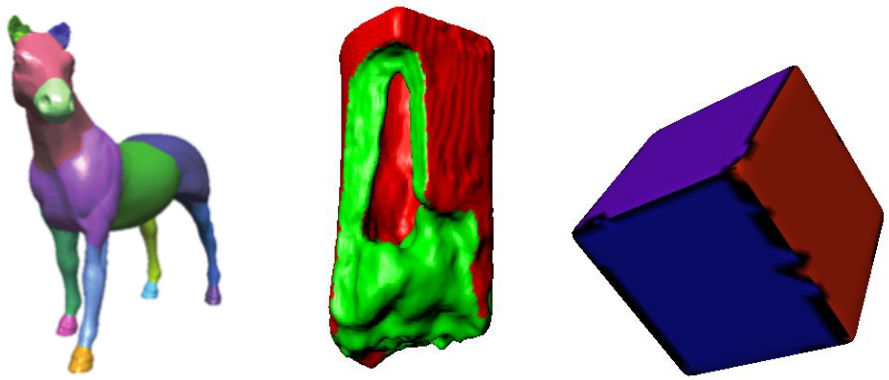
1.2.1 Geometric 3D Surface Segmentation and Performance Evaluation

The prevalence of 3D surface modeling and 3D capture technologies has made it much easier to generate complex models. In order to work with these models, there is often a need to decompose a single complex model into separate parts where each part has simplistic geometry (see figure 1.1). Surface segmentation algorithms segment a polygonal surface into different regions which have uniform properties, either from a geometric point of view or from a semantic point of view [11]. This dissertation proposes a new segmentation algorithm that segments a mesh surface into regions by computing surface contours that traverse specific surface substructures. The specific surface substructures are: (1) convex ridge-like surface substructures, (2) concave valley-like surface substructures, and (3) both concave and convex surface substructures. The method is referred as *ridge-walking* algorithm.

Mesh segmentation is a key step for many mesh manipulation applications. One application focuses on animation and deformation of three-dimensional surfaces. In this context, segmentation is useful for extracting the complex transformations between articulated shapes and for helping simplify these transformations [12, 13]. Segmentation is also used

for content-based 3D model retrieval. In this context, 3D models are divided into a collection of surface patches, which are subsequently used as query objects in order to retrieve similar models which include similar surface patches [14]. For mesh simplification, segmentation is used to decompose meshes into surface patches. Within each resulting surface patch, the mesh is simplified which helps preserve distinctive features of the model while simultaneously reducing its size [15]. For texture mapping, segmentation algorithms have been used to minimize texture stretching and distortion. This is possible by segmenting the mesh into regions with disk-like topology and then applying textures to each region [13]. Segmentation of mesh surfaces is also important in medical imaging applications. For example, segmentation of bone fragment surfaces into anatomically meaningful regions allows them to be analyzed and allows their parts to be matched to reconstruct the bone [16].

In the literature, most segmentation algorithms use the minima rule as a basis of segmentation, which states that humans tend to perceptually decompose 3D surfaces into parts along the concave seams of the surface [17] (examples are shown in figure 1.2(a) and 1.3). As a result, most automatic segmentation methods seek to segment 3D surfaces along concave surface contours. However, there are several special cases of particular interest to this investigation where this rule may not generate desirable results. For bone reconstruction, 3D bone fragment surfaces need to be segmented into parts in order to be matched together to reconstruct the bone (see chapter 4 for details). In this case, the minima rule does not produce desirable results because bone fragment surfaces are delimited by contours that follow the convex ridges of the fragment surfaces, as shown in figure 1.2 (b). For Computer Aided Design (CAD) and reverse engineering [13], it is often necessary to decompose an object into its geometric primitives such as planes, cylindrical patches, spherical parts, etc. In these situations, the minima rule may also fail to produce desirable results. An example is shown in figure 1.2 (c), where a cube is segmented into six planes. *This motivates the development of algorithms that can change their segmentation behavior to suit the criteria of the application. Chapter 2 of this dissertation discusses a segmentation algorithm that*



(a) (b) (c)

Figure 1.2: Different segmentation requirements for different applications. (a) shows a segmentation application where the minima rule is appropriate to divide the object into physical "semantic parts" (taken from [18]). (b) shows a medical imaging segmentation application where it is appropriate to divide the bone fragment into parts along convex surface ridges. (c) shows a CAD segmentation requires the cube to be segmented into six planes via the convex ridges that bound these planes.

has this dynamic behavior and allows the input mesh to be segmented in a way appropriate to the specific application.

Chapter 3 of this dissertation describes the methods used to quantitatively evaluate the proposed ridge walking algorithm and a detailed analysis of a select subset of segmentation results that serve to help understand the strengths and weaknesses of this algorithm. Quantitative evaluation is accomplished by comparing the ridge walking segmentation results with a collection of ground truth segmentation results generated by humans. Four evaluation metrics taken from [19] are used to score the difference between ridge walking and ground truth segmentations. These metrics compare two segmentation results for the same surface by measuring the differences/similarities between their segmented surface boundaries or regions. Analysis is accomplished by studying the geometric properties of the segmentation boundaries and regions generated by the ridge walking algorithm for a select group of interesting models.

Contribution

The ridge walking algorithm developed as part of this dissertation represents an advance-

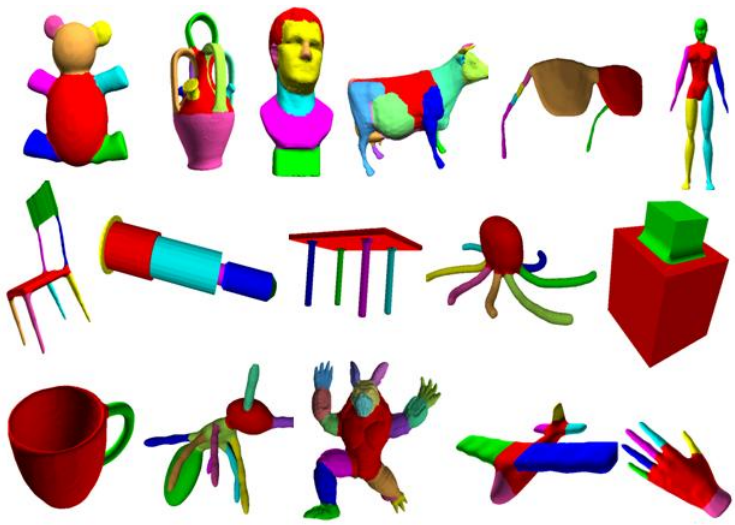
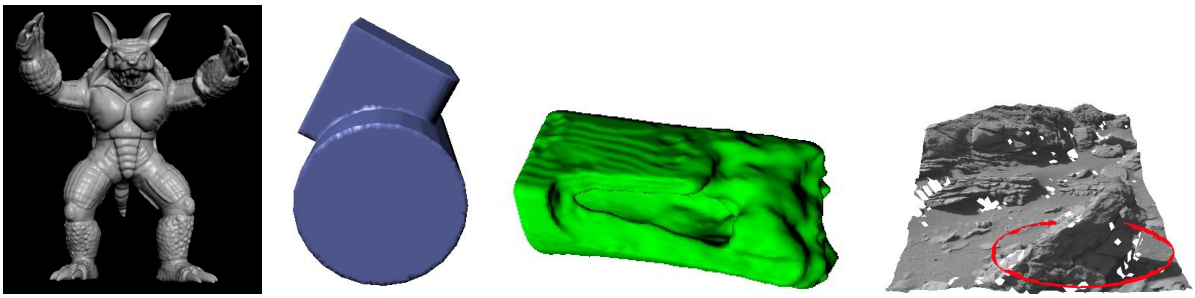


Figure 1.3: Representative 3D surfaces from [20] that have been manually segmented by humans.

ment over the state-of-the-art in 3D surface segmentation approaches. The following list specifies contributions associated with the proposed method that are supported by the performance analysis discussed in Chapter 3:

1. It can be applied to a wide variety of segmentation problems,
2. The segmentation boundary can be controlled.

For contribution (1), the dynamic segmentation criterion allows the ridge walking algorithm to perform well on a broad number of surface categories. Some categories include: (1) mesh surfaces generated from 3D capture technologies using laser scanners as shown in figure 1.4(a), (2) mesh surfaces generated from CAD programs as shown in figure 1.4(b), (3) medical surfaces generated from CT images as shown in figure 1.4(c), and (4) 3D models reconstructed from stereoscopic image sequences as shown in figure 1.4(d), etc. These different surface types require different segmentation criteria. The different segmentation criteria available in the ridge walking algorithm allow users to segment surfaces along: (1) concave valleys, (2) convex ridges, or (3) both concave and convex structures. These flexible criteria make the ridge walking algorithm suitable for processing a larger variety of input surfaces than current segmentation methods.



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

Figure 1.4: Surfaces that present challenges for current mesh segmentation approaches. (a) Armadillo model scanned by Stanford University Computer Graphics Laboratory, (b) CAD model from [20], (c) bone fragment surfaces generated from 3D CT images, (d) the Martian stereoscopic reconstructed surface by the NASA sponsor Mars Exploratory Rovers (REM) of *Spirit* and *Opportunity*, which is noisy and has some holes on the surface.

For contribution (2), the proposed algorithm allows the user to control attributes of the segmentation boundary to enforce that it has specific properties. Controlling contours directly in this way allows the user to enforce constraints on the computed segment boundaries that may be difficult to impose using region based methods. Important constraints typically include properties of the boundary curve such as total length, smoothness and curvature.

1.2.2 Improving Bone Fragment Geometric Segmentation using Appearance Data

Segmentation of bone fragments is an important computational step for developing computational tools for bone fragment analysis and virtual bone fragment reconstruction. In the process of clinical bone fragment reconstruction, good prognosis for the patient requires that the aligned bone fragments agree geometrically and provide both a good mechanical union between the fragments as well as an accurate reproduction of the original smooth outer bone surface which is especially important in the articular regions, i.e., regions of bone surfaces at a joint where the ends of bones meet. Virtual bone reconstruction seeks to reconstruct a bone by piecing together bone fragment models within a virtual environment. To facilitate analysis and virtual reconstruction, it is of importance to accurately segment bone fragment surfaces into surface patches. Each surface patch is estimated to come from

one of the following three categories: (i) *periosteal surface patches*, i.e., surface regions that were part of the original outer surface of the intact bone, (ii) *fracture surface patches*, i.e., surface regions generated when the bone fragments broke apart, and (iii) *articular surface patches*, i.e., surface regions located at joints where the ends of bones meet.

Segmentation of bone fragments is a difficult task. Most existing geometric segmentation methods are unsatisfactory, as their strategy for partitioning surfaces seeks to divide surfaces along concave valleys which is not an effective approach for bone fragment segmentation. For bone fragments, the desired segmentation boundaries often lie along highly convex ridges. The ridge walking algorithm can segment the bone fragments into patches along convex ridges of the surfaces based their geometry. Unfortunately, but the segmented patches do not have medically-relevant semantic information which is related to one of the three categories above.

Chapter 4 of this dissertation describes a method which uses both the geometry and CT intensity values to segment bone fragments that are reconstructed from 3D CT data. The approach uses CT intensity values taken from the image of bone fragment tissues to segment the surface into medically meaningful semantic parts. Different regions of a tibia bone have different tissues and these tissues have different CT intensities. As shown in figure 1.5, the diaphysis part is made up of solid dense cortical bone having high intensities. The metaphysis part is made up of an outer cortical shell having high intensities and a less dense and porous cancellous bone on the interior having lower intensities. The articular part is at the end of the long bone and consists of relatively dense subchondral bone having intensities that lie in-between the intensities for cortical and cancellous bone tissues. These tissue variations show up as intensity variations in the 3D CT images and are used by this new method to classify segmented surface patches.

Contribution

The proposed shape and appearance segmentation approach provides several benefits not available in competing approaches. The following list describes the contributions of the

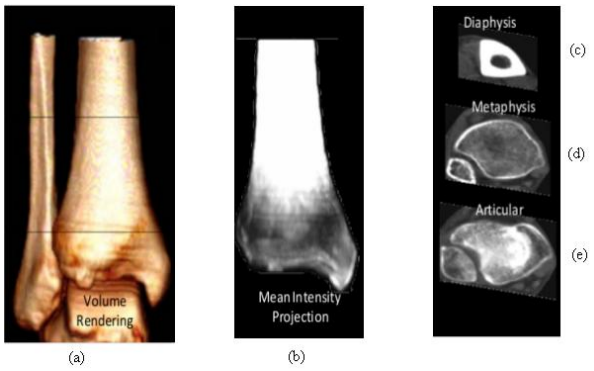


Figure 1.5: Images(a-e) depict how bone tissue intensities vary along the length of the tibia. (a) shows a coronal view of the tibia. Planar cross section images are extracted at 3 locations along the bone giving (c), (d) and (e). (b) The average intensity of bone tissue in the image is projected into the viewing plane.

proposed bone fragment surface segmentation algorithm:

1. It improves upon the unreliable segmentation results often generated by segmenting the surface using only geometric information,
2. It generates a segmentation result that includes important medically meaningful semantic information.

For contribution (1), the algorithm improves the segmentation result by fusing shape (geometric) and appearance (CT voxel intensities of bone fragment surfaces) attributes for bone fragment surface segmentation. The ridge walking algorithm uses geometric information to segment the surface and an appearance classifier uses the CT intensity values to label the points within surface patches to three medically meaningful classes. The appearance model for classification of the surfaces represents a significant improvement over previous methods [21]. By combining the geometric segmentation produced by ridge walking with an appearance-based classifier, more accurate segmentations are obtained that include a clinically relevant labeling of the fragment surface data.

For contribution (2), the appearance model allows the surface to be segmented into medically relevant semantic parts. This is achieved by inferring a medically relevant semantic class from the observed CT intensity data near the bone surface which is different

for different tissues of the bone. This semantic information for fragment sub-surfaces is particularly useful for reconstructing of the bone from its fragments and for analysis and visualization of fracture case data.

1.2.3 Geometric Methods for Image Segmentation

Image segmentation is the process of partitioning a 2D image into regions where each region represents a separate object or structure in the image. The input of an image segmentation algorithm is an image which specifies pixel values defined at a collection of (x,y) grid locations. The output of an image segmentation algorithm is a collection of pixel groups or segments. Image segmentation algorithms seek to construct segments that contain pixels which have homogeneous properties. For example, a color based segmentation algorithm may seek to group pixels in the same set that are adjacent and have similar colors.

Image segmentation is an initial and vital step in applications which seek to explain the semantic structure of the image. For example, in photo-editing, image segmentation can be used to isolate objects in the image so that they can be manipulated independently. In the medical field, image segmentation tools are used to locate organs for surgeons to view, analyze, and devise treatment plans. In the security field, image segmentation can be used for automatic face recognition and fingerprint recognition. In remote sensing field, image segmentation tools are used to automatically locate objects in satellite images such as roads and forests.

This dissertation describes an extension of the 3D ridge walking segmentation algorithm so that it may be used for 2D image segmentation. The extension changes the original assumption that the surface being segmented is a generic 3D surface to assume that the surface may be represented as a Monge patch. A Monge patch can be represented mathematically as the graph of a two dimensional function, $f(x,y)$, where the function value is taken as the image intensity value at image grid location (x,y). The principal curvatures of images can be computed numerically using the Monge patch representation as shown in figure 1.6. The ridge walking algorithm uses these principal curvatures to compute the

(a) (b)

Figure 1.6: The curvatures of the image. The left column is the original image, the right column shows the maximum curvatures of the left image

contours that divide the image into distinct parts.

Contribution

The following list describes the contributions of the proposed ridge walking image segmentation:

1. It derives a version of the ridge walking algorithm appropriate for 2D image data,
2. It performs initial experiments that show this algorithm may perform well for some 2D image segmentation problems.

For the contribution (1), the image intensity surface is modeled as a Monge patch which has the parametric form $M(x,y) = (x,y,f(x,y))$, where (x,y) denotes the pixel location, and $f(x,y)$ denotes the pixel intensity value at that location. Using the Monge patch representation, the principal curvatures and principal directions can be computed. The ridge walking algorithm uses this curvature information to segment images.

For the contribution (2), the ridge walking algorithm extends the 3D ridge walking algorithm to detect ridge and valley structures in the image. The ridges and valleys in the image correspond to locations where the image pixel intensities exhibit large curvatures. The segmentation results on several test images show that the ridge walking can capture the interesting structures for those images, and the segmentation results are comparable with other competing algorithms.

1.2.4 Application: A System for Geological Analysis of 3D Martian Surface Data

Images from the Mars Rovers Spirit and Opportunity have been streaming to the Earth since their landing on the Martian surface in January 2004. These images are periodically released to the public organized by the number of Earth days (sols) since the landing, where each day commonly includes several hundred images. Current estimates indicate that there are roughly 200k images available from the Mars Exploratory Rover (MER) missions which includes 2209 sols of data available from Spirit and 2700 sols of data available from Opportunity. These images have been transferred to the Earth and reconstructed as a raw data referred to as Experimental Data Records (EDRs). Among the EDR data, Effective Full Frame (EFF) images are images collected by the rover Pancam instrument having 1024×1024 resolution. Examples of several different EFF images are shown in figure 1.7. EFF images are often processed using a suite of software tools developed by the Multimission Image Processing Laboratory (MIPL) which is a part of the NASA Jet Propulsion Laboratory (JPL). These programs take as input one or more EFF images and integrate the measured data into a new output image referred to as a Reduced Data Record (RDR). The RDRs are numerous and include color images formed from multiple images of a surface using different optical filters [22, 23], image mosaics and 3D stereoscopic images reconstructed from the Pancam stereo system.

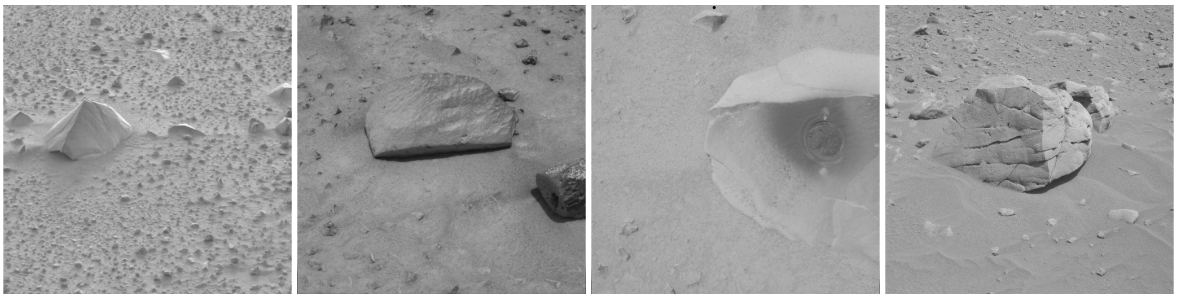


Figure 1.7: Exemplar Effective Full Frame (EFF) data records.

The 3D stereoscopic data produced by the Pancam imaging system is of particular interest for geological research. These datasets are generated via a technique known as

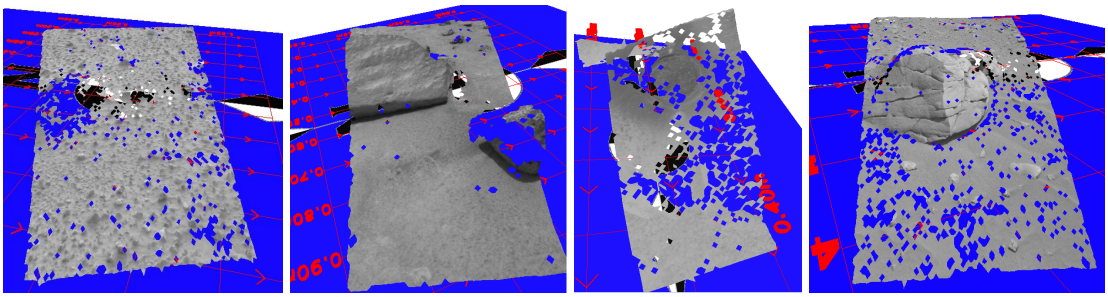


Figure 1.8: Exemplar Reduced Data Record (RDR) stereoscopic reconstructed images.

stereoscopic 3D reconstruction and are stored in an XYZ RDR that contains estimates for 3D (x, y, z) surface positions of objects viewed within a stereoscopic pair of Pancam EFF images [24]. The underlying concept of 3D reconstruction from images taken by a pair of cameras is that by knowing or estimating the image formation properties of each camera, their relative poses, and the pixel pair in each digital image that corresponds to a specific 3D surface location, one may invert the image formation process and find the 3D locations responsible for reflecting the light sensed by the cameras. Examples of several different XYZ RDR stereoscopic reconstruction images are shown in figure 1.8.

Several geological hypotheses can be tested using 3D measurements of the Martian surface as provided by stereoscopic reconstruction images. These hypotheses include:

1. The thickness and orientation of bedding planes such as those visible in figures 1.7 might provide insight into past aeolian or fluvial processes acting on the Mars surface. Aeolian processes refer to the activity of winds to shape the surface of a planet. Fluvial processes refer to the activity of rivers or streams to create landforms on planets. Wind and water may erode, transport, and deposit materials, and are effective agents in regions with a large supply of unconsolidated sediments. Measuring the orientation of these bedrock features shows promise for enabling geo-scientists to test hypotheses related to possible past plate tectonics on Mars [25].
2. The size and distances between rocks on Martian surface such as those visible in figure 1.7 have the potential to provide unknown insights on ejecta emplacement.

Ejecta emplacement describes the process for volcanic particles deposition on the surface of terrestrial planets.

Our goal is to provide tools capable of extracting 3D information from this data that can contribute to furthering our understanding of these phenomenon.

Current MER data analysis software only enable scientists to view the 2D images from Martian surface and includes methods to query the database of images based on time and place. There is no software that provides the ability to view the 3D XYZ RDR images, not to mention providing the ability to make measurements directly from the MER 2D/3D image data (see details in §6.1), which limits the utility of the recorded information (crack length, orientation, distance between rocks, etc.). This motivates the development of software that allows users to visualize and analyze the 2D and 3D Martian surface image data and to extract geometric measurement data from these images.

Contribution

The proposed software includes the following contributions:

1. It enables users to efficiently locate the images of interest to them within the MER image database,
2. It enables users to visualize the 2D and 3D images from Martian surfaces,
3. It provides interactive methods to make geometric measurements on Martian surfaces shown in these 2D and 3D images.

For contribution (1), the software provides three methods that enable users to efficiently locate images of interest to them within the MER database. Querying methods available are: time, spatial location and desirable image content. The time querying method finds images based on the sol date the images was collected. The spatial location querying method finds images based on the geographical location where the images were recorded. The image content querying method finds images using a query-by-image method. This method asks the user to provide an image that is an example of the desired object and the

system finds images that include similar content. For example, a user can submit an image of rocks to find images that contain rocks. These querying methods seek to allow scientists to quickly find data of relevance to their scientific investigations.

For contributions (2) and (3), the software provides the ability to visualize 2D EFF EDR images and their corresponding 3D RDR images at the same time. The visualization of 3D RDR images enables users to extract geometric information directly from recorded Martian EFF and 3D RDR data. The geometric measurement tools include capabilities for measuring the orientation, size and surface area of rocks. Such measurements are useful to investigate geological hypotheses, such as the physical weathering processes of the Martian surface.

1.3 Summary

This dissertation focuses on geometric methods for surface segmentation and includes a description for an algorithm that advances the state-of-the-art in this area. Enhancements of this core concept are described that include one extension and two applications for this segmentation method. The dissertation introduces a new geometric 3D mesh surface segmentation algorithm called the ridge walking algorithm. The main benefit of this method is that it has dynamic segmentation behavior which allows it to be used for a wide variety of input surfaces. The dynamic segmentation behavior allows users to extract segments from the surface that are delineated by special sub-structures such as: (1) convex ridges, (2) concave valleys, (3) both concave and convex structures. The ridge walking method is quantitatively evaluated by comparing its results with those of other segmentation algorithms and with ground truth examples. The results show the geometric segmentations produced by the ridge walking algorithm are close to the ground truth.

The dissertation also introduces two applications and one extension of the ridge walking algorithm. The first application uses the algorithm for bone fragment surface segmentation. Here, appearance data (CT intensity values) are fused with the ridge walking algorithm to provide more accurate segmentation results with a clinically relevant labeling. The bone

fragment surfaces are segmented into periosteal, fracture and articular surface patches. The 3D ridge walking algorithm is then extended to work on 2D images. Here, the 2D image is modeled as a Monge patch and the principal curvatures of the image surface are computed for each image pixel. The ridges and valleys of 2D images correspond to locations where the image pixel intensities exhibit large principal curvatures. The goal of 2D image segmentation is to find segmented boundaries which delimit homogenous regions of the image. These boundaries tends to lie along contours that traverse the ridges/valleys of the image. The second application uses the ridge walking algorithm to facilitate analysis of virtual 3D terrain models. Specifically, the algorithm is integrated as a part of a larger software system designed to enable users to browse, visualize and analyze 3D surface data recorded by the Mars Exploratory Rovers Spirit and Opportunity. The ridge walking algorithm is used to identify surface features such as rocks in the terrain models.

CHAPTER 2: GEOMETRIC 3D SURFACE SEGMENTATION USING RIDGE WALKING

This chapter describes the *"ridge walking"* algorithm for 3D surface segmentation. The algorithm separates a mesh surface into regions by computing surface contours that traverse geometrically-salient surface substructures. A typical mesh model is shown in figure 2.1. Geometric mesh surface segmentation is an important surface analysis step needed when processing both synthetic (CAD-generated) and real-world (via 3D scanning) models which are commonplace in many contexts today. Applications of segmentation include surface compression, object recognition, texture mapping, surface re-parametrization, animation, collision detection and reverse engineering.

Mesh segmentation algorithms decompose a mesh surface into different regions (i.e. connected set of vertices or facets) where each region is intended to have some semantic meaning [11]. The input of a 3D surface segmentation algorithm is a mesh surface. The output is a labeling of the mesh vertices such that each vertex is assigned a value from the label set. Vertices sharing the same label are grouped into spatially contiguous regions called segments. Hence, a mesh segmentation algorithm takes a surface mesh as input and

Figure 2.1: Examples of mesh surfaces.

provides a partitioning of the surface into regions as output. The boundary of a segmentation is defined as those vertices which belong to more than one region.

Most methods in the literature make use of the *minima rule* which states that human perception divides a surface into parts along concave contours of the surface [17], i.e., we perceive concavities as appropriate regions for separating a surface into parts. However, as discussed in §1.2.1, the minima rule is not suitable for all cases.

This chapter proposes a novel geometric segmentation algorithm that solves for closed ridge contours on the surface, each of which serves to divide the surface into two disjoint regions. A collection of such contours then provides a segmentation of the surface into surface patches which is the segmentation result. The method is referred as *ridge-walking* since these contours tend to follow convex ridge-like structures and/or concave valley-like structures present within the geometry of the model. The proposed approach has a dynamic segmentation criteria for different surfaces types. Three different segmentation criteria are proposed: (1) a concave ridge walking criteria, (2) a convex ridge walking criteria and (3) a mixed concave/convex ridge walking criteria.

2.1 Previous Work

The breadth of application for 3D surface shape based segmentation algorithms has made it a consistent topic of interest for the research community for over two decades and recent surveys in [2] and [26] provide comparative analysis for leading contemporary methods. These methods can be grouped into the following five categories: (1) region growing methods, (2) clustering methods, (3) feature point methods, (4) volumetric methods, and (5) contour based methods.

1. Region growing methods

One approach that has enjoyed long-lasting popularity is segmentation via region-growing [27, 28] which proceeds by selecting (at random) polygons on the surface as seeds to a region and subsequently merging neighboring polygons into each seeded region using to a compatibility criterion, e.g., the dihedral angle is less than a threshold [27]. Region

growth stops when all neighboring polygons fail to satisfy the compatibility criterion. For example, the work in [29] labels mesh vertices with highly negative Gaussian curvature as boundaries using a user-specified threshold. Then seed points are selected randomly among the non-boundary points. The region-growing process is performed that iteratively merges non-boundary points into the seed regions. The process terminates when the grown region is surrounded by boundary vertices. The disadvantage of this method and region growing methods in general is that the output is heavily dependent on the choice of the seed points and the threshold. Random walks segmentation algorithm [30] uses the idea of region growing segmentation, but it includes enhancements such as multi-scale surface-smoothing, user-interactivity, and a stochastic filter to propagate growth more rapidly in flat areas than in regions of complex structure.

The watershed segmentation algorithm can also be classified as a region-based segmentation approach. The watershed method was originally proposed in [31] when it was used for grayscale image segmentation. The algorithm operates on a height function defined on the image pixels. The specific choice of the height function depends on the application. Some algorithms use the pixel intensity value as the height [32], other algorithms use the magnitude of gradient at each pixel as the height [33]. The watershed method derives its name from a metaphor that likens the image height function to a topographic relief. The topographic relief is referred to as: a landscape which the algorithm floods with water. Watersheds are the lines that must be introduced to separate different regions which collect the rain water [34]. Water accumulates at local minima of the landscape. Eventually pools of water from different minima merge and dams are built where water from these different minima meet [35]. This resulting collection of dams define contours that partition the landscape into regions called watersheds. Mangan was the first to extend the watershed algorithm from 2D image segmentation to 3D mesh [36]. Here, the height function for every mesh vertex (x, y, z) is defined as the curvature of that point. Te height function is the combination of mean and Gaussian curvature, see [36] for details. Work in [37] changed

the height function definition to be: $h(x) = 1 - cos(\alpha)$, where α is the dihedral angle between two adjacent faces. One of the main disadvantages of the watershed segmentation algorithm is that it tends to produce too many segments, i.e., it "over-segments" the input data. This is due to the fact that the number of segments is equal to the number of local minima in the height function which typically leads to solutions that are over-segmented. Work in [38] attempts to solve this problem by selecting only the significant local minimas using a hybrid of the height function that considers scale. Other work in [39] attempts to solve the over-segmentation problem by thresholding the height function. The thresholding operation sets all values less than the threshold to be equal to the threshold. This operation is followed by 3D morphological operations which serve to eliminate local minima associated with small regions. Afterwards, the watershed algorithm is applied.

2. Clustering methods

The second category for 3D surface segmentation uses clustering algorithms from pattern recognition to group together faces of the mesh into semantically meaningful groups. It can be further divided into three kinds of clustering methods: (1) iterative clustering, (2) hierarchical clustering, and (3) graph cut clustering.

Iterative clustering methods typically take as input the number of clusters to create. A clustering algorithm then iteratively searches for the best grouping of the data for the given number of clusters. The approach starts with K representative seed pixels which define the initial K clusters. More pixels are iteratively assigned to the closest representative pixel cluster using a similarity metric. Each time the pixels are assigned to a cluster, the cluster center is adjusted to lie at the mean value of the pixels assigned to the cluster. The iterations stop when there are no pixels left to assign. The K-means algorithm is simple and efficient, but the user has to pick these seed pixels of the clusters at the initial step, and the final segmentation results will greatly be affected by the choice for the seed pixels. Another shortcoming of this approach is that, similar to the thresholding approach, the K-means algorithm often generates spatially disconnected regions.

The second kind of clustering approach is referred to as hierarchical clustering. In this approach, each face is initialized as its own cluster. A cost function is defined for cluster pairs that is used to determine which cluster pairs are merged into a single cluster. Cluster pairs are then merged iteratively until a target number of segments is reached. Different algorithms define different merge cost functions. For example, the fitting primitives method [40] is a hierarchical clustering approach. In this case, each cluster of faces defines a subsurface and cost functions fit geometric primitives such as a plane, cylinder or sphere to the faces of a cluster pair. The merge cost function is defined as the fitting error associated with fitting a geometric primitive to a cluster of faces. Specifically, at each iteration, all pairs of adjacent clusters are considered, and the one that can be best approximated by one of the primitives cost functions will be merged to form a single new cluster. The work in [41] describes an efficient hierarchical clustering algorithm. In this work, each pair of adjacent faces is assigned a edge contraction cost that is incurred by merging the faces into the same segment. The edge contraction cost is based on the similarity of the face-pair to a plane and a least cost sequence is used to merge faces into regions.

The third kind of clustering approach is referred to as the graph cut clustering method. This method operates from a dual graph of the mesh. A mesh dual graph is a graph where graph nodes correspond to mesh polygons and graph edges exist between any pair of polygons that share a common edge in the 3D model, as shown in figure 2.2. Each dual graph edge is given a weight, the goal of graph cut is to partition or "cut" the dual graph into two disjoint subsets so that the total weight of the edges along the cut is minimal [42]. The weight can be defined in different ways. The work in [43] defines the weight as the dihedral angle across an edge where the weight is normalized to lie between 0 and 1. The resulting weights are low for concave edges and high for convex ones. However, this definition for edge weight can often generate trivial cuts which partition one face from the rest of the mesh and provide similar small area segments. The work in [42] defines the normalized cut algorithm for mesh segmentation. It uses a method similar to [43], but it re-scales the edge

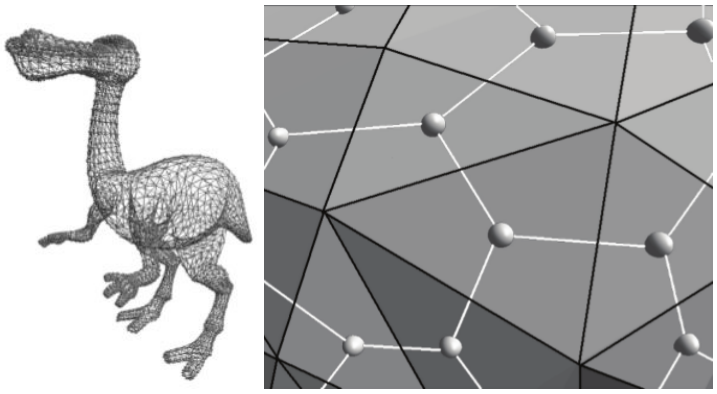


Figure 2.2: Face-adjacent dual-graph of a mesh

weights to avoid generating small regions by weighting each cut by the segment's perimeter divided by its area. As a result, the normalized cut algorithm encourages segments to have short boundaries along concave seams while trying to maintain segmented parts with roughly similar area.

3. Feature point methods

A fourth category of mesh segmentation approaches are those based on finding features, also called landmarks, on the mesh surface. Landmarks are points that are deemed to be visually salient on the object [44]. Feature point methods use these landmarks as a basis of segmenting the surface into parts. Landmark-based segmentation approaches typically place landmarks at local convex extrema of the surface. Regions around these points are then computed which typically divide surfaces along paths that delimit protrusions of an object. In the core extraction segmentation [45], the mesh is transformed into a pose invariant representation using multi-dimensional scaling as described in [3]. In this transformed space, landmarks are extracted as the points that locally maximize a custom-defined protrusion function. Since each critical point is assumed to correspond to a distinct protrusion of the mesh, the algorithm segments the mesh into a collection of surface regions, one for each protrusion of the mesh, and a "remainder" or "core" surface region. The work in [46] also uses landmarks to achieve segmentation. In this method, a root point is manually selected as an extremal vertex located at the tip of a protrusion of the 3D model. The landmarks

of the surface are then computed using a geodesic tree which constrains the detected landmarks to be located at local extrema or saddle points of the surface. Segmented regions are extracted by a "flooding technique" that expands segments starting from the landmark points and stops at saddle points.

4. Volumetric methods

A fifth category for mesh segmentation approaches are those methods that embed the mesh surface into 3D volume, which are referred to as "volumetric" methods. The Shape Diameter Function (SDF) [4] is one of such segmentation method. It defines a metric called the "shape diameter" to segment the surface which approximates the interior volume of the surface within the line-of-sight of each surface point by looking "inside" the mesh. The values of SDF function at each surface point are used to separate the surface into different parts by detecting collections of surface points that have different SDF values. The definition of the SDF function makes it invariant to deformations of the same mesh model, therefore, it is especially useful in segmenting different versions of a model where each version may have different poses.

5. Contour based methods

Contour based segmentation methods seek to compute surface contours that divide the 3D surface into distinct regions. One group of contour-based segmentation methods are the so called "snake" segmentation algorithms. Snake segmentation algorithms start with a contour in the image and subsequently deform this contour until its shape matches the boundary of one or more objects in the image. Early snake methods were described in [47] for 2D image segmentation and these methods were extended in [18] for 3D segmentation. The "mesh scissoring" approach described in [48] improves upon these snake methods by allowing the estimated boundary curves to propagate into concave boundary regions. As noted in [2], one shortcoming that snake models have is that the boundaries can converge to local minima of the curve-evolution function. This can result in undesirable segmentations. Work in [49] proposes a 3D surface segmentation technique based on levelset

methods. Levelset methods, like snake models, also propagate surface contours to object boundaries. For this work, a differential equation for geodesic curvature flow is used to evolve the surface contours. However, this method requires user interaction to help initialize the contours such that they will evolve to a satisfactory segmentation solution. The randomized cut method described in [42] computes segmentation boundaries using a statistical approach. To do so, a random set of mesh segmentations are computed using leading algorithms such as K-means [4], normalized cuts [5], or minimum cut [43]. Statistics are then computed to estimate how likely each edge of the mesh is to be a member of the true segmentation boundary. The randomized cut uses these statistics and the segmentation results to select a segmentation that includes the most consistent collection of edges. The randomized cut method can provide good segmentation results, but its results depend on the statistical analysis of the segmentation results generated by other algorithms.

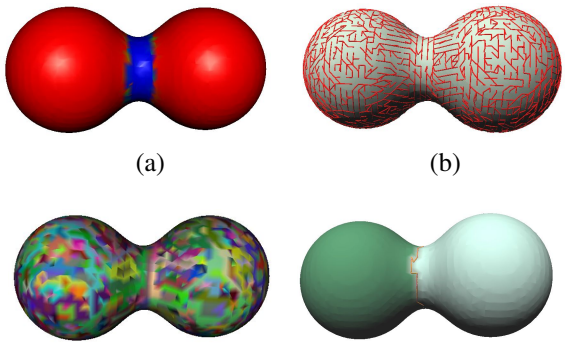
Of these methods, the ridge walking algorithm is most closely related to the mesh scissoring method described in [48]. Both approaches involve methods that solve for contours on the surface that satisfy a salience criteria. The ridge walking algorithm is also related to the graph-cut methods of [50, 42] since the contours the ridge walking algorithm generates are estimated using a graph representation for the points and edges of the polygonal model. Typical graph cut methods define a graph based on the dual-mesh. In this case the graph nodes correspond to mesh polygons and graph edges exist between those polygons that share a common edge in the 3D model. In contrast, the graph used by the ridge walking algorithm is defined over the points and edges of the mesh, where the graph nodes are the points of the mesh and the graph edges are the mesh edges. The ridge walking algorithm is also related to the contemporary work [49], where the authors propose a 3D surface segmentation technique based on a fusion of user interaction and levelset methods that seek concave surface contours that are satisfactory to the user and are solutions to a geodesic curvature flow differential equation. Contours extracted by the ridge walking algorithm can be characterized as contours of geodesic curvature flow [49] for a suitable definition

of the curvature flow and an added constraint that the geodesic solution must follow the discrete parametrization offered by the edges of the mesh. Yet there are a number of differences: (1) the ridge walking algorithm requires a single input parameter (λ or N, see §2.2 for details), (2) the ridge walking algorithm does not rely on user interaction, (3) the ridge walking algorithm includes a variety of salience functions, each of which provides a different segmentation result, and (4) the ridge walking algorithm computes the segmentation explicitly without the need for the iterative methods required by the levelset method, i.e., there is no need to define or solve a partial differential equation on the surface. When compared to the work in [49], the results will be most similar when using the salience function that divides the surface along contours that traverse concave ridges.

The ridge walking algorithm has the following benefits over existing approaches: (1) it can guarantee that the boundary of segmented surface regions will satisfy specific salience criterion, a constraint that is very difficult to enforce for region based methods, and (2) it can directly impose constraints on the form of the segmented surface patch boundaries. By solving directly for the boundary of the surface region, new capabilities are provided that allow one to control the form of the segmentation solution in ways that cannot be easily controlled with polygon-based parametrizations typically used in region-based and clustering segmentation algorithms. This can include control of the segment boundary length, shape, and, most importantly, the overall salience of the region boundary as defined by an optimization function such as the minima rule, which seeks to place boundaries along concave surface regions.

2.2 Methodology

To specify functions that operate on meshes and their parts, we will need to define some mathematical notation. A mesh surface, M, can represent the shape of an object as a collection of vertices, P, edges, E, and faces. Let $\mathbf{p}_i$ denote the i^{th} mesh vertex, and denote the collection of all vertices as the set: $P = \{\cup_i \mathbf{p}_i | \, \mathbf{p}_i = (x_i, y_i, z_i)\}$. The collection of mesh edges are line segments that connect pairs of vertices. Let $(\mathbf{p}_i, \mathbf{p}_j)$ denote one such vertex



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

Figure 2.3: (a) a "dumbbell" surface colorized by its maximum curvature (b) edges of the ridge-tree are shown as a collection of edges superimposed on the surface. The ridge-tree is computed by solving for the spanning tree of a graph defined where graph nodes are mesh vertices and graph edges are the edges of the mesh and the edge weights measure how "ridge-like" the surface is in the direction of the mesh edge. A surface segmentation is obtained by forming cycles in the ridge tree via the insertion of mesh edges referred to as "loopy edges", which is any edge not included in the spanning tree. Each loopy edge insertion creates a cycle in the tree, or equivalently, a simple closed contour on the surface that divides the surface into two parts. (c) shows the surface patches in different color after adding the loopy edges, (d) shows a final surface segmentation after suppressing contours that are nearly equivalent or have very short arc-length, the contour that delimits the green and cyan patches is colored in pink.

pair and $\mathbf{e}_{ij}$ denote the edge that connects this pair. The set of all mesh edges is denoted as: $E = \{\cup_{i,,j}\mathbf{e}_{ij}\}$. The faces of the mesh are convex polyhedra, e.g. triangles, quadrilaterals or other simple convex polygons. In most cases they are triangles specified by their three corner vertices denoted as: $F = \{\cup_i \mathbf{f}_i \mid \mathbf{f}_i = (\mathbf{p}_m, \mathbf{p}_n, \mathbf{p}_k), m \neq n \neq k\}$. We refer to the neighbors of a vertex, $\mathbf{p}_i$, as the set of mesh vertices that are connected by a mesh edge to $\mathbf{p}_i$ which is denoted as: $N_e(\mathbf{p}_i) = \{\mathbf{p}_j | \mathbf{e}_{ij} \in E\}$.

The ridge walking segmentation algorithm seeks to find a parametrization of the surface in terms of the mesh edges where there is a unique path along the parametrization between any two points on the surface. As with other graph based methods [50, 42], a undirected graph over the polygonal mesh is defined. However, in a departure from the norm, we define graph edges to be the mesh edges and graph nodes to be the mesh vertices (note this is a significant departure from standard approaches as described in § 2.1). Let the

graph $G(E,P)$ denote a graph having edges E and nodes P. This graph is an equivalent representation for the mesh and is the object of computation for our segmentation approach.

As with other graph-based methods, a weight is associated with each edge of the graph. This edge weight is a *salience measure*, $w(\mathbf{e}_{ij})$, which measures of how salient the edge is for some segmentation goal. A collection of three salience measures based upon the principal directions of the surface and the minimum and maximum curvatures observed along these directions is proposed. For each surface point, the directions of principal curvature are estimated as two vectors, $\mathbf{v}_i$ and $\mathbf{u}_i$, in the local surface tangent plane and the curvatures of the surface in these directions are denoted as $(\kappa_{max},\, \kappa_{min})_i$ respectively. The computation of curvatures at mesh points in this dissertation is accomplished by applying the quadratic complexity algorithm of Chen and Schmit [51]. *Salience functions* specify the weight of an edge and incorporate two criterion: (1) the curvature of the surface in the direction perpendicular to the edge, and (2) how well the edge aligns with a direction of principal curvature. This is accomplished by approximating the direction of principal curvature at the midpoint of the edge. We utilize three different salience functions as provided in equations (2.1), (2.2), and (2.3).

$$w_{ridge}(\mathbf{e}_{ij}) = \left| \frac{\mathbf{u}_i + \mathbf{u}_j}{\left\|\mathbf{u}_i + \mathbf{u}_j\right\|} \cdot \frac{\mathbf{e}_{ij}}{\left\|\mathbf{e}_{ij}\right\|} \right| \overline{\kappa}_{max} \tag{2.1}$$

$$w_{valley}(\mathbf{e}_{ij}) = \left| \frac{\mathbf{v}_i + \mathbf{v}_j}{\left\|\mathbf{v}_i + \mathbf{v}_j\right\|} \cdot \frac{\mathbf{e}_{ij}}{\left\|\mathbf{e}_{ij}\right\|} \right| (-\overline{\kappa}_{min}) \tag{2.2}$$

$$w_{curv}(\mathbf{e}_{ij}) = \begin{cases} w_{ridge}(\mathbf{e}_{ij}) & |\overline{\kappa}_{max}| \geq |\overline{\kappa}_{min}| \\ w_{valley}(\mathbf{e}_{ij}) & |\overline{\kappa}_{max}| < |\overline{\kappa}_{min}| \end{cases} \tag{2.3}$$

In equations (2.1), (2.2), and (2.3), the quantity $\left| \frac{\mathbf{u}_i+\mathbf{u}_j}{\left\|\mathbf{u}_i+\mathbf{u}_j\right\|} \cdot \frac{\mathbf{e}_{ij}}{\left\|\mathbf{e}_{ij}\right\|} \right|$ represents how well the direction of the edge agrees with the estimated direction of the ridge-line on the surface (this is the direction of minimum curvature for convex regions, $\mathbf{u}_i$, and the direction of maximum curvature for concave regions, $\mathbf{v}_i$). The values $\overline{\kappa}_{max} = \frac{(\kappa_{max})_i + (\kappa_{max})_j}{2}$ and $\overline{\kappa}_{min} = \frac{(\kappa_{min})_i + (\kappa_{min})_j}{2}$ are approximations for the principal curvatures averaged over the extent of the graph edge $\mathbf{e}_{ij}$.

Our approach for segmenting a surface using these salience functions consists of five steps:

1. Compute a weight for each edge using a prescribed salience function such as equations (2.1),(2.2) or (2.3),
2. Compute the (maximum) spanning tree of the graph based on the weight of edges. The edges not in the spanning tree are called loopy edges (see §2.2.1),
3. Loopy edges are stored in order of decreasing edge weight in an edge stack. Contours are generated by adding loopy edges from this stack to the tree (see §2.2.2),
4. Loopy edges are inserted into the maximum spanning tree. When inserted, each edge creates a new closed contour on the surface. Each time an edge is added, a newly created contour is tested to ensure it satisfies a uniqueness criterion and a curve-length criterion. These criterion suppress contours that are nearly equivalent or have very short arc-length (see §2.2.3),
5. The surface segmentation is computed by finding all contiguous surface regions within the contours created in step (4).

Step (2) makes use of standard textbook algorithms to compute the spanning tree of the surface, e.g., Kruskal's Algorithm, the details are described in §2.2.1. The computational challenge to step (3) is to find the closed surface contour created by adding a cycle to the spanning tree of the graph, this can be efficiently computed using the structure inherent to trees, see §2.2.2 for details. Step (4) serves to eliminate contours that are undesirable for our segmentation goal. In theory, this aspect of the algorithm could enforce arbitrary constraints on the boundary such as its global shape or smoothness. In our case, we implement two constraints for the segmentation boundary contours: (1) each contour must be distinct from other extracted surface contours and (2) each contour must be of sufficient length (at least 5% of the longest extracted curve loop in the segmentation), see §2.2.3 for details

2.2.1 Compute the Spanning Tree of the Graph

Given an undirected graph, a spanning tree of that graph is a subgraph that is a tree, i.e., a collection of connected nodes with no cycle or "loops", that connects all the vertices together. A single graph can have many different spanning trees. The maximum spanning tree is the spanning tree with weight larger than or equal to the weight of every other spanning tree. In this dissertation, the maximum spanning tree of the graph is computed by making use of the method similar to Kruskal's algorithm [52]. The details of the method are as followings:

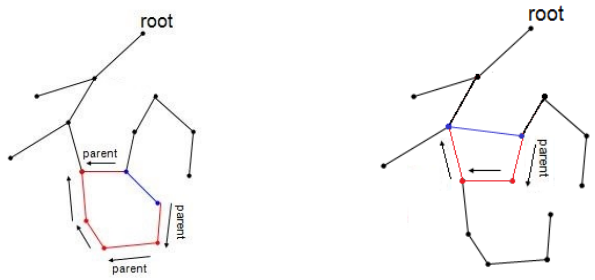
1. Create a forest F (a set of trees). At the initial step, each vertex in the graph is a separate tree.
2. Initialize a set X which contains all the edges in the graph. The edges in the set X are sorted by their weight in decreasing order.
3. While X is nonempty, remove an edge with maximum weight from X, add it to the forest F. If the edge connects two different trees, then add it to the forest and the two trees are merged into one tree. If the edge forms a closed loop in a tree, discard that edge.

At the termination of the algorithm, the forest has only one component, that component is the maximum spanning tree of the graph. After completing Kruskal's algorithm, the maximum spanning tree is known and it contains all the mesh vertices and some of the mesh edges as part of the tree.

2.2.2 Extract Contours from the Tree

Surface contours are created by inserting "loopy" edges into the maximal spanning tree. Each contour is extracted by following the tree ancestry until a common ancestor is found. The tree nodes traversed then constitute the edges and vertices of the surface contour (including the common ancestor). This process is shown graphically in figure 2.4.

Contours are generated by inserting loopy edges that are not in the maximum spanning tree. The sequence of contours generated is determined by the order used to insert these



(a) (b)

Figure 2.4: Extracting contours from the spanning tree: (a,b) show two trees where blue edges are loopy edges inserted into the tree. Inserted loopy edges will connect two leaves of the spanning tree creating a closed loop. The members of the cycle (loop) can be computed by tracing the parents of the two leaf nodes until they meet a common ancestor. The closed loop is represented as a series of edges and vertices shown in red and blue.

loopy edges. This insertion order is determined by sorting the loopy edges in order of decreasing edge weight. The number of loopy edges stored in the stack is controlled by a user-specified parameter λ. The parameter λ is specified as a percentage of the total number of loopy edges used to segment the surface as described in equation (2.4). For example, if there are 100 loopy edges in the mesh and $\lambda = 0.1$ then the stack will contain the loopy edges having the 10 highest weights. Higher values of λ will tend to generate more segments as shown in figure 2.9. Hence, λ approximately controls how many segments will be in the segmentation output.

$$\lambda = \frac{the\ number\ of\ loopy\ edges\ used\ to\ segment\ the\ surface}{total\ number\ of\ loopy\ edges} \tag{2.4}$$

A closed loop on the surface consists of edges and vertices of the surface mesh. Since each closed loop is created uniquely by a single loopy edge, we refer to each closed contour as $c(\mathbf{l}_{ij})$, where $\mathbf{l}_{ij}$ denotes the loopy edge connecting point $\mathbf{p}_i$ to point $\mathbf{p}_j$ on the mesh surface. Each contour is represented as a collection of edges as shown in equation (2.5).

$$c(\mathbf{l}_{ij}) = \left\{ \cup_{\mathbf{e}_{mn} \in c(\mathbf{l}_{ij})} \mathbf{e}_{mn} \right\} \tag{2.5}$$

In equation (2.5), the edge $\mathbf{e}_{mn}$ is one of the edges in the contour $c(\mathbf{l}_{ij})$, the collections of such edges constitute the contour $c(\mathbf{l}_{ij})$.

2.2.3 Enforce Contour Constraints

The ridge walking segmentation algorithm includes a post-processing step to detect and eliminate contours that over-segment the surface, i.e., these contours needlessly split surface regions. This is accomplished by enforcing that each contour in the final solution satisfy two constraints: (1) each contour must be sufficiently long to delineate a significant region and (2) each contour must be significantly distinct from the other contours provided as part of the segmentation solution. Enforcing these constraints greatly improves the quality of the segmentation result.

The first constraint is trivial to implement from knowledge of the points and edges of the contour. Towards this end, a threshold, τ, is defined that requires all contours in the solution to have a length that is at least 5% of the length of the longest contour. The constraint is implemented by computing the length of all contours and discarding those contours having length below the threshold.

The second constraint is implemented by detecting pairs of similar contours and, for each detected pair of contours, performing a test which evaluates the salience of each contour in the pair and, if necessary, deleting the contour from the pair that is not salient. A process is defined which determines the order by which contours are compared and deleted, it is summarized in the following 5 steps:

1. Compute a salience score for each contour.
2. Compute a global salience threshold, $\overline{w}$, from the contour salience scores.
3. Sort the contours in order of decreasing salience in a list $\mathbf{L}$.
4. Take contour $\mathbf{L}_0$ from the list of contours. For each less-salient contour in the solution, $\mathbf{L}_m$, (where $m > 0$), detect if the contour $\mathbf{L}_m$ and contour $\mathbf{L}_0$ are similar. If so, a salience test is performed on the contour pair. The salience test computes the salience of the non-overlapping parts of contours $\mathbf{L}_0$ and $\mathbf{L}_m$. If both non-overlapping contours are salient, neither contour is deleted. If one or both contours are not salient, the contour having lowest salience is deleted.

5. Steps 4 is iteratively executed until the list $\mathbf{L}$ is empty.

Steps (1)-(3) are trivial to implement based on the knowledge of the graph edges. Step (4) describes the order of comparing the contours pairs, and the rule applied to delete similar contour pairs. The details of steps (1), (2) and (4) above are discussed in the following sections.

1. Compute a salience score for each contour

The salience score for a contour, $c(\mathbf{l}_{ij})$, is its average edge weight, it is computed in equation (2.6).

$$\overline{w_{c(\mathbf{l}_{ij})}} = \frac{1}{N_{ij}} \sum_{\mathbf{e}_{mn} \in c(\mathbf{l}_{ij})} w(\mathbf{e}_{mn}) \tag{2.6}$$

In equation (2.6), $w(\mathbf{e}_{mn})$ denotes the edge weight for the edge $\mathbf{e}_{mn}$, and N_{ij} denotes the number of edges in the contour $c(\mathbf{l}_{ij})$.

2. Compute a global salience threshold $\overline{w}$ for the salience scores

The global salience threshold $\overline{w}$ is the average of salience scores for all the contours, it is computed in equation (2.7).

$$\overline{w} = \frac{1}{K} \sum \overline{w_{c(\mathbf{l}_{ij})}} \tag{2.7}$$

In equation (2.7), K denotes the number of contours in the stack.

3. Detect and delete similar contour pairs

A similar contour pair is detected when it satisfies the following two conditions: (1) some portion of the contour pair overlap and (2) the total length of the non-overlapping portions of the contour pair is shorter than the threshold τ. Let $c(\mathbf{l}_{ij})$ denote the surface contour $\mathbf{L}_0$. Let $c(\mathbf{l}_{mn})$ denote another surface contour $\mathbf{L}_m$. The contours $c(\mathbf{l}_{ij})$ and $c(\mathbf{l}_{mn})$ are said to be overlap if for some portion of their extent, they traverse the same path on the mesh. This is detected by testing to see if any of the edges in the contour $c(\mathbf{l}_{ij})$ are also found to be present in the contour $c(\mathbf{l}_{mn})$. Once a pair of contours are detected to overlap, the length of the non-overlapping parts of these contours is computed. If the length is shorter than the threshold τ, the pair of contours are deemed to be similar. Once a pair of contours are detected to be similar, a test is performed on the contour pair ($c(\mathbf{l}_{ij})$, $c(\mathbf{l}_{mn})$)

that has three possible outcomes: (1) both contours are kept and the segmentation result is unchanged, (2) $c(\mathbf{l}_{ij})$ is deleted which effectively merges the regions that this contour subdivides, and (3) $c(\mathbf{l}_{mn})$ is deleted which effectively merges the regions that this contour subdivides. The test is carried out by computing a salience score for the non-overlapping parts of each contour as shown in equation (2.8) and (2.9).

$$\overline{w'_{c(\mathbf{l}_{ij})}} = \frac{1}{K_{ij}} \sum_{\mathbf{e}_{pq} \in c(\mathbf{l}_{ij}), \mathbf{e}_{pq} \notin c(\mathbf{l}_{mn})} w(\mathbf{e}_{pq}) \tag{2.8}$$

$$\overline{w'_{c(\mathbf{l}_{mn})}} = \frac{1}{K_{mn}} \sum_{\mathbf{e}_{pq} \notin c(\mathbf{l}_{ij}), \mathbf{e}_{pq} \in c(\mathbf{l}_{mn})} w(\mathbf{e}_{pq}) \tag{2.9}$$

In equation (2.8) and (2.9), $\omega'_{c(\mathbf{l}_{ij})}$ computes the edge salience feature values by average salience of the edges that in the contour $c(\mathbf{l}_{ij})$ but not in contour $c(\mathbf{l}_{mn})$. $\omega'_{c(\mathbf{l}_{mn})}$ computes the edge salience feature values by average salience of the edges that in the contour $c(\mathbf{l}_{mn})$ but not in contour $c(\mathbf{l}_{ij})$.

Let the variables ($\omega'_{c(\mathbf{l}_{ij})}$, $\omega'_{c(\mathbf{l}_{mn})}$) denote the pair of salience score computed for the contour pair ($c(\mathbf{l}_{ij})$, $c(\mathbf{l}_{mn})$). Using these salience score, a simple threshold-based classifier uses the global salience threshold, $\overline{w}$, to determine which of the three outcomes occurs. If $\omega'_{c(\mathbf{l}_{ij})} > \overline{w}$ and $\omega'_{c(\mathbf{l}_{mn})} > \overline{w}$, then both contours are kept. If $\omega'_{c(\mathbf{l}_{ij})} > \omega'_{c(\mathbf{l}_{mn})}$ and $\omega'_{c(\mathbf{l}_{mn})} < \overline{w}$, then contour $c(\mathbf{l}_{mn})$ is deleted, both contours are kept. If $\omega'_{c(\mathbf{l}_{ij})} < \omega'_{c(\mathbf{l}_{mn})}$ and $\omega'_{c(\mathbf{l}_{ij})} < \overline{w}$, then $c(\mathbf{l}_{ij})$.

Figure 2.5 shows the merging process. Two similar contours on the surface are shown in pink and in cyan. In this case, the pink that divided the thigh into two pieces was merged to the cyan contour. This decision is based on the relative value of average edge salience found in the non-overlapping parts of the competing contours.

2.2.4 Find Contiguous Surface Regions within the Contours

The contours after constraint filtering are not the boundaries for the segmented regions. Simple closed loops have to be extracted so that each loop does not contain other loops. Loop extraction is done by choosing a random point and direction on the merged contour and then traversing the contour such that, at each intersection point one follows ei-

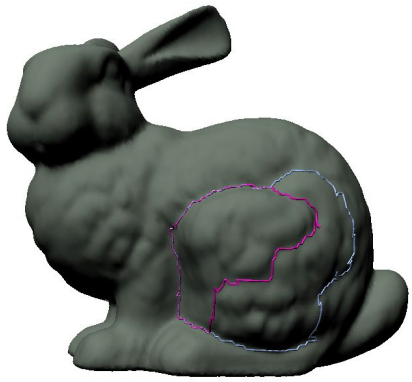


Figure 2.5: Two partially overlapping contours on the surface are shown in pink and in cyan. Boundaries such as this may be merged as part of the post-processing of the segmentation. In this case, the pink contour that divided the thigh into two pieces was discarded in preference for the larger cyan contour. This decision is based on the relative value of edge weights found in the non-overlapping regions of the competing contours.

ther a clockwise or anti-clockwise direction until you arrive at the point originally chosen. This process continues until all edges of the merged contour have been traversed in both the clockwise and anti-clockwise directions. used commonly to automatically solve 2-dimensional mazes. Contiguous surface regions encircled by the extracted simple closed contours are trivially computed given the graph and the boundary values.

2.2.5 The Input Parameter of Ridge Walking Algorithm

This section describes a modification to the ridge walking algorithm to allow the algorithm to take as input the number of desired output segments denoted as an integer, N, rather than the λ parameter discussed in section §2.2.2. The number of segments, N, is a more common parameter for mesh segmentation algorithms and can lead to better segmentation results in many cases. This is true because, for some models, such as the stuffed bear, desk, and coffee mug, a user can often look at the model and know or closely guess the number of segments that they desire. In contrast, the input parameter,λ, only provides approximate control over how many segments are generated (see figure 2.9 for details). As such, it is difficult to know what value for λ is appropriate for a specific model. For example, higher values of λ will tend to generate “more” segments in the resulting segmentation, while

higher values of N will always produce exactly N segments in the resulting segmentation.

When the algorithm takes the number of segments, N, as input parameter, the ridge walking algorithm is implemented in three steps:

1. Convert the value of N into a appropriate value of λ,
2. Segment the 3D model using ridge walking algorithm with λ to generate M segments ($M > N$),
3. Merge the $M - N$ segments into the N largest segments of the initial segmentation.

Step (1) converts the input parameter, N, to the ridge walking algorithm parameter λ automatically. Step (2) applies the ridge walking algorithm using the converted value of λ to generate an over-segmentation including M segments. Figure 2.6(b) shows one such the segmentation result. Step (3) merges segments associated with small regions until the output of the ridge walking algorithm consists of only N regions. Figure 2.6(c) shows the merged segmentation result for $N = 8$. Steps (1) and (3) are discussed in detail in the following sections. Step (2) is discussed in §2.2.1-§2.2.3.

1. Convert the value of N into a appropriate value of λ

In order to convert the value of N into an appropriate value of λ, we have to ensure that the solution using the given value of λ will create at least N segments. To do so, we can immediately choose $\lambda = 1$ which will create as many segments as possible using the ridge walking algorithm. To be more computational efficient, we actually choose the value of λ less than 1. The value of λ we choose is the value that assume all of the edges in the contours of interest will have the weights that are greater or equal than 0. The loopy edge in the contour of interest must be at least in flat location or the location that satisfies the salience function to give it a positive value. In this context, the λ can be computed automatically by equation (2.10).

$$\lambda = \frac{loopy\ edges\ having\ weight \geq 0}{total\ number\ of\ loopy\ edges} \tag{2.10}$$

2. Merge the over-segmented regions

The ridge walking algorithm will over-segment the surface due to the large value of

λ is used to segment the surface. For example, the stuffed bear model in figure 2.6(b) is over-segmented using the λ generated by equation (2.10). A post-processing step is used to merge over-segmented regions generated by the ridge walking algorithm. Suppose the ridge walking algorithm will generated M segments using the λ computed by equation (2.10). Our goal is to merge the $M-N$ extra over-segmented regions. To do that, N largest segments are selected and $M-N$ small regions are put in a list $\mathbf{S}$. The process of merging small over-segmented regions to N largest regions is describe in the following 3 steps:

1. Take region $\mathbf{S}_0$ from the list of $\mathbf{S}$, compute its neighboring regions.
2. Computational decision is made for merging this region $\mathbf{S}_0$ to its neighboring regions, $\mathbf{S}_0$ is deleted from the list $\mathbf{S}$.
3. Steps (1) and (2) are iteratively executed until the list $\mathbf{S}$ is empty.

Step (1) computes neighboring regions for the region $\mathbf{S}_0$, the neighboring regions are regions which share portions of their contours with the contour of region $\mathbf{S}_0$. Step (2) describes the criteria to merge the region $\mathbf{S}_0$ to one of its neighbors, the details of this step is descries in the following section.

The computational decisions for merging a small region to one of its adjacent regions are described using some mathematical notation. Let $c(\mathbf{S}_0)$ be the contour of the region $\mathbf{S}_0$, R_i be the one of the adjacent regions of the segment $\mathbf{S}_0$, and $c(R_i)$ be the contour of segment R_i. Contours $c(\mathbf{S}_0)$ and $c(R_i)$ are not contained by each other, but they share some common edges. The over-segmented region $\mathbf{S}_0$ has to be merged to one of its touching regions, R_i. For each region, R_i, the average weight for the edges belonging to both contours $c(R_i)$ and $c(\mathbf{S}_0)$ is defined in equation 2.11.

$$\bar{w_{c(R_i)}} = \frac{1}{N_i} \sum_{\mathbf{e}_{jk} \in c(R_i), \mathbf{e}_{jk} \in c(\mathbf{S}_0)} w(\mathbf{e}_{jk}) \tag{2.11}$$

In equation 2.11, $w(\mathbf{e}_{jk})$ is the weight for edge $\mathbf{e}_{ij}$, N_i is the number of shared edges for contours $c(R_i)$ and $c(\mathbf{S}_0)$. The segment S^c is merged into its adjacent region, R_x, where $x = \max_i \bar{w_{c(R_i)}}$. The merged region will contain both region $\mathbf{S}_0$and R_x. The region R_x can be either one of N largest regions or the other small regions in the list $\mathbf{S}$. New contour rep-

(a) (b) (c)

Figure 2.6: Change the input parameter for ridge walking algorithm. (a) shows merge criterion, small region C is merged to larger region A or B, based on the average shared edge weight, (b,c) show segmentation results for stuffed bears before and after region merge step respectively.

resents the boundary for the merged region, $\mathbf{S}_0 \cup R_x$, will be computed. Using the method described in §2.2.4, contour pair $c(R_x)$ and $c(\mathbf{S}_0)$ will generate 3 contours, the contour for the merged region, $\mathbf{S}_0 \cup R_x$, is the contour with the longest length.

Figure 2.6 shows this merging process graphically. Figure 2.6(a) shows the small segment S^c will be merged to one of its neighboring segments: S^A or S^B, figure 2.6(b) shows the segments for stuffed bear models before merging process, many small segments are created due to large value of λ computed automatically by equation 2.10, figure 2.6(c) shows the segmentation result after merging small regions.

2.3 Results

Figure 2.7 shows segmentation results for three different surfaces using the three different salience functions: $w_{ridge}(\mathbf{e}_{ij})$ (equation (2.1), left column), $w_{valley}(\mathbf{e}_{ij})$ (equation (2.2), middle column) and $w_{curv}(\mathbf{e}_{ij})$ (equation (2.3), right column). The first two surfaces are 3D models of a bunny and a horse. These models are commonly used for presenting 3D surface processing results. The third surface is a laser scan of a bone fragment from a replica of a human tibia. For the bunny model, the minima rule seems to work well for segmentation, i.e., the segmentation in both (b) and (c) seem perceptually satisfactory whereas (a) is not. For the horse model, segmentation results (d) and (f) seem perceptually satisfactory whereas (e) is not. For the bone model, segmentation results (g) and (i) seem satisfactory

whereas (h) is not. Interestingly, the bone fragment represents a somewhat non-standard surface since its surface is convex nearly everywhere. Hence, satisfactory segmentation of that surface relies on extraction of convex ridges where the opposite is true for the bunny model which is convex almost everywhere, i.e., sphere-like, and whose shape details are almost exclusively offered by the concavities carved into this surface. Solutions offered by the salience function $w_{curv}(\mathbf{e}_{ij})$ tend to strike a compromise between these two extremes which may be desirable especially when the input surface is completely generic, i.e., when one is attempting to partition generic objects which may encode their semantic information via the traditional minima rule as is the case for figure 2.7(a-c) and (d-f) or via convex ridges as is the case for figure 2.7(g-i).

Figure 2.8 provides views of two segmentations of the stuffed bear model. Figure 2.8(a,b) are views of a segmentation using $w_{ridge}(e_{ij})$(equation (2.1)) and Figure 2.8(c,d,e) are views of a segmentation using the $w_{curv}(e_{ij})$ (equation (2.2)). For animal models such as these there is limited information in the convex ridges of the surface. Semantic information for the stuffed bear includes the front/back of the ears and the ends of the limbs. The mixed model captures some structures delimited by concave ridges (ears, limbs, head) as well as structures delimited by convex ridges. However some may consider other structures to be over-segmented such as the cylindrical portion of the limbs which are divided into two parts along a convex ridge running length-wise down each limb.

For the surfaces with an unknown number of parts, a user-input parameter, λ, is specified which controls, as a percentage, the total number of loopy edges used to segment the model. This indirectly determines the segmentation resolution for the input surface. Higher values of λ generate more detailed segmentations as shown in figure 2.9.

For the surfaces with a known number of segments, a user-input parameter, N, is specified which is the number of segments the user wants to generate. Higher values of N generate more detailed segmentations as shown in figure 2.10.

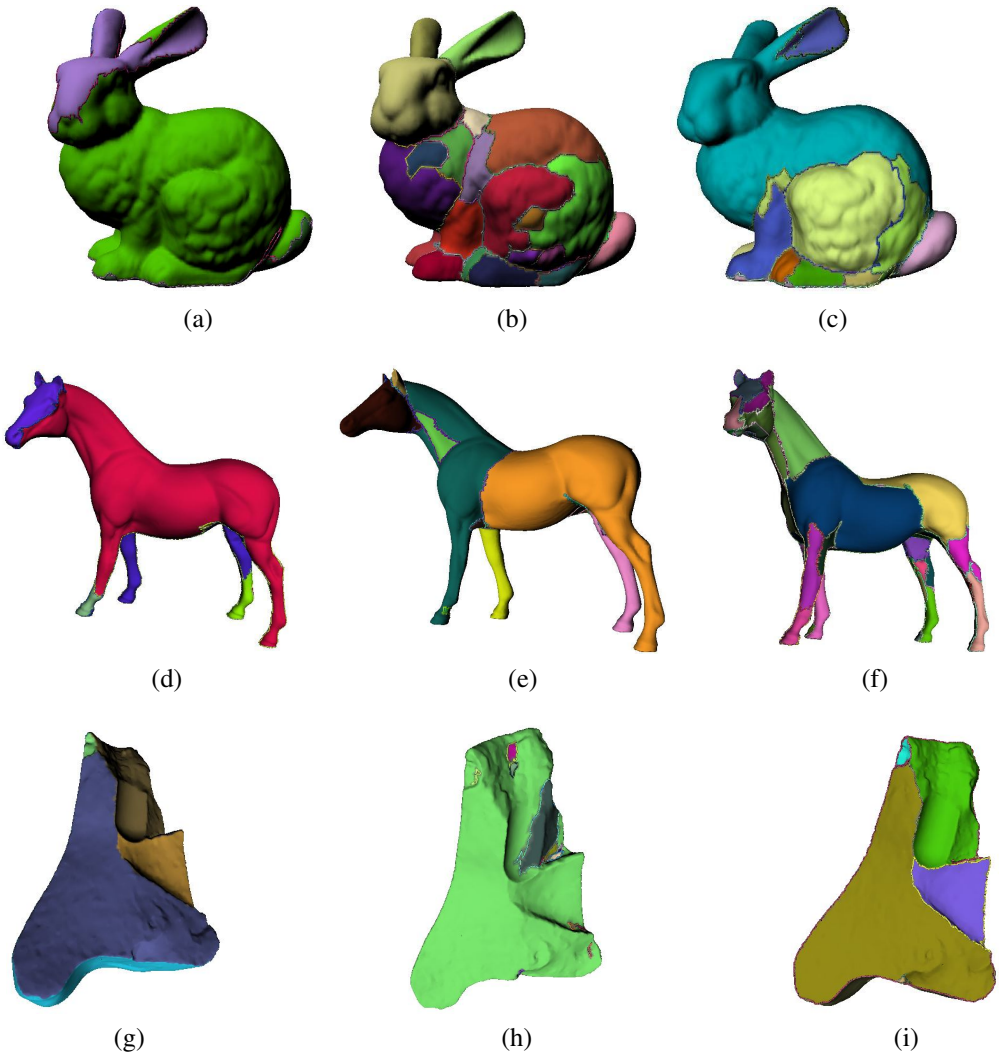
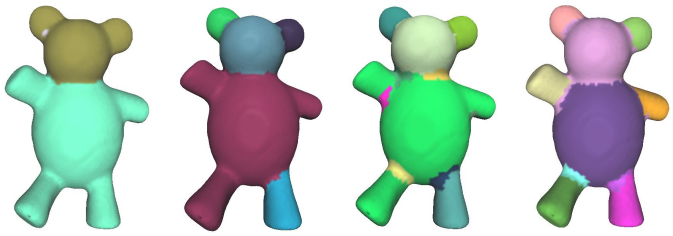


Figure 2.7: Results are shown that segment three different surfaces using three discussed salience criteria. (left column) segmentation by convex ridges, (middle column) segmentation by concave valleys, (right column) segmentation by ridges and valleys. Different salience functions as defined by equations (2.1) (2.2) and (2.3) generate optimization criterion for the segmentation that can be made sensitive to specific surface sub-structures which can generate acceptable results as in (a,c,d,f,g,i) and also poor results as in (b,e,h). A mixed segmentation model offered by equation (2.3), seems to be most reliable for the surfaces shown here.

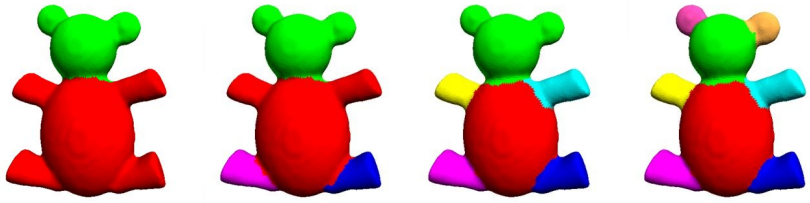
(a) (b) (c) (d) (e)

Figure 2.8: (a,b) are two views of a convex-ridge segmentation of the stuffed bear dataset given by the $w_{ridge}(\mathbf{e}_{ij})$ salience function ($\lambda = 0.4$). For this model, the semantic information offered by the convex ridges include the front/back of the ears and the ends of the limbs. (c,d,e) are three views of the mixed ridge/valley segmentation of the stuffed bear dataset given by the $w_{curv}(\mathbf{e}_{ij})$ salience function ($\lambda = 0.2$). The mixed model captures some structures delimited by concave ridges (ears, limbs, head) as well as structures delimited by convex ridges.



(a) $\lambda = 0.1$ (b) $\lambda = 0.2$ (c) $\lambda = 0.3$ (d) $\lambda = 0.4$

Figure 2.9: Results for our proposed algorithm for increasing values of the parameter λ: (a) $\lambda = 0.1$, (b) $\lambda = 0.2$, (c) $\lambda = 0.3$, (d) $\lambda = 0.4$. Higher values for λ generate more detailed segmentations of the surface.



(a) N=2 (b) N=4 (c) N=6 (d) N=8

Figure 2.10: Results for our proposed algorithm for increasing values of the parameter N: (a) $N = 2$, (b) $N = 4$, (c) $N = 6$, (d) $N = 8$. Higher values for N generate more detailed segmentations of the surface.

2.4 Conclusion

This chapter describes a new surface segmentation algorithm that parametrizes a surface in terms of a ridge-tree which is a spanning tree computed from curvature data estimated at the model vertices. Different salience functions are used to make the segmentation approach sensitive to specific kinds of surface sub-structures which can help effectively segment difficult or unusual surfaces such as the bone fragment shown in figure 2.7(g,h,i). As with other segmentation algorithms, the output does not always produce geometrically meaningful parts as separate components. A mixed segmentation model using both convex and concave contours was proposed for segmentation of generic, i.e., non-anthropomorphic, models which can provide desirable results for surfaces that may be difficult to segment by straight-forward application of the minima rule. The approach also motivates discussion about generic surface segmentation where visual perception and concave ridges may not necessarily provide the semantic information needed for a "good" segmentation.

CHAPTER 3: PERFORMANCE EVALUATION OF RIDGE WALKING ALGORITHM

This chapter describes the methods used to quantitatively evaluate the ridge walking algorithm and a detailed analysis of a select subset of segmentation results to better understand the strengths and weaknesses of this algorithm. The quantitative evaluation is performed by comparing results from a collection of segmentation algorithms, including results from the ridge walking segmentation, with results established as the ground truth. Analysis proceeds by studying the geometric properties of the segmentation boundaries and regions for a select group of interesting models. The 3D segmentation evaluation techniques described in [19] will be used to evaluate our 3D segmentation approach and these results will enable us to compare our approach with other leading methods.

The data from [20] is used as the data for segmentation performance analysis. This dataset consists of 380 mesh models which are further broken down into 19 different shape categories. These categories include (1) humans, (2) cups, (3) glasses, (4) airplanes, (5) ants, (6) chairs, (7) octopus, (8) tables, (9) stuffed bears, (10) human hands, (11) pliers, (12) fishes, (13) birds, (14) armadillos, (15) busts, (16) mechanical parts, (17) bearings, (18) vases, and (19) four-legged animals. In some categories (armadillos and pliers), the same model appears in different poses. In other categories (humans, ants, octopus, stuffed bears, human hands, birds, four legged animals), different models of the same type may occur and they may also have different poses. In the third collection of categories (cups, vases, and chairs), the models may have different genus. Example models from each of the categories in the dataset are shown in each row of figure 3.1.

The proposed segmentation algorithm evaluation approach requires a definition of a ground truth segmentation for each model. Work in [19] established ground truth segmen-

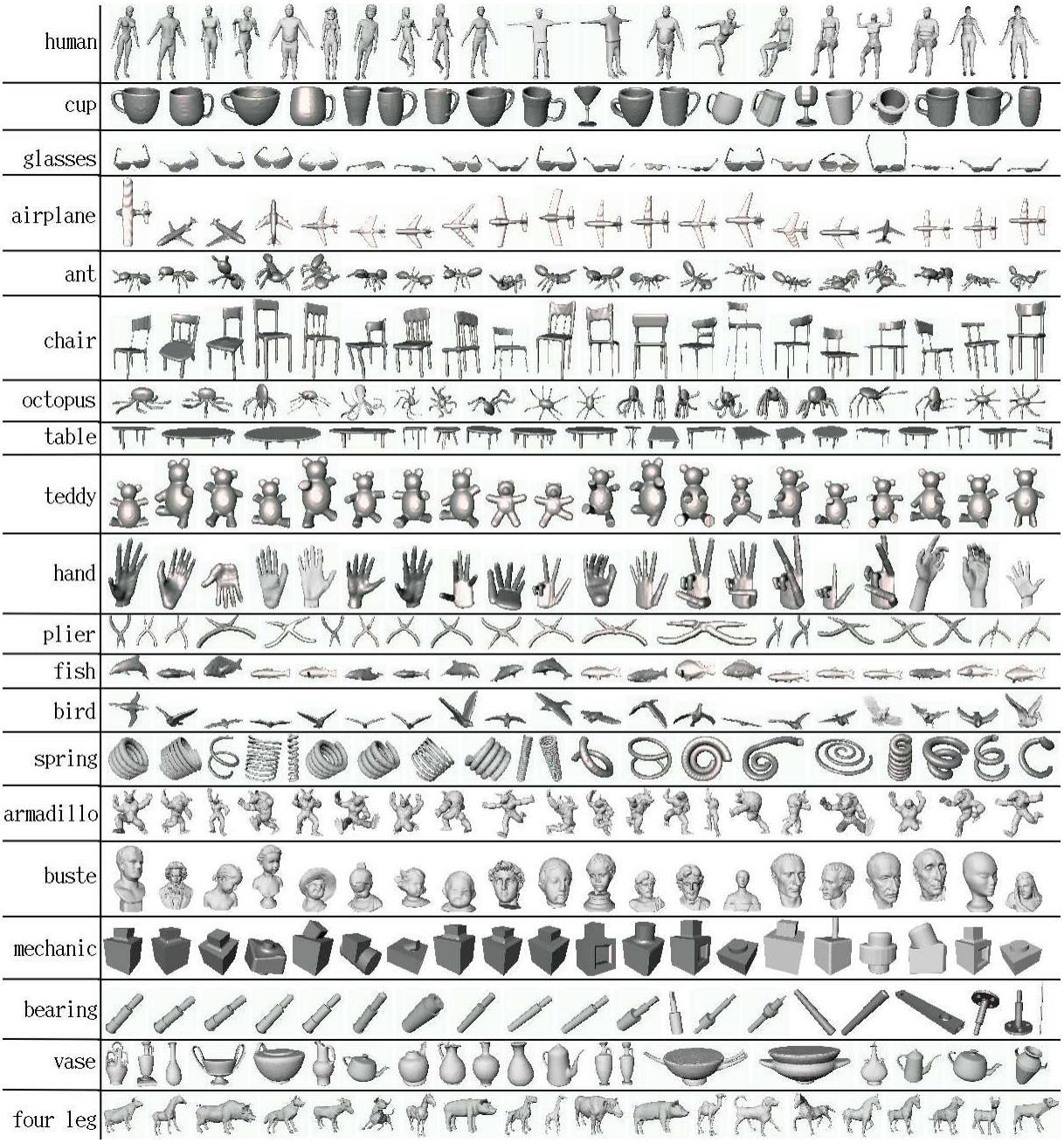


Figure 3.1: Representatives of 3D surfaces from [20]

tations for the 380 models of the dataset by employing humans to manually segment each model using Amazon's Mechanical Turk. Amazon's Mechanical Turk is an on-line platform, where the requesters can post tasks to the website. Workers from around the world can contract to complete the requested tasks and submit their results for the required tasks to the Mechanical Turk system online. The authors of [19] requested that workers segment these models using a manual segmentation tool that they provided. They also provide an accompanying set of instructions on how to use the segmentation tool to segment the models. Segmentations from Amazon's Mechnical Turk were reviewed independently to ensure they were valid and bad segmentations such as models without any segments were rejected. The total number of valid segmentations produced via this method was 4300. An average of 11 segmentations for each model were marked as valid. The set of valid human generated segmentations were taken as a collection of "ground truth" examples. Evaluation is accomplished by comparing the segmentations produced by automatic algorithms with each of the "ground truth" examples.

A set of four metrics serve to score the difference between two segmentations of a given surface. These metrics measure the difference between two segmentations as a function of the segmented region boundaries or the contents of the segmented regions. The metrics are applied by choosing a model and ground truth segmentation of that model and comparing segmentations generated by automatic algorithms (including ridge walking) with the ground truth segmentation. Using this comparison strategy, the ridge walking algorithm was ranked with respect to other leading segmentation algorithms using the same metrics.

3.1 Methodology for Evaluation

The evaluation for the proposed ridge walking algorithm uses four evaluation metrics taken from [19] to compare segmentations generated with the ground truth examples. One of these metrics measures the segmentation difference by comparing the boundaries of the two segmentations using a boundary difference score. The other three metrics measure the segmentation difference by comparing regions of two segmentations using three distinct

approaches for measuring the difference between two regions. Rather than selecting one of these metrics, we present results for all four to develop insights on the strengths and weaknesses of the ridge walking algorithm which are analyzed in detail afterwards.

3.1.1 Experimental Setup for Comparative Evaluations

Comparative evaluation of segmentation algorithms is difficult because the algorithms themselves often have different inputs and the outputs they generate may have different formats. For example, each algorithm will often have completely different input parameters and the values of the input parameters that generate a "good" segmentation are often different for each 3D model. The experimental approach seeks to use input parameters for each algorithm that generate the best possible segmentation results for each model. By constructing the experiment in this way, we seek to eliminate any bias in the results that may come from how the specific values of the input parameters were chosen.

Segmentation algorithms were divided into two classes. Class 1 algorithms were defined as those algorithms that require a set of input parameters to automatically determine the number of segments. Class 2 algorithms were defined as those algorithms that require the number of segments to be specified with a set of input parameters. The algorithms belonging to class 1 were given the input parameter settings recommended by the authors for all of the models in the dataset. The algorithms belonging to class 2 were given a different set of input parameters for each model, which included the correct value for the number of segments parameter as appropriate for the number of segments identified in each ground truth example.

The ridge walking algorithm has two modes of operations. Each mode has a different input parameter. The first mode takes λ as an input parameter as described in §2.4. The second mode takes the number of segments, N, as input parameter as described in §2.2.5. For the evaluation, the ridge walking algorithm uses the second mode. In this mode, the ridge walking algorithm is a class 2 segmentation algorithm, i.e., it takes a set of different of number of segments as input for each model.

As mentioned in §2.2, the ridge walking algorithm can apply one of three different criteria to segment a 3D surfaces: (1) to divide the surface along convex ridges of the surface, (2) to divide the surface along concave valleys of the surface, or (3) to divide the surface along both convex ridges and concave valleys of the surface. In the literature, most segmentation algorithms use the minima rule to try to mimic how humans segment shapes as discussed in §1.2.1. To compare fairly with other segmentation algorithms, the second criteria was used to segment surfaces with the ridge walking algorithm which is similar to the minima rule as it divides surfaces along concave contours.

3.1.2 Compute Evaluation Metrics

The performance analysis uses four metrics originally proposed in [19] to evaluate how well the ridge walking algorithm performs relative to the ground truth examples. Each metric can be seen as an adaptation of a similar metric used to evaluate the performance of image segmentation algorithms. All the metrics suffer from two shortcomings, the values for each metric can be unstable when either segmentation contains too few or too many segments. There are two extreme situations in the segmentation results: (1) the segmentation algorithm segments the every vertex of the mesh surface into a different segment, (2) the segmentation algorithm segments the mesh surface into a single segment. In these two circumstances, the values for each metric will not present the true quality of the results. A detailed description of each metric is provided in the following sections.

1. Cut Discrepancy

The cut discrepancy is a boundary-based metric that measures the difference between two segmentation results by comparing their segmentation boundaries. Segmentation boundaries are compared by measuring the curve-to-curve distance from the set of points on one segmentation boundary to the set of points on the second segmentation boundary using a measure originally proposed in [53]. Assume a surface is segmented by two algorithms giving segmentations S_1 and S_2. Let B_1 and B_2 denote the segmentation boundary for S_1 and S_2 respectively. The cut discrepancy measures the distance between the segmentation

boundaries B_1 and B_2. Suppose two points from these boundaries $\mathbf{p}_1 \in B_1$ and $\mathbf{p}_2 \in B_2$ are selected, then the distance from point $\mathbf{p}_1$ to the segmentation boundaries B_2 can be computed by finding the minimum distance between the point $\mathbf{p}_1$ and the segmentation boundary of the second object, B_2, as shown in equation (3.1).

$$d_G(\mathbf{p}_1, B_2) = \min_{\forall p_2 \in B_2} d_G(\mathbf{p}_1, \mathbf{p}_2) \tag{3.1}$$

The directional cut discrepancy from S_1 to S_2, denoted as $DCD(S_1 \Rightarrow S_2)$, is the mean of these point-to-boundary distances, $d_G(\mathbf{p}_1, B_2)$, taken over all the points in the segmentation boundary of the first object B_1 as shown in equation (3.2), where N is the number of points in B_1.

$$DCD(S_1 \Rightarrow S_2) = \frac{1}{N} \sum_{\forall p_1 \in B_1} d_G(\mathbf{p}_1, B_2) \tag{3.2}$$

Note that the *DCD* metric is asymmetric. To eliminate this asymmetry, the cut discrepancy is taken as the sum of the two directional cut discrepancy functions as shown in equation (3.3). The result is divided by the average Euclidean distance between a point on the surface to the centroid of the mesh, which is denoted as *avgRadius*.

$$CD(S_1, S_2) = \frac{DCD(S_1 \Rightarrow S_2) + DCD(S_2 \Rightarrow S_1)}{avgRadius} \tag{3.3}$$

The cut discrepancy is divided by *avgRadius* to make the metric value invariant to scale transformations of the model data. Low values of the cut discrepancy metric indicate that the boundaries of the two segmentations are similar.

2. Hamming Distance

The Hamming distance is a region-based metric that measures the difference between two regions in terms of the shared point membership of the two regions [53]. Let S be the set of all polygons in the mesh and let one generated segmentation be $S_1 = \{S_1^1, S_1^2, ..., S_1^m\}$, and a second generated segmentation be $S_2 = \{S_2^1, S_2^2, ..., S_2^n\}$, where S_1^i denotes the i^{th} segmented region from segmentation S_1, and S_2^i is the i^{th} segmented region for S_2. Using this notation, the directional Hamming distance $D_H(S_1 \Rightarrow S_2)$ is defined in equations (3.4)

and (3.5).

$$D_H(S_1 \Rightarrow S_2) = \sum_i area(S_2^i \setminus S_1^{i_t}) \tag{3.4}$$

$$i_t = max_k area(S_2^i \cap S_1^k) \tag{3.5}$$

In equations (3.4) and (3.5), "\" denotes the set difference operator, and $area(x)$ denotes the total area of all faces in the set x. The variable i_t in equation (3.5) determines the index of the region from the second segmentation that has largest overlap with region i from segmentation S_1. $D_H(S_1 \Rightarrow S_2)$ denotes to the sum of the areas of the non-overlapping part of segments in S_1 found to correspond to some segment in S_2. For the same reason, $D_H(S_2 \Rightarrow S_1)$ denotes the sum of the areas of the non-overlapping part of segments in S_2 found to correspond to some segment in S_1.

Equation (3.6) and (3.7) incorporate a normalization factor to the metric by dividing the total of the non-overlapping surface area by the total area of all faces on the surfaces denoted as $area(S)$. R_{m1} and R_{m2} are denoted as the "missing area" rate. The Hamming distance between two segmentations $HD(S_1,S_2)$ is the average of the R_{m1} and R_{m2}, as shown in equation (3.8).

$$R_{m2}(S_1,S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{area(S)} \tag{3.6}$$

$$R_{m1}(S_1,S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{area(S)} \tag{3.7}$$

$$HD(S_1,S_2) = \frac{1}{2}(R_{m1}(S_1,S_2) + R_{m2}(S_1,S_2)) \tag{3.8}$$

The weakness of the Hamming distance is that it relies upon finding correct correspondences between segments. Hence, the proposed metric only provides a meaningful score when these correspondences are "correct".

3. Rand Index

The Rand index is a region-based metric that measures the likelihood that a pair of faces are either in the same segment in two segmentations, or in different segments in both segmentations [54].

To specify how to compute the Rand index for two segmentations, we will need to define some mathematical notation. Let S_1 and S_2 denote two segmentations and let s_1^i and

s_2^i denote the segment IDs of face i in S_1 and S_2 respectively. Let M denote the number of faces in the model. Let $C_{ij} = 1$ iff $s_1^i = s_1^j$, which means the face pair (i, j) are in the same segment in segmentation S_1, and let $C_{ij} = 0$ if the face pair (i, j) are in the different segments in segmentation S_1. Let $P_{ij} = 1$ iff $s_2^i = s_2^j$, which means the face pair (i, j) are in the same segment in segmentation S_2, and let $P_{ij} = 0$ means the face pair (i, j) are in the different segments in segmentation S_2. With these definitions, the product $C_{ij}P_{ij} = 1$ will only occur when the face pair (i, j) are in the same segment in both segmentations S_1 and S_2. Likewise, the product $(1-C_{ij})(1-P_{ij}) = 1$ will only occur when the face pairs (i, j) are in different segments in both segmentation S_1 and S_2. The Rand index is defined in equation (3.9).

$$RI(S_1, S_2) = \begin{pmatrix} 2 \\ M \end{pmatrix}^{-1} \sum_{i,j,i<j} \left[C_{ij}P_{ij} + (1-C_{ij})(1-P_{ij})\right] \tag{3.9}$$

In equation (3.9), $\sum_{i,j,i<j} C_{ij}P_{ij}$ counts the number of times the face pair (i, j) lies in the same regions in both segmentation S_1 and S_2. $\sum_{i,j,i<j}(1-C_{ij})(1-P_{ij})$ counts the number of times the face pair (i, j) lies in the different regions in both segmentation S_1 and S_2. $\sum_{i,j,i<j}(C_{ij}P_{ij} + (1-C_{ij})(1-P_{ij}))$ counts the number of times that the two segmentations agree that a given face pair (i, j) should have either different labels or the same labels for the face pair (i, j). $\begin{pmatrix} 2 \\ M \end{pmatrix}$ denotes the number of face pairs. Then $RI(S_1, S_2)$ denotes the likelihood that the labels for pairs of faces have been assigned consistently for the regions defined by the two segmentations S_1 and S_2. To be consistent with the other metrics (the lower the value, the better the automatic segmentation results are), the paper [19] uses $1 - RI(S_1, S_2)$ to report dissimilarities rather than similarities.

The Rand index does not need to compute correspondences between segmented regions or the actual overlap area between corresponding segments. Because of this, it is more computationally efficient to compute this metric than the Hamming distance metric.

4. Consistency Error

The consistency error is a region-based hierarchical similarity metric that scores the

similarities between two segmentations. This metric is motivated by the observed tendency for humans to impose a hierarchical decomposition of an object as a collection of incrementally smaller parts. It is designed so that it will not penalize when a segment in S_1 is divided into 10 segments in S_2.

To specify how to compute the consistency error for two segmentations, we will need to define some mathematical notation. Assume S_1 and S_2 are two segmentations. Let f_i be a mesh face, "\" be the set difference operator, and $area(x)$ represent the surface area of the faces in the set x. Let $R(S, f_i)$ be the segment in segmentation S that contains face f_i. Using these definitions, equation (3.10) specifies the local refinement error. The refinement error measures the "missing area" rate for corresponding segments in two segmentations. Here, the corresponding segments are segments from two segmentations which contain the same face index. The refinement error is an asymmetric metric.

$$E(S_1, S_2, f_i) = \frac{area(R(S_1, f_i) \setminus R(S_2, f_i))}{area(R(S_1, f_i))} \tag{3.10}$$

Two metrics are defined for the entire 3D mesh using the refinement error: (1) Global Consistency Error, $GCE(S_1, S_2)$, as shown in equation (3.11), (2) Local Consistency Error, $LCE(S_1, S_2)$, as shown in equation (3.12). where, n is the number of faces in the polygonal model.

$$GCE(S_1, S_2) = \frac{1}{n} \min \left\{ \sum_i E(S_1, S_2, f_i), \sum_i E(S_2, S_1, f_i) \right\} \tag{3.11}$$

$$LCE(S_1, S_2) = \frac{1}{n} \sum_i \min \{E(S_1, S_2, f_i), E(S_2, S_1, f_i)\} \tag{3.12}$$

Both GCE and LCE are symmetric, and $GCE(S_1, S_2) \geqslant LCE(S_1, S_2)$.

3.1.3 Scores Used for Analysis

Collectively, there are eight different meaningful scores that are proposed to measure the difference between two segmentations. Each score must be merged across multiple segmentations to provide a value that summarizes the score of the group. To merge scores, the group score is taken as the arithmetic mean of the scores for each of the different segmentations. The scores computed for analysis are as follows:

1. The Cut Discrepancy, CD, as in equation (3.3),
2. The Hamming Distance, HD, as in equation (3.8), and the missing rates, $R_{m1}$and R_{m2}, as in equation (3.6) and equation (3.7),
3. The Rand Index, RI, as in equation (3.9),
4. The Global Consistency Error, GCE, as in equation (3.11) and the Local Consistency Error, LCE, as in equation (3.12).

The details of each score are discussed in §3.1.2. Comparative evaluations are accomplished by observing these scores for the different segmentation approaches.

3.1.4 Ranking Algorithms by their Rand Index Score

The Rand index score for the eight automatic segmentation approaches are shown in figure 3.13. The scores were computed using scripts made available by the authors of [19]. The eight segmentation algorithms are: (1) K-means [12], (2) random walks [30], (3) fitting primitives [40], (4) normalized cuts [5], (5) randomized cuts [42], (6) core extraction [45], (7) shape diameter function [4] and (8) ridge walking. The Rand index score for each of these segmentation algorithm was used to rank the algorithms where the lowest scoring algorithm was assigned the lowest rank. Table 3.1 shows the ranked results. The entries of this table contain the rank of the algorithms according the Rand index evaluation metric, where 1 denotes the best and 8 denotes the worst. From the table, we see that no one segmentation algorithm is best for every category. We can also see that the ridge walking algorithm performs well on the cup, airplane, ant, stuffed bear, pliers, fish, bust, CAD, and vase models, however, it does not perform well on human, chair, bird and four-legged animal models.

3.2 Analysis

Analysis focuses on answering several important questions motivated by observations from the ranking results shown in table 3.1. The general questions posed are as follows:

1. Why does the ridge walking algorithm perform well on the stuffed bear category but poorly on the human category?

Table 3.1: Ranking of different segmentation algorithms for each category when evaluated against the "ground truth" examples using Rand index metric (1 is the best, and 8 is the worst).

object category	K-means	Random Walks	Fitting primitives	Normalized Cuts	Randomized Cuts	Core Extraction	Shape Diameter	Ridge Walking
human	4	6	3	2	1	7	5	8
cups	8	5	7	3	2	4	6	1
glasses	3	8	6	2	1	7	5	4
airplanes	6	7	4	5	2	8	1	3
ants	8	6	7	4	2	5	1	3
chairs	8	3	7	1	4	5	2	6
octopus	6	5	8	3	4	2	1	7
tables	7	2	3	1	8	6	4	5
stuffed bears	8	6	7	5	1	4	3	2
human hands	2	6	7	3	1	4	8	5
pliers	7	6	4	5	3	2	8	1
fishes	7	5	8	6	4	2	1	3
birds	5	8	6	4	1	3	2	7
armadillo	7	5	2	6	3	8	1	4
busts	8	3	5	7	1	6	4	2
CAD parts	8	3	6	2	5	7	4	1
bearings	7	6	4	3	2	8	1	5
vases	8	6	7	4	1	3	5	2
four legged	5	7	3	6	2	4	1	8
overall	8	7	5	3	2	6	4	1

2. Why does the ridge walking algorithm outperform other methods for the cup category?
3. Why does the ridge walking algorithm typically out-perform the analyzed competing segmentation methods?

The stuffed bear and human models may be categorized generically as 4-limbed animal shapes. Limbs of these animals can move giving the model different gestures. The ridge walking algorithm performs well on the stuffed bear models, but performs poorly on human models. To better understand this unexpected result, analysis focuses on the stuffed bear and human models. Cups are man-made models. The most significant variation for this model category is the presence of a "handle". Topologically, models having a different number of handles have different genus. Since the ridge walking algorithm performs the best on the cup models, analysis also focuses on segmentation results for this model category. After analyzing these specific interesting cases, an analysis of the global results is provided.

3.2.1 Geometric Attributes for Segmentation Analysis

Analysis is accomplished by comparing the geometric attributes of the segmentation boundaries and regions of the ground truth segmentations with the geometric attributes of the segmentation boundaries and regions of the ridge walking algorithm and the seven other analyzed segmentation algorithms. While the four evaluation metrics described in §3.1.2 can be used to rank each algorithm, what perhaps is more important for analysis is to understand why each algorithm works well or poorly on a specific model or category.

The following geometric properties are computed for each segmentation result: (1) the histogram of minimum principal curvatures on the segmentation boundary, (2) the histogram of minimum principal curvatures in the segmentation regions, (3) the histogram of normalized segment boundary length, (4) the histogram of normalized segment surface area. Values for these properties are computed for each automatic segmentation result are compared with similar values for the "ground truth" examples to analyze each approach.

Histograms provide a compact way to represent the statistical distribution of a collection of values. The analysis approach computes histograms of geometric properties associated with points that come from the segmentation boundaries or from inside segmentation regions. Histograms plot these values as a collection of bins on the x-axis, i.e., intervals defined on the x-axis. The histogram counts the relative frequency of occurrence for the values by counting the number of values that fall in each bin. Visual inspection of the histograms allow one to understand trends in the data that may be difficult to detect in other representations. Analysis proceeds by computing the histogram of a geometric property for the ground truth segmentations and observing the shape of the resulting distribution as a characteristic pattern. A similar process is applied for each automatic segmentation algorithm. Inferences regarding the segmentation results can be made by looking for similarities and differences in the shapes of the histograms of the ground truth and the histograms of the automatic segmentation algorithms. If the shapes of these histograms are similar, we infer that the automatic algorithm segments the object in a way similar to the ground truth segmentations. The details regarding each geometric property is discussed in the following sections.

1. Histogram of minimum principal curvatures from the segmentation boundary

The histogram of minimum principal curvatures for the boundary points indicates the shape of the surface where it has been divided, i.e., how concave is the surface at the locations chosen as segmentation boundaries.

2. Histogram of minimum principal curvatures from the segmentation regions

The histogram of minimum principal curvatures for the segmented region points indicates the shape of the surface in the segmented region, i.e., how convex or how flat is the surface at the locations chosen as segmented regions.

3. Histogram of normalized segment boundary length

The histogram of normalized segment boundary length indicates the length of segment boundaries, i.e., whether the segment boundaries are too short or too long to segment the

surface. The lengths of segment boundaries are normalized by dividing the length of each boundary by the average boundary length of all the segments to make the value invariant to scale transformations of the model data.

4. Histogram of normalized segment surface area

The histogram of normalized segment surface area indicates the size of segment regions, i.e., whether the segment areas are too small or large. The region areas are normalized by dividing the area of each region by the average area for all the regions to make the value invariant to scale transformations of the model data.

3.2.2 Evaluation and Analysis for Stuffed Bear Models

The stuffed bear model category includes 20 models of a stuffed bear in different poses. Figure 3.2 shows segmentation results for 6 of these models using 8 different segmentation algorithms and includes examples 6 ground truth segmentations in the first row. The ridge walking algorithm segmentation results are shown the last row of figure 3.2. The results shown suggest that the ridge walking algorithm can consistently segment the stuffed bear models into arms, legs, ears, and head regions for these poses.

Figure 3.3 shows a bar chart of the evaluation scores for the 9 segmentation methods investigated for each of stuffed bear model. Each bar chart shows the values of different important scores for each evaluation metric. In general, one can interpret the quality of the segmentation as "good" for small bars and "bad" for large bars. Figure 3.3(a) shows scores for cut discrepancy metric. Here, the ridge walking segmentation score is close to the score for the ground truth examples. This indicates that the positions of the segmentation boundaries generated by the ridge walking algorithm are close to the boundaries of the ground truth. Figure 3.3(b,c,d) show scores for the Hamming distance, the Rand index, and the consistency error respectively. Here, the scores for ridge walking algorithm are similar to the scores for the ground truth examples. This indicates that the regions computed by the ridge walking algorithm are similar to the regions of the ground truth. Figure 3.3(b) shows a small difference between the scores R_{m1} and R_{m2} for the ridge walking algorithm. This

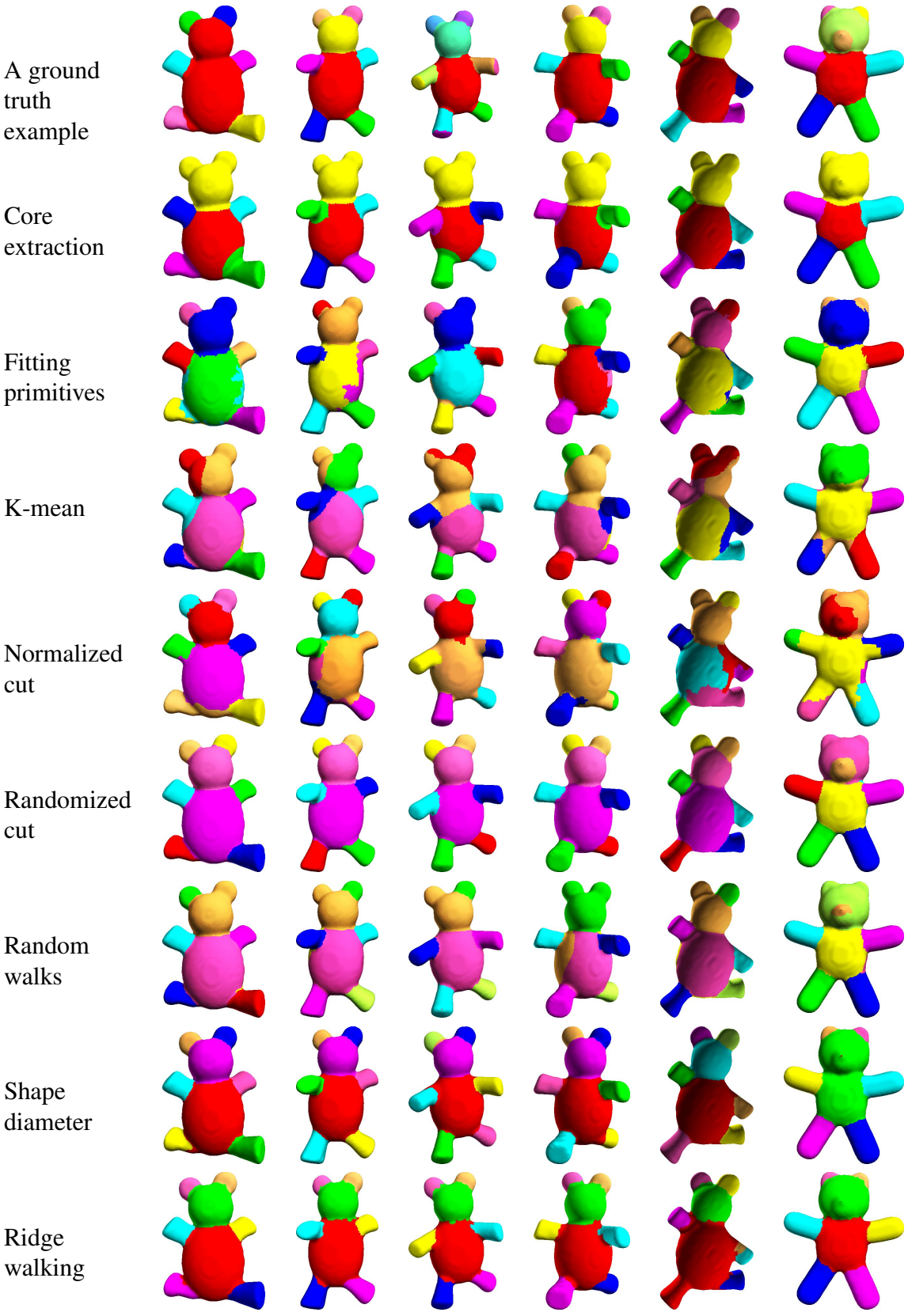
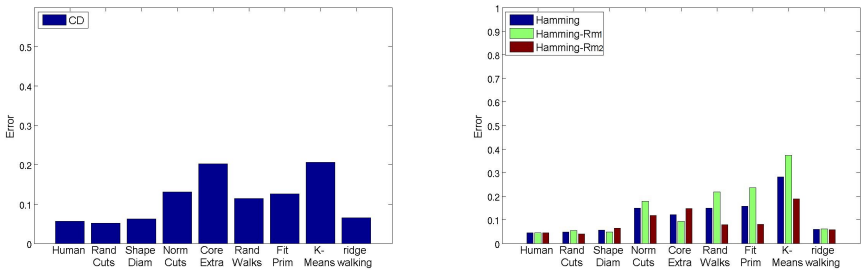
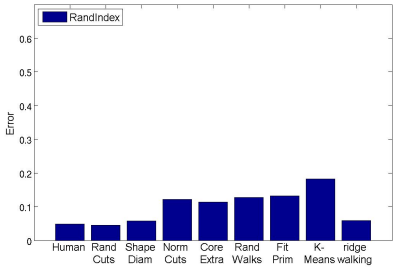


Figure 3.2: Segmentation results for 6 different stuffed bear models having different poses using 9 different segmentation methods.

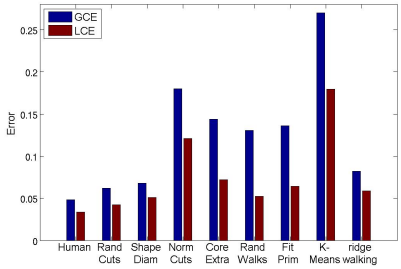


(a) cut discrepancy

(b) Hamming distance



(c) Rand index



(d) consistency error

Figure 3.3: Comparison of segmentation algorithms with four evaluation metrics for stuffed bear models using different segmentation algorithms

indicates that the ridge walking algorithm did not tend to over-segment or under-segment the different stuffed bear models. This is also reflected in figure 3.3(d) which shows a small difference between *GCE* and *LCE* for the ridge walking algorithm. Overall, these evaluation scores indicate that the ridge walking algorithm produces segmentations similar to the ground truth examples.

Figure 3.4 contains three columns of histograms, each column shows a histogram of a different geometric property of the segmentation results for the stuffed bear models. From left-to-right, the histograms shown are: (1) the minimum curvatures for each point on segmentation boundary (shown as a solid line), the minimum curvatures for each point in the segmentation regions (shown as a dashed line), (2) the normalized segment boundary length, and (3) the normalized segment surface area. Each row shows histogram of the values of different segmentation methods. Analysis of the ridge walking algorithm and other algorithms is accomplished by comparing the shape of the histograms from figure

3.4. It is proposed that this analysis demonstrates that the ridge walking algorithm has the following three properties: (1) it tends to segment the stuffed bear models along concavities in a way similar to the ground truth, (2) it tends to generate segment boundary lengths that are approximately the same in a way similar to the ground truth, (3) it tends to generate a large number of small segments and a small number of large segments in a way similar to the ground truth. The support for these claims are discussed in the following paragraphs.

The left-most column of figure 3.4 shows the histograms of minimum curvature for the points on the segmentation boundary and for the points within the segmented regions. The similarities in the histograms for the ground truth examples and the histograms for the ridge walking and randomized cuts algorithms suggest that the segment boundaries for these three approaches indeed follow the concave contours of the stuffed bear models, i.e., most of the boundary vertices have negative minimum curvatures. For the K-means and core extraction algorithms, almost half of the boundary vertices are at convex locations of the surface. This suggests that the segmentations produced by these methods are significantly different from ground truth. The segmentation results in figure 3.2 also reflect this tendency. This claim is also supported by the poor evaluation scores for the K-means and core extraction algorithms shown in figure 3.3.

The middle column of figure 3.4 shows the histograms of normalized segment boundary lengths for the segmented bear models. The histogram for the ridge walking algorithm suggests that this algorithm tends to generate segment boundary lengths that are approximately the same, which is indicated by the peak of histogram at a normalized length value of 1. The ground truth examples shown in first row of the figure 3.4 also have this property. However, significant differences exist between the histogram generated by the K-means algorithm and the ground truth histogram. Comparison of these histograms suggest that the K-means algorithm tends to generate short boundaries. This tendency is different from the ground truth histogram and indicates undesirable results, which are also reflected by the high evaluation scores in figure 3.3.

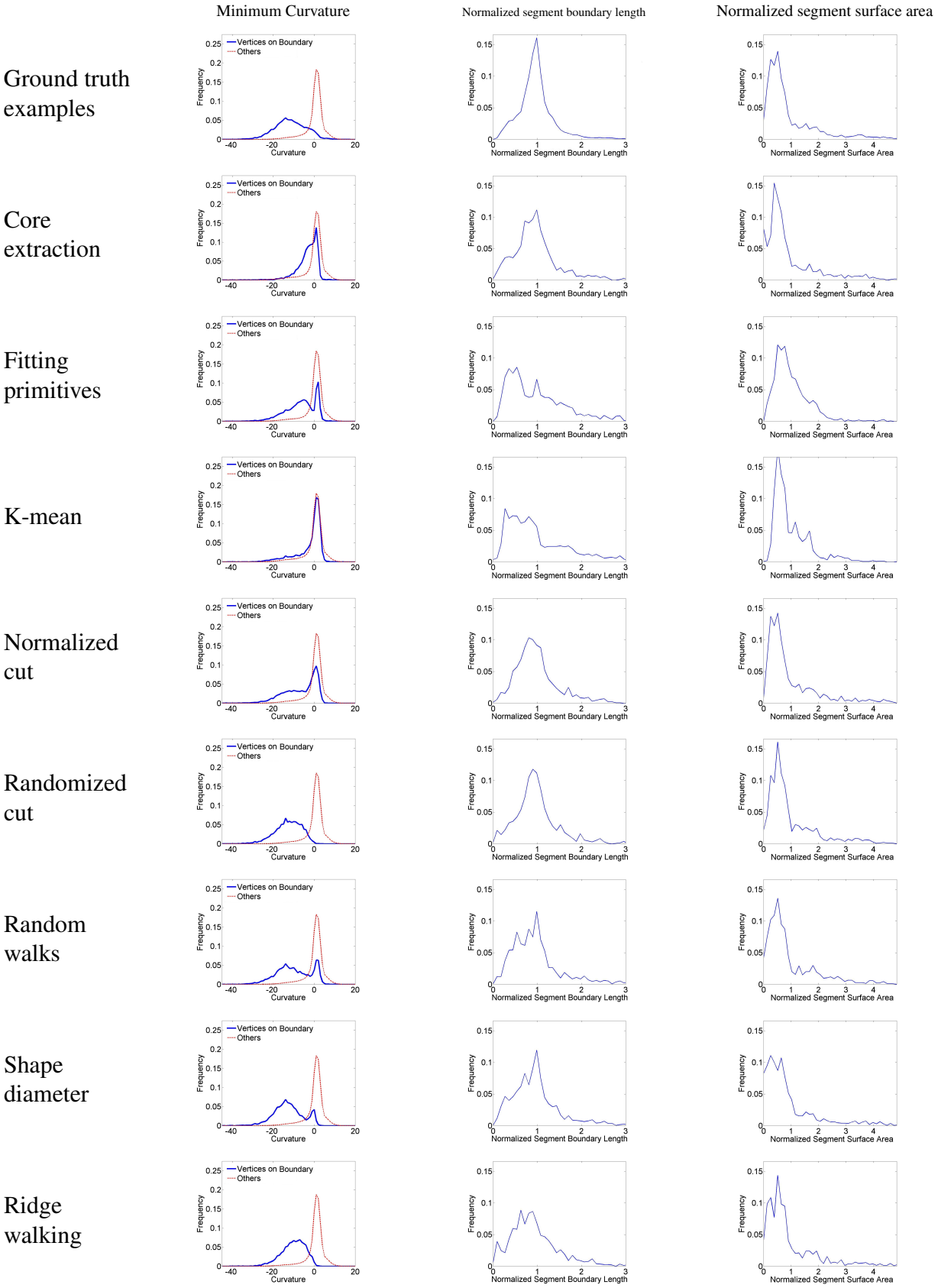


Figure 3.4: Histograms of 3 different geometric properties measured for segmentations of the 20 stuffed bear models.

The right-most column of figure 3.4 shows the histograms of the normalized segment surface areas for the segmented bear models. In general, besides the fitting primitives algorithm, the peaks of histograms are located at a value smaller than 1. These histograms suggest that the majority of the segmented regions are small and there are also several large regions. However, there is a small perturbation near the peaks of these histograms. The histograms for the ridge walking and randomized cut algorithms both have local maximum immediately to the left of the peak at approximately 0.2. This phenomenon is repeated in the histogram of the segment areas for the ground truth examples. One can also see this in the ridge walking segmentations of the stuffed bear models shown on the bottom row of the figure 3.2. In particular, the bear models shown have consistent segment boundaries that separate the model into "body" and "appendages". The areas of appendages such as ears, legs, and arms are smaller than the area of the body. Among the "appendages", the legs and arms are almost the same size. Also, the area of the ears are slightly smaller than the areas of the legs and arms. It is suspected that these subtle variations in the areas of the bear parts are responsible for this phenomenon in the histogram. However, the histogram for the fitting primitives algorithm has a peak at 1. This indicates that this algorithm tends to segment the models into regions of equal size. This does not follow the ground truth, which explains why the fitting primitives algorithm has poor evaluation scores in figure 3.3.

In summary, the evaluation scores and the histograms of the geometric properties show that the ridge walking algorithm performs well for the stuffed bear models. This is not surprising because the semantic components of stuffed bear models are typically taken as the arms, legs, ears and head, and all of these tend to be separated by following the concave contours of the surface. The evaluation scores indicate that the ridge walking algorithm tends to reliably find segment boundaries that are very similar to those drawn by humans as provided by the ground truth dataset. The ridge walking algorithm also generates segment regions that have similar geometric properties as the ground truth. This is supported by the visual evidence in figure 3.2, the evaluation scores of figure 3.3, and analysis of the

histograms in figure 3.4.

3.2.3 Evaluation and Analysis for Human Models

The human model category includes 20 models of humans in different poses. Figure 3.5 shows segmentation results for 6 of these models using 8 different segmentation algorithms and includes examples of 6 ground truth segmentations in the first row. In contrast to the stuffed bear model analyzed in §3.2.2, the results show that segmentations by the ridge walking algorithm are significantly different from those of the ground truth dataset. For example, the human arms can be segmented from the body by ridge walking algorithm, but these segment boundaries are not around the shoulders as shown in the ground truth examples. The ridge walking algorithm segments the models into several parts by dividing it along the nose or eyes.

Figure 3.6 shows a bar chart of the evaluation scores for the 9 segmentation methods investigated for each of human model. Figure 3.6(a) shows scores for cut discrepancy metric. The cut discrepancy metric score of the ridge walking segmentation is much worse than the ground truth examples. This indicates that the positions of segmentation boundaries generated by the ridge walking algorithm are much different from the ground truth examples. Figure 3.6(b,c,d) show scores for the Hamming distance, the Rand index, and the consistency error respectively. Here, the ridge walking algorithm has worse scores than the ground truth other segmentation algorithms. This indicates that the regions computed by the ridge walking algorithm are different from the regions of the ground truth segmentations. Figure 3.6(b) shows a high difference between the scores R_{m1} and R_{m2} for the ridge walking algorithm. This indicates some of the ridge walking segmentation results are over-segmented and some of the ridge walking segmentation results are under-segmented. This is also reflected in 3.6(d), which shows the high difference between *GCE* and *LCE* for the ridge walking algorithm. Since all evaluation scores indicate that the ridge walking algorithm produces undesirable segmentations for human models. One can conclude that the ridge walking segmentation results are different from the ground truth examples.

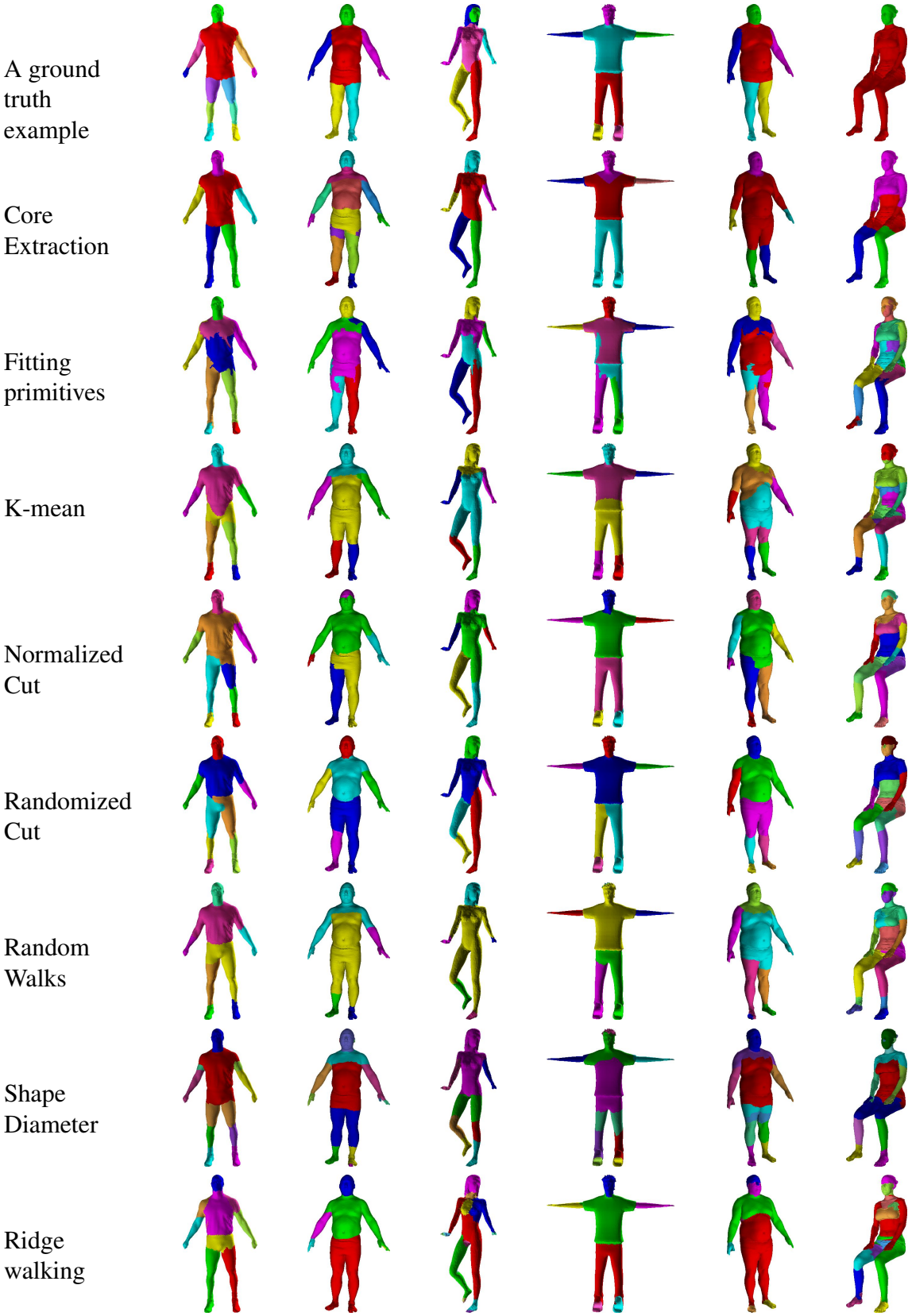
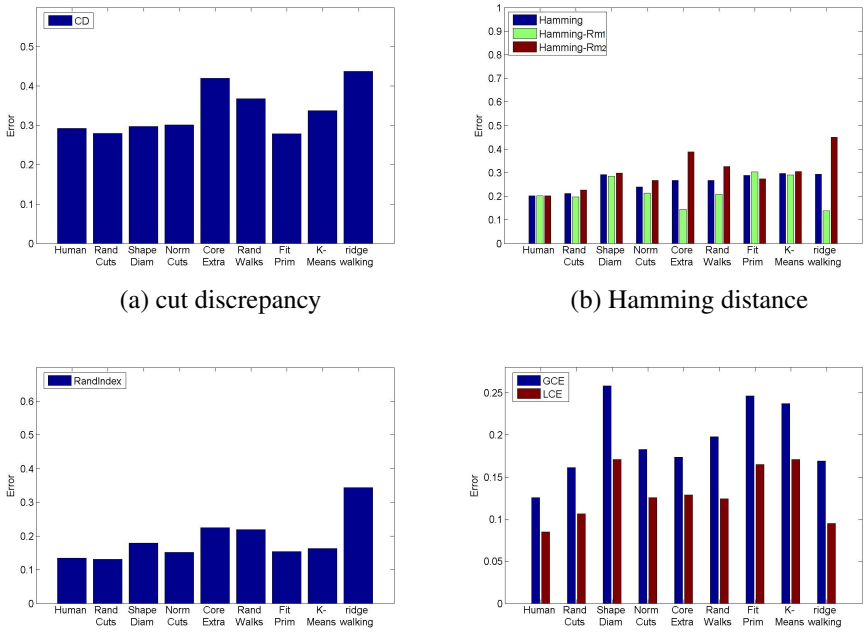


Figure 3.5: Segmentation results for 6 models from human category having different gestures using nine different segmentation methods.



(a) cut discrepancy

(b) Hamming distance

(c) Rand index

(d) consistency error

Figure 3.6: Comparison of segmentation algorithms with four evaluation metrics for human models using different segmentation algorithms

Figure 3.7 contains three columns of histograms, each column shows histograms of a different geometric property of the segmentation results for the human models. Analysis of the ridge walking algorithm and other algorithms is accomplished by comparing the shape of the histograms from figure 3.7. It is proposed that this analysis demonstrates that the ridge walking algorithm has the following three properties: (1) it tends to segment the human models along concavities, (2) it tends to generate the small segment boundary length, this is consistent with the ground truth, (3) it tends to generate either very large segments or very small segments, this is different from the ground truth as it generates a small number of large segments. The support for these claims are discussed in the following paragraphs.

The left-most column of figure 3.7 shows the histogram of minimum curvatures for the points on the segmentation boundary and for points within the segmented regions for human models. The histogram for the ground truth examples suggests that the segment

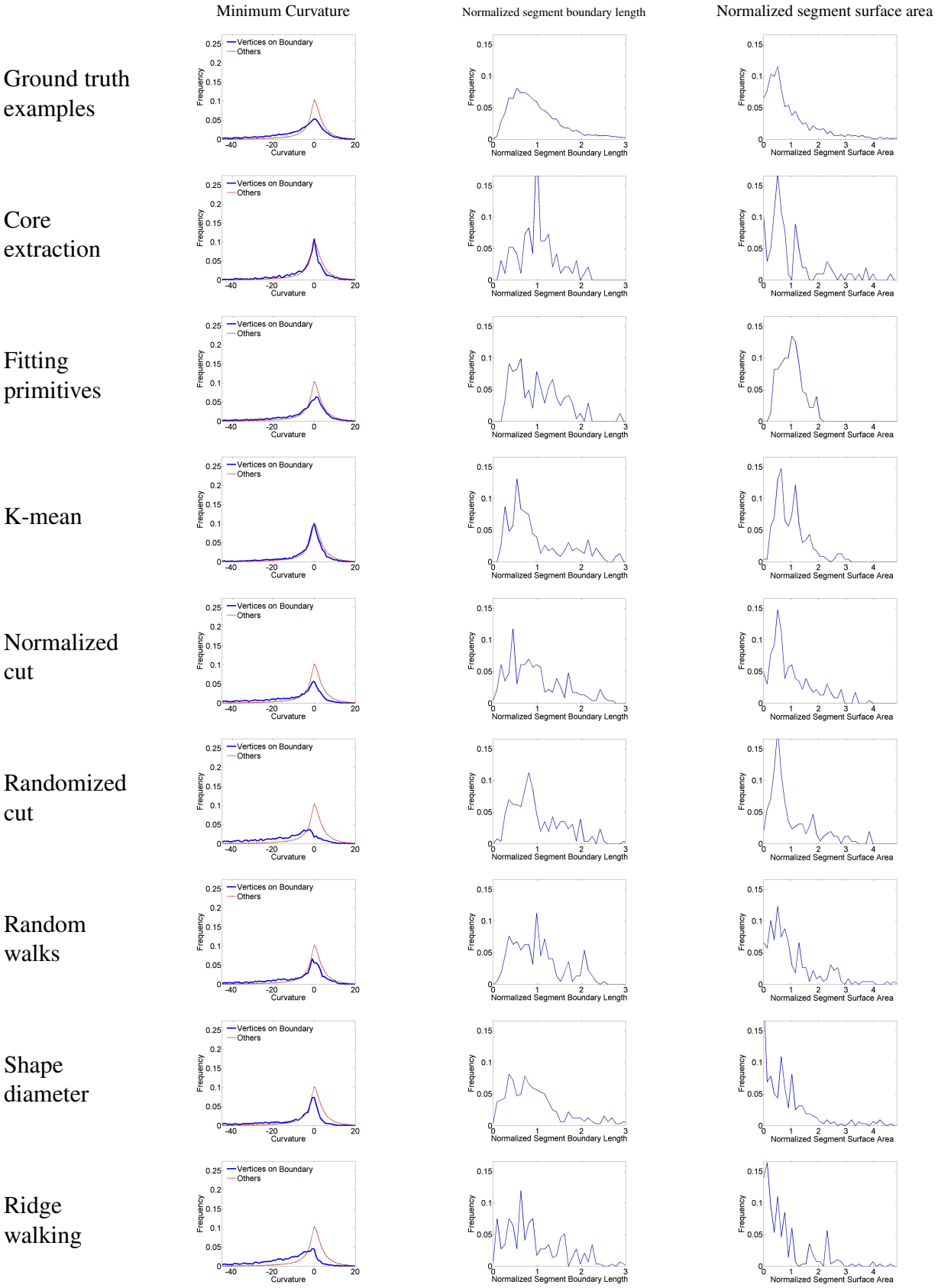


Figure 3.7: Histograms of 3 different geometric properties measured for segmentations of the 20 human models.

boundaries for this approach follow both concave, convex and flat regions of the surface, i.e., the histogram of minimum curvatures for points on the segmentation boundary is similar to a Gaussian function with mean at 0. However, the histogram for the ridge walking algorithm suggests that the segment boundaries for this approach follow the concave valleys and flat areas of the surface, i.e., most of the boundary vertices have negative or zero minimum curvatures. The segmentation results in figure 3.5 also show this difference. This is likely due to the segmentation criterion of the ridge walking algorithm which is set to segment the models exclusively along concavities (see equation (2.10) for details). Hence, the segmented boundaries and regions tend to deviate away from the ground truth examples in flat and convex surface regions. These deviations can explain the poor evaluation scores in figure 3.6.

The middle column of figure 3.7 shows the histograms of the normalized segment boundary lengths for the segmented human models. The similarities in the histograms for the ground truth examples and the histograms for the ridge walking, K-means, normalized cut, randomized cut, random walks and shape diameter function algorithms suggest that segment boundaries for these seven approaches are short, i.e., 75% of the normalized boundary length is located at value smaller than 1. However, significant differences exist between the histograms generated by the core extraction algorithm and that for the ground truth. This suggests that the core extraction algorithm tends to generate boundaries with equal length, i.e., the peak of the histogram is located at value of 1. This tendency is different from the ground truth histogram and indicates undesirable results, which are also reflected by the poor evaluation scores in figure 3.6.

The right-most column of figure 3.7 shows the histograms of the normalized segment surface areas for the segmented human models. The histogram for the ridge walking algorithm suggests that it tends to segment the surface into small and large regions as indicated by the peaks of histogram located at values of 0.1 and 2.5. The segmentation results for the ridge walking algorithm in the last row of figure 3.5 also reflect this tendency, i.e., the

fingers are segmented into distinct parts having small surface area (see third column of figure 3.5). In some segmentations, the abdomen region is not separate from the leg region and these regions are combined to form a single large region (see fifth column of figure 3.5). One can also see this is significantly different from the histograms computed from the ground truth segmentation. This may explain why the ridge walking algorithm has poor evaluation scores in figure 3.6. The histogram of randomized cuts, normalized cuts and fitting primitives algorithms has a peak located at a value around 0.5, which is similar to the ground truth. Those three algorithms have good evaluation scores in figure 3.6.

In summary, the ridge walking algorithm performs the worst for the human models, because the ridge walking algorithm seeks to compute contours that exclusively follow the concave contours of the surface. In cases where these contours need to traverse surface regions that are flat or even convex, as in the human models, the algorithm tends to deviate away from human-generated segmentations. Hence, the ridge walking algorithm cannot effectively segment the human appendages from body. This is supported by the visual evidence in figure 3.5, the evaluation scores of figure 3.6, and analysis of the histograms in figure 3.7. If the segmentation criterion for ridge walking algorithm is adjusted to segment along both concave and convex regions of the surface (see equation (2.3)), segmentation result will have too many details, i.e. the segment boundaries will traverse the apex and sides of each finger separating the finger surfaces into two parts, as shown in figure 3.8.

3.2.4 Evaluation and Analysis for Cup Models

The cup model category includes 20 models of cups. Figure 3.9 shows the segmentation results for 6 of these models using 8 different segmentation algorithms and includes examples of 6 ground truth segmentations in the first row. The ridge walking algorithm segmentation results are shown in the last row of figure 3.9. The results shown suggest that the ridge walking algorithm can consistently segment the cup models into three parts: the body, the bottom and the handles.

Figure 3.10 shows a bar chart of the evaluation scores for the 9 segmentation meth-

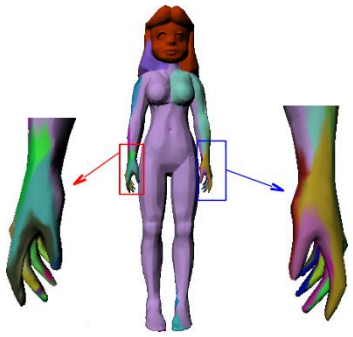


Figure 3.8: The middle of the figure shows the segmentation result for a human model by segmenting it along both concave and convex curves of the surface. The hands of the human are amplified as shown in the left and right side of the human model. The segmentation result indicates that the hands of the human are over-segmented, i.e., the segmented boundaries traverse the apex of the fingers causing finger to be segmented into two parts.

ods investigated for cup models. Figure 3.10(a) shows scores for cut discrepancy metric. The cut discrepancy scores for most of the segmentations are abnormally large (i.e., they are greater than 1) . This is because some cup models are tall and thin which causes the *avgRadius* values from equation (3.3) to be abnormally small for these models. Figure 3.10(b,c,d) show scores for the Hamming distance, the Rand index, and the consistency error respectively. Here, among all the automatic segmentation algorithms, the ridge walking algorithm has the score closest to the ground truth examples. This indicates that the regions computed by the ridge walking algorithm are close to the regions of the ground truth segmentations. Figure 3.10(b) shows a slight difference between the scores R_{m1} and R_{m2} for the ridge walking algorithm. This indicates that the ridge walking algorithm will not over-segment or under-segment the cup models. This is also reflected in the small difference between *GCE* and *LCE*. So, all four evaluation metrics scores support the conclusion that the ridge walking algorithm produces good segmentations for cup models and that these segmentation results are similar to the ground truth segmentations.

Figure 3.11 contains three columns of histograms, each column shows histograms of a different geometric property for a collection of cup model segmentation results. Analysis of the ridge walking algorithm and other algorithms is accomplished by comparing the shape

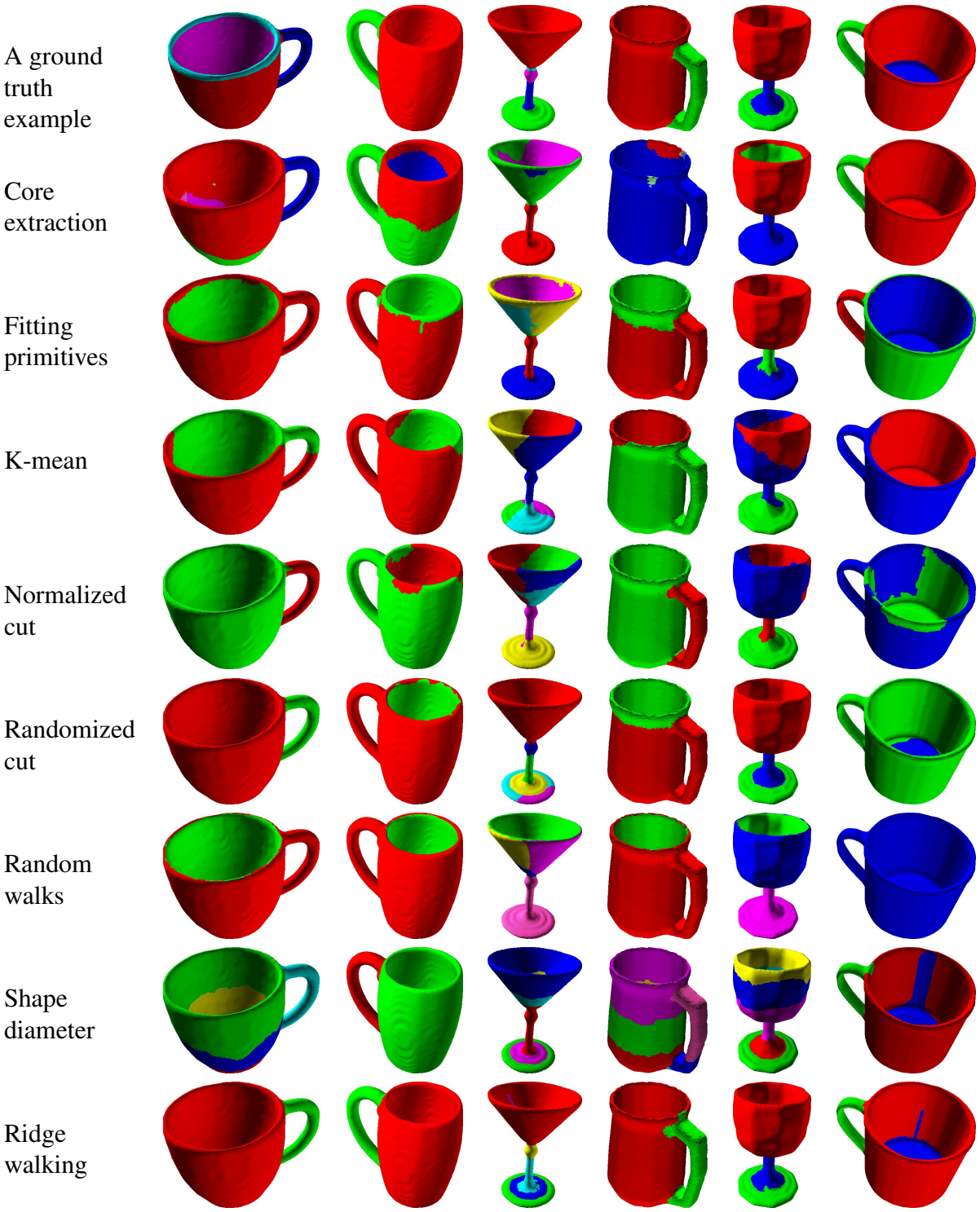
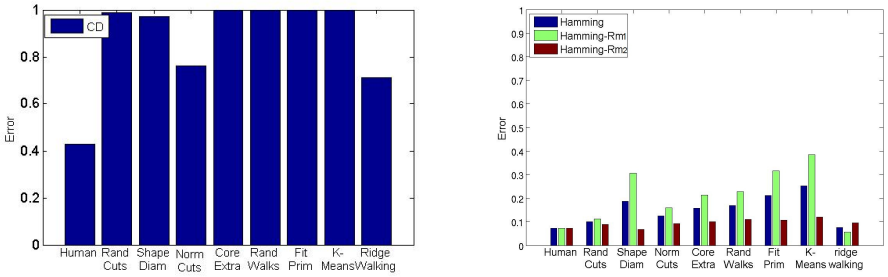
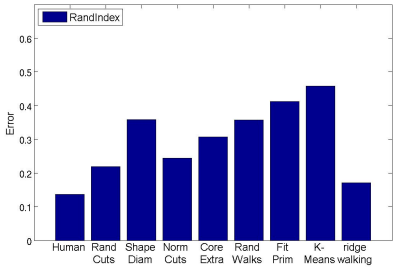


Figure 3.9: Segmentation results for 6 models from cup category having different gestures using nine different segmentation methods.

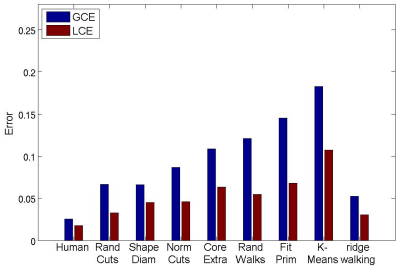


(a) cut discrepancy

(b) Hamming distance



(c) Rand index



(d) consistency error

Figure 3.10: Comparison of segmentation algorithms with four evaluation metrics for cup models using different segmentation algorithms.

of the histograms from figure 3.11. It is proposed that this analysis demonstrates that the ridge walking algorithm has the following three properties: (1) it tends to segment the cup models along concavities in a way similar to the ground truth, (2) it tends to generate short segment boundaries in a way consistent with the ground truth examples, and (3) it tends to generate more large surface regions than the ground truth examples. The support for these claims are discussed in the following paragraphs.

The left-most column of figure 3.11 shows the histograms of minimum curvature for points on the segmentation boundary and within the segmented regions as two different curves. The similarities in the histograms for the ground truth examples and the histograms for the ridge walking and randomized cuts algorithms suggest that the segment boundaries for these three approaches indeed follow the concave contours of the cup models, i.e., most of the boundary vertices have negative minimum curvatures. However, for the ridge walking algorithm, the number of segment boundaries along concavities is slightly more than

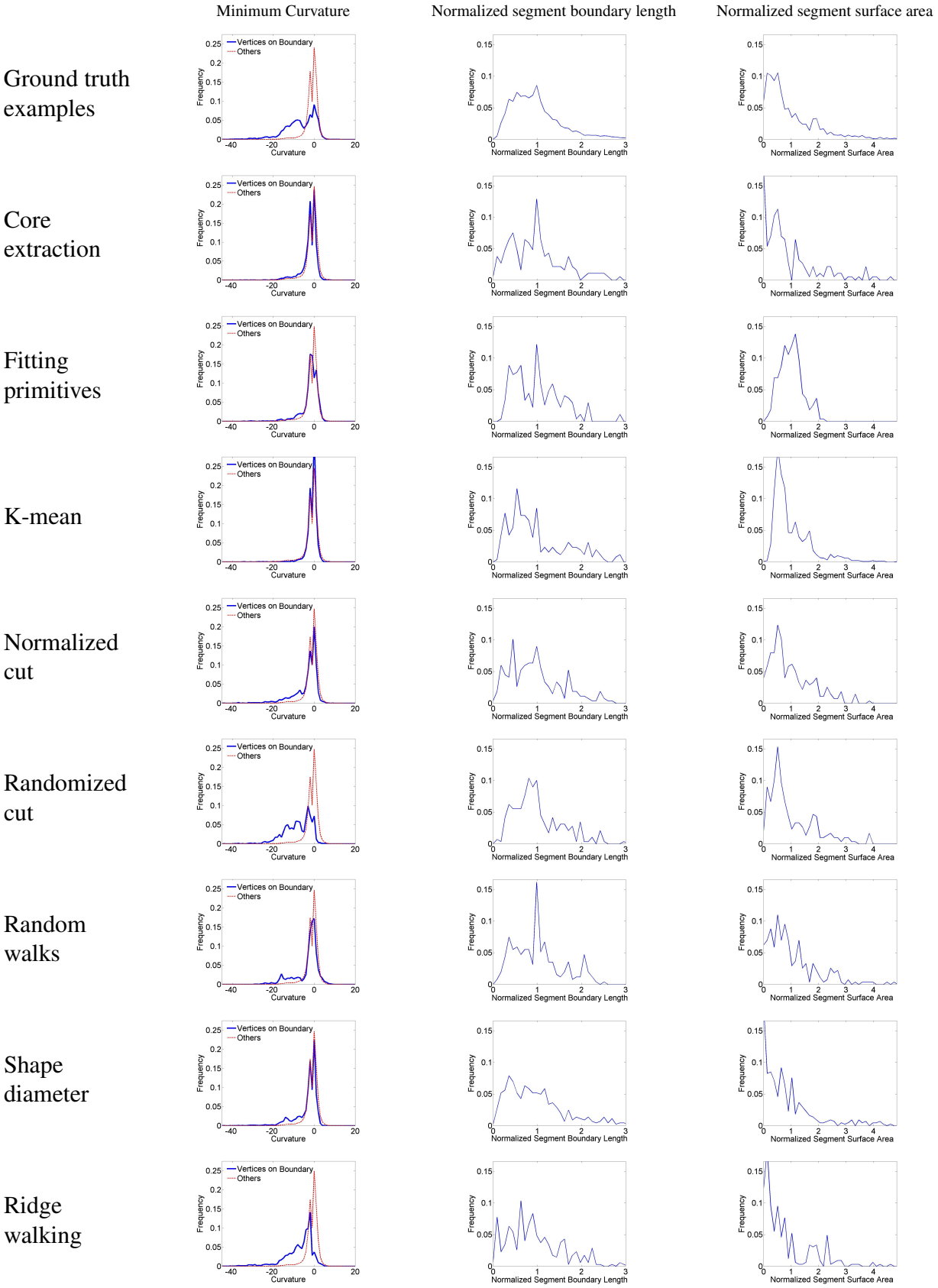


Figure 3.11: Histograms of 3 different geometric properties measured for segmentations of the 20 cup models.

the ground truth. This may be explained by the fact that the ridge walking algorithm favors computing segment boundaries that lie along concave contours of the surfaces, i.e., boundaries that include contours with large average edge weights (see §2.2.3 for details). In contrast, at least half of the boundary vertices generated by core extraction, K-means, normalized cut, and shape diameter function algorithms are at convex locations of the surface. This suggests that the segmentations produced by these methods are significantly different from ground truth. Visual inspection of the segmentation results in figure 3.9 also reflect this tendency. This claim is also supported by the poor evaluation scores for the core extraction, K-means, normalized cut, and shape diameter function algorithms shown in figure 3.10.

The middle column of figure 3.11 shows the histograms of normalized segment boundary lengths for the segmented cup models. The histogram for the ridge walking algorithm suggests that this algorithm tends to generate a large number of segments that have small segment boundary lengths, which is indicated by the fact that most of the histogram values lie at a normalized length value smaller than 1. The ground truth examples shown in the first row of the figure 3.11 also have this property. However, significant differences exist between the ground truth histogram and histogram generated by the random walks algorithm. This suggests that the random walks algorithm tends to generate boundaries with approximately equal length. This tendency is different from the ground truth histogram and indicates undesirable results, which are also reflected by the poor evaluation scores in figure 3.10.

The right-most column of figure 3.11 shows the histograms of the normalized segment surface areas for the segmented cup models. In general, besides the fitting primitives algorithm, the peaks of these histograms are located at a value smaller than 1. The location of the peaks in these histograms suggest that the majority of the segmented regions have small area with the exception of a small number regions having large areas. One can also see this in the ridge walking segmentations of the cup models shown on the bottom row

of the figure 3.9. In particular, the cup models shown have consistent segment boundaries that separate the model into a “body”, a “bottom” and one or more “handles”. The areas of bottom and handle regions tend to be smaller than the area of the body region. However, as indicated in the histogram, the number of large segments generated by ridge walking algorithm is larger than the ground truth examples. This is because some ground truth examples also subdivide the cup body into an inner body and outer body surface. The ridge walking algorithm does not tend to subdivide this region due to the segmentation criterion that restricts the segmentation boundaries to exclusively traverse concave surface regions. The histogram for the fitting primitives algorithm has a peak at a value larger than 1, which indicates that this algorithm tends to generate more large area segments. This does not follow the ground truth, which explains why the fitting primitives algorithm has poor evaluation scores in figure 3.10.

In summary, the evaluation scores and geometric segmentation properties show that the ridge walking algorithm performs well for cup models. This is not surprising because the semantic components of cup models are typically taken as the body, bottom and handles, and all of these tend to be separated by following contours of the surface that traverse concave surface regions. The results also show that the ridge walking algorithm tends to reliably find segment boundaries that are similar to those drawn by humans as provided by the ground truth dataset. This is supported by the visual evidence in figure 3.9, the evaluation scores of figure 3.10, and analysis of the histograms in figure 3.11.

3.2.5 Evaluation and Analysis for All Models

The entire collection of models includes 380 models from 19 categories. Figure 3.12 shows the segmentation results for 5 of these models using 8 different segmentation algorithms and includes examples of 5 ground truth segmentations in the first row. The ridge walking algorithm segmentation results are shown the last row of figure 3.12. The results shown suggest that the ridge walking algorithm can generate similar segmentation results as the ground truth examples.

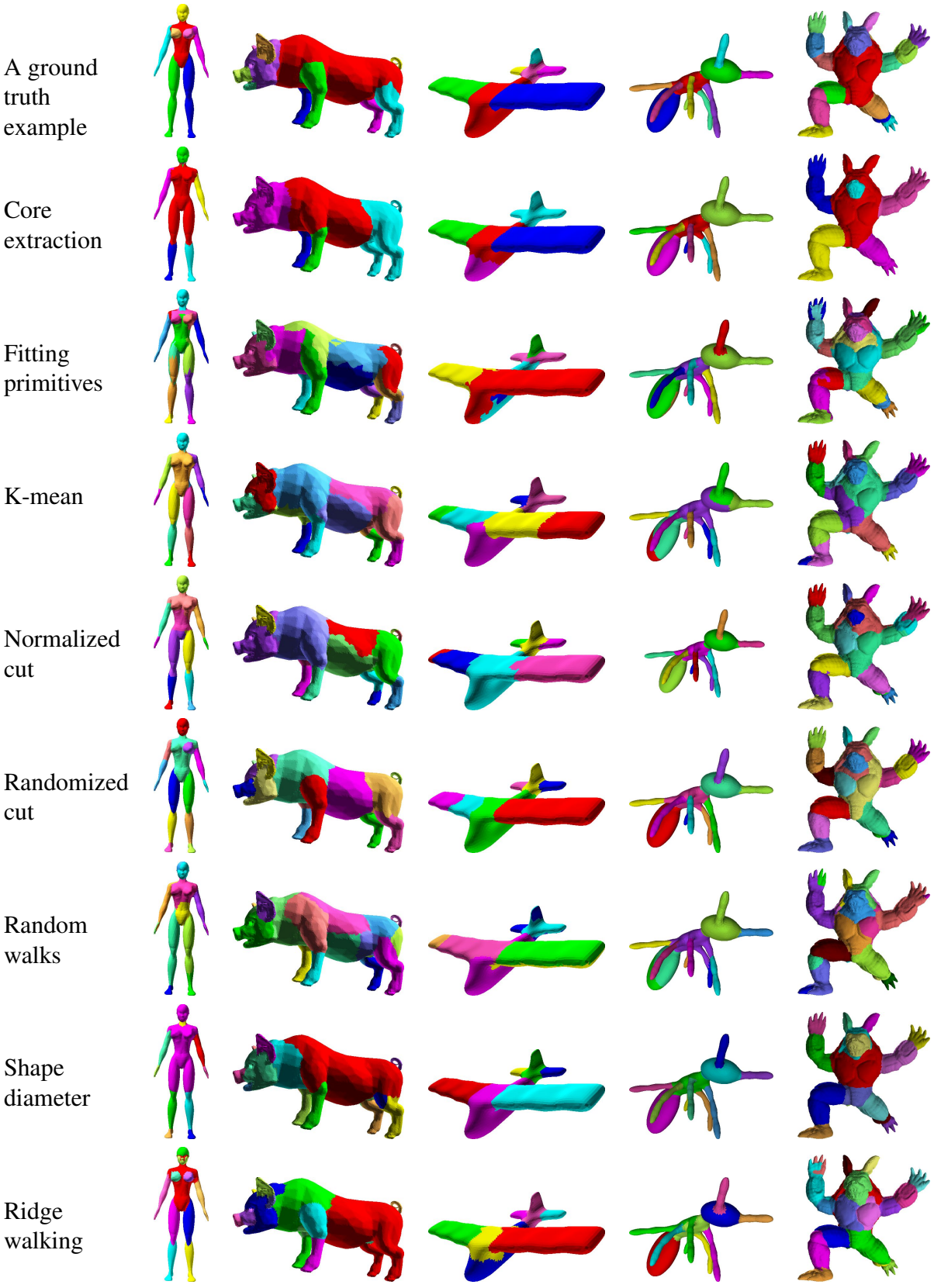
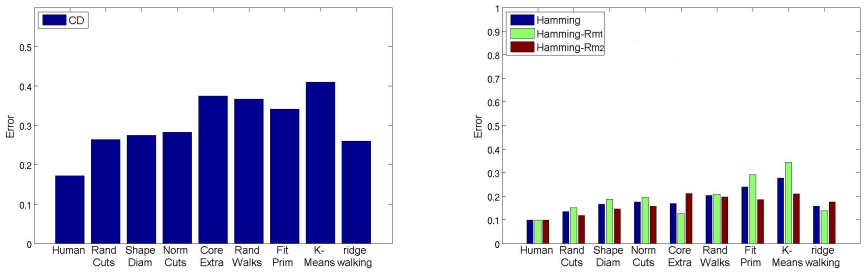


Figure 3.12: Segmentation results for models from 5 different categories using nine different segmentation methods.

Figure 3.13 shows a bar chart of the evaluation scores for the 9 segmentation methods when evaluated over all of the models. Figure 3.13(a) shows scores for the cut discrepancy metric. In this case, the ridge walking segmentation has the score closest to the score for the ground truth examples among all the automatic segmentation algorithms. Figure 3.13(b,c,d) show scores for the Hamming distance, the Rand index, and the consistency error respectively. In this case, the ridge walking algorithm also has the score closest to the scores for the ground truth examples. This indicates that the regions computed by the ridge walking algorithm are close to the regions specified in the ground truth segmentations. Figure 3.13(b) shows a slight difference between the scores R_{m1} and R_{m2} for the ridge walking algorithm. This indicates that the ridge walking algorithm will not over-segment or under-segment most of models in the dataset. This is also reflected in the small difference between GCE and LCE. Hence, all four evaluation scores indicate that the ridge walking algorithm tends to produce good segmentations when evaluated over all of the models in the test dataset.

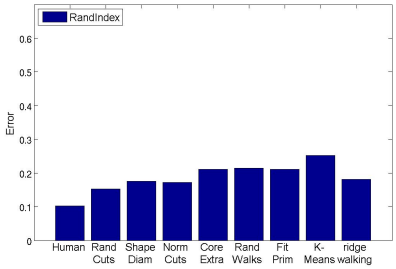
Figure 3.14 contains three columns of histograms, each column shows histograms of a different geometric property for the segmentation results for all of the models. Analysis of the ridge walking algorithm and other algorithms is accomplished by comparing the shape of the histograms from figure 3.14. It is proposed that this analysis demonstrates that the ridge walking algorithm has the following three properties: (1) it tends to segment the models along concavities in a way similar to the ground truth segmentations, (2) it tends to generate segment boundaries that have approximately the same arc-length as the boundaries in the ground truth segmentations, but the ridge walking algorithm tends to generate slightly more long segment boundaries than the ground truth boundaries, and (3) it tends to generate a large number of small segments or a small number of large segments in a way consistent with the ground truth segmentations. The support for these claims are discussed in the following paragraphs.

The left-most column of figure 3.14 shows the histograms of minimum curvature for

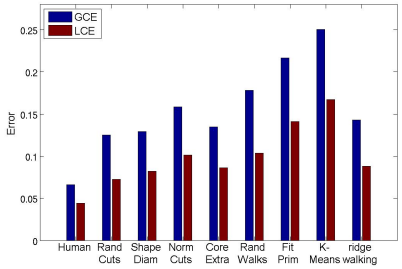


(a) cut discrepancy

(b) Hamming distance



(c) Rand index



(d) consistency error

Figure 3.13: Comparison of segmentation algorithms with four evaluation metrics for all models in the dataset using different segmentation algorithms

points on the segmentation boundary and within the segmented regions. The histograms for the ridge walking segmentations and the ground truth segmentations are similar. This suggests that the segment boundaries are indeed more likely to lie along concave contours of the model, i.e., most boundary vertices have negative minimum curvatures. However, the ridge walking algorithm seems to choose points that have slightly lower curvatures than those chosen in the ground truth segmentations. This is because the ridge walking algorithm favors segment boundaries with large average edge weights (see §2.2.3 for details).

The middle column of figure 3.14 shows the histograms of normalized segment boundary lengths for all of the segmented models. The histogram for the ridge walking algorithm suggests that this histogram tends to generate segment boundaries with approximately equal size as indicated by the peak of histogram located at the x axis value 1. As shown in figure 3.14, ground truth examples also have this property. However, the ridge walking algorithm tends to generate segment boundaries that are slightly longer (normalized segment bound-

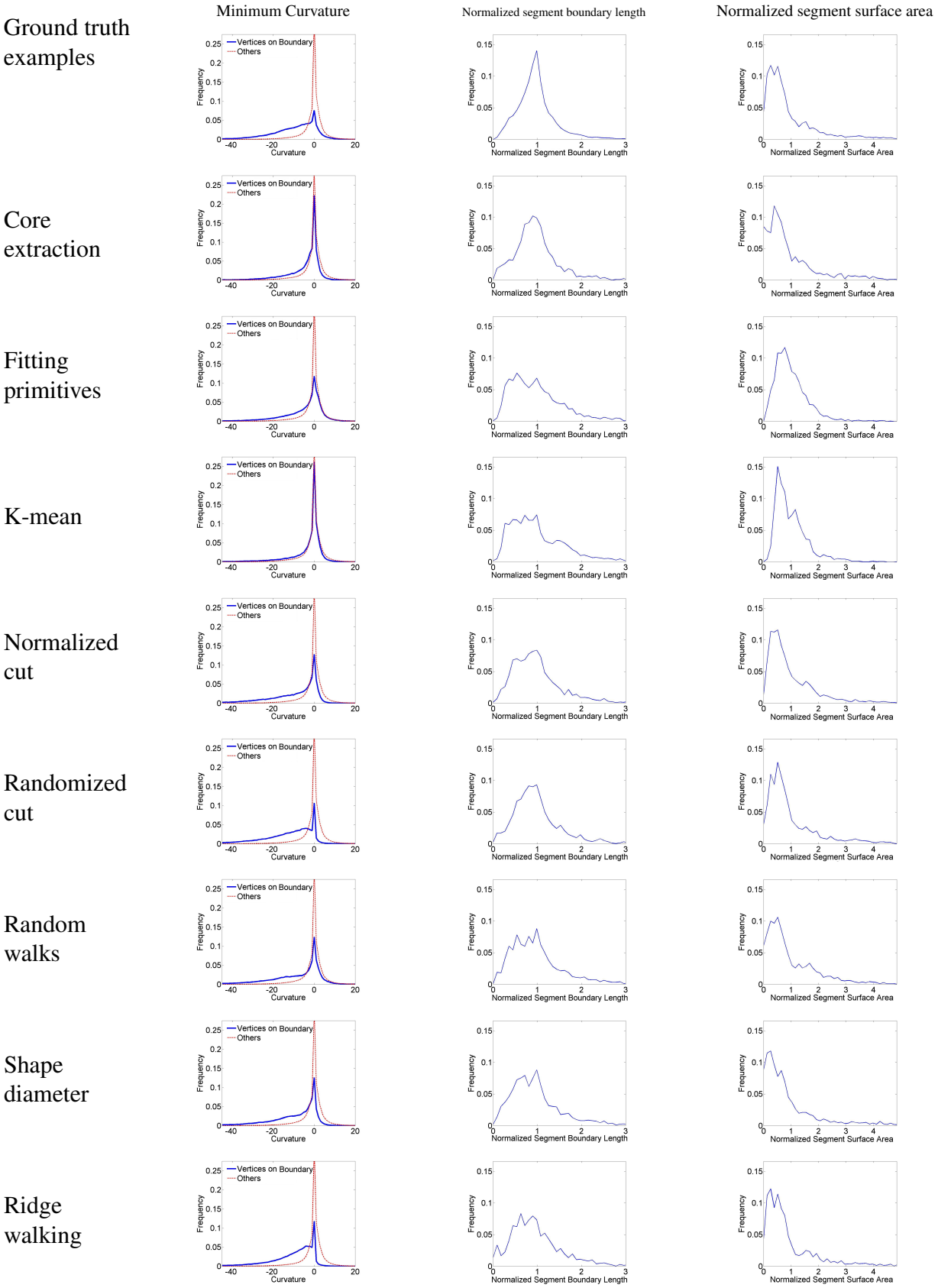


Figure 3.14: Histograms of 3 different geometric properties measured for segmentations of the all the models in the dataset.

ary length>1) than the boundaries of the ground truth segmentations. This is not surprising, since the ridge walking tries to keep long segment boundaries using the contour constraints (see §2.2 for details).

The right-most column of figure 3.14 shows the histograms of the normalized segment surface areas for all of the segmented models. The histogram for the ridge walking algorithm suggests that it tends to generate a large number of segments having small surface area and small number of segments having large surface area as indicated by the peak of histogram which is located at a value significantly smaller than 1. This suggests that the ridge walking algorithm generates surface regions whose areas are consistent with the areas of regions found in the ground truth segmentations. This phenomenon may be due to the contour constraints enforced on the length and similarity of segmentation boundaries in the ridge walking algorithm (see §2.2.3 for details). In summary, the ridge walking algorithm produces segmentations that have similar geometric properties to the geometric properties found in the ground truth segmentations. This is supported by the visual evidence in figure 3.12, the evaluation scores of figure 3.13, and analysis of the histograms in figure 3.14.

3.3 Conclusion

In this chapter, the ridge walking segmentation algorithm is quantitatively evaluated by comparing its results with ground truth segmentation results using four evaluation metrics. A similar evaluation is done for seven other algorithms. The geometric properties of the segmentation results are also studied to explore how the ridge walking algorithm segments the model. Overall, the evaluation scores indicate that the ridge walking algorithm is more similar to the ground truth than other segmentation algorithms. The geometric segmentation properties for segmented boundaries and regions generated by the ridge walking algorithm are also close to the ground truth. However, the ridge walking algorithm has some shortcomings. Specifically, it tends to generate contours that are slightly longer than the ground truth contours and these contours tend to traverse surface regions that are slightly more concave than the ground truth contours.

However, the ridge walking algorithm does not work well for all categories of models. For categories such as cup, airplane, ant, stuffed bear, plier, fish, bust, CAD model, and vase, the ridge walking algorithm has good performance. For these categories, the evaluation results and segmentation geometric properties are close to those generated from the ground truth segmentations. For these models, the ridge walking algorithm segments the surface along concave valleys which closely reproduces how human have segmented these models. For categories such as the human body, the chair, the bird and four-legged animals, the ridge walking algorithm does not outperform competing methods. In these cases, the segmentation boundaries generated by humans tend to traverse both concave regions and convex regions. In contrast, the segmentation boundaries computed by the ridge walking algorithm almost exclusively traverse concave and flat surface regions. This behavior in the ridge walking algorithm is due to the segmentation criterion that restricts the ridge walking algorithm to follow these boundaries. Future work would improve the ridge walking algorithm by making this criterion less restrictive in order to more accurately segment all kinds of models.

CHAPTER 4: IMPROVING BONE FRAGMENT SURFACE GEOMETRIC SEGMENTATION BY USING APPEARANCE DATA

This section describes an adaptation of the ridge walking algorithm for the purpose of segmenting bone fragment surfaces. This is a problem of importance for understanding bone fracture severity and for 3D virtual reconstruction of highly fragmented bone fractures. In this context, a patient's fractured limb is imaged to generate a 3D Computerized Tomography (CT) image. An image segmentation algorithm then extracts the bone fragment surfaces from the 3D image with the goal of generating a 3D surface that closely approximates the surface geometry and pose of each bone fragment. Each of the bone fragment surfaces must then be subdivided into surface patches for geometric matching algorithms which attempt to reassemble the bone fragments automatically by computing pair-wise matches between these surface patches.

The modified algorithm seeks to provide reliable automatic classification of bone fragment surfaces. It accomplishes this goal by extending the ridge walking algorithm so that it uses both the geometry of the bone fragment surfaces and the appearance of the bone fragments in the CT intensity image. The ridge walking algorithm can be used to directly segment these bone fragments using only their shape (geometry). However, the segmented surface patches generated using this approach are often undesirable and also do not include any medically relevant semantic meaning. The 3D CT image provides important additional information via the image intensities in the vicinity of each surface point. The conceptual approach is to purposefully over-segment the geometry of each bone fragment using the ridge walking algorithm and to subsequently merge surface patches having the same surface type using the CT image intensities observed in the vicinity of each surface patch. The

resulting merged surface patches will incorporate new medically-relevant semantic information derived from these CT intensities.

For the purposes of experimentation, we focus on segmenting the anatomically distinct parts of bone fragments generated by fracturing the human tibia bone. As described in § 1.2.2, the semantic parts of interest for tibial bone fragments fall into three categories: (i) *periosteal surface patches*, (ii) *fracture surface patches,* and (iii) *articular surface patches*. Segmented surface patches are labeled to come from one of these three categories using the values of CT image data inside the estimated bone fragment surfaces. Since this intensity information determines the appearance of the fragment in a CT image, the intensity-based labeling process is referred to as appearance-based classification. The input to the algorithm is a CT image and a 3D mesh of a bone fragment registered to the image. The output of the algorithm is a collection of surface patches where each surface patch as labeled to be either a periosteal, fracture or articular surface.

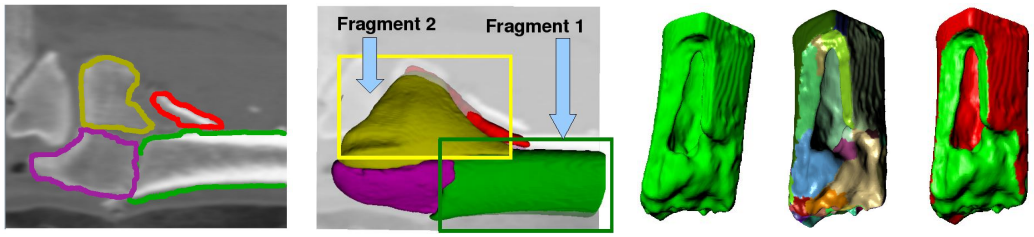
4.1 Previous Work

Appearance-based segmentation algorithms are widely used for medical image segmentation [55, 56, 57]. AAM (Active Appearance Models) is one of the most popular algorithms. AAM [58] is a statistical model of the shape and appearance of the object in an image. In the AAM algorithm, during the training phase, the shape and appearance of an object of interest is parametrized and Principal Component Analysis (PCA) is performed on these parameters to generate a low dimensional representation for the object shape and appearance. New images are segmented by registering, i.e., aligning, the AAM with objects in the new image. At each iteration of the registration, the difference between the shape and appearance of the image object and that of the AAM estimation is minimized to get better match. The AAM algorithm has wide use in medical fields, for example, it is applied to segment tumors in brain MR images in [55] and to segment the lesion in the ultrasound breast images in [56]. The work in [59] also integrates shape and appearance models to segment the whole brain into classes that includes white matter, gray matter and various

subcortical structures (such as caudate, putamen, pallidum, thalamus, etc.). Since the intensity histograms of different neuroanatomical structures of the brain overlap for different classes, spatial information is necessary to correctly segment brain structures. The method incorporates learned shape and intensity priors, then intensity statistics are tabulated within regions throughout the space occupied by the brain data. The intensity distribution for each brain structure class is assumed to be Gaussian with mean and variance estimated from manually labeled training data. The probability of a voxel belonging to a class is defined as the proportion of the voxels (from the training data) that were mapped to each of the brain structure classes. A Markov Random Field (MRF) is used to label image voxels by examining the labels of its neighbors.

Automated Nonlinear Image Matching and Anatomical Labeling (ANIMAL) [57] is also an appearance and shape based segmentation method. The ANIMAL algorithm was first used to segment MRI brain volumes. The idea behind the ANIMAL algorithm is that if two brain volumes (template and target) can be perfectly registered via some nonlinear deformation, the labels defined on the template brain can be transferred to the target brain volume, therefore segmenting it. A multiscale optimization approach is used to find the deformation between template and target using their voxel values. However, this approach depends on a good prior model and it is biased towards the choice of the particular brain template. The ANIMAL algorithm is mainly used to segment different brain tissues from MRI images [57].

The specific task of bone fragment surface segmentation has not been heavily studied. The work in [16] describes a method that uses bone surface CT intensity values to segment the bone fragment surfaces into fracture parts and periosteal parts. But this approach for appearance segmentation only works for a small subset of surface, especially does not work for tibia model. That is because, the tibia metaphysis and diaphysis are both portions of the periosteal surface, yet the hounsfield densities in these two tissues can be significantly different. Furthermore, this work did not segment articular surface which is an important



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

Figure 4.1: The overview of the proposed method for segmenting 3D bone fragment surfaces. (a) shows a slice of CT data and boundaries of the segmented bone fragment surfaces drawn as curves within the image. (b) shows the 3D surfaces registered to the CT data that together form the input to the classification algorithm. (c) shows the detailed geometry of bone Fragment 1 outlined in (b). (d) shows the ridge-walking partitioning of the surface and (e) shows the classification results for Fragment 1 into periosteal (red), and fracture (green) surfaces.

aspect to the bone fragment surface registration problem.

Our algorithm fuses the geometry and appearance information to segment the bone fragment surfaces. The ridge walking algorithm segments the bone fragment surfaces along convex ridges and the appearance model classifies each segmented surface patch to one of three categories: periosteal, fracture and articular. When the geometric partitioning is combined with the appearance-based classification, a segmentation result is obtained that provides a detailed labeling of the fragment as a collection of medically relevant sub-surfaces.

4.2 Methodology

The goal of the bone fragment surface segmentation algorithm is to classify the bone fragment surface into 3 different classes: (1) periosteal surface patch, (2) fracture surface patch, and (3) articular surface patch. The input of the segmentation algorithm is a 3D bone fragment mesh surface which is geometrically registered to the 3D CT image. The output of segmentation algorithm is a collection of fragment surface patches that are labeled to have one of the three types discussed above.

We accomplish this goal through the following sequence of 6 steps:

1. Segment the bone fragment surface into surface patches using the geometric ridge

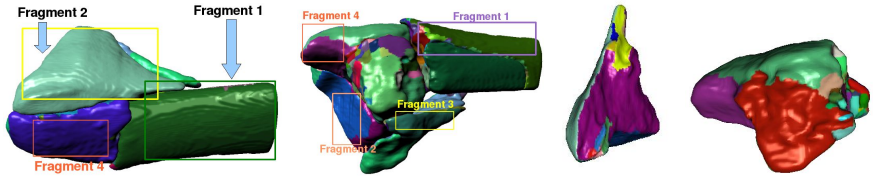
walking algorithm using an artificially high value for the λ input parameter (see §2.2 for details),

2. Compute an appearance feature vector for each point on the segmented surface mesh (see §4.2.1 for details),
3. User interaction is required to train a classifier that classifies feature vectors from (2) to the periosteal, fracture, and articular classes (see §4.2.2 for details),
4. The classifier from (3) is applied to classify each feature vector from (2) and its associated surface point to one of these three semantic classes (see §4.2.3 for details),
5. Each surface patch is classified to a semantic class using the majority vote of the classified points that it contains (see §4.2.3 for details),
6. Adjacent surface patches having the same class label are merged (see §4.2.3 for details).

Interim results for some steps of the appearance-enhanced ridge walking algorithm are shown in figure 4.1. Step (1) is accomplished by a straight-forward application of the ridge walking algorithm using the ridge salience function $w_{ridge}(\mathbf{e}_{ij})$ as shown in equation (2.1). Since the number of segments for each bone fragment surface is unknown, the input parameter λ is used for this segmentation. Since our approach requires that bone fragment surfaces to be purposely over-segmented, an artificially high value for the segmentation parameter was used. Specifically, $\lambda = 0.4$ for each of the shown results such as that shown in figure 4.2. Steps (2)-(6) extend the ridge walking algorithm and serve to incorporate appearance information with the geometric segmentation results. These steps are discussed in detail in §4.2.1-4.2.3.

4.2.1 Surface Appearance Feature Vectors

An appearance feature vector is computed for each point on the segmented surface. The appearance feature vector serves to represent the values of the CT image in the vicinity of each point. These intensities are different for each of the semantic classes and these intensity features allow us to classify each surface point to one of these classes. The appearance



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

Figure 4.2: (a-b) shows the ridge walking segmentation results for bone fragment surfaces in different colors. (c) shows the segmentation results of fragment 2, (d) shows the segmentation results of fragment 4. The input parameter of ridge walking algorithm, λ, is 0.4.

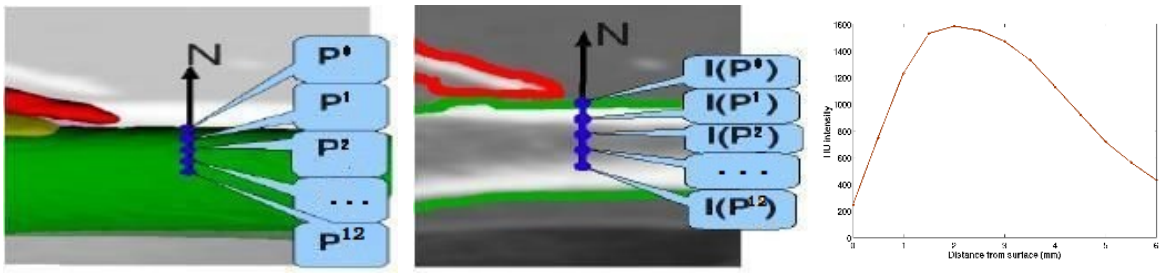
feature vector is computed from a sequence of CT values generated by taking intensities from the CT image at a set of prescribed locations. These locations are determined by incrementally "drilling down" into the bone from each surface point (as shown in figure 4.3(a)). We accomplish this computationally by computing a sequence of positions, $\mathbf{p}_i^k$, for each surface point, $\mathbf{p}_i$. The sequence starts with the surface point itself, i.e., $\mathbf{p}_i^0 = \mathbf{p}_i$. Then N additional positions are generated by moving along the inward pointing normal of the surface at a prescribed interval, Δs, as described by equation (4.1).

$$\mathbf{p}_i^k = \mathbf{p}_i - k\Delta s\mathbf{n}_i, \;\; k = 0,1,2,..,12 \tag{4.1}$$

In equation (4.1), $\mathbf{n}_i$ denotes the direction of the outward pointing normal at the point $\mathbf{p}_i$. The interval Δs depends upon the resolution of the provided CT data (for these experiments $\Delta s = 0.5mm$ and $N = 13$). The CT intensity at each position, $I(\mathbf{p}_i^k)$, is computed by interpolating the CT image intensity at each position. The resulting sequence of intensities forms an appearance feature vector of N values, $\mathbf{x}_i$, as shown in equation (4.2) and figure 4.3(b). The appearance feature vector for each point is referred to as a *CT-profile*.

$$\mathbf{x}_i = \left(I(\mathbf{p}_i^0), I(\mathbf{p}_i^1), \ldots, I(\mathbf{p}_i^{12})\right)^t \tag{4.2}$$

The CT-profile has different values for different anatomical locations on the bone fragment surfaces. For example, CT-profiles taken from the proximal and distal surface of the tibia will generate two significantly different sequences of intensity values (see Figure 4.4(b)). For this reason *a set CT-profile patterns* are computed that characterize the distinct



(a) (b) (c)

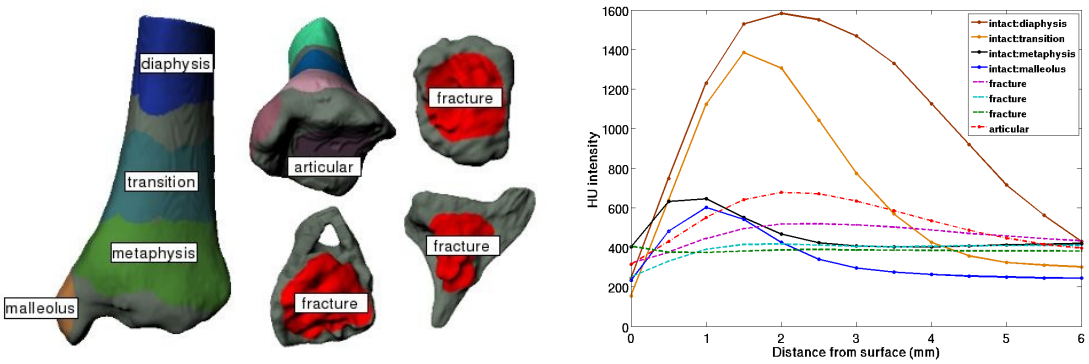
Figure 4.3: (a,b,c) depict graphically the method used to compute a CT-profile appearance feature vector. (a) shows a surface (in green) and a point on the surface $\mathbf{p}$. $\mathbf{p}$ and the normal, $\mathbf{N}$, are used to generate a sequence of points $(\mathbf{p}^0, \mathbf{p}^1, ..., \mathbf{p}^{12})$ using equation (4.1). (b) shows the CT intensities which are sampled at each of these locations giving a vector $\mathbf{x}_i$ as in equation (4.2). (c) shows a CT-profile as a sequence of values plotted as a function of distance along the inward pointing surface normal.

tissue patterns that one can expect to observe for each class. For our experiments, four CT-profiles, were computed to model periosteal surface regions, three CT-profiles were computed to model fracture surface regions, and one CT-profile was computed to model articular surface regions. Each CT-profile pattern is computed by manually selecting regions from a segmented bone fragment surface that is registered to a CT image. A CT-profile is computed for each surface point within the manually selected regions and these CT-profiles are the training data used to compute each of the appearance classifiers.

4.2.2 Training the Surface Point Classifiers

A classifier is used to estimate a semantic label for each bone fragment surface point. The approach for computing the classifier falls into the generic category of supervised learning and applies two techniques taken from the pattern recognition literature: (1) work from [60] is used to reduce the dimensionality of the appearance data and (2) work from [61] is used to perform minimum-error classification. The classifier labels each surface point to one of three classes using the CT-Profile appearance features computed for each point. The methods used to compute these classifiers are discussed in this section.

For each class a classifier is computed from training data that is obtained by having a user interactively label the data. This is accomplished through a 3D interface where the user



(a) (b)

Figure 4.4: (a) shows the periosteal, fracture, and articular surface regions of a tibia as distinct colors. CT-profile features are collected from each region to create a CT-profile classifier. The CT-profiles are determined by interactively selecting surface regions from bone fragments as shown in (a). (b) shows a set of *CT-profile*s that are used to classify bone fragment surface points.

selects surface patches and indicates the correct class for each surface patch. CT-profiles are computed for the surface points in each patch to generate a collection of training vectors. These training vectors are used to compute a surface point classifier as described by the following two steps:

1. An eigenspace is computed that captures the dimension and directions of significant variation within the training data,
2. A minimum-error Gaussian classifier is defined for each collection of training data that uses the eigenspace from (1) to classify a CT-profile to a specific surface type such as periosteal, fracture or articular.

Once computed, the classifier is used to estimate the correct label of surface points coming from the segmented bone fragment surfaces. The following sections describe how steps (1) and (2) above are accomplished.

1. Compute the eigenspace of the feature vectors

As shown in Figure 4.4, CT-profiles may be similar in shape and, in some cases, important differences between the CT-profile classes may be subtle. Direct application of a minimum-error classifier in such cases can lead to numerical instabilities due to an approx-

imately linear dependence between the parameters of the CT-profile classes. Eigenspace methods provide a technique to compute a linear subspace which is better for recognition. This subspace maximizes the variance between the observed training data and eliminates these numerical instabilities by discarding directions in the feature space having little variation, or equivalently, discarding directions in the feature space that provide little discriminative information.

For the experiments of this paper, an eigenspace consisting of 4 dimensions is derived from the CT-profile data which originally consisted of 13 dimensional feature measurements. Only 4 dimensions are used because it was empirically observed that CT-profile measurement data tended to only significantly vary in 4 of the 13 dimensions. From a mathematical perspective, the eigenspace can be viewed as a projective transform where a CT-profile, $\mathbf{x}_i$, is seen as a 13-dimensional vector operated on by a 4x13 dimension matrix, $\mathbf{E}$, providing a 4-dimensional "projected" version of the CT-profile into the eigenspace, $\widetilde{\mathbf{x}}_i$, using the projective transformation $\widetilde{\mathbf{x}}_i = \mathbf{E}\mathbf{x}_i$.

The matrix $\mathbf{E}$ is computed from the CT-profile training data via the following steps:

1. Compute the global mean value of the CT-profiles, $\overline{\mathbf{x}} = \frac{1}{N_t}\sum_{i=1}^{N_t}\mathbf{x}_i$, where N_t denotes the total number of CT-profiles in the training data.
2. Compute the scatter matrix of the CT-profiles: $\mathbf{S} = \sum_{i=1}^{N_t}(\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^t$ (see [61] for a definition of the scatter matrix).
3. Perform an eigenvalue decomposition of $\mathbf{S}$ and define a new matrix $\mathbf{E}^t$ which has as its 4 rows eigenvectors associated with the 4 largest eigenvalues of $\mathbf{S}$. Since the eigenvalues of $\mathbf{S}$ have dimension 13, the resulting matrix $\mathbf{E}$ will have dimension 4x13.
4. The matrix $\mathbf{E}$ then characterizes the projective transformation from the CT-profile space to the eigenspace within which we will perform recognition. Hence, if we have a measured CT-profile, $\mathbf{x}_i$, derived from a bone fragment surface point, $\mathbf{p}_i$, we can compute the eigenspace projection of that point, $\widetilde{\mathbf{x}}_i$, as shown in equation (4.3).

$$\widetilde{\mathbf{x}}_i = \mathbf{E}\mathbf{x}_i \tag{4.3}$$

2. Compute a Gaussian Classifier for each CT-profile

A classical Gaussian classifier is used to compute the likelihood of each surface class given a CT-profile value. Yet, since the data only significantly varies in a 4 dimensional subspace, the classifier is trained on the eigenspace-transformed training data. This data has better numerical properties for accurate classification and requires less computation for each classification. Each collection of transformed training data is used to train a 4 dimensional Gaussian classifier. Using the eigenspace projection **E**, all of the training data for each CT-profile class is projected into the eigenspace using equation (4.3). The mean, μ_k, and covariance matrix, $\sum_k$, of the 4-dimensional Gaussian distributions for the k^{th} collection of CT-profile training data are computed as specified in the equations of (4.4), where N_k denotes the total number of training vectors in the k^{th} collection of training data.

$$\mu_k = \frac{1}{N_k}\sum_{i=1}^{N_k}\widetilde{\mathbf{x}}_i, \;\; \sum_k = \frac{1}{N_k - 1}\sum_{i=1}^{N_k}(\widetilde{\mathbf{x}}_i - \mu_k)(\widetilde{\mathbf{x}}_i - \mu_k)^t \tag{4.4}$$

The 4-dimensional Gaussian distribution in the eigenspace is then given by equation (4.5).

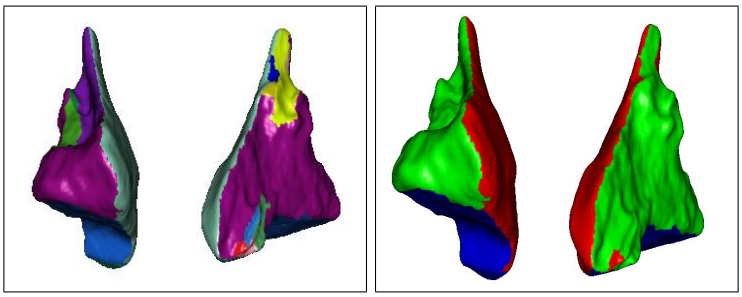
$$p(\widetilde{\mathbf{x}}_i|\omega = \omega_k) = \frac{1}{(2\pi)^2|\sum_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(\widetilde{\mathbf{x}}_i - \mu_k)^t \sum_k^{-1}(\widetilde{\mathbf{x}}_i - \mu_k)} \tag{4.5}$$

Equation (4.5) is the likelihood distribution for the eigenspace feature vector $\widetilde{\mathbf{x}}_i$ given that it is a member of the CT-profile class ω_k.

4.2.3 Computing the Solution

The CT-profile appearance model is integrated with the standard ridge walking algorithm as a post-processing step. Specifically, the ridge walking algorithm is run to divide the bone fragment surface into parts along convex ridge-like contours of the surface. After this, each surface patch is processed to compute a surface type label for the patch. This is accomplished by computing a surface type for each point within the surface patch and then integrating the point labels to estimate a single surface type label for the entire patch. The solution is then computed by merging adjacent surface patches having the same type label. The details of the process are provided in the list below.

1. For each surface point, compute the CT-profile, $\mathbf{x}_i$, and project the CT-profile into the



(a) (b)

Figure 4.5: Segmentation results using different attributes of the model. (a) shows two views of the segmentation results using geometric attributes of the model, (b) shows two views of the classification results for each segmented surface patch, red shows the periosteal surface, green shows fracture surface, and blue shows the articular surface.

computed eigenspace, $\widetilde{\mathbf{x}}_i = \mathbf{E}\mathbf{x}_i$.

2. Find the most likely class given the observed CT-profile by choosing the class $\hat{k}$ as indicated in equation (4.6).

$$\hat{k} = \max_k p(\widetilde{\mathbf{x}}_i | \omega = \omega_k) \tag{4.6}$$

3. The classification for the entire surface patch is determined by a majority vote of individual point classifications within the surface patch.

4. Adjacent surface patches that have the same label are merged.

Figure 4.5(a) shows the segmentation results using the ridge walking algorithm alone, figure 4.5(b) shows the segmentation results after post-processing the geometrical segmentation results of figure 4.5(a) using the proposed appearance-based classifications for the bone fragment surface points.

4.3 Results

Figure 4.6(a,b) show two views of the classification results for the fracture case shown in figure 4.1(a). In figure 4.6(b), the fragments have been slightly displaced for purposes of visualization. This fracture includes 12 fragments. Of the 12 fragments, 6 may be clearly seen and are of sufficient size to include both periosteal and fracture surfaces. In general, the classification of the larger bone fragment surfaces is accurate, particularly

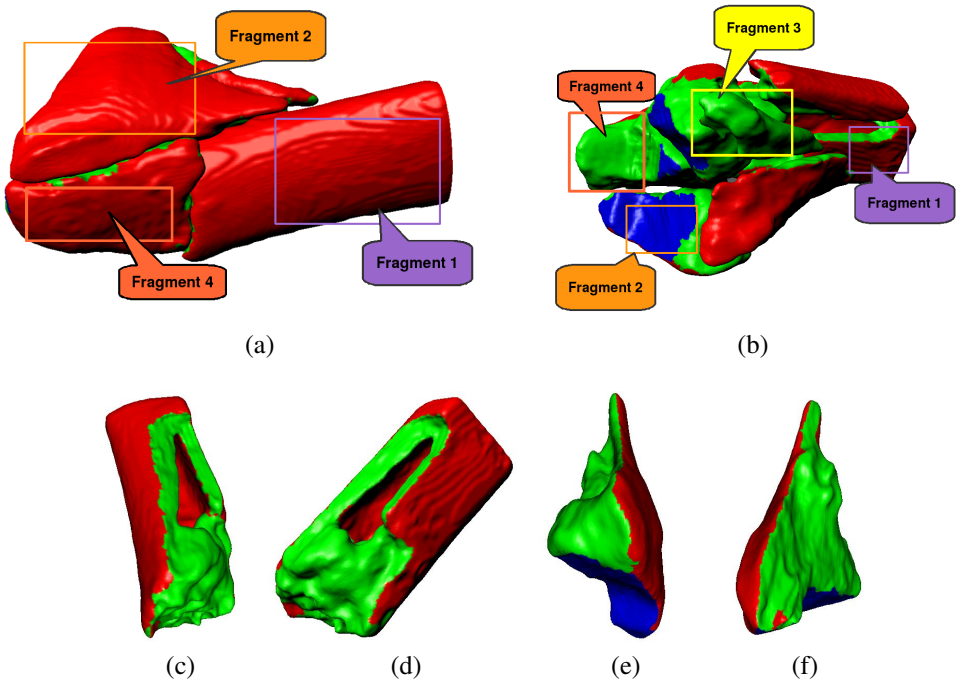


Figure 4.6: Classification results for the fracture case shown in Figure 4.1(a). Accurate results are obtained for fragments of considerable size and thickness such as fragments 1 and 2 shown in (c,d) and (e,f) respectively. However incorrect classifications can occur in regions where the bone surface is thin as is the case for fragments 3 and 4 highlighted in (a,b) (see § 4.3 for further details).

for fragments having considerable size and thickness as is the case for fragments 1 and 2 shown in figure 4.6(a,b). However, our appearance model breaks down on some of the smaller fragments in the fracture case. Specifically, we highlight two fragments, labeled as Fragment 3 and Fragment 4, in Figure 4.6(b) that show incorrect classifications. For fragment 3 the periosteal surface has been misclassified as fracture surface and for fragment 4 the articular surface has been misclassified as fracture surface. Such misclassification errors typically arise in surface regions where the fragment is thin which can corrupt the computed appearance models (the CT-profiles) and result in incorrect classifications.

4.4 Conclusion

This chapter described a new 3D surface partitioning algorithm for the purpose of classifying bone fragment surfaces segmented from CT data as a preliminary step toward computa-

tional bone reconstruction. The algorithm consists of two steps: (1) a geometric partitioning of the surface via ridge-walking algorithm and (2) an appearance-based classification that uses CT voxel intensities in the vicinity of each surface point to classify each surface patch. Each surface is classified to be one of the following classes: (i) periosteal surfaces, defined as surfaces that were part of the original periosteal bone exterior, (ii) fracture surfaces, defined as new surfaces generated when the bone was fractured, and (iii) articular surfaces, defined as surfaces from the bone/cartilage interface at a joint. Application of the existing geometric ridge walking segmentation method alone provides unsatisfactory results. Specifically, bone fragment surfaces are over-segmented and the segmented patches do not provide a medically relevant semantic meaning for the surfaces.

The proposed approach extends the geometric ridge walking algorithm to incorporate appearance information into the segmentation result. This is accomplished by computing appearance features referred to as CT-profiles at each surface point. These CT-profiles are used to infer a semantic surface type for each segmented surface patch. This leads to segmentations that include important medically relevant information that can not be accurately inferred from the geometry alone.

CHAPTER 5: RIDGE WALKING SEGMENTATION OF 2D IMAGES

In this chapter, the 3D ridge walking segmentation algorithm is adapted for the purpose of segmenting 2D images. The 2D image data is represented as a collection of pixel values measured on an (x,y) grid having the form in $I = f(x,y)$, where pixel value, I, may be a scalar value, e.g., a gray scale intensity, or a vector value, e.g., a color as represented by a triplet of red, green and blue components, (r,g,b).

Image segmentation is a fundamental problem in image processing that is important for many applications [62]. For example, segmentation of the organs from an ultrasound image for medical diagnosis [62], or segmentation of faces or the objects within images for recognition [63]. Generally, segmentation approaches attempt to separate pixels into connected groups that share some common semantic property. Many segmentation approaches detect segments by finding spatial regions which typically have low image intensity differences within the regions, while the boundaries between adjacent regions have large intensity differences [64, 65, 62].

The approach applied to adapt the ridge walking algorithm for the purpose of image segmentation is similar in many ways to that used for 3D surface segmentation. When applied in 3D, the ridge walking algorithm tries to find ridges and valleys of the 3D surfaces. These ridges and valleys define boundaries which delimit more smooth homogeneous surface regions. Similarly, for 2D images, the ridges and valleys correspond to locations where the image pixel intensities exhibit large curvatures. Hence, these images can be segmented using similar methods as those used for 3D surfaces.

5.1 Previous Work

Image segmentation has been a core topic of image processing and computer vision research for more than 20 years, and there have been a very large number of published approaches that attempt to solve this difficult problem. We group these methods into four categories: (1) pixel clustering methods, (2) region-growing methods, (3) graph-based methods, and (4) contour-based methods. A brief review of work and approaches from each category are described in the following sections:

1. Pixel clustering methods

Pixel clustering methods typically proceed by computing image features for each pixel and then clustering approaches are used to group similar image pixels together to form a collection of image regions. The pixel features and the similarity between two features are often based on the intensity, color, texture, gradient and the spatial location of the pixels. Thresholding is one common pixel intensity-based clustering method. Thresholding approaches use the image histogram to segment the pixel data into two groups [66]. The image histogram is a function that counts the number of pixels for each pixel intensity value in the image. Thresholding approaches seek to group similar pixels by choosing a threshold value from the x-axis that splits the histogram of pixel intensities into two semantic parts. These values are often chosen to occur at local minima of the histogram [66, 67, 68]. The thresholding method is effective when the objects in the image have different intensity values. However, this method does not consider the spatial positions of the pixels. As such, this method often segments objects into regions that are not connected.

The K-means algorithm [69] is another popular clustering method used for segmentation. The approach starts with K representative seed pixels which define the initial K clusters. More pixels are iteratively assigned to the closest representative pixel cluster using a similarity metric. Each time the pixels are assigned to a cluster, the cluster center is adjusted to lie at the mean value of the pixels assigned to the cluster. The iterations stop when there are no pixels left to assign. The K-means algorithm is simple and efficient,

but the user has to pick these seed pixels of the clusters at the initial step, and the final segmentation results will greatly be affected by the choice for the seed pixels. Another shortcoming of this approach is that, similar to the thresholding approach, the K-means algorithm often generates spatially disconnected regions.

The mean shift algorithm [70] is similar to the K-means algorithm, but it uses search windows to cluster similar pixels. Initially, the search windows are located uniformly in the image. For each search window, the mean-shift algorithm iteratively recomputes the mean position of the image pixels in the search window and shifts the center of the search window to the new mean position until the length of the shift is below some threshold. Search windows whose mean positions shift to lie at similar positions are merged together. The resulting set of clusters serve to segment the image. The mean shift algorithm is computationally expensive and the output depends on the size of search window.

2. Region-growing methods

Region-growing method such as that in [71] segment images by specifying an initial set of seed pixels within an image. Pixels adjacent to the seed pixels are then merged into the seeded pixel regions using a similarity function. The similarity function is defined by the user and typically depends on the pixel intensity value, texture, and color. The algorithm stops when all the pixels in the image have been merged into one of the seeded regions. The region growing method tends to perform well when the image data has clear edges. However, the region-growing algorithm is time-consuming, and the segmentation results are dependent on the user-specified seed pixels. Several enhanced versions of this approach have been proposed by researchers. For example, the work in [72] uses a Harris corner detector to find seed pixels automatically. The work in [73] develops a region growing algorithm that learns a homogeneity criterion automatically from statistical characteristics of the image data.

The watershed segmentation algorithm is another region-growing method. The watershed method was originally proposed in [31] when it was used for grayscale image seg-

mentation. The algorithm operates on a height function defined on the image pixels. The specific choice of the height function depends on the application. Some algorithms use the pixel intensity value as the height [32], other algorithms use the magnitude of gradient at each pixel as the height [33]. The watershed method derives its name from a metaphor that likens the image height function to a topographic relief. The topographic relief is referred to as: a landscape which the algorithm floods with water. Watersheds are the lines that must be introduced to separate different regions which collect the rain water [34]. Water accumulates at local minima of the landscape. Eventually pools of water from different minima merge and dams are built where water from these different minima meet [35]. This resulting collection of dams define contours that partition the landscape into regions called watersheds.

3. Graph-based methods

Image segmentation techniques that use this approach represent the image as a graph. The work in [74] develops an efficient image segmentation algorithm, where the graph nodes are pixels and graph edges exist between nodes associated with adjacent pixels. The weight of each edge is based on difference between the pixels values that it connects. The approach partitions the image using the criterion that any pair of adjacent regions should have larger variations between the regions than the variations present within the regions. The minimum cut algorithm [75] uses the weight (capacity) of the edge to reflect the similarity between pairs of adjacent pixels. Segments are formed by deleting graph edges such that the sum of the edge weights for the deleted edges has minimum total weight. Segmentations can be computed efficiently using the minimum cut algorithm. However these segmentations tend to produce short boundaries which sometimes generate trivial partitions. The normalized cut algorithm [5] overcomes these problems by normalizing the edge weight, so that the cut criterion considers both the dissimilarity between the different groups as well as the similarity within the groups.

4. Contour-based methods

Contour based segmentation approaches use curves (2D) or surfaces (3D) to approximate the boundary of image objects. These deformable curves or surfaces evolve (deform) using an energy function that incorporates the image intensities to evolve the boundaries of the contour to lie at positions that outline objects in the image [76, 47]. These energy functions typically consist of two terms: (1) a smoothness term whose magnitude becomes large for non-smooth boundaries and (2) a data term whose magnitude becomes large when the boundary is far from the object boundaries. Deformable contour approaches are popular for segmenting objects in CT (compute tomography), X-ray, and magnetic resonance (MR) images. In these context, they have been used to segment brain tumors, kidneys and lungs within medical images [77]. However, the deformable contours approach suffers several shortcomings: (1) it is very sensitive to noise, (2) the result can vary significantly depending on the initial contour, and (3) it may converge to local minima in the energy function giving an undesirable boundary.

The level set algorithm [78] is a specific type of deformable model, which improves over some shortcomings in previous deformable contour methods [79, 80]. The main idea of the level set algorithm is to represent a contour as the zero level set of a higher dimension function, which is called the level set function. The level set algorithm tries to find object boundaries in the image by formulating the motion of the contour as the evolution of the level set function.

In contrast to existing image segmentation algorithms as discussed above, our method fuses graph and contour based methods. In our graph, the weight of a graph edge is based on the principal curvatures and principal directions of curvatures measured at the two end points of the edge. However, instead of cutting the graph by deleting edges having small weights, our segmentation algorithm connects graph edges that have large weights to form contours. These contours end up defining the segmentation boundaries. By solving for the image contours directly, new capabilities are provided that allow one to control the form of the segmentation solution in ways that cannot be easily controlled with region-growing

or pixel clustering methods. This includes controlling the length, shape, and salience of the segmentation boundaries. When compared against other contour-based algorithms, the ridge walking algorithm avoids the complex edge linking process often associated with other contour-based approaches such as edge-based segmentation methods.

5.2 Methodology

To apply the 3D ridge walking algorithm to a 2D image, the image intensity surface is modeled as a Monge patch having the parametric form $M(x,y) = (x,y,f(x,y))$, where (x,y) denotes the pixel location, and $f(x,y)$ denotes the pixel intensity value at that location. Using the Monge patch representation, the principal curvatures and principal directions needed by the ridge walking algorithm can be computed. As in 3D, the segment boundaries in the image traverse the ridge-like and valley-like structures in the image intensity surface. The process of image segmentation using 2D ridge walking algorithm is as follows:

1. Define a graph over the image (see §5.2.1).
2. Compute the principal curvatures for graph vertices (see §5.2.2).
3. Compute the weight for each edge of the graph (see §5.2.3).
4. Compute the (maximum) spanning tree of the graph based on the weight of edges. The edges not in the spanning tree are called loopy edges. Loopy edges are stored in order of decreasing edge weight into an edge stack.
5. Loopy edges are inserted into the maximum spanning tree. When inserted, each edge creates a new closed contour on the surface. For each new contour, a uniqueness criterion and a curve-length criterion are applied to suppress contours that are nearly equivalent or have very short arc-length.
6. The surface segmentation is computed by finding all contiguous surface regions within the contours created in step 5.

Steps (1)-(3) are preparation steps for image segmentation using ridge walking algorithm. In step (1), a quadrangular graph is defined over the 2D image (see §5.2.1 for details). Step (2) uses a Monge patch representation for the image to compute the principal curvatures of

(a) (b) (c)

Figure 5.1: (a,b,c) show steps in the process for image segmentation. (a) shows is the input image, (b) shows the maximum curvature for each vertex of the image where white indicates highly positive curvature, (c) shows the segmented result using 2D ridge walking algorithm ($\lambda = 0.5$).

the image intensity surface at the node locations ((see §5.2.2 for details). Step (3) computes the weight for each graph edge. Edges that align with ridge-like or valley-like contours in the image have larger edge weights (see §5.2.3 for details). Steps (4)-(6) use the data computed in step (1)-(3) to find the segmentation contours using a technique similar to that used by 3D ridge walking segmentation algorithm (see §2.2.1-§2.2.3 for details). As with the 3D ridge walking algorithm, the input parameter for the algorithm, λ, is the percentage of the total number of loopy edges as described in §2.2.2. This parameter indirectly determines the segmentation resolution for the input surface. Higher values of λ generate more detailed segmentations. A graphical outline of the process is shown in figure 5.1.

5.2.1 Define a Graph over the Image

As with the ridge walking segmentation for 3D surfaces, a graph is defined over the image surface. The graph has a structure that is shown in figure 5.2. This structure creates a graph node for each pixel in the image and a collection of graph edges to connect nodes that are associated with adjacent pixels in the image as depicted in figure 5.2. For the purposes of notation, let $\mathbf{p}_i$ denote the graph node for the (x,y) location of a pixel and let $\mathbf{e}_{ij}$ denote the graph edge that connects the pair of graph nodes, $(\mathbf{p}_i, \mathbf{p}_j)$. Using this notation, let $E = \{\cup_{ij}\mathbf{e}_{ij}\}$ denote the set of all graph edges and let $P = \{\cup_i \mathbf{p}_i\}$ denote the set of all

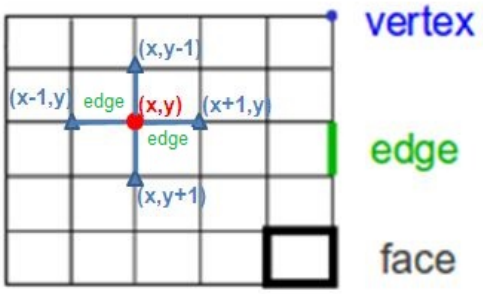


Figure 5.2: This image shows the structure of the graph defined over the 2D image. For each pixel, a node in the graph is created (shown as a circle on the grid). Each node is connected by a graph edge to the 4 pixels that are adjacent (shown as triangles on the grid).

graph nodes. The graph, $G(P,E)$, is then an equivalent representation for the image and is the object of computation for our segmentation approach.

5.2.2 Compute the Curvatures for Graph Vertices

The ridge walking segmentation requires the principal curvatures and directions of principal curvature for each (x,y) pixel to compute a weight for each graph edge. These edge weights are computed from the 2D image assuming that the image intensities are measurements from a Monge patch and are key components for computing the segmentation boundaries.

The principal curvatures of a Monge patch surface are computed from the partial differentials of the parametric equations of the Monge patch representation. One popular approach for curvature computation uses the Hessian matrix of the surface. The Hessian matrix is a square matrix of second order partial derivatives of a surface function $f(x,y)$. This matrix characterizes the local curvature of the function as shown in equation (5.1).

$$H(x,y) = \begin{bmatrix} f_{xx}(x,y) & f_{xy}(x,y) \\ f_{xy}(x,y) & f_{yy}(x,y) \end{bmatrix} \tag{5.1}$$

In equation (5.1), $f_{xx}(x,y)$ and $f_{yy}(x,y)$ represent the second order central partial differentials of the function $f(x,y)$ in the x and y directions. The eigenvalues and eigenvectors of the Hessian matrix at each (x,y) point provide estimates of the principal curvatures and the

directions of these principal curvatures on the surface respectively. However these eigenvalues cannot be used as edge weight values by the ridge walking algorithm. The limitation in using these value is that the curvatures computed in this way are different from the true principal curvatures by a constant scale factor that changes at each surface point.

Accurate values for the principal curvatures and principal directions are computed using local discrete approximations of the first and second fundamental forms at each surface point. These forms depend on the first and second partial derivatives of the parametrically defined Monge patch surface which are shown in equations (5.2) and (5.3) respectively.

$$M_x(x,y) = (1,0,f_x(x,y)),\ M_y(x,y) = (0,1,f_y(x,y)) \tag{5.2}$$

$$M_{xx}(x,y) = (0,0,f_{xx}(x,y)),\ M_{yy}(x,y) = (0,0,f_{yy}(x,y)),\ M_{xy}(x,y) = (0,0,f_{xy}(x,y)) \tag{5.3}$$

Equation (5.2) and (5.3) use $f_x(x,y)$ and $f_y(x,y)$ to denote the discrete partial differentials of the function $f(x,y)$ in the x and y directions.

The first fundamental form characterizes the orientation of the tangent plane of the surface at each data point. This information is encoded as a collection of three coefficients, E, F, and G, which collectively specify the coefficients of the first fundamental form. These coefficients can be computed as shown in equations (5.4)-(5.6).

$$E = M_x(x,y) \cdot M_x(x,y) = 1 + f_x^2(x,y) \tag{5.4}$$

$$F = M_x(x,y) \cdot M_y(x,y) = f_x(x,y) f_y(x,y) \tag{5.5}$$

$$G = M_y(x,y) \cdot M_y(x,y) = 1 + f_y^2(x,y) \tag{5.6}$$

Geometrically speaking, the coefficient E is the squared magnitude of the vector obtained as partial differential of the parametric surface along the x-axis. The coefficient F is the inner product of the vectors obtained as partial differential of the surface along the x-axis and y-axis. The coefficient G is the square magnitude of the vector obtained as partial differential of the surface along the y-axis. As such, these coefficients describe the magnitude of the slope of the parametric surface in 3 different directions: (1) E characterizes the slope in the direction of the x-axis, (2) G characterizes the slope in the direction of the y-axis, (3) F characterizes the slope in the direction at a 45 degree angle to both the x and y axis.

The second fundamental form characterizes how much the surface bends away from the tangent plane, i.e. how fast the slope of the tangent plane is changing in the different directions. This information is encoded as a collection of three coefficients, e, f, and g, which collectively specify the coefficients of the second fundamental form. The second fundamental form depends on the normal of the surface which can be computed as shown in equation (5.7).

$$\mathbf{n} = \frac{M_x(x,y) \times M_y(x,y)}{\left\| M_x(x,y) \times M_y(x,y) \right\|} \tag{5.7}$$

Using this definition for the normal of the surface, the three coefficients can be computed as shown in equations (5.8)-(5.10).

$$e = M_{xx}(x,y) \cdot \mathbf{n} = \frac{f_{xx}(x,y)}{\sqrt{1 + f_x^2(x,y) + f_y^2(x,y)}} \tag{5.8}$$

$$f = M_{xy}(x,y) \cdot \mathbf{n} = \frac{f_{xy}(x,y)}{\sqrt{1 + f_x^2(x,y) + f_y^2(x,y)}} \tag{5.9}$$

$$g = M_{yy}(x,y) \cdot \mathbf{n} = \frac{f_{yy}(x,y)}{\sqrt{1 + f_x^2(x,y) + f_y^2(x,y)}} \tag{5.10}$$

Geometrically speaking, the coefficient e is the projection of the second order partial differential of the parametric surface in the direction of the x-axis onto the surface normal. The coefficient f is the projection of the vector given by taking the second order partial differential of the parametric surface in the direction of a vector pointing at 45 degree angle to the x-axis onto the surface normal. The coefficient g is the projection of the second order partial differential of the parametric surface in the direction of the y-axis onto the surface normal. These coefficients describe how fast the tangent plane is changing as it leaves the tangent plane in 3 different directions: (1) e characterizes how much the surface is bending in the direction of the x-axis, (2) g characterizes how much the surface is bending in the direction of the y-axis, (3) f characterizes how much the surface is bending in the direction at a 45 degree angle to both the x and y axis.

The Weingarten matrix combines the first and second fundamental forms into a single shape operator. The shape operator describes how the surface bends away from the tangent

plane at each point. The Weingarten matrix is expressed in equation (5.11).

$$S = \frac{1}{(EG-F^2)} \begin{bmatrix} eG-fF & fG-gF \\ fE-eF & gE-fF \end{bmatrix} \tag{5.11}$$

The principal curvatures and directions of principal curvatures of the Monge patch surface can be obtained as the eigenvalues and eigenvectors of the Weingarten matrix. The Weingarten matrix is scaled by the factor $\frac{1}{EG-F^2}$. This scale factor makes the computation of eigenvalues and eigenvectors numerically correct and is conspicuously absent from the Hessian matrix of equation (5.1). Let S_i denote the discrete approximation of the Weingarten matrix at the surface point $\mathbf{p}_i$. The larger eigenvalue of S_i and its corresponding eigenvector are the value of maximum curvature $(k_{max})_i$ and the direction of maximum curvature $\mathbf{v}_i$ at surface point $\mathbf{p}_i$. The smaller eigenvalue of S_i and its corresponding eigenvector are the value of minimum curvature $(k_{min})_i$ and the direction of minimum curvature $\mathbf{u}_i$ at surface point $\mathbf{p}_i$.

5.2.3 Compute the Edge Weight of the Graph

As with the ridge walking algorithm, a weight is associated with each graph edge as defined by one of three salience functions. These salience functions serve to detect image contours that delimit regions in the image that have significant local intensity changes. Salience functions specify the weight of an edge and incorporate two criterion: (1) the curvature of the image intensity surface in the direction perpendicular to the edge, and (2) how well the edge aligns with a direction of principal curvature. The ridges and valleys in 2D images correspond to locations where the image pixel intensities exhibit large changes, we utilize the salience function as provided in equation (5.12) to segment the images along both ridges and valleys of image intensity surface.

$$w(\mathbf{e}_{ij}) = \begin{cases} \left| \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} \cdot \frac{\mathbf{u}_i+\mathbf{u}_j}{\|\mathbf{u}_i+\mathbf{u}_j\|} \right| \overline{\kappa}_{max} & |\overline{\kappa}_{max}| \geq |\overline{\kappa}_{min}| \\ -\left| \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} \cdot \frac{\mathbf{v}_i+\mathbf{v}_j}{\|\mathbf{v}_i+\mathbf{v}_j\|} \right| \overline{\kappa}_{min} & |\overline{\kappa}_{max}| < |\overline{\kappa}_{min}| \end{cases} \tag{5.12}$$

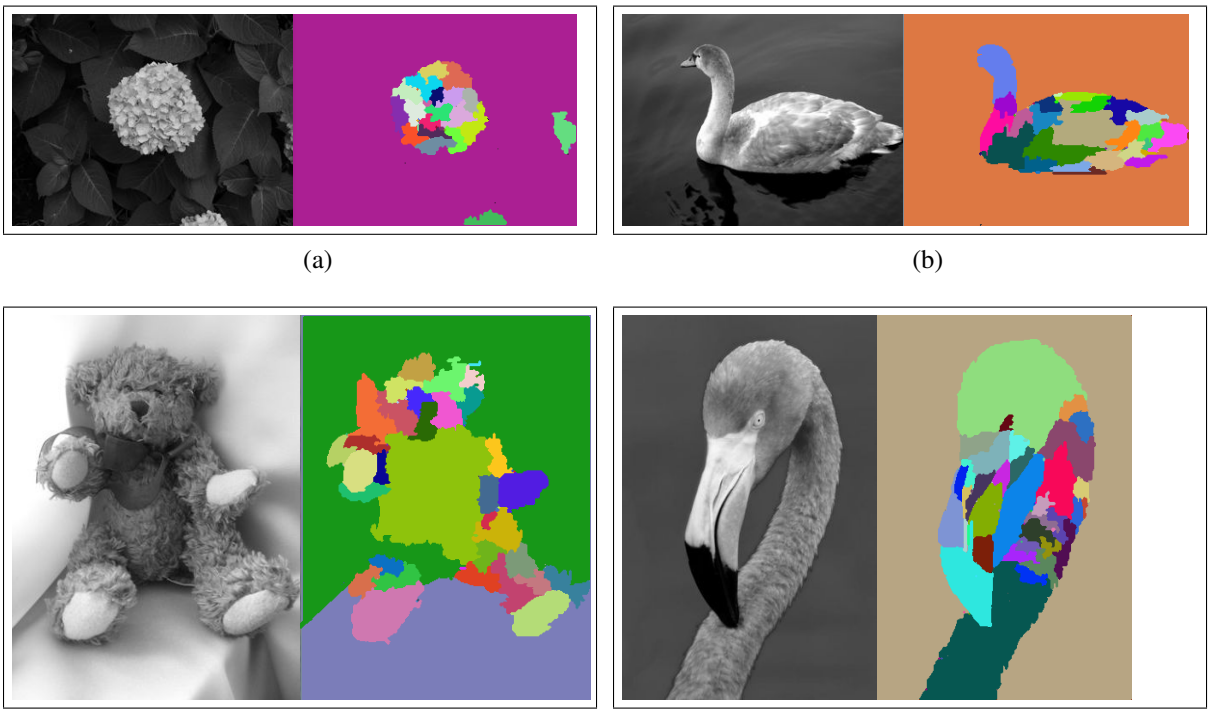
In equation (5.12), the quantity $\left| \frac{\mathbf{u}_i+\mathbf{u}_j}{\|\mathbf{u}_i+\mathbf{u}_j\|} \cdot \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} \right|$ represents how well the direction of the edge agrees with the estimated direction of the ridge-line on the surface (this is the direction

of minimum curvature for convex regions, $\mathbf{u}_i$), and the quantity $\left| \frac{\mathbf{v}_i+\mathbf{v}_j}{\|\mathbf{v}_i+\mathbf{v}_j\|} \cdot \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} \right|$ represents how well the direction of the edge agrees with the estimated direction of the ridge-line on the surface (this is the direction of maximum curvature for concave regions, $\mathbf{v}_i$). The values $\overline{\kappa}_{max} = \frac{(\kappa_{max})_i+(\kappa_{max})_j}{2}$ and $\overline{\kappa}_{min} = \frac{(\kappa_{min})_i+(\kappa_{min})_j}{2}$ are approximations of the principal curvatures averaged over the extent of the graph edge $\mathbf{e}_{ij}$.

5.3 Initial Results

Figure 5.3 shows segmentation results for four different images using the extended ridge walking algorithm. The left image in each box is the input image. The right image in each box is the segmented result. Different segmented regions are shown in different colors. The single input parameter of ridge walking algorithm is λ, which is set to 0.3 for all results shown in figure 5.3. Figure 5.3(a) shows that the algorithm can segment the flower as a collection of regions distinct from the background. Yet, the flower itself is segmented into several parts due to the pixel intensity variations present within the flower. In figure 5.3(b), the swan is segmented into several parts according to the pixel intensity changes of its feathers. However, the beak of the swan is not segmented as a distinct part. This may due to the constraint included in the ridge walking algorithm that filters out the short contours. In figure 5.3(c), the body, head and legs of the bear are segmented as separate regions, but the head of bear is segmented into several parts where these parts are not semantically meaningful. In figure 5.3(d), the neck, beak and head of the bird are segmented as separate regions, but the neck is segmented into several parts due to pixel intensity variations present therein.

Figure 5.4 shows image segmentation results for three different images by four different image segmentation methods. They are shown together to allow for visual comparison of the results and to better understand the strengths and weaknesses of the 2D ridge walking algorithm. The first row of figure 5.4 shows the original input images, the remaining rows of figure 5.4 shows segmentation results for each of the four different methods. These methods, from top-to-bottom, are (1) a human generated segmentation, (2) a ridge walking



(a) (b)

(c) (d)

Figure 5.3: The segmentation results for ridge walking algorithms. In each box, the left image is the input image and the right image is the segmentation result.

segmentation, (3) a mean shift segmentation [70] and (4) a graph based segmentation [74]. The segmentation results for these algorithms are shown in rows (2), (3), (4) and (5) of figure 5.4 respectively.

The human generated segmentations are treated as "ground truth" and serve as the benchmark against which the other segmentations are compared. The human generated segmentations are taken from the human image segmentation database published by D. Martin and et. al. from Berkeley University in [81]. The figure shows each ground truth segmentation as a collection of boundaries where the boundary is given an intensity attribute. Boundaries in these images that have higher intensities indicate that the human segmentations agree on the location of the boundary within the image.

In order to compare the segmentation algorithms fairly, the same parameter setting was used for all the test images for each algorithm. For the ridge walking algorithm described

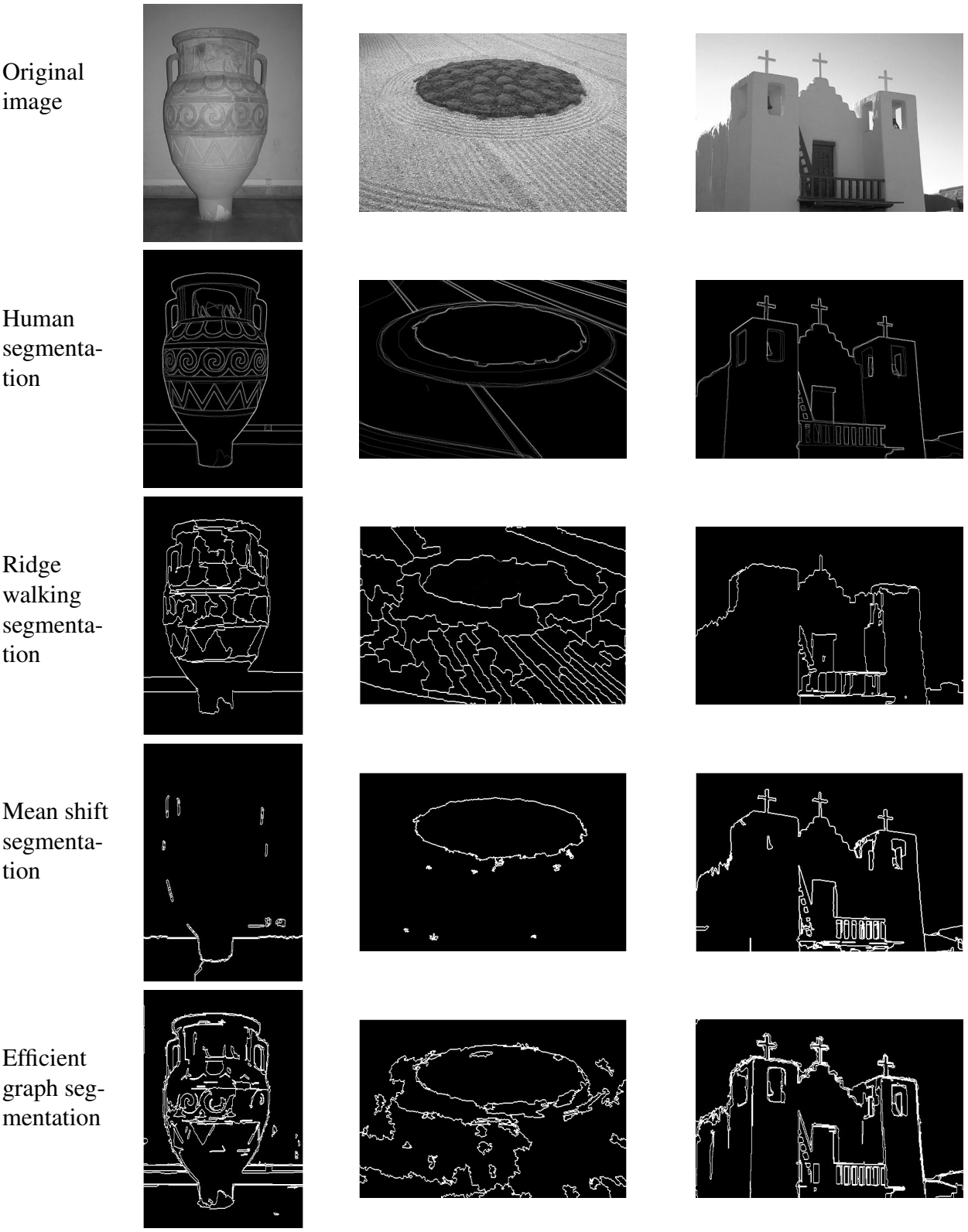


Figure 5.4: Image segmentation results for three different images using different algorithms. The first row shows the input images. The remaining rows show segmentation results generated by humans, the ridge walking algorithm, the shift algorithm [70] and graph based algorithm [74] respectively.

in this chapter, the only input parameter is the λ which was set to 0.3 for all the test images. For the mean shift algorithm [70], the input parameters are: (1) the feature (range) bandwidth, h_r, (2) the spatial bandwidth, h_s, and (3) a minimum segment size (in pixels), M. The two parameters $[h_s, h_r]$ jointly determine the resolution of the segmentation. These parameters were set to $(h_s, h_r) = (16, 7)$, as suggested by the author in [70]. The parameter, M, determines the minimum size of a segment and is used to discard small image regions. For all tests conducted, M was set to the value 20. The graph based algorithm [74] requires two input parameters: (1) the scale of observation, k, and (2) the minimum segment size, M. The parameter settings recommended by the author, k=300 and M=20, were used to segment the test images.

The segmentation results in figure 5.4 show that no single algorithm can work well on all of the images. For example, the mean shift algorithm does not work well on the vase image using the recommended parameter settings as shown in the first column in figure 5.4. This may be due to the fact that the intensity differences between the vase body and the background is small which causes the mean shift algorithm to merge these objects. All three algorithms do not work well on textured image shown in the second column in figure 5.4. This may be due to the numerous local variations in intensity present in the texture image that tend to make these algorithms generate many small segments. The ridge walking algorithm can segment the castle into meaningful parts, but does not separate the small structures in the castle image (the third column in figure 5.4) into meaningful parts. For example, the cross symbols on the top of the castle are not segmented out. This behavior may be caused by the constraint enforced on the contour length in ridge walking algorithm which may delete these short contours.

In §3.1, we discussed four evaluation metrics to evaluate how well the segmentations generated by algorithms agree with the ground truth segmentations. They are: (1) the Cut Discrepancy, (2) the Hamming Distance, (3) the Rand Index, and (4) the Consistency Error. The same metrics can be for evaluation of 2D image segmentation algorithms. However,

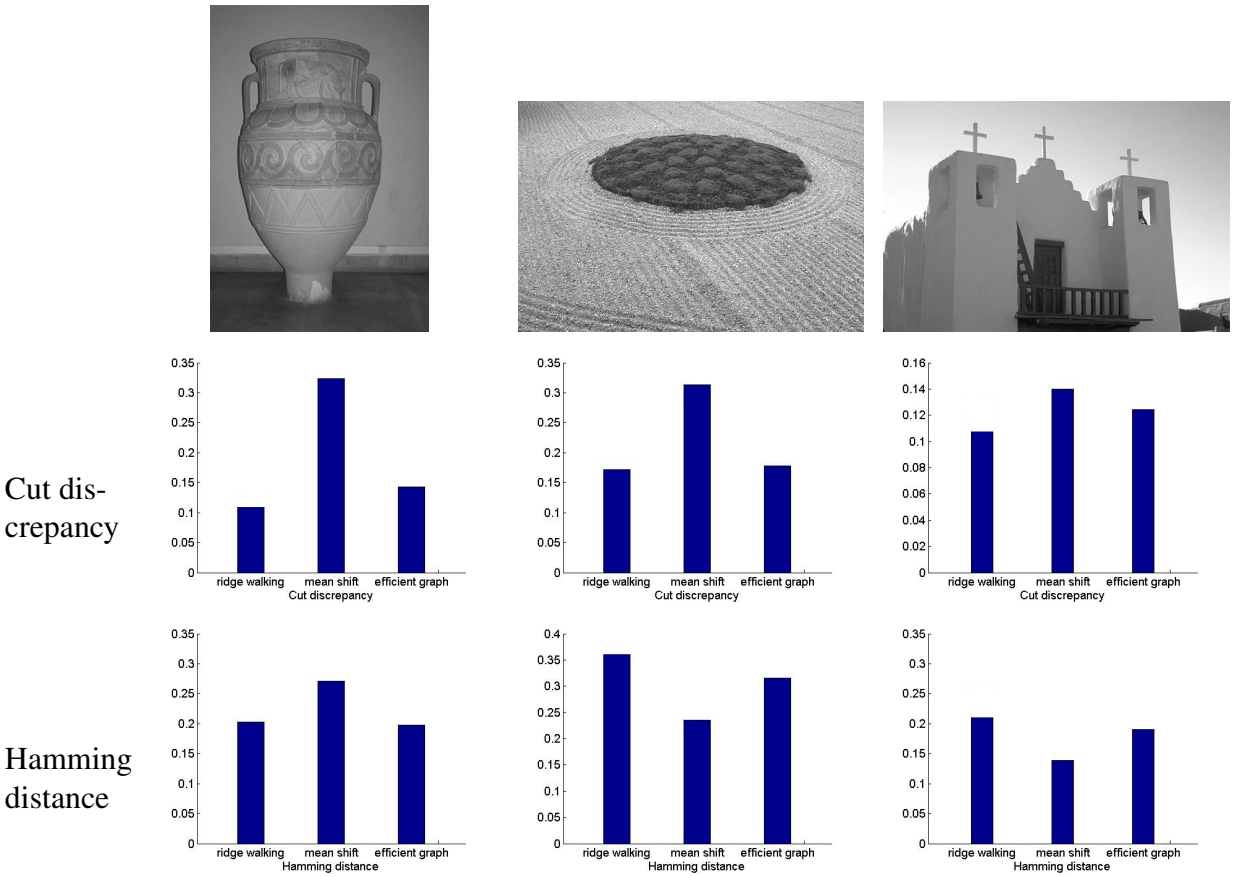


Figure 5.5: Quantitative evaluation of the ridge walking algorithm is accomplished by comparing its results with the ground truth results. Two other algorithms, mean shift algorithm [70] and graph based algorithm [74], are similarly evaluated. The first row shows the input image. The second and third rows show the cut discrepancy and Hamming distance scores for these three algorithms (smaller values indicate better results).

for this evaluation, only two of the four metrics were used: (1) the Cut Discrepancy (CD) and (2) the Hamming Distance (HD). These two evaluation metrics measure the difference between two segmentations by comparing their boundaries (CD) and by comparing the contents of their segmented regions (HD). The Rand index is not selected because the number of possible pixel pairs for a image is a large number which makes the computation of Rand index very time consuming. The Consistency error is not selected because the three automatic image segmentation algorithms selected in this paper do not produce hierarchical segmentation results.

An initial evaluation of 2D ridge walking algorithm was performed. The evaluation compares segmentation results produces by three different algorithms with ground truth segmentations using the two selected metrics. The initial evaluation examines the performance of these algorithms for three different input images. Figure 5.5 shows the evaluation results. The first row of figure 5.5 shows the input images. The second row of figure 5.5 shows the cut discrepancy score for each of the algorithms. The third row of figure 5.5 shows the Hamming distance scores for each of the algorithms. The lower the CD and HD values are, the better the segmentation results agree with the human segmentations.

The results for the vase image are shown as the left-most column of figure 5.4. One can see here that the ridge walking algorithm has smaller CD and HD values than other two algorithms. This indicates that the segment boundaries and regions generated by the ridge walking algorithm for this image are more similar to the human segmentations than the other two algorithms. This is also reflected in the ridge walking segmentation result shown in the first column and third row of figure 5.4 where one can that vase has been segmented from the background and that the textures within the vase have also been delineated. The evaluation results also suggest that the mean shift algorithm performs poorly for the vase image as suggested by large CD and HD values.

For the image of a field shown in the middle column of figure 5.4, all of the algorithms have large CD ($\geqslant 0.15$) and HD ($\geqslant 0.25$) values. These results indicate that all of the

algorithms have a tendency to over-segment this image. For this image, the ridge walking algorithm has the worst performance with largest HD value, which indicates that the ridge walking algorithm may not work well on the images that include textures having large intensity variations.

For the castle image shown in the right-most column of figure 5.4, the CD values for all of the algorithms seems similar. However, the ridge walking algorithm has the largest HD value. These results may indicate that the ridge walking algorithm may not capture small structures present in images.

5.4 Findings

This chapter describes the adaption of the ridge walking segmentation algorithm into 2D images. Toward this end, the intensity image was represented by a Monge patch and methods were given to compute the principal curvatures and principal directions of each image pixel of this surface. The ridge walking algorithm can detect the ridges and valleys of the image, which are used to compute the segmentation boundaries. As an initial experiment, the 2D ridge walking segmentation algorithm was evaluated using 3 test images. The segmentation results were compared with a ground truth segmentation of these images and with results generated from two other algorithms to understand the relative strengths and weaknesses of the ridge waking algorithm on these 3 images. Future work will include a more comprehensive evaluation that includes more test images to generalize the behavior and better characterize the potential advantages of the 2D ridge walking algorithm in this context.

CHAPTER 6: APPLICATION: A SYSTEM FOR GEOLOGICAL ANALYSIS OF 3D MARTIAN DATA

In this chapter, a software system is described that enables geologists to find and extract geometric information from images of the Martian surface recorded by the Mars rovers Spirit and Opportunity. The software is named "ImageRover" and provides researchers computational tools to visualize and analyze images of the Martian surface. Software to facilitate this analysis is important because existing tools make these tasks difficult to accomplish. Without tools such as this, many hours would be spent exploring the very large collection of Martian images (>200K images) for images of interest. Even when an image of interest has been found, existing software packages do not provide the tools necessary to enable geologists to extract geometric measurements directly from the image data directly. The "ImageRover" system seeks to fill this need.

Images from the Mars rovers are transmitted continuously to the Earth and reconstructed as raw images referred to as an EDR (Experimental Data Record) images. Among the EDR images, EFF (Experimental Full Frame) images are 1024X1024 images collected by two high-resolution cameras on the rover imaging device called the "Pancam". EFF images are often processed using a suite of software tools. These tools take as input one or more EDR images and integrate the measured image data into a new output image referred to as a RDR (Reduced Data Record). The variety of different RDRs is numerous. Of specific interest for this system are XYZ RDRs which contain 3D stereoscopic reconstructed image data that provide 3D (x, y, z) surface position measurements for objects viewed within a stereoscopic pair of EDR images [24].

The "ImageRover" software provides users three methods for finding images of interest within the large collection of images collected by the Mars rovers. These methods are

intended to enable users to quickly locate image data with queries that are based on time, spatial location or desired image content. The time querying method takes as input a list of one or more sols, i.e., the solar day that the data was collected. The space querying method takes as input a list of one or more sites, i.e., designated geographical locations for each rover during the mission. The image content querying method takes as input a list of one or more exemplar images and the computer searches for images that have content similar to the input images. For example, if a user wishes to find images that contain sand dunes, a query would consist of one or more representative images of sand dunes.

The "ImageRover" software provides visualization tools that enable users to view 2D EFF and 3D XYZ stereoscopic images of the Martian surface (see §1.2.4 for details). The 2D and especially the 3D stereoscopic images provide important information to users. The visualization tools allow researchers to visually inspect 3D images of the Martian surface with unprecedented ease and detail.

The "ImageRover" software also provides analysis tools that allow users to take geometric measurements from images of the Martian surface. Measurements of interest are typically geological and include length and orientation measurements of surface features such as rocks and cracks within rocks. These measurements can allow scientists to better understand the geological processes that shaped the Martian landscape. They can also contribute to answering other important questions such as determining when and where liquid water existed on Mars and if these environments were once capable of supporting life.

The ridge walking segmentation algorithm is used in "ImageRover" software to help extract geometric objects of interest within the 3D Martian surface data. Towards this end, the ridge walking algorithm computes regions by dividing the surface within contours that follow concave or convex structures of the surface. Concave surface regions tend to delimit the imaged rocks and other objects that protrude from the ground surface. Convex surface regions tend to delimit objects into parts and may correspond to cracks within the rocks present in the image. These structures are important in analyzing the geometry of a rock

on Martian surface, i.e., its surface size.

The description of the "ImageRover" software is split into two parts. The first part introduces the interface of the software (discussed in §6.2), the second part introduces the underlying algorithms behind the interface (discussed in §6.3), .

6.1 Previous Work

NASA currently has a large suite of software system developed in-house for the analysis of all mission data. These software packages are components of the PDS (Planetary Data System) [82] and provide access to this data. Some of these software packages provide general data processing tools such as BulkDownloader [83], NASAVIEW [84], and ISIS (Integrated System for Images and Spectrometers) [85]. A smaller subset of software packages such as the MER (Mars Exploration Rover) Analyst Notebook and the Planetary Image Analysis are specifically designed to aid the user in visualizing MER images. All of these packages are available, although not uniformly, for varying operating systems (Linux, Windows, etc.) or through the web. Despite the availability of this software, some drawbacks still exist for users that wish to extract more than the visual information provided through JPEG versions of MER images. MER Analyst Notebook (MERAN) [86], for example, is the most popular portal to MER data for typical users. MERAN allows the user to browse through MER data by visualizing all images by sol. Once an image is chosen, data accompanying that image are provided as links which can then be downloaded. As such, each image and its accompanying data must be downloaded individually. However, the MER Analyst Notebook does not provide 3D image analysis capabilities. This requires that the user must download, install, and access additional software separately in order to access the 3D image data. Many of these systems, in turn, require platform specific software applications and also require the user to maintain large databases of image data. These tasks alone represent significant hurdles for the non-computer savvy scientist, teacher or general user to overcome, and therefore, in many cases, likely preclude any complex or significant exploitation of the data.

In [87], Frawley et. al. described data mining as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data." The abundance of imaging systems available today are generating unprecedented amounts of data. In many cases researchers need specific information from a small subset of the recorded imagery. Rather than painstakingly reviewing each image manually, data mining algorithms automatically process images to determine those relevant to the researchers need. Implementation of such systems requires image processing and pattern recognition algorithms which compute features and classify pixels within images to determine how relevant an given image is to a researcher's query. Many papers have investigated methods for extracting image features such as wavelets [88], vocabulary trees [89], hierarchical K-means dictionaries [90], and bag-of-features methods [91].

For geological features, there are several methods that have been proposed in the literature. For example, in [92], the authors present a 6-step method for detection of geological features within images from a robotic rover that uses multiple input images of the same surface region and a Bayesian Belief Network to classify rocks within images. The methods proposed therein often make use of color or stereoscopic data which is time consuming to generate if these images are not already available and, in cases where the data is available, the additional information can significantly increase the processing time required to generate results for general queries over a very large number of images. In work in [93] tries to prioritize image data sent from Mars Rovers. To do that, the information that extracted from Mars images such as the location of the rocks and several properties of the rocks such as size, texture and shape are detected. The work in [94] aimed at detecting rocks from MER images. The detection technique in [94] uses a series of denoising operations to remove spurious texture, then uses the Canny edge detection algorithm to detect edges in the image, and then detects rocks as collections of edges that form closed loops in the image. At last, the work in [94] concludes the vast majority of closed edge shapes lies on the surface of rocks or are rocks themselves.

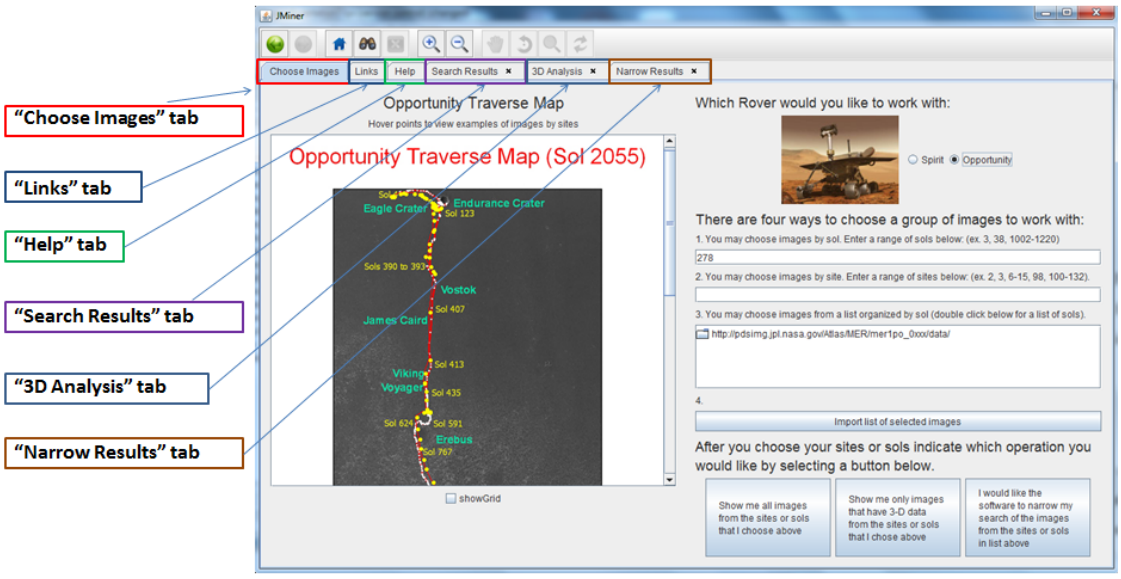


Figure 6.1: The "ImageRover" software interface contains six tabs: (1) the "Choose Images" tab, (2) the "Links" tab, (3) the "Help" tab, (4) the "Search Results" tab, (5) the "3D Analysis" tab and (6) the "Narrow Results" tab.

6.2 The Software Interface

The "ImageRover" software is implemented in Java and can be launched from any Java-enabled computer. To launch the program, users access a web site where the software is available and click on a button to start the program using Java web-start. Java web-start technology provides software developers the ability to deploy software to users and run that software on the user's machine via a single mouse-click.

The "ImageRover" software interface provides several search methods, an interface for data visualization and analysis, and documentation to assist users in operating the software. These software tasks are made available to user in an interface consisting of six different screens that are collected as a sequence of "tabs". All of these tabs are shown in figure 6.1, and a short description for each of these tabs is provided below.

1. The "Choose Images" tab is used to initiate searches to find images of interest. It is shown in figure 6.2 (see §6.2.1 for details),
2. The "Search Results" tab is used to show search results based on either space or time

queries. It is shown in figure 6.4 (see §6.2.2 for details),

3. The "Narrow Results" tab is used to show the search results based on image contents queries. It is shown in figure 6.5 (see §6.2.3 for details),
4. The "3D Analysis" tab is used to visualize 2D EFF and 3D XYZ surface images and take measurements from these images. It is shown in figure 6.6 (see §6.2.4 for details),
5. The "Links" tab shows web links that are useful for understanding the data and software. It is shown in figure 6.7(a) (see §6.2.5 for details),
6. The "Help" tab shows documentation that describes how to use the software. It is shown in figure 6.7(b) (see §6.2.5 for details).

When selected, each tab will expose an application window that provides functionality relevant to the task described on the tab. For example, the "3D Analysis" tab opens a window that allows users to visualize and extract geometric information from 3D XYZ surface images.

6.2.1 "Choose Images" Tab

The Spirit and Opportunity Rovers have collected a large number of images which are publicly available for viewing and analysis. Because of the relatively large file size for each image, it is not practical to view all images at once. The goal of the "Choose Images" tab is to provide a variety of methods to users for efficiently finding the images that they seek. The two conventional methods for finding images are searching by the time the image was recorded or by the place the image was recorded. The interface shown in figure 6.2 provides methods for users to specify this information. The screen is divided into two parts:

1. The traverse map (left side of the screen),
2. The options for browsing images (right side of the screen).

On the left side of the screen, the path of the rover across the Martian surface is displayed on top of a satellite image (outlined in blue). This is called the traverse map of the rover. On the right side of the screen, a collection of fields are shown which are modified by the

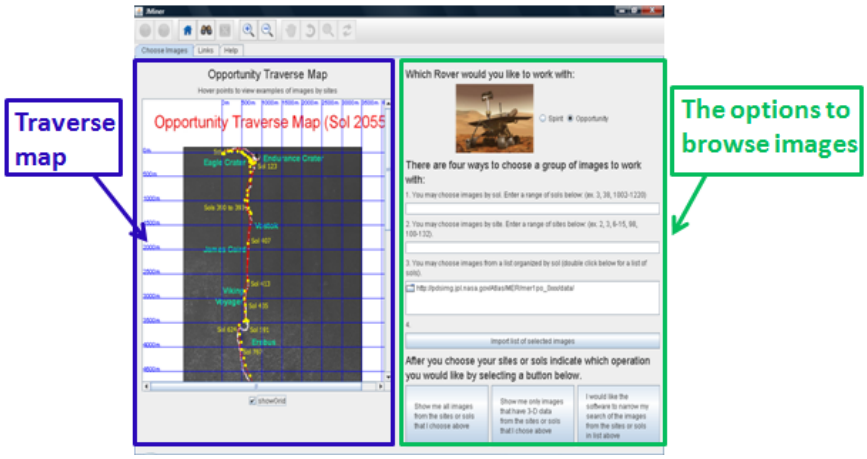


Figure 6.2: The "Choose Images" tab for "ImageRover" software interface. The left part of the screen shows the traverse map for the Opportunity rover with a spatial grid superimposed. The right part of the screen shows three different options for selecting or browsing images.

user to generate a data query (outlined in green).

1. The traverse map

The traverse map is a satellite image of the surface of Mars with the path of the robot on the Martian surface superimposed as a curve, represented as a sequence of connected points within the image. Each point on the traverse map represents a different site. There are two interactions possible with the traverse map:

1. The user can move the mouse cursor over site locations (indicated by yellow dots). When this occurs, a representative image from that site will be displayed as a thumbnail image.
2. The user can toggle the visibility of a grid which is superimposed on the image and indicates approximate distance on the image.

Figure 6.2 shows the traverse map of opportunity with an overlaid grid. Figure 6.3 shows the traverse map of Spirit with both a thumbnail image showing and an overlaid grid.

2. The options for browsing images

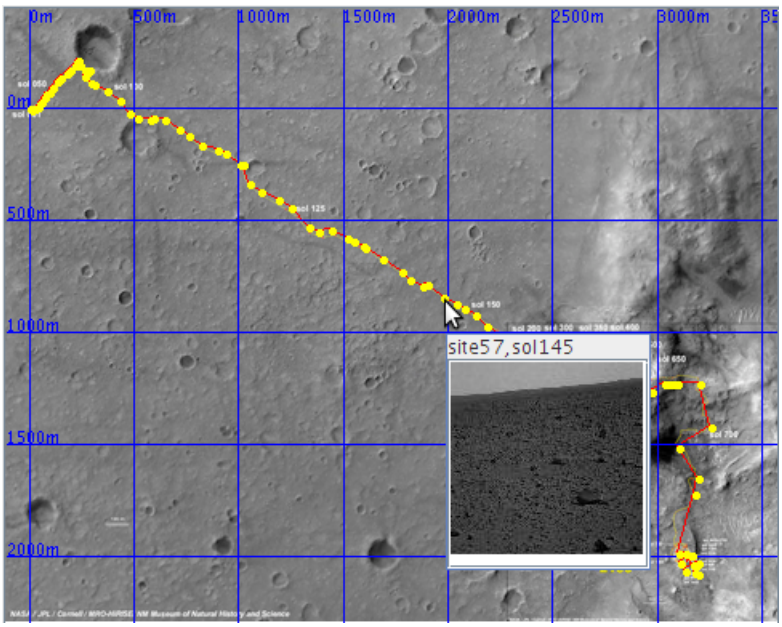


Figure 6.3: The traverse map for the Spirit Rover is shown. A grid is superimposed that indicates intervals of distance in the image. An image site 57 is being displayed in response to the position of the mouse cursor which is hovering over site 57.

This part of the screen allows users to initiate searches. The top-most search criterion in the screen requires users to select the rover dataset which must be either Spirit or Opportunity and is specified by selecting the appropriate check-box item. Below this region, there are four input fields, each of which provides a different method for initializing a search. These methods are:

1. Select images by sol: Users input a list of sols by typing in a series of sol numbers separated by commas (e.g., 1, 3, 5, 18, 200) or intervals of numbers (e.g., 1-30).
2. Select images by site: Users input a list of sites by typing in a series of site numbers separated by commas or intervals of numbers.
3. Select images by file name: Instead of searching by sol or site number, the users can browse through the files that are listed by a NASA-sponsored web server. The web server data listed is organized into folders by sol, EDR and RDR.
4. Import a pre-existing list of images: Users can recall images found by previous "ImageRover" searches by importing a list of filenames (see §6.2.2 on Export functions

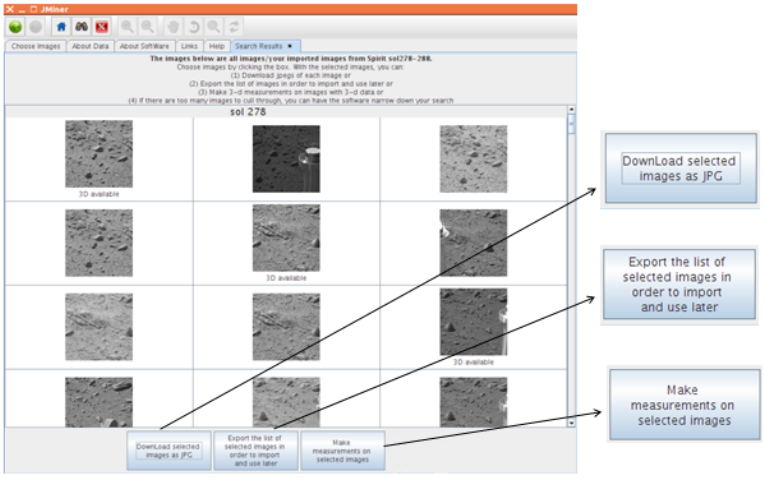


Figure 6.4: The "Search Results" tab for "ImageRover" software shows a table of image thumbnails for each image that satisfies the search criteria. Buttons on the bottom of the screen are enlarged on the right to be legible.

for details).

Users employ one of these four methods to indicate the images they wish to retrieve.

There are three operations that can be invoked over the resulting group of images. They are indicated as buttons at the bottom of the window. The functions for these three buttons are as follows (from left to right):

1. Show all the images from the selected sols/sites,
2. Show only those images from the selected sols/sites that contain 3D XYZ data,
3. Narrow the search of images from the selected sols/sites.

Pressing button (1) or (2) listed above will create a "Search Results" tab that will display a collection of thumbnail (small) images, one for each EFF image that satisfies the search criteria. Pressing button (3) above will create a "Narrow Search" tab that will display a thumbnail image for each EFF image for the selected sites/sols. The "Narrow Search" tab allows the user to iteratively improve the search results based on the image content of the selected images (see §6.2.3 for details).

6.2.2 "Search Results" Tab

The "Search Results" tab displays a table of small "thumbnail" images that satisfy the

search criteria. Each image is a reduced resolution image of a full-frame EFF image. Each table cell may have a "3D available" annotation added underneath the thumbnail image. This annotation exists only for those EFF images that also have associated 3D XYZ RDR data. Users can move the mouse over each thumbnail image. If they do this, the NASA-provided image filename will appear as a "tooltip." Users can select an image by clicking on it or can select multiple images by clicking images and holding down the "Ctrl" key at the same time. Users can operate on the selected set of images by clicking on one of the three buttons at the bottom of the window. The functions for these three buttons (from left to right) are as follows:

1. Download the full resolution image for each of the selected images in JPEG format.
2. Save the current search results so they may be imported later (see §6.2.1 for details).
3. Make measurements on the objects within the selected images (see §6.2.4 for details). This option is only available for images that include the "3D available" label. If selected, a "3D Analysis" tab (see §6.2.4 for details) will appear.

When button (1) from the list above is pressed, a window will prompt the users to select a destination folder. When this information is provided, the software will download the full resolution image for each of the selected thumbnails and save these images to the destination folder in JPEG format. When button (2) from the list above is pressed, a window will prompt the user to indicate a destination filename. When this information is provided, the search results will be saved in the indicated destination file. This file contains the filenames of all the currently selected images. These search results may be re-loaded from this file using the "import list of selected images" button in the "Choose Images" tab. When button (3) from the list above is pressed, a "3D Analysis" tab will be displayed. The interface here enables users to make geometric measurements in the selected 2D EFF and the 3D XYZ images (see §6.2.4 for details).

6.2.3 "Narrow Search" Tab

The goal of "Narrow Search" tab is to allow users to perform image content queries, also known as "query-by-image". The image content based search function allows users to specify a query as a combination of two types of images: (1) desirable images, i.e., images that contain information of interest, and (2) undesirable images, i.e., images that do not contain information of interest.

The search process starts with a search based on sol or site. This search provides an initial group of exemplar images that are shown in an "Exemplar" table. Exemplar images for each sol are determined manually beforehand by selecting those images that best summarize the collected data within each sol. The user then selects images from the exemplar set that are desirable or undesirable. The list of desirable and undesirable images are integrated into a new image-content search criteria which allows users to iteratively improve the search results. At each step in the process, the user can select images to modify the criteria to encourage more "desirable" images to be found and to suppress "undesirable" images. Users can select "desirable" images by holding down the "Ctrl" key and clicking the left button of the mouse. The images that have been specified as "desirable" are shown with a green background as shown in figure 6.5. Users can also select "undesirable" images by holding down the "Ctrl" key and clicking the right button of the mouse. The images that have been specified as "undesirable" are shown with a red background as shown in figure 6.5.

At the bottom of the tab, there is a slider bar and four buttons. The number of images shown in the tab at any time is controlled by this slider. The functions performed by these four buttons are as follows:

1. Search for desirable features in all images that satisfy the initially specified sol or site search criteria.
2. Save the current search criterion.
3. Load a previously saved search criterion.

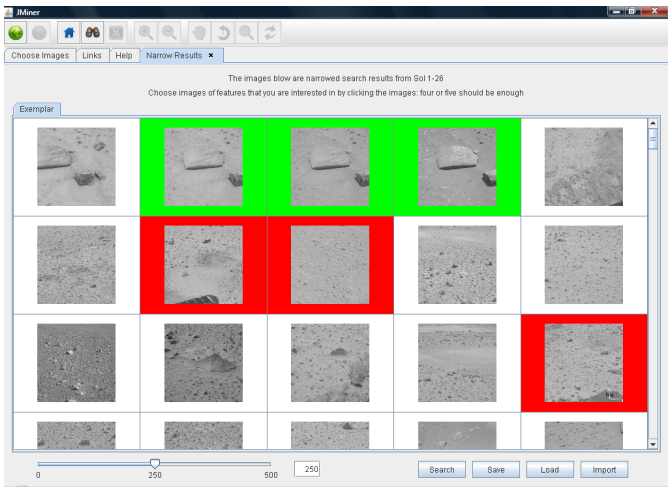


Figure 6.5: The "Narrow Search" tab for "ImageRover" software. The window contains a grid of thumbnail images. Users select these images to indicate images containing data of interest (shown with green background), and also images containing data they wish to exclude (shown with red background).

When button (1) from the list above is pressed, a search will be initiated over images that are included in the initial list of sites or sols provided. The search results will appear in a new tab. When button (2) from the list above is pressed, the current search criteria, as represented by a list of "desirable" and "undesirable" images, will be saved to disk in a user-specified file. When button (3) from the list above is pressed, the user is asked to specify a file containing a previously saved image-content search criteria (a file produced using button (2)). After specifying the file, the search criteria will be loaded and a set of results will be displayed for the loaded search criteria.

6.2.4 "3D Analysis" Tab

The "3D Analysis" tab provides tools to collect geometric measurements from surfaces shown in 2D and 3D Martian images. Collectively, these interface components enable geologists to work directly on the image data to collect geometric measurement data which would otherwise be inaccessible to them. Figure 6.6 shows the "3D Analysis" tab interface which consists of eight parts:

1. A 2D EFF image, shown in the region outlined in red,

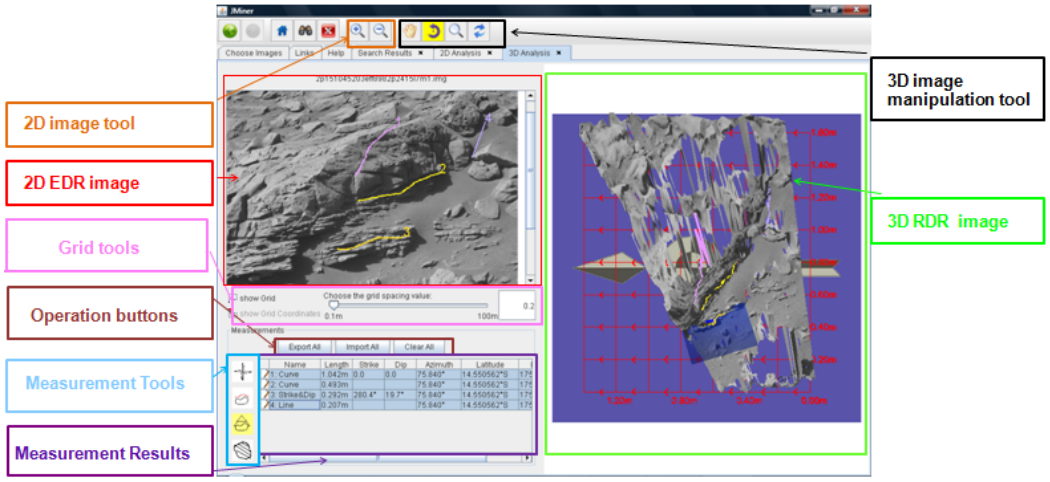


Figure 6.6: The "3D Analysis" tab for "ImageRover" software. The left image is a single 2D image from a stereoscopic pair taken of the Martian surface (2p151045203eff8982p2415l7m1.img), the right image shows a view of the corresponding surface described by the XYZ RDR file reconstructed surface. Curves (labeled as #1-#4) have been manually drawn onto the image by the user to denote the paths of cracks on a Martian rock of interest. These curves are plotted as a space-curve of the same color on the 3D surface image in the right window.

2. A 2D image zoom tool, shown in the region outlined in orange,
3. A 3D visualization window for manipulating XYZ RDR images, shown in the region outlined in green,
4. A collection of 3D image manipulation tools, shown in the region outlined in brown,
5. A collection of "Grid tools", shown in the region outlined in pink,
6. A collection of measurement tools, shown in the region outlined in blue.
7. A table of measurement data, shown in the region outlined in purple.
8. A collection of import and export functions, shown in the region outlined in brown.

These components collectively allow users to visualize, import and collect measurements from measured data in a virtual environment. The details describing how these components work are provided in the following sections.

1. The 2D EFF image

This component within the screen shows a 2D EFF image that the user has selected from the "Search Results" tab (see §6.2.2 for details). This window is used in conjunction with the measurement tools component to create measurements. Within this component, the users can draw lines and curves on the 2D EFF image to create new measurements that appear in the EFF image as a line or curve, in the 3D visualization window as 3D space-curve and in the table of measurements (see measurement tools for details).

2. The 2D image zoom tool

This component within the screen includes two buttons. The first button enables users to zoom in, i.e., enlarge, the 2D EFF image in the window. When pressed, the image within this window will double in resolution. The second button enables users to zoom out, i.e., reduce, the size of the 2D EFF image in the window. When pressed, the image within this window will be reduced by half.

3. 3D visualization window

This component within the screen shows the 3D XYZ RDR image that corresponds to the 2D EFF image displayed on the left. This component is used in conjunction with the 3D manipulation tools to provide a virtual 3D space to visualize, inspect and manipulate the 3D surface data. The blue plane shown underneath the 3D Martian surface approximates the local ground-plane. The arrow in the blue plane indicates the orientation of "North" on the Martian surface. This helps orient the user within the virtual space which can occur when user is allowed unconstrained manipulation of the 3D surface. The grid in the blue plane indicates intervals of distance on the ground-plane.

4. 3D image manipulation tools

This component within the screen includes four buttons. The first button enables users to move or translate 3D surfaces shown in the 3D surface visualization window. The second button enables users to rotate 3D surfaces shown in the 3D surface visualization window. The third button enables users to zoom in/out on the 3D surfaces shown in 3D surface visualization window. The fourth button enables users to restore the default positions of the

3D surfaces in 3D surface visualization window, the default position of surface is shown in figure 6.6.

5. “Grid tools”

This component within the screen includes two check boxes and one slider. When the check box labeled “Show Grid” is selected, a grid will be superimposed over the EFF image. The grid is formed by projecting the lines of the grid shown in 3D onto the 2D EFF image. The underlying algorithm that computes this projection is discussed in §6.3.3. When the check box labeled “Show Grid Coordinates” is selected, a list of coordinates will be displayed on the 2D EFF image at locations where the grid lines intersect. The slider bar controls the spatial interval between the grid lines. Users can click and drag the slider to change this value. When the spatial interval between the grid lines is changed, the grid lines shown in the 3D surface visualization interface and in the 2D EFF image will be updated.

6. Measurement tools

This component within the screen provides measurement tools that enable users to take geometric measurements from the EFF and XYZ image data. The interface allows users to extract several different types of measurements emphasizing those most commonly made in the field by geologists, i.e., the size, orientation and the distance between geological features. The four buttons for measurement tools are as follows:

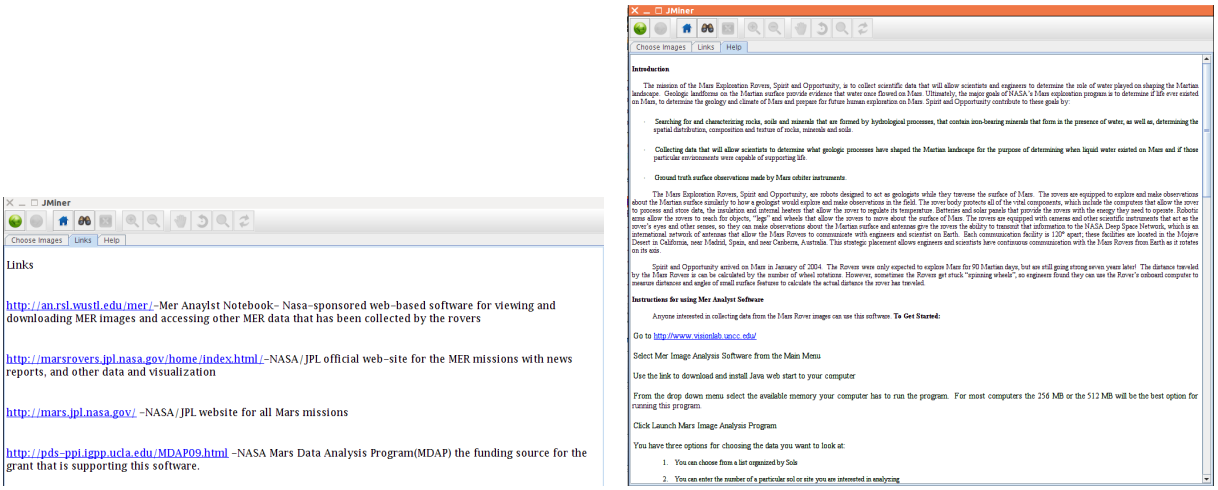
1. Measuring the orientation and length of the rock fractures,
2. Measuring the length of the rock fractures,
3. Measuring the distance between geological features,
4. Measuring the rock surface area.

When button (1) from the list above is pressed, the user may extract the geometry of a rock fracture by tracing the path of a rock fracture in the 2D EFF image with a mouse (one example is shown as the curve #2 in figure 6.6). The corresponding 3D space curve is also shown as a space curve in the 3D XYZ data and visualized in the 3D surface visualization window. At the same time, a plane is automatically fit to the 3D space curve points as shown

in figure 6.6. This plane serves to coarsely approximate the measured surface. The length of the traced curve in meters is shown in the measurement window. The orientation of the measured surface containing the curve is also shown in the measurement window as a pair of angle referred to as the strike and dip. If the user thinks that the computation of the strike and dip is not accurate, they can modify the orientation of the measured plane to obtain more accurate values for the strike and dip. As the user changes the plane interactively, the displayed strike and dip values are updated. The algorithm that manages the 2D and 3D curve data between 2D EFF and 3D XYZ images is discussed in §6.3.1. The algorithm that automatically computes the orientation of the traced curves is discussed in §6.3.2. When button (2) from the list above is pressed, the user may extract the geometry of the rock fracture by tracing the path of a rock fracture in the 2D EFF image with a mouse (one example is shown as the curve #1 in figure 6.6), the corresponding 3D space curve is also shown on the 3D XYZ data displayed in the 3D surface visualization window. The length of the traced curve in meter is shown in the measurement window. When button (3) from the list above is pressed, the user may specify a chord (line segment) on the 2D EFF image with the mouse (one example is shown as the line #3 in figure 6.6). Using the associated 3D XYZ image, the chord is drawn as a 3D line segment in the 3D surface visualization window. The distance between the start and end points of the chord in meters is shown in the measurement window. When button (4) from the list above is pressed, the user can specify a region on the 2D EFF image with the mouse, the rock within that region will be segmented from the ground surface using the ridge walking algorithm, and surface area of the rock in that region will be reported to the user. The underlying algorithm for segmenting the rock from the 3D surface and measuring the rock surface area is discussed in §6.3.4.

7. Measurements table

This component within the screen shows the numeric values of the measurements created by the user using the measurement tools. The values displayed in the table are the length of the curves, the strike and dip of measured geological features (see figure 6.8), the



(a) (b)

Figure 6.7: The "Links" (a) and "Help" (b) tabs for "ImageRover" software

azimuth, the latitude and the longitude of the measurement. Each line of the table corresponds to one measurement. The user can select lines in the table, the corresponding lines or curves that users draw in 2D EFF image will appear in the 2D EFF image and in the 3D XYZ data visualized in the 3D surface visualization window (as shown in figure 6.6).

8. Import and export functions

This component within the screen includes three buttons. The first button enables users to export the recorded measurements to a Comma-Separated Values (CSV) destination file. The saved file contains the 2D EFF image file name, the curves and lines drawn in the image, and the recorded measurements shown in the measurements table. This file may be imported to many popular spreadsheet software packages for further analysis. The second button enables users to import the saved measurement results. The third button enables users to clear the measurement results in the table.

6.2.5 Documentation Provided by the "Link" Tab and the "Help" Tab

The software includes documentation to help users better understand the MER data and the software. The "Link" tab provides a list of useful links that assist users in finding useful information for understanding the data and software, as shown in figure 6.7(a). The list of

related links includes hyper-links to similar software systems that can allow users to view and download MER images such as MER Analyst Notebook. The "Link" tab also provides some links to general information on NASA Mars missions and to related websites, such as the NASA/JPL website. The "Help" tab provides a help document containing directions on how to use the "ImageRover" software, as shown in figure 6.7(b).

6.3 Underlying Algorithms

Implementation of the "ImageRover" software system requires application-specific solutions to provide users the computational tools to necessary measure geological features of the Martian surface. The tools available for analysis rely upon several underlying algorithms which process the image data to extract the information of interest. A list of these algorithms is provided below:

1. An algorithm to match (x,y) pixels in 2D EFF images with (x,y,z) surface data in 3D XYZ images (see §6.3.1 for detail),
2. An algorithm to estimate the orientation of 3D space curves drawn in cracks of rocks (see §6.3.2 for detail),
3. An algorithm to project a grid of 3D lines onto the pixels shown in a 2D EFF image (see §6.3.3 for details),
4. An algorithm to segment rocks from 3D XYZ surface data and measure their areas (see §6.3.4 for details).

All the algorithms from the list above are controlled via the "3D Analysis" tab in the system interface as shown in figure 6.6. Algorithms (1), (2) and (4) are used by the "measurement tools" functions of the tab. A algorithm (3) above is used by the "grid tools" functions of part in the tab.

6.3.1 Matching 2D EFF Images with 3D XYZ Surface Data

To enable users to interact and make measurements from both the 2D EFF image data and the 3D XYZ surface data, the data within these images must be matched.

Each 3D XYZ RDR file is generated by 3D reconstruction surface from the MIPL

(Multimission Image Processing Laboratory). The underlying concept of this software is to use images taken from a pair of cameras to reconstruct the 3D positions of surface viewed by both cameras. This is possible by knowing or estimating the image formation properties of each camera, their relative pose, and the pixel pair in each digital image that corresponds to a specific 3D surface location. In such cases, one may invert the image formation process and find the 3D locations responsible for reflecting the light sensed by the camera. During the reconstruction process pixels in two EFF images are matched to produce an estimate for a single (X,Y,Z) surface position. This process creates a natural correspondence between each (X,Y,Z) measurement in the XYZ RDR file and the pixel data of the corresponding two EFF files.

A 2D EFF image is created by mapping appearance of the Martian surface to it, any point in 3D XYZ surface can be mapped to one pixel in 2D EFF image. Let A denote the set that consists of the 2D (x,y) positions for each of the image pixel in 2D EFF image. Let B denote the set containing the estimated 3D (X,Y,Z) surface positions in XYZ image. The structure of the XYZ image file specifies a correspondence between point from set B and a pixel from the set A. In general, there are some pixels from the pair of EFF images that do not overlap. For these pixels, it is not possible to estimate a 3D (X,Y,Z) position. Hence there are always less (X,Y,Z) measurements than (x,y) pixels. For this reason, one must represent the correspondence mathematically as a mapping of the set B onto the set A as expressed in equation (6.1).

$$g : B \to A \;\; (x,y) \in A, (X,Y,Z) \in B \tag{6.1}$$

Using this mapping, (x,y) positions in the 2D EFF image can also be mapped to points in the 3D XYZ surface.

6.3.2 Estimating the Orientation of 3D Space Curves in Cracks of Rocks

In field geology, it is common to measure the strike and dip, i.e., the orientation of cracks within rock. In geology, the dip is the angle of tilt of the crack surface away from the horizontal. The line of strike is the line created by intersecting a plane tilted by the dip

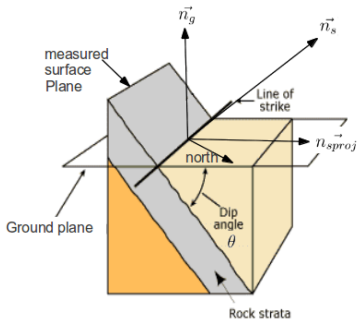


Figure 6.8: The computation for strike and dip, the gray plane is the plane of measured surface.

angle with a horizontal plane. The strike is the angle between the line of strike and a vector pointing towards North. In our system, the user can measure the orientation of a crack by tracing a curve in the image over the crack (as shown by curve #2 in figure 6.6). The corresponding 3D space curve is displayed on the 3D surface data using the correspondence information described in §6.3.1. A plane is then fit to the 3D space-curve using a least squares fitting method. This fitting plane is taken as an estimate for the orientation of the crack surface and is called measured surface plane.

To specify how to compute the measured surface plane, we will need to define some mathematical notation. Let $\mathbf{p} = [\mathbf{p}_1, \mathbf{p}_2, \cdots \mathbf{p}_N]$ denote the points of the 3D space curve. The plane coefficients are estimated by forming the matrix $\mathbf{M}$ (as shown in equation 6.2) and taking the 4×1 eigenvector with the smallest non-negative associated eigenvalue of $\mathbf{M}^t\mathbf{M}$.

$$\mathbf{M} = \begin{bmatrix} \mathbf{p}_1 & 1 \\ \mathbf{p}_2 & 1 \\ \vdots & \vdots \\ \mathbf{p}_N & 1 \end{bmatrix} \tag{6.2}$$

To specify how to compute the strike and dip, we will need to define some mathematical notation. Let $\mathbf{n}_g$ denote the normal of ground plane and let $\mathbf{n}_s$ denote the normal of the measured surface plane. The dip angle measures how much the measured surface plane tilts away from the ground plane, it is the angle between the ground plane and the measured

plane as shown in figure 6.8. The dip angle, θ, can be computed in equation (6.3).

$$\theta = \arccos(\mathbf{n}_g \cdot \mathbf{n}_s), \theta \in (0, \frac{\pi}{2}) \tag{6.3}$$

The strike line denotes the intersection line between the measured surface plane and the ground plane, the strike line is perpendicular to the normal of both the ground plane and the measured surface plane. The direction of the strike line can be computed using equations (6.4) and (6.5).

$$\mathbf{n}_{sproj} = \frac{\mathbf{n}_s - (\mathbf{n}_g \cdot \mathbf{n}_s)\mathbf{n}_g}{\left\|\mathbf{n}_s - (\mathbf{n}_g \cdot \mathbf{n}_s)\mathbf{n}_g\right\|} \tag{6.4}$$

$$\mathbf{n}_{strike} = \mathbf{n}_s \times \mathbf{n}_{sproj} \tag{6.5}$$

In equation (6.4), the $\mathbf{n}_{sproj}$ is the projection of the normal of measured surface plane, $\mathbf{n}_s$, onto the ground plane. The vector $\mathbf{n}_{sproj}$ is perpendicular to both strike line and the normal of ground plane, so the direction of strike line, $\mathbf{n}_{strike}$, is the cross product of $\mathbf{n}_s$ and $\mathbf{n}_{sproj}$. The strike angle, φ, can also be computed by rotating the vector pointing in the direction of North around the ground plane normal in the clockwise direction until it aligns with the direction of the strike line.

6.3.3 Projecting Grid of 3D Lines to 2D EFF Image

The grid superimposed over the 2D EFF image as described in §6.2.4 is formed by projecting the coordinates of the grid underneath 3D XYZ surface onto the 2D EFF image. This is accomplished using a homography transform. A homography is an invertible transformation that can map straight lines between a plane and its projective plane. Here, the plane is the 3D plane containing the grid of lines underneath the 3D XYZ surface and the projective plane is the ground plane as it appears in the 2D EFF image. Let $\mathbf{q}_i$ denote the position of a point in the 3D plane of the grid, where $\mathbf{q}_i = [X, Y, Z]^t$. Let $\mathbf{p}_i$ denote the position of a 2D projected point in the EFF image, where $\mathbf{p}_i = [x, y, 1]^t$. Using the homography, the intersection points of the grid can be mapped to the 2D EFF image using $\mathbf{p}_i = \mathbf{H}\mathbf{q}_i$, where $\mathbf{q}_i$ is a 3×1 matrix, H is the 3×3 homography matrix, and $\mathbf{p}_i$ is a 3×1 matrix. The intersection points form the grid that is superimposed over 2D EFF image.

The homography matrix, $\mathbf{H}$, can be computed by mapping four known points in the 3D

grid plane to four corresponding points in the 2D EFF image. Let $\mathbf{Y} = [\mathbf{q}_1^t, \mathbf{q}_2^t, \mathbf{q}_3^t, \mathbf{q}_4^t]$ denote a matrix which contains the four corner points of the 3D grid plane, each corner point is a separate row of the matrix making $\mathbf{Y}$ a 4×3 matrix. Each corner point, $\mathbf{q}_i$, is known to correspond to a 3D surface point, $\mathbf{m}_i$, that is obtained by computing the intersection point between the normal of the grid plane at $\mathbf{q}_i$ with the 3D XYZ surface. Let $\mathbf{M} = [\mathbf{m}_1^t, \mathbf{m}_2^t, \mathbf{m}_3^t, \mathbf{m}_4^t]$ denote the positions of the 3D surface data at the grid locations $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ and $\mathbf{q}_4$. As described in §6.3.1, each point in the 3D XYZ surface, $\mathbf{m}_i$, can be mapped to exactly one image pixel location, $\mathbf{p}_i$, in the corresponding 2D EFF image using equation (6.1). Let $\mathbf{X} = [\mathbf{p}_1^t, \mathbf{p}_2^t, \mathbf{p}_3^t, \mathbf{p}_4^t]$ denote a matrix which contains the four points in the 2D EFF image that correspond to $\mathbf{m}_1^t, \mathbf{m}_2^t, \mathbf{m}_3^t$ and $\mathbf{m}_4^t$ where each point is a row of the matrix $\mathbf{X}$ (a 4×3 matrix). Then the homography matrix can be computed as shown in equation (6.6).

$$\mathbf{H} = \mathbf{X}\mathbf{Y}^{-1} \tag{6.6}$$

6.3.4 Segmenting Rocks from 3D XYZ Surface and Measuring their Areas

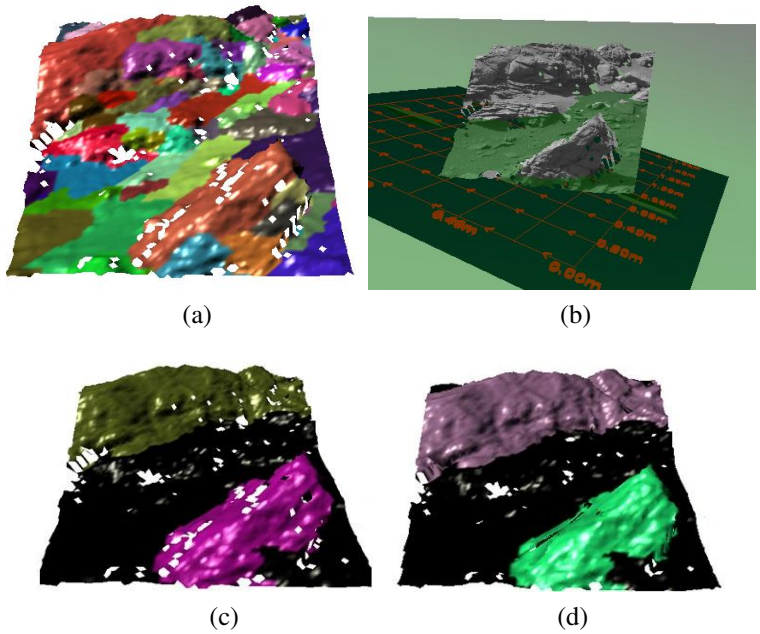
The system allows users to segment rock data from 3D XYZ image. This operation allows the user to extract and analyze the rock surface data. The ridge walking algorithm is applied here to segment rocks from the 3D surface data. However, the 3D surface is noisy with many holes as a stereoscopic reconstruction. The ridge walking algorithm has to be adjusted in order to segment this kind of surface. The following two main steps are involved in segmentation of rocks from the 3D surface and measuring their surface area.

1. Segment the rocks from the 3D surface,
2. Compute the surface area for a segmented rock.

Step (1) uses the ridge walking algorithm to segment the 3D surface. Step (2) fills the holes on the segmented rock surface, then the surface area of the segmented rock is computed. The details of each step are discussed in the following sections.

1. Segment the rocks from 3D surface

The adjusted ridge walking algorithm is applied here to segment the rocks from 3D surface. The challenges here are: (1) the ridge walking algorithm has to handle the holes



(a) (b)

(c) (d)

Figure 6.9: The process of compute the rock area of the Martian surface. (a) shows the segmented results using ridge walking algorithm. (b) shows the green plane are fitted to the Martian surface using the least square fitting method. (c) shows the merged segmentation result. (d) shows the Martian surface after filling the hole on the segmented rock surface.

on the surface, (2) the ridge walking algorithm has to overcome the noise on the surface. A graphical outline of the process is shown in figure 6.9. The adjusted ridge walking algorithm is accomplished using a three steps process discussed as follows:

1. Compute the weight for each edge of the surface. Since the rocks are delimited by concave valleys of the 3D surface, the equation (2.2) as described in §2.2 is used here to compute the edge weights of the surface. As indicated in equation (2.2), the edge weights depend on the minimum principal curvature and the direction of maximum curvature at each point on the surface. On the hole boundaries, these values cannot be computed because the surface at these locations is not continuous or manifold. For this reason, the hole of the 3D surface are taken to be elements for the segmented boundaries. This is accomplished by setting the weights of edges that lie on a hole boundary to an excessively large value. To do that, the weights for edges not in the hole are computed and the largest weight is picked as w_{max}. The weights for all the

edges that lie along hole boundaries are set to the value w_{max}.

2. The ridge walking algorithm described in §2.2 is executed to segment the rocks from surface. The input parameter λ of the ridge walking algorithm is computed automatically using equation (6.7). A segmentation result using this setting is shown in figure 6.9(a). In this approach, the rocks are over-segmented due to noise on the surface, fractures on the rocks, and high value of λ.

$$\lambda = \frac{Number\,of\,concave\,loopy\,edges}{Number\,of\,loopy\,edges} \tag{6.7}$$

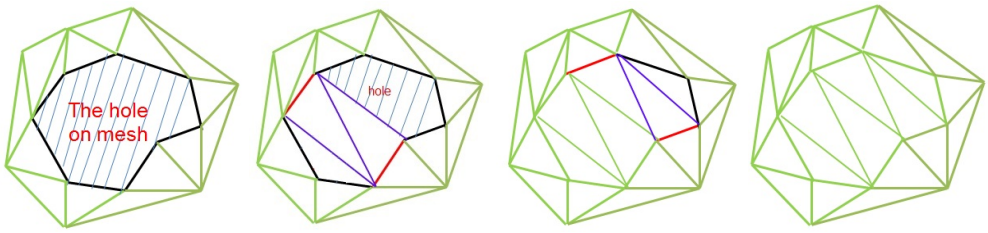
3. A post-process step is used here to merge the over-segmented rock regions. Here, a plane is fitted to the 3D surface using the a least squares fitting method, the fitted plane is shown in figure 6.9(b) in green. The centroid of each segmented patch is computed and a decision is made: if the centroid of a segmented patch is above the fitted plane, the segmented patch is a part of rock surface, otherwise it a part of ground surface. Neighboring segmented patches belonging rock surface are merged together. The merged segmentation result is shown in figure 6.9(c) in different color.

The above 4 steps adjust the ridge walking algorithm by manually setting the weight for hole edges (step (1)), and merging over-segmented rock regions (step (3)).

2. Compute the surface area for a segmented rock

To measure the surface area for a rock, the holes on the rock have to be filled. The steps for filling one of the holes on the segmented rock are discussed in the following:

1. Find all the edges that are lies on hole boundaries, these edges are only owned by one graph face. The hole edges are shown in black in figure 6.10(a).
2. Find a pair of edges lying on hole boundaries that are most parallel to each other, the parallel edges are shown in red in figure 6.10(b). The parallel edges are connected by adding new edges, two triangular faces are created as shown in blue in figure 6.10(b). New holes will be generated after connecting parallel edges.
3. Step (2) is repeated until there are no holes left. This process is shown in figure 6.10(c)-(d).



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

Figure 6.10: The process of filling the hole of a triangular mesh.

After those steps, the segmented rock is a connected triangular mesh surfaces (as shown in figure 6.9(d)). The area of the rock is the area of all triangular faces within the rock. For example, the rock surface area for the green rock in figure 6.9(d) is $0.225m^2$.

6.4 Conclusion

This chapter introduces the Martian view software which has several functions: (1) it provides methods that enable users to find images of interest by time, space and image content, the search results enable geologists to directly find images of interest. (2) it provides methods to visualize the 2D EFF image and corresponding 3D XYZ stereoscopic reconstructed surface at the same time, which is not available for most existing Martian view software; (3) it provide user-interactive analysis tool on 2D and 3D Martian images to make measurement of the geometric features of the Martian surfaces such as crack length, orientation and etc. Measurement results are useful to address geologist hypotheses. The ridge walking algorithm is used in this software to identify surface features such as rocks in the Martian surfaces.

CHAPTER 7: CONCLUSION AND FUTURE WORK

This dissertation discusses new methods for surface analysis to help users better understand these surfaces. It introduces a new geometric 3D mesh surface segmentation algorithm called ridge walking. The ridge walking algorithm computes a collection of closed contours on the surface that serve as boundaries to segment the surface into parts. Controlling the geometric properties of contours allows the ridge walking algorithm to have dynamic segmentation behavior. The dynamic segmentation behavior includes segmenting the surfaces along: (1) regions delineated by convex ridges, (2) regions delineated by concave valleys, and (3) regions delineated by both concave and convex curves. The The dynamic behavior makes the ridge walking algorithm can work on a broad number of categories of surfaces.

The ridge walking segmentation method is quantitatively evaluated by comparing results with ground truth segmentations and other segmentation algorithms using several evaluation metrics. The geometric properties of the segmentation results are also studied to explore how the ridge walking algorithm segments the model. Overall, the evaluation results show that the ridge walking algorithm is more similar to the ground truth than other segmentation algorithms. The ridge walking algorithm also has similar geometric properties for the segmentation results as the ground truth. However, the ridge walking algorithm does not work well for all kind of models.

Then the ridge walking algorithm is applied to three domain-specific segmentation problems. For the problem of bone fragment segmentation, the dissertation extends the ridge walking geometric segmentation by incorporating it with the CT intensity values taken from the appearance of bone fragment tissues. The algorithm for bone fragment

segmentation consists of two steps: (1) a geometric segmentation of the surface via ridge-walking algorithm and (2) an appearance-based classification that uses CT voxel intensities in the vicinity of each surface point to classify each segmented surface patch. Bone fragment surfaces are classified into one of the following classes: (i) periosteal surfaces, defined as surfaces that were part of the original periosteal bone exterior, (ii) fracture surfaces, defined as new surfaces generated when the bone was fractured, and (iii) articular surfaces, defined as surfaces from the bone/cartilage interface at a joint.

For the problem of image segmentation, the 3D ridge walking algorithm is adapted to segment 2D images. The 2D image is modeled as a Monge patch, the principal curvatures and directions of principal curvatures for each image pixel are computed from the Monge patch. These curvatures information is then used by ridge walking algorithm to segment the image along both ridges and valleys of the image. The quantitative evaluation of ridge walking 2D image segmentation is tested on 3 different images to find if its segmentation results on these images are comparable with other competing algorithms.

For the problem of analysis of virtual 3D terrain models, the ridge walking algorithm is integrated as a part of a larger software system designed to enable users to browse, visualize and analyze 3D geometric data generated by the Mars Exploratory Rovers Spirit and Opportunity. The software enables users efficiently browse images of interest from Mars using three methods: (1) time querying, (2) place/location querying, and (3) image content querying. The software also provides the ability to visualize 2D EFF images and their corresponding 3D XYZ images at the same time. For 3D XYZ images, the system provides measurement tools to gather geometrical features of the Martian surfaces. The tools include measuring the orientation, size and surface area of the rocks.

7.1 Future Work

Although the segmentation algorithm developed in this dissertation offers a powerful segmentation tool to segment different types of surfaces into meaningful parts, there are some limitations that need to be taken into account for future research. The most important lim-

itation is that the evaluation of 2D ridge walking image segmentation is only based on 3 different images. We can not draw any general conclusions about the strength and weakness of the ridge walking algorithm. The future work includes a more comprehensive quantitatively evaluation and analysis the 2D ridge walking image segmentation on a large number of images. The quantitative evaluation is calculated by comparing the ridge walking segmentation results with the results generated by human, which is considered as the ground truth. D. Martin and et. al. from Berkeley university has established a human image segmentation database. The 2D ridge walking algorithm will be test on all the images in the dataset, the segmentation results will be compared with the ground truth by computing the difference between segment boundaries and regions. The geometric properties of segment boundaries and regions generated by the ridge walking algorithm will be analyzed to generalize the behavior and better characterize the potential advantages of the 2D ridge walking image segmentation algorithm, i.e, what type of image the ridge walking algorithm is good at segmenting? and what kind of structures in the image the ridge walking algorithm tends to capture?

BIBLIOGRAPHY

[1] L. G. Shapiro and G. C. Stockman, *Computer Vision*. Prentice-Hall, 2001.

[2] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis, "3D Mesh Segmentation Methodologies for CAD applications," *Computer Aided Design & Applications*, vol. 4, no. 6, pp. 827–841, 2007.

[3] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by Example," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 652–663, 2004.

[4] L. Shapira, A. Shamir, and D. Cohen-or, "Consistent Mesh Partitioning and Skeletonisation using the Shape Diameter Function," *Visual Computer*, vol. 24, no. 4, pp. 249–259, 2008.

[5] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine*, vol. 22, no. 8, pp. 888–905, 2000.

[6] D. D. Anderson, V. L. Muehling, J. L. Marsh, and T. D. Brown, "Precise Identification of Bone Fragment Boundaries to Assist in Reduction of Highly Comminuted Fractures," *Computer-Aided Surgery*, vol. 9, no. 3, p. 116, 2004.

[7] A. Willis, W. Shadid, and M. C. Eppes, "Mining Remote-image Repositories with Application to Mars Rover Stereoscopic Image Datasets," in *Proceedings of SPIE on Electronic Imaging*, 2009, pp. 37–45.

[8] M. Harders, A. Barlit, C. Gerber, J. Hodler, and G. Székely, "An Optimized Surgical Planning Environment for Complex Proximal Humerus Fractures," in *Proceedings of Workshop on Interaction in Medical Image Analysis and Visualization*, 2007, pp. 1–8.

[9] R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, 1993.

[10] E. Zhu, J. Yin, C. Hu, and G. Zhang, "A Systematic Method for Fingerprint Ridge Orientation Estimation and Image Segmentation," *Pattern Recognition*, vol. 39, no. 8, pp. 1452–1472, 2006.

[11] M. A. G. Vicki Bruce, Patrick R. Green, *Visual Perception: Physiology, Psychology, and Ecology*. Psychology Press, 2003.

[12] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of Polyhedral Surfaces using Decomposition," in *Proceedings of Computer Graphics Forum*, 2002, pp. 219–228.

[13] A. Shamir, "A Survey on Mesh Segmentation Techniques," *Computer Graphics Forum*, vol. 27, no. 5, pp. 1539–1556, 2008.

[14] M. Elad, A. Tal, and S. Ar, "Content Based Retrieval of VRML Objects - An Iterative and Interactive aAproach," in *Proceedings of IEEE Aerospace Conference*, 2001, pp. 97–108.

[15] D. P. Luebke, "A Developer's Survey of Polygonal Simplification Algorithms," *IEEE Computer Graphics and Applictions*, vol. 21, no. 1, pp. 24–35, 2001.

[16] A. R. Willis, D. D. Anderson, T. P. Thomas, T. D. Brown, and J. L. Marsh, "3D Reconstruction of Highly Fragmented Bone Fractures," in *Proceedings of SPIE Conference on Medical Image*, 2007, pp. 65 121P1–65 121P10.

[17] D. Hoffman, W. Richards, A. Pentl, J. Rubin, and J. Scheuhammer, "Parts Of Recognition," *Cognition*, vol. 18, no. 1, pp. 65–96, 1983.

[18] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel, "Intelligent Mesh Scissoring using 3D Snakes," in *Proceedings of the Computer Graphics and Applications Pacific Conference*, 2004, pp. 279–287.

[19] X. Chen, A. Golovinskiy, and T. Funkhouser, "A Benchmark for 3D Mesh Segmentation," in *Proceedings of Special Interest Group on Graphics and Interactive Techniques*, 2009, pp. 1–12.

[20] D. Giorgi, S. Biasotti, and L. Paraboschi, "Shape Retrieval Contest 2007: Watertight Models Track," in *Proceedings of Shape Retrieval Contest conference*, 2007, pp. 733 – 740.

[21] K. Y, E. K, and K. WA, "A New Accurate and Precise 3-D Segmentation Method for Skeletal Structures in Volumetric CT Data," *IEEE Transactions on Medical Imaging*, vol. 22, no. 5, pp. 586–598, 2003.

[22] J. F. Bell, "Mars Exploration Rover Athena Panoramic Camera (Pancam) investigation," *Journal of Geophysical Research*, vol. 108, no. E12, p. 8063, 2003.

[23] J. F. B. Iii, J. Joseph, J. N. Sohl-dickstein, H. M. Arneson, M. J. Johnson, M. T. Lemmon, and D. Savransky, "In-flight Calibration and Performance of the Mars Exploration Rover Panoramic Camera (Pancam) Instruments," *Journal of Geophysical Research: Planets*, vol. 111, pp. 44–48, 2006.

[24] L. Matthies, M. Maimone, A. Johnson, Y. Cheng, R. Willson, C. Villalpando, S. Goldberg, A. Huertas, A. Stein, and A. Angelova, "Computer Vision on Mars," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 67–92, 2007.

[25] N. H. Sleep, "Martian Plate Tectonics," *Geophysical Research*, vol. 99, pp. 1–4, 1994.

[26] A. Shamir, "Segmentation and Shape Extraction of 3D Boundary Meshes," in *Proceedings of Eurographics*, 2006, pp. 137–149.

[27] K. DongHwan, Y. TiDong, and L. SangUK, "Triangular Mesh Segmentation Based on Surface Normal," in *Proceedings of Asian Conference on Computer Vision*, 2002, pp. 23–25.

[28] M. Vieira and K. Shimada, "Surface Mesh Segmentation and Smooth Surface Extraction through Region Growing," *Computer Aided Geometric Design*, vol. 22, no. 8, pp. 771–792, 2005.

[29] Y. Zhang, J. Paik, A. Koschan, and M. A. Abidi, "A Simple and Efficient Algorithm for Part Decomposition of 3D Triangulated Models Based on Curvature Analysis," in *Proceedings of the International Conference on Image Processing*, 2002, pp. 273–276.

[30] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, "Rapid and Effective Segmentation of 3D Models using Random Walks," *Computer Aided Geometric Design*, vol. 26, no. 6, pp. 665–679, 2009.

[31] S. Beucher and C. Lantuejoul, "Use of Watersheds in Contour Detection," in *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation*, 1979, pp. 1–12.

[32] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 583–598, June 1991.

[33] J. M. Gauch, "Image Segmentation and Analysis via Multiscale Gradient Watershed Hierarchies," *Computer Graphics Forum*, vol. 8, no. 1, pp. 69–79, 1999.

[34] J. Serra, *Image Analysis and Mathematical Morphology*. Orlando, FL, USA: Academic Press, Inc., 1983.

[35] F. Meyer, "Topographic Distance and Watershed Lines," *Signal Processing*, vol. 38, no. 1, pp. 113–125, July 1994.

[36] A. P. Mangan and R. T. Whitaker, "Partitioning 3D Surface Meshes using Watershed Segmentation," *IEEE Transaction on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.

[37] E. Zuckerberger, A. Tal, and S. Shlafman, "Polyhedral Surface Decomposition with Applications," *Computers & Graphics*, vol. 26, no. 5, pp. 733–743, 2002.

[38] L. Moumoun, M. Chahhou, T. Gadi, and R. Benslimane, "Solving the 3D Watershed Over-Segmentation Problem using the Generic Adjacency Graph," in *Proceeding of I/V Communications and Mobile Network*, 2010, pp. 1–4.

[39] D. L. Page, A. F. Koschan, and M. A. Abidi, "Perception-based 3D Triangle Mesh Segmentation using Fast Marching Watersheds," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2003, pp. 27–32.

[40] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical Mesh Segmentation Based on Fitting Primitives," *Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.

[41] M. Garland, A. Willmott, and P. S. Heckbert, "Hierarchical Face Clustering on Polygonal Surfaces," in *Proceedings of the Symposium on Interactive 3D Graphics*, 2001, pp. 49–58.

[42] A. Golovinskiy and T. Funkhouser, "Randomized Cuts for 3D Mesh Analysis," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 1–12, 2008.

[43] S. Katz and A. Tal, "Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts," *ACM Transactions on Graphics*, vol. 22, pp. 954–961, July 2003.

[44] H.-Y. S. Lin, H.-Y. M. Liao, and J.-C. Lin, "Visual Salience-guided Mesh Decomposition," in *Proceedings of IEEE Workshop on Multimedia Signal*, 2004, pp. 331–334.

[45] S. Katz, G. Leifman, and A. Tal, "Mesh Segmentation using Feature Point and Core Extraction," *Visual Computer*, vol. 21, no. 8-10, pp. 649–658, 2005.

[46] Y. Zhou and Z. Huang, "Decomposing Polygon Meshes by Means of Critical Points," in *Proceedings of the International Multimedia Modelling Conference*, 2004, pp. 187–195.

[47] C. Xu and J. Prince, "Snakes, Shapes, and Gradient Vector Flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, Mar 1998.

[48] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel, "Mesh Scissoring with Minima Rule and Part Salience," *Computer Aided Geometric Design*, vol. 22, no. 5, pp. 444–465, 2005.

[49] J. Zhang, C. Wu, J. Cai, J. Zheng, and X. Tai, "Mesh Snapping: Robust Interactive Mesh Cutting using Fast Geodesic Curvature Flow," in *Proceedings of European Association for Computer Graphics*, 2010, pp. 517–526.

[50] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. J. Guibas, "Example-based 3D Scan Completion," in *Proceedings of the Eurographics Symposium on Geometry Processing*, 2005, pp. 23–31.

[51] X. Chen and F. Schmitt, "Intrinsic Surface Properties from Surface Triangulation," in *Proceedings of the European Conference on Computer Vision*, 1992, pp. 739–743.

[52] J. Kruskal, Joseph B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *the American Mathematical Society*, vol. 7, no. 1, pp. pp. 48–50, 1956.

[53] Q. Huang and B. Dom, "Quantitative Methods of Evaluating Image Segmentation," in *Proceedings of the International Conference on Image Processing*, 1995, pp. 3053–3060.

[54] W. Rand, "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[55] K. O. Babalola, T. F. Cootes, C. J. Twining, V. Petrovic, and C. J. Taylor, "3D Brain Segmentation using Active Appearance Models and Local Regressors," *Medical Image Computing and Computer-Assisted Intervention*, vol. 11, pp. 401–408, 2008.

[56] C. J. Rose and C. J. Taylor, "A Generative Statistical Model of Mammographic Appearance," in *Proceedings of Medical Image Understanding and Analysis*, 2004, pp. 89–92.

[57] D. L. Collins and A. C. Evans, "ANIMAL: Validation and Applications of Nonlinear Registration-based Segmentation," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 11, no. 8, pp. 1271–1294, 1997.

[58] D. Metaxas, X. Huang, and T. Chen, "Integrating Shape and Texture in Deformable Models: from Hybrid Methods to Metamorphs," in *Proceedings of International Conference on Control, Automation, Robotics and Vision*, 2006, pp. 1–6.

[59] B. Fischl, D. H. Salat, E. Busa, M. Albert, M. Dieterich, C. Haselgrove, A. Van Der Kouwe, R. Killiany, D. Kennedy, S. Klaveness, and et al., "Whole Brain Segmentation: Automated Labeling of Neuroanatomical Structures in the Human Brain," *Neuron*, vol. 33, no. 3, pp. 341–355, 2002.

[60] M. Turk and A. Pentland, "Face Recognition using Eigenfaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1991, pp. 586–591.

[61] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, 2000.

[62] W. Skarbek and A. Koschan, "Colour Image Segmentation: A Survey," Ph.D. dissertation, Technical University of Berlin, 1994.

[63] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, Dec. 2003.

[64] H. D. Cheng, X. H. Jiang, Y. Sun, and J. L. Wang, "Color Image Segmentation: Advances and Prospects," *Pattern Recognition*, vol. 34, pp. 2259–2281, 2001.

[65] O. Morris, M. J. Lee, and A. G. Constantinides, "Graph Theory for Image Analysis: An Approach Based on the Shortest Spanning Tree," in *Proceedings of Communications, Radar and Signal Processing*, 1986, pp. 146–150.

[66] C. A. Glasbey, "An Analysis of Histogram-based Thresholding Algorithms," *Graphical Models and Image Processing*, vol. 55, pp. 532–537, November 1993.

[67] N. Otsu, "A Threshold Selection Method from Gray-level Histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[68] N. R. Pal and S. K. Pal, "A Review on Image Segmentation Techniques," *Pattern Recognition*, pp. 1277–1294, 1993.

[69] L. Lucchese and S. K. Mitra, "Unsupervised Segmentation of Color Images Based on k -means Clustering in the Chromaticity Plane," in *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1999, pp. 74–81.

[70] D. Comaniciu, P. Meer, and S. Member, "Mean shift: A Robust Approach Toward Feature Space Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[71] R. Adams and L. Bischof, "Seeded Region Growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[72] W. Cui, Z. Guan, and Z. Zhang, "An Improved Region Growing Algorithm for Image Segmentation," in *Proceedings of the International Conference on Computer Science and Software Engineering*, 2008, pp. 93–96.

[73] R. Pohle and K. D. Toennies, "Segmentation of Medical Images using Adaptive Region Growing," in *Proceedings of SPIE on Medical Imaging*, 2001, pp. 98–105.

[74] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-based Image Segmentation," *International Journal of Computer Vision*, vol. 59, p. 2004, 2004.

[75] Z. Wu and R. Leahy, "An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1101–1113, November 1993.

[76] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *International Jounal of Computer Vison*, vol. 1, no. 4, pp. 321–331, 1988.

[77] T. Mcinerney and D. Terzopoulos, "Deformable Models in Medical Image Analysis: A Survey," *Medical Image Analysis*, vol. 1, pp. 91–108, 1996.

[78] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape Modeling with Front Propagation: A Level Set Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 158–175, 1995.

[79] B. Dong, A. Chien, Y. Mao, J. Ye, and S. Osher, "Level Set Based Surface Capturing in 3D Medical Images," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2008, pp. 162–169.

[80] M. Droske, B. Meyer, M. Rumpf, and C. Schaller, "An Adaptive Level Set Method for Medical Image Segmentation," in *Proceedings of the International Conference on Information Processing in Medical Imaging*, 2001, pp. 416–422.

[81] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," in *Proceedings of International Conference on Computer Vision*, 2001, pp. 416–423.

[82] "Nasa Planetary Data System," http://pds.nasa.gov/.

[83] "Bulk Downloader," http://starbeam.jpl.nasa.gov/tools/release/.

[84] "NASAView Image Display Software," http://pds.nasa.gov/tools/nasa-view.shtml.

[85] "Integrated System for Imagers and Spectrometers," http://isis.astrogeology.usgs.gov/.

[86] "MER Analyst's Notebook," http://an.rsl.wustl.edu/mer/.

[87] Frawley, *Knowledge Discovery In Database*. AAAI Press/MIT Press., 1996.

[88] T. Li, Q. Li, S. Zhu, and M. Ogihara, "A Survey on Wavelet Applications in Data Mining," *SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 49–68, Dec. 2002.

[89] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2161–2168.

[90] S. Lazebnik, C. Schmid, and J. Ponce, "A Sparse Texture Representation using Local Affine Regions," *IEEE Transations on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1265–1278, Aug. 2005.

[91] E. Nowak, F. Jurie, and B. Triggs, "Sampling Strategies for Bag-of-features Image Classification," in *Proceedings of European Conference on Computer Vision*. Springer, 2006, pp. 490–503.

[92] D. R. Thompson, T. Smith, and D. Wettergreen, "Data Mining During Rover Traverse: From Images to Geologic Signatures," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, October 2005.

[93] R. C. Castano, A.and Anderson, R. Castano, T. Estlin, and M. Judd, "Intensity-based Rock Detection for Acquiring Onboard Rover Sciences," in *Proceedings of Lunar and Planetary Science Conferencen*, 2004, pp. 1–8.

[94] D. Thompson, "Performance Comparison of Rock Detection Algorithms for Autonomous Planetary Geology," in *Proceedings of IEEE Aerospace Conference*, 2007, pp. 1–9.